Universidad de Jaén

Escuela de Doctorado

TESIS DOCTORAL

•

REDES DE ARQUITECTURA PROFUNDA Y ENSEMBLES PARA EL TRATAMIENTO DE LA ALTA DIMENSIONALIDAD Y EL DESBALANCEO EN APRENDIZAJE SUPERVISADO

PRESENTADA POR:
FRANCISCO JAVIER PULGAR RUBIO

DIRIGIDA POR:
FRANCISCO CHARTE OJEDA
MARÍA JOSÉ DEL JESUS DÍAZ

JAÉN, 10 DE OCTUBRE DE 2019

ISBN

UNIVERSIDAD DE JAÉN

TESIS DOCTORAL

Redes de arquitectura profunda y ensembles para el tratamiento de la alta dimensionalidad y el desbalanceo en aprendizaje supervisado

---

Autor:
Francisco Javier Pulgar Rubio

Directores de tesis:
Dr. D. Francisco Charte Ojeda
Dra. Dª María José del Jesus Díaz

Programa Oficial de Doctorado en
Tecnologías de la Información y la Comunicación

Tesis sometida para optar al grado de Doctor

por la

Universidad de Jaén

Jaén, octubre de 2019

Tesis doctoral parcialmente financiada por el programa de Ayudas para la Formación de Personal Investigador (Acción 15) de la Universidad de Jaén según la convocatoria publicada en BOJA de 20 de octubre de 2014, por el programa predoctoral de Formación del Profesorado Universitario (FPU) del Ministerio de Educación, Cultura y Deporte (Ref. FPU16/00324) según convocatoria publicada en BOE de 17 de enero de 2017 y parcialmente subvencionada por el Ministerio de Ciencia y Tecnología de España bajo el proyecto TIN2015-68454-R.

El doctorando, Francisco Javier Pulgar Rubio, y los directores de la tesis Dr. D. Francisco Charte Ojeda y Dra. Dª María José del Jesus Díaz declaramos, al firmar esta tesis titulada, "Redes de arquitectura profunda y ensembles para el tratamiento de la alta dimensionalidad y el desbalanceo en aprendizaje supervisado" que el trabajo presentado es de autoría propia y confirman que, hasta donde nuestro conocimiento alcanza, cuando se han consultado o utilizado resultados de otros autores, estos han sido citados adecuadamente.

Jaén, 10 de octubre de 2019

Doctorando:

PULGAR RUBIO FRANCISCO JAVIER - 77360107K

Firmado digitalmente por PULGAR RUBIO FRANCISCO JAVIER - 77360107K
Nombre de reconocimiento (DN): c=ES, serialNumber=77360107K, sn=PULGAR RUBIO, givenName=FRANCISCO JAVIER, cn=PULGAR RUBIO FRANCISCO JAVIER - 77360107K
Fecha: 2019.10.10 11:49:06 +01'00'

Francisco Javier Pulgar Rubio

Directores de la tesis:

CHARTE OJEDA FRANCISCO - 25996290B

Firmado digitalmente por CHARTE OJEDA FRANCISCO - 25996290B
Nombre de reconocimiento (DN): c=ES, serialNumber=25996290B, sn=CHARTE OJEDA, givenName=FRANCISCO, cn=CHARTE OJEDA FRANCISCO - 25996290B
Fecha: 2019.10.10 12:54:43 +02'00'

Francisco Charte Ojeda

DEL JESUS DIAZ MARIA JOSE - 26485651R

Firmado digitalmente por DEL JESUS DIAZ MARIA JOSE - 26485651R
Fecha: 2019.10.10 13:06:39 +02'00'

María José del Jesus Díaz

"Actuar sin método oscurece la razón y ciega la inteligencia."

(R. Descartes)

Resumen

Redes de arquitectura profunda y ensembles para el tratamiento de la alta dimensionalidad y el desbalanceo en aprendizaje supervisado

por Francisco Javier Pulgar Rubio

La presente tesis doctoral aborda el estudio de un campo novedoso y prometedor, el uso de nuevas técnicas basadas en aprendizaje profundo y ensembles que afronten los problemas de alta dimensionalidad y desbalanceo de datos. El motivo de optar por este tipo de técnicas se debe al importante auge que han experimentado en los últimos años, ofreciendo resultados realmente relevantes en muchos campos de aplicación. Así mismo, la razón de afrontar los dos problemas anteriormente mencionados es que las características inherentes a los datos cambian constantemente y la tendencia es que tanto la dimensionalidad como el desbalanceo continúen aumentado, por lo que es necesario desarrollar nuevas propuestas que mejoren a los métodos tradicionales.

El trabajo realizado se centra fundamentalmente en afrontar la tarea de reducción de dimensionalidad mediante el uso de un modelo *Deep Learning* concreto, conocido como *Autoencoder* (AE). En este sentido, se realizan tanto análisis experimentales de modelos existentes como nuevas propuestas de métodos de clasificación basados en AE. No obstante, esta tesis analiza también el problema del desbalanceo desde la perspectiva del aprendizaje profundo. Así, los principales estudios desarrollados en la misma son:

- **AEkNN**. Es un nuevo método de clasificación basado en *k-nearest neighbors* (kNN) que incorpora AE para reducir la dimensionalidad del espacio de entrada mejorando el rendimiento predictivo y la eficiencia.
- **Caracterización de AE**. Se trata de un análisis experimental de los modelos más conocidos de AE existentes que permita determinar su rendimiento con distintos algoritmos de clasificación.

- **CNN y desbalanceo**. Este estudio consiste en analizar los efectos que tienen los datos desbalanceados en el rendimiento predictivo de las redes convolucionales, aspecto que no había sido estudiado hasta la fecha.
- **ClEnDAE**. Es un nuevo algoritmo de clasificación basado en ensembles que incorpora un tipo concreto de AE, conocido como *Denoising* AE, para reducir la dimensionalidad del espacio de entrada, con el objetivo de mejorar el rendimiento predictivo de algunos de los clasificadores tradicionales.

Además de la descripción de los trabajos anteriormente citados con su correspondiente experimentación, la tesis incluye un amplio marco teórico donde se describen todos los conceptos relevantes relacionados con los distintos estudios desarrollados.

Agradecimientos

La vida es un camino que, por suerte, no se recorre en soledad. La presente memoria de tesis supone la culminación de una etapa de mi vida, cuatro años centrados en el desarrollo de los trabajos que aquí se presentan, y quisiera recordar con estas palabras, las últimas tras la elaboración de la misma, a todas aquellas personas que me han acompañado a lo largo de esta etapa.

En primer lugar, querría dedicar unas palabras a mis padres, Antonio y Rosa, dos de las personas más importantes de mi vida, que han sido grandes ejemplos a seguir y que me han dado todo para contribuir a mi desarrollo tanto personal como profesional. No hay palabras que puedan agradecer todo lo que han hecho por mí a lo largo de sus vidas. Todo lo que soy es gracias a ellos.

Entre todo lo que mis padres me han dado, lo más importante, sin lugar a dudas, son mis dos hermanos: Jesús y Antonio. Los mejores compañeros y amigos que se pueden tener, dos personas con las que compartir tanto los buenos como los malos momentos y que sé que siempre estarán ahí cuando los necesite. Muchas gracias por todo lo compartido juntos.

Quisiera hacer mención especial a mis abuelas, Carmen y Juana, por todo lo que me han dado a lo largo de sus vidas y me siguen dando en el día a día, y a mis abuelos, Tomás y Paco, por todo lo que me han enseñado, sé que habrían estado orgullosos de mí. Los cuatro darían lo que fuera por mí y por mi bienestar. Siempre tendrán un hueco en mi corazón.

Además, me gustaría mostrar mi agradecimiento al resto de mi familia, a la familia de Silvia y a mis amigos, personas que comparten su tiempo conmigo, se alegran de mis logros y constituyen un apoyo fundamental en los malos momentos. Todos contribuyen, de una forma u otra, mejorar mi día a día. Especialmente, quisiera dar las gracias a Juani por tratarme siempre como a un hijo, y a Antonio por todo lo que hemos pasado juntos. Así mismo, me gustaría nombrar a los niños de la familia: Alejandro, David y África, ellos son los que dan color y alegría a la vida.

De manera especial, quisiera agradecer la labor de mis directores de tesis, Paco y María José, y mi tutor, Antonio. Ellos han contribuido a elaborar y mejorar en gran medida los trabajos presentados en esta memoria. La tesis no ha sido una labor individual, sino que ha sido un trabajo en equipo en el que ellos han tenido un papel fundamental. Muchas gracias.

Es justo también, hacer extensible este agradecimiento al resto de miembros del grupo de investigación SIMIDAT, del que formo parte, por la gran labor de investigación que realizan y por su disponibilidad para ayudarme cuando lo he necesitado. Así mismo, me gustaría agradecer su labor al resto de personas del Departamento de Informática y a todo el personal de administración de la Universidad de Jaén, siempre dispuestos a resolver cualquier cuestión o problema que me haya podido surgir.

Finalmente, quisiera dedicar este trabajo a una persona esencial para mí, sin la que no puedo imaginar mi vida, mi mayor apoyo, mi otra mitad. Ella es la persona que ha elegido compartir su vida conmigo, mi mayor sostén en los momentos de dificultad y que ha tenido una contribución fundamental durante esta etapa de mi vida. Me siento afortunado por haber podido encontrar a una persona capaz de mejorar cada día de mi vida, saber sacar una sonrisa cuando es posible y aguantarme en los malos ratos. Gracias por hacer de mí una persona mejor. *Contigo al lado es más fácil caminar*, gracias Silvia.

Índice general

Índice de figuras

Índice de tablas

Lista de abreviaturas

AE	**A**uto**E**ncoder
AEkNN	**A**uto**E**ncoder **kNN**–based classifier
AUC	**A**rea **U**nder the receiver operating characteristic **C**urve
AI	**A**rtificial **I**ntelligence
BAE	**B**asic **A**uto**E**ncoder
CAE	**C**ontractive **A**uto**E**ncoder
ClEnDAE	**Cl**assifier based on **En**sembles and **DAE**
CNN	**C**onvolucional **N**eural **N**etwork
DAE	**D**enoising **A**uto**E**ncoder
DBN	**D**eep **B**elief **N**etwork
DL	**D**eep **L**earning
DRN	**D**eep **R**esidual **N**etwork
DT	**D**ecision **T**rees
ELM	**E**xtreme **L**earning **M**achine
GAN	**G**enerative **A**dversarial **N**etwork
GRU	**G**ated **R**ecurrent **U**nit
IBL	**I**nstanced **B**ased **L**earning
IR	**I**mbalance **R**atio
ISOMAP	**Iso**metric feature **map**ping
KDD	**K**nowledge **D**iscovery in **D**atabases
kNN	**k**-**N**earest **N**eighbors
LDA	**L**inear **D**iscriminant **A**nalysis
LLE	**L**ocally **L**inear **E**mbedding
LSTM	**L**ong **S**hort-**T**erm **M**emory
MLP	**M**ulti-**L**ayer **P**erceptron
MSE	**M**ean **S**quare **E**rror
NFL	**N**o **F**ree **L**unch
OLTP	**O**n-**L**ine **T**ransaction **P**rocessing
PCA	**P**rincipal **C**omponent **A**nalysis
RAE	**R**obust **A**uto**E**ncoder
RBM	**R**estricted **B**oltzmann **M**achine

RELU	**Re**ctified **L**inear **U**nits
RL	**R**einforcement **L**earning
RNN	**R**ecurrent **N**eural **N**etwork
SELU	**S**caled **E**xponential **L**inear **U**nits
SGE	**S**tochastic **G**radient **D**escent
SVM	**S**upport **V**ector **M**achine

Lista de símbolos

b_j	Parámetro *bias* (sesgo) de la *j*-ésima neurona
E_i	Entrada asociada a la neurona *i*-ésima
γ_i	Función de activación de la neurona *i*-ésima
$max()$	Función máximo
W	Matriz completa de pesos de una red neuronal
w_{ij}	Pesos asociados a la conexión *i*-ésima de la neurona *j*-ésima
FN	Número de falsos negativos
FP	Número de falsos positivos
TN	Número de verdaderos negativos
TP	Número de verdaderos positivos
TPR	Ratio de verdaderos positivos
FPR	Ratio de falsos positivos

Para mi familia.

Capítulo 1

Introducción

El conocimiento extraído de los datos ha jugado un papel vital en la mayor parte de los sectores de la sociedad a lo largo de la historia. Este hecho se ha ido haciendo más relevante en los últimos años, debido fundamentalmente al crecimiento exponencial que ha experimentado la información generada y almacenada por personas, empresas e instituciones. Este crecimiento se debe principalmente a la evolución tecnológica que ha supuesto una revolución en la mayor parte de ámbitos de la vida humana. Las personas, empresas o cualquier otra institución han sufrido cambios importantes en su forma de funcionamiento, necesarios para adaptarse a este nuevo contexto donde las nuevas tecnologías tienen un papel indispensable.

Esta gran cantidad de datos generados no aporta un beneficio por sí misma. La verdadera ventaja consiste en extraer información relevante de los mismos que pueda ser útil a la hora de tomar decisiones, es decir, generar a partir de los datos conocimiento que proporcione una ventaja competitiva en etapas posteriores del proceso de toma de decisiones de cualquier empresa o institución. Este proceso requiere el uso de sistemas de aprendizaje automático que extraigan conocimiento de grandes cantidades de datos, ya que afrontar esta tarea de forma manual no es viable.

En este nuevo contexto, donde la necesidad de extraer información relevante de los datos es cada vez mayor, es esencial desarrollar procedimientos que permitan tratar grandes cantidades de datos facilitando dicha extracción de conocimiento, cobrando especial relevancia el área de investigación conocida como Ciencia de Datos [Hayashi (1998)]. La importancia de este tipo de sistemas se pone de manifiesto en el hecho de que las grandes empresas invierten cada vez más en equipos de análisis y minería de datos cuyo objetivo fundamental es extraer conocimiento relevante de los datos analizados.

Las herramientas de minería de datos tienen diferentes objetivos como puede ser la exploración de las características de los datos, la realización del análisis de los datos que permitan determinar las operaciones de preprocesamiento necesarias para prepararlos de cara al procesamiento posterior y la extracción de conocimiento a partir de los resultados de los métodos previos [Fayyad et al. (1996a) y Tan et al. (2005)]. La aplicación de las herramientas que se acaban de enumerar requiere aplicar métodos existentes o desarrollar propuestas novedosas que solucionen los problemas vayan surgiendo. Esto implica que la minería de datos y el desarrollo de métodos asociados es un área de investigación abierta que evoluciona continuamente adaptándose a las necesidades de la sociedad, a las nuevas características de los datos y a las actuales posibilidades tecnológicas.

Así mismo, esta evolución continua implica que surjan otros tipos de fuentes de datos y que la naturaleza y características de los mismos cambie. Este hecho está directamente relacionado con la aparición de conceptos recientes, como Big Data, de modo que cualquier trabajo que se plantee en esta área debe tener en cuenta los elementos que caracterizan los datos en este contexto [Minelli et al. (2013) y Marr y Von Scheel (2015)].

Este escenario implica que las características de los datos cambien, surgiendo nuevas o acentuándose la aparición de otras ya existentes. Dentro del contexto de esta tesis se analizarán dos de las propiedades que pueden dificultar el aprendizaje como son la alta dimensionalidad (uno de los posibles efectos del volumen de datos) [Guyon y Elisseeff (2003)] que hace referencia a conjuntos de datos con un número ingente de instancias o de atributos asociados a las mismas , y el desbalanceo de datos [López et al. (2013)] que tiene relación con una distribución desequilibrada entre las clases existentes en el dominio del problema. Ambas propiedades serán descritas con mayor detalle en el Capítulo 2 de la presente memoria.

Por otra parte están surgiendo modelos de aprendizaje novedosos que intentan extraer conceptos y relaciones de alto nivel que aparecen en las actuales bases de datos y que implican el uso de arquitecturas más complejas, surgiendo así el concepto de aprendizaje profundo o *Deep Learning* (DL) [Deng (2014), LeCun et al. (2015) y Goodfellow et al. (2016)]. Una de las propiedades más importantes de estos modelos es que pueden reducir la dimensionalidad de los datos al extraer, a partir de los atributos originales, un conjunto de características reducido pero de mayor nivel de abstracción. Esta idea será una de las bases sobre la que se desarrollará la presente tesis, siendo uno

de los principales objetivos el análisis y uso de estas nuevas características en clasificadores tradicionales, así como proponer y desarrollar cambios en las arquitecturas profundas actuales para mejorar la generación de estas abstracciones de alto nivel.

Además, el interés por el desbalanceo de los conjuntos de datos también ha ido aumentando en la comunidad científica, sin embargo sus efectos y posibles soluciones no han sido analizados en el campo del DL, marcándose este estudio como uno de los objetivos de la tesis doctoral que aquí se presenta.

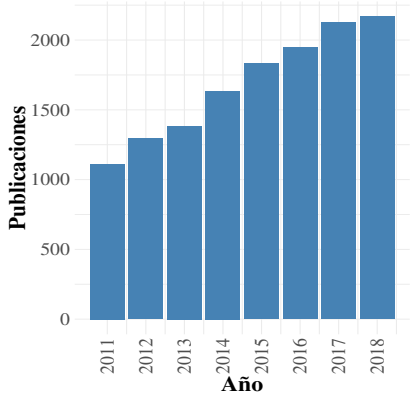
Por último y no menos interesante, se plantea también el estudio del uso de ensembles [Hansen y Salamon (1990)] para mejorar la eficiencia obtenida mediante métodos individuales basados en DL utilizados de forma aislada, dado que la combinación de modelos ha proporcionado buenos resultados en metodologías tradicionales [Rokach (2010b)]. Por esta razón se propone desarrollar ensembles con el objetivo de mejorar la eficiencia de arquitecturas profundas individuales.

Para destacar el creciente interés de los conceptos que se han presentado anteriormente se pueden observar las Figuras 1.1, 1.2 y 1.3, que reflejan el crecimiento, en algunos casos exponencial, en número de publicaciones y citas, en áreas como *"High Dimensionality"*, *"Imbalance"*, *"Big Data"*, *"Deep Learning"* o *"Ensemble"*en el campo de Ciencias de la Computación e Inteligencia Artificial.

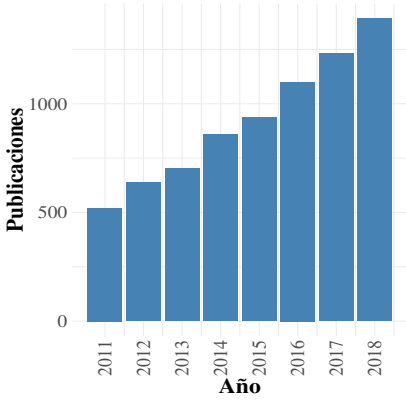
En el resto de secciones del presente capítulo introductorio se presentan los problemas que guiarán el desarrollo de la presente tesis, la motivación asociada a los mismos, los objetivos que se pretenden alcanzar y, finalmente, se describirá la estructura de la memoria.

1.1. Planteamiento del problema

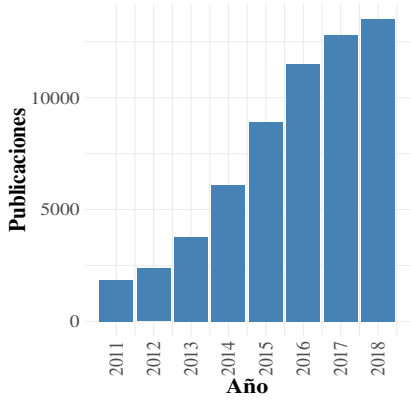
En una sociedad como la actual, que produce millones de datos al día, cobran cada vez más relevancia las áreas de estudio propias de la Inteligencia Artificial (*Artificial Intelligence*, AI), que puede ser definida como la disciplina encargada de desarrollar herramientas que permitan a las máquinas poseer capacidades propias de los seres humanos. Dentro del contexto del análisis de datos destacan campos como el Descubrimiento de Conocimiento en Bases



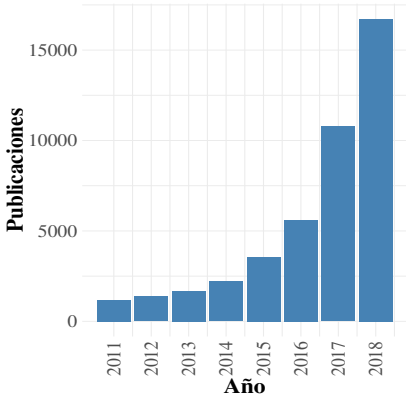
(a) Término de consulta "High Dimensionality"



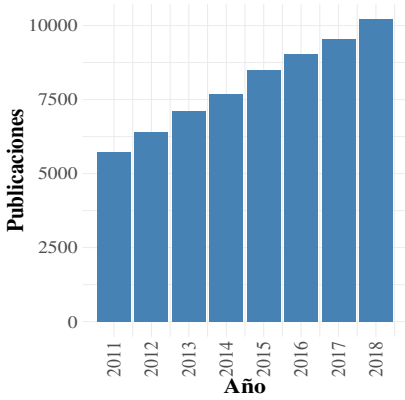
(b) Término de consulta "Imbalance"



(c) Término de consulta "Big Data"



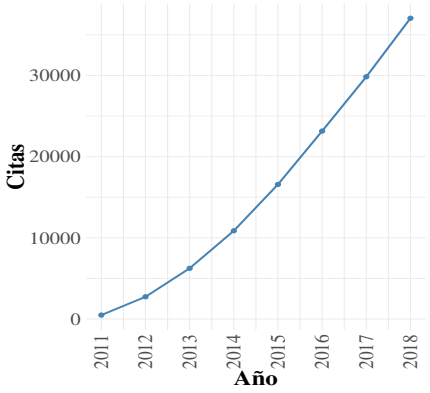
(d) Término de consulta "Deep Learning"



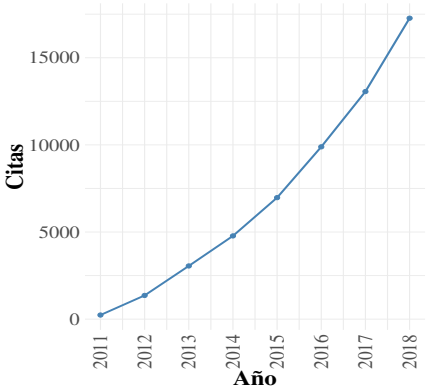
(e) Término de consulta "Ensemble"

Figura 1.1. Publicaciones en los últimos años relacionadas con los términos: *"High Dimensionality"*, *"Imbalance"*, *"Big Data"*, *"Deep Learning"* y *"Ensemble"*. Fecha de consulta: Junio 2019. Fuente: Elaboración propia a partir de los datos consultados en WOS (Clarivate).

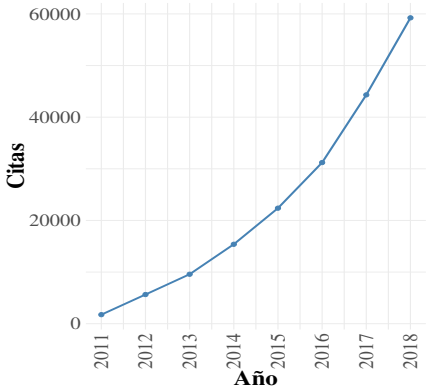
de Datos (*Knowledge Discovery in Databases*, KDD) [Fayyad et al. (1996b)] o la Ciencia de Datos, cuyo objetivo final es obtener conocimiento a partir de dichos datos.



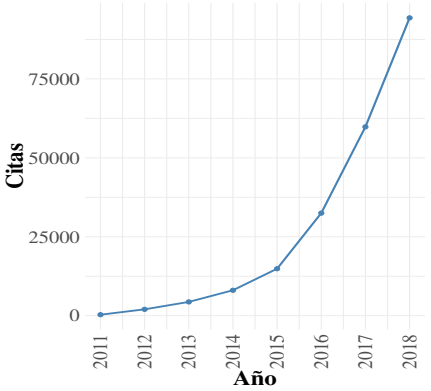
(a) Término de consulta "High Dimensionality"



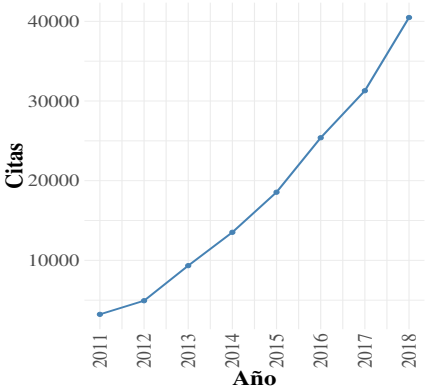
(b) Término de consulta "Imbalance"



(c) Término de consulta "Big Data"



(d) Término de consulta "Deep Learning"



(e) Término de consulta "Ensemble"

Figura 1.2. Citas en los últimos años relacionadas con los términos: *"High Dimensionality"*, *"Imbalance"*, *"Big Data"*, *"Deep Learning"* y *"Ensemble"*. Fecha de consulta: Junio 2019. Fuente: Elaboración propia a partir de los datos consultados en WOS (Clarivate).

Los problemas a los que se enfrentan las técnicas de extracción de conocimiento son cada vez más complejos debido a las características de los datos que se producen en la sociedad actual. Una de las propiedades más importantes de estos datos es su gran Volumen, que unido a otras características

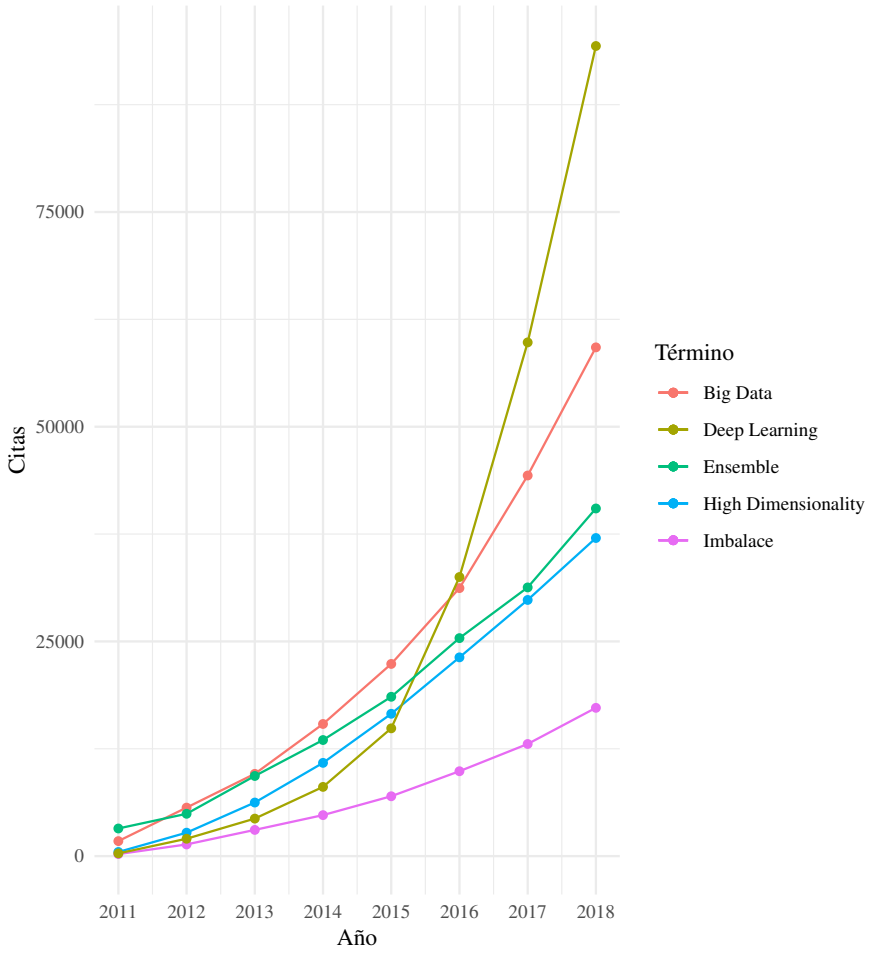


Figura 1.3. Gráfica unificada de las citas en los últimos años relacionadas con los términos: *"High Dimensionality"*, *"Imbalance"*, *"Big Data"*, *"Deep Learning"* y *"Ensemble"*. Fecha de consulta: Junio 2019. Fuente: Elaboración propia a partir de los datos consultados en WOS (Clarivate).

como son la Velocidad o la Variedad, hace que se acuñe para estas nuevas bases de datos el término *Big Data* [Najafabadi et al. (2015)]. Estas propiedades relevantes, que suelen empezar por la letra V, asociadas a dichos conjunto de datos se conocen como *las V's del Big Data*, entre las que destacan, además de las anteriormente mencionadas, la Veracidad, el Valor, la Variabilidad y la Visualización.

En cuanto al volumen de datos hay que destacar que esta citada característica no solo implica una gran cantidad en el número de datos sino también gran dimensionalidad tanto en los espacios de entrada como de salida. Este hecho representa, como se ha indicado anteriormente, un problema dado que un número importante de métodos de clasificación no funciona bien

con conjuntos de datos de alta dimensionalidad [Guyon y Elisseeff (2003)] denominándose a este hecho la maldición de la dimensionalidad (*curse of dimensionality*) [Bellman (1957)].

Para mitigar este problema aparece, en la fase de pre-procesamiento propia de cualquier metodología de extracción de conocimiento, una etapa de reducción de la dimensionalidad [Van Der Maaten et al. (2009)] que tiene como objetivos eliminar características irrelevantes, obtener un modelo más comprensible o reducir tiempos de entrenamiento en los modelos generados. Los métodos usados en este campo tienen como fin la selección de características o la creación de otras nuevas a partir de las antiguas, siempre dando como resultado un nuevo conjunto de datos de menor dimensionalidad que el original. Tanto este problema como las diferentes propuestas que surgen para mitigarlo serán tratados con mayor detalle en la Sección 2.2.1.

Otro tipo de problema que continúa teniendo gran relevancia, debido a la frecuencia con la que aparece en situaciones reales y a que no está totalmente resuelto, es el relativo al desbalanceo en las bases de datos [Chawla et al. (2004)] que hace referencia, como se ha descrito anteriormente, a la existencia de grandes diferencias en la distribución de ejemplos de las clases del conjunto de datos de entrada. Existen distintas aproximaciones que intentan manejar los obstáculos asociados con las bases de datos no balanceadas [Sun et al. (2009)], las cuales se describirán con mayor detalle en la Sección 2.2.2.

En esta tesis doctoral, el objetivo principal es afrontar los problemas anteriormente citados desde una perspectiva que ha tomado una gran relevancia en los últimos años. Esta área de investigación conocida como *Deep Learning* (Aprendizaje Profundo) [Bengio (2009), Deng (2014), Schmidhuber (2015), Goodfellow et al. (2016) y Berzal (2018)] ha aportado grandes resultados a la hora de afrontar tareas inteligentes de orden superior. Es cierto que el DL no es un área nueva pero también es verdad que actualmente acontecen una serie de escenarios que están propiciando el desarrollo de sus técnicas. Estos factores incluyen el desarrollo de tecnologías, tanto hardware como software, para el procesamiento distribuido de la información, el citado incremento en la generación y almacenamiento de grandes volúmenes de datos, y los avances conseguidos en el tratamiento inteligente de la información.

A la hora de diseñar redes neuronales artificiales [Haykin (1994)], tradicionalmente se han usado arquitecturas que se pueden denominar poco profundas (*shallow*), es decir, con pocos niveles o capas. Estas topologías han obtenido buenos resultados para problemas simples y bien delimitados. Sin

embargo, su limitada capacidad de modelado puede causar dificultades, generalmente asociadas a pobres generalizaciones de los datos de entrada, cuando se afrontan aplicaciones del mundo real cuya complejidad es mucho más elevada. Algunos ejemplos de estas aplicaciones son las relacionadas con el reconocimiento de imágenes o sonidos, con el aprendizaje de funciones con una alta variación o con grandes cantidades de datos.

Si bien hace tiempo que se diseñan arquitecturas profundas, su entrenamiento ha sido y sigue siendo un reto. Así por ejemplo, dentro del campo de los perceptrones multicapa han sido varios los intentos de trabajar con arquitecturas con varias capas ocultas pero su capacidad de aprendizaje se ha demostrado muy limitada [Bengio (2009)] debido a que este algoritmo de aprendizaje suele quedar atrapado en óptimos locales. De hecho el trabajo de Hinton [Hinton et al. (2006)] se considera un punto de inflexión en el entrenamiento de este tipo de redes gracias a los resultados obtenidos al entrenar una red profunda denominada *Deep Belief Networks* (DBN). A grandes rasgos este algoritmo entrena vorazmente (*greedily*) cada capa de forma consecutiva y local utilizando aprendizaje no supervisado. Una detallada historia de la evolución de arquitecturas, métodos y aplicaciones en DL puede encontrarse en [Schmidhuber (2015)].

La Sección 2.4 profundiza en el concepto DL, analizando con mayor detalle los motivos que han contribuido a su expansión y las características de este tipo de técnicas. Así mismo se presentan algunas de las aplicaciones más importantes en distintos ámbitos y se describen los modelos más conocidos que se han propuesto. Entre ellos, durante el desarrollo de la presente memoria nos centraremos en analizar algunos aspectos del funcionamiento de las Redes Neuronales Convolucionales (*Convolucional Neural Networks*, CNN) [LeCun et al. (1989)] y, de forma mucho más detallada y precisa, en el análisis y utilización de los *AutoEncoders* (AE) [Olshausen y Field (1996)]. En los siguientes párrafos se exponen los motivos que han llevado a la elección de ambos modelos.

Con respecto a los denominados AE [Olshausen y Field (1996)], estos modelos tienen como objetivo esencial reconstruir la entrada a la salida de la forma más fiel posible, aprovechando para ello la información aportada por los atributos en un proceso de aprendizaje no supervisado. Durante la fase de entrenamiento el modelo genera características de alto nivel a partir de los patrones de entrada. Se trata, por tanto, de una metodología de aprendizaje cuyo objetivo es extraer conocimiento de la estructura interna de los datos

con los que se trabaja a fin de utilizar dicho conocimiento para reconstruir la entrada proporcionada. Los detalles concretos de este tipo de modelos se abordarán en la Sección 2.5 de la memoria.

Los AE en sus diferentes variantes son herramientas que pueden aplicarse de forma general a problemas de reducción de dimensionalidad [Hinton y Salakhutdinov (2006)], generación de características de alto nivel [Coates et al. (2011a)] y reducción de ruido en la información [Vincent et al. (2008a)] debido a su filosofía de funcionamiento y su arquitectura característica. Esto hace que los AE sean una herramienta a tener en cuenta a la hora de tratar la información con la que operan a diario empresas y organizaciones de todo tipo, la cual presenta inconvenientes como la alta dimensionalidad o la existencia de información irrelevante o ruido. A lo largo de la memoria se plantearán diferentes propuestas basadas en la utilización de los AE para afrontar la tarea de reducción de dimensionalidad.

En cuanto a las CNN [LeCun et al. (1989)], estas son uno de los modelos basados en aprendizaje profundo más utilizados y estudiados en la literatura. Esto se pone de manifiesto en las Figuras 1.4 y 1.5 que representan el número de publicaciones y citas, respectivamente, asociadas a algunas de las técnicas de DL que se describen en la Sección 2.4.1 y que se aplican a clasificación. Además del elevado uso de las CNN por parte de la comunidad científica, estos métodos han obtenido un buen rendimiento al afrontar tareas como la clasificación de imágenes [Krizhevsky et al. (2012)] y sonido [LeCun y Bengio (1995)], entre otras áreas de aplicación. No obstante, no existen estudios que analicen cómo afecta el desbalanceo de los datos de entrada al rendimiento predictivo de dichas redes. Por todo ello, uno de los estudios que se han realizado durante la presente tesis doctoral y que queda reflejado en esta memoria es el análisis de los efectos del desbalanceo en los modelos basados en CNN.

Finalmente, otra metodología que se debe tener en cuenta a la hora de desarrollar algoritmos de aprendizaje automático son los *ensembles* o conjuntos de clasificadores [Galar et al. (2012)], ya que han demostrado una mejora en el rendimiento en muchos ámbitos de aplicación. Esta filosofía se basa en la idea de que la integración de varios predictores débiles (*weak learners*) permiten alcanzar grados de precisión prácticamente arbitrarios, es decir, un rendimiento tan alto como sea necesario [Freund y Schapire, 1997]. Esto implica que dicha unión de modelos sencillos tiene un mejor comportamiento que métodos mucho más complejos utilizados de forma aislada. En este

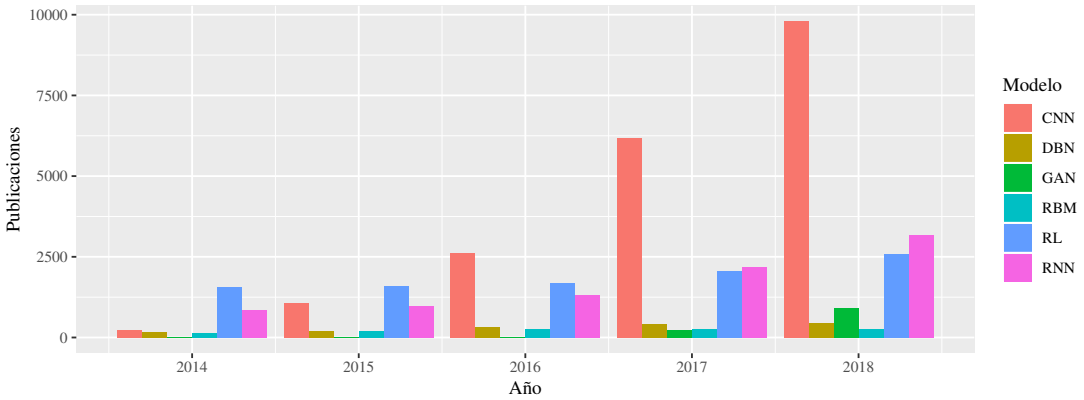


Figura 1.4. Publicaciones en los últimos años relacionadas con los modelos de DL: *Convolucional Neural Network* (CNN), *Deep Belief Network* (DBN), *Generative Adversarial Network* (GAN), *Restricted Boltzmann Machine* (RBM), *Reinforcement Learning* (RL) y *Recurrent Neural Network* (RNN). Fecha de consulta: Junio 2019. Fuente: Elaboración propia a partir de los datos consultados en WOS (Clarivate).

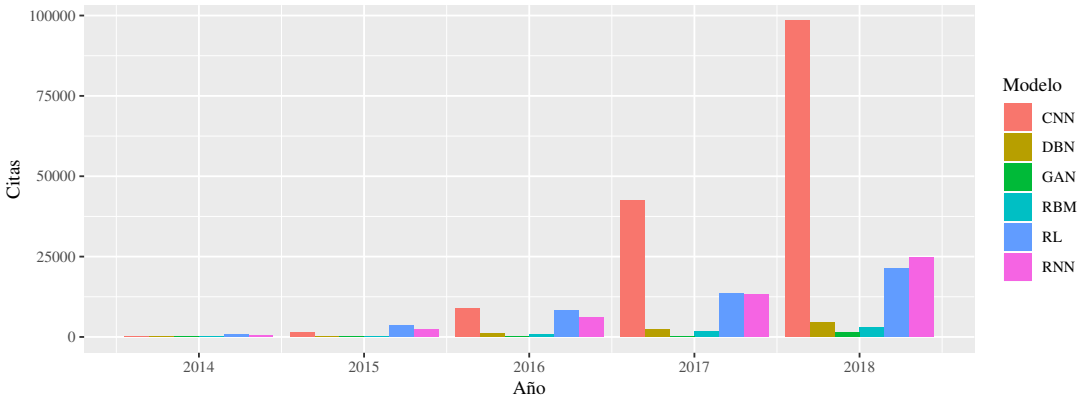


Figura 1.5. Citas en los últimos años relacionadas con los modelos de DL: *Convolucional Neural Network* (CNN), *Deep Belief Network* (DBN), *Generative Adversarial Network* (GAN), *Restricted Boltzmann Machine* (RBM), *Reinforcement Learning* (RL) y *Recurrent Neural Network* (RNN). Fecha de consulta: Junio 2019. Fuente: Elaboración propia a partir de los datos consultados en WOS (Clarivate).

sentido, la Sección 2.3 realiza una descripción detallada de las diferentes tipologías de *ensembles* que existen, así como las metodologías de aplicación más importantes.

En el área de la minería de datos, los *ensembles* se han aplicado no sólo a la resolución de problemas de clasificación sino también a regresión y predicción de series temporales [Sheng et al. (2013), Gabralla et al. (2015) y Rivera et al. (2015)], combinando diferentes tipos de modelos base y considerando en

ocasiones la intervención de métodos estadísticos [Martínez et al. (2014)]. En este contexto, la presente tesis doctoral aborda la creación de *ensembles* de modelos basados en DL que afronten el problema de alta dimensionalidad, citado anteriormente, y que combinen los puntos fuertes de cada una de estas metodologías para proporcionar mejoras en rendimiento con respecto a métodos tradicionales.

En definitiva, esta memoria presentará diferentes estudios cuyo objetivo fundamental es analizar o afrontar algunos de los problemas más relevantes dentro del aprendizaje automático mediante la aplicación y desarrollo de modelos basados en DL.

1.2. Motivación

Esta sección pretende recoger de una forma más esquemática los motivos principales que han llevado a plantear el desarrollo de la presente tesis. El contexto planteado anteriormente lleva asociado la aparición de nuevas líneas de investigación cuyo objetivo es estudiar problemas tradicionales desde esta nueva perspectiva. En este sentido, esta tesis trata de estudiar y dar respuesta a algunas de estas cuestiones:

- ¿Es posible mitigar los efectos de la alta dimensionalidad utilizando métodos con arquitecturas profundas?
- ¿Cómo afectan los datos desbalanceados a la clasificación mediante técnicas DL?
- ¿La combinación de modelos (*ensembles*) basados en DL ofrecerá mejoras con respecto a arquitecturas sencillas?

En general, la meta que se persigue es dar respuesta a problemas abiertos en el campo de la clasificación mediante técnicas basadas en DL. Para alcanzarla, algunas de las tareas que se han planteado resolver durante el desarrollo de este trabajo han sido las siguientes:

- Uno de los problemas más relevantes en la actualidad es la alta dimensionalidad de los datos de entrada, debido fundamentalmente a que se ha incrementado la generación de información en todos los ámbitos. Esta característica de los datos de entrada tiene un efecto negativo en gran parte de los algoritmos tradicionales de aprendizaje automático. En los últimos años han surgido técnicas basadas en DL que afrontan

la tarea de reducción de dimensionalidad. Por esta razón, además de analizar las propuestas existentes, nos planteamos el desarrollo de métodos basados en DL que mitiguen los efectos de la alta dimensionalidad sobre algoritmos de clasificación tradicionales.

- En muchos problemas reales, los datos que se manejan no disponen de una distribución homogénea, es decir el número de instancias de las distintas clases es muy variable. Este problema, conocido como desbalanceo, afecta a la gran mayoría de los métodos de clasificación tradicionales. No obstante, este factor no ha sido analizado profundamente en modelos basados en DL. En este sentido, nos planteamos estudiar cómo afectan los conjuntos de datos desbalanceados al rendimiento predictivo de algunas de estas técnicas, en concreto las redes neuronales convolucionales.
- La combinación de modelos o *ensembles* es una metodología que ha producido un gran rendimiento en entornos donde los datos presentan características que dificultan su aprendizaje de forma automática. Sin embargo, hasta que se planteó el estudio no se habían utilizado *ensembles* de AE para generar características aplicadas a clasificación. Por ello, nos propusimos el desarrollo de *ensembles* con este tipo de técnicas.
- Una de las habilidades del DL es extraer características o abstracciones de alto nivel a partir de una arquitectura de aprendizaje con suficientes capas. La aplicación de las herramientas DL al análisis y extracción de conocimiento a partir de grandes cantidades de datos se presenta como una línea de investigación prometedora, viable y refrendada.

Anteriormente se han descrito las razones principales que han llevado a plantear y desarrollar la presente tesis doctoral. A continuación, a partir de dicha información, se definen unos objetivos y una serie de hipótesis que establecen de forma más concreta el proceso a seguir durante su desarrollo.

1.3. Hipótesis y objetivos

El objetivo general de la tesis doctoral descrita en la presente memoria es el análisis del uso de arquitecturas profundas y *ensembles* en problemas de alta dimensionalidad y desbalanceo. Este objetivo genérico, unido a las preguntas y tareas planteadas en la motivación previa, permiten determinar las hipótesis de partida que guiarán los diferentes estudios presentados en esta memoria:

- El uso de modelos de clasificación que incorporen arquitecturas de aprendizaje profundo para afrontar la reducción de dimensionalidad podría mejorar el rendimiento predictivo en comparación con métodos tradicionales, debido a que las técnicas de DL ayudarán a mitigar los efectos negativos asociados a la alta dimensión de los datos, generando nuevas características de alto nivel.
- Un análisis detallado de las técnicas basadas en DL propuestas para reducir la dimensionalidad permitirá establecer una serie de recomendaciones generales para potenciales usuarios de este tipo de métodos. Dicha guía de uso permitirá seleccionar el modelo más adecuado según el clasificador utilizado en la fase de predicción y las características de los datos de entrada.
- Se sabe que el desbalanceo en los datos de entrada afecta negativamente al rendimiento predictivo de muchos algoritmos de clasificación tradicionales. Por esta razón se pretende analizar los efectos del desbalanceo en el rendimiento predictivo obtenido con CNN, partiendo de la premisa de que, al igual que otros métodos clásicos, las CNN podrían verse afectadas negativamente por dicho factor.
- El desarrollo de modelos basados en *ensembles* que incorporen técnicas de DL para tratar la alta dimensionalidad de los datos podría ofrecer mejoras en rendimiento predictivo frente a clasificadores tradicionales o técnicas DL utilizadas de forma individualizada.

Estas hipótesis permiten definir de forma más precisa los objetivos a cumplir para satisfacerlas. En concreto, en el campo de la alta dimensionalidad se estudiarán las posibilidades de las diferentes arquitecturas profundas para su uso y adaptación a este tipo de problemas, mitigando los obstáculos asociados. En el área del desbalanceo de datos se analizará cómo influye esta característica de los datos en modelos basados en arquitecturas de aprendizaje profundo. Por último se estudiarán y desarrollarán *ensembles* de arquitecturas profundas con el objetivo de mejorar el rendimiento de los métodos individuales utilizados de forma aislada. Así, se proponen los siguientes objetivos específicos:

- **Revisar el estado del arte:** dentro de cualquier metodología de investigación, el primer paso es analizar y determinar el estado del problema planteado en la literatura. En esta primera fase se pretende identificar las

principales arquitecturas de aprendizaje profundo, con especial interés en aquellas con capacidad para reducir la dimensionalidad extrayendo características de más alto nivel.

- **Proponer modelos de clasificación que incorporen reducción de dimensionalidad mediante técnicas de DL:** el objetivo es diseñar y desarrollar nuevos clasificadores que incorporen internamente mecanismos de reducción de dimensionalidad basados en DL. Así, se trata de mitigar los efectos negativos de la alta dimensionalidad.
- **Analizar y caracterizar distintas propuestas basadas en DL existentes para reducción de dimensionalidad:** caracterizar algunos de las metodologías basadas en DL más conocidas y utilizadas que afrontan la reducción de dimensionalidad, determinando las diferencias en cuanto a arquitectura y en cuanto a rendimiento predictivo de cada una de ellos.
- **Analizar el desbalanceo en arquitecturas profundas de aprendizaje:** estudiar la respuesta de arquitecturas DL, en concreto CNN, ante conjuntos de datos desbalanceados. La meta fundamental en este sentido es confirmar si, como ocurre con otras metodologías tradicionales, el utilizar datos de este tipo afecta negativamente a los modelos basados en DL.
- **Desarrollar clasificadores mediante *ensembles* de arquitecturas profundas:** estudiar las métodos existentes en la literatura que combinen técnicas basadas en DL y proponer nuevos *ensembles* de arquitecturas profundas que reduzcan los efectos negativos de la alta dimensionalidad de los datos, mejorando el rendimiento predictivo obtenido mediante los métodos individuales de forma aislada.

Una vez establecidos las hipótesis y los objetivos específicos asociados a la presente tesis, el siguiente apartado presenta la estructura de la memoria donde se abordan cada uno de ellos.

1.4. Estructura

La presente memoria se estructura en una serie de capítulos que presentan los fundamentos teóricos y los resultados de investigación de esta tesis. En concreto, los capítulos, además de la Introducción (Capítulo 1), son los siguientes:

2. **Marco teórico:** en este segundo capítulo se realiza una presentación de los principales conceptos teóricos relacionados con esta tesis. En primer lugar, se contextualiza la tarea de clasificación dentro del proceso general de descubrimiento de conocimiento. Posteriormente, se analizan los principales problemas que se dan en clasificación y que se abordan en esta memoria: alta dimensionalidad y desbalanceo de datos. Finalmente, se introducen los conceptos de *ensemble* y aprendizaje profundo, detallando los modelos más relevantes en relación con los estudios llevados a cabo durante el desarrollo de la tesis.

3. **Tratamiento de alta dimensionalidad con técnicas DL:** el tercer capítulo se dedica a analizar el problema de la alta dimensionalidad del espacio de entrada en clasificación mediante algoritmos basados en instancias (IBL), describiendo algunas de las propuestas existentes en la literatura para mitigar este problema. En este sentido, se presenta un nuevo método, llamado AEkNN, que combina la reducción de dimensionalidad mediante técnicas DL y la clasificación a través de algoritmos IBL con el objetivo de mitigar los efectos negativos de la alta dimensionalidad. Finalmente, se describe una amplia experimentación que refleja la mejora de rendimiento predictivo obtenida por el modelo propuesto.

4. **Estudio sobre la relación entre *autoencoder*, método de aprendizaje y problema:** en el cuarto capítulo se describe un estudio exhaustivo de algunos de los modelos de AE más utilizados y conocidos para afrontar la reducción de dimensionalidad del espacio de características. En concreto, se realiza un análisis completo con el objetivo de determinar qué propuesta es la más adecuada considerando las propiedades de los datos de entrada y la tipología de los clasificadores utilizados. La utilidad fundamental de este capítulo es establecer una serie de líneas de actuación a la hora de afrontar la reducción de dimensionalidad utilizando este tipo de modelos.

5. **Análisis del desbalanceo en DL:** en el quinto capítulo se aborda el análisis de los efectos de datos desbalanceados en el rendimiento predictivo de modelos con arquitecturas profundas. Para ello, se plantea un estudio donde se clasifican conjuntos de imágenes reales con diferente grado de desbalanceo mediante una red convolucional. De esta forma se pretende constatar los efectos de las alteraciones en la distribución de ejemplos entre las distintas clases de entrada sobre el rendimiento de los clasificadores.

6. ***Ensembles* de arquitecturas profundas:** el sexto capítulo se centra en aprovechar las ventajas de los *ensembles* a la hora de mejorar el rendimiento predictivo de los modelos basados en arquitecturas profundas. Por esta razón se presenta un nuevo método, llamado ClEnDAE, cuya arquitectura está formada por un conjunto de modelos individuales basados en DL que son combinados para generar la salida final. Para verificar el buen funcionamiento de la combinación de técnicas DL se describe una amplia experimentación donde los resultados del método propuesto son comparados con modelos aislados, obteniéndose mejoras significativas en este sentido.

7. **Conclusiones y trabajos futuros:** finalmente, el capítulo séptimo presenta las principales conclusiones alcanzadas tras el desarrollo de esta tesis. En primer lugar, se analizan los objetivos inicialmente planteados y los resultados obtenidos en relación a ellos. Así mismo, se enumeran las principales contribuciones que se han realizado a lo largo del desarrollo de la presente tesis y, por último, se exponen las líneas de investigación abiertas de cara a trabajos futuros relacionados con las aportaciones de la presente tesis doctoral.

Capítulo 2

Marco teórico

Los modelos de aprendizaje automático basados en arquitecturas profundas han experimentado, tal y como se ha descrito en el Capítulo 1, un importante auge en los últimos años, debido principalmente al aumento de la capacidad de cómputo de los equipos informáticos y a la gran cantidad de datos disponibles para entrenarlos. Estos factores son fundamentales dada la creciente complejidad en las arquitecturas propias de este tipo de metodologías. En este contexto, han surgido numerosas propuestas aplicadas a campos muy diversos como puede ser la clasificación de imágenes [Krizhevsky et al. (2012)], reconocimiento de voz [Hinton et al. (2012)], traducción automática [Sutskever et al. (2014)] o conducción de vehículos autónomos [Bojarski et al. (2016a)], entre otras.

El hecho de que el uso de las técnicas basadas en DL haya aumentado considerablemente lleva asociado la aparición de nuevas líneas de investigación. En esta tesis se pretende dar respuesta a algunas de estas preguntas abiertas, las cuales se han enumerado en la Sección 1.2.

El primer paso para analizar y abordar cualquier proyecto de investigación es establecer un marco teórico donde se incluyan los principales conceptos a tener en cuenta. En los artículos que se adjunta a esta memoria, como es habitual, no es posible incluir una descripción detallada de dicho contexto. Por está razón se ha considerado adecuado incluirlo aquí.

Este capítulo está dividido en cuatro apartados. La Sección 2.1 se centra en analizar el contexto general donde se sitúa el trabajo, es decir se presentan los conceptos de ciencia de datos y de descubrimiento de conocimiento, así como, se describen las tareas de pre-procesamiento y clasificación. En la Sección 2.2 se exponen los principales problemas asociados a los datos que se abordan en este trabajo. La Sección 2.3 introduce el concepto de *ensembles* o combinación

de modelos desde un punto de vista global. Finalmente, en la Sección 2.4 se pone el foco, entre otros aspectos, en describir el concepto de DL y las principales técnicas que se han utilizado durante el desarrollo de esta tesis.

2.1. Ciencia de datos

A lo largo de la historia, los seres humanos han utilizado información obtenida de distintas fuentes para extraer algún tipo de beneficio. De igual modo, con el paso de los años se han ido creando herramientas cuyo objetivo era tanto la generación como el procesamiento automático de estos datos.

Esta evolución ha llevado asociado un crecimiento exponencial en el volumen de recursos disponibles, así como la aparición de sistemas de almacenamiento como son las Bases de Datos, fundamentales para el almacenamiento y gestión de ingentes cantidades de datos. Sin embargo, los métodos de extracción de conocimiento asociados a estos sistemas tienen el inconveniente de ser pocos flexibles y no ser escalables a grandes volúmenes de información. Por esta razón surgen los conocidos como almacenes de datos (*data warehouses*) [Berson y Smith (1997)] que pretenden reducir los inconvenientes anteriormente descritos. Estos sistemas disponen de un esquema unificado para el almacenamiento de fuentes de datos heterogéneas que facilita su análisis y permite apoyar el proceso de toma de decisiones posterior. Para ello disponen de una serie de operaciones para realizar procesamiento analítico en línea (*On-Line Transaction Processing*, OLTP) orientadas fundamentalmente a extraer información avanzada de los registros analizados [Berson y Smith (1997)]. No obstante, estas técnicas no permiten generar conocimiento que pueda ser utilizado posteriormente. Este hecho lleva a la aparición de un nuevo área de conocimiento conocida como *minería de datos* [Fayyad et al. (1996a), Maimon y Rokach (2010) y Han et al. (2011)] cuyo objetivo fundamental es generar procedimientos que permitan extraer conocimiento útil de los datos disponibles y que pueda ser empleado en etapas posteriores.

El principal factor diferencial de la minería de datos con respecto a técnicas de procesamiento anteriores, como OLTP, es que la información extraída de los datos es inherente a los mismos, es decir este tipo de técnicas genera conocimiento novedoso y útil basándose en los espacios de entrada [Hernández Orallo et al. (2004)]. Sin embargo, la minería de datos es una etapa dentro de un proceso mucho más complejo denominado extracción de conocimiento

desde base de datos (*Knowledge Discovery in Databases*, KDD) [Fayyad et al. (1996a)]. Este procesamiento, además de la minería de datos propiamente dicha, incluye métodos orientados a preparar los datos de entrada que será proporcionada a los distintos métodos, así como formatear y optimizar la información de salida con el objetivo de facilitar la comprensión del conocimiento generado.

Así mismo, el proceso de extracción de conocimiento desde base de datos está dentro de un concepto más general, como es la *ciencia de datos* [Hayashi (1998), Provost y Fawcett (2013) y Waller y Fawcett (2013)]. Este término hace referencia a un área de investigación interdisciplinar que incluye cualquier tipo de procedimiento, método o sistema que se utilice con el fin de extraer conocimiento no obvio a partir de datos en cualquiera de sus posibles formas, estructurados o no estructurados. Se trata de un área global que incluye diversos campos dedicados al análisis de datos como son: la minería de datos, la estadística o el aprendizaje automático.

2.1.1. El proceso de KDD

El objetivo final que se persigue con el proceso de KDD es, como se ha indicado anteriormente, la extracción eficiente de conocimiento útil partiendo de un conjunto de datos originales, en concreto, un proceso KDD abarca un conjunto amplio de métodos y herramientas para conseguir dicho objetivo. De esta forma, las distintas propuestas que han surgido y continúan surgiendo tratan de afrontar el problema actual en el que, a pesar del gran volumen de datos disponible, el conocimiento que se puede sacar de ellos es pequeño debido a la dificultad para tratar estas grandes cantidades de datos.

Desde el punto de vista teórico, existen muchas definiciones del proceso de descubrimiento de conocimiento en bases de datos. En Fayyad et al. (1996a) se define el KDD como *"el proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y, en última instancia, comprensibles a partir de los datos"*. Esta definición pone de manifiesto diferentes aspectos relativos al concepto descrito. Por un lado se refiere a un proceso, lo que implica la realización de diferentes pasos iterativos para conseguir el objetivo final: preparación de los datos de entrada, obtención de patrones válidos, evaluación de dichos patrones y procesamiento final de los resultados obtenidos. Por otro lado indica que se trata de un proceso no trivial, es decir implica la realización de una búsqueda o inferencia asociada a los datos. Finalmente, es importante destacar

que los datos generados deben ser válidos, novedosos, potencialmente útiles y comprensibles, aspectos asociados al fin último del proceso que implica que los usuarios puedan utilizar el conocimiento generado para obtener algún tipo de beneficio.

Como se acaba de describir, el proceso KDD está formado por una serie de fases realizadas de forma iterativa, la Figura 2.1 representa de forma esquemática los diferentes pasos involucrados en esta tarea. En concreto, el punto de partida consiste en comprender el problema que se pretende resolver para poder establecer los objetivos específicos a alcanzar. Posteriormente, es necesario recopilar los datos asociados al contexto analizado, dicha información puede tener orígenes heterogéneos, así como contener errores. Por esta razón, la siguiente fase consiste en realizar labores de pre-procesamiento sobre los datos de entrada, donde se eliminen posibles errores, se unifique la información o se seleccionen los datos más relevantes. En definitiva, esta fase tiene como objetivo preparar los datos de cara a la etapa fundamental del proceso, la minería de datos. El objetivo de esta fase es realizar la extracción de conocimiento útil y no obvio. Para ello utilizará algoritmos de aprendizaje que, a partir de los datos procesados en las fases anteriores, generarán y ajustarán modelos para describir dicha información o realizar predicciones futuras basándose en ella. Finalmente, pueden llevarse a cabo labores de post-procesamiento, que pueden estar orientadas a mejorar la precisión del modelo o a hacer que los resultados proporcionados sean más comprensibles para el usuario final, es decir a mejorar su interpretabilidad. De igual manera, es necesario establecer medidas de evaluación del proceso, a fin de poder comparar el rendimiento global de diferentes modelos y configuraciones.

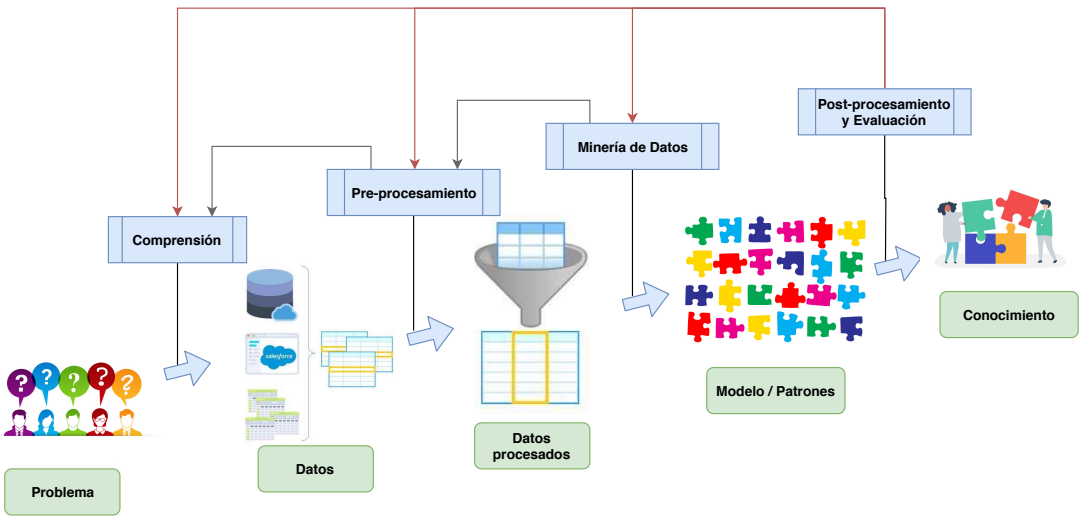


Figura 2.1. Etapas del proceso de KDD.

En este punto, es importante destacar que, como se refleja en la Figura 2.1, este proceso, además de ser iterativo, es interactivo, es decir existe retroalimentación entre cada una de las etapas y la anterior, así como entre la evaluación final y cada una de las fases del proceso.

2.1.2. Pre-procesamiento

Los datos utilizados por los algoritmos de aprendizaje automático requieren, en la mayor parte de las ocasiones, un tratamiento previo para adaptarlos al procesamiento posterior. En concreto, los distintos conjuntos de datos pueden contener valores que no sean válidos o nulos, así como inconsistencias o formatos incompatibles. Por esta razón, la fase de pre-procesamiento es de vital importancia para corregir gran parte de estos problemas.

El objetivo fundamental de esta fase es adaptar los datos de entrada, eliminando los principales inconvenientes que pueden repercutir en los algoritmos de minería de datos, así se optimizará el rendimiento de estos. En este sentido, la primera labor es determinar qué tipo de métodos será necesario aplicar según la naturaleza de los datasets proporcionados [Zhang et al. (2003) y García et al. (2014)].

Este tipo de técnicas son fundamentales para mitigar los efectos negativos de dos de las características que serán analizadas a lo largo de la presente tesis: la alta dimensionalidad y el desbalanceo de los datos de entrada. Así, por ejemplo, en esta etapa es posible aplicar métodos orientados a reducir la dimensionalidad del espacio de entrada mediante selección o fusión de características. Además, se pueden utilizar técnicas de remuestreo que permitan equilibrar la distribución de ejemplos de las distintas clases del problema mitigando los efectos del desbalanceo. No obstante, estas son sólo dos ejemplos de las muchas tareas que se pueden llevar a cabo en la fase de pre-procesamiento. Estas técnicas se agrupan según la tipología de la tarea que pretenden resolver, a continuación se detallan algunas de las más importantes [Ghosh y Jain (2005)].

- **Limpieza de datos.** Este conjunto de tareas tiene como objetivo fundamental la eliminación de inconsistencias y la detección de posibles valores anómalos, es decir, la eliminación del ruido existente en los datos

de entrada, el tratamiento de los valores perdidos o nulos y la identificación de valores extremos. A la hora de afrontar problemas reales es habitual encontrarse con este tipo de inconvenientes en los datos de entrada.

- **Integración de los datos.** El hecho de que los datos procedan de un gran número de fuentes distintas hace fundamental la labor de integrarlos, de tal forma que se puedan tratar de forma conjunta, eliminando posibles incompatibilidades y conflictos [Batini et al. (1986)].
- **Transformación.** Los algoritmos de minería de datos requieren datos de entrada que cumplan ciertas restricciones relativas a su formato y tipo, además en muchos casos determinados algoritmos funcionan mejor con ciertos tipos de datos. Por ello, es necesario realizar transformaciones en el espacio de entrada con el objetivo de que los datos puedan ser correctamente procesados, optimizando el rendimiento de dichos algoritmos.
- **Reducción de dimensionalidad.** El hecho de utilizar datos con un elevado número de instancias y/o variables repercute negativamente tanto en el rendimiento predictivo como en el coste computacional de los algoritmos tradicionales de minería de datos. Por esta razón, existen un gran número de técnicas que afrontan la tarea de reducir el espacio de entrada [Chizi y Maimon (2005)] y mitigar estos problemas.
- **Técnicas de remuestreo.** En determinados contextos donde existe un gran desequilibrio entre las distintas clases asociadas al problema es necesario utilizar herramientas que equilibren dicha distribución, ese es uno de los objetivos de las técnicas de remuestreo [Batista et al. (2004)]. Así, se consigue mitigar los efectos negativos del desbalanceo sobre los métodos de aprendizaje automático tanto generando nuevos ejemplos de las clases minoritarias como eliminando de las mayoritarias.

La Figura 2.2 resume las tareas asociadas a la fase de pre-procesamiento de datos propia del proceso de KDD que se acaban de describir.

Entre la tipología de tareas anteriormente descritas, esta tesis se centrará en afrontar la reducción de dimensionalidad, poniendo especial énfasis en el análisis y utilización de técnicas DL, así como estudiar determinados factores como los efectos del desbalanceo en el rendimiento de determinados modelos

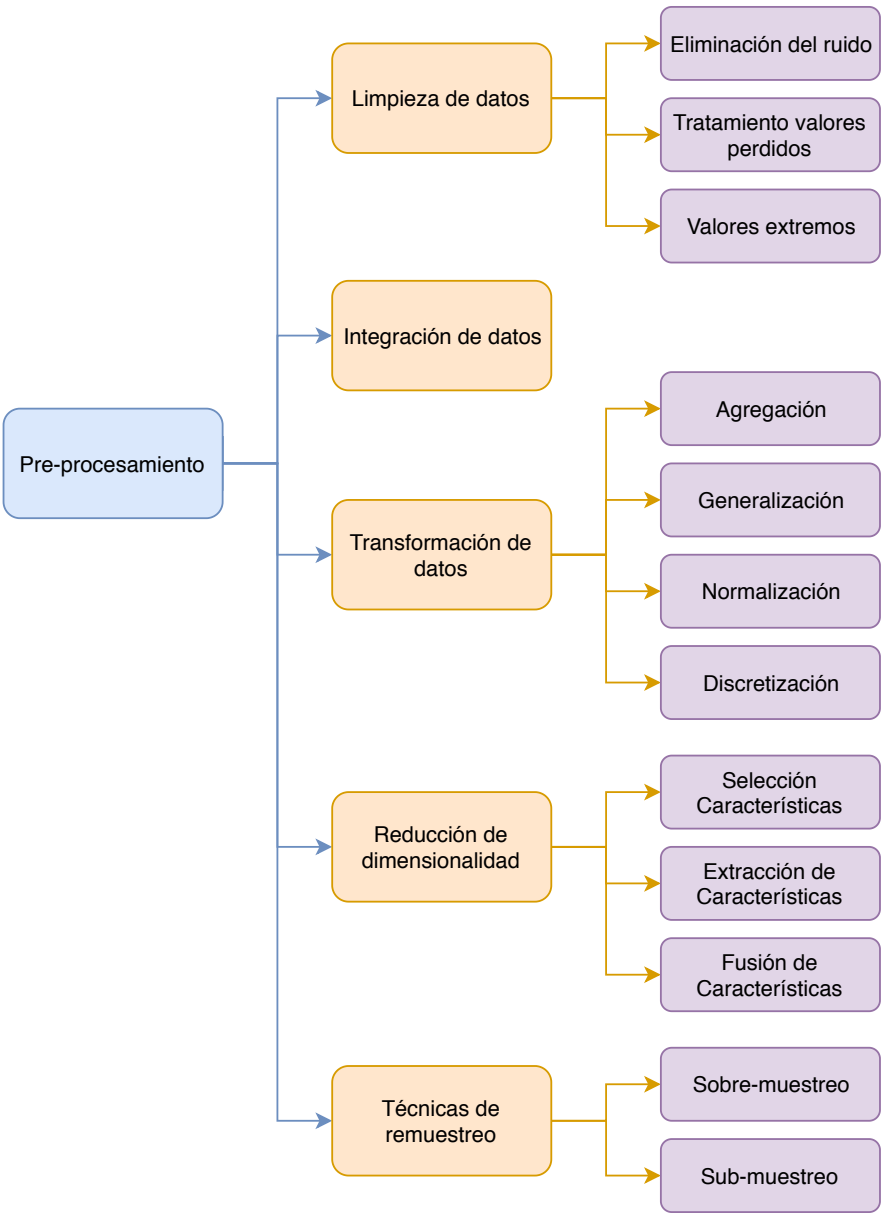


Figura 2.2. Tareas asociadas a la fase de pre-procesamiento.

DL. Por ello, en la Sección 2.2 se profundizará tanto en el problema de la alta dimensionalidad como en el desbalanceo de los datos, así como en las principales propuestas que han surgido para mitigar sus efectos.

2.1.3. Minería de datos

El proceso de KDD, como se ha mencionado anteriormente, tiene como objetivo la extracción de conocimiento a partir de los datos de entrada. Esta labor recae sobre la etapa conocida como minería de datos, las fases anteriores

están orientadas a preparar los datos para optimizar los resultados proporcionados por los algoritmos que se aplican en dicha etapa. Se trata, por tanto, de la fase central del proceso, por ello algunos autores toman el término *minería de datos* como equivalente a todo el proceso de KDD [Han et al. (2011)].

En concreto, los modelos generados en esta etapa tienen como objetivo describir los datos de entrada o establecer relaciones entre los mismos que permitan obtener algún beneficio en etapas posteriores, ya sea realizar predicciones futuras o explicar comportamientos pasados. Así, la minería de datos se puede definir como un conjunto de metodologías y técnicas automáticas basadas en la explotación de grandes volúmenes de dados con el objetivo de proporcionar un modelo que permita explicar situaciones pasadas o determinen el comportamiento futuro asociado a dichos datos [Fayyad et al. (1996a)].

Las técnicas de minería de datos pueden agruparse en distintas categorías atendiendo a múltiples criterios. Teniendo en cuenta el objetivo final que se pretenda conseguir en esta fase, existen dos grandes grupos, los cuales se describen a continuación:

- La minería de datos **predictiva**, cuyo objetivo fundamental es generar modelos basándose en una serie de instancias de entrenamiento que permitan predecir el valor de salida que tendrá una determinada entrada similar en el futuro. Entre las principales tareas que se encuentran dentro de este grupo destacan la clasificación o la regresión [Hand et al. (2001)].
- La minería de datos **descriptiva**, que tiene como meta la descripción de determinados aspectos relacionados con los datos de entrada, como pueden ser su estructura o las relaciones existentes entre ellos. Algunos de los métodos característicos de este segundo tipo son los algoritmos de descubrimiento de asociaciones [Zhang y Zhang (2002)] o *clustering* [Berkhin (2006)].

De igual manera, la filosofía de aprendizaje asociada a los modelos de minería de datos se puede clasificar atendiendo a la existencia o no de etiquetas asociadas a los datos de entrada, así como según el funcionamiento del modelo. En concreto, es posible distinguir entre:

- **Aprendizaje supervisado:** los algoritmos utilizados se basan en datos que, en general, suelen haber sido etiquetados previamente de forma manual por un experto en el área de investigación asociada. Dichos

métodos utilizan la información proporcionada por los datos de entrada para generar modelos que permitan predecir la etiqueta adecuada para instancias que no han sido etiquetadas [Kotsiantis (2007)].

- **Aprendizaje no supervisado:** los datos de entrenamiento de este tipo de modelos no contienen ningún tipo de etiqueta asociada. Por tanto, los métodos utilizados se basan únicamente en la información proporcionada por cada una de las instancias para extraer conocimiento que permita describir la naturaleza de dichos datos [Barlow (1999)].
- **Aprendizaje por refuerzo:** este tipo de métodos utiliza los datos proporcionados para tomar una serie de decisiones que producen una serie de efectos en el entorno. Una vez observados los resultados generados, si son positivos el modelo obtiene una recompensa (refuerzo) con lo que aprende a repetir esa acción en el futuro, mientras que si son negativos recibe una penalización con lo que evitará esa acción [Sutton y Barto (1998)].

Hasta ahora se han descrito varias clasificaciones de las técnicas de minería de datos, sin embargo, los métodos pertenecientes a las categorías anteriores pueden tener objetivos muy diversos. Estos vendrán determinados por la meta asociada al problema que se pretende resolver. De esta forma, las principales tareas que los algoritmos de minería de datos afrontan son las siguientes:

- **Clasificación:** se trata, probablemente, de la tarea más conocida y analizada dentro de la minería de datos. Este tipo de métodos parten de una serie de instancias, cada una de las cuales pertenece a una clase, indicada mediante un valor de tipo categórico [Kotsiantis (2007)]. De esta forma, el método utiliza ejemplos etiquetados para inferir la clase a asignar a patrones que no están etiquetados. El objetivo fundamental de este tipo de algoritmos es maximizar el rendimiento predictivo de nuevas instancias sin etiquetar.
- **Regresión:** la diferencia fundamental con respecto a la clasificación es que, en este caso, el valor que se predice será numérico en lugar de categórico [Draper y Smith (1998)]. Para ello, el modelo aprende una función real que permite asignarle a cada nuevo patrón un valor de salida determinado. El objetivo final es minimizar el error obtenido entre el valor de salida y el valor real.

- **Series temporales:** este tipo de métodos tiene como objetivo el tratamiento de series temporales, es decir datos que guardan entre sí relaciones relativas al tiempo. Por ejemplo, podrían ser lecturas de sensores tomadas en instantes sucesivos de tiempo. En estos casos, son necesarios una serie de métodos específicos para generar conocimiento a partir de ellas, mediante agrupación, clasificación o segmentación de las instancias de entrada. De igual forma, dada la peculiaridad de este tipo de datos, existen determinadas tareas específicas para su tratamiento, como la predicción de series futuras o *forecasting* [De Gooijer e Hyndman (2006)] o la indexación [Faloutsos et al. (1994)].
- **Asociación:** este tipo de métodos tienen como objetivo fundamental establecer asociaciones entre los datos de entrada [Höppner (2005)]. De esta forma, es posible identificar relaciones entre determinadas variables o hechos asociados a valores de atributos concretos. Esto tiene gran aplicación en campos como el marketing. Por ejemplo, es posible identificar qué productos suelen comprarse juntos, pudiendo tomar diferentes decisiones de negocio asociadas a este hecho.
- **Análisis de correlaciones entre variables:** el objetivo de estas técnicas consiste en extraer relaciones entre los atributos de entrada. De esta forma es posible basarse en dichas relaciones para interpretar la estructura interna de los mismos o para afrontar otras tareas como puede ser la reducción de dimensionalidad, eliminando atributos que, dadas las relaciones existentes, sean innecesarios o redundantes [Yu y Liu (2003)].
- **Agrupamiento:** esta tarea consiste en analizar la estructura de los instancias de entrada con el fin de establecer agrupaciones atendiendo a diferentes criterios [Rokach (2010a)]. En este sentido existen métodos para agrupar elementos de una determinada categoría que tenga interés en el contexto del problema, o, por ejemplo, para identificar patrones propios de los valores extremos.

La Figura 2.3 sintetiza las distintas tipologías asociadas a la fase de minería de datos que se han descrito a lo largo de esta sección en forma de taxonomía, con el objetivo de ofrecer una representación visual de todos los conceptos introducidos previamente.

Entre las diferentes tareas de minería de datos, esta tesis aborda diversas cuestiones relacionadas con la tarea de clasificación. Por ello, la Sección 2.1.4 se centra en dicha tarea, así como en describir algunos de los paradigmas y

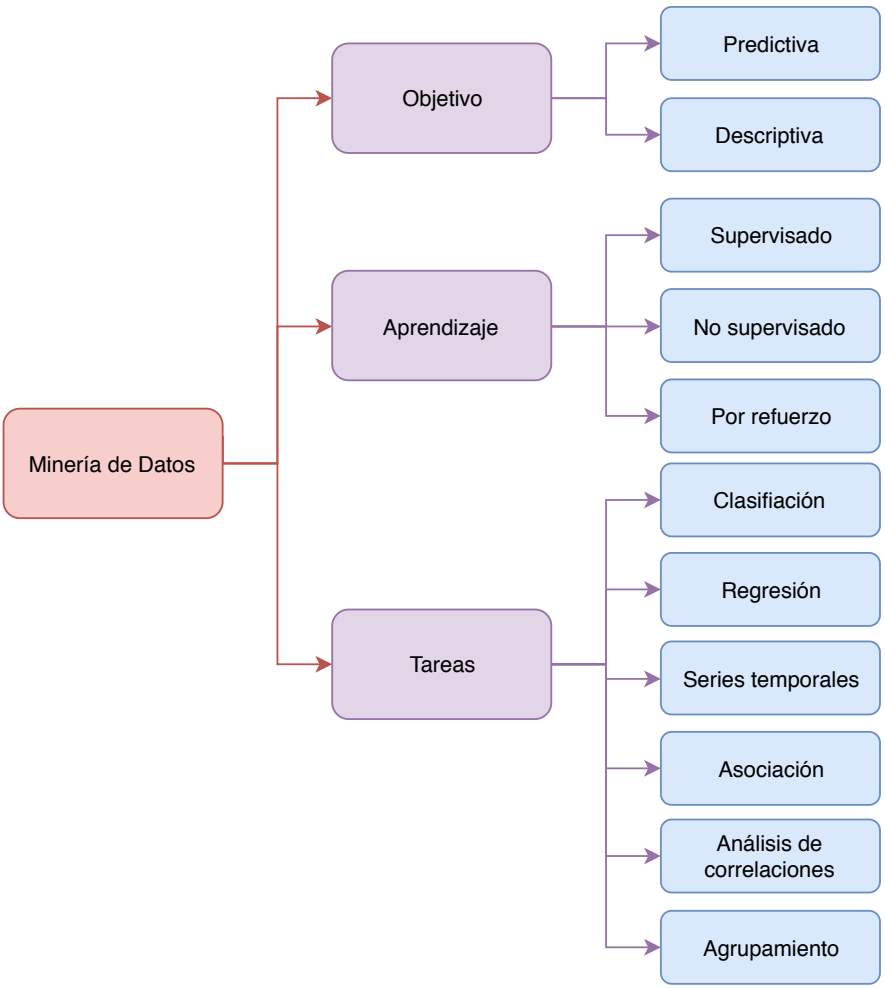


Figura 2.3. Objetivos, técnicas de aprendizaje y tareas de la minería de datos.

algoritmos concretos más conocidos y utilizados, destacando aquellos que tiene relación con los diferentes estudios realizados y descritos en la presente memoria.

2.1.4. Clasificación

La clasificación es, como se ha indicado, una de las tareas más conocidas y abordadas dentro de la minería de datos. El objetivo fundamental es predecir la etiqueta o clase que se asigna a una instancia dada sin etiquetar, basándose en una serie de ejemplos de entrenamiento correctamente etiquetados. En general, se utilizan métodos de aprendizaje supervisados utilizados para realizar dicha predicción [Kotsiantis (2007)]. En este contexto, existen, por un lado, algoritmos que construyen, a partir de los datos de entrenamiento, un modelo que aprenda relaciones de alto nivel presentes en dichos datos y que

permita asociar una clase a nuevos elementos sin etiquetar a partir de esa información. Por otro lado, existen métodos que no necesitan construir un modelo, metodología que recibe el nombre de perezosa o *lazy*, simplemente se basan en la información proporcionada por los atributos de las instancias de entrada de forma directa. Por ejemplo, las redes neuronales [Schalkoff (1997)] construyen un modelo para predecir nuevos patrones, mientras que los algoritmos basados en instancias [Aha et al. (1991)] se basan en la información del resto de ejemplos para generar la predicción.

En clasificación tradicional, los conjuntos de datos utilizados para clasificar están estructurados en una serie de atributos de entrada, conocidos como características, que proporcionarán la información a los distintos métodos para poder llevar a cabo su tarea, y un valor de salida, la etiqueta o clase. El número de valores que pueda tomar la clase o etiqueta, suponiendo que se tiene una sola variable de salida, hace que se hable de dos tipos básicos de clasificación:

- **Binaria:** cada instancia puede pertenecer únicamente a dos posibles clases, es decir la variable de salida sólo puede tomar dos valores.
- **Multi-clase:** el valor de la clase de salida es uno entre un conjunto limitado de posibilidades. Según esta definición, la clasificación binaria es un caso concreto de la clasificación multi-clase.

Desde el siglo XX, se han desarrollado un gran número de métodos que afrontan el problema de la clasificación tradicional, tanto binaria como multi-clase, así como soluciones asociadas a otros tipos de clasificación, como es el caso de la clasificación multi-etiqueta [Herrera et al. (2016)]. Estas propuestas se han agrupado en diferentes paradigmas teniendo en cuenta la filosofía de funcionamiento y la arquitectura de los modelos generados. Los paradigmas existentes son muy diversos y abren grandes posibilidades a la hora de optar por un método de clasificación para resolver un determinado problema. Algunas de estas metodologías son: aprendizaje basado en instancias [Aha et al. (1991)], redes neuronales artificiales [Schalkoff (1997)], redes bayesianas [Heckerman et al. (1995)], máquinas de vectores soporte [Hearst et al. (1998)], árboles de decisión [Quinlan (1986)], sistemas basados en reglas difusas [Zadeh, 1973], técnicas evolutivas y bioinspiradas [Bäck et al. (2000)], algoritmos genéticos [Davis (1991)], lógica difusa [Zadeh (1965)] o programación lógica inductiva [Muggleton (1991)], entre otros.

Los algoritmos propuestos dentro de cada una de estas metodologías tienen características muy diversas que hacen que cada uno de ellos sea adecuado para afrontar un determinado problema o utilizar un conjunto de datos concreto. En este punto es importante destacar que, de acuerdo con el teorema conocido como *no free lunch* (NFL) [Wolpert (1996)], no existe un paradigma o algoritmo que pueda ser aplicado para resolver cualquier problema, ya que si un método demuestra muy buen comportamiento en un determinado contexto, necesariamente ve degradado su rendimiento en otros escenarios. Por tanto, es fundamental conocer la características de cada dominio correctamente, así como el objetivo que se persigue de cara a determinar qué paradigma y algoritmo es el más adecuado. No obstante, en muchos casos es necesario probar diferentes métodos y configuraciones para lograr obtener el mejor rendimiento.

Esta gran variedad de opciones disponibles requiere que los expertos conozcan las características de cada algoritmo y las peculiaridades del problema concreto, entre las que destacan las características de los datos analizados, con el objetivo de determinar el modelo que puede obtener un mejor rendimiento predictivo en el contexto estudiado.

Durante el desarrollo de la presente tesis se han afrontado problemas relacionados con la clasificación, tanto con conjuntos de datos binarios como multi-clase. En concreto, en los diferentes estudios realizados se han utilizado clasificadores pertenecientes a cuatro de los paradigmas más estudiados y conocidos, los cuales se describen a continuación:

- **Aprendizaje basado en instancias (IBL, *Instance-based learning*):** esta metodología basa la predicción en la información proporcionada por las instancias de entrenamiento de forma directa en la fase de inferencia, es decir no necesita construir un modelo [Aha et al. (1991)]. Este tipo de algoritmos se conocen como perezosos o *lazys*. Sus aplicaciones son muy variadas, incluyendo clasificación [Cover y Hart (1967)] o regresión [Kibler et al. (1989)].
- **Redes neuronales artificiales (ANN, *Artificial Neural Networks*):** estos modelos están inspirados en la estructura y funcionamiento del cerebro humano. Los elementos fundamentales utilizados en este tipo de métodos son las neuronas y las conexiones entre ellas. Dichas neuronas son agrupadas en diferentes capas que se encadenan para formar la estructura del modelo [Schalkoff (1997)]. Una idea fundamental asociada a su funcionamiento es que se utilizan transformaciones no lineales

sobre los datos de entrada para determinar las salidas, no solo importan los pesos. Las ANN utilizan su propia experiencia para identificar relaciones entre los datos de entrada, pudiendo ser aplicadas a tareas de clasificación [Dreiseitl y Ohno-Machado (2002)], regresión [Specht (1991)] o predicción de series temporales [Khashei y Bijari (2010)], entre otras.

- **Máquinas de vectores soporte (SVM, *Support Vector Machines*)**: esta metodología se basa en la separación espacial de las instancias de entrenamiento mediante una serie de herramientas conocidas como vectores soporte. En este contexto, el proceso, llamado *kernel trick*, consiste en proyectar los patrones de datos en un espacio de mayor dimensionalidad. De esta forma el modelo es capaz de separar linealmente instancias que, inicialmente no estaban separadas, mediante fronteras de división [Hearst et al. (1998)]. En sus orígenes fueron diseñadas para aplicar clasificación binaria, sin embargo se han desarrollado métodos para clasificación multi-clase [Duan y Keerthi (2005)] o regresión [Cherkassky y Ma (2004)].
- **Árboles de decisión (DT, *Decision trees*):** el funcionamiento y estructura de este tipo de algoritmos está basado en árboles. Así, la arquitectura de estos modelos está formada por diferentes ramificaciones, donde se evalúan los atributos que generan una división más representativa de los datos de entrada, y que terminan en una serie de nodos hoja que contienen los valores de la clase objetivo o de interés [Quinlan (1986)]. Dentro de este paradigma existen métodos aplicados a tareas muy diversas, como clasificación [Vens et al. (2008)], regresión [Breiman (1984)] o descubrimiento de reglas de asociación [Wang et al. (2000)].

Dentro de cada uno de los paradigmas anteriores existen gran cantidad de algoritmos. Durante la elaboración de la presente tesis doctoral se han empleado algoritmos concretos pertenecientes a cada uno de ellos para analizar diferentes cuestiones relacionadas con la tarea de clasificación. En concreto, para los distintos análisis realizados se seleccionó uno de los algoritmos más representativos de cada una de dichas metodologías. A continuación se presentan dichos clasificadores:

- Dentro de las metodologías IBL, uno de los métodos más conocidos y usados es ***k-nearest neighbors* (kNN)**. Se trata de un algoritmo no paramétrico utilizado para tareas de clasificación y regresión [Altman (1992) y Cover y Hart (1967)]. kNN no necesita construir un modelo para

llevar a cabo la tarea de predicción. Se trata de un funcionamiento *lazy* que no hace ningún trabajo hasta que no es necesario predecir una nueva instancia [Atkeson et al. (1997)]. Una vez que llega un nuevo ejemplo, kNN asigna el valor de salida utilizando la información proporcionada por los k ejemplos más cercanos, siendo dicho valor el más común entre los k vecinos.

- ***Multi-layer perceptron* (MLP)** es una de las propuestas tradicionales para afrontar clasificación dentro de las ANN [Hornik et al. (1989)]. El algoritmo puede modelar funciones no lineales y se entrena para aprender y generalizar conocimiento a partir de datos de entrada, ya sea en forma de relaciones entre los datos o características de alto nivel presentes en los mismos. Un MLP tiene la estructura propia de las ANN, la cual se ha descrito anteriormente, estando formado por una serie de neuronas interconectados, organizadas en diferentes capas y ponderadas mediante pesos asociados a dichas conexiones entre ellas. Durante el proceso de entrenamiento, la red utiliza el algoritmo de propagación hacia atrás (*back-propagation*) para transmitir el error a través de la red y ajustar los pesos para minimizarlo [Rumelhart et al. (1986)]. El objetivo fundamental del modelo generado es proyectar los datos de entrada a través de la red, generando el vector de salida [Gardner y Dorling (1998)].

- ***SVM*** es un método de aprendizaje comúnmente utilizado para clasificación. El algoritmo original fue introducido en 1992 [Boser et al. (1992)]. Este tipo de modelos se basan en la separación de las instancias de entrenamiento dentro de un espacio mediante una herramienta conocida como hiperplano que permite maximizar las distancias entre ellas. De esta forma, elementos de una misma clase estarán más cercanos entre sí que elementos de clases diferentes, identificando claramente las fronteras de división existentes. En los casos donde no exista posibilidad de separación lineal entre los elementos, el algoritmo emplea técnicas para llevar a cabo una asociación no lineal del espacio de entrada [Hearst et al. (1998)].

- Existen diversas propuestas basadas en DT para afrontar la tarea de clasificación. Uno de los métodos clásicos en este sentido es **C4.5** [Quinlan (1986)], el cual es uno de los algoritmos más conocidos y utilizados dentro de este paradigma. Fundamentalmente, este algoritmo representa las características que permiten una separación más significativa de

los datos de entrada a lo largo de las ramas del árbol, mientras que los distintos valores de la clase son representados en las hojas. Esto permite obtener reglas de clasificación, una vez el modelo haya sido entrenado [Witten et al. (2016)].

En este punto es importante destacar que, como se ha indicado, existen otras muchas metodologías y algoritmos aplicados a clasificación, únicamente se han enumerado los más relevantes dentro de la presente tesis doctoral. Así mismo, como resumen de las tipologías que se acaban de describir se presenta la Figura 2.4, en la que se han destacado en negrita los principales métodos que han sido objeto de estudio a lo largo de este trabajo.

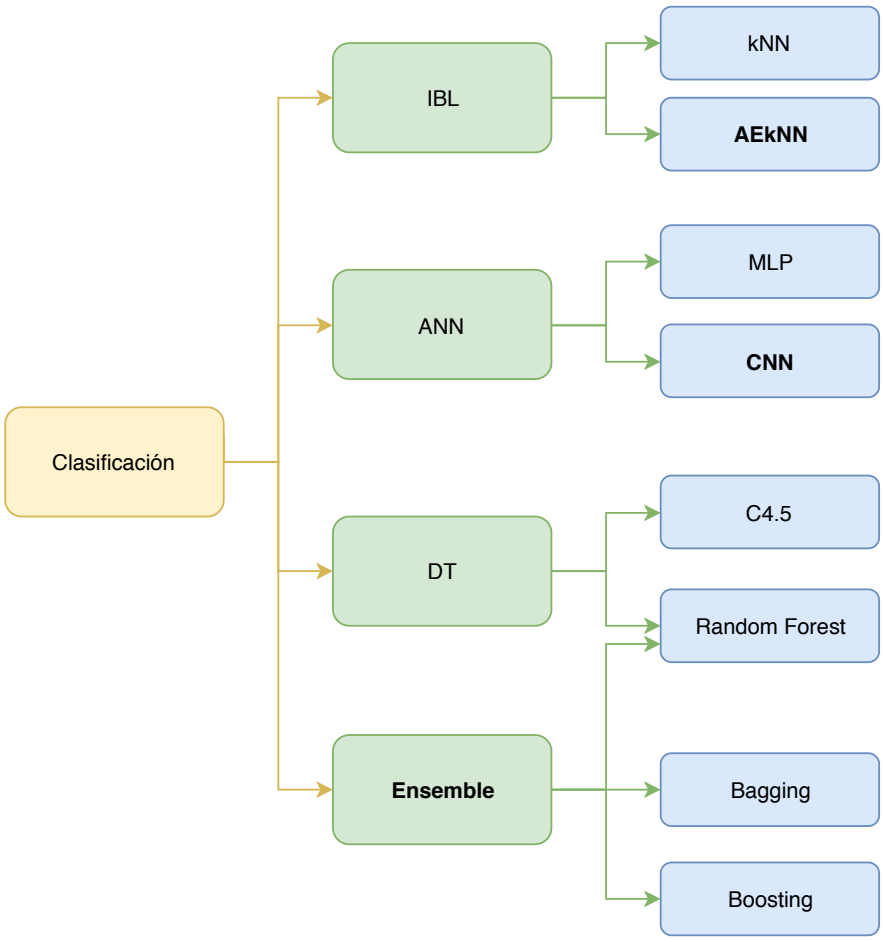


Figura 2.4. Taxonomía simplificada de metodologías de clasificación, en la qe se sitúan las técnicas de interés en el contexto de la presente memoria.

Las distintas metodologías y algoritmos de clasificación presentados anteriormente deben tener en cuenta las características inherentes a los datos asociados al problema que se pretende resolver. Entre estas propiedades se encuentran la alta dimensionalidad o el desbalanceo en los datos de entrada.

En concreto, la Sección 2.2 presenta dichos problemas y describe algunas propuestas tradicionales para resolverlo. Igualmente, en el resto de la memoria se analizan diferentes cuestiones relacionadas con la clasificación tanto de datos de alta dimensionalidad como de conjuntos desbalanceados desde la perspectiva del DL.

2.2. Problemas asociados a los datos

En las secciones previas se ha destacado que la tarea de gestionar la información existente para tomar decisiones en cualquier ámbito continúa resultando de vital importancia. Sin embargo, este proceso ha sufrido grandes cambios debido a distintas razones. Por un lado, el nuevo escenario asociado a la revolución tecnológica ha implicado un aumento exponencial de los datos almacenados y gestionados, que crece de forma continua. Por otro lado, cada día surgen fuentes de información distintas que aportan nuevos datos, siendo fundamental determinar la relevancia de los mismos en función del contexto del problema planteado. De la misma forma, el aumento en la producción y almacenamiento de la información hace que, en determinadas situaciones, aumente considerablemente el desequilibrio entre las distintas clases o grupos de instancias, ya que la producción asociada a las clases mayoritarias es mucho mayor.

Este nuevo escenario requiere cambios profundos a la hora de analizar y utilizar la información existente. Dada la gran cantidad de datos disponible, las tareas de localización y filtrado de la información más relevante cobran una gran importancia. Una correcta selección de los datos que pueden aportar un mayor conocimiento es fundamental para tomar las decisiones correctas y extraer beneficio de ellos. Así mismo, es necesario establecer medidas para tratar de mitigar los efectos negativos que tiene el desbalanceo en muchos algoritmos de aprendizaje automático.

En definitiva, la presente tesis doctoral se centra, como ya se ha comentado, en analizar desde la perspectiva de los modelos basados en aprendizaje profundo, dos de ellas: la alta dimensionalidad y el desbalanceo de los datos, las cuales serán presentadas con mayor detalle en las siguientes subsecciones.

2.2.1. Alta dimensionalidad

En los últimos años, como se ha mencionado anteriormente, la cantidad de datos disponible ha crecido de una forma exponencial. Por tanto, las metodologías y algoritmos de minería de datos deben adaptarse al nuevo escenario, debido fundamentalmente a que, en este contexto, la dimensionalidad de los datos crece tanto en número de instancias como de atributos asociados a los mismos. Este hecho implican que se agraven los problemas relacionados con la creciente dimensionalidad de los datos.

En general, los algoritmos de minería de datos son capaces de trabajar con un número elevado tanto de instancias como de atributos asociados a las mismas. No obstante, el rendimiento de los mismos en muchos casos se ve afectado negativamente a medida que el número de variables crece. Este hecho se debe fundamentalmente al fenómeno conocido como *maldición de la dimensionalidad* [Bellman (1957) y Bellman (1961)]. Esta frase, atribuida a Richard Bellman, fue acuñada para expresar la dificultad para utilizar el método de fuerza bruta para optimizar una función con demasiadas variables de entrada.

Actualmente, este fenómeno puede hacer referencia a varios problemas potenciales que pueden surgir cuando los datos tienen una cantidad elevada de dimensiones:

- Si el conjunto de datos dispone de más características que observaciones surge el riesgo de sobreajustar el modelo de aprendizaje, lo que generaría un rendimiento deficiente fuera de la muestra de entrenamiento.
- Cuando existen demasiadas características, las observaciones se vuelve más difíciles de agrupar, ya que en este escenario las distintas instancias se vuelven equidistantes entre sí, lo que implica que no se puedan formar grupos significativos a partir de los ejemplos de entrenamiento.

Esta circunstancia se puede apreciar en la Figura 2.5 donde se puede observar gráficamente cómo decrece el rendimiento predictivo de los clasificadores al superar un determinado número de características óptimo. Así mismo, otra consecuencia asociada al crecimiento de la dimensionalidad es la necesidad de aumentar el número de instancias de entrenamiento para mantener un nivel de rendimiento aceptable, lo que se conoce como *fenómeno de Hughes* [Hughes (1968)].

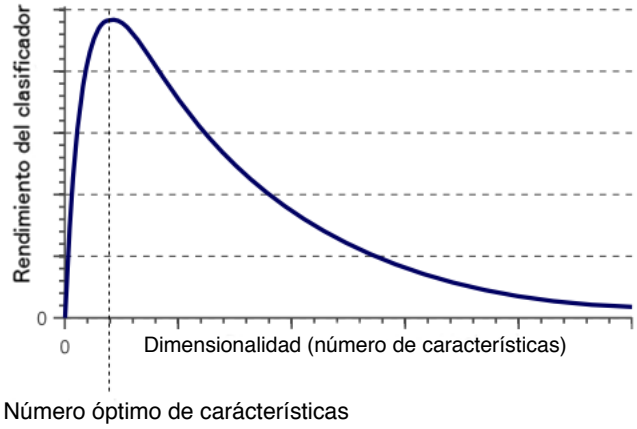


Figura 2.5. Efectos de la alta dimensionalidad en el rendimiento de los clasificadores. Fuente: Towards Data Science [Shetty (2015)].

La alta dimensionalidad de los datos afecta de diferente forma a los distintos paradigmas de minería de datos que se han presentado en la Sección 2.1.4. A continuación, se describen brevemente los principales factores que influyen en cada tipo de métodos:

- Los algoritmos IBL basan su funcionamiento en la información proporcionada por los vecinos más cercanos, por lo que las distancias entre los ejemplos es un elemento fundamental para establecer grados de similitud entre los mismos. En espacios de alta dimensionalidad, las distancias tienden a igualarse, ya que el espacio entre instancias aumenta considerablemente. Por tanto, las distancias entre los distintos ejemplos son menos significativas, lo que afecta de forma directa al rendimiento predictivo obtenido mediante estos algoritmos, que proporcionan resultados mucho menos relevantes.
- Las ANN se basan en la construcción de un modelo que depende de la dimensión de los datos de entrada. Al aumentar la dimensionalidad de dicho espacio, el modelo generado por este tipo de algoritmos aumenta considerablemente. En concreto, el número de neuronas de la capa de entrada es mayor y, en consecuencia, las necesarias en las capas siguientes. Este hecho tiene efectos directos en el coste computacional de dichos métodos. De igual manera, el hecho de aumentar el número de atributos implica que, en muchas ocasiones, se añada información irrelevante o redundante lo que afecta negativamente al rendimiento predictivo de este tipo de modelos.

- En los métodos SVM, al igual que en los IBL, la distancia entre los distintos ejemplos tiene un papel fundamental, ya que este tipo de métodos se basan en la separación de las instancias por medio del hiperplano, obteniendo fronteras de separación entre los elementos de las distintas clases. Por tanto, a medida que aumente la dimensionalidad, las distancias son menos significativas, lo que afecta de forma evidente al funcionamiento de este tipo de algoritmos.
- El proceso de aprendizaje de los métodos propios del paradigma DT consiste en la construcción de un modelo basado en árboles. En espacios de alta dimensionalidad, estos algoritmos requieren construir unas estructuras de mayor tamaño en las que la relevancia de los atributos importantes se disipa debido a la alta dimensión del modelo. Por tanto, el rendimiento de los resultados se ve afectado negativamente.

Estos problemas contribuyeron a la aparición de un nuevo campo de estudio, la *Ingeniería de atributos* [Domingos (2012)], cuyo objetivo fundamental es reducir el espacio de características para mitigar los efectos de la alta dimensionalidad de los datos. Las primeras propuestas consistían en un proceso manual, donde un experto en el dominio del problema se encargaba de decidir por observación qué atributos eran los más relevantes para conseguir el objetivo final. Sin embargo, el proceso se automatizó y comenzaron a surgir métodos de selección automática de características [Marill y Green (1960), Kittler (1978) y Dash y Liu (1997)].

La selección de características es una de las metodologías que existen para afrontar la tarea de reducción de dimensionalidad. El proceso de seleccionar el mejor conjunto de atributos entre todos los posibles es complejo desde el punto de vista computacional. Además, muchos de estos métodos realizan una evaluación individual de cada variable, sin embargo en muchos casos existen atributos que no proporcionan información de forma individual pero sí cuando se combinan con otros. Estos factores contribuyeron a la aparición de nuevos paradigmas, como la extracción de características, todos ellos con el objetivo de reducir el espacio de entrada original. Una completa taxonomía de los paradigmas y términos tradicionales asociados con la tarea de reducción de dimensionalidad se puede encontrar en Charte et al. (2018) y establece la siguiente categorización de conceptos, desde más generales a más específicos:

- **Ingeniería de características:** se trata del término más general, que incluye a algunos de los que se describirán a continuación. Fundamentalmente, hace referencia al proceso general de selección de un nuevo

espacio de características con una dimensionalidad reducida. Los nuevos atributos pueden ser seleccionados de forma directa desde los originales o creados mediante transformaciones a partir de ellos [Domingos (2012)]. Este concepto no realiza ninguna distinción entre procesos automáticos o manuales.

- **Aprendizaje automático de características:** esta metodología incluye el término automático en su denominación, esto quiere decir que hace referencia a aquellos métodos que realizan el proceso de selección o generación del nuevo espacio de características de forma automática [Bengio et al. (2013)]. Esta definición no hace distinción entre métodos de selección o de extracción de características.

- **Aprendizaje de nuevas representaciones de los datos:** este concepto está directamente relacionado con el uso de ANN para llevar a cabo el proceso de generación de características de forma automática. No obstante, este término esta muy relacionado con el anterior, siendo usado a menudo de forma indistinta. La utilización de las ANN para obtener representaciones de los datos es una técnica muy utilizada en la actualidad debido a su funcionamiento y estructura, generando un gran rendimiento a la hora de procesar imágenes [Ciresan et al. (2012a)], texto [Chaturvedi et al. (2016)] o sonido [Hinton et al. (2012)], entre otros.

- **Selección de características:** el objetivo fundamental de los métodos propios de este paradigma es la selección directa del conjunto de atributos más representativo del espacio de características original, es decir, la tarea llevada a cabo consiste en seleccionar un subconjunto de los atributos originales, descartando el resto [García et al. (2014)]. Entre los algoritmos existentes es posible encontrar propuestas basadas en aprendizaje supervisado, que analizan la correlación entre las variables de entrada y salida [Andrew Hall (2000)], y no supervisado, cuyo objetivo es la eliminación de la redundancia de información [Mitra et al. (2002)]. En general, este proceso es muy utilizado en la etapa de pre-procesamiento propia del KDD [García et al. (2016)].

- **Extracción de características:** el objetivo de estas técnicas consiste en generar la mejor representación de la información de entrada, la cual dependerá de las características de los datos y de los algoritmos de aprendizaje utilizados en etapas posteriores [Guyon y Elisseeff (2006)]. El proceso de generar las nuevas características consiste en la aplicación de diversas técnicas, algunas de ellas son: transformaciones básicas de

los datos, normalización, discretización y escalado. Los métodos están basados tanto en aprendizaje supervisado (LDA [Yu y Yang (2001)]) como no supervisado (PCA [Pearson (1901)]). De la misma forma, el nuevo espacio de características puede ser obtenido a través de combinaciones lineales de los datos de entrada, como en PCA y LDA, o mediante combinaciones no lineales, como son ISOMAP [Tenenbaum (2000)] y LLE [Roweis y Saul (2000)]. Los métodos que aplican técnicas no lineales de reducción de dimensionalidad son conocidos como *manifold learning* [Lee y Verleysen (2007)], los cuales han mostrado ser muy efectivos en la generación de nuevos espacios de características, pero con un alto coste computacional.

- **Fusión de características:** este término ha surgido en los últimos años asociado principalmente al procesamiento de datos multimedia, especialmente imágenes, texto y sonido [Mangai et al. (2010), Lin et al. (2014) y Du et al. (2019)]. El principal objetivo de este tipo de algoritmos es generar nuevos atributos a partir de combinaciones de las variables originales. De esta forma, la información menos relevante y redundante se elimina, haciendo que los algoritmos de aprendizaje automático sean mucho más efectivos [Mangai et al. (2010)]. En este contexto, el uso de métodos basados en DL ha experimentado un importante auge, especialmente los AE [Hinton y Salakhutdinov (2006), Chen y Li (2017), Charte et al. (2018) y Maurya et al. (2018)]. Este tipo de modelos han mostrado un gran rendimiento a la hora de descubrir relaciones entre los datos de forma automática, con un coste computacional mucho menor que los métodos tradicionales. Por esta razón, los AE han sido utilizados en distintos estudios realizados durante el desarrollo de la presente tesis doctoral y serán analizados con mayor detalle en la Sección 2.5.

Algunos de los términos introducidos anteriormente se incluyen entre sí, existiendo conceptos generales que engloban a otros más específicos. Para aclarar este aspecto se presenta la Figura 2.6 que incluye una caracterización de los conceptos anteriores.

Las distintas metodologías de algoritmos mencionadas anteriormente incluyen un gran número de métodos que afrontan la tarea de reducción de dimensionalidad desde distintas perspectivas. No obstante, existen métodos clásicos que son muy utilizados en la literatura debido al buen rendimiento

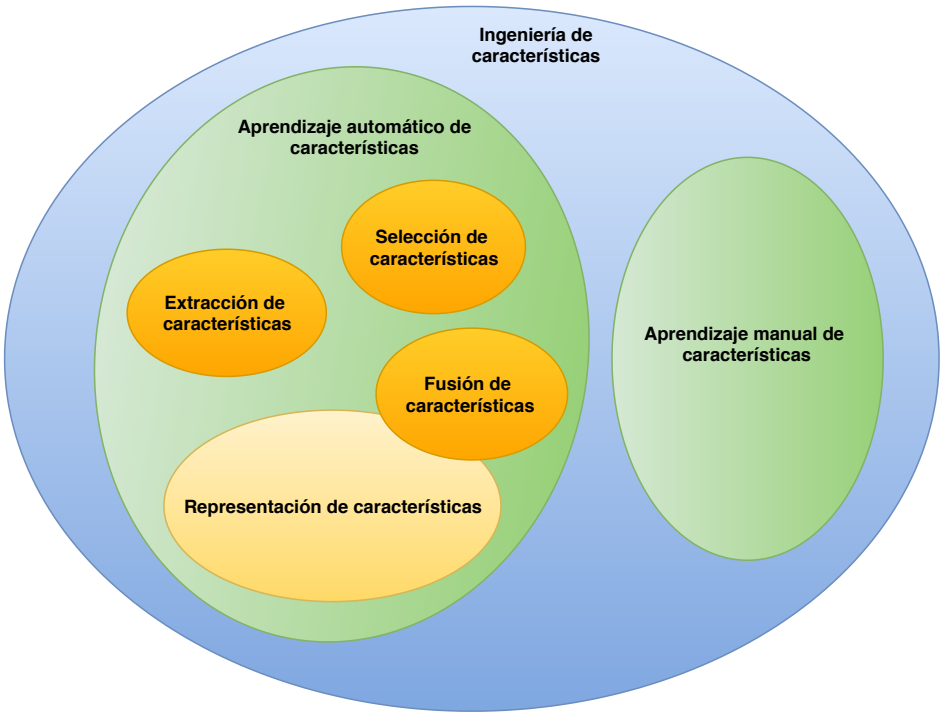


Figura 2.6. Caracterización de conceptos asociados a la tipología de tareas de reducción de dimensionalidad.

que ofrecen. A continuación se describen aquellos que han servido como comparativa para evaluar el comportamiento de las propuestas realizadas en los estudios presentados en esta memoria.

- **Principal Component Analysis (PCA)** [Pearson (1901)]: es uno de los algoritmos de reducción de dimensionalidad clásicos. No obstante, dado su buen rendimiento sigue siendo muy utilizado en la actualidad [Aymaz y Köse (2019) y Artoni et al. (2018)]. El proceso comienza con un conjunto de atributos correlacionados que son convertidos en un conjunto de atributos lineales no correlacionados. Las variables obtenidas son conocidas como componentes principales y su número debe ser menor o igual que el número de atributos originales. Frecuentemente, la estructura interna generada en este proceso refleja la varianza de los datos utilizados.

- **Linear Discriminant Analysis (LDA)** [Yu y Yang (2001)]: se trata, al igual que el anterior método, de una propuesta tradicional que afronta la reducción de dimensionalidad del espacio de entrada y que sigue siendo muy utilizada actualmente dado su rendimiento [Khan et al. (2019) y Yu et al. (2018)]. El algoritmo tiene como objetivo encontrar una combinación lineal de atributos que permite una caracterización o

separación óptima de dos o más clases de objetos. De esta forma, durante el proceso se trata de obtener una nueva representación que refleje las diferencias entre las distintas clases.

- **Isometric feature mapping (ISOMAP)** [Tenenbaum (2000)]: es un método de reducción de dimensionalidad por medio de transformaciones no lineales basadas en el escalado multidimensional básico. La meta de este tipo de métodos es utilizar las distancias entre los distintas ejemplos para preservar la geometría intrínseca de los datos. La clave del proceso consiste en determinar la forma de calcular dicha distancia, conocida como distancia geodésica, partiendo de la distancia Euclídea en un espacio de alta dimensionalidad.
- **Locally Linear Embedding (LLE)** [Roweis y Saul (2000)]: se trata de un algoritmo reducción de dimensionalidad que aplica, al igual que ISOMAP, una serie de transformaciones no lineales sobre los datos originales con el fin de obtener un nuevo espacio de menor dimensión. El proceso seguido por LLE tiene como objetivo proyectar el espacio de características originales en un subespacio de menor dimensión, pero con la condición de preservar las propiedades asociadas a la geometría del espacio original.

En resumen, en esta sección se ha puesto de manifiesto la relevancia de la alta dimensionalidad a la hora de afrontar cualquier tarea de aprendizaje automático, en especial la clasificación. En este contexto, se han presentado también algunas de las propuestas tradicionales que han surgido para mitigar sus efectos. La presente tesis doctoral presenta diferentes propuestas que se basan en la utilización de modelos DL, en concreto AE, para afrontar la reducción de dimensionalidad.

2.2.2. Desbalanceo

El desbalanceo es otra de las propiedades inherentes a los datos que tienen un efecto negativo en una gran parte de los algoritmos de aprendizaje automático. Este factor hace referencia a las situaciones donde existe una distribución no uniforme de muestras por clase o categoría. Esta situación es muy común en algunas de las aplicaciones reales de los algoritmos de aprendizaje automático [Chawla et al. (2004), Sun et al. (2009) y He y Garcia (2009)], como en medicina, biología o telecomunicaciones, entre otras áreas de

conocimiento. El principal factor que afecta al rendimiento es que, en estos campos, la clase de mayor interés es la minoritaria y una predicción errónea puede tener un gran coste [Elkan (2001)].

En clasificación, el proceso de entrenamiento llevado a cabo por muchas de las propuestas tradicionales implica que se produzca un sesgo hacia la clase mayoritaria. En este contexto, estos métodos a menudo obtienen un buen rendimiento al clasificar elementos pertenecientes a las clase con más instancias de entrenamiento. Sin embargo, el rendimiento predictivo a la hora de clasificar ejemplos de la clase minoritaria decrece considerablemente. Por este motivo, algunos de los algoritmos que obtienen buenos resultados con datasets balanceados, no tienen un buen comportamiento con conjuntos desbalanceados [López et al. (2013)]. Existen diferentes razones que podrían explicar este hecho:

- Durante el proceso de entrenamiento utilizado en los modelos tradicionales, se utilizan métricas de evaluación global, que dan la misma importancia a todas las muestras del conjunto de datos sin importar la clase a la que pertenecen. Este hecho implica que las instancias con etiquetas más frecuentes influyen mucho más en el valor final de la evaluación que las pertenecientes a clases minoritarias.
- Dado el menor número de ejemplos de las clases minoritarias, las reglas para predecirlas están muy especializadas y su cobertura global es muy baja. Por este motivo, a menudo son descartadas en favor de otras reglas más generales, es decir, aquellas que predicen a las clases mayoritarias.
- En muchas ocasiones, los métodos de pre-procesamiento, como el tratamiento del ruido, pueden afectar a las clases minoritarias, ya que estas pueden ser identificadas como ruido y descartadas de forma errónea. De igual forma, en caso de no eliminar el ruido, este puede tener una gran repercusión en las clases con menos ejemplos.

Estos motivos ponen de manifiesto que el hecho de entrenar a los clasificadores tradicionales con conjuntos de datos con un alto nivel de desbalanceo implica que el rendimiento al predecir elementos de las clases minoritarias se vea afectado negativamente. Por todo ello, en los últimos años han surgido diferentes propuestas que tratan de mitigar los efectos de este problema, tanto basados en metodologías tradicionales como en combinación de modelos [Galar et al. (2012)]. Los diferentes métodos propuestos se pueden clasificar fundamentalmente en tres grandes grupos:

- **Muestreo de datos:** este tipo de métodos se basan en modificar la distribución de ejemplos del conjunto de datos utilizado por el algoritmo de clasificación. El objetivo fundamental es procesar el dataset original y generar un nuevo conjunto con una distribución de clases más balanceada. Esta filosofía no tiene ninguna repercusión en la estructura y funcionamiento de los algoritmos de clasificación tradicionales utilizados [Batista et al. (2004)]. En la literatura existen diferentes propuestas en este sentido, alguna de ellas son los algoritmos SMOTE [Chawla et al. (2002)] y ENN [Wilson (1972)].
- **Adaptación de algoritmos:** la filosofía seguida por esta tipología de métodos consiste en adaptar los algoritmos de clasificación tradicionales para que trabajen de forma adecuada con conjuntos de datos desbalanceados [Fernández et al. (2013)]. El objetivo es realizar modificaciones internas en los algoritmos a nivel de estructura o funcionamiento que permitan mitigar los efectos del desbalanceo. En estos casos, los conjuntos de datos originales se mantienen sin ninguna modificación [Zadrozny y Elkan (2001)]. Existen multitud de métodos que adoptan esta filosofía, algunos de ellos están recogidos en los siguientes artículos: [Barandela et al. (2003) y Diamantini y Potena (2009)].
- **Aprendizaje sensible al coste:** las propuestas basadas en esta tipología pueden implicar adaptaciones a nivel de conjunto de datos, a nivel de algoritmos o hibridaciones de ambas [Krawczyk et al. (2014)]. Fundamentalmente, el proceso de entrenamiento seguido por este tipo de métodos incorpora penalizaciones o coste más elevado a los errores cometidos al predecir ejemplos de las clases minoritarias, en comparación con los cometidos con las clases mayoritarias. De esta forma, se pretende compensar la diferencia de elementos entre ambos tipos de clases [Zadrozny et al. (2003)]. Una propuesta incluida dentro de esta metodología es el algoritmo *MetaCost* [Domingos (2002)].

Existen muchos estudios que han relacionado de forma directa el desbalanceo entre las clases del dominio del problema con la pérdida significativa de rendimiento en muchos de los clasificadores tradicionales. Este desequilibrio a menudo es identificado como ratio de desbalanceo (*imbalance ratio*, IR), que hace referencia a la proporción entre casos de las clases mayoritarias y casos de las clases minoritarias [Orriols-Puig y Bernadó-Mansilla (2008) y García et al. (2012)].

Tal y como se ha explicado anteriormente, el problema del desbalanceo ha sido y es ampliamente estudiado dentro del área de aprendizaje automático. En la actualidad, la importancia del tratamiento del desbalanceo, al igual que la alta dimensionalidad, está creciendo debido a los cambios profundos que se han producido en las propiedades de los datos utilizados en muchos campos de aplicación. Así, existen numerosas propuestas que tratan de mitigar los efectos de este tipo de conjuntos de datos en muchos campos, algunos de los cuales son: reconocimiento facial [Ko et al. (2008) y Huang et al. (2019)], tratamiento de imágenes [Chen et al. (2011)] o diagnóstico médico [Mazurowski et al. (2008) y Khatami et al. (2018)].

En resumen, en esta sección se ha analizado el problema asociado al desequilibrado entre las clases existentes en un determinado conjunto de datos a la hora de afrontar la tarea de clasificación, así como sus efectos en muchos de los algoritmos tradicionales. Además, se ha plasmado el interés existente en la comunidad científica por mitigar los efectos de este problema presentando algunas de las metodologías tradicionales para resolverlo. No obstante, la evolución de los paradigmas de clasificación, así como la aparición de nuevas técnicas hace que sea necesario analizar los efectos que tienen las distintas propiedades de los datos, en concreto el desbalanceo, sobre las mismas. Por esta razón, uno de los estudios planteados durante el desarrollo de la presente tesis pretende analizar la respuesta de determinadas arquitecturas profundas, específicamente las CNN, ante la presencia de conjuntos de datos desbalanceados.

2.3. *Ensembles*

El aprendizaje basado en *ensembles* o conjuntos de clasificadores pretende mejorar el comportamiento de los algoritmos de aprendizaje automático mediante la combinación de varios modelos. Este paradigma permite desarrollar propuestas que mejoran el rendimiento predictivo frente a modelos individuales utilizados de forma aislada.

Así, un *ensemble* es un algoritmo que combina un conjunto de modelos de aprendizaje automático en un mismo modelo predictivo [Hansen y Salamon (1990)]. Esta metodología utiliza estructuras que integran varios componentes

que resuelven la misma tarea, mejorando el rendimiento de los métodos individuales utilizados aisladamente [Rokach (2010b)]. Los modelos basados en *ensembles* proporcionan una serie de ventajas que se describen a continuación:

- En determinadas metodologías, por ejemplo *bagging* [Breiman (1996)], cada uno de los modelos que forma el *ensemble* utiliza su propio subconjunto de datos a partir del conjunto original. De esta forma, cada uno de ellos puede aprender relaciones entre los datos de entrada diferentes, mientras que los modelos individuales no disponen de variabilidad en el conjunto de datos utilizado.
- La información de alto nivel generada por cada modelo se combina, obteniendo un perfil de clasificador que no sería posible generar utilizando modelos individuales aislados.
- La región de búsqueda de los algoritmos basados en *ensembles* puede ser mucho más amplia que la de los métodos utilizados de forma aislada. Esto es debido a que la elección de diferentes subconjuntos de datos, tanto a nivel de instancias como características, permite explorar determinadas regiones espaciales, algo que no sería posible si se tratará únicamente el conjunto completo.
- La arquitectura de los *ensembles* es muy adecuada para computación distribuida, dado que su naturaleza permite que los diferentes modelos se adapten perfectamente para ser ejecutados de forma paralela.

En este contexto, existen diferentes categorizaciones de los métodos basados en combinación de modelos atendiendo a diferentes criterios. En primer lugar, si se tiene en cuenta la forma en la que se generan los distintos modelos que componen el *ensemble* se puede tener:

- **Secuencial:** los modelos se generan de forma secuencial durante la ejecución del método de aprendizaje global. El objetivo fundamental de este tipo de algoritmos es que cada modelo dependa del anterior. Una propuesta clásica incluida en esta categorización es el algoritmo AdaBoost [Freund y Schapire (1997)].
- **Paralelo:** los distintos modelos que componen el *ensemble* se generan de forma paralela durante el proceso asociado al modelo global. Así, cada uno de los sub-modelos de aprendizaje explota el espacio de entrada

de forma independiente. Finalmente, la salida de cada uno de ellos es agrupada para generar la salida final. Un algoritmo clásico incluido en esta categoría es Random Forest [Breiman (2001)].

En segundo lugar, atendiendo a la variabilidad entre los diferentes modelos que forman el *ensemble*, se pueden distinguir dos categorías. Por un lado, *ensembles* **homogéneos** que están compuestos por modelos similares entre sí, que tiene pocas variaciones en cuanto a estructura y funcionamiento, y multiclasificadores o *ensembles* **heterogéneos** donde se utilizan modelos con arquitecturas distintas y funcionamiento muy diverso. En general, la diversidad aumenta el rendimiento predictivo de los *ensembles*, ya que permite ampliar el espacio de búsqueda y explorar determinadas regiones que no sería posible sin esa diversificación.

Finalmente, existen diferentes paradigmas orientados al desarrollo de modelos basados en *ensembles*. A continuación, se presentan los más conocidos y usados en la literatura relacionada [Galar et al. (2012)]:

- ***Bagging*:** también conocido como *Boostrap aggregation* es uno de los paradigmas básicos utilizado para el desarrollo de *ensembles* [Breiman (1996)]. Esta metodología se basa en el método estadístico de *boostrapping* [Efron y Tibshirani (1994)] para evaluar de forma conjunta la ejecución de varios modelos ejecutados en paralelo. El objetivo que se persigue es la reducción de la varianza mediante y la generación de resultados más estables y precisos. *Bagging* no usa el conjunto completo original para entrenar cada uno de los sub-modelos, sino que cada uno de ellos se entrena con un subconjunto de las instancias seleccionado aleatoriamente.
- ***Boosting*:** es un algoritmo clásico utilizado en el desarrollo de métodos basados en *ensembles* [Freund y Schapire (1996)], que se basa en la reducción del sesgo mediante la generación de diferentes modelos que se centran en las instancias mal clasificadas. Fundamentalmente, los diferentes modelos se agregan de forma secuencial y los pesos asociados al *ensemble* se ajustan en función del rendimiento de cada uno de ellos. Durante el proceso, el modelo tiende a dar más peso a aquellas instancias que son mal clasificadas, por lo que los siguientes sub-modelos se centrarán en ellas. De esta forma, el *ensemble* realiza una actualización de pesos por sí mismo para obtener un modelo final que se adapte de forma más precisa al conjunto de entrenamiento. Existen diferentes propuestas dentro de este paradigma, entre ellas, AdaBoost [Freund y Schapire (1997)] es uno de los algoritmos más conocidos.

- **Stacking:** también conocido como apilamiento, es otro paradigma utilizado para el desarrollo de *ensembles*. El objetivo fundamental es entrenar un modelo global a partir de las salidas proporcionadas por dos o más modelos utilizados previamente [Wolpert (1992)]. Por tanto, se trata de una estructura secuencial, donde las salidas proporcionadas por los componentes previos se combinan y se utilizan como entrada de los modelos sucesivos que generan un nuevo conjunto de predicciones. Un aspecto importante es que todos los modelos que se incluyen utilizan el conjunto de datos original completo.

A la hora de generar la salida final del modelo, tanto *bagging*, como *boosting* agregan las salidas de cada uno de los modelos que componen el *ensemble*. Para ello, se suelen utilizar diferentes técnicas, en clasificación destaca el voto mayoritario, mientras que en regresión se suele optar por la media aritmética.

Hasta este punto, esta sección ha presentado algunas de las categorizaciones que pueden encontrarse en relación con la combinación de modelos. En este sentido, la Figura 2.7 pretende resumir las distintas tipologías que se han descrito anteriormente.

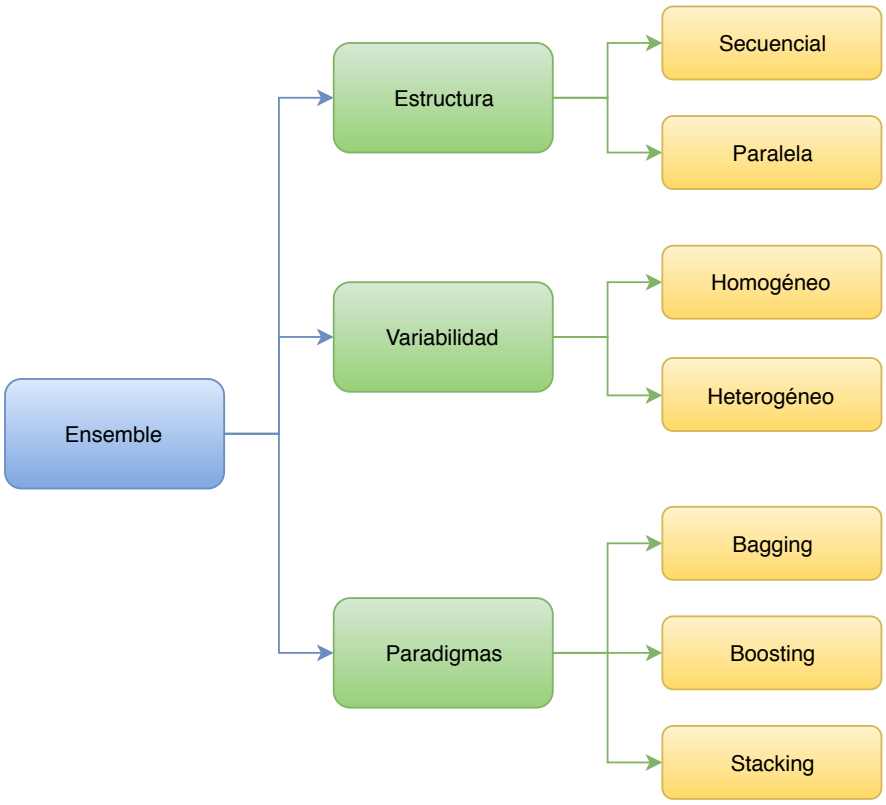


Figura 2.7. Taxonomía de las tipologías de *ensembles*.

Tras realizar la categorización anterior, cabría destacar el algoritmo *Random Forest* dada su relación con una de las propuestas realizadas en la presente tesis doctoral. Se trata de un método de aprendizaje automático que se basa en la construcción de una estructura formada por la combinación de un conjunto de árboles de decisión [Breiman (2001)]. El método introduce distintas variaciones en el conjunto de árboles de decisión que forman el modelo mediante la combinación de *bagging* tradicional y la selección de atributos del conjunto original de forma aleatoria. Durante el proceso, cada árbol es construido a partir de una muestra distinta extraída del conjunto de entrenamiento. Así mismo, el hecho de utilizar un subconjunto de las características originales obtenidas de forma aleatoria, permite conseguir una diversificación adicional del árbol.

Durante el desarrollo de la presente tesis se ha desarrollado un *ensemble* que guarda similitudes con el método *Random Forest*. La Figura 2.8 presenta de forma esquemática el proceso asociado a la propuesta realizada, la cual se presenta de forma detallada en el Capítulo 6.

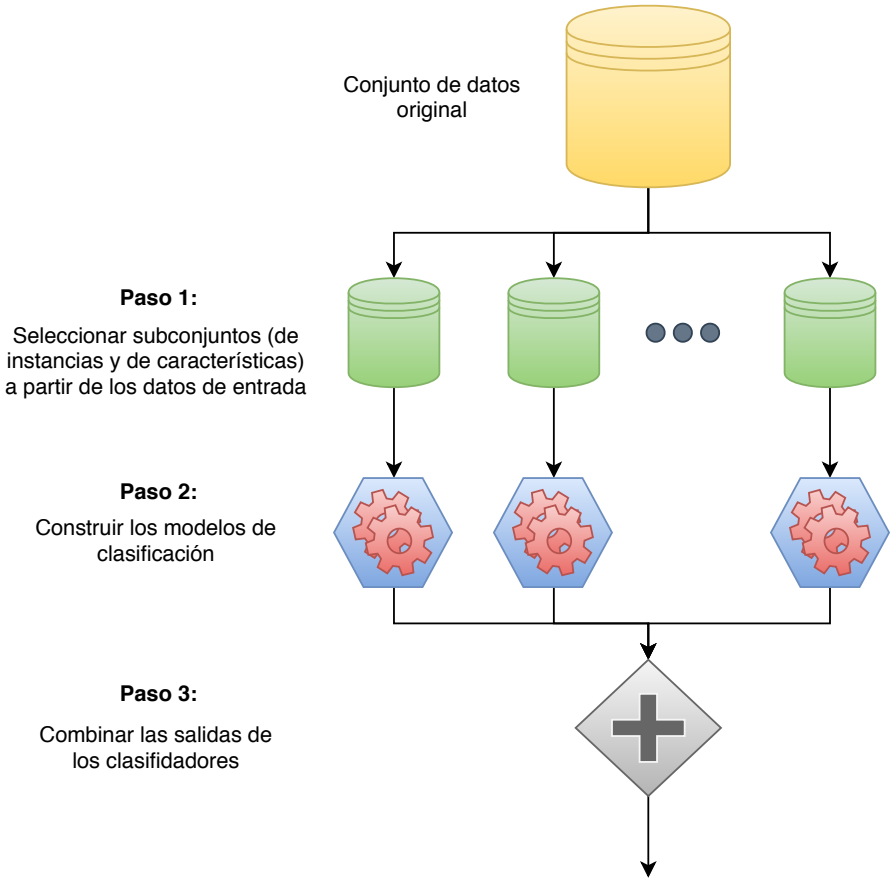


Figura 2.8. Estructura básica de un *ensemble* basado en *Random Forest*.

En resumen, los *ensembles* han sido aplicados para resolver multitud de problemas propios de la minería de datos, tanto para clasificación como regresión y predicción de series temporales [Sheng et al. (2013), Gabralla et al. (2015) y Rivera et al. (2015)]. En este sentido, la presente tesis doctoral desarrolla nuevas propuestas que combinen modelos basados en DL para afrontar la tarea de reducción de dimensionalidad. De esta forma, el modelo generado combinará las ventajas de ambas técnicas para mejorar el rendimiento predictivo.

2.4. *Deep Learning*

Uno de los desafíos fundamentales de la AI consiste en desarrollar software que permita automatizar diferentes tipos de tareas llevadas a cabo por los seres humanos. El verdadero reto surgió a la hora de plantear la resolución de tareas fáciles desde el punto de vista humano, pero que no tienen una descripción formal sencilla, es decir, problemas que los seres humanos resolvemos de forma intuitiva, por ejemplo el reconocimiento de imágenes o sonido. Esta sección presenta una metodología que afronta este tipo de problemas, conocida como *Deep Learning* (DL) [Deng (2014), LeCun et al. (2015) y Goodfellow et al. (2016)]. Este campo de investigación se encuentra dentro del aprendizaje automático, la Figura 2.9 sitúa el concepto DL dentro de la AI.

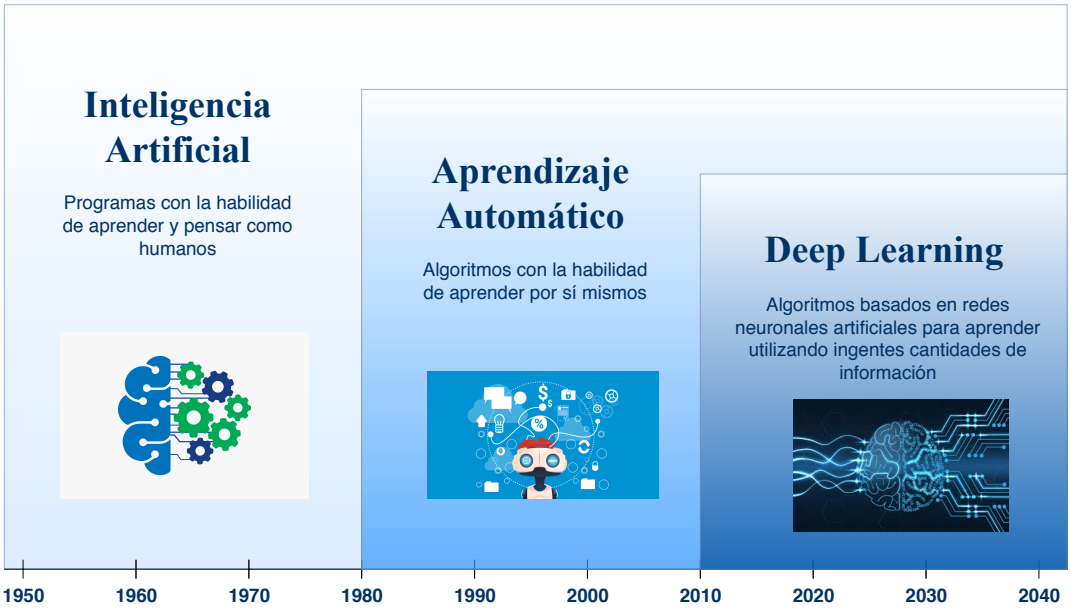


Figura 2.9. Situación *Deep Learning* dentro de la Inteligencia Artificial. Fuente: Elaboración propia basada en contenido de Data Science Central [Venkatesan (2018)].

Los orígenes del DL están directamente relacionados con los orígenes de las redes neuronales artificiales. No obstante, hasta hace algunos años, una serie de restricciones, entre ellas las relacionadas con el propio hardware, hacían que no fuera posible plantearse la ejecución de este tipo de modelos. En este sentido, algunas de las técnicas DL que hoy se utilizan se basan en propuestas realizadas en los años 80 o 90, pero con un rendimiento y capacidad de ejecución limitados. Existen dos factores fundamentales que han repercutido directamente en la proliferación de propuestas basadas en DL [Goodfellow et al. (2016)]:

- El aumento en la cantidad de datos disponibles. Este factor es fundamental ya que muchos de los modelos basados en aprendizaje profundo requieren un gran número de instancias para llevar a cabo el proceso de entrenamiento. Esto es debido a la creciente complejidad de los modelos y, por consiguiente, a la necesidad de grandes cantidades de datos para ajustar todos los parámetros existentes en los mismos para obtener buenos resultados. En muchos casos, la eficacia de este tipo de propuestas está relacionada con la cantidad de datos utilizados para entrenarlas.
- Las infraestructuras tanto software como hardware han mejorado considerablemente. Este hecho implica que haya sido posible el desarrollo y ejecución de arquitecturas de DL cada vez más complejas, lo que repercute en la obtención de mejores resultados. No obstante, las limitaciones hardware siguen estando presentes, por lo que su evolución está estrechamente ligada con el desarrollo de nuevas propuestas de aprendizaje automático.

Un hecho que pone de manifiesto la gran relevancia que tiene el área de DL en la actualidad es la gran cantidad de modelos que surgen dentro de este ámbito. Algunas de las propuestas más conocidas y usadas son: *Deep Belief Networks* (DBN) [Hinton et al. (2006)], *Convolucional Neural Networks* (CNN) [LeCun et al. (1989)], *AutoEncoders* (AE) [Olshausen y Field (1996)], *Restricted Boltzmann Machines* (RBM) [Smolensky (1986)], *Recurrent Neural Networks* (RNN) [Graves et al. (2013)], *Long Short-Term Memory* (LSTM) [Hochreiter y Schmidhuber (1997)], *Gated Recurrent Units* (GRU) [Chung et al. (2015)], *Generative Adversarial Networks* (GAN) [Goodfellow et al. (2014)], *Deep Residual Networks* (DRN) [He et al. (2016)], *Extreme Learning Machines* (ELM) [Huang et al. (2011)]. Durante el desarrollo de esta tesis se pondrá el foco en la utilización

de algunas de las técnicas anteriores, en concreto AE, para afrontar la tarea de reducción de dimensionalidad, así como el análisis de los efectos del desbalanceo en otro de los modelos más usados, CNN.

En principio, los modelos de DL se basan fundamentalmente en dos conceptos: redes neuronales artificiales [Schalkoff (1997)] y algoritmo de *back-propagation* [Rumelhart et al. (1986)]. Un ejemplo básico de algoritmo de DL puede ser un MLP [Hornik et al. (1989)] que asocie unos valores de entrada a unos valores de salida a través de una serie de capas ocultas. Esta asociación final se forma mediante la composición de una serie de funciones matemáticas simples, cada una de las cuales puede ser considerada una representación más sencilla de la entrada. En la Sección 2.4.1 se profundizará sobre los conceptos básicos asociados al DL.

Existe un gran número de áreas de investigación donde se han aplicado las técnicas de DL, dado el importante auge que han experimentado. La Sección 2.4.2 recoge algunas de las propuestas más relevantes realizadas en diferentes ámbitos de aplicación.

Desde un punto de vista metodológico, las técnicas basadas en DL se basan en el aprendizaje a través de la experiencia. Así, el propio modelo parte de conceptos sencillos para generalizar información cada vez más compleja y de más alto nivel. Las técnicas de DL se basan en ANN de arquitecturas profundas cuyo objetivo fundamental es generalizar información de alto nivel a partir de los datos analizados. El factor diferencial de los modelos DL con respecto a cualquier otro modelo de aprendizaje automático está en la capacidad de este tipo de técnicas para generar diversas representaciones del espacio de entrada. Por esta razón, las principales líneas de investigación en esta área se centran en obtener dichas representaciones de los datos de entrada y desarrollar modelos que las aprendan.

La creación de una nueva representación de los datos de entrada o, en otras palabras, la tarea de extracción de características de alto nivel de los datos llevada a cabo por parte de los diferentes modelos de aprendizaje automático es fundamental para que puedan afrontar determinados tipos de tareas que los humanos realizamos de forma intuitiva. En este contexto, surge la metodología conocida como *representation learning* (aprendizaje de nuevas representaciones), descrita en la Sección 2.2.1, cuyo objetivo es obtener una representación de los datos de entrada que permita al modelo generalizar relaciones de alto nivel entre dichos datos [Bengio et al. (2013)].

Existen diferentes técnicas de DL que han mostrado un buen rendimiento a la hora de afrontar el aprendizaje de representaciones. Estas técnicas producen representaciones que llevan asociado un rendimiento mucho mayor que las extraídas de forma manual. Así mismo, estos modelos pueden adaptarse fácilmente a nuevos datos, sin necesidad de intervención externa.

El modelo típicamente usado para *representation learning* es el AE [Olshausen y Field (1996)]. El funcionamiento de este tipo de modelos permite extraer ciertas características interesantes de los datos originales y utilizar dicha información como nueva representación de los mismos. Las propiedades de esta codificación dependerán en gran medida del tipo de AE utilizado, en la Sección 2.5 se profundizará en este concepto, así como se presentarán algunos de los tipos de AE existentes.

Finalmente, la Sección 2.6 introduce los conceptos teóricos asociados a las CNN [LeCun et al. (1989)]. Este modelo de DL es uno de los más utilizados, tal y como se comentó en el Capítulo 1. Por esta razón, esta tesis doctoral plantea el estudio de determinados factores, en concreto el desbalanceo de los datos, sobre este modelo de clasificación.

2.4.1. Características de las técnicas *Deep Learning*

Esta metodología, como su nombre sugiere, está basada en arquitecturas multi-capa (profundas), que son usadas para asociar las características propias de los datos de entrada con la salida esperada [Bengio (2009) y Goodfellow et al. (2016)]. El término profundo (*deep*) también hace referencia al hecho de que los modelos DL generalizan relaciones complejas a partir de conceptos sencillos de forma jerárquica. De esta forma, estas técnicas aprenden múltiples representaciones, que se corresponden con diferentes niveles de abstracción. Algunas de las ventajas de este tipo de modelos son:

- Los modelos DL aprenden nuevas representaciones de los datos por sí mismos, no de forma externa como los modelos tradicionales [Deng (2014)].
- El comportamiento, en términos de rendimiento predictivo, de este tipo de algoritmos mejora considerablemente al de métodos tradicionales en ciertos campos de aplicación como, por ejemplo, el reconocimiento de imágenes [Ciresan et al. (2012a)], sonido [Hinton et al. (2012)] o detección del fraude [Pumsirirat y Yan (2018)].

- Este tipo de técnicas obtienen mejoras en términos de tiempo con respecto a algoritmos tradicionales que afrontan tareas propias de la ingeniería de características, algunas de las cuales son muy costosas computacionalmente [Goodfellow et al. (2016)].
- Los modelos basados en DL tienen una alta capacidad de adaptación cuando utilizan nuevos datos, así como para tratar problemas relacionados [Bengio (2009)].

Estas ventajas se generan debido a las diferencias de los algoritmos de aprendizaje profundo con respecto a metodologías tradicionales. El DL, como se ha indicado, basa el aprendizaje en una serie de estructuras que proporcionan una jerarquía de conceptos, desde los más simples a los más complejos y abstractos [Goodfellow et al. (2016) y Deng (2014)]. Los métodos tradicionales, por su parte, se basan en procesos lineales que implican la realización de varias etapas diferenciadas. La Figura 2.10 compara de forma gráfica la metodología tradicional con la basada en DL.

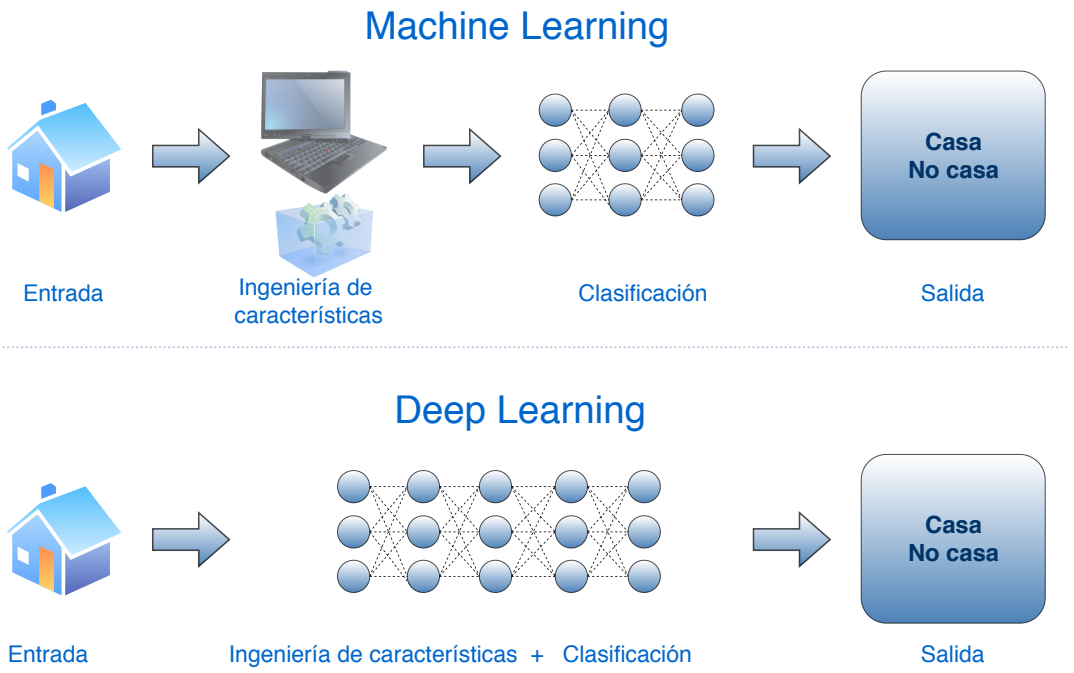


Figura 2.10. *Deep Learning* frente a *machine learning* tradicional.

Sin embargo, uno de los principales inconvenientes que ha influido en que los modelos DL no hayan experimentado un auge hasta ahora está asociado a la cantidad de datos que necesitan. Las técnicas DL requieren grandes conjuntos de datos para obtener unos niveles de rendimiento aceptables, mientras que las técnicas de aprendizaje automático tradicionales necesitan una cantidad menor para generar resultados aceptables. Así mismo, los modelos DL

aumentan su rendimiento a medida que se agregan más datos, mientras que en los métodos tradicionales se produce convergencia, es decir, llegado un punto no se aumenta el rendimiento a pesar de añadir más datos.

Actualmente, debido a los buenos resultados que ofrecen en distintos ámbitos de aplicación [Deng (2014) y Bengio (2013)], continuamente surgen nuevas propuestas que utilizan técnicas DL para afrontar problemas muy diversos. No obstante, los primeros modelos basados en DL surgieron en los años 80 y 90 [Smolensky (1986) y Olshausen y Field (1996)], pero su relevancia ha aumentado considerablemente en los últimos tiempos. En la Figura 2.11 se representa de forma esquemática el año de aparición de algunos de las propuestas más relevantes dentro del área del DL.

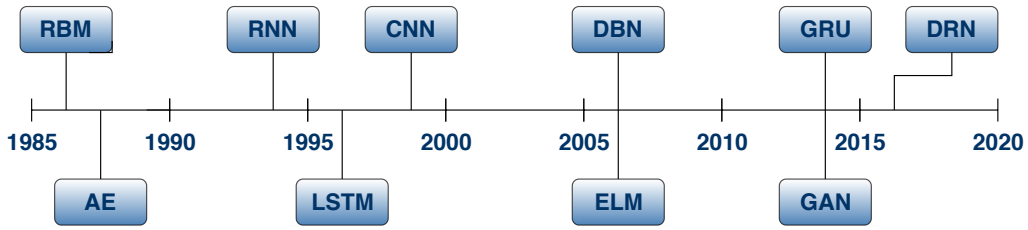


Figura 2.11. Año de aparición de algunas de las principales arquitecturas *Deep Learning*: RBM, AE, RNN, LSTM, CNN, DBN, ELM, GRU, GAN, DRN.

Las propuestas presentadas en la Figura 2.11 son, como se ha indicado, algunas de las más conocidas y utilizadas. A continuación, se explican brevemente cada una de ellas:

- **RBM** [Smolensky (1986)]: se trata de un tipo de red neuronal artificial cuyo objetivo es aprender la probabilidad de distribución asociada a los datos de entrada. Desde un punto de vista estructural, este tipo de modelos está formado por una serie de unidades binarias estocásticas y conexiones no dirigidas entre ellas. Las RBM surgieron para superar las limitaciones que tienen las Bolzmann Machines [Hinton y Sejnowski (1983)] a la hora de escalarse, es decir cuando el tamaño del modelo aumenta considerablemente. Por esta razón, las RBM incorporan restricciones en las conexiones entre las unidades que componen las distintas capas ocultas. Estas condiciones dan lugar a estructuras cuyo proceso de aprendizaje es más eficiente. Existen múltiples aplicaciones de las RBM, algunas de ellas son: clasificación [Larochelle y Bengio (2008)], aprendizaje de características [Coates et al. (2011b)] o reducción de dimensionalidad [Hinton y Salakhutdinov (2006)].

- **DBN** [Hinton et al. (2006)]: una RBM por sí misma tiene limitaciones en cuanto a su capacidad de representación. La verdadera potencia de estos modelos surge cuando varios componentes basados en RBM se apilan para formar un modelo más profundo y complejo. En esta idea se basan las DBN propuestas por Hinton en 2006 [Hinton et al. (2006)]. La estructura de este tipo de modelos está formada, normalmente, por múltiples capas compuestas por un conjunto de unidades. Dos capas adyacentes comparten una serie de conexiones entre ellas, pero no se conectan dos unidades pertenecientes a una misma capa entre sí. El entrenamiento de este modelo consiste en entrenar vorazmente cada capa de forma consecutiva mediante aprendizaje supervisado [Hinton et al. (2006)]. Este proceso supuso un punto de inflexión en el desarrollo de modelos DL. Las aplicaciones de esta técnica son muy variadas: clasificación [Lee et al. (2009b)], aprendizaje de representaciones [Lee et al. (2009a)], procesamiento del lenguaje natural [Sarikaya et al. (2014)] y tratamiento de audio [Mohamed et al. (2012)], entre otros.

- **CNN** [LeCun et al. (1989)]: este tipo de redes neuronales artificiales se basan en el proceso llevado a cabo por los humanos y animales a la hora de visualizar e identificar los objetos. Fundamentalmente, estas redes se basan en la correlación espacial existente en los datos de entrada. Su estructura está formada por una serie de capas compuestas por un conjunto de neuronas. Sin embargo, cada una de estas unidades únicamente se conecta con una región concreta de la capa anterior. De esta forma, el ajuste de cada neurona vendrá determinado por los valores de sus elementos adyacentes y no por todos los componentes de la entrada. Esta metodología hace que las CNN sean muy efectivas a la hora de afrontar tareas como clasificación de imágenes [Krizhevsky et al. (2012)], reconocimiento de vídeo [Karpathy et al. (2014)] o tratamiento del lenguaje natural [Hu et al. (2014)]. Durante el desarrollo de la presente tesis doctoral, se han estudiado diferentes aspectos relacionados con las CNN, en concreto cómo afecta el desbalanceo de los datos de entrada cuando estos modelos se usan en clasificación. Por ello, los conceptos relacionados con las CNN serán ampliados en la Sección 2.6.

- **AE** [Olshausen y Field (1996)]: se trata de redes neuronales artificiales caracterizadas por poseer una estructura simétrica. El principal objetivo que persiguen estos modelos es reconstruir la entrada en la salida de la red. Los AE aprenden por sí mismos, a partir de los atributos de entrada

sin necesidad de etiquetas, a realizar dicho proceso. No obstante, es importante incluir ciertas restricciones en la red para evitar que el modelo se limite a copiar la entrada en la salida. De esta forma, se pueden obtener diferentes tipos de AE que varían en distintos aspectos, como la arquitectura o el proceso de aprendizaje. Debido a su estructura y funcionamiento, los AE son modelos especialmente apropiados para afrontar la tarea de reducción de dimensionalidad. Si dichas capas ocultas son de menor dimensionalidad que la entrada y la salida, la codificación obtenida puede utilizarse como una nueva representación de la entrada con menor dimensión [Charte et al. (2018), Yang et al. (2018) y Fan (2019)]. Por ello, esta técnica ha sido utilizada a lo largo de esta tesis y será estudiada con mayor detalle en la Sección 2.5.

- **RNN** [Graves et al. (2013)] **con LSTM** [Hochreiter y Schmidhuber (1997)]: este tipo de modelos permiten que la información manejada persista en el tiempo. De esta forma, el comportamiento se acerca al pensamiento humano, donde los razonamientos están relacionados entre sí. Para hacerlo, las RNN incluyen una serie de conexiones, que permiten la retroalimentación del modelo. En general, las RNN pueden verse como un conjunto de réplicas de una misma red conectadas entre sí de forma secuencial. En este contexto, las LSTM pueden presentarse como un tipo de RNN con una serie de características especiales. La peculiaridad de las LSTM es que incluyen unas unidades más complejas. Estas neuronas están formadas por un mayor número de componentes, entre los que destacan las llamadas puertas (*gates*) de configuración, las cuales pueden ser de distinto tipo según su función: *input gate* que controla el flujo de entrada, *forget gate* encargada de gestionar la información que es olvidada por la unidad, *peephole gate* que permite a la neurona disponer información sobre su propio estado y *output gate* que determina el flujo de salida, entre otros. Estos modelos han obtenido grandes resultados en múltiples campos de aplicación, algunos de los cuales son: traducción [Auli et al. (2013)], reconocimiento de voz [Graves et al. (2013)], generación de imágenes [Toderici et al. (2017)] o creación de subtítulos de forma automática [Venugopalan et al. (2015)].
- **GRU** [Cho et al. (2014)]: este tipo de red surgió para mitigar los efectos del problema asociado a la degradación del gradiente que tiene lugar en las RNN tradicionales. El diseño de este modelo está muy relacionado con las LSTM descritas anteriormente y los resultados obtenidos son

similares. En este sentido, al igual que las LSTM, las GRU tienen una arquitectura más compleja que la RNN tradicional. Este hecho permite que los valores de la red sean permanentes en el tiempo, pudiendo tener relevancia en el futuro para actualizar el estado del modelo. Para llevar a cabo el proceso de entrenamiento, estos modelos tienen dos funcionalidades características: *update gate*, que determina qué información se mantienen en el tiempo, y *reset gate*, que decide lo que la red debe olvidar [Chung et al. (2015)]. Esta técnica ha ofrecido resultados relevantes en muchos campos, algunos de ellos son: reconocimiento de audio [Ravanelli et al. (2018)], clasificación [Tang et al. (2015)] o segmentación de vídeo [Siam et al. (2017)].

- **GAN** [Goodfellow et al. (2014)]: se trata de modelos cuya arquitectura está compuesta por dos redes que se enfrentan entre sí (*adversarial*). En concreto, una de las redes utilizadas, conocida como generador, se encarga de la creación de nuevas instancias de datos, mientras que la otra red, conocida como discriminador, tiene el objetivo de determinar su autenticidad mediante la evaluación de diferentes factores. De esta forma, la labor del discriminador consiste en determinar qué instancias pertenecen al conjunto de datos real, para ello deberá extraer ciertas características que le ayuden a realizar dicha selección. Estos algoritmos tienen un gran potencial, ya que su funcionamiento les permite generar cualquier distribución de datos. Esto implica que este tipo de modelos sean capaces de crear nueva información similar a la obtenida en dominios reales. En este sentido, existen resultados sorprendentes que ponen de manifiesto el buen comportamiento de estos modelos en campos como: la generación de imágenes [Xu et al. (2018)], mejora del habla [Donahue et al. (2018)] o creación de música [Chen et al. (2018)], entre otros.

- **DRN** [He et al. (2016)]: es un tipo de red neuronal cuyo funcionamiento se basa en las células piramidales presentes en el cerebro humano. Fundamentalmente, las DRN utilizan bloques residuales que preservan la información mediante conexiones entre capas no adyacentes del modelo. De esta forma, el flujo de información no sólo va de una capa a la siguiente, sino que, en ocasiones, se salta algunas de ellas durante el proceso. Las técnicas tradicionales incluyen conexiones que implican saltos de doble o triple capa [He et al. (2016)]. Este proceso ayuda a que la red pueda preservar la información más relevante a lo largo de su

arquitectura y evitar el problema de la degradación del gradiente. Esto hace que el aprendizaje se realice de forma más rápida, pero explorando menos el espacio de características. Por ello, esta metodología es más vulnerable al ruido o las posibles perturbaciones que puedan ocurrir, siendo necesarios más datos de entrenamiento para ajustar adecuadamente los parámetros del modelo. No obstante, estas redes neuronales profundas más sofisticadas permiten llevar a cabo tareas complejas, obteniendo buenos resultados en múltiples campos como el reconocimiento de imágenes [He et al. (2016)], sonido [Tan et al. (2018)] o clasificación [Wang et al. (2017)].

- **ELM** [Huang et al. (2006)]: se trata de un tipo de red neuronal que surgió para mitigar algunos de los problemas tradicionales de las ANN, entre los que destaca la velocidad de aprendizaje. Este factor supone un cuello de botella en muchas aplicaciones y se debe fundamentalmente a los procesos de aprendizaje basados en la adaptación del gradiente y a la necesidad de ajustar un gran número de parámetros de forma iterativa. Las ELM tratan de solventar estos problemas, para ello el aprendizaje se realiza sin necesidad de entrenar de forma iterativa todos los nodos de la red. Durante este proceso algunas de las neuronas se actualizan con un valor al azar, heredan valores de sus antepasados de forma directa o no son actualizadas [Huang et al. (2011)]. De esta forma, en la mayoría de lo casos los pesos de salida se aprenden en un sólo paso, lo que equivale a aprender un modelo lineal. Las propuestas basadas en ELM no sólo reducen en gran medida el tiempo de entrenamiento, sino que logran resultados relevantes en múltiples áreas de investigación, algunas de ellas son: clasificación y regresión [Huang et al. (2012)], optimización [Huang et al. (2010)] o desbalanceo [Zong et al. (2013)].

Los modelos anteriores afrontan, como se ha indicado, tareas muy diversas. En la siguiente sección se presentan algunas de las aplicaciones y resultados más relevantes de las técnicas DL. Así mismo, en las Secciones 2.5 y 2.6 se profundiza en los principales modelos de DL utilizados a lo largo de los distintos estudios llevados a cabo durante la elaboración de la presente tesis: AE y CNN, respectivamente.

2.4.2. Aplicaciones del *Deep Learning*

Un gran número de modelos DL fueron desarrollados, como se ha indicado en la sección anterior, en la década de los 90, algunos de los conocimientos teóricos asociados incluso antes. No obstante, ha sido en los últimos años cuando las distintas metodologías de DL han despertado un creciente interés. El aumento en la utilización de este tipo de técnicas se ha producido dada la gran aplicación que tienen estos modelos en la resolución de problemas reales. A continuación, se presentan algunas de las áreas en las que se han aplicado con éxito diferentes técnicas de DL:

- **Clasificación de imágenes y vídeos:** se trata de una de las primeras tareas donde los métodos de DL han obtenido resultados realmente relevantes, comenzado a despertar interés en el resto de la comunidad científica. Las arquitecturas de DL han mostrado su eficiencia a la hora de clasificar imágenes [Ciresan et al. (2012a), Ciresan et al. (2012b) y Krizhevsky et al. (2012)], así como identificar determinados objetivos en vídeos [Karpathy et al. (2014) y Yue-Hei Ng et al. (2015)], mejorando ampliamente a muchas de las técnicas tradicionales.
- **Reconocimiento de voz:** esta tarea tiene como objetivo permitir la comunicación entre una persona y un equipo informático, para ello es necesario que el ordenador sea capaz de reconocer el lenguaje hablado. En este sentido, existen diferentes modelos DL que han sido aplicados a la resolución de dicha tarea [Deng y Yu (2011), Hinton et al. (2012) y Ravanelli et al. (2018)], obteniendo importantes mejoras con respecto a algoritmos clásicos.
- **Reconocimiento de texto:** se tratan de aplicaciones cuyo objetivo fundamental es la extracción y el análisis de la información textual, normalmente dicho contenido se presenta en forma de imagen. Uno de los enfoques tradicionales para afrontar esta tarea son las ANN, por ello el auge de los métodos de DL ha tenido una gran repercusión a la hora de afrontar esta tarea [Wang et al. (2012)].
- **Traducción automática:** esta aplicación hace referencia a la traducción de forma automática tanto de texto escrito como de audios, por lo que, en cierta medida, implica a las dos tareas que se acaban de describir, ya que en primer lugar es necesario realizar un reconocimiento de dicha

información. En este sentido, han surgido propuestas basadas en DL que afrontan la traducción automática de forma simultánea obteniendo resultados relevantes [Bahdanau et al. (2014) y Good (2015)].

- **Asistentes digitales:** son sistemas cuyo objetivo es ayudar a las personas a realizar tareas con equipos informáticos con la mínima interacción posible, involucrando una comunicación natural y fluida, en general, por medio de la voz. En este sentido, los avances que el uso de las técnicas DL han implicado en otros campos han contribuido a que la utilidad de los asistentes virtuales aumente considerablemente, realizando un mejor procesamiento del lenguaje y generando respuestas más sofisticadas [Sarikaya (2017)].
- **Tratamiento de imagen:** las técnicas DL han mostrado buenos resultados a la hora de afrontar un gran número de tareas relacionadas con el tratamiento de imágenes, no sólo la clasificación la cual se ha descrito anteriormente. En este contexto, son numerosas las propuestas que han surgido para afrontar tareas muy diversas como pueden ser: eliminación del ruido [Bako et al. (2017)], coloreado automático [Zhang et al. (2016)], transferencia de estilos entre imágenes [Gatys et al. (2015)] o segmentación [Badrinarayanan et al. (2017)], entre otras.
- **Aplicaciones médicas:** se trata fundamentalmente de software aplicado al campo de la medicina. Este es uno de los campos tradicionales de aplicación del aprendizaje automático y se han generado soluciones basadas en métodos tradicionales que afrontan diferentes problemas [Kononenko (2001)]. En este sentido, las técnicas DL han abierto nuevas posibilidades de aplicación y se han desarrollado un gran número de propuestas. Fundamentalmente, las soluciones aportadas en este campo explotan los beneficios de utilizar arquitecturas profundas para tratar imágenes médicas [Litjens et al. (2017) y Singh et al. (2019)], obteniéndose un gran rendimiento con respecto a metodologías tradicionales.
- **Marketing y publicidad:** este sector es uno de los más importantes económicamente dentro de muchas empresas, dado los beneficios que extraen de su explotación. Desde el punto de vista de la aplicación de técnicas DL, existen diferentes tareas que han sido analizadas y afrontadas con este tipo de métodos, por ejemplo, la generación de publicidad en tiempo real adaptada a las necesidades de cada usuario [Zhao et al. (2018)].

- **Detección de fraude bancario:** es un área de investigación orientada a analizar las transacciones que se producen de forma continua en las sucursales bancarias con el objetivo de detectar posibles movimientos anómalos que podrían corresponderse con actuaciones fraudulentas. Este tipo de tareas destaca, entre otras cosas, por utilizar datos excesivamente desbalanceados, debido al desequilibrio entre transacciones correctas y anómalas [Wei et al. (2013)]. En este contexto, han surgido diferentes propuestas basadas en DL que permiten realizar detecciones de transacciones fraudulentas de forma más eficiente que las técnicas tradicionales [Gómez et al. (2018), Pumsirirat y Yan (2018) y Zheng et al. (2018)].

- **Seguridad digital:** hace referencia a tareas orientadas a la protección tanto de equipos informáticos como de perfiles virtuales ante cualquier tipo de actividad no autorizada. En los últimos años han surgido diferentes propuestas basadas en DL que afrontan este tipo de problemas, destacando las soluciones orientadas a detección de *malware* [Yuan et al. (2014), HaddadPajouh et al. (2018) y Li et al. (2018)].

- **Ciudades inteligentes**: esta área de investigación ha sido un área emergente en los últimos años, debido a que propone un desarrollo basado en la sostenibilidad y la mejora de la calidad de vida de las personas. En este sentido, las herramientas tecnológicas juegan un papel fundamental, ya que contribuyen a la realización de un gran número de tareas. Dentro de este campo se han propuesto diferentes soluciones basadas en modelos de aprendizaje profundo [Polishetty et al. (2016), He et al. (2017) y Mohammadi et al. (2018)].

- **Domótica**: se trata de técnicas y tareas orientadas a la automatización de una vivienda, incluye la seguridad, las comunicaciones o el control energético, entre otras. Algunas de las propuestas basadas en DL que han surgido en relación con esta área tienen como objetivo la gestión de edificios inteligentes [Manic et al. (2016)], la predicción del comportamiento humano [Choi et al. (2013)] o automatización de tareas domésticas [Cowdrey y Malekian (2018)], entre otros.

- **Conducción autónoma:** el objetivo fundamental de este campo es asistir a un humano a la hora de conducir un vehículo, dicha asistencia tiene diferentes grados según la necesidad de actuación humana, considerando a los vehículos totalmente automatizados aquellos que llevan a cabo

la conducción sin ningún tipo de apoyo. En este caso, también existen diferentes propuestas basadas en DL que afrontan tareas relacionadas [Bojarski et al. (2016b) y Tian et al. (2018)].

- **Búsqueda Web:** los buscadores web evolucionan continuamente para ofrecer a sus usuarios los mejores resultados de acuerdo a sus criterios de búsqueda. En este sentido, los avances en DL también han contribuido a dicha evolución, surgiendo propuestas que afrontan dicha tarea [Shen et al. (2014) y Xu et al. (2018)] cuyo objetivo es optimizar dichos procesos de búsqueda de información.
- **Procesamiento del lenguaje natural:** esta área de investigación tiene como objetivo la comunicación efectiva entre las personas y los sistemas informáticos, mediante el uso del lenguaje habitualmente empleado por los primeros. Existen diferentes tipos de información a procesar, entre los que destacan la textual, la hablada o el lenguaje de signos. Este campo incluye a algunas de las áreas mencionadas anteriormente, como reconocimiento de voz o texto, pero también involucra otras tareas en las que se han propuesto soluciones basadas en DL, como por ejemplo, identificación de sentimientos [Deng et al. (2014)] o desambiguación [Zuheros et al. (2019)], entre otras.
- **Juegos:** en los últimos años han tenido gran relevancia diferentes propuestas basadas en DL que han mostrado un rendimiento superior a los humanos en determinados juegos. Fundamentalmente, estos sistemas aprenden por sí mismos a través de las sucesivas partidas, llegando a niveles realmente elevados. Uno de los resultados más relevantes en este sentido es el programa AlphaGo que fue la primera máquina que logró vencer a un profesional en el juego Go [Chen (2016)]. No obstante, existen otras muchas propuestas orientadas tanto a juegos tradicionales [David et al. (2016) y Moravčík et al. (2017)] como a videojuegos [Mnih et al. (2015) y Van Hasselt et al. (2016)]

Esta tendencia relacionada con el creciente interés en el desarrollo de propuestas basadas en DL no parece que vaya a detenerse. Así mismo, la proliferación de plataformas software que facilitan la creación de este tipo de modelos, como *TensorFlow*, *Theano* o *Keras*, entre otras, implica que cada vez más personas tengan la posibilidad de desarrollarlos, sin necesidad de disponer de un conocimiento profundo sobre los detalles de bajo nivel asociados

a los mismos. Por todo ello, en el futuro las técnicas de DL pueden ayudar a resolver problemas que hoy no parecen accesibles desde la perspectiva del aprendizaje automático.

No obstante, es importante indicar que las técnicas DL también presentan una serie de limitaciones que es necesario conocer a la hora de utilizar este tipo de métodos. Una de las restricciones típicas de las ANN en general y de los modelos DL en particular es la falta de interpretabilidad del proceso que llevan a cabo, es decir, su funcionamiento es visto como una caja negra que es difícil de comprender. Así mismo, las técnicas actuales de DL no son capaces de afrontar tareas que requieran la manipulación algorítmica de datos dada su naturaleza. Otro aspecto negativo es que es posible manipular la salida de la red si se conoce su estructura, mediante modificaciones mínimas de la entrada [Papernot et al. (2016)]. Este hecho puede suponer una brecha de seguridad cuando se utilizan este tipo de modelos.

2.5. *Autoencoders*

El objetivo fundamental de esta sección es profundizar en el concepto de AE, describiendo su arquitectura y funcionamiento básico, las diferentes categorizaciones que existen y algunos de los modelos más conocidos y utilizados, en particular, los más relevantes para el desarrollo de los estudios relativos a la presente tesis doctoral.

Un AE, tal y como se ha descrito en la Sección 2.4.1, es una red neuronal artificial capaz de aprender nuevas representaciones de los datos de entrada mediante un proceso de aprendizaje no supervisado [Charte et al. (2018)]. De esta forma, los AE aprenden utilizando únicamente la información relativa a los atributos de entrada, sin necesidad de ningún tipo de etiquetado o procesamiento previo. La estructura propia de este tipo de modelos es simétrica y su objetivo fundamental es aprender a reconstruir la entrada en la salida. Para evitar que la red se limite a copiar la información de la entrada, estos modelos incluyen ciertas restricciones que deben satisfacer, ya sea a nivel de estructura o a nivel funcional. Así, esta técnica permite generar una nueva representación de los datos de entrada en las capas ocultas de la red que contiene información de más alto nivel que la original y que cumple ciertas características, según la estructura y parametrización propia de cada modelo. A pesar de que el concepto más utilizado hoy en día para hacer referencia a

este modelo es *autoencoder*, también se han empleado otros términos como *diabolo networks* [Schwenk y Bengio (1998)], redes neuronales replicadoras [Hecht-Nielsen (1995)] o autoasociativas [Kramer (1991)].

Los AE son muy utilizados en la actualidad debido a su arquitectura y funcionamiento para afrontar diferentes tareas entre las que destaca, por su interés para esta tesis, la reducción de dimensionalidad [Hinton y Salakhutdinov (2006), Rifai y Muller (2011) y Wang et al. (2014)]. No obstante, para ello, la arquitectura propia del modelo debe cumplir ciertas restricciones. En concreto, la alternativa más habitual para obtener una codificación de menor tamaño que la original requiere que la capa intermedia deba tener menor dimensionalidad que la entrada y la salida. De esta forma, la nueva representación generada en dicha capa intermedia contiene información con un nivel de abstracción y generalización mayor que los datos originales, que puede ser extraída y utilizada en etapas posteriores, por ejemplo, como entrada de algoritmos tradicionales de clasificación.

Desde un punto de vista general, la estructura básica de un AE se corresponde con una red neuronal prealimentada (*feedforward*), donde no existen ciclos, es decir, la información siempre fluye en la misma dirección [Hornik et al. (1989)]. La arquitectura de los AE está formada por una secuencia de capas: una capa de entrada, un conjunto de capas ocultas y una capa de salida, donde cada una de las neuronas que componen cada capa está conectada con todas las unidades de la capa siguiente. Además, la cantidad de capas ocultas puede ser variable, si es mayor que uno normalmente las capas intermedias se estructuran de forma simétrica [Bengio (2009)]. Dado que el objetivo principal es reproducir la entrada en la salida, tanto la capa de inicial como la final deben tener el mismo número de elementos. La Figura 2.12 presenta la arquitectura típica de un AE.

En este caso, el AE tiene cinco capas, tres de ellas ocultas. Así mismo, existen dos bloques: por un lado la parte de codificación (*encoder*) que está compuesta por las tres primeras capas, incluyendo la capa central; por otro lado la parte de descodificación (*decoder*) que comienza en la capa central, agrupando las tres capas finales. La capa central que es compartida por los dos bloques se conoce como capa de codificación. Además, se puede observar que cada neurona está conectada con todas las unidades de la capa siguiente, estando dichas conexiones ponderadas mediante pesos, denotados por w_i en la figura.

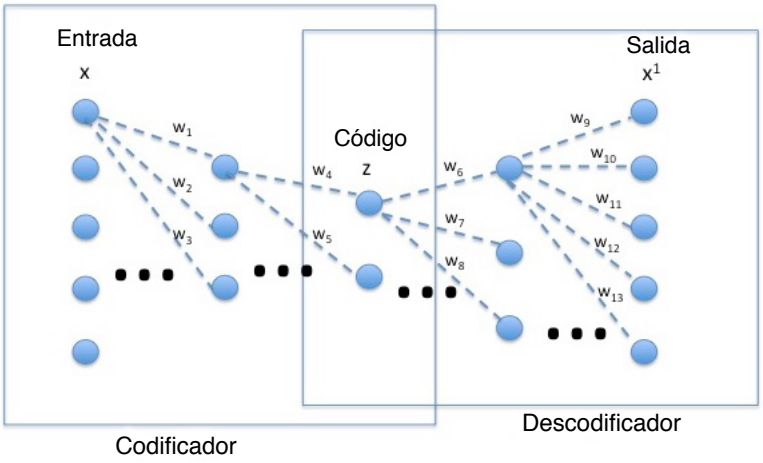


Figura 2.12. Arquitectura de un AE con cinco capas.

Los AE pueden tener tantas capas ocultas como sea necesario, en general dispuestas de forma simétrica. En este contexto, la capa de interés puede variar según el problema que se desee resolver. De esta forma, si el objetivo es reducir la dimensionalidad, las capas ocultas, en concreto, la capa central, serán la que proporcionen dicha codificación. En cambio, si el objetivo es eliminar el ruido presente en la entrada, la capa de interés será la capa de salida que contiene la reconstrucción de la entrada. Estos son sólo dos ejemplos de posibles tareas que se pueden afrontar con los AE y que ponen de manifiesto la importancia de las distintas capas según el contexto.

En este sentido, la presente tesis doctoral se centrará en afrontar la tarea de reducción de dimensionalidad mediante AE. Por ello, la codificación aprendida en las capas ocultas de los AE adquieren una mayor importancia. No obstante, para lograr el objetivo perseguido, se deben incluir ciertas restricciones en la arquitectura de la red. Así, el modelo será capaz de aprender, por sí mismo, una representación interna de la entrada. En este punto, es importante destacar que existen diferentes tipos de AE según su arquitectura. En concreto, teniendo en cuenta la cantidad de capas que forman el modelo se puede considerar la siguiente tipología.

- ***Shallow***: se corresponde con los modelos más sencillos posibles, formados únicamente por tres capas: entrada, una capa oculta y salida. En este caso, la capa oculta contiene la codificación aprendida por el modelo.
- ***Deep***: se trata de modelos más complejos, formados por más de una capa oculta. Generalmente, estas capas están dispuestas de forma simétrica, siendo la capa central la que genera la codificación de interés.

Además de la tipología anterior, los AE pueden clasificarse según el número de elementos contenidos en la capa de codificación. En este sentido, considerando este factor se tiene:

- ***Undercomplete***: el número de neuronas de la capa de codificación es menor que el número de elementos de la entrada (y salida). Este tipo de arquitecturas fuerza a la red a aprender una nueva representación comprimida y codificada del espacio de características original. Este modelo es típicamente usado para afrontar la fusión de características, donde nuevas variables de un nivel más alto de abstracción son generadas.
- ***Overcomplete***: el tamaño de la capa de codificación es mayor que el de la capa de entrada (y salida). Este tipo de modelos se pueden limitar a copiar la entrada en la salida a través de las capas intermedias sin aprender nada de utilidad, dado que son de mayor dimensionalidad. Por tanto, es necesario incluir ciertas restricciones para impedir esto [Olshausen y Field (1997) y Deng et al. (2014)]. Por ejemplo, es posible forzar que sólo un conjunto limitado de neuronas de las capas intermedias se activen. El objetivo de este tipo de modelos es aprender una representación dispersa del espacio de entrada original.

Este trabajo pone el foco en la reducción de dimensionalidad mediante modelos DL. Por esta razón, nos centraremos en el tipo de AE conocido como *undercomplete*. Este tipo de modelos permiten afrontar dicha tarea, ya que son capaces de aprender nuevas representaciones del espacio de entrada original, que se obtienen mediante una fusión de características originales que produce una codificación con un mayor nivel de abstracción y una menor dimensionalidad.

En la literatura existen diferentes modelos de AE que pueden ser utilizados para afrontar la reducción de dimensionalidad. No obstante, no existen estudios que, de acuerdo a determinados factores relacionados con el contexto del problema, faciliten la elección del modelo más apropiado a la hora de realizar esta tarea. Por ello, durante el desarrollo de esta tesis, además de proponer nuevos modelos basados en AE para reducir la dimensionalidad, se ha llevado a cabo un estudio exhaustivo con el objetivo de determinar qué modelo de AE se adapta mejor según el contexto del problema. Los modelos de AE utilizados en dicho estudio son algunos de los más conocidos y utilizados en la literatura, a continuación se describe cada uno de ellos, comenzando con el modelo básico de AE que constituye la base sobre la que se construyen el resto de propuestas.

2.5.1. Modelos de *Autoencoders*

Tal y como se ha indicado anteriormente, la tarea de fusión de características puede ser afrontada usando diferentes aproximaciones basadas en AE. A continuación, se describen algunas de estas propuestas. En concreto, se presentan cuatro de los modelos más utilizados y conocidos: básico, *contractive*, *denoising* y *robust*. Sin embargo, es importante indicar que existen otras variantes en la literatura. De hecho, dada la relevancia de este tipo de modelos, surgen nuevas propuestas continuamente. Algunas de estas son: *Variational AE* [Kingma y Welling (2013)], *Convolutional AE* [Masci et al. (2011)], *LSTM AE* [Hou et al. (2019)], *Sparse AE* [Ng (2011)], *Adversarial AE* [Makhzani et al. (2016)], *Correspondence AE* [Feng et al. (2014)], *AE Node Saliency* [Fan (2019)] y *AE con funciones invertidas* [Yang et al. (2018)].

2.5.1.1. *Autoencoder* Básico

El AE básico (BAE) [Bourlard y Kamp (1988)] es el modelo más sencillo y que constituye la base para el desarrollo del resto de variaciones existentes. La arquitectura básica de un BAE se corresponde con la que se ha presentado anteriormente, es decir se trata de una red neuronal prealimentada cuya estructura de capas es simétrica. No obstante, esta simetría no tiene que verse reflejada en los pesos y funciones de activación propias de cada una de las neuronas. La Figura 2.13 refleja la arquitectura de un BAE.

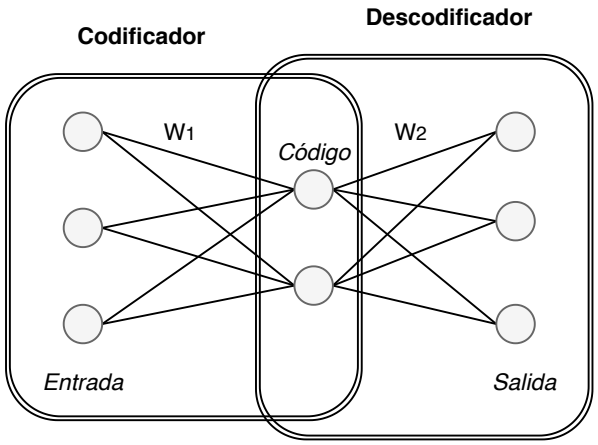


Figura 2.13. Arquitectura de un BAE con tres capas.

La Figura 2.13 muestra que un BAE está formado, al igual que cualquier tipo de AE, por dos bloques: un bloque de codificación (*encoder*) y uno de descodificación (*decoder*). Estos bloques están definidos por las funciones ω (Eq. (2.1)) y β (Eq. (2.2)), de la siguiente forma:

$$\omega : X \rightarrow F \tag{2.1}$$

$$\beta : F \rightarrow X \tag{2.2}$$

Donde $x \in \mathbb{R}^d = X$ se corresponde con el espacio de entrada que recibe el AE, y $z \in \mathbb{R}^p = F$ es la representación de dicha entrada, contenida en la capa de codificación, a partir de la cual el AE es capaz de reconstruir la salida.

A la hora de llevar a cabo el proceso de codificación de la entrada, si se considera la arquitectura más sencilla de AE (*shallow*), es decir, únicamente una capa oculta, existe una matriz de pesos, W que pondera cada una de las conexiones existentes, y un vector *bias*, b. De esta forma, este proceso puede ser expresado de la siguiente forma:

$$z \; = \; f(x) \; = \; \gamma_1 \left(W \, x \; + \; b \right) \tag{2.3}$$

El siguiente paso consiste en descodificar la nueva representación z para obtener x', que debe corresponderse con la reconstrucción de la entrada. Por tanto, de forma similar a la codificación, el espacio z se proyecta en x', por medio de la matriz de pesos W' y el vector *bias* b'. La siguiente expresión se corresponde con este proceso:

$$x' \; = \; g(x) \; = \; \gamma_2 \left(W' \, z \; + \; b' \right) \tag{2.4}$$

Eq. (2.3) se corresponde con la función de compresión, a partir de la cual se obtiene una nueva representación codificada de la entrada. Así mismo, Eq. (2.4) expresa el proceso de descodificación, donde el AE reconstruye la entrada a partir de la información contenida en la capa de codificación. En estas expresiones, γ_1 y γ_2 son las funciones de activación, que son habitualmente no lineales. En este sentido, algunas de las funciones de activación más utilizadas en los modelos de DL en general y en los AE en particular son: *rectified linear units* (RELU) [Glorot et al. (2011) y Gülçehre y Bengio (2016)], *scaled exponential linear units* (SELU) [Klambauer et al. (2017) y Kuchaiev y Ginsburg (2018)], hiperbólica tangencial y función sigmoidal, que es probablemente la más frecuentemente usada [LeCun et al. (2012)].

En cuanto a la función objetivo utilizada por los AE, generalmente se trata de una función de pérdida por instancia. Es decir, el proceso de aprendizaje de los AE consiste en optimizar las matrices de pesos y los vectores *bias* con el objetivo de minimizar la función de perdida. En este sentido, una función típicamente usada es el error cuadrático medio (*mean square error*, MSE).

Con respecto a los métodos de optimización de las matrices de pesos y *bias* durante el entrenamiento de los AE, se utilizan generalmente el algoritmo *stochastic gradient descent* (SGE) [Robbins y Monro (1951)] y sus variantes, como RMSProp [Tieleman e Hinton (2012)] o AdaGrad [Duchi et al. (2011)].

La técnica del gradiente descendente consiste en modificar los parámetros para lograr minimizar la función objetivo basándose para ello en la información proporcionada por la derivada de dicha función [Cauchy (1847)]. En este proceso, el algoritmo de propagación hacia atrás [Rumelhart et al. (1986)] tiene un papel fundamental. Dicho método comienza calculando los gradientes de las últimas capas del modelo y propagando dichos valores a través de la red hasta la capa de entrada. Así, la red es capaz de ajustar los valores de sus parámetros para optimizar la función objetivo. En muchos casos, un término de regularización se añade para prevenir que el modelo se ajuste demasiado a los datos de entrenamiento, lo que se conoce como *sobre-aprendizaje*.

En resumen, durante el proceso de entrenamiento, un AE pretende reducir el error de reconstrucción, para ello debe transmitir el error generado y ajustar los parámetros del modelo. El Algoritmo 1 refleja el pseudo-código asociado.

ALGORITMO 1: Pseudo-código asociado al algoritmo de entrenamiento propio de un AE.

Entradas:
TrainData ▷ Datos de entrenamiento

▷ Para cada instancia de entrenamiento:
for each *instancia* **in** *TrainData* **do**
    ▷ Obtener la salida generada por el AE:
    newCod ← feedForwardAE(*aeModel*, *instancia*)
    ▷ Calcular el error:
    error ← calculateError(*newCod*, *instancia*)
    ▷ Propagar hacia atrás el error a través de la red y ajustar los pesos:
    aeModel ← backpropagateError(*error*)
end for

En conclusión, el proceso que acaba de describirse permite que los AE sean capaces de generar un nuevo espacio de características de menor dimensionalidad pero mayor nivel de abstracción. En concreto, el modelo BAE descrito proporciona una metodología básica sobre la arquitectura y funcionamiento de los AE. A continuación, se describirán otras propuestas que incorporan restricciones y mecanismos sobre BAE con el objetivo de mejorar el rendimiento.

2.5.1.2. *Contractive Autoencoder*

Los AE son muy sensibles a las posibles variaciones presentes en los datos de entrada, ya que pequeñas perturbaciones en ellos podrían generar codificaciones muy diversas. Esto supone un claro inconveniente y es uno de los factores que motivó la aparición de un nuevo modelo conocido como *contractive* AE (CAE) [Rifai et al. (2012)]. Este modelo de AE logra invarianza local a los cambios en las instancias de entrenamiento. De esta forma la red es capaz de identificar estructuras *manifold* de menor dimensionalidad de forma más sencilla, es decir, conjuntos de instancias asociadas mediante relaciones no lineales [Cayton (2005)].

Para lograr esto, los CAE incluyen un término de regularización que permite estabilizar las codificaciones generadas por el modelo a pesar de las perturbaciones que se puedan dar en los datos de entrada. En cuanto a la arquitectura de este tipo de redes es similar a la vista anteriormente para BAE (véase Figura 2.13).

Las aplicaciones de los CAE son muy diversas. Además de su aplicación para reducción de dimensionalidad, pueden ser utilizados para afrontar, entre otros problemas, la generación de nuevas instancias durante el proceso de aprendizaje, mediante la introducción de ruido en diferentes puntos y el cálculo de las codificaciones asociadas [Rifai et al. (2012)].

2.5.1.3. *Denoising Autoencoder*

Los modelos de AE conocidos como *Denoising* (DAE) [Vincent et al. (2008b)] introducen una serie de modificaciones en el proceso básico propio de los AE para conseguir una mayor robustez, para ello se pretende lograr que el modelo sea capaz de reconstruir la entrada eliminando el ruido existente en ella.

Una de las aplicaciones de los AE ha sido la eliminación del ruido presente en los datos de entrada [Yann (1987)]. Sin embargo, los DAE tienen un objetivo más ambicioso, ya que pretenden aprovechar las ventajas que tiene la existencia de ruido en la entrada a la hora de entrenar el modelo. En este contexto, el nuevo espacio de características generado es más resistente ante la presencia de corrupción en la entrada y el rendimiento del modelo aumenta.

La arquitectura de los DAE es prácticamente idéntica a la correspondiente al modelo BAE, descrito anteriormente. El factor clave que incorporan los DAE es el hecho de introducir ruido en la entrada durante el proceso de entrenamiento. La Figura 2.14 permite observar la capa adicional que refleja dicha modificación de la entrada.

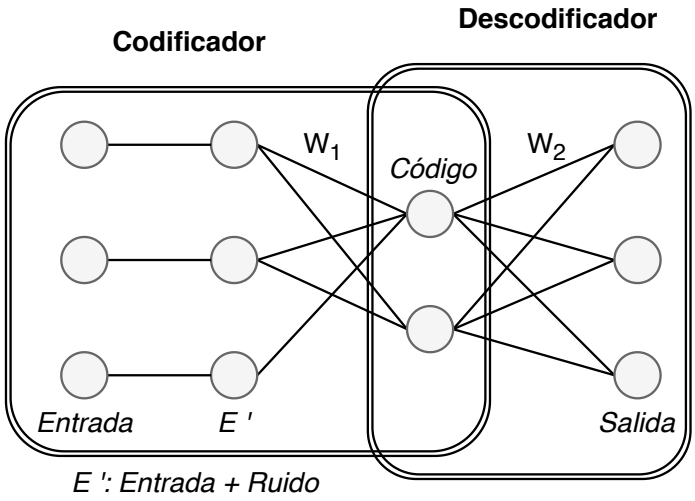


Figura 2.14. Arquitectura de un DAE con tres capas.

Un ejemplo de este tipo de metodología se puede encontrar en el trabajo Vincent et al. (2008b), donde un número determinado de variables de entrada son aleatoriamente modificadas y establecidas a 0. Sin embargo, en estos casos, la reconstrucción es comparada con la entrada original sin modificar. Este proceso permite que el AE sea capaz de detectar el ruido o perturbaciones existentes en la entrada.

La arquitectura de los DAE puede ser variable, teniendo múltiples capas ocultas. Así mismo, el proceso de entrenamiento puede ser adaptado según la forma en la que se corrompa la entrada [Vincent et al. (2010)]. En este sentido, existen distintos tipos de ruido que se pueden añadir, por ejemplo el ruido gaussiano, de Bernoulli o de Poisson.

2.5.1.4. *Robust Autoencoder*

Robust AE (RAE) son redes entrenadas para tolerar el ruido presente en los datos de entrenamiento [Qi et al. (2014)], al igual que los DAE explicados anteriormente. Sin embargo, esta tarea es afrontada de forma diferente. En el caso de los RAE, la variación incorporada consiste en modificar la función de pérdida usada durante la fase de entrenamiento del modelo. De esta forma los cambios incorporados permiten que la red, al mismo tiempo que minimiza el error de reconstrucción, identifique el ruido existente y el modelo sea más robusto ante su presencia. La arquitectura, por su parte, es similar a la vista para BAE (véase Figura 2.13).

Los RAE incorporan esta metodología con el objetivo de ser más tolerantes ante el ruido presente en los datos de entrada que los modelos BAE. Así, por ejemplo, la propuesta realizada en Liu et al. (2006) utiliza una función de error basada en la entropía cruzada (*correntropy*).

La *correntropy* permite medir la distribución asociada a la probabilidad de que dos eventos o instancias sean iguales. Esta medida se ve menos afectada por los valores extremos que otras medidas tradicionales, como MSE. Por ello, la robustez de la *correntropy* ante posibles perturbaciones en la entrada es mayor. De esta forma, al incorporar esta métrica los RAE son más tolerantes ante la presencia de ruido en el espacio de entrada.

En definitiva, en esta sección se han presentado los conceptos teóricos relacionados con los AE. La estructura y funcionamiento propias de este tipo de redes justifican de forma clara su elección para afrontar la tarea de reducción de dimensionalidad. Así mismo, se han descrito algunos de los modelos de AE más importantes y conocidos, todos ellos tendrán relevancia en los estudios que se describirán en los siguientes capítulos de la presente memoria.

2.6. Redes Neuronales Convolucionales

Las ANN tradicionales, en general, reciben una entrada que se puede representar como un vector y realizan una serie de transformaciones a través de las capas ocultas que componen su arquitectura. Cada una de las unidades que forman las distintas capas está totalmente conectada con todos los elementos de la capa anterior, pero trabajan de forma independiente con respecto al resto de neuronas de una misma capa. Este hecho implica que estas unidades no

comparten ningún tipo de parámetro entre sí, es decir, no existen conexiones entre ellas y los pesos asociados son independientes. Estas propiedades de los modelos ANN tradicionales hace que no ofrezcan buen rendimiento a la hora de procesar imágenes completas. Por ejemplo, si el tamaño de las imágenes tratadas es de 32x32 con tres canales de color, una única neurona de la primera capa oculta de la red tendrá 3072 conexiones con sus pesos asociados, ya que estará conectada con todos los píxeles de cada una de las bandas de entrada. Este número de conexiones no parece excesivamente grande, pero resulta sencillo observar que, a medida que la arquitectura del modelo aumente con más capas y neuronas, esta cantidad crecerá rápidamente, incrementando de forma excesiva la complejidad del modelo. En el contexto de las imágenes, es evidente que la conectividad completa entre los distintos elementos es innecesaria, ya que, en la mayor parte de los casos, las relaciones realmente relevantes se dan entre píxeles adyacentes en la imagen. Por tanto, la conectividad total supone un desperdicio de recursos y puede conducir a otros problemas típicos durante el proceso de aprendizaje, como es el caso del sobreaprendizaje. Las redes convolucionales mitigan, en cierta medida, estos problemas.

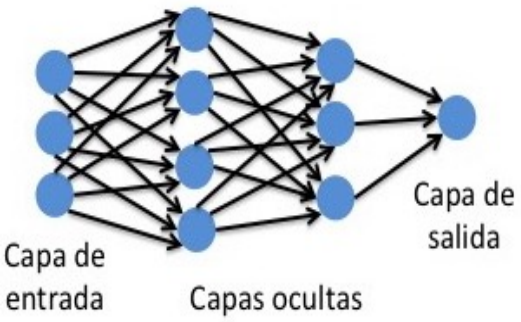
Las CNN son redes neuronales profundas que se basan en la forma en la que los animales llevan a cabo el proceso de visualización y detección de objetos [LeCun et al. (2010)]. Estas redes son uno de los tipos de redes DL más conocidas y utilizadas en la actualidad, debido a los buenos resultados que ofrecen a la hora de tratar diferentes tipos de datos. En concreto destaca el comportamiento de las CNN en campos como la clasificación de imágenes [Krizhevsky et al. (2012)] o vídeos [Karpathy et al. (2014)].

Una CNN se basa fundamentalmente en la aplicación de una serie de filtros sobre los datos de entrada que le permite identificar características de alto nivel presentes en ellos. De esta forma, las CNN no necesitan llevar a cabo un pre-procesamiento previo donde se realice una extracción manual de las variables más relevantes. Durante el proceso de entrenamiento, la red ajusta los parámetros propios de su arquitectura para aprender las características más relevantes de cara a llevar a cabo el objetivo final de la red que, en lo relativo a la presente tesis doctoral, se corresponde con la tarea de clasificación.

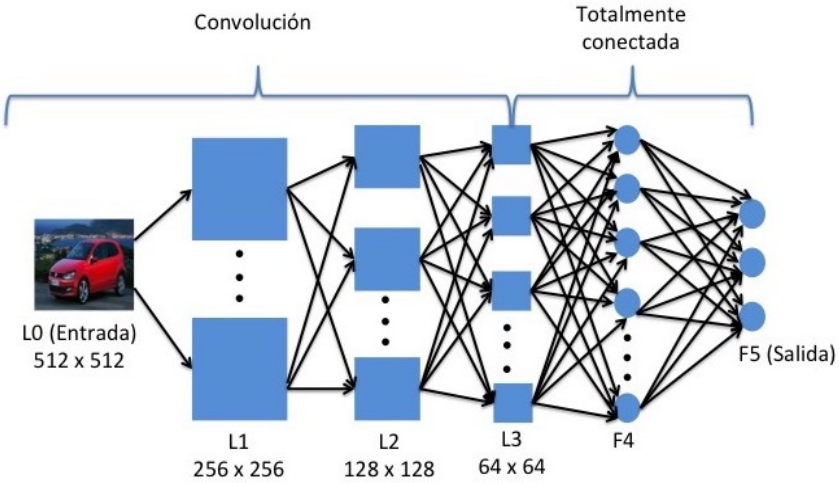
En este contexto, las CNN son capaces de identificar diversas características inherentes a los datos de entrada mediante complejas arquitecturas formadas por un número variable de capas ocultas, desde estructuras sencillas con 4 o 6 capas hasta tremendamente complejas formadas por cientos de capas ocultas. El aumento en la complejidad del modelo hace que el nivel de abstracción de

las características de alto nivel aprendidas por la red también aumente, ya que en cada capa aumenta la complejidad de dichas características con respecto a la anterior, es decir las capas se van especializando en el reconocimiento de formas cada vez más complejas. Por ejemplo, las primeras capas ocultas de una CNN podrían ser capaces de detectar bordes horizontales, vértices o esquinas, a medida que se profundice en la arquitectura sería posible identificar ojos, boca o nariz, mientras que las capas más profundas podrían aprender a detectar objetos complejos como caras.

En las CNN, frente a algunos modelos de ANN tradicionales donde cada neurona se conecta con todos elementos de la capa adyacente, las unidades que componen las capas tienen únicamente conexiones con una región limitada de la capa anterior. Esto implica una reducción considerable en el número de conexiones existentes y, por tanto, en la complejidad del modelo generado y en el coste computacional asociado al proceso de entrenamiento. En este sentido, la Figura 2.15 representa la diferencia estructurales entre ambos modelos.



(a) Red Neuronal Tradicional.



(b) Red Neuronal Convolucional.

Figura 2.15. Comparativa entre las arquitecturas de redes neuronales tradicionales y CNN.

De esta forma, tal y como se ha descrito anteriormente, la arquitectura típica de una red convolucional consiste en una secuencia de capas, que están divididas en dos partes: el bloque de convolución y las capas totalmente conectadas. Desde un punto de vista general, las capas destinadas a aplicar la convolución generan mapas de características que cada vez son más numerosos pero de menor dimensionalidad que el original. En este bloque cobran importancia las capas de *pooling* para reducir la dimensión del volumen de salida generado. Por otro lado, las capas totalmente conectadas agregan toda la información del modelo para realizar la tarea final, por ejemplo, la clasificación.

La Figura 2.16 representa un ejemplo de red convolucional concreta, que incorpora todos los tipos de capas que se han enumerado anteriormente: convolución, *pooling* y totalmente conectadas.

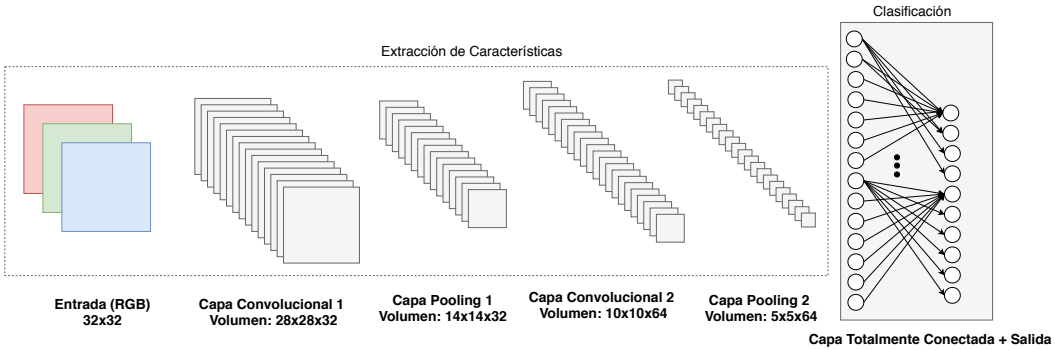


Figura 2.16. Ejemplo de arquitectura de CNN.

La arquitectura de ejemplo incluida en la Figura 2.16 se corresponde con el modelo utilizado en uno de los estudios que se describirá en los siguientes capítulos y está compuesta por las siguientes capas:

- **Capa de entrada:** la entrada de la red se corresponde con imágenes de 32x32 con 3 canales de color.
- **Par convolución+*pooling* 1:** se utilizan 32 filtros de tamaño 5x5 que generan un volumen de salida compuesto por 32 mapas de características. Posteriormente, se aplican filtros de *pooling* de tamaño 2x2 y paso 2 para reducir la dimensionalidad de dicho espacio de salida.
- **Par convolución+*pooling* 2:** se utilizan 64 filtros de tamaño 5x5 que generan un volumen de salida compuesto por 64 mapas de características. Al igual que antes, se aplican un filtro de *pooling* de tamaño 2x2 y paso 2 para reducir la dimensionalidad.

- **Capa totalmente conectada:** las unidades de salida de la capa anterior son combinadas a través de conexiones de entrada de las neuronas que componen esta capa.
- **Capa de salida:** estará totalmente conectada con la capa anterior y tendrá tantas unidades como clases tenga el problema, en este caso 10.

La arquitectura que se acaba de presentar es bastante sencilla. No obstante, este tipo de modelos pueden ser realmente complejos, añadiendo un gran número de capas e introduciendo modificaciones en el funcionamiento tradicional. Por ejemplo, existen CNN complejas formadas por una gran cantidad de niveles que incluyen una serie de capas preentrenadas, así el modelo sólo necesita entrenar las últimas capas para ajustarlas a los datos de entrada. Algunas propuestas recientes en este sentido son: ResNet [He et al. (2016)] o Inception [Szegedy et al. (2016)]. En este mismo contexto, la Figura 2.17 presenta la arquitectura de una CNN preentrenada conocida como GoogLeNet compuesta por 22 capas y que resultó ganadora de la competición de reconocimiento visual ImageNet en 2014 [Szegedy et al. (2015)].

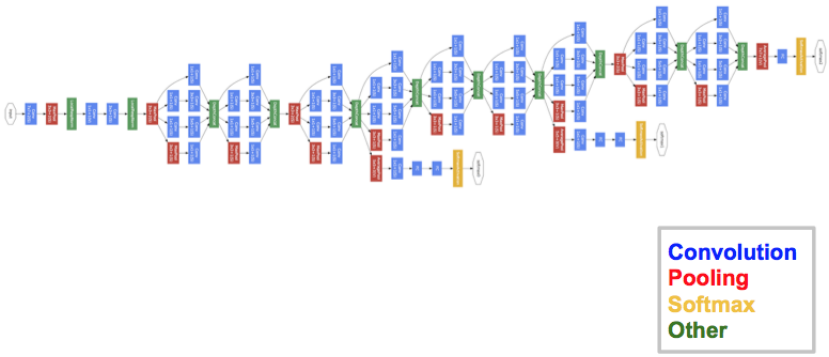


Figura 2.17. Arquitectura de la red convolucional GoogLeNet.
Fuente: Towards Data Science [Xu (2017)].

En definitiva, las CNN presentan una serie de características propias de su estructura y funcionamiento que las hacen diferentes de las ANN tradicionales. En este sentido, la Tabla 2.1 presenta una comparativa entre las CNN (aplicadas a clasificación de imágenes) y las redes *feedforward* tradicionales.

Estas propiedades diferenciadoras hacen que las CNN hayan sido utilizadas para afrontar problemas distintos a los propios de las ANN y que hayan ofrecido muy buenos resultados, por ejemplo en la clasificación de imágenes [Krizhevsky et al. (2012)]. Este gran rendimiento ha contribuido al auge en la

TABLA 2.1: Comparativa entre ANN *feedforward* tradicionales y CNN.

	ANN tradicional	CNN
Datos de entrada	Características de los datos	Píxeles de la imagen
Capas ocultas	Totalmente conectadas	Diferentes tipos (convolución, pooling y totalmente conectadas)
Capa de salida	Clases del problema	Clases del problema
Conexiones	Cada unidad de una capa con todas las de la siguiente	Asociados a los filtros de convolución (excepto capas finales totalmente conectadas)
Significado de las capas	Caja negra	Propiedades de alto nivel (desde líneas hasta formas complejas)
Parámetros de ajuste	Todos los pesos asociados a las conexiones de la red	Los filtros de convolución y pesos de las últimas capas totalmente conectadas

utilización de este tipo de modelos, siendo la arquitectura de DL más utilizada en la actualidad, como se ha indicado en el Capítulo 1. Por esta razón, las CNN han sido estudiadas durante esta tesis doctoral.

En concreto, dado que los efectos del desbalanceo en clasificación con modelos DL no han sido analizados, se ha considerado adecuado investigar este factor sobre la arquitectura profunda más frecuentemente utilizada en la literatura, las CNN.

Capítulo 3

Tratamiento de alta dimensionalidad con técnicas DL

El Capítulo 1 ha puesto de manifiesto que la tarea de clasificación es una de las más estudiadas dentro del aprendizaje automático, existiendo una gran variedad de familias de algoritmos que afrontan este problema. No obstante, como se ha descrito en la Sección 2.2, existe una serie de problemas asociados a los datos de entrada que influyen negativamente en el rendimiento de la mayor parte de los clasificadores.

En concreto, el presente capítulo se centra en analizar el problema de los algoritmos basados en distancias cuando tratan datos de la alta dimensionalidad. En particular, se analizará el algoritmo kNN [Altman (1992)], uno de los métodos clásicos pertenecientes a esta familia. En la Sección 2.2.1 se describieron las dificultades que tienen este tipo de algoritmos debido a los efectos de la *maldición de la dimensionalidad*, este factor repercute de forma directa en el cálculo de las distancias entre las distintas instancias de entrenamiento, las cuales son mucho menos significativas y, por tanto, el rendimiento predictivo decrece considerablemente. A pesar de estas dificultades, kNN es uno de los métodos más conocidos y utilizados debido a su buen comportamiento y a la alta interpretabilidad de sus resultados, por ello ha sido seleccionado en el presente estudio.

Para mitigar estos efectos negativos se han propuesto un gran número de métodos como se ha descrito en la Sección 2.2.1 que transforman el espacio de entrada en uno de menor dimensionalidad. En los últimos años, han surgido diferentes propuestas basadas en DL que aplican transformaciones no lineales de la entrada, frente a las combinaciones lineales de los métodos clásicos,

obteniendo resultados significativos. En concreto, los AE son estructuras muy adecuadas para afrontar este problema, tal y como se ha justificado en la Sección 2.5.

En este contexto, el presente capítulo describe una nueva propuesta. Se trata de un algoritmo basado en instancias que internamente incorpora una fase de reducción de dimensionalidad por medio de AE. El objetivo principal es ofrecer un nuevo método IBL basado en el tradicional kNN, pero que incorpore mecanismos para reducir los efectos de la alta dimensionalidad.

El método propuesto, llamado AEkNN [Pulgar et al. (2018b)], es un clasificador basado en kNN que incorpora AE para reducir la dimensionalidad del espacio de entrada. AEkNN proyecta los ejemplos en un espacio de menor dimensionalidad mediante un AE. Así, el método utiliza nuevas características de más alto nivel de abstracción y mayor calidad, lo que se traduce en una mejora en el rendimiento predictivo.

Con el objetivo de evaluar la propuesta realizada, el algoritmo AEkNN ha sido comparado con el método kNN clásico y con dos algoritmos tradicionales de reducción de dimensionalidad, como son PCA y LDA. Esta comparativa se ha realizado teniendo en cuenta tanto el rendimiento predictivo como el tiempo de ejecución. Los resultados muestran que el algoritmo AEkNN mejora en ambos aspectos tanto a kNN clásico como a PCA y LDA. Esta experimentación demuestra la utilidad del modelo propuesto y abre nuevas vías de trabajos futuros desarrollando métodos similares.

En definitiva, las aportaciones de este estudio son las siguientes:

- La propuesta de un nuevo clasificador, AEkNN, que hibrida métodos eficientes de reducción de dimensionalidad con un método de clasificación tradicional.
- El análisis detallado de la parametrización propia de AEkNN con el objetivo de optimizar los resultados.
- Una demostración experimental de las mejoras de AEkNN frente al algoritmo kNN clásico.
- Una comparativa entre la propuesta AEkNN y los métodos clásicos de reducción de dimensionalidad PCA y LDA.
- Un conjunto de guías o recomendaciones dirigidas a potenciales usuarios del método AEkNN que facilitan la parametrización del mismo en función de las características de los datos de entrada.

- La aplicación de AEkNN a dos datasets correspondientes a problemas reales.

A continuación, se presentan una serie de apartados que, junto con el marco teórico ofrecido en el Capítulo 2, contextualizan y describen el método AEkNN desarrollado. En concreto, en primer lugar, la Sección 3.1 presenta el algoritmo AEkNN. A continuación, la Sección 3.2 resume la experimentación llevada a cabo, así como los resultados obtenidos. En la Sección 3.3 se describen las conclusiones asociadas a este trabajo. Finalmente, en la Sección 3.4 se cita la aportación, en forma de artículo de revista, asociada a este trabajo [Pulgar et al. (2018b)], cuyo texto integro se puede consultar en la Sección A.1 del Apéndice A.

3.1. Propuesta de reducción de dimensionalidad con kNN: AEkNN

El objetivo principal de este capítulo es la presentación del método AEkNN, un clasificador basado en distancias que incorpora una fase interna de reducción de dimensionalidad mediante AE. En las siguientes secciones se expone de forma detallada dicho método: una base teórica en la Subsección 3.1.1, la descripción detallada del método en la Subsección 3.1.2 y una justificación de las principales aportaciones en la Subsección 3.1.3.

3.1.1. Base teórica AEkNN

AEkNN es un método que pretende mitigar los efectos de la alta dimensionalidad que afectan en gran medida a los algoritmos IBL. Para ello, incorpora una fase de reducción de dimensionalidad basada en el uso de AE. Este tipo de red neuronal es muy apropiada para esta tarea, tal y como se explicó en la Sección 2.5. En este caso, los AE utilizados tienen una capa oculta intermedia de menor dimensionalidad que la capa de entrada y salida. Esto permite que la codificación generada en dicha capa puede ser extraída como nueva representación de la entrada original pero con menor dimensionalidad y más alto nivel.

En concreto, AEkNN está dividido fundamentalmente en dos fases. Por un lado, una primera fase donde genera un modelo basado en AE que permite reducir la dimensionalidad del espacio de entrada, es decir, el método proyecta el espacio N-dimensional X en un nuevo espacio M-dimensional Z con $M < N$. En este nuevo escenario, las características serán generalmente de mayor nivel y, por tanto, las distancias entre los ejemplos serán más representativas. Por otro lado, una segunda fase donde realiza la clasificación de forma similar a la llevada a cabo por el algoritmo kNN clásico. No obstante, al contrario que kNN, el método AEkNN no es *lazy* puesto que construye un modelo para llevar a cabo la reducción de dimensionalidad.

3.1.2. Descripción de AEkNN

El proceso de entrenamiento de AEkNN consiste en entrenar el modelo basado en AE para aprender a generar el nuevo espacio de características. Una vez completo, ante una instancia de test, el algoritmo utiliza el modelo anterior para obtener la nueva representación, la cual es utilizada para generar la clasificación en la segunda fase del proceso. La Figura 3.1 presenta un esquema general del procedimiento seguido por AEkNN. Así mismo, a continuación se muestra y se describe el pseudo-código Algoritmo 2 asociado a AEkNN.

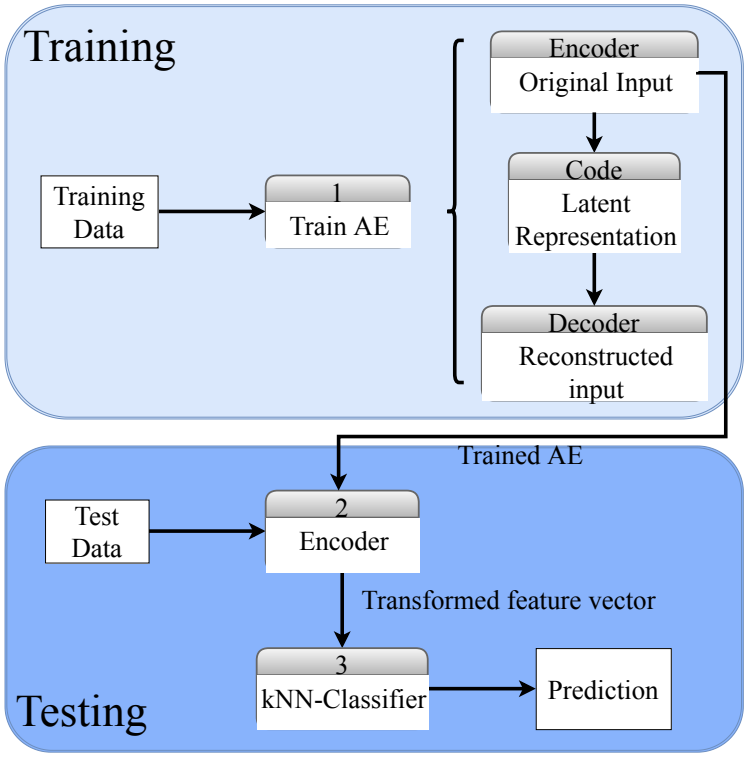


Figura 3.1. Esquema de funcionamiento de AEkNN.

Algoritmo 2 Pseudo-código del algoritmo AEkNN.

```
Inputs:
    TrainData                          ▷ Datos de entrenamiento
    TestData                           ▷ Datos de test
    PPL                                ▷ Porcentaje por capa
    k                                  ▷ Número vecinos más cercanos
▷ Fase de entrenamiento:
modelData ← TrainData
aeModel ← ()
for each layer in PPL do
    sizeLayer ← getSizeLayer(modelData, layer)
    aeLayer ← getAELayer(modelData, sizeLayer)
    modelData ← applyAELayer(aeLayer, modelData)
    aeModel ← addAEModel(aeModel, aeLayer)
end for
▷ Fase de clasificación:
result ← classification(TrainData, TestData, k, aeModel)
return result

function GETAELAYER(modelData, sizeLayer)
    aeLayer ← initializeAE(modelData, sizeLayer)
    for each instance in modelData do
        outPut ← feedForwardAE(aeLayer, instance)
        error ← calculateDeviation(instance, outPut)
        aeLayer ← updateWeightsAE(aeLayer, error)
    end for
    return aeLayer
end function

function CLASSIFICATION(TrainData,TestData,k, aeModel)
    error ← 0
    for each instance in TestData do
        newCod ← feedForwardAE(aeModel, instance)
        outPut ← distanceBased(newCod, k, TrainData)
        if outPut != realOutPut(instance) then
            error ← error + 1
        end if
    end for
    result ← error / size(TestData)
    return result
end function
```

Las entradas del método descrito en Algoritmo 2 son *TrainData* y *TestData* que se corresponden con los datos de entrenamiento y test respectivamente, el número de vecinos más cercanos k y el parámetro PPL, que define la arquitectura del AE interno. Este último es un vector con tantos elementos como capas ocultas tenga el AE, cada valor hace referencia al número de

unidades de cada capa, en concreto indica el porcentaje con respecto a la cantidad de atributos de entrada. Una de las aportaciones del trabajo consiste en estudiar diferentes arquitecturas.

Tal y como se puede comprobar, el algoritmo se divide en dos partes. La primera parte (líneas 2-9) se corresponde con la etapa de entrenamiento de AEkNN. La segunda fase (lineas 11-12) hace referencia a la clasificación. Así, durante el entrenamiento, el método se centra en aprender un modelo que permite generar una nueva representación de los datos de entrada, mediante AE. Existen diferentes formas de entrenar un AE, en este caso, se utiliza un entrenamiento capa a capa mediante un bucle incluido entre las líneas 4-9, que genera el modelo encargado de reducir los efectos de la alta dimensionalidad en la etapa posterior de clasificación.

En este punto, es importante destacar que la modificación de pesos asociada al AE (líneas 16-20) se lleva a cabo aplicando *mini-batch gradient descent* [Hinton (2010)]. Este proceso es una variación del algoritmo del gradiente descendente que distribuye las instancias de entrenamiento en pequeños conjuntos o lotes que son usados de forma conjunta para obtener el error asociado al modelo y modificar los parámetros necesarios para optimizar la función objetivo.

Finalmente, en la fase de clasificación (líneas 11-12) el objetivo es predecir la clase asociada a los datos de test. Para ello, en primer lugar se lleva a cabo una transformación de las instancias de test usando el modelo de AE entrenado en la etapa anterior. Esta nueva codificación de menor dimensionalidad se utiliza para predecir la clase mediante el clasificador basado en distancias kNN.

3.1.3. Contribuciones de AEkNN

AEkNN presenta diferencias significativas con respeto a otras propuestas relacionadas presentes en la literatura [Hinneburg et al. (2000), Yu et al. (2001), Min et al. (2009), Radovanović et al. (2010) y Wang (2011)]. Las características más relevantes de AEkNN son las siguientes:

- AEkNN es un método basado en distancias que integra reducción de dimensionalidad. Por esta razón, no necesita una fase externa de pre-procesamiento para llevar a cabo dicha reducción. Esto implica una mejora clara con respecto al método clásico kNN.

- AEkNN lleva a cabo una fusión de características que implica la generación de una nueva representación que agregue la información más relevante de la entrada.
- La parametrización propia de AEkNN permite aplicar diferentes grados de reducción de dimensionalidad, según las características de los datos y las necesidades asociadas al problema.
- AEkNN ha sido diseñado para mejorar tanto el rendimiento predictivo como el tiempo de ejecución en comparación con el algoritmo kNN clásico.
- AEkNN puede ser utilizado con datasets con características muy diversas, demostrando un gran rendimiento en general.

Estas contribuciones asociadas al método propuesto AEkNN reflejan la utilidad del nuevo algoritmo, así como sus ventajas con respecto a propuestas anteriores. A continuación, se describe la experimentación llevada a cabo para demostrar las mejoras obtenidas con AEkNN.

3.2. Experimentación y resultados

Esta sección presenta un resumen de la experimentación diseñada para demostrar las mejoras asociadas al método AEkNN, los detalles concretos de la misma se pueden encontrar en el artículo publicado en la revista International Journal of Computational Intelligence Systems [Pulgar et al., 2018b], cuya referencia completa se ha indicado en la Sección 3.4.

La experimentación está estructurada en cuatro fases fundamentales:

- Un estudio de la parametrización del método AEkNN. Para ello se consideran diferentes arquitecturas con el objetivo de seleccionar la óptima desde el punto de vista del rendimiento predictivo. La Subsección 3.2.1 resume este fase.
- Una comparativa entre el método AEkNN y el algoritmo kNN clásico. Una vez seleccionada la mejor configuración, se realiza una comparativa con respecto al método clásico para demostrar las mejoras obtenidas tanto desde el punto de vista del rendimiento predictivo como del tiempo de ejecución. Este análisis es descrito brevemente en la Subsección 3.2.2.

- Un análisis del rendimiento predictivo de AEkNN frente a algoritmos clásicos de reducción de dimensionalidad como PCA y LDA, reflejado en la Subsección 3.2.3
- Una aplicación real del algoritmo AEkNN utilizando dos datasets asociados a problemas reales y unas conclusiones generales, incluidas en la Subsección 3.3.

El marco experimental asociado a las fases que se acaban de enumerar esta descrito en el artículo final [Pulgar et al. (2018b)] incluido en la Sección A.1 del Apéndice A. En este sentido, cabe destacar que se han utilizado un conjunto de 14 datasets con características muy diversas. Así mismo, se han utilizado tres métricas diferentes de evaluación del rendimiento predictivo: *Accuracy*, *F-Score* y área bajo la curva ROC (*AUC*), además del tiempo de ejecución para valorar el coste computacional asociado a cada método. Finalmente, es importante indicar que todos los resultados han sido verificados utilizando los test estadísticos propios de este tipo de experimentaciones.

3.2.1. Análisis de la parametrización de AEkNN

Esta primera fase tiene como objetivo determinar la parametrización del modelo AEkNN que mejores resultados proporciona. En concreto, consiste en determinar el valor del parámetro PPL, el cual permite cambiar la arquitectura del AE interno tanto en número de capas como en cantidad de unidades por capa.

En este marco experimental, dado que existen un gran número de conjuntos de datos, se ha optado por definir 6 arquitecturas diferentes, las cuales se pueden observar en la Tabla 3.1. En concreto, existen 3 arquitecturas con una capa oculta y 3 arquitecturas con tres capas ocultas. En esta experimentación no se han considerado arquitecturas más complejas dado que se trata de un análisis del equilibrio entre la eficacia y la eficiencia del modelo utilizando un conjunto amplio de datasets, no de obtener el mejor rendimiento posible, en cuyo caso el modelo debería de adaptarse al conjunto de datos utilizado.

A partir de los resultados obtenidos en esta fase de la experimentación se pueden extraer una serie de tendencias que se enumeran a continuación:

- **Modelos de 1 capa oculta:** en general los modelos de una capa oculta funcionan mejor que los de tres capas. En torno a un 80 % de los datasets analizados generan el mejor rendimiento con estas arquitecturas.

TABLA 3.1: Configuraciones usadas en la primera fase de la experimentación (parámetro PPL).

	Número de capas ocultas	Número de neuronas (%) Capa 1	Capa 2	Capa 3	Parámetro PPL
AEkNN 1	1	25	-	-	(25)
AEkNN 2	1	50	-	-	(50)
AEkNN 3	1	75	-	-	(75)
AEkNN 4	3	150	25	150	(150, 25, 150)
AEkNN 5	3	150	50	150	(150, 50, 150)
AEkNN 6	3	150	75	150	(150, 75, 150)

- **PPL=(75):** un 50 % de los mejores resultados se obtienen con esta configuración. Además, el rendimiento de esta arquitectura está cerca del óptimo en la mayor parte de los casos.
- **PPL=(50):** un 29 % de los casos analizados generan los mejores resultados con esta configuración. En el resto de casos, el rendimiento es relativamente bueno.
- **Otros comentarios:** el resto de configuraciones generan los mejores resultados en ciertas ocasiones, sin embargo en otros casos el comportamiento es bastante deficiente, mostrando un rendimiento más irregular.
- **Tiempo de ejecución:** los mejores resultados en cuanto a tiempo de ejecución se obtienen con los datasets que más reducen la dimensionalidad del modelo. Esto es obvio, dado que la carga computacional asociada es menor a medida que decrece el número de variables del mismo.

En definitiva, tras realizar la experimentación se seleccionaron los modelos de una capa oculta que comprimen al 75 % y al 50 % los datos de entrada. Estos resultados están apoyados en los test estadísticos apropiados que muestran diferencias significativas entre estas configuraciones y el resto.

3.2.2. AEkNN vs kNN

Una vez seleccionadas las mejores configuraciones para AEkNN, el siguiente paso de la experimentación pretende demostrar la mejora con respecto al método kNN clásico. Dicha comparativa se realiza tanto desde el punto de vista del rendimiento predictivo como del tiempo de ejecución. Para ello, los resultados asociados a las dos configuraciones óptimas seleccionadas en el paso anterior se comparan con los obtenidos tras aplicar kNN sobre los mismos conjuntos de datos.

En este contexto, las conclusiones asociadas a esta experimentación se detallan a continuación:

- **Rendimiento predictivo:** considerando las distintas métricas utilizadas, las dos configuraciones del algoritmo AEkNN mejoran en un 95 % de los casos al método kNN clásico. La configuración con $PPL = (75)$ muestra un comportamiento ligeramente superior en este aspecto.
- **Tiempo de ejecución:** ambas configuraciones del algoritmo AEkNN mejoran en el 100 % de los casos el rendimiento de kNN. En este caso, la configuración de AEkNN con $PPL = (50)$ genera el mejor resultado para todos los datasets. Esto es obvio dado que es la configuración que genera un espacio de dimensionalidad más reducido.
- **Test estadísticos:** estos test reflejan que existen diferencias significativas entre el algoritmo AEkNN y el método kNN, considerando tanto las métricas analizadas como el tiempo de ejecución.

En resumen, a pesar de que AEkNN proyecte el espacio original en uno de menor dimensionalidad, la calidad de los resultados mejora considerablemente teniendo en cuenta el rendimiento predictivo. En este sentido, la configuración de una capa que reduce la cantidad de características al 75 % ofrece los mejores resultados. Así mismo, si se atiende a tiempo de ejecución, AEkNN también genera mejores resultados que kNN, algo obvio dado que utiliza un espacio de menor dimensionalidad, por tanto el coste computacional es menor. Así, la configuración que mejor funciona en este caso es la que más comprime, en concreto al 50 %.

3.2.3. AEkNN vs PCA y LDA

El objetivo de la tercera fase de la experimentación es evaluar el rendimiento de AEkNN frente a dos algoritmos tradicionales de reducción de dimensionalidad, como son PCA y LDA. Estos modelos son muy conocidos y siguen siendo muy utilizados en la literatura [Yu et al. (2018) y Aymaz y Köse (2019)]. Esta comparativa se ha realizado a partir de los resultados de las dos mejores configuraciones de AEkNN y los generados mediante PCA y LDA, sobre el mismo conjunto de datos. Es importante destacar que la comprensión llevada a cabo por PCA y LDA debe ser equivalente a la realizada con AEkNN, por ello se han realizado dos ejecuciones para cada método con los parámetros equivalentes a las dos configuraciones de AEkNN.

Una vez llevada a cabo la experimentación, los resultados obtenidos reflejan una serie de tendencias que se detallan a continuación:

- **LDA:** AEkNN supera al método LDA en más del 70 % de los casos analizados considerando las distintas métricas. LDA obtiene los mejores resultados en el 30 % de los datasets.
- **PCA:** AEkNN obtiene mejores resultados que PCA en más del 95 % de los casos. PCA no obtiene en ningún caso el mejor rendimiento.
- **Tiempo de ejecución:** el tiempo de ejecución es equivalente en las distintas configuraciones analizadas, dado que el nivel de compresión es similar. Así, las configuraciones que más comprimen genera un mejor rendimiento en este sentido.
- **Test estadísticos:** los test realizados muestran la existencia de diferencias significativas en los resultados obtenidos, lo que confirma los beneficios proporcionados por AEkNN.

Esta fase de la experimentación muestra claramente que los resultados de AEkNN mejoran a los obtenidos mediante algoritmos clásicos como PCA y LDA. Esto significa que el nuevo espacio generado internamente por AEkNN produce nuevas características de mayor calidad y con información más relevante, que las generados por PCA y LDA.

3.3. Conclusiones

Las secciones anteriores ponen de manifiesto que AEkNN es un algoritmo robusto y cuyo comportamiento mejora a propuestas clásicas. Así mismo, la Sección 3.2.1 concluía que existen dos configuraciones de AEkNN cuyo rendimiento es mejor. A continuación se detallan unas recomendaciones a la hora de seleccionar la configuración más apropiada en función de los datos de entrada y en base a la experimentación llevada a cabo:

- Cuando se utilizan datasets con un número de características muy alto, los mejores resultados se obtienen utilizando $PPL = (50)$ para AEkNN. La experimentación ha reflejado este comportamiento con datasets de más de 600 características. Esto se debe fundamentalmente a que la comprensión en espacios de dimensionalidad tan elevada puede ser mayor sin perder información relevante.

- Cuando se utilizan datasets binarios con dimensionalidad más reducida, la mejor opción continúa siendo considerar $PPL = (50)$ para AEkNN. En la experimentación, esta configuración ha mostrado el mejor rendimiento con datasets binarios con en torno a 100 características. En estos casos, las compresión también puede ser mayor.
- En el resto de casos, si se pretende optimizar el rendimiento predictivo se debe optar por la configuración que comprime menos la información, es decir AEkNN con $PPL = (75)$. En cambio, si el objetivo es reducir el tiempo de ejecución se debe optar por la configuración que comprime más, $PPL = (50)$.

Así, la experimentación descrita en el presente capítulo pone de manifiesto que, al igual que ocurre con todos los métodos de aprendizaje automático, es necesario tener en cuenta que la parametrización de AEkNN debe adaptarse al contexto del problema y a las características de los datos para optimizar los resultados obtenidos.

En el artículo, incluido en el Anexo A.1, estas recomendaciones se han tenido en cuenta a la hora de aplicar AEkNN sobre dos datasets de gran dimensionalidad asociados a problemas reales. Los resultados generados se han comparado con el método kNN clásico mostrando mejoras claramente significativas en ambos casos [Pulgar et al. (2018b)].

En conclusión, en este trabajo se propone un algoritmo llamado AEkNN que hibrida reducción de dimensionalidad basada en AE con el clasificador tradicional kNN con el objetivo de mitigar los efectos de la alta dimensionalidad que tanto afecta a todos los métodos basados en distancias. La experimentación llevada a cabo demuestra la eficacia del método tanto respecto al kNN clásico como a propuestas tradicionales de reducción de dimensionalidad, lo que pone de manifiesto la utilidad del algoritmo AEkNN y sus posibles aplicaciones en la solución de problemas reales.

3.4. Publicaciones asociadas

El trabajo descrito a lo largo del presente capítulo, cuyo texto integro se puede consultar en la Sección A.1 del Apéndice A, está asociado a la publicación [Pulgar et al. (2018b)] cuyos datos son los siguientes:

- **Título:** AEkNN: An AutoEncoder kNN-Based Classifier With Built-in Dimensionality Reduction.

  Autores: Francisco J. Pulgar, Francisco Charte, Antonio J. Rivera, María J. del Jesus.

  Revista: International Journal of Computational Intelligence Systems, Volume 12, Issue 1, November 2018, Pages 436 - 452.

  ISSN (Online): 1875-6883.

  DOI: 10.2991/ijcis.2018.125905686.

  Estado: Publicado.

Capítulo 4

Estudio sobre la relación entre *autoencoder*, método de aprendizaje y problema

Tal y como se ha descrito en los capítulos anteriores, en los últimos años han surgido nuevas propuestas basadas en DL que reducen la dimensionalidad del espacio de entrada desde una perspectiva conocida como fusión de características. Estas técnicas tienen como objetivo principal obtener nuevas representaciones de los datos de entrada mediante la combinación de las características originales más relevantes y que aportan la información más útil. En este sentido, uno de los modelos más adecuados para afrontar esta tarea son los AE.

La Sección 2.5 ha descrito con detalle el funcionamiento de este tipo de modelos. Uno de los aspectos más relevantes es que la estructura interna de los AE permite obtener una nueva representación de los datos de entrada, cuyas características dependerán de la propia arquitectura del modelo. Así, en general, para afrontar la tarea de reducción de dimensionalidad, la capa oculta de codificación suele tener menor tamaño que la entrada y la salida [Charte et al. (2018)].

Este capítulo estudia el rendimiento de algunas de las variantes de AE más conocidas, en concreto: AE básico [Bourlard y Kamp (1988)], *denoising* AE [Vincent et al. (2008b)], *contractive* AE [Rifai et al. (2012)] y *robust* AE [Qi et al. (2014)], todas ellas descritas en la Sección 2.5. El objetivo perseguido es llevar a cabo un análisis exhaustivo de su rendimiento a la hora de afrontar la tarea de reducción de dimensionalidad con diferentes paradigmas de clasificación.

De esta forma, el estudio realizado permite establecer una serie de líneas de actuación dirigidas a orientar a cualquier usuario sobre el uso de AE a la hora de afrontar la reducción de dimensionalidad.

En resumen, este estudio pretende guiar a los usuarios en la elección del modelo de AE más adecuado para llevar a cabo la fusión de características teniendo en cuenta las propiedades de los datos de entrada y el tipo de clasificador. Así, las principales aportaciones asociadas a este análisis son:

- Un exhaustivo análisis paramétrico de cuatro modelos de AE: básico, *denoising*, *contractive* y *robust*, que permita seleccionar la mejor arquitectura desde el punto de vista del rendimiento predictivo.
- Una comparativa entre el rendimiento predictivo asociado a los modelos de AE y el obtenido utilizando los datos originales, considerando los clasificadores: kNN [Cover y Hart (1967)], SVM [Boser et al. (1992)], MLP [Hornik et al. (1989)] y C4.5 [Quinlan (1986)].
- Una demostración experimental que determine qué modelo de AE ofrece el mejor rendimiento para cada paradigma de clasificación considerado.
- Una comparación entre el rendimiento de los distintos modelos de AE y cuatro algoritmos clásicos de reducción de dimensionalidad: PCA [Pearson (1901)], LDA [Yu y Yang (2001)], ISOMAP [Tenenbaum (2000)] y LLE [Roweis y Saul (2000)].
- Una serie de líneas de actuación que permita al usuario final decidir qué modelo de AE es el más adecuado según las características tanto del clasificador como de los datos de entrada.

Los resultados obtenidos en la experimentación reflejan el gran rendimiento de los AE a la hora de afrontar la tarea de reducción de dimensionalidad. En general, el comportamiento de cualquier modelo de AE considerado mejora al obtenido mediante los datos originales sin procesar. Así mismo, entre los tipos de AE, los más sofisticados, es decir, *contractive*, *denoising* y *robust*, mejoran claramente al modelo básico. Además, la experimentación demuestra que los modelos basados en AE funcionan mejor que los métodos tradicionales PCA, LDA, ISOMAP y LLE.

El marco teórico asociado a este estudio se ha descrito en el Capítulo 2. A continuación, las siguientes secciones del presente capítulo describen la experimentación realizada, así como los principales resultados obtenidos. En concreto, la Sección 4.1 describe con detalle la experimentación llevada

a cabo. En la Sección 4.2 se presentan los principales resultados de dicha experimentación. La Sección 4.3 establece las conclusiones del trabajo y las recomendaciones destinadas a los usuarios de este tipo de modelos. Finalmente, en la Sección 4.4 se enumeran las publicaciones asociadas a este estudio, entre ellas, la aportación principal [Pulgar et al. (2020a)] está incluida de forma integra en la Sección A.2 del Apéndice A.

4.1. Selección de *autoencoder* en base a las características del conjunto de datos y al método de aprendizaje

Para analizar el comportamiento de los diferentes modelos de AE a la hora de afrontar la reducción de dimensionalidad, además de estudiar el rendimiento desde un punto de vista individual, se ha analizado el tipo de AE con mejor funcionamiento en función de los datos de entrada y el clasificador utilizado. Para ello, el proceso se divide en dos etapas fundamentales:

- **Reducción de dimensionalidad:** el primer paso consiste en obtener nuevas representaciones de menor dimensionalidad de los datos de entrada utilizando los modelos de AE básico, *contractive*, *denoising* y *robust*. En cada caso, se han tenido en cuenta distintas arquitecturas para evaluar la que genera un mejor rendimiento. Como resultado de esta fase, se obtienen diferentes representaciones de la entrada original con distinto grado de reducción.
- **Clasificación:** se utilizan distintos algoritmos para evaluar el rendimiento predictivo generado a partir de los conjuntos de datos obtenidos en el análisis anterior. Los resultados generados servirán como base para establecer las comparativas asociadas al presente estudio. Los clasificadores utilizados son: kNN, MLP, SVM y C4.5.

Una vez llevado a cabo el proceso anterior, se realiza un análisis detallado de los resultados obtenidos que permita establecer conclusiones asociadas al uso de este tipo de modelos. En concreto, los objetivos que se persiguen son:

- **Analizar la arquitectura de los AE:** el primer objetivo perseguido es determinar qué estructura de los AE ofrece el mejor rendimiento predictivo. Así, las comparativas realizadas se basarán en la mejor arquitectura.

- **Comparar modelos de AE:** en este caso, el propósito es estudiar el rendimiento de los distintos AE, realizando una comparativa entre ellos y los resultados obtenidos a partir de los datos sin procesar. La arquitectura de AE considerada en esta fase es la que mejor rendimiento haya obtenido en el análisis previo.
- **Evaluar modelos de AE frente a técnicas tradicionales:** el tercer objetivo pretende analizar el comportamiento de los modelos de AE frente a algoritmos tradicionales de reducción de dimensionalidad como PCA, LDA, ISOMAP y LLE.
- **Extraer conclusiones y recomendaciones:** finalmente, tras analizar todos los resultados obtenidos, se establecen una serie de conclusiones generales, así como recomendaciones dirigidas a usuarios potenciales de este tipo de modelos basados en AE aplicados a reducción de dimensionalidad.

En este contexto se ha utilizado un conjunto amplio de datasets con características muy diversas, entre las que destacan variabilidad en el número de ejemplos, atributos y clases. Así mismo, la métrica de evaluación utilizada en la experimentación es AUC. Finalmente, es importante indicar que se han utilizado los test estadísticos apropiados para confirmar los resultados obtenidos. Los detalles concretos del marco experimental asociado a este trabajo se pueden consultar en el artículo incluido en la Sección A.2 del Apéndice A.

En las siguientes secciones se presenta un resumen de los resultados más relevantes obtenidos en este estudio, así como las conclusiones más destacadas. Finalmente, se describen las recomendaciones dirigidas a los posibles usuarios.

4.2. Experimentación y resultados

Esta sección contiene los resultados más relevantes de la experimentación relacionada con el estudio descrito anteriormente. Debido a la magnitud de las pruebas llevadas a cabo, no se mostrarán todos los datos obtenidos, sino que se presentarán figuras y rankings que ofrecerán una visión global de los mismos. De esta forma, se podrá tener una perspectiva general del rendimiento de los AE. Los resultados detallados se encuentran en el artículo asociado a esta investigación [Pulgar et al. (2020a)].

4.2.1. Análisis de la arquitectura de los AE

En este primer apartado, el objetivo perseguido es determinar qué arquitectura de AE aplicada a fusión de características ofrece un mejor rendimiento predictivo. En este estudio, sólo se tienen en cuenta arquitecturas de una capa oculta (tres capas en total, contando la de entrada y la de salida), dado que en el Capítulo 3 se ha verificado que, en principio, son las que mejores resultados producen. Así, las arquitecturas consideradas se describen en la Tabla 4.1.

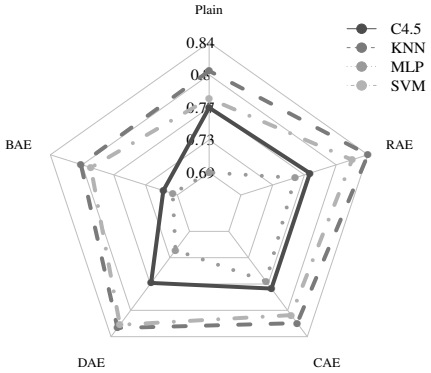
TABLA 4.1: Arquitecturas de *autoencoder* usadas en la experimentación.

	Número de capas	Número de unidades (% del total) de la capa oculta
ARQ_25	3	25
ARQ_50	3	50
ARQ_75	3	75

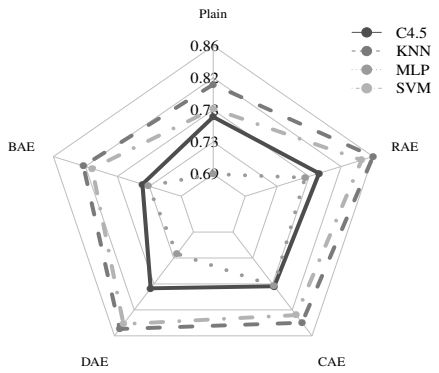
Tal y como se puede comprobar las tres arquitecturas utilizadas son simétricas y tienen una única capa oculta, la cual, en todos los casos, es de menor dimensionalidad que la entrada y la salida, algo propio de los modelos aplicados a reducción de dimensionalidad. En concreto, los AE generan nuevas representaciones cuya dimensión se corresponde con el 25 %, 50 % y 75 % del tamaño original.

En este contexto, para ofrecer una primera aproximación a los resultados obtenidos se muestra la Figura 4.1, compuesta por tres gráficos de radar, que presenta el rendimiento predictivo (métrica AUC) de las tres arquitecturas anteriormente enumeradas, así como el obtenido utilizando los datos originales sin procesar. En los tres casos se realiza una agregación de los datos por configuración, modelo de AE utilizado y clasificador. Cada clasificador se representa con un tipo de línea diferente, mientras que los distintos AE corresponden con los vértices.

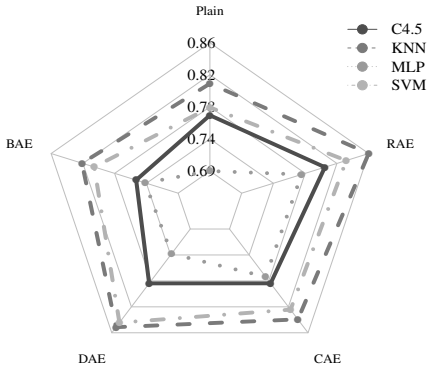
La Figura 4.1 refleja el rendimiento de los cuatro modelos de AE considerados: *básico* (BAE), *contractive* (CAE), *denoising* (DAE) y *robust* (RAE). En general, las gráficas muestran que el modelo RAE funciona mejor en la mayor parte de los casos. Los peores resultados se obtienen con el modelo BAE y los datos originales sin procesar (*plain*). Este comportamiento es el esperado, ya que es normal que los modelos más sofisticados generen representaciones de mayor calidad que la configuración básica.



(a) ARQ_25: Reducción al 25 % del total de características originales.



(b) ARQ_50: Reducción al 50 % del total de características originales.



(c) ARQ_75: Reducción al 75 % del total de características originales.

Figura 4.1. Rendimiento predictivo (AUC) de los diferentes modelos de *autoencoder* y algoritmos de clasificación para cada arquitectura. Tipos de AE considerados: *básico* (BAE), *contractive* (CAE), *denoising* (DAE) y *robust* (RAE).

Tras tener una visión general de los resultados, el siguiente paso consiste en determinar qué arquitectura ofrece un mayor rendimiento. Para ello, es necesario establecer una comparativa entre las configuraciones consideradas. En este sentido, la Tabla 4.2 presenta los rankings que agregan todos los resultados obtenidos con los distintos datasets y modelos de AE, para tener una perspectiva global por arquitectura que permita determinar cuál de ellas funciona mejor desde el punto de vista del rendimiento predictivo.

TABLA 4.2: Rankings considerando rendimiento predictivo de las diferentes arquitecturas de AE por método de clasificación.

kNN		SVM		MLP		C4.5	
Arquitectura	Ranking	Arquitectura	Ranking	Arquitectura	Ranking	Arquitectura	Ranking
ARQ_75	1.850	ARQ_75	2.037	ARQ_75	1.850	ARQ_75	1.825
ARQ_50	2.075	ARQ_50	2.050	ARQ_50	2.150	ARQ_50	2.300
ARQ_25	2.750	ARQ_25	2.662	ARQ_25	2.725	Original	2.837
Original	3.325	Original	3.250	Original	3.275	ARQ_25	3.037

Tal y como se puede verificar en la Tabla 4.2, la configuración que genera un mejor rendimiento predictivo para los cuatro clasificadores seleccionados es la que reduce el número de características a un 75 % del tamaño original (ARQ_75). Así mismo, los peores resultados se obtienen a partir de los datos originales sin procesar, excepto para C4.5 donde se obtienen con el modelo ARQ_25.

Dado que este trabajo se centra en evaluar el rendimiento predictivo, la configuración seleccionada para llevar a cabo la comparativa entre modelos de AE y con otros métodos tradicionales de reducción de dimensionalidad ha sido ARQ_75. No obstante, es importante tener en cuenta las mejoras en cuanto a tiempo de ejecución que se obtienen a medida que decrece la dimensionalidad del modelo, ya que, en determinadas ocasiones, puede ser interesante sacrificar un mejor rendimiento predictivo en favor de un comportamiento más eficiente considerando tiempo de ejecución. En este sentido, se presenta la Tabla 4.3 que muestra los rankings agregados por arquitectura y ordenados según el tiempo de ejecución.

TABLA 4.3: Rankings considerando tiempo de ejecución de las diferentes arquitecturas de AE por clasificador.

kNN		SVM		MLP		C4.5	
Arquitectura	Ranking	Arquitectura	Ranking	Arquitectura	Ranking	Arquitectura	Ranking
ARQ_25	1.031	ARQ_25	1.025	ARQ_25	1.012	ARQ_25	1.037
ARQ_50	2.056	ARQ_50	2.075	ARQ_50	2.000	ARQ_50	2.068
ARQ_75	2.912	ARQ_75	3.031	ARQ_75	3.012	ARQ_75	2.981
Original	4.000	Original	3.868	Original	3.975	Original	3.912

4.2.2. Comparativa entre modelos de AE

Una vez realizada la selección de la mejor arquitectura para los tipos de AE, el siguiente paso consiste en determinar qué modelo tiene un mejor rendimiento a la hora de afrontar la tarea de reducción de dimensionalidad. La Figura 4.1 presentada en la Subsección 4.2.1 ya ha mostrado una visión general asociada al comportamiento de los distintos tipos de AE. No obstante, a continuación se realiza un análisis más detallado.

Para ello, en primer lugar se muestra la Tabla 4.4 que contiene cuatro rankings, uno para cada algoritmo de clasificación utilizado, donde se ordenan los distintos modelos de AE según el rendimiento predictivo. Además de

los tipos de AE, también se incluye el rendimiento obtenido a partir de los datos originales sin procesar. Esta clasificación se obtiene agregando todos los resultados de los distintos datasets utilizados para clasificar.

TABLA 4.4: Rankings de los diferentes modelos de *autoencoder* por método de clasificación.

kNN		SVM		MLP		C4.5	
Modelo	Ranking	Modelo	Ranking	Modelo	Ranking	Modelo	Ranking
RAE	1.750	RAE	2.025	RAE	1.750	RAE	1.700
DAE	2.400	DAE	2.200	CAE	2.400	CAE	2.450
CAE	2.700	CAE	2.850	DAE	3.100	DAE	3.000
BAE	4.000	BAE	3.850	BAE	3.500	Original	3.800
Original	4.150	Original	4.075	Original	4.250	BAE	4.050

A partir de la Tabla 4.4 se pueden extraer una serie de conclusiones sobre el comportamiento de los diferentes modelos de AE:

- En general, el modelo que mejores resultados proporciona es RAE (*robust*), que genera el mejor rendimiento para los cuatro clasificadores utilizados.
- Los modelos DAE (*denoising*) y CAE (contractive) suelen tener un comportamiento similar, alternando la segunda y tercera posición según el clasificador considerado.
- Entre los tipos de AE, el modelo básico (BAE) es el que ofrece peor rendimiento.
- Todos los modelos de AE, exceptuando el básico para C4.5, superan a los resultados obtenidos a partir de los datos originales sin procesar.

Estos resultados ponen de manifiesto que el hecho de afrontar la reducción de dimensionalidad mediante el uso de AE conlleva mejoras claras en cuanto a rendimiento predictivo. Así mismo, es posible confirmar que los modelos más sofisticados de AE (*contractive*, *robust* y *denoising*) producen unos resultados que mejoran claramente a los obtenidos a partir de los datos originales sin procesar. A pesar de que el modelo *robust* parece mostrar un comportamiento óptimo, en algunos casos no existen diferencias estadísticamente significativas entre estas tres técnicas más complejas. Por tanto, su elección dependerá del contexto del problema y las características asociadas a los datos de entrada. En la Sección 4.3 se ofrecen una serie de recomendaciones para facilitar su elección.

4.2.3. AE frente a técnicas tradicionales

Esta sección pretende estudiar la competitividad de los modelos más sofisticados de AE frente a algunos de los métodos tradicionales de reducción de dimensionalidad, en concreto cuatro de las técnicas más usadas y conocidas que han sido descritas en la Sección 2.2.1: PCA, LDA, ISOMAP y LLE.

En este sentido, a continuación se presenta la Tabla 4.5 que, al igual que en la tabla presentada en la sección anterior, contiene un ranking para cada uno de los clasificadores considerados, ordenados según el rendimiento predictivo generado por las distintas técnicas de reducción de dimensionalidad consideradas: los tres modelos de AE más sofisticados y los cuatro algoritmos tradicionales anteriormente indicados.

TABLA 4.5: Ranking considerando los modelos de reducción de dimensionalidad CAE, DAE, RAE, PCA, LDA, ISOMAP y LLE por clasificador.

kNN		SVM		MLP		C4.5	
Modelo	Ranking	Modelo	Ranking	Modelo	Ranking	Modelo	Ranking
RAE	1.925	RAE	2.600	RAE	2.250	RAE	2.025
DAE	2.700	DAE	3.150	CAE	3.025	CAE	3.075
CAE	2.750	CAE	3.550	DAE	4.125	DAE	3.275
LDA	4.475	LDA	4.200	ISOMAP	4.350	LDA	4.275
LLE	4.775	ISOMAP	4.275	LDA	4.425	LLE	4.750
ISOMAP	4.925	LLE	4.300	LLE	4.800	ISOMAP	4.950
PCA	6.450	PCA	5.925	PCA	5.025	PCA	5.650

En general, los rankings anteriores muestran que, desde un punto de vista global, los modelos de AE generan un mejor rendimiento predictivo. No obstante, se extraen algunas otras conclusiones que se enumeran a continuación.

- El modelo RAE (*robust*) genera los mejores resultados considerando los cuatro clasificadores.
- Los modelos *denoising* (DAE) y *contractive* (CAE) alternan la segunda y tercera posición según el clasificador considerado.
- Las técnicas LDA, ISOMAP y LLE tienen un rendimiento similar, alternando las posiciones cuarta, quinta y sexta, según el método de clasificación.
- El modelo PCA ofrece, en general, el peor rendimiento predictivo.

Los diferentes test estadísticos incluidos en el artículo completo (Anexo A.2) muestran que existen diferencias estadísticamente significativas entre los modelos anteriores, confirmando los resultados obtenidos.

En resumen, toda la experimentación llevada a cabo escenifica la competitividad de los modelos basados en AE para afrontar la reducción de dimensionalidad, comparando su rendimiento con algunas de las propuestas clásicas existentes en la literatura. Este mejor rendimiento puede estar asociado a que las características de alto nivel obtenidas mediante AE sean de mayor calidad y contengan información más relevante que las generadas mediante técnicas tradicionales.

4.3. Conclusiones

Esta sección pretende extraer conclusiones que permitan establecer relaciones entre las características de los datasets utilizados y los resultados obtenidos. De esta forma, es posible establecer recomendaciones dirigidas a potenciales usuarios de este tipo de modelos a la hora de seleccionar el más adecuado según el contexto del problema, es decir, los datos y el clasificador utilizado [Pulgar et al. (2020a)].

A continuación, se presenta la Tabla 4.6 que indica qué modelo de AE es el más adecuado según el número de características de los datos de entrada y el clasificador. Estas recomendaciones se basan en los resultados obtenidos para los conjuntos de datos utilizados en esta experimentación.

TABLA 4.6: *Autoencoder* recomendado según número de características y clasificador.

Número de características	Clasificador			
	kNN	SVM	MLP	C4.5
> 1000	RAE DAE	DAE	RAE CAE (elevado número de clases)	RAE
> 500 - < 1000	RAE CAE	RAE (no binario) DAE (binario)	RAE (no binario) DAE (binario)	RAE CAE (más de 10 clases)
> 100 - < 500	RAE CAE-DAE (binario)	RAE CAE (binario)	CAE RAE (binario)	RAE CAE (binario)
< 100	RAE CAE	RAE CAE	RAE CAE	RAE CAE

La Tabla 4.6 pone de manifiesto que, a pesar de que el modelo *robust* (RAE), ha mostrado en la sección anterior ser el que mejor rendimiento generaba, existen determinadas situaciones donde los modelos *contractive* (CAE) y *denoising* (DAE) funcionan mejor. Así mismo, es importante destacar que, en algunos casos, el número de clases del problema influye también en la elección del modelo más adecuado.

En conclusión, el estudio que se acaba de describir logra cumplir los objetivos inicialmente establecidos. Desde un punto de vista general, este análisis permite guiar a los usuarios potenciales de este tipo de modelos en la elección de la variante más adecuada según las características de los datos del problema y el clasificador utilizado. Para lograr este propósito genérico ha sido necesario cumplir varios objetivos específicos. En primer lugar, se ha determinado la arquitectura de AE más adecuada entre un conjunto preestablecido. Posteriormente, se ha analizado el comportamiento de diferentes modelos de AE, comparándolos entre sí, así como con los resultados obtenidos a partir de los datos sin procesar. Finalmente, para evaluar la competitividad de los AE, estos se han comparado con técnicas tradicionales de reducción de dimensionalidad. Todo esto ha llevado a verificar el buen rendimiento de este tipo de modelos a la hora de generar nuevas representaciones de los datos, así como a establecer una serie de recomendaciones a la hora de usarlos [Pulgar et al. (2020a)].

4.4. Publicaciones asociadas

La publicación integra asociada con el estudio descrito en el presente capítulo se puede consultar en la Sección A.2 del Apéndice A. Los datos de referencia de la misma son los siguientes [Pulgar et al. (2020a)]:

- **Título:** Choosing the proper autoencoder for feature fusion based on data complexity and classifiers: Analysis, tips and guidelines.

  Autores: Francisco J. Pulgar, Francisco Charte, Antonio J. Rivera, María J. del Jesus.

  Revista: Information Fusion, Volume 54, February 2020, Pages 44-60.

  ISSN: 1566-2535.

  DOI: 10.1016/j.inffus.2019.07.004.

Estado: Publicado.

Así mismo, también se ha realizado un análisis más específico de los modelos *denoising* que guarda gran relación con este trabajo [Pulgar et al. (2018a)]. La publicación asociada a este estudio se realizó en el congreso internacional *International Conference on Intelligent Data Engineering and Automated Learning* (IDEAL) y sus datos detallados son:

- **Título:** A First Approach to Face Dimensionality Reduction Through Denoising Autoencoders.

  Autores: Francisco J. Pulgar, Francisco Charte, Antonio J. Rivera, María J. del Jesus.

  Congreso: 19th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2018).

  Fecha y lugar: Noviembre 2018, Madrid.

  Páginas: 439-447.

Capítulo 5

Análisis del desbalanceo en DL

El presente capítulo tiene como objetivo el estudio, desde la perspectiva del aprendizaje profundo, de otro problema que afecta a muchos de los algoritmos de clasificación tradicionales, el desbalanceo de los datos.

En problemas reales es habitual que exista cierto grado de desequilibrio entre las clases del problema. Además, en la mayoría de estos casos las categorías de interés son las minoritarias. Por esta razón, los efectos del desbalanceo en los algoritmos de clasificación tradicionales han sido ampliamente estudiados en la literatura [Fernández et al. (2013), García et al. (2012) y Krawczyk et al. (2014)], tal y como se ha indicado en los capítulos anteriores. Sin embargo, la relación de este factor con algunas de las técnicas basadas en DL no ha sido analizado, dado que el auge de este tipo de métodos es reciente.

En este contexto se sitúa el estudio expuesto en el presente capítulo. El objetivo fundamental es presentar un análisis experimental de los efectos del desbalanceo de los datos de entrada en uno de los modelos DL más utilizados en la actualidad, las CNN. Para hacerlo, se ha diseñado un marco experimental compuesto por diferentes test donde se utiliza una misma CNN para clasificar varios conjuntos de datos con un grado de desbalanceo variable. En este sentido, se plantea la hipótesis de partida que establece que, al igual que ocurre con otros modelos tradicionales, el desbalanceo afectará negativamente al rendimiento predictivo de las CNN.

En el Capítulo 2 se han introducido algunos conceptos teóricos relacionados con las CNN. No obstante, la Sección 5.1 incluye una descripción más detallada. Así mismo, dicha sección presenta los detalles de la estructura de la red convolucional utilizada para clasificar y del conjunto de datos utilizado en

la experimentación. La Sección 5.2 resume los resultados obtenidos. La Sección 5.3 contiene el análisis general y las conclusiones alcanzadas. Finalmente, la Sección 5.4 enumera las contribuciones relacionadas con este estudio.

5.1. Análisis del impacto de datos desbalanceados en el rendimiento predictivo de redes neuronales convolucionales

Esta sección está dividida en varios apartados cuyo objetivo fundamental es contextualizar el estudio descrito en el presente capítulo. En la Subsección 5.1.1 se profundiza en el concepto de CNN, introducido en la Sección 2.6, describiendo de forma más pormenorizada los distintos componentes de este tipo de redes. La Subsección 5.1.2 muestra la arquitectura de CNN utilizada en la experimentación. Finalmente, la Subsección 5.1.3 detalla el conjunto de datos experimental empleado en este estudio.

5.1.1. Redes neuronales convolucionales

Durante el proceso de funcionamiento de una CNN, cada una de las capas transforma el volumen de entrada mediante la aplicación de una función concreta. A continuación se describen, desde un punto de vista más específico, las diferentes fases propias del proceso de aprendizaje de las CNN, así como sus capas características y su funcionamiento básico.

5.1.1.1. Entrada

La primera fase propia de la arquitectura de las CNN tiene lugar a través de la capa de entrada de datos, donde la red recibe la información del exterior para comenzar el proceso de aprendizaje. Los datos que manejan las CNN se distribuyen en tres dimensiones. Como ejemplo se consideran imágenes como entrada de la red, dado que es uno de los formatos más frecuentemente usados en este tipo de modelos [LeCun et al. (2010)]. En estos casos, las tres dimensiones se corresponderían con el ancho, el alto y las bandas de color propias de una imagen. Así, si el tamaño de las imágenes es de 32 píxeles de alto y 32 de ancho y se tiene una sola banda de color (escala de grises), la

capa de entrada estará compuesta por 1024 unidades. En este mismo caso, si en lugar de una banda de color se tiene una imagen RGB con tres bandas, el número de unidades sería igual a 3072. De esta forma, cada uno de los valores de los píxeles considerando las bandas existentes será asignado a una neurona de la capa de entrada.

Aunque se ha indicado que la primera fase de este tipo de modelos se corresponde con la entrada de datos, es importante destacar que en ocasiones es necesario tratar los valores iniciales antes de comenzar el proceso, para ello pueden utilizarse algunas de las técnicas de pre-procesamiento descritas en la Sección 2.1.2.

La Figura 5.1 muestra dos ejemplos de posibles formatos de entrada propios de las CNN. Por un lado, se muestra un volumen de entrada de una dimensión, propio, por ejemplo, de las imágenes en escala de grises (1 banda de color) y, por otro lado, una entrada con tres dimensiones, correspondiente a imágenes RGB (3 bandas de color), entre otros tipos de posibles formatos.

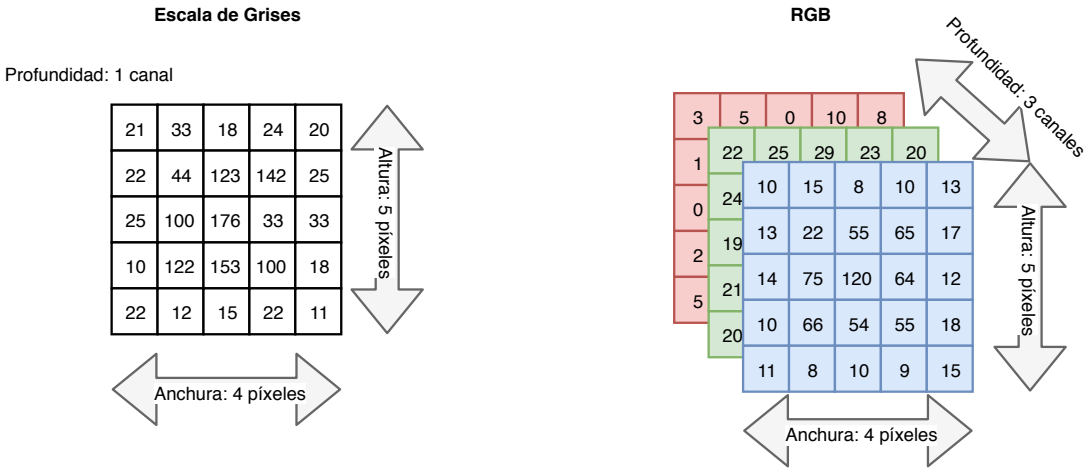


Figura 5.1. Ejemplos de formatos de de entrada en CNN: escala de grises (izqda.) y RGB (dcha.).

5.1.1.2. Capa de convolución

El proceso de convolución constituye la operación característica de las CNN. Fundamentalmente, esta fase consiste en realizar operaciones matemáticas sencillas que involucren a pequeñas regiones de píxeles adyacentes [LeCun et al. (2010)]. En concreto, a partir de un conjunto de *kernels* o filtros de convolución, se realizan una serie productos escalares que generan las

matrices de salida de cada capa. El número de filtros que se apliquen es variable constituyendo uno de los parámetros de la estructura de la red. No obstante, según el número de filtros que se apliquen en una capa, el tamaño y complejidad de la salida aumentará o disminuirá. Por ejemplo, si se aplican 24 filtros distintos sobre una matriz de entrada, se obtendrán 24 matrices de salida, conocidas como *feature mapping*. Cada una de estas matrices de salida puede ser vista como una nueva imagen que acentúa determinadas características de la entrada y que ayudará a poder identificar determinados objetos complejos.

Los parámetros que la red ajusta o aprende durante el proceso de aprendizaje en la capa de convolución son los distintos filtros que se aplican. Dichos filtros tienen, en general, unas dimensiones espaciales pequeñas, siendo comunes los tamaños 3x3 o 5x5, pero se extienden a lo largo de la profundidad del volumen de entrada. Por ejemplo, si se procesan imágenes RGB con tres canales de entrada y se utilizan filtros 3x3, el bloque de *kernels* que se aprende en cada una de las capas será de 3x3x3. Es importante destacar que, en este ejemplo, el número de atributos aprendidos por la red habrá que multiplicarlo por el número de filtros que se aplican en la capa considerada. Así, si en la capa anterior se aplican 24 filtros, se deben ajustar 3x3x3x24 parámetros relativos a los filtros de convolución durante el proceso de aprendizaje.

En este contexto, la convolución consiste en desplazar cada uno de los filtros a través del volumen de entrada a lo ancho (de izquierda a derecha) y del alto (de arriba a abajo), y calcular los productos escalares entre la matriz del filtro y los elementos de la entrada relativos a cada una de las posibles posiciones. En este punto, es importante destacar que el filtro se desplaza de acuerdo a un valor de paso (*stride*) que determina la cantidad de elementos que se avanza hacia la derecha o hacia abajo en cada desplazamiento.

A medida que el filtro se desplaza espacialmente sobre la matriz de entrada, se producirá un mapa de activación bidimensional que contiene las respuestas del filtro sobre cada una de las posiciones recorridas. Finalmente, el valor de salida para una unidad concreta consistirá en la suma de los valores obtenidos tras aplicar el filtro correspondiente sobre cada uno de los niveles de profundidad y un parámetro adicional conocido como *bias*, obteniendo el mapa de características (*feature mapping*) final.

De esta forma, tras aplicar cada uno de los filtros propios de una capa convolucional se producirá un mapa de activación bidimensional independiente. La unión o apilación de todos estos mapas de características genera el volumen de salida de la capa de convolución [LeCun y Bengio (1995)].

Durante la fase de *back-propagation*, las red aprenderá a ajustar los distintos parámetros para optimizar la función objetivo. En lo relativo a las capas convolucionales, el modelo ajustará los valores asociados a los filtros anteriormente descritos. Así, de forma intuitiva, la red será capaz de aprender filtros que se activan cuando se da en la entrada algún tipo de característica especial, como pueden ser elementos sencillos como un borde o un área de un color concreto en las primeras capas, u objetos complejos, como animales o caras, en las últimas capas de la red [LeCun y Bengio (1995)].

La Figura 5.2 ilustra de forma gráfica el proceso anteriormente descrito propio de la fase de convolución. En este caso, únicamente se muestra la aplicación de uno de los filtros, esta operación es similar para todos los que se utilicen. De esta forma, las diferentes matrices generadas por todos los filtros aplicados componen el volumen de salida de la capa de convolución. Los valores de dichos filtros son los que el modelo aprende o ajusta durante el proceso de aprendizaje.

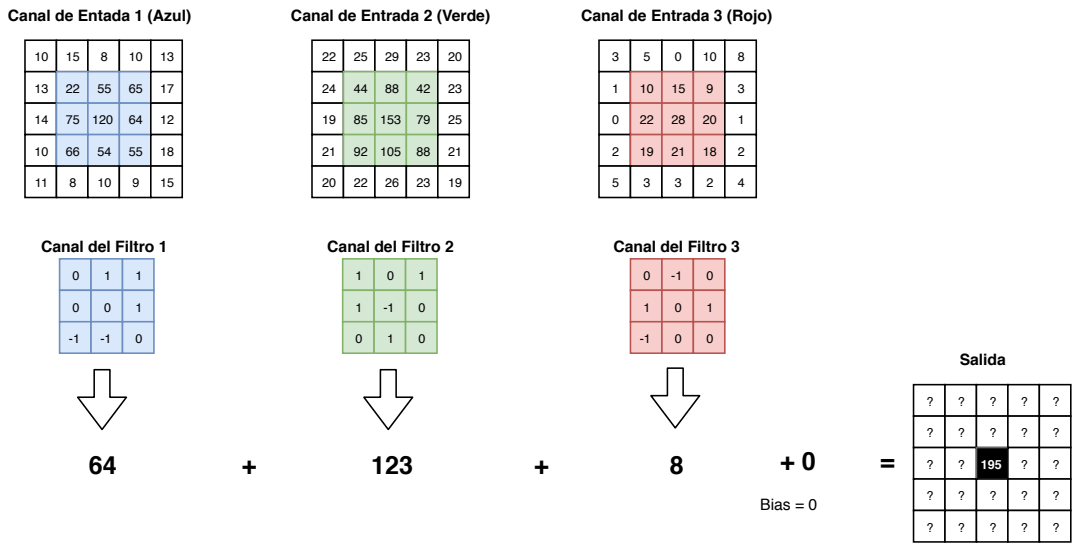


Figura 5.2. Procesamiento realizado en la primera capa de convolución de una CNN asociado a uno de los filtros utilizados y considerando una entrada en formato RGB.

Esta operación es la base del proceso de aprendizaje propio de las CNN. Como se ha indicado, cada unidad de la capa de convolución realiza una operación sobre la región de la imagen correspondiente. De forma matemática, la salida de cada neurona viene determinada por la siguiente expresión (Eq. (5.1)):

$$S_j = f\left(b_j + \sum_i F_{ij} \otimes E_i\right) \tag{5.1}$$

Donde S_j es la salida de la unidad, b_j se corresponde con el parámetro *bias*, K_{ij} son los distintos valores del filtro y E_i es la entrada de la neurona. Por tanto, la salida de la capa de convolución se puede tratar como una combinación lineal de las entradas de dicha capa y el *kernel* de convolución correspondiente.

Las capas de convolución van seguidas, habitualmente, de una capa adicional conocida como RELU, que aplica la función de activación característica de los modelos basados en redes neuronales. Una de las funciones más utilizadas se conoce como RELU [Glorot et al. (2011) y Gülçehre y Bengio (2016)], de ahí el nombre dado a esta capa. Esta función viene dada por la expresión (Eq. (5.2)):

$$f(x) = max(0, x) \tag{5.2}$$

Donde x es la entrada de la función y la salida será 0 si dicho valor es menor o igual que 0.

En definitiva, el objetivo fundamental de esta operación de convolución es extraer propiedades de alto nivel de los datos de entrada, estas pueden ser, como se ha indicado antes, aristas, esquinas o curvas existentes en una imagen original. En este punto, es importante destacar que las redes convolucionales no se limitan a una única capa de convolución, sino que pueden existir varias capas de este tipo. De esta forma, las primeras capas convolucionales capturan información de menor nivel de abstracción o complejidad, mientras que las capas más profundas obtienen características de mayor nivel de abstracción o más complejas, sirviéndose de la información proporcionada por las capas anteriores. Finalmente, la red es capaz de identificar, por ejemplo, el objeto representado en una determinada imagen. Este proceso es similar al usado por los humanos para identificar objetos basándose en sus componentes más sencillos.

5.1.1.3. Capa de *pooling*

Las capas de *pooling*, junto con las de convolución descritas anteriormente, son las más características de las CNN. El objetivo principal de esta fase es reducir el tamaño espacial de los mapas de características obtenidos como salida de las capas de convolución. En caso de no aplicar este tipo de capas, la complejidad (número de neuronas) del modelo crecería de forma drástica y el proceso de entrenamiento requeriría un procesamiento mucho más costoso computacionalmente. Esto justifica el hecho de llevar a cabo un proceso de *subsampling* en el que se reducirá la dimensión del volumen de salida de las capas convolucionales. No obstante, es importante tener en cuenta que tras aplicar esta fase deben prevalecer las características de alto nivel más relevantes que el modelo haya detectado en las capas anteriores. Así, esta fase puede contribuir también a extraer dichas propiedades más dominantes, manteniendo la efectividad del modelo a pesar de la reducción de la dimensionalidad llevada a cabo. Además de las dos ventajas que se acaban de describir, esta fase tiene otros efectos positivos, por ejemplo, al reducir el número de elementos y conexiones se disminuye la posibilidad de que el modelo se sobre-ajuste a los datos.

El proceso de *subsampling* consiste en, a partir de un filtro de *pooling* o *kernel*, aplicar una función entre los elementos cubiertos por dicho filtro. Existen dos tipos fundamentales de operaciones llevadas a cabo en las capas de *pooling*: máximo (*Max Pooling*) y media (*Average Pooling*). Por un lado, la operación máximo devuelve el mayor valor entre las unidades involucradas. Por otro lado, la función media devuelve el valor promedio de los valores de la entrada cubiertos por el filtro.

En general, la operación *max pooling* funciona mejor que *average pooling*, ya que al seleccionar el máximo este proceso actúa también como supresor del posible ruido existente. Este factor hace que la función máximo sea la más frecuentemente utilizada [LeCun et al. (2010)].

A continuación se plantea un ejemplo de aplicación del filtro de *pooling* con el objetivo de aclarar su funcionamiento. Los *kernels* utilizados en esta capa suelen corresponderse con matrices cuadradas de dimensión pequeña, en este caso se considerará un filtro de 2x2, el cual se desplaza a lo largo de la dimensión de entrada de acuerdo a un valor de paso (al igual que en la convolución) que habitualmente coincide con la dimensión del filtro, por tanto en el ejemplo considerado será 2. El filtro se aplica de forma independiente a

lo largo de los diferentes niveles de profundidad de la entrada, por lo que la dimensión relativa a la misma no se ve afectada por esta operación. En este punto, es necesario determinar la función de *pooling* que se va a utilizar, en este ejemplo, se toma el máximo. Por tanto, cada operación de *pooling* tomará cuatro unidades (correspondientes a una región 2x2 de cualquiera de los segmentos de profundidad) y seleccionará el máximo valor entre ellas como salida. Una vez realizado este proceso para toda la entrada, se obtendrán un volumen de salida de menor dimensionalidad, en concreto, se descartarán el 75 % de las unidades iniciales.

La Figura 5.3 representa de forma gráfica el proceso llevado a cabo en las capas de *pooling* que se acaba de describir. Si el volumen de entrada de una capa de *pooling* es de 32x32x24 y se aplica un filtro de tamaño 2x2 con paso 2, la salida que se obtendrá tendrá una dimensión de 16x16x24.

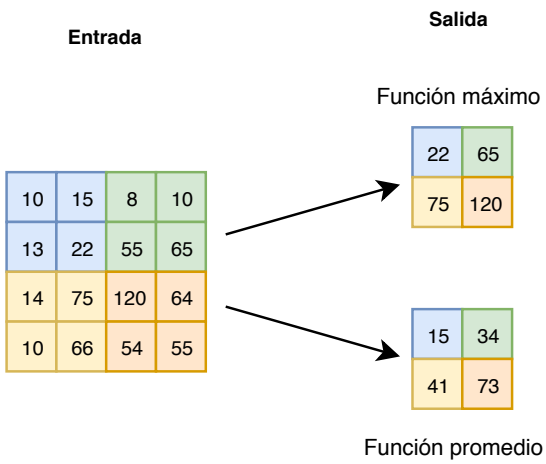


Figura 5.3. Procesamiento realizado sobre uno de los niveles de entrada en una de las capas de *pooling* que componen una CNN, considerando las funciones máximo y promedio.

Esta operación es fundamental para reducir la capacidad de cómputo necesaria para llevar a cabo el proceso de aprendizaje propio de las CNN. Como se ha indicado, cada unidad de la capa de *pooling* realiza una operación sobre la región de la entrada ocupada por el filtro utilizado. De forma matemática, la salida de cada neurona, considerando la operación de *pooling* máximo y un filtro 2x2, viene determinada por la siguiente expresión (Eq. (5.3)):

$$S_j = max\left(E_{i,j}, E_{i+1,j}, E_{i,j+1}, E_{i+1,j+1}\right) \tag{5.3}$$

Donde S_j es la salida de la unidad, $E_{x,y}$ se corresponde con cada uno de los valores del volumen de entrada abarcados por el filtro de *pooling*, en este caso, al utilizar filtros 2x2 se tienen 4 elementos.

Dependiendo de la complejidad de las imágenes de entrada, un número variable de pares formados por una capa convolucional y una capa de *pooling* formarán la arquitectura de la red. No obstante, a medida que el número de capas se incremente, la complejidad del modelo crecerá y, con ello, el coste computacional asociado.

5.1.1.4. Capa totalmente conectada

Una vez llevado a cabo el proceso asociado a los pares de capas convolucionales y *pooling* que tienen el objetivo de extraer características de alto nivel a partir de los datos de entrada, es necesario agregar toda la información obtenida para generar la salida final del modelo [LeCun et al. (2010)]. Esto se lleva a cabo mediante capas totalmente conectadas. La dimensionalidad de la primera capa dependerá del número y tamaño de filtros del último par de capas de convolución y pooling, mientras que la dimensionalidad de la última capa dependerá de la tarea a realizar. Por ejemplo, si la red se usa para clasificar, esta última capa tendrá tantos elementos como clases tenga el problema.

Fundamentalmente, la utilización de este tipo de capas tiene como objetivo que la red aprenda combinaciones no lineales entre las características de alto nivel extraídas por las capas previas. Para ello, se utilizan capas con una estructura y funcionamiento similar a las utilizadas en las redes neuronales tradicionales, es decir, están formadas por un conjunto de unidades o neuronas que tienen conexiones con todos los elementos de las capas siguientes [LeCun et al. (2010)]. Finalmente, la última capa totalmente conectada está asociada a la capa de salida de la red, habitualmente mediante una función conocida como *Softmax*. La dimensionalidad de la salida será, como se ha indicado anteriormente, igual al número de clases que tenga el problema al afrontar la tarea de clasificación.

En este contexto, la salida de una unidad que forma parte de una capa totalmente conectada viene determinada de forma matemática por la siguiente expresión (Eq. (5.4)):

$$S_j \;=\; f\big(b_j + \sum_i w_{ij} \otimes E_i\big) \tag{5.4}$$

Donde S_j es la salida de la unidad, b_j se corresponde con el parámetro *bias*, w_{ij} son los pesos asociados a las distintas conexiones de la neurona y E_i es la entrada de la misma. Por tanto, la salida de la capa totalmente conectada es una combinación lineal de todas las conexiones de entrada asociadas a cada unidad.

En conclusión, el objetivo fundamental de las capas totalmente conectadas es la agregación de toda la información o características de alto nivel generadas por las capas de convolución y *pooling* previas, aplanando el volumen tridimensional generado en dichas capas y obteniendo una salida unidimensional, asociada directamente al contexto del problema tratado por la red.

5.1.1.5. Proceso de *back-propagation*

La fase de *back-propagation* es fundamental dentro del proceso de aprendizaje propio de cualquier tipo de red neuronal [Rumelhart et al. (1986)]. Al describir las capas de convolución ya se ha presentado brevemente su aplicación dentro de las CNN.

Esta operación se basa en una serie de entradas y salidas esperadas que se utilizan como datos de entrenamiento, de ahí que se trate de aprendizaje supervisado. En estos casos, la salida del modelo una vez entrenado es comparada con la salida prevista y el error cometido en la predicción se propaga hacia atrás a través de la red para ajustar todos los pesos y optimizar el resultado. En el caso de las redes convolucionales, durante esta fase se ajustan los valores asociados a los distintos filtros de convolución, así como, los pesos de las conexiones existentes en las capas totalmente conectadas.

En este sentido, el hecho de que las capas de convolución y *pooling* involucren a regiones concretas y limitadas (generalmente pequeñas) de la entrada implica que el número de conexiones existentes es menor y, por tanto, se reduce el procesamiento necesario asociado al método de *back-propagation*.

5.1.2. Arquitectura CNN utilizada

Esta sección se centra en describir la arquitectura de la red convolucional utilizada en este estudio. En este sentido, un aspecto fundamental es tener en cuenta que, para poder detectar fielmente los efectos del desbalanceo, la arquitectura utilizada debe ser la misma en todos los casos, es decir, los cuatro subconjuntos generados deben ser clasificados utilizando la misma configuración de CNN.

En concreto, la estructura de ejemplo mostrada en la Sección 2.6 se corresponde, tal y como se indicó, a la arquitectura propia de la CNN utilizada en el presente estudio:

- **Capa de entrada:** en este caso su composición viene determinada por el tamaño de las imágenes utilizadas (32x32 con 3 canales de color).
- **Convolución+*pooling* 1:** se aplican 32 filtros de convolución de tamaño 5x5, lo que genera un volumen de salida de tamaño 28x28x32 (32 mapas de características de 28x28). A continuación, se reduce la dimensionalidad de dicha salida aplicando un filtro de *pooling* de tamaño 2x2 y con paso 2, lo que genera una salida de tamaño 14x14x32.
- **Convolución+*pooling* 2:** sobre la salida de la capa anterior se aplican 64 filtros de convolución de tamaño 5x5, obteniendo 64 mapas de características de tamaño 10x10. Seguidamente, se vuelve a aplicar un filtro de *pooling* con las mismas características que anteriormente para reducir la dimensionalidad, generando una salida de tamaño 5x5x64.
- **Capa totalmente conectada:** una vez acabada la fase de convolución, es necesario agregar toda la información generada. Para ello, se utiliza una capa totalmente conectada cuyas unidades tienen conexiones con todas las neuronas de la capa anterior.
- **Capa de salida:** esta capa genera la salida final del clasificador, por ello tendrá tantos elementos como clases o categorías tenga el problema, en este caso 10 (5 clases mayoritarias y 5 clases minoritarias). Así mismo, estará totalmente conectada con la capa anterior.

La estructura descrita anteriormente está caracterizada, como todas las CNN, por dividirse en una zona de convolución, donde se extraen las características de alto nivel, y una serie de capas totalmente conectadas, donde se realiza la clasificación. La Figura 5.4 presenta una visión gráfica de la estructura de la CNN.

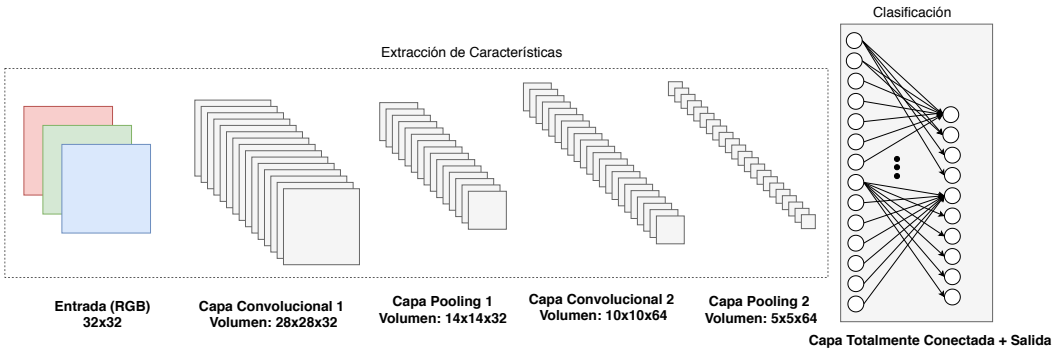


Figura 5.4. Ejemplo de arquitectura de CNN.

5.1.3. Conjunto de datos experimental

La experimentación desarrollada en el estudio descrito en el presente capítulo tiene como objetivo analizar los efectos del desbalanceo en el rendimiento predictivo obtenido mediante CNN. Por esta razón, se llevan a cabo diferentes test donde se utilizan datos con distinto grado de desbalanceo.

En concreto, para llevar a cabo la experimentación se ha partido de un dataset con imágenes reales de señales de tráfico, llamado *German Traffic Sign Benchmark* [Stallkamp et al. (2012)]. El conjunto de datos está formado por 51 839 imágenes (39 209 de entrenamiento y 12 630 de test), cada una de ellas pertenece a una única clase (problema de clasificación multi-clase) entre las 43 posibles. La Figura 5.5 muestra algunos ejemplos de las imágenes del dataset.

Este dataset presenta un alto desequilibrio entre algunas de sus clases, llegando a tener una proporción de 1 a 10 entre las clases mayoritarias y minoritarias. Sin embargo, la utilización del dataset completo, en el que existen muchas clases intermedias, puede hacer que los efectos del desbalanceo no sean fácilmente detectables. Por esta razón, se ha optado por obtener distintos subconjuntos con grado de desbalanceo variable a partir del dataset original. El proceso seguido para obtener dichos subconjuntos se resume a continuación:

Figura 5.5. Dataset *German Traffic Sign Benchmark* [Stallkamp et al. (2012)].

1. **Pre-procesamiento:** las imágenes son recortadas para eliminar el fondo y seleccionar únicamente la señal de tráfico, y escaladas para unificar el tamaño de todas ellas.

2. **Selección:** en primer lugar, se seleccionan las 5 clases con más ejemplos y las 5 con menos ejemplos, obteniendo el subconjunto de partida con mayor grado de desbalanceo. En la Tabla 5.1 se pueden observar las clases seleccionadas marcadas en rojo y azul, siendo el ratio de desbalanceo entre ellas de 1/10.

3. **Número de imágenes por test:** se determina el número de imágenes que deben contener los distintos subconjuntos, para evitar que una cantidad variable pueda influir en los resultados obtenidos.

4. **Creación de los subconjuntos:** utilizando la selección de clases realizada anteriormente y el número de imágenes que deben tener los subconjuntos, se crean de forma proporcional las distintas particiones de datos con grado de desbalanceo diferente, en concreto: 1/10 (grado inicial), 1/5, 1/3 y 1/1 (totalmente balanceado). En los distintos casos se descartan aleatoriamente ejemplos de la clases mayoritarias o minoritarias, según corresponda. Los subconjuntos generados se pueden observar en la Tabla 5.2.

TABLA 5.1: Número de ejemplos por clase del conjunto de entrenamiento del dataset, destacando las clases seleccionadas.

Clase	Número de ejemplos	Clase	Número de ejemplos
1	210	23	390
2	2 220	24	510
3	2 250	25	270
4	1 410	26	1 500
5	1 980	27	600
6	1 860	28	240
7	420	29	540
8	1 440	30	270
9	1 410	31	450
10	1 470	32	780
11	2 010	33	240
12	1 320	34	689
13	2 100	35	420
14	2 160	36	1 200
15	780	37	390
16	630	38	210
17	420	39	2 070
18	1 110	40	300
19	1 200	41	360
20	210	42	240
21	360	43	240
22	330	Total	39 210

TABLA 5.2: Ejemplos de entrenamiento y test de cada subconjunto.

	IR 1/10		IR 1/5		IR 1/3		IR 1/1	
Clase	Train	Test	Train	Test	Train	Test	Train	Test
1	36	11	66	20	98	31	203	67
2	351	115	320	106	288	95	203	67
3	392	130	361	118	325	106	203	67
4	365	121	334	111	301	100	203	67
5	376	125	345	115	311	103	203	67
6	36	11	65	21	97	32	203	67
7	39	13	72	24	108	36	203	67
8	36	11	65	21	97	32	203	67
9	360	120	330	110	297	99	203	67
10	39	13	72	24	108	36	203	67
Total	2 030	670	2 030	670	2 030	670	2 030	670

En resumen, este proceso ha permitido obtener cuatro subconjuntos de datos con distinto grado de desbalanceo. En la Tabla 5.2 se puede comprobar que todos ellos tienen el mismo número de imágenes total (2 030 de entrenamiento y 670 de test), pero el desequilibrio entre clases disminuye desde 1/10 hasta llegar al conjunto totalmente balanceado (1/1).

5.2. Experimentación y resultados

La presente sección pretende ofrecer los principales resultados obtenidos en la experimentación. Para ello, se ofrecen en primer lugar, la Tabla 5.3 que incluye los ejemplos de test y los errores cometidos en cada clase de los conjuntos de datos con ratio de desbalanceo 1/10, 1/5, 1/3 y 1/1. Así mismo, la Tabla 5.4 muestra el rendimiento predictivo generado por la CNN con cada uno de dichos conjuntos.

TABLA 5.3: Número de instancias total y errores en test por conjunto de datos.

	IR 1/10		IR 1/5		IR 1/3		IR 1/1	
Clase	Test	Error	Test	Error	Test	Error	Test	Error
1	11	4	20	4	31	2	67	1
2	115	1	106	3	95	2	67	0
3	130	1	118	2	106	2	67	3
4	121	2	111	0	100	2	67	0
5	125	0	115	2	103	0	67	0
6	11	4	21	2	32	2	67	0
7	13	2	24	0	36	0	67	0
8	11	4	21	1	32	0	67	0
9	120	1	110	1	99	1	67	1
10	13	3	24	0	36	0	67	3
Total	670	22	670	15	670	11	670	8

TABLA 5.4: Rendimiento predictivo por conjunto de datos.

	IR 1/10			IR 1/5			IR 1/3			IR 1/1		
Clase	Error	Precision	Recall	Error	Precision	Recall	Error	Precision	Recall	Error	Precision	Recall
1	0.364	**1.000**	0.636	0.200	**1.000**	0.800	0.065	**1.000**	0.935	**0.015**	0.985	**0.985**
2	0.009	0.966	0.991	0.028	0.954	0.972	0.021	0.989	0.979	**0.000**	**1.000**	**1.000**
3	**0.008**	0.963	**0.992**	0.017	0.951	0.983	0.019	0.990	0.981	0.045	**1.000**	0.955
4	0.017	0.983	0.983	**0.000**	**1.000**	**1.000**	0.020	0.990	0.980	**0.000**	0.985	**1.000**
5	**0.000**	0.977	**1.000**	0.017	0.983	0.983	**0.000**	0.956	**1.000**	**0.000**	**0.985**	**1.000**
6	0.364	0.875	0.636	0.095	**1.000**	0.905	0.062	0.968	0.937	**0.000**	0.985	**1.000**
7	0.154	0.917	0.846	**0.000**	0.960	**1.000**	**0.000**	0.947	**1.000**	**0.000**	**1.000**	**1.000**
8	0.364	**1.000**	0.636	0.048	**1.000**	0.952	**0.000**	**1.000**	**1.000**	**0.000**	0.985	**1.000**
9	**0.008**	0.952	**0.992**	0.009	0.991	0.991	0.010	**1.000**	0.990	0.015	0.956	0.985
10	0.231	**1.000**	0.769	**0.000**	**1.000**	**1.000**	**0.000**	**1.000**	**1.000**	0.045	**1.000**	0.955
Media	0.033	0.963	0.848	0.022	0.984	0.958	0.016	0.984	0.980	**0.012**	**0.988**	**0.988**

Adicionalmente, a continuación se ofrece de forma gráfica la tasa de error obtenida al clasificar mediante la misma CNN los ejemplos de test de los cuatro conjuntos de datos con ratio de desbalanceo 1/10, 1/5, 1/3 y 1/1. Para ello, la Figura 5.6 representa la tasa de error obtenida para cada una de las clases del problema.

En general, los resultados por clase presentes tanto en las Tablas 5.3 y 5.4 como en la Figura 5.6 muestran que el mejor rendimiento para cada una de las clases se obtiene con el conjunto de datos totalmente balanceado. No obstante,

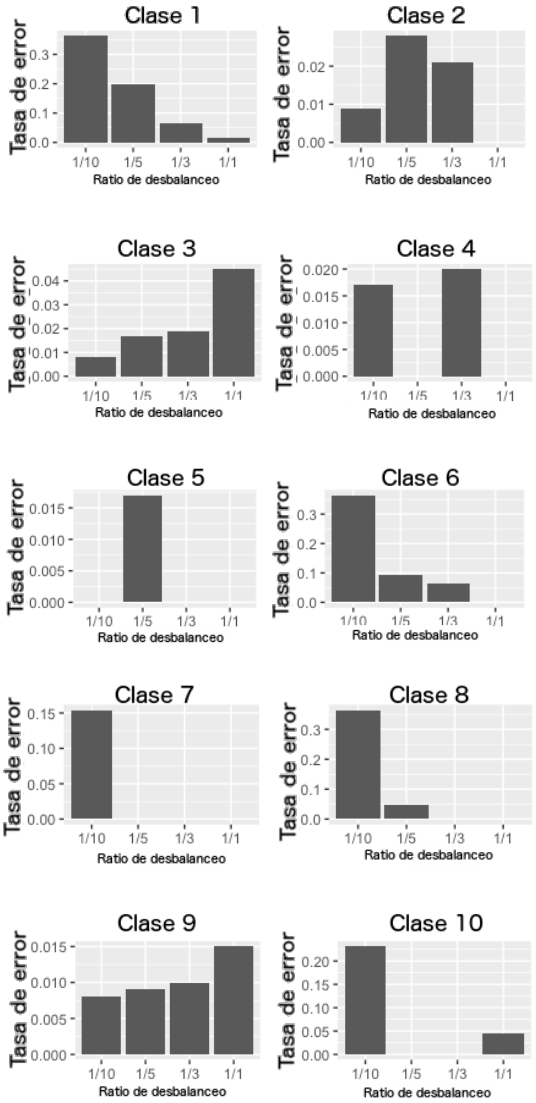


Figura 5.6. Tasa de Error por clase y experimento. Dataset: *German Traffic Sign Benchmark*.

existen ciertas excepciones como son la clase 3 y 9. Estas clases, si se observa la Tabla 5.1, se puede apreciar que son mayoritarias, esto implica que a medida que el conjunto de datos se balancea, el número de ejemplos de estas clases disminuye, tal y como se ha indicado en la Tabla 5.2. Por tanto, la red tiene menos ejemplos de estas clases lo que implica que el rendimiento predictivo decae. Sin embargo, esto no ocurre con todas las clases mayoritarias, véanse las clases 2 o 5. En cuanto a las clases minoritarias, el comportamiento es mucho más uniforme, mejorando claramente el rendimiento predictivo a medida que el desequilibrio entre clases decrece.

Así mismo, desde un punto de vista global, el rendimiento unificando todas las clases mejora claramente a medida que decrece el desbalanceo, tal y como se puede observar en la Figura 5.7.

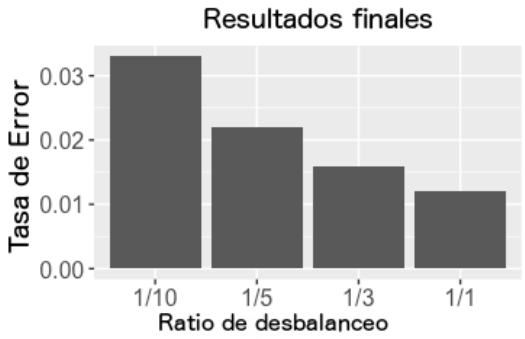


Figura 5.7. Tasa media de Error por experimento.

En definitiva, estos datos confirman la hipótesis de partida establecida en este estudio, es decir, el desbalanceo en los datos de entrada afecta negativamente al rendimiento predictivo de las CNN.

5.3. Conclusiones

Tras mostrar una visión global de los resultados obtenidos, es posible extraer un conjunto de conclusiones generales asociadas a los mismos que se enumeran a continuación:

- **Visión general:** la CNN tiene un mejor rendimiento predictivo a medida que el conjunto de datos utilizado está más balanceado, lo que demuestra que el desbalanceo afecta negativamente a este tipo de modelos, al igual que ocurre con otros métodos de clasificación tradicionales.
- **Clases minoritarias:** el comportamiento de las clases minoritarias es uniforme, es decir, todas estas clases tienen un comportamiento similar. En concreto, se puede observar que el rendimiento predictivo de estos casos aumenta al diminuir el desequilibrio entre clases. El motivo fundamental es que en los conjuntos más balanceados, además de que la distribución es más homogénea, el número de ejemplos de estas clases aumenta con respecto a los subconjuntos con un mayor desbalanceo.

- **Clases mayoritarias:** estas clases presentan un comportamiento más variable, ya que en determinados casos el rendimiento con los dataset desbalanceados es mayor (clases 3 y 9) y, en otros casos, el mejor comportamiento se obtiene con el dataset totalmente balanceado (clases 2 y 5). Los efectos apreciados en las clases 3 y 9 pueden estar asociados al hecho de que, a medida que se balancea el dataset, el número de imágenes utilizadas de estas clases concretas disminuye con respecto a los dataset con mayor grado de desbalanceo. Por tanto, la CNN dispone de menos cantidad de información para entrenarse, lo que afecta al rendimiento de estas clases.

En este punto, es importante destacar que el comportamiento asociado a las clases minoritarias es especialmente interesante, ya que, en muchos de los problemas reales donde está presente el desbalanceo en los datos de entrada, la clase que presenta un mayor interés a la hora de clasificar es la minoritaria, teniendo un gran coste en caso de no hacerlo correctamente. En este estudio, se ha mostrado que, al balancear el dataset, el rendimiento predictivo obtenido en estas clases mejora considerablemente.

Finalmente, para tratar de determinar las causas que conducen a estos resultados, es necesario detectar cuáles son los principales elementos propios de las CNN que podrían verse afectados por el desbalanceo. Esta búsqueda es fundamental para poder proponer mejoras en este tipo de modelos que mitiguen los efectos del desequilibrio entre clases. Estas propiedades podrían ser:

- **Capa totalmente conectada y salida:** estas capas que agregan toda la información anterior para realizar la clasificación final podrían dar mayor importancia a las clases mayoritarias, debido a la existencia de muchos más ejemplos de entrenamiento.
- **Pesos filtros de convolución:** estos parámetros se ajustan en las distintas capas de convolución para detectar las características más representativas de los datos de entrada. Dado que los ejemplos de las clases mayoritarias son mucho más numerosos, estos pesos se podrían ajustar para detectar las propiedades de alto nivel asociadas a dichas clases, en detrimento de otras que permitan identificar a las clases minoritarias.

No obstante, esto es sólo una breve aproximación a las posibles causas relacionadas con la estructura de las CNN que pueden influir en los efectos negativos asociados al uso de datos desbalanceados. En este sentido, es fundamental realizar un análisis más detallado para determinar con exactitud qué elementos propios de la arquitectura de la CNN tienen una mayor influencia en estos resultados.

En resumen, el estudio llevado a cabo pone de manifiesto la importancia de utilizar métodos que mitiguen los efectos del desbalanceo, abriendo vías de trabajo futuro en este sentido, ya sea aplicando métodos tradicionales, como el muestreo de datos o los aprendizajes sensibles al coste, o nuevas propuestas que combinen dichos métodos con las técnicas DL generando modelos híbridos.

5.4. Publicaciones asociadas

El estudio que se presenta en este capítulo está asociado a dos contribuciones realizadas en congresos, uno de ellos internacional y otro nacional. El primero tiene los siguientes datos [Pulgar et al. (2017)]:

- **Título:** On the Impact of Imbalanced Data in Convolutional Neural Networks Performance.

  Autores: Francisco J. Pulgar, Antonio J. Rivera, Francisco Charte, María J. del Jesus.

  Congreso: 12th International Conference on Hybrid Artificial Intelligent Systems (HAIS 2017).

  Fecha y lugar: Junio 2017, Logroño.

  Páginas: 220–232.

Así mismo, a continuación se muestran los datos asociados a la segunda de las contribuciones [Pulgar et al. (2018c)]:

- **Título:** Análisis del impacto de datos desbalanceados en el rendimiento predictivo de redes neuronales convolucionales.

  Autores: Francisco J. Pulgar, Antonio J. Rivera, Francisco Charte, María J. del Jesus.

Congreso: XVIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2018).

Fecha y lugar: Octubre 2018, Granada.

Páginas: 1213–1218.

Capítulo 6

Ensembles de arquitecturas profundas

El trabajo expuesto en el presente capítulo sigue la misma línea mostrada en los Capítulos 3 y 4 donde el objetivo principal es mitigar los efectos de la alta dimensionalidad presente en los datos de entrada.

En concreto, en este estudio se propone un nuevo método, llamado ClEn-DAE, un clasificador basado en *ensembles* que incorpora DAE para reducir la dimensionalidad del espacio de entrada [Pulgar et al. (2020b)]. Así, esta propuesta está basada en dos pilares básicos: DAE y *ensemble*.

Por un lado, el uso de DAE permite reducir la dimensionalidad del espacio de entrada. Estos modelos se encargan de aprender una nueva representación de la entrada de más alto nivel pero de menor dimensionalidad. En general, los AE han mostrado ser una técnica adecuada para afrontar el problema de la reducción de dimensionalidad [Charte et al. (2018) y Pulgar et al. (2020a)]. No obstante, existen diferentes modelos de AE, en este trabajo se ha optado por utilizar DAE. El motivo fundamental es que este tipo de AE introduce ruido en la entrada y, posteriormente, trata de aprender a reconstruir la entrada original, eliminado dicho ruido. Esto permite que la nueva representación generada sea mucho más robusta y de mayor calidad. Este modelo ha sido descrito con mayor detalle en la Sección 2.5.

Por otro lado, el modelo propuesto está basado en *ensembles* con el propósito de mejorar el rendimiento predictivo, ya que, en general, la utilización de varios componentes trabajando conjuntamente produce mejores resultados que si se utilizan de forma aislada e individual. Este concepto ha sido explicado en detalle en la Sección 2.3.

Una vez descritos brevemente los dos conceptos claves del modelo propuesto, es importante detallar el método de clasificación que se utiliza. En este sentido, el algoritmo ClEnDAE incorpora cuatro núcleos de clasificación distintos basados en cuatro algoritmos clásicos: kNN, SVM, MLP y C4.5. De esta forma, es posible seleccionar el método cuando se utiliza ClEnDAE a partir de uno de sus parámetros. Así mismo, esto permite evaluar el rendimiento del método propuesto considerado distintos algoritmos de clasificación tradicionales.

Además de la descripción del nuevo algoritmo, el presente trabajo incorpora una exhaustiva experimentación para evaluar su comportamiento. Para ello se realizan dos comparativas independientes. Por un lado, se estudia el rendimiento de ClEnDAE frente a un modelo basado en la utilización de DAE de forma individual y los resultados obtenidos utilizando los datos sin procesar. Por otro lado, el método ClEnDAE es comparado con cuatro algoritmos clásicos de reducción de dimensionalidad: PCA, LDA, ISOMAP y LLE. Los resultados demuestran que ClEnDAE obtiene mejoras significativas en comparación con el resto de modelos considerados.

En resumen, el método ClEnDAE propuesto en este trabajo combina la información proporcionada por los distintos componentes del *ensemble* para clasificar. Cada uno de ellos afronta la reducción de dimensionalidad mediante DAE y la clasificación utilizando cuatro algoritmos clásicos. En concreto, las principales contribuciones de este estudio son las siguientes:

- El desarrollo de un nuevo clasificador, ClEnDAE, basado en *ensembles* y que incorpora DAE para reducir la dimensionalidad del espacio de entrada.
- Una exhaustiva experimentación para verificar el comportamiento de ClEnDAE frente a un modelo DAE individual y a los datos originales sin procesar.
- Una comparativa entre ClEnDAE y cuatro algoritmos clásicos de reducción de dimensionalidad: PCA [Pearson (1901)], LDA [Yu et al. (2001)], ISOMAP [Tenenbaum (2000)]y LLE [Roweis y Saul (2000)].
- La extracción de una serie de conclusiones generales en base a los resultados obtenidos en la experimentación llevada a cabo.

Las siguientes secciones se centran en describir el método desarrollado, así como la experimentación realizada. La contextualización y descripción de los principales conceptos teóricos asociados se encuentra en el Capítulo 2. Así mismo, en la Sección 6.1 se describe el método ClEnDAE. La Sección 6.2 presenta los principales resultados de la experimentación diseñada para evaluar el comportamiento de ClEnDAE. En la Sección 6.3 se enumeran las principales conclusiones alcanzadas. Finalmente, la Sección 6.4 contiene los detalles de la referencia asociada a la aportación descrita en el presente capítulo [Pulgar et al. (2018b)], la cual puede ser consultada en la Sección A.3 del Apéndice A.

6.1. Propuesta de *ensemble* de *denoising autoencoder*: ClEnDAE

Esta sección describe la principal aportación descrita en este capítulo, el método ClEnDAE. Esta nueva propuesta es un clasificador basado en *ensembles* que incorpora reducción de dimensionalidad mediante el uso de DAE. Las siguientes subsecciones exponen de forma detallada dicho método: la Subsección 6.1.1 contiene la base teórica del mismo, la Subsección 6.1.2 describe el método propuesto y la Subsección 6.1.3 presenta las principales contribuciones asociadas a ClEnDAE.

6.1.1. Base teórica ClEnDAE

Este trabajo parte de la hipótesis de que el desarrollo de un clasificador basado en *ensembles* que incorpore DAE para reducir la dimensionalidad permitirá obtener mejoras en cuanto a rendimiento predictivo frente a clasificadores tradicionales. Esto se basa en el buen rendimiento de los DAE para reducir la dimensionalidad y en las ventajas asociadas al uso de los *ensembles*, ambos aspectos descritos en el Capítulo 2.

A partir de las ideas asociadas a las diferentes metodologías basadas en *ensembles* descritas en la Sección 2.3, se ha propuesto ClEnDAE. En concreto, este método realiza una selección aleatoria tanto de instancias como de características para entrenar los diferentes componentes del *ensemble*. Finalmente,

la salida de todos ellos se combina para generar la salida final del algoritmo. Para ello, se utiliza el voto mayoritario, es decir, el resultado será la salida generada por el mayor número de componentes.

El número de componentes de un *ensemble* depende de muchos factores, incluyendo las propiedades de los datos de entrada y el número de clases de salida [Bonab y Can (2019)]. En este trabajo, la misma configuración es utilizada en todos los datasets, ya que el objetivo es determinar desde un punto de vista general el rendimiento del modelo propuesto con un conjunto variable de datasets. Por esta razón, se ha optado por utilizar 90 componentes, ya que este valor representa un número significativo de unidades para cada una de las tres arquitecturas de DAE consideradas. Sin embargo, a la hora de utilizar ClEnDAE para resolver un problema concreto, el número de componentes debe ajustarse a las características de los datos para optimizar los resultados obtenidos.

Así, el modelo ClEnDAE está compuesto por 90 componentes, cada uno de ellos contiene un DAE interno para reducir la dimensionalidad del espacio de entrada. En las siguientes etapas, el nuevo espacio de características es utilizado por el clasificador seleccionado para generar la salida del componente. Cada uno de los componentes utiliza un espacio de características distinto generado por su propio DAE. De esta forma, el proceso asociado a ClEnDAE tiene tres fases fundamentales:

- **Reducción de dimensionalidad:** a partir del subconjunto aleatorio obtenido a partir de la entrada, cada componente genera un nuevo espacio de características de menor dimensionalidad usando varios DAE con diferentes arquitecturas.
- **Clasificación:** el nuevo espacio de características generado en la fase anterior es usado para generar la salida del componente mediante diferentes algoritmos de clasificación.
- **Salida:** la salida final del modelo se corresponde con la salida mayoritaria generada por los componentes del *ensemble*.

En este punto, es importante destacar que, como se ha indicado anteriormente, las arquitecturas de los DAE utilizados en los componentes de ClEnDAE son variables. En concreto, se utilizan tres arquitecturas con distinto grado de reducción de dimensionalidad: 30 componentes que aplican una compresión al 75 %, 30 que reducen al 50 % y 30 que lo hacen al 25 %. El hecho de utilizar varias arquitecturas permite ampliar el espacio de búsqueda, ya

que cada tipo de modelo aprende sus propias características dependientes de la arquitectura. A continuación, se describe de forma más detallada el algoritmo ClEnDAE.

6.1.2. Descripción ClEnDAE

El funcionamiento del algoritmo ClEnDAE está compuesto por una serie de etapas que se describen a continuación. El primer paso consiste en la selección aleatoria del subconjunto utilizado en cada componente obtenido a partir de conjunto original. La segunda fase se corresponde con el entrenamiento de cada una de las 90 unidades del *ensemble*, tanto del DAE para aprender una nueva representación de la entrada como del clasificador que genera la predicción. El tercer y último paso consiste en generar la salida final del modelo agregando la información proporcionada por todos los componentes. Este proceso es representado visualmente en la Figura 6.1, donde se puede apreciar las distintas fases descritas. Así mismo, el Algoritmo 3 muestra el pseudo-código asociado al método ClEnDAE.

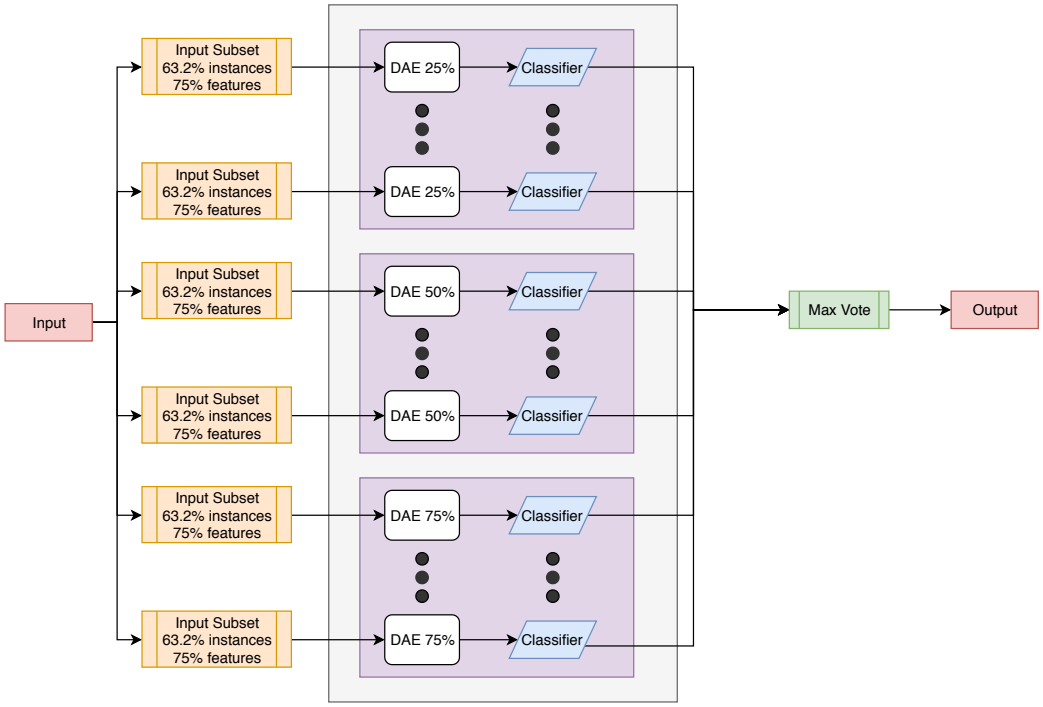


Figura 6.1. Esquema de funcionamiento de AEkNN.

Algoritmo 3 Pseudo-código del algoritmo ClEnDAE.

```
Inputs:
    TrainData                ▷ Datos de entrenamiento
    TestData                 ▷ Datos de test
    NumComponent             ▷ Número de componentes del ensemble
    ArqDAE                   ▷ Arquitectura del DAE
    Clas                     ▷ Clasificador
    Par                      ▷ Parámetros del clasificador
▷ Entrenamiento:
ensemble ← initEnsemble(NumComponent, ArqDAE)
for each component in ensemble do
    ▷ Selección de instancias y características:
    componentData ← randomSelection(TrainData)
    ▷ Entrenar DAE:
    daeModel ← trainDAE(componentData, component)
    ▷ Nueva representación de los datos mediante DAE:
    componentData ← applyDAE(componentData, daeModel)
    ▷ Entrenar clasificador:
    clasModel ← trainClassifier(componentData, Clas, Par)
    ▷ Actualizar componente:
    ensemble ← updateComponent(component, daeModel, clasModel)
end for

▷ Clasificación:
result ← classification(TestData, ensemble)
return result

function TRAINDAE(componentData, component)
    modelData ← corruptedInput(componentData)
    daeModel ← ()
    for each layer in component do
        sizeLayer ← getSizeLayer(modelData, layer)
        daeModel ← addDAELayer(sizeLayer)
    end for
    daeModel ← compileDAEModel(modelData, daeModel)
    component ← addDAEModel(daeModel)
    return component
end function

function COMPILEDAEMODEL(modelData, daeModel)
    for each instance in modelData do
        outPut ← feedForwardDAE(daeModel, instance)
        error ← calculateDeviation(instance, outPut)
        daeModel ← updateWeightsDAE(daeModel, error)
    end for
    return daeModel
end function

function CLASSIFICATION(TestData,ensemble)
    error ← 0
    for each instance in TestData do
        outputs ← ()
        for each component in ensemble do
            newRep ← applyDAE(instance, component)
            componentOutput ← applyClassifier(newRep, component)
            outputs ← addOutput(outputs, componentOutput)
        end for
        finalOutput ← selectMaxorityOutput(outputs))
        error ← updateError(instance, outPut))
    end for
    result ← calculateError(TestData, error)
    return result
end function
```

Los parámetros de entrada que se han indicado en Algoritmo 3 son los siguientes:

- *TrainData y TestData*: los conjuntos de datos de entrenamiento y test procesados por el algoritmo.
- *NumComponent*: el número de componentes que forman el *ensemble*. En este estudio se han utilizado 90, tal y como se ha especificado anteriormente.
- *ArqDAE*: las arquitecturas utilizadas por los DAE. En este trabajo se han utilizado tres arquitecturas distintas, por tanto cada una de ellas será utilizada en 30 componentes.
- *Clas*: el clasificador usado en la fase de clasificación del algoritmo. En este caso existen cuatro opciones: kNN, SVM, MLP y C4.5.
- *Par:* los parámetros asociados al algoritmo de clasificación anteriormente seleccionado.

El método descrito en Algoritmo 3 muestra tanto la fase de entrenamiento del modelo (líneas 1-14) como la fase de clasificación de las instancias de test (líneas 16-18).

El proceso de entrenamiento se divide a su vez en una serie de pasos que persiguen varios objetivos. Por un lado, reducir la dimensionalidad del espacio de entrada mediante DAEs y, por otro, entrenar el método encargado de realizar la clasificación de las nuevas instancias. A continuación se describe de forma más pormenorizada dicho código:

- **Inicialización del *ensemble* (línea 2):** se crea la estructura asociada al modelo y todos los componentes que lo forman. En este trabajo se consideran 90 componentes con tres arquitecturas distintas.
- **Entrenamiento de los componentes (líneas 3-14):** cada modelo que forma el *ensemble* se entrena de forma independiente. Este proceso tiene cinco pasos fundamentales: selección aleatoria del subconjunto a partir de la entrada (línea 5), entrenamiento del DAE (línea 7), obtención de una nueva representación de menor dimensionalidad (línea 9), entrenamiento del clasificador (línea 11) y actualización del componente (línea 13). En este sentido, es importante destacar que existen métodos, como kNN, que no requieren entrenamiento, en estos casos únicamente se inicializa el modelo con los parámetros indicados (línea 11).
- **Función de entrenamiento del DAE (líneas 20-30)**: esta función se corresponde con el proceso de entrenamiento de los DAE. El primer paso es introducir ruido en la entrada (línea 21). Posteriormente, se inicializa

el modelo mediante un bucle en el que cada iteración añade una nueva capa de la estructura (líneas 23-26). Finalmente, se entrena el DAE con el propósito de actualizar los pesos para optimizar la función objetivo (línea 27, función definida en líneas 32-39) y se actualiza el componente (línea 28).

El segundo bloque (líneas 16-18) se corresponde con la clasificación de las instancias de test. Este proceso consiste en predecir la clase de cada uno de los ejemplos contenidos en el conjunto de evaluación (parámetro *TestData*). Esto se realiza mediante la función *classification* definida en las líneas 41-55, en ella existe un bucle donde cada componente del *ensemble* genera su propia salida (línea 45-49) y, finalmente, se selecciona la mayoritaria (línea 50), a partir de la cual se obtiene el error cometido (línea 51). El proceso de clasificación de cada *ensemble* tiene los siguientes pasos:

1. Utilizando el DAE se obtienen una nueva representación de la instancia de entrada de menor dimensionalidad (línea 46).
2. El clasificador se aplica para generar la salida a partir de la nueva representación obtenida anteriormente (línea 47).
3. Se almacena la salida proporcionada por el componente para poder determinar posteriormente la salida mayoritaria (línea 48).

Finalmente, es importante detallar varios aspectos relacionados con el proceso descrito anteriormente. En primer lugar, el método de entrenamiento se lleva a cabo aplicando *mini-batch gradient descent* [Hinton (2010)], al igual que en el método AEkNN descrito en el Capítulo 3. Otro aspecto importante es que, tal y como se ha indicado, todas las arquitecturas usadas tienen una única capa oculta, debido a que diferentes estudios han demostrado que, en general, funcionan mejor a la hora de afrontar la reducción de dimensionalidad [Pulgar et al. (2018b), Pulgar et al. (2018a) y Pulgar et al. (2020a)]. Así mismo, es importante especificar el número de ejemplos y características seleccionados en cada uno de los modelos que forman el *ensemble*. En concreto, cada componente selecciona el 63,2 % de las instancias de entrenamiento y el 75 % de las características de entrada. El hecho de seleccionar el 63,2 % de los ejemplos asegura que todas las instancias son seleccionadas al menos una vez [Bauer y Kohavi (1999) y Quinlan (1996)]. En definitiva, este muestreo aleatorio permite ampliar el espacio de búsqueda, diversificando la obtención de características de alto nivel por parte de los diferentes modelos.

6.1.3. Contribuciones ClEnDAE

Las principales aportaciones de este trabajo se enumeran a continuación:

- El método incorpora DAE para reducir la dimensionalidad del espacio de entrada. Esto se traduce en mejoras en cuanto a rendimiento predictivo frente a algoritmos de clasificación tradicionales.
- ClEnDAE está basado en *ensembles*, lo que implica que el rendimiento del modelo donde varios componentes trabajan de forma conjunta supere a los métodos individuales utilizados de forma aislada.
- Cada componente del *ensemble* se entrena con un conjunto aleatorio tanto de instancias como de atributos, así se diversifica el espacio de búsqueda, mejorando los resultados generados.
- El algoritmo está parametrizado para utilizar cuatro clasificadores diferentes: kNN, SVM, MLP y C4.5. Esto permite evaluar el comportamiento de ClEnDAE considerando cuatro de los métodos de clasificación más conocidos y utilizados.

Estas contribuciones ponen de manifiesto la principales novedades aportadas por el método propuesto. A continuación, con el objetivo de demostrar los beneficios aportados por ClEnDAE frente a otros métodos, se propone la experimentación descrita en la Sección 6.2.

6.2. Experimentación y resultados

Esta sección describe de forma resumida la experimentación cuyo objetivo es verificar las mejoras asociadas al algoritmo ClEnDAE, el estudio detallado asociado se puede encontrar en el artículo indicado en la Sección 6.4 [Pulgar et al. (2020b)].

Esta experimentación se centra en determinar si el método ClEnDAE mejora tanto el rendimiento obtenido utilizando DAE de forma aislada como al generado a partir de los datos originales sin procesar. Así mismo, es necesario determinar si el método propuesto mejora a algunos de los algoritmos tradicionales de reducción de dimensionalidad: PCA [Pearson (1901)], LDA [Yu et al. (2001)], ISOMAP [Tenenbaum (2000)]y LLE [Roweis y Saul (2000)]. Estos dos requisitos constituye las dos fases de la experimentación descrita en las Subsecciones 6.2.1 y 6.2.2.

Finalmente, es importante indicar que el marco experimental asociado a la experimentación está descrito en destalle en el artículo [Pulgar et al. (2020b)]. En este sentido, uno de los aspectos más relevantes es la utilización de un conjunto formado por 20 datasets con características muy diversas. Así mismo, la métrica utilizada para evaluar el rendimiento predictivo asociado a cada método es el área bajo la curva ROC (AUC). Además, cabe destacar que se han utilizado distintos test estadísticos para verificar todos los resultados obtenidos.

6.2.1. Análisis de ClEnDAE por clasificador

El objetivo de esta sección es analizar el rendimiento de ClEnDAE frente al generado utilizando DAE de forma aislada y a través de los datos originales sin procesar utilizando cuatro clasificadores clásicos. De esta forma es posible estudiar cómo afecta la utilización de ClEnDAE al rendimiento predictivo de dichos métodos de clasificación.

Los resultados obtenidos en este apartado se encuentra de forma integra en el artículo incluido al final de la presente memoria (Anexo A.3). En general, se puede apreciar que los resultados generados por el algoritmo ClEnDAE son los mejores en la mayor parte de los casos. A continuación se enumeran algunas conclusiones destacadas a nivel de clasificador:

- **kNN:** el método ClEnDAE obtiene el mejor rendimiento predictivo en 17 de los 20 casos analizados, mientras que en dos de los tres casos restantes el mejor resultado se obtiene a partir de los datos originales sin procesar. Existe un dataset donde se produce empate entre varios modelos.
- **SVM:** en este caso, el algoritmo ClEnDAE genera los mejores resultados en 16 de los 20 casos, mientras que en otros dos casos se obtienen a partir de los datos originales sin procesar. En el resto de los casos se producen empates entre métodos.
- **MLP:** al igual que en el método anterior, ClEnDAE genera el mejor rendimiento predictivo en 16 de los 20 casos, en dos de los restantes el mejor resultado se obtiene con los datos originales y en los otros dos se producen empates.

- **C4.5:** los resultados de ClEnDAE son los mejores en 18 de los 20 dataset utilizados, mientras que en los dos casos restantes el mejor rendimiento se obtiene a través de los datos originales sin procesar.

Un detalle importante en que en todos los casos el método ClEnDAE mejora (o empata) a los resultados obtenidos utilizado DAE básico, lo que demuestra que la utilización de varios componentes de forma conjunta mejora a los resultados obtenidos mediante los modelos utilizados aisladamente. Así mismo, el método basado en DAE básico mejora a los resultados obtenidos a partir de los datos sin procesar en la mayoría de los casos, lo que pone de manifiesto los beneficios del uso de DAE par reducir la dimensionalidad. Finalmente, es importante destacar que los test estadísticos que se pueden observar en el artículo completo (Anexo A.3) verifican la existencia de diferencias significativas entre ClEnDAE y el resto de métodos.

En resumen, estos resultados permiten determinar que existe una clara mejora en cuanto a rendimiento predictivo cuando se utiliza el método ClEnDAE. El algoritmo propuesto obtiene los mejores resultados en más del 80 % de los casos analizados. Esto se produce fundamentalmente por el uso de DAE para reducir la dimensionalidad del espacio de entrada y por la utilización de distintos componentes que trabajan de forma conjunta para generar el resultado final. Ambos aspectos (DAE y *ensemble*) constituyen, como se ha explicado, las bases principales del método propuesto.

6.2.2. ClEnDAE frente a algoritmos clásicos de reducción de dimensionalidad

En la sección anterior se ha mostrado la efectividad del método frente a un modelo basado en DAE básico y la utilización de los datos sin procesar. Sin embargo, es necesario analizar de igual forma el comportamiento de ClEnDAE frente a algoritmos clásicos de reducción de dimensionalidad como PCA, LDA, ISOMAP y LLE. Este es el objetivo principal de la presente sección.

En este punto, es importante destacar que el número de características generadas por los 4 algoritmos de clasificación debe ser la misma, en concreto se obtendrán el 75 % de las características originales. Este valor debe ser común para establecer una comparación fiable y extraer las conclusiones adecuadas. Finalmente, los nuevos espacios de características generados por

los algoritmos de reducción de dimensionalidad serán utilizados para clasificar utilizando los métodos: kNN, SVM, MLP y C4.5, al igual que en la sección anterior, para poder analizar el comportamiento asociado a cuatro de los clasificadores tradicionales más utilizados.

Los resultados de clasificación asociados a los distintos métodos de reducción de dimensionalidad se pueden consultar en el artículo incluido en el Anexo A.3. Desde un punto de vista general se puede decir que los mejores resultados se obtienen con el método ClEnDAE en la mayor parte de los casos. Así mismo, los resultados por clasificador se describen a continuación:

- **kNN**: el algoritmo ClEnDAE genera el mejor rendimiento predictivo en 16 de los 20 datasets analizados. Los métodos LDA, ISOMAP y LLE generan los mejores resultados en un dataset cada uno, mientras que el método PCA no lo hace en ningún caso. Existe un caso donde tanto ClEnDAE como LLE tienen el mejor comportamiento. Es decir, la representación obtenida mediante ClEnDAE es la mejor en el 80 % de los casos, sin tener en cuenta posibles empates.
- **SVM**: la propuesta realizada en este estudio genera los mejores resultados en 16 de 20 casos analizados, mientras que LLE tiene el mejor rendimiento en 3 de los casos restantes. Existe un dataset en el que varios algoritmos presentan un comportamiento similar. En resumen, al igual que en el caso anterior, el método ClEnDAE gana en el 80 % de los casos (sin contar empates).
- **MLP**: en este caso el método ClEnDAE presenta el mejor rendimiento en el 63 % de los casos analizados. En concreto, ClEnDAE gana en 13 de 20 datasets, LDA en 2 de 20 y PCA, ISOMAP y LLE en un caso cada uno de ellos. Así mismo, existen dos datasets donde varios métodos tienen rendimiento predictivo equivalente.
- **C4.5**: el algoritmo ClEnDAE genera los mejores resultados en 16 de 20 datasets, LDA en 3 de 20 e ISOMAP en 1 de 20. Los métodos PCA y LLE no tienen rendimiento óptimo en ningún caso. Esto se traduce en que ClEnDAE mejora al resto de algoritmos en el 80 % de los casos estudiados.

Los resultados descritos anteriormente asociados a los cuatro algoritmos de clasificación presentan ciertas similitudes. En general, el rendimiento predictivo de ClEnDAE es mejor que el obtenido mediante PCA, LDA, ISOMAP y LLE. Esto significa que el nuevo espacio generado a partir del proceso de

fusión de características llevado a cabo por ClEnDAE es de mayor calidad que el obtenido a partir del resto de métodos, es decir, las nuevas características generadas proporcionan información más relevante y útil para el proceso de clasificación posterior. Esta tendencia se confirma en los resultados detallados que se pueden consultar en el artículo asociado al presente capítulo [Pulgar et al. (2020b)] (Anexo A.3), donde se muestran los test estadísticos que confirman la existencia de diferencias significativas entre ClEnDAE y el resto de métodos analizados.

En definitiva, los resultados descritos en la presente sección permiten verificar los beneficios de utilizar el método ClEnDAE propuesto frente a otros métodos clásicos de reducción de dimensionalidad. Por tanto, ClEnDAE es una buena alternativa para mejorar el rendimiento cuando los datos del problema presenten alta dimensionalidad.

6.3. Conclusiones

Este trabajo propone un nuevo método, ClEnDAE, un clasificador basado en *ensembles* que incorpora DAE para reducir la dimensionalidad del espacio de entrada.

ClEnDAE está basado en dos pilares fundamentales. Por un lado, es un método basado en *ensembles*. Esto contribuye a mejorar el rendimiento predictivo, ya que disponer de varios componentes trabajando de forma conjunta mejora con respecto a la utilización de los modelos de forma aislada. Por otro, cada componente utiliza internamente un DAE para reducir la dimensionalidad del espacio de entrada. Este tipo de redes contribuye a mitigar los efectos de la alta dimensionalidad de los datos de entrada. Otro aspecto a destacar es que el núcleo de clasificación utilizado por los distintos componentes es variable, pudiendo utilizar los algoritmos tradicionales kNN, SVM, MLP y C4.5.

La efectividad del nuevo método propuesto ha sido estudiada a través de un exhaustivo análisis experimental. Fundamentalmente, esta fase consiste en evaluar el comportamiento de ClEnDAE frente a otros modelos utilizando los cuatro clasificadores anteriormente citados. En primer lugar se compara ClEnDAE con el rendimiento obtenido a partir de los datos generados por un DAE aislado y con los datos sin procesar. En segundo lugar, se realiza una comparativa entre ClEnDAE y cuatro propuestas clásicas de reducción de dimensionalidad: PCA, LDA, ISOMAP y LLE.

En relación con la primera parte de la experimentación, los resultados obtenidos han mostrado que ClEnDAE genera un mejor rendimiento predictivo que el modelo basado en DAE utilizado de forma aislada y los datos originales sin procesar. En concreto, ClEnDAE obtiene los mejores resultados en más del 72 % de los casos analizados.

La segunda parte de la experimentación ha mostrado que ClEnDAE tiene, en general, un mejor comportamiento que PCA, LDA, ISOMAP y LLE. En concreto, ClEnDAE genera los mejores resultados en el 80 % de los casos para kNN, SVM y C4.5 y en el 65 % para MLP. Esto pone de manifiesto la efectividad del modelo propuesto frente a cuatro de los métodos más utilizados en este contexto.

En definitiva, el trabajo expuesto en el presente capítulo presenta un nuevo clasificador, ClEnDAE, que incorpora mecanismos para reducir la dimensionalidad del espacio de entrada. La experimentación llevada a cabo confirma la utilidad del método propuesto y abre nuevas vías de trabajo futuro, como puede ser, entre otras, la aplicación de ClEnDAE a la resolución de problemas específicos o el desarrollo de nuevos métodos similares utilizando otros tipos de AE.

6.4. Publicaciones asociadas

El estudio descrito a lo largo del presente capítulo se encuentra de forma integra en la Sección A.3 del Apéndice A, se corresponde con un artículo que está sometido y en revisión, cuyos datos son los siguientes [Pulgar et al. (2020b)]:

- **Título:** ClEnDAE: A classifier based on ensembles with built-in dimensionality reduction through denoising autoencoders.

  Autores: Francisco J. Pulgar, Francisco Charte, Antonio J. Rivera, María J. del Jesus.

  Revista: Information Sciences.

  ISSN: 0020-0255.

  Estado: En revisión.

Capítulo 7

Conclusiones y trabajos futuros

El objetivo de este capítulo final es ofrecer, desde una perspectiva global, una visión del trabajo realizado durante la presente tesis doctoral. En primer lugar, la Sección 7.1 describe las principales aportaciones realizadas en relación con los objetivos establecidos inicialmente. A continuación, en la Sección 7.2 se enumeran todas las publicaciones relacionadas con los distintos trabajos presentados. Finalmente, la Sección 7.3 presenta las principales líneas de investigación abiertas de cara a posibles trabajos futuros.

La primera fase del proceso se centró en realizar una exhaustiva revisión de la bibliografía relacionada con el tema tratado: la aplicación de arquitecturas profundas a los problemas de alta dimensionalidad y desbalanceo en clasificación. El objetivo fundamental de esta primera fase es tener una visión completa y profunda del estado del arte en la temática estudiada, para poder determinar posibles cuestiones abiertas y de interés para la comunidad científica en las que trabajar para ofrecer nuevas soluciones.

Tras esta primera fase, fue posible establecer un objetivo general: analizar el uso de arquitecturas profundas y *ensembles* para mejorar la clasificación con datos desbalanceados y con alta dimensionalidad. A partir de dicho propósito general se definieron una serie de objetivos específicos, los cuáles se enumeran a continuación:

- Proponer modelos de clasificación que incorporen reducción de dimensionalidad mediante técnicas de DL.
- Analizar y caracterizar las distintas propuestas basadas en DL existentes para reducción de dimensionalidad.
- Analizar el desbalanceo en arquitecturas profundas de aprendizaje.

- Desarrollar clasificadores mediante *ensembles* de arquitecturas profundas.

Esta tesis doctoral ha tratado de cubrir los objetivos que se acaban de enumerar, a continuación se pretende ofrecer una visión general de los diferentes estudios realizados y su relación con los objetivos anteriores.

7.1. Conclusiones

Cada uno de los objetivos enumerados anteriormente han generado como resultado un conjunto de análisis experimentales y nuevos métodos de clasificación. Estas aportaciones han sido descritas a lo largo de la memoria, donde cada objetivo se corresponde con un capítulo independiente. A continuación, se describen los principales resultados obtenidos.

7.1.1. Tratamiento de alta dimensionalidad con técnicas DL

En el Capítulo 3 se ha propuesto un nuevo método de clasificación que incorpora técnicas basadas en aprendizaje profundo para reducir la dimensionalidad del espacio de entrada. En concreto, el método, llamado AEkNN, es un clasificador basado en kNN que utiliza AE para generar un nuevo espacio de características reducido.

El hecho de desarrollar un modelo basado en kNN se debe a que los algoritmos basados en distancias como kNN se ven muy afectados por la alta dimensionalidad de los datos, ya que en este escenario las distancias entre los ejemplos tienden a igualarse y son menos significativas, lo que se traduce en un rendimiento predictivo menor. Por esta razón, AEkNN incorpora una fase interna de reducción de dimensionalidad basada en AE antes de afrontar la clasificación con kNN.

En resumen, las principales aportaciones descritas en el Capítulo 3 son las siguientes:

- Presentación y descripción del nuevo clasificador, AEkNN.
- Análisis detallado de la parametrización de AEkNN que permite la selección de la mejor arquitectura entre un conjunto inicialmente establecido.

- Una demostración experimental de las mejoras obtenidas con AEkNN frente a kNN clásico.
- Una comparativa que verifica que AEkNN genera mejor rendimiento predictivo que PCA y LDA, métodos clásicos de reducción de dimensionalidad.
- Una serie de recomendaciones dirigidas a posibles usuarios del método AEkNN a la hora de configurar sus parámetros en base a la experimentación realizada.

En definitiva, AEkNN, frente a otros métodos basados en distancias, presenta la ventaja de mitigar los efectos de la alta dimensionalidad presente en el espacio de entrada. Esto se traduce en una clara mejora en cuanto a rendimiento predictivo.

7.1.2. Estudio sobre la relación entre *autoencoder*, método de aprendizaje y problema

El Capítulo 4 presenta un estudio cuyo objetivo es la realización de un análisis exhaustivo de distintos modelos basados en aprendizaje profundo que afrontan la tarea de reducción de dimensionalidad. En este sentido, los AE son los modelos basados en DL más utilizados para esta tarea. Por ello, este trabajo se centra en analizar cuatro de las variantes más conocidas. En concreto, los tipos de AE considerados son: *básico*, *denoising*, *contractive* y *robust*.

Para evaluar esto se han utilizado cuatro métodos de clasificación: kNN, SVM, MLP y C4.5. De esta forma, es posible establecer qué tipo de AE es más adecuado según las propiedades de los datos de entrada y el clasificador utilizado posteriormente. Así mismo, el rendimiento de los AE se ha comparado con el de cuatro métodos de reducción de dimensionalidad clásicos: PCA, LDA, ISOMAP y LLE.

Los resultados obtenidos en la experimentación confirman que el uso de AE contribuyen a mitigar los efectos de la alta dimensionalidad presente en los datos de entrada. En general, el rendimiento de los cuatro clasificadores con los datos obtenidos a partir de los distintos tipos de AE mejora considerablemente al generado a partir de los datos originales sin procesar. En concreto, los AE más sofisticados y complejos (*robust*, *contractive* y *denoising*) ofrecen los mejores

resultados. Así mismo, la experimentación demuestra que el rendimiento de los AE supera claramente al de otros métodos clásicos como PCA, LDA, ISOMAP y LLE.

Finalmente, como resultado de la amplia experimentación llevada a cabo, se propone una serie de líneas de actuación o recomendaciones dirigidas a usuarios potenciales de AE aplicados a reducción de dimensionalidad. En concreto, dichas conclusiones relacionan las características de los datos de entrada y el clasificador utilizado con el modelo de AE que proporcionará los resultados óptimos. De esta forma se cubre el objetivo inicialmente propuesto relacionado con el análisis y caracterización de diferentes modelos DL aplicados a reducción de dimensionalidad.

7.1.3. Análisis del desbalanceo en DL

El Capítulo 5 analiza otro de los problemas más estudiados en el ámbito del aprendizaje automático: el desbalanceo en los datos de entrada. El objetivo fundamental es confirmar si, como ocurre con otros clasificadores tradicionales, el desbalanceo afecta negativamente al rendimiento predictivo de las CNN. Para ello, el primer paso es establecer la hipótesis de partida que plantea que, efectivamente, este factor estará asociado con un peor rendimiento de las CNN. A continuación, es necesario diseñar un marco experimental que permita confirmar dicha hipótesis. Para hacerlo se utilizan varios conjuntos de datos con grado de desbalanceo variable.

Los resultados obtenidos en la experimentación anterior confirman que se cumple la hipótesis inicialmente planteada, ya que el rendimiento predictivo de la CNN mejora a medida que disminuye el ratio de desbalanceo asociado al conjunto de entrada. Así, es posible observar como el estudio cubre otro de los objetivos anteriormente enumerados: el análisis del desbalanceo en arquitecturas profundas.

Finalmente, este trabajo permite abrir nuevas vías de trabajo futuro que lleven a cabo un análisis más profundo y exhaustivo sobre los efectos de este factor en otros modelos DL y propongan nuevos métodos que mitiguen los efectos del desbalanceo, ya sea basándose en métodos tradicionales de pre-procesamiento o creando modelos híbridos que combinen técnicas DL con métodos clásicos de reducción del desbalanceo.

7.1.4. *Ensembles* de arquitecturas profundas

El Capítulo 6 persigue el último de los objetivos anteriormente enumerados, el desarrollo de nuevos modelos basados en *ensembles* que incorporen arquitecturas profundas. En concreto, se propone un clasificador, llamado ClEnDAE, basado en *ensembles* que incorpora DAE para reducir la dimensionalidad del espacio de entrada original, es decir, se trata, al igual que AEkNN, de un método orientado a mitigar los efectos negativos de la alta dimensionalidad de los datos.

El algoritmo ClEnDAE se basa en dos pilares fundamentales. Por un lado, está basado en *ensembles* lo que implica que varios componentes trabajan de forma cooperativa para conseguir un mismo objetivo. En este sentido, existen estudios que demuestran que estos modelos tienen un mejor comportamiento que los mismos componentes utilizados de forma aislada. Por otro, el hecho de incorporar DAE permite que cada componente genere su propio espacio de características de menor dimensionalidad, consiguiendo mitigar los efectos asociados al tamaño de los datos de entrada y diversificando el espacio de búsqueda, lo que implica un mejor rendimiento predictivo. Así mismo, es importante indicar que ClEnDAE incluye cuatro núcleos diferentes de clasificación basados en kNN, SVM, MLP y C4.5, que pueden ser seleccionados cuando se utiliza dicho método.

Los resultados asociados a la experimentación desarrollada para evaluar el rendimiento de ClEnDAE permiten obtener ciertas conclusiones:

- ClEnDAE mejora considerablemente a los métodos de clasificación tradicionales kNN, SVM, MLP y C4.5.
- ClEnDAE ofrece mejor rendimiento que uno de sus componentes utilizado de forma aislada que también incorpora DAE para reducir la dimensionalidad.
- ClEnDAE obtiene mejoras significativas con respecto a cuatro algoritmos clásicos de reducción de dimensionalidad: PCA, LDA, ISOMAP y LLE.

De esta forma, se confirma que el método ClEnDAE contribuye a mejorar el rendimiento predictivo y mitiga los efectos de la alta dimensionalidad de los datos de entrada. Tanto el hecho de combinar la información proporcionada por los distintos componentes como incluir DAE para reducir la dimensionalidad repercuten en este comportamiento.

7.2. Publicaciones

Tras resumir en la sección previa las principales aportaciones de la presente tesis doctoral, a continuación se enumeran las publicaciones asociadas que ya se han incluido anteriormente en cada uno de los capítulos correspondientes.

7.2.1. Revistas JCR

- Pulgar F. J., Charte F., Rivera A. J., del Jesus M. J., *AEkNN: An AutoEncoder kNN-Based Classifier With Built-in Dimensionality Reduction*, International Journal of Computational Intelligence Systems, Vol. 12, Issue 1, pp. 436 - 452, Nov. 2018. doi: 10.2991/ijcis.2018.125905686.
- Pulgar F. J., Charte F., Rivera A. J., del Jesus M. J., *Choosing the proper autoencoder for feature fusion based on data complexity and classifiers: Analysis, tips and guidelines.*, Information Fusion, Vol. 54, pp. 44-60, Feb. 2020. doi: 10.1016/j.inffus.2019.07.004.
- Pulgar F. J., Charte F., Rivera A. J., del Jesus M. J., *ClEnDAE: A classifier based on ensembles with built-in dimensionality reduction through denoising autoencoders*, Information Sciences (en revisión).

7.2.2. Congresos

- Pulgar F. J., Rivera A. J., Charte F., del Jesus M. J., *On the Impact of Imbalanced Data in Convolutional Neural Networks Performance*, 12th International Conference on Hybrid Artificial Intelligent Systems, pp. 220–232, 2017.
- Pulgar F. J., Charte F., Rivera A. J., del Jesus M. J., *A First Approach to Face Dimensionality Reduction Through Denoising Autoencoders*, 19th International Conference on Intelligent Data Engineering and Automated Learning, pp. 439-447, 2018. doi: 10.1007/978-3-030-03493-1_46.
- Pulgar F. J., Rivera A. J., Charte F., del Jesus M. J., *Análisis del impacto de datos desbalanceados en el rendimiento predictivo de redes neuronales convolucionales*, XVIII Conferencia de la Asociación Española para la Inteligencia Artificial, pp. 1213–1218, 2018.

7.2.3. Otras publicaciones

En este apartado se enumeran otras publicaciones que guardan menos relación con la tesis doctoral presentada en esta memoria.

7.2.3.1. Revistas

- Charte, F., Rivera A. J., Pulgar F. J., del Jesus M. J., *Explotación de la potencia de procesamiento mediante paralelismo: un recorrido histórico hasta la GPGPU*, Enseñanza y aprendizaje de ingeniería de computadores. Revista de experiencias docentes en ingeniería de computadores, Vol. 6, pp. 19-33, 2016.
- Charte, F., Espinilla M., Rivera A. J., Pulgar F. J., *Uso de dispositivos FPGA como apoyo a la enseñanza de asignaturas de arquitectura de computadores*, Enseñanza y aprendizaje de ingeniería de computadores. Revista de experiencias docentes en ingeniería de computadores, Vol. 7, pp. 17-52, 2017.
- Pulgar F. J., Rivera A. J., Pérez M.D., González P., Carmona C. J., del Jesus M. J., *MEFASD-BD: Multi-Objective Evolutionary Algorithm for Subgroup Discovery in Big Data Environments - A MapReduce Solution*, Knowledge-Based Systems, Vol. 117, pp. 70-78, 2017. doi: 10.1016/j.knosys.2016.08.021.

7.2.3.2. Congresos

- Carmona C.J., Pulgar F.J., García A.M., González P., del Jesus M.J. *Análisis descriptivo mediante aprendizaje supervisado basado en patrones emergentes*, VII Simposio de Teoría y Aplicaciones de Minería de Datos, pp. 685-694, 2015.
- Pulgar, F. J., Carmona C.J., Rivera A. J., González P., del Jesus M.J., *Una primera aproximación al descubrimiento de subgrupos bajo el paradigma MapReduce*, 1er Workshop en Big Data y Análisis de Datos Escalable, pp. 991-1000, 2015.
- Rivera A. J., Pérez M.D., Charte F., Pulgar, F. J., del Jesus M., *An ensemble strategy for forecasting the extra-virgin olive oil price in Spain*, International work-conference on Time Series (ITISE 2015), pp. 506–516, 2015.

- Rivera A. J., Pérez M. D., Pulgar F. J., del Jesus M. J., *GenRBFNSpark: A first implementation in Spark of a genetic algorithm to RBFN design*, XVI Conferencia de la Asociación Española para la Inteligencia Artificial, pp. 1011-1020, 2015.
- Carmona C.J., Pulgar F. J., Rivera A. J., del Jesus M.J., Aguilera J., *Estimating the Maximum Power Delivered by Concentrating Photovoltaics Technology Through Atmospheric Conditions Using a Differential Evolution Approach*, 11th International Conference on Hybrid Artificial Intelligent Systems, pp. 273-282, 2016.
- Pulgar F. J., *Deep learning networks and ensembles for the treatment of high dimensionality and imbalance in supervised learning*, Abstract book of the Doctoral Conference for Young Researchers at the University of Jaén, pp. 58, 2016.
- Pulgar F. J., *On the Impact of Imbalanced Data in Convolutional Neural Networks Performance*, Abstract book of the Doctoral Conference for Young Researchers at the University of Jaén, pp. 47, 2017.
- Rivera, A. J., Charte F., Pulgar F. J., del Jesus M. J., *A Transformation Approach Towards Big Data Multilabel Decision Trees*, 14th International Work-Conference on Artificial Neural Networks (IWANN 2017), pp. 73–84, 2017.
- Pulgar F. J., *A first approach to face dimensionality reduction through denoising autoencoders*, Abstract book of the Doctoral Conference for Young Researchers at the University of Jaén, pp. 51, 2018.

7.3. Trabajos futuros

El desarrollo de los diferentes estudios asociados a la presente tesis doctoral ha conllevado la apertura de nuevas posibilidades de investigación que pueden dar lugar a aportaciones futuras. A continuación se describen las principales líneas de trabajo asociadas a cada contribución:

- **Tratamiento de alta dimensionalidad con técnicas DL:** en este sentido la presente tesis doctoral aporta, como se ha indicado, un nuevo método llamado AEkNN que ha mostrado grandes resultados a la hora de mejorar el rendimiento predictivo con datos de alta dimensionalidad. Este hecho abre nuevas vías de trabajo orientadas a generar nuevos modelos

que combinen otros clasificadores con AE para mejorar el rendimiento predictivo de los mismos en espacios de alta dimensionalidad. Así mismo, existe la posibilidad de cambiar el tipo de AE utilizado, por otros modelos más complejos que generen espacios de características de mayor calidad.

- **Estudio sobre la relación entre *autoencoder*, método de aprendizaje y problema:** para lograr este objetivo se ha diseñado una exhaustiva experimentación que incluye un conjunto amplio de datasets con características muy diversas, cuatro tipos diferentes de AE y cuatro clasificadores para evaluar los nuevos espacios de características generados por los distintos modelos. Este estudio ha mostrado el gran rendimiento de los AE al afrontar esta tarea y las mejoras con respecto a métodos tradicionales de reducción de dimensionalidad. En este sentido existen varias vías de trabajo futuro. Por un lado, es posible ampliar la experimentación con nuevos modelos de AE, ya que continuamente surgen nuevas variantes que podrían mejorar los resultados obtenidos. Por otro, dado que este estudio ha demostrado cómo los AE superan a métodos tradicionales de reducción de dimensionalidad, surge la posibilidad de aplicar esta nueva alternativa a la solución de problemas reales, mejorando así los resultados que podrían proporcionar dichos métodos clásicos.
- **Análisis del desbalanceo en DL:** la experimentación realizada en este sentido ha mostrado que el desbalanceo afecta negativamente al rendimiento predictivo obtenido mediante redes neuronales convolucionales. Sin embargo, este es un estudio preliminar, dado que sólo analiza un arquitectura concreta y un dataset específico. Por ello, una línea de investigación futura es realizar una experimentación mucho más exhaustiva donde se incluyan un conjunto mucho más amplio de dataset, diferentes arquitecturas de CNN e incluir otros modelos DL. Así mismo, se abre la posibilidad de aplicar métodos tradicionales de reducción del desbalanceo como pre-procesamiento para mejorar el rendimiento de las CNN o incluso diseñar nuevos modelos de clasificación que combinen dichos algoritmos clásicos con técnicas DL.
- ***Ensembles* de arquitecturas profundas:** dentro de este área de investigación la presente tesis aporta un nuevo método llamado ClEnDAE, se trata de un clasificador basado en *ensembles* que incorpora DAE para reducir la dimensionalidad del espacio de entrada. La experimentación

desarrollada ha mostrado que la combinación de *ensembles* y DAE contribuye a mejorar el rendimiento predictivo tanto frente a los clasificadores tradicionales como en comparación con otras técnicas de reducción de dimensionalidad. Tras esta propuesta, se abren nuevas vías de trabajo como podrían ser la aplicación del modelo propuesto a la resolución de problemas reales o el desarrollo de nuevos modelos. Por un lado, dado su gran rendimiento, el método ClEnDAE es una alternativa a tener en cuenta para aplicarlo a determinadas situaciones reales, sobre todo cuando los datos analizados estén caracterizados por una alta dimensionalidad. Por otro, existe la posibilidad de desarrollar nuevos modelos basados en *ensembles* que incorporen otros tipos de AE, como *robust* o *contractive*, o crear *ensembles* de otros tipos de arquitecturas profundas.

Bibliografía

Aha, D. W., Kibler, D. y Albert, M. K. (1991). "Instance-Based Learning Algorithms". En: *Machine Learning* 6.1, págs. 37-66. ISSN: 08856125. DOI: 10.1023/A:1022689900470.

Altman, N. S. (1992). "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression". En: *The American Statistician* 46.3, págs. 175-185. ISSN: 0003-1305. DOI: 10.1080/00031305.1992.10475879.

Andrew Hall, M. (2000). "Correlation-Based Feature Selection for Machine Learning". En: *Department of Computer Science* 19.

Artoni, F., Delorme, A. y Makeig, S. (2018). "Applying dimension reduction to EEG data by Principal Component Analysis reduces the quality of its subsequent Independent Component decomposition". En: *NeuroImage* 175, págs. 176 -187. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2018.03.016.

Atkeson, C. G. et al. (1997). "Locally Weighted Learning". En: *Artificial Intelligence* 11, págs. 11-73. ISSN: 02692821. DOI: 10.1023/A:1006559212014.

Auli, M. et al. (2013). "Joint Language and Translation Modeling with Recurrent Neural Networks". En: *Proc. of EMNLP*.

Aymaz, S. y Köse, C. (2019). "A novel image decomposition-based hybrid technique with super-resolution method for multi-focus image fusion". En: *Information Fusion* 45, págs. 113 -127. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2018.01.015.

Badrinarayanan, V., Kendall, A. y Cipolla, R. (2017). "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12, págs. 2481-2495. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2644615.

Bahdanau, D., Cho, K. y Bengio, Y. (2014). "Neural machine translation by jointly learning to align and translate". En: *arXiv preprint arXiv:1409.0473*.

Bako, S. et al. (2017). "Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings". En: *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2017)* 36.4.

Barandela, R. et al. (2003). "Strategies for Learning in Class Imbalance Problems". En: *Pattern Recognition* 36, págs. 849-851. DOI: 10.1016/S0031-3203(02)00257-1.

Barlow, H. (1999). "Unsupervised Learning". En: vol. 1, págs. 1-17. DOI: 10.1162/neco.1989.1.3.295.

Batini, C., Lenzerini, M. y Navathe, S. B. (1986). "A Comparative Analysis of Methodologies for Database Schema Integration". En: *ACM Comput. Surv.* 18.4, págs. 323-364. ISSN: 0360-0300. DOI: 10.1145/27633.27634.

Batista, G. E., Prati, R. C. y Monard, M. C. (2004). "A study of the behavior of several methods for balancing machine learning training data". En: *ACM SIGKDD explorations newsletter* 6.1, págs. 20-29.

Bauer, E. y Kohavi, R. (1999). "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants". En: *Machine Learning* 36.1, págs. 105-139. ISSN: 1573-0565. DOI: 10.1023/A:1007515423169.

Bellman, R (1957). *Dynamic Programming*. Princeton University Press, pág. 342. ISBN: 978-0-691-07951-6.

Bellman, R. (1961). *Adaptive control processes: A guided tour*. Princeton University Press, pág. 276. ISBN: 9781400874668.

Bengio, Y, Courville, A y Vincent, P (2013). "Representation Learning: A Review and New Perspectives". En: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35.8, págs. 1798-1828. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2013.50.

Bengio, Y. (2009). "Learning Deep Architectures for AI". En: *Foundations and Trends in Machine Learning* 2.1, págs. 1-127. ISSN: 1935-8237. DOI: 10.1561/2200000006.

Bengio, Y. (2013). "Deep Learning of Representations: Looking Forward". En: *International Conference on Statistical Language and Speech Processing*, págs. 1-37. ISBN: 9783642395925. DOI: 10.1007/978-3-642-39593-2_1.

Berkhin, P. (2006). "A Survey of Clustering Data Mining Techniques". En: *Grouping Multidimensional Data: Recent Advances in Clustering*. Ed. por J. Kogan, C. Nicholas y M. Teboulle. Berlin, Heidelberg: Springer Berlin Heidelberg, págs. 25-71. ISBN: 978-3-540-28349-2. DOI: 10.1007/3-540-28349-8_2.

Berson, A. y Smith, S. J. (1997). *Data Warehousing, Data Mining, and Olap*. 1st. New York, NY, USA: McGraw-Hill, Inc. ISBN: 0070062722.

Berzal, F. (2018). *Redes Neuronales and Deep Learning*. Independently Published. ISBN: 9781731265388.

Bojarski, M. et al. (2016b). "End to end learning for self-driving cars". En: *arXiv preprint arXiv:1604.07316*.

Bojarski, M. et al. (2016a). "End to End Learning for Self-Driving Cars". En: *CoRR* abs/1604.07316. arXiv: 1604.07316.

Bonab, H. y Can, F. (2019). "Less Is More: A Comprehensive Framework for the Number of Components of Ensemble Classifiers". En: *IEEE Transactions on Neural Networks and Learning Systems* 30.9, págs. 2735-2745. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2018.2886341.

Boser, B. E., Guyon, I. M. y Vapnik, V. N. (1992). "A training algorithm for optimal margin classifiers". En: *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, págs. 144-152.

Bourlard, H. y Kamp, Y. (1988). "Auto-association by multilayer perceptrons and singular value decomposition". En: *Biological cybernetics* 59.4-5, págs. 291-294. DOI: 10.1007/BF00332918.

Breiman, L. (1984). *Classification and regression trees*. Routledge. DOI: 10.1201/9781315139470.

Breiman, L. (1996). "Bagging Predictors". En: *Machine Learning* 24.2, págs. 123 -140. ISSN: 1573-0565. DOI: 10.1023/A:1018054314350.

Breiman, L. (2001). "Random Forests". En: *Machine Learning* 45.1, págs. 5-32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324.

Bäck, T., Fogel, D. y Michalewicz, Z. (2000). *Evolutionary Computation 1—Basic Algorithms and Operators*. DOI: 10.1887/0750306653.

Cauchy, A. (1847). "Méthode générale pour la résolution systemes d'équations simultanées". En: *Comp. Rend. Sci. Paris* 25.1847, págs. 536-538.

Cayton, L. (2005). "Algorithms for manifold learning". En: *Univ. of California at San Diego Tech. Rep* 12.1-17, pág. 1.

Charte, D. et al. (2018). "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines". En: *Information Fusion* 44, págs. 78-96. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2017.12.007.

Chaturvedi, I. et al. (2016). "Learning word dependencies in text by means of a deep recurrent belief network". En: *Knowledge-Based Systems* 108, págs. 144-154. DOI: 10.1016/j.knosys.2016.07.019.

Chawla, N. et al. (2002). "SMOTE: Synthetic Minority Over-sampling Technique". En: *J. Artif. Intell. Res. (JAIR)* 16, págs. 321-357. DOI: 10.1613/jair.953.

Chawla, N. V., Japkowicz, N. y Kotcz, A. (2004). "Special Issue on Learning from Imbalanced Data Sets". En: *SIGKDD Explorations Newsletter* 6.1, págs. 1-6. ISSN: 1931-0145. DOI: 10.1145/1007730.1007733.

Chen, G. et al. (2018). "Musicality-Novelty Generative Adversarial Nets for Algorithmic Composition". En: *Proceedings of the 26th ACM International Conference on Multimedia*. MM '18. Seoul, Republic of Korea: ACM, págs. 1607-1615. ISBN: 978-1-4503-5665-7. DOI: 10.1145/3240508.3240604.

Chen, J. X. (2016). "The Evolution of Computing: AlphaGo". En: *Computing in Science & Engineering* 18.4, págs. 4-7. DOI: 10.1109/MCSE.2016.74.

Chen, X. et al. (2011). "Graph-Based Feature Selection for Object-Oriented Classification in VHR Airborne Imagery". En: *IEEE Transactions on Geoscience and Remote Sensing* 49.1, págs. 353-365. ISSN: 0196-2892. DOI: 10.1109/TGRS.2010.2054832.

Chen, Z. y Li, W. (2017). "Multisensor Feature Fusion for Bearing Fault Diagnosis Using Sparse Autoencoder and Deep Belief Network". En: *IEEE Transactions on Instrumentation and Measurement* 66.7, págs. 1693-1702. DOI: 10.1109/tim.2017.2669947.

Cherkassky, V. y Ma, Y. (2004). "Practical selection of SVM parameters and noise estimation for SVM regression". En: *Neural Networks* 17.1, págs. 113 -126. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(03)00169-2.

Chizi, B. y Maimon, O. (2005). "Dimension Reduction and Feature Selection". En: *Data Mining and Knowledge Discovery Handbook*. Ed. por O. Maimon y L. Rokach. Boston, MA: Springer US, págs. 93-111. ISBN: 978-0-387-25465-4. DOI: 10.1007/0-387-25465-X_5.

Cho, K. et al. (2014). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". En: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, págs. 1724-1734. DOI: 10.3115/v1/D14-1179.

Choi, S., Kim, E. y Oh, S. (2013). "Human behavior prediction for smart homes using deep learning". En: *2013 IEEE RO-MAN*, págs. 173-179. DOI: 10.1109/ROMAN.2013.6628440.

Chung, J. et al. (2015). "Gated Feedback Recurrent Neural Networks". En: *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, págs. 2067-2075.

Ciresan, D., Meier, U. y Schmidhuber, J. (2012a). "Multi-column deep neural networks for image classification". En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, págs. 3642-3649. ISBN: 978-1-4673-1228-8. DOI: 10.1109/CVPR.2012.6248110.

Ciresan, D. et al. (2012b). "Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images". En: *Advances in neural information processing systems*, págs. 2843-2851. ISBN: 9781627480031.

Coates, A., Ng, A. y Lee, H. (2011a). "An Analysis of Single-Layer Networks in Unsupervised Feature Learning". En: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. por G. Gordon, D. Dunson y M. Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, págs. 215-223.

Coates, A., Ng, A. y Lee, H. (2011b). "An Analysis of Single-Layer Networks in Unsupervised Feature Learning". En: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. por G. Gordon, D. Dunson y M. Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, págs. 215-223.

Cover, T. y Hart, P. (1967). "Nearest neighbor pattern classification". En: *IEEE Transactions on Information Theory* 13.1, págs. 21-27. ISSN: 0018-9448. DOI: 10.1109/TIT.1967.1053964.

Cowdrey, K. W. G. y Malekian, R. (2018). "Home automation - an IoT based system to open security gates using number plate recognition and artificial neural networks". En: *Multimedia Tools and Applications* 77.16, págs. 20325-20354. ISSN: 1573-7721. DOI: 10.1007/s11042-017-5407-1.

Dash, M. y Liu, H. (1997). "Feature selection for classification". En: *Intelligent data analysis* 1.3, págs. 131-156. DOI: 10.3233/IDA-1997-1302.

David, O. E., Netanyahu, N. S. y Wolf, L. (2016). "DeepChess: End-to-End Deep Neural Network for Automatic Learning in Chess". En: *Artificial Neural Networks and Machine Learning – ICANN 2016*. Ed. por A. E. Villa, P. Masulli y A. J. Pons Rivero. Cham: Springer International Publishing, págs. 88-96. ISBN: 978-3-319-44781-0. DOI: 10.1007/978-3-319-44781-0_11.

Davis, L. (1991). *Handbook of genetic algorithms*. CUMINCAD.

De Gooijer, J. e Hyndman, R. (2006). "25 years of time series forecasting". En: *International Journal of Forecasting* 22.3, págs. 443 -473. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2006.01.001.

Deng, J. et al. (2014). "Sparse Autoencoder-Based Feature Transfer Learning for Speech Emotion Recognition". En: *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction(ACII)*. Vol. 00, págs. 511-516. DOI: 10.1109/ACII.2013.90.

Deng, L. (2014). "Deep Learning: Methods and Applications". En: *Foundations and Trends in Signal Processing* 7.3-4, págs. 197-387. ISSN: 1932-8346. DOI: 10.1561/2000000039.

Deng, L. y Yu, D. (2011). "Deep convex net: A scalable architecture for speech pattern classification". En: *Proceedings of the Annual Conference of the International Speech Communication Association*, págs. 2285-2288. ISBN: 19909772 (ISSN).

Diamantini, C. y Potena, D. (2009). "Bayes Vector Quantizer for Class-Imbalance Problem". En: *IEEE Transactions on Knowledge and Data Engineering* 21.5, págs. 638-651. DOI: 10.1109/TKDE.2008.187.

Domingos, P. (2002). "MetaCost: A General Method for Making Classifiers Cost-Sensitive". En: *Proceedings of the Fifth ACM SIGKDD Int'l. Conf. on Knowledge Discovery & Data Mining*. DOI: 10.1145/312129.312220.

Domingos, P. (2012). "A Few Useful Things to Know About Machine Learning". En: *Commun. ACM* 55.10, págs. 78-87. ISSN: 0001-0782. DOI: 10.1145/2347736.2347755.

Donahue, C., Li, B. y Prabhavalkar, R. (2018). "Exploring Speech Enhancement with Generative Adversarial Networks for Robust Speech Recognition". En: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, págs. 5024-5028. DOI: 10.1109/ICASSP.2018.8462581.

Draper, N. R. y Smith, H. (1998). *Applied regression analysis*. Vol. 326. John Wiley & Sons.

Dreiseitl, S. y Ohno-Machado, L. (2002). "Logistic regression and artificial neural network classification models: a methodology review". En: *Journal of Biomedical Informatics* 35.5, págs. 352 -359. ISSN: 1532-0464. DOI: 10.1016/S1532-0464(03)00034-0.

Du, Y. et al. (2019). "Deep learning with multi-scale feature fusion in remote sensing for automatic oceanic eddy detection". En: *Information Fusion* 49, págs. 89 -99. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2018.09.006.

Duan, K.-B. y Keerthi, S. S. (2005). "Which is the Best Multiclass SVM Method? An Empirical Study". En: *Proceedings of the 6th International Conference on Multiple Classifier Systems*. MCS'05. Seaside, CA: Springer-Verlag, págs. 278-285. DOI: 10.1007/11494683_28.

Duchi, J., Hazan, E. y Singer, Y. (2011). "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". En: *J. Mach. Learn. Res.* 12, págs. 2121-2159. ISSN: 1532-4435.

Efron, B. y Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC press.

Elkan, C. (2001). "The Foundations of Cost-sensitive Learning". En: *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI'01. Seattle, WA, USA: Morgan Kaufmann Publishers Inc., págs. 973-978. ISBN: 1-55860-812-5, 978-1-558-60812-2.

Faloutsos, C., Ranganathan, M. y Manolopoulos, Y. (1994). "Fast Subsequence Matching in Time-series Databases". En: *SIGMOD Rec.* 23.2, págs. 419-429. ISSN: 0163-5808. DOI: 10.1145/191843.191925.

Fan, Y. J. (2019). "Autoencoder node saliency: Selecting relevant latent representations". En: *Pattern Recognition* 88, págs. 643 -653. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2018.12.015.

Fayyad, U., Piatetsky-Shapiro, G. y Smyth, P. (1996a). "From Data Mining to Knowledge Discovery in Databases". En: *AI Magazine* 17, págs. 37-54. DOI: 10.1609/aimag.v17i3.1230.

Fayyad, U., Piatetsky-Shapiro, G. y Smyth, P. (1996b). "Knowledge Discovery and Data Mining: Towards a Unifying Framework". En: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, págs. 82-88.

Feng, F., Wang, X. y Li, R. (2014). "Cross-modal Retrieval with Correspondence Autoencoder". En: *Proceedings of the 22nd ACM International Conference on Multimedia*. MM '14. Orlando, Florida, USA: ACM, págs. 7-16. ISBN: 978-1-4503-3063-3. DOI: 10.1145/2647868.2654902.

Fernández, A. et al. (2013). "Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches". En: *Knowledge-Based Systems* 42, págs. 97 -110. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2013.01.018.

Freund, Y. y Schapire, R. E. (1996). "Experiments with a New Boosting Algorithm". En: *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*. ICML'96. Bari, Italy: Morgan Kaufmann Publishers Inc., págs. 148-156. ISBN: 1-55860-419-7.

Freund, Y. y Schapire, R. E. (1997). "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". En: *Journal of Computer and System Sciences* 55.1, págs. 119 -139. ISSN: 0022-0000. DOI: 10.1006/jcss.1997.1504.

Gabralla, L. A., Mahersia, H. y Abraham, A. (2015). "Ensemble Neurocomputing Based Oil Price Prediction". En: *Afro-European Conference for Industrial Advancement*. Ed. por A. Abraham, P. Krömer y V. Snasel. Cham: Springer International Publishing, págs. 293-302. ISBN: 978-3-319-13572-4.

Galar, M. et al. (2012). "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches". En: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.4, págs. 463-484.

García, S., Luengo, J. y Herrera, F. (2014). *Data Preprocessing in Data Mining*. Springer Publishing Company, Incorporated. DOI: 10.1007/978-3-319-10247-4.

García, S., Luengo, J. y Herrera, F. (2016). "Tutorial on practical tips of the most influential data preprocessing algorithms in data mining". En: *Knowledge-Based Systems* 98, págs. 1 -29. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2015.12.006.

García, V., Sánchez, J. y Mollineda, R. (2012). "On the effectiveness of preprocessing methods when dealing with different levels of class imbalance". En: *Knowledge-Based Systems* 25.1. Special Issue on New Trends in Data Mining, págs. 13 -21. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2011.06.013.

Gardner, M. W. y Dorling, S. (1998). "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences". En: *Atmospheric environment* 32.14-15, págs. 2627-2636.

Gatys, L., Ecker, A. y Bethge, M. (2015). "A Neural Algorithm of Artistic Style". En: *arXiv*. DOI: 10.1167/16.12.326.

Ghosh, A. y Jain, L. (2005). *Evolutionary Computation in Data Mining*. ISBN: 978-3-540-22370-2. DOI: 10.1007/3-540-32358-9.

Glorot, X., Bordes, A. y Bengio, Y. (2011). "Domain Adaptation for Large-scale Sentiment Classification: A Deep Learning Approach". En: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML'11. Bellevue, Washington, USA: Omnipress, págs. 513-520. ISBN: 978-1-4503-0619-5.

Good, O. (2015). "How google translate squeezes deep learning onto a phone". En: *Google Research Blog* 504.

Goodfellow, I. et al. (2014). "Generative Adversarial Nets". En: *Advances in Neural Information Processing Systems 27*. Ed. por Z. Ghahramani et al. Curran Associates, Inc., págs. 2672-2680.

Goodfellow, I., Bengio, Y y Courville, A (2016). *Deep Learning*. The MIT Press, pág. 800. ISBN: 978-0262035613.

Graves, A., Mohamed, A. e Hinton, G. (2013). "Speech recognition with deep recurrent neural networks". En: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, págs. 6645-6649. DOI: 10.1109/ICASSP.2013.6638947.

Gülçehre, Ç. y Bengio, Y. (2016). "Knowledge Matters: Importance of Prior Information for Optimization". En: *Journal of Machine Learning Research* 17.8, págs. 1-32.

Guyon, I. y Elisseeff, A. (2003). "An Introduction to Variable and Feature Selection". En: *J. Mach. Learn. Res.* 3, págs. 1157-1182. ISSN: 1532-4435.

Guyon, I. y Elisseeff, A. (2006). "An Introduction to Feature Extraction". En: *Feature Extraction: Foundations and Applications*. Ed. por I. Guyon et al. Berlin, Heidelberg: Springer Berlin Heidelberg, págs. 1-25. ISBN: 978-3-540-35488-8. DOI: 10.1007/978-3-540-35488-8_1.

Gómez, J. A. et al. (2018). "End-to-end neural network architecture for fraud scoring in card payments". En: *Pattern Recognition Letters* 105. Machine Learning and Applications in Artificial Intelligence, págs. 175 -181. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2017.08.024.

HaddadPajouh, H. et al. (2018). "A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting". En: *Future Generation Computer Systems* 85, págs. 88 -96. ISSN: 0167-739X. DOI: 10.1016/j.future.2018.03.007.

Han, J., Pei, J. y Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier. DOI: 10.1016/C2009-0-61819-5.

Hand, D. J., Smyth, P. y Mannila, H. (2001). *Principles of Data Mining*. Cambridge, MA, USA: MIT Press. ISBN: 0-262-08290-X, 9780262082907.

Hansen, L. y Salamon, P. (1990). "Neural Network Ensembles". En: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 12, págs. 993 -1001. DOI: 10.1109/34.58871.

Hayashi, C. (1998). "What is Data Science ? Fundamental Concepts and a Heuristic Example". En: *Data Science, Classification, and Related Methods*. Ed. por C. Hayashi et al. Tokyo: Springer Japan, págs. 40-51. ISBN: 978-4-431-65950-1.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR. ISBN: 0023527617.

He, H. y Garcia, E. A. (2009). "Learning from Imbalanced Data". En: *IEEE Transactions on Knowledge and Data Engineering* 21.9, págs. 1263-1284. ISSN: 1041-4347. DOI: 10.1109/TKDE.2008.239.

He, K. et al. (2016). "Deep Residual Learning for Image Recognition". En: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, págs. 770-778. DOI: 10.1109/CVPR.2016.90.

He, K. et al. (2016). "Deep Residual Learning for Image Recognition". En: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

He, Y. et al. (2017). "Software-Defined Networks with Mobile Edge Computing and Caching for Smart Cities: A Big Data Deep Reinforcement Learning Approach". En: *IEEE Communications Magazine* 55.12, págs. 31-37. ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1700246.

Hearst, M. A. et al. (1998). "Support vector machines". En: *IEEE Intelligent Systems and their applications* 13.4, págs. 18-28.

Hecht-Nielsen, R. (1995). "Replicator neural networks for universal optimal source coding". En: *Science* 269.5232, págs. 1860-1863. DOI: 10.1126/science.269.5232.1860.

Heckerman, D., Geiger, D. y Chickering, D. M. (1995). "Learning Bayesian networks: The combination of knowledge and statistical data". En: *Machine Learning* 20.3, págs. 197-243. ISSN: 1573-0565. DOI: 10.1007/BF00994016.

Hernández Orallo, J., Ferri Ramírez, C. y Ramírez, M. J. (2004). *Introducción a la Minería de Datos*. Pearson Prentice Hall.

Herrera, F. et al. (2016). "Multilabel Classification". En: *Multilabel Classification : Problem Analysis, Metrics and Techniques*. Cham: Springer International Publishing, págs. 17-31. ISBN: 978-3-319-41111-8. DOI: 10.1007/978-3-319-41111-8_2.

Hinneburg, A., Aggarwal, C. C. y Keim, D. A. (2000). "What Is the Nearest Neighbor in High Dimensional Spaces?" En: *Proceedings of the International Conference on Very Large Databases*, págs. 506-515.

Hinton, G. (2010). "A practical guide to training restricted Boltzmann machines". En: *Momentum* 9.1, pág. 926. DOI: 10.1007/978-3-642-35289-8_32.

Hinton, G. et al. (2012). "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". En: *IEEE Signal Processing Magazine* 29.6, págs. 82-97.

Hinton, G. E. y Salakhutdinov, R. R. (2006). "Reducing the Dimensionality of Data with Neural Networks". En: *Science* 313.5786, págs. 504-507. ISSN: 1095-9203. DOI: 10.1126/science.1127647.

Hinton, G. E. y Sejnowski, T. (1983). "Optimal perceptual inference". En: págs. 448-453.

Hinton, G. E., Osindero, S. y Teh, Y.-W. (2006). "A Fast Learning Algorithm for Deep Belief Nets". En: *Neural Comput.* 18.7, págs. 1527-1554. ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.7.1527.

Hochreiter, S. y Schmidhuber, J. (1997). "Long Short-Term Memory". En: *Neural Computation* 9.8, págs. 1735-1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

Höppner, F. (2005). "Association Rules". En: *Data Mining and Knowledge Discovery Handbook*. Ed. por O. Maimon y L. Rokach. Boston, MA: Springer US, págs. 353-376. ISBN: 978-0-387-25465-4. DOI: 10.1007/0-387-25465-X_16.

Hornik, K., Stinchcombe, M. y White, H. (1989). "Multilayer feedforward networks are universal approximators". En: *Neural networks* 2.5, págs. 359-366.

Hou, B. et al. (2019). "LSTM Based Auto-Encoder Model for ECG Arrhythmias Classification". En: *IEEE Transactions on Instrumentation and Measurement*, págs. 1-1. ISSN: 0018-9456. DOI: 10.1109/TIM.2019.2910342.

Hu, B. et al. (2014). "Convolutional Neural Network Architectures for Matching Natural Language Sentences". En: *Advances in Neural Information Processing Systems 27*. Ed. por Z. Ghahramani et al. Curran Associates, Inc., págs. 2042-2050.

Huang, C. et al. (2019). "Deep Imbalanced Learning for Face Recognition and Attribute Prediction". En: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, págs. 1-1. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2019.2914680.

Huang, G. et al. (2012). "Extreme Learning Machine for Regression and Multiclass Classification". En: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42.2, págs. 513-529. ISSN: 1083-4419. DOI: 10.1109/TSMCB.2011.2168604.

Huang, G.-B., Zhu, Q.-Y. y Siew, C.-K. (2006). "Extreme learning machine: Theory and applications". En: *Neurocomputing* 70.1. Neural Networks, págs. 489 -501. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2005.12.126.

Huang, G.-B., Ding, X. y Zhou, H. (2010). "Optimization method based extreme learning machine for classification". En: *Neurocomputing* 74.1. Artificial Brains, págs. 155 -163. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2010.02.019.

Huang, G.-B., Wang, D. H. y Lan, Y. (2011). "Extreme learning machines: a survey". En: *International Journal of Machine Learning and Cybernetics* 2.2, págs. 107-122. ISSN: 1868-808X. DOI: 10.1007/s13042-011-0019-y.

Hughes, G. F. (1968). "On the Mean Accuracy of Statistical Pattern Recognizers". En: *IEEE Transactions on Information Theory* 14.1, págs. 55-63. ISSN: 15579654. DOI: 10.1109/TIT.1968.1054102.

Karpathy, A. et al. (2014). "Large-scale Video Classification with Convolutional Neural Networks". En: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Khan, A., Sung, J. E. y Kang, J.-W. (2019). "Multi-channel fusion convolutional neural network to classify syntactic anomaly from language-related ERP components". En: *Information Fusion* 52, págs. 53 -61. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2018.10.008.

Khashei, M. y Bijari, M. (2010). "An artificial neural network (p,d,q) model for timeseries forecasting". En: *Expert Systems with Applications* 37.1, págs. 479 -489. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2009.05.044.

Khatami, A. et al. (2018). "Parallel deep solutions for image retrieval from imbalanced medical imaging archives". En: *Applied Soft Computing* 63, págs. 197 -205. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2017.11.024.

Kibler, D., Aha, D. W. y Albert, M. K. (1989). "Instance-based prediction of real-valued attributes". En: *Computational Intelligence* 5.2, págs. 51-57. DOI: 10.1111/j.1467-8640.1989.tb00315.x.

Kingma, D. P. y Welling, M. (2013). "Auto-encoding variational bayes". En: *arXiv preprint arXiv:1312.6114*.

Kittler, J. (1978). "Feature set search algorithms". En: *Pattern recognition and signal processing*.

Klambauer, G. et al. (2017). "Self-normalizing neural networks". En: *Advances in neural information processing systems*, págs. 971-980.

Ko, A. H., Sabourin, R. y Alceu Souza Britto, J. (2008). "From dynamic classifier selection to dynamic ensemble selection". En: *Pattern Recognition* 41.5, págs. 1718 -1731. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2007.10.015.

Kononenko, I. (2001). "Machine learning for medical diagnosis: history, state of the art and perspective". En: *Artificial Intelligence in Medicine* 23.1, págs. 89 -109. ISSN: 0933-3657. DOI: 10.1016/S0933-3657(01)00077-X.

Kotsiantis, S. B. (2007). "Supervised machine learning: A review of classification techniques". En: *Informatica* 31, págs. 249-268.

Kramer, M. A. (1991). "Nonlinear principal component analysis using autoassociative neural networks". En: *AIChE journal* 37.2, págs. 233-243. DOI: 10.1002/aic.690370209.

Krawczyk, B., Woźniak, M. y Schaefer, G. (2014). "Cost-sensitive decision tree ensembles for effective imbalanced classification". En: *Applied Soft Computing* 14, págs. 554 -562. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2013.08.014.

Krizhevsky, A., Sutskever, I. y Geoffrey E., H. (2012). "ImageNet Classification with Deep Convolutional Neural Networks". En: *Advances in neural information processing systems*, págs. 1097-1105.

Kuchaiev, O. y Ginsburg, B. (2018). "Training deep autoencoders for recommender systems". En: *International Conference on Learning Representations*.

Larochelle, H. y Bengio, Y. (2008). "Classification Using Discriminative Restricted Boltzmann Machines". En: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: ACM, págs. 536-543. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390224.

LeCun, Y y Bengio, Y (1995). "Convolutional networks for images, speech, and time series". En: *The handbook of brain theory and neural networks* 3361, págs. 255-258. ISSN: 1098-7576. DOI: 10.1109/IJCNN.2004.1381049.

LeCun, Y. et al. (1989). "Backpropagation Applied to Handwritten Zip Code Recognition". En: *Neural Computation* 1.4, págs. 541-551. ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.4.541.

LeCun, Y., Kavukcuoglu, K. y Farabet, C. (2010). "Convolutional networks and applications in vision". En: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, págs. 253-256. DOI: 10.1109/ISCAS.2010.5537907.

LeCun, Y., Bengio, Y. e Hinton, G. (2015). "Deep learning". En: *Nature* 521.7553, págs. 436-444. ISSN: 0028-0836. DOI: 10.1038/nature14539.

LeCun, Y. A. et al. (2012). "Efficient BackProp". En: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. por G. Montavon, G. B. Orr y K.-R. Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, págs. 9-48. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_3.

Lee, H. et al. (2009a). "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations". En: *Proceedings of the 26th Annual International Conference on Machine Learning*. Vol. 2008. ACM Press, págs. 609-616. ISBN: 9781605585161. DOI: 10.1145/1553374.1553453.

Lee, H. et al. (2009b). "Unsupervised feature learning for audio classification using convolutional deep belief networks". En: *Advances in Neural Information Processing Systems 22*. Ed. por Y. Bengio et al. Curran Associates, Inc., págs. 1096-1104.

Lee, J. A. y Verleysen, M. (2007). *Nonlinear Dimensionality Reduction*. Springer Science & Business Media, págs. 300-15. ISBN: 9780387393506.

Li, D., Wang, Z. y Xue, Y. (2018). "Fine-grained Android Malware Detection based on Deep Learning". En: *2018 IEEE Conference on Communications and Network Security (CNS)*, págs. 1-2. DOI: 10.1109/CNS.2018.8433204.

Lin, G. et al. (2014). "Feature structure fusion and its application". En: *Information Fusion* 20, págs. 146 -154. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2014.01.002.

Litjens, G. et al. (2017). "A survey on deep learning in medical image analysis". En: *Medical Image Analysis* 42, págs. 60 -88. ISSN: 1361-8415. DOI: 10.1016/j.media.2017.07.005.

Liu, W., Pokharel, P. P. y Principe, J. C. (2006). "Correntropy: A Localized Similarity Measure". En: *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, págs. 4919-4924. DOI: 10.1109/IJCNN.2006.247192.

López, V. et al. (2013). "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics". En: *Information Sciences* 250, págs. 113 -141. ISSN: 0020-0255. DOI: 10.1016/j.ins.2013.07.007.

Maimon, O. y Rokach, L. (2010). *Data Mining and Knowledge Discovery Handbook*. Boston, MA: Springer. ISBN: 9780387244358. DOI: 10.1007/978-0-387-09823-4.

Makhzani, A. et al. (2016). "Adversarial Autoencoders". En: *International Conference on Learning Representations*.

Mangai, U. G. et al. (2010). "A survey of decision fusion and feature fusion strategies for pattern classification". En: *IETE Technical review* 27.4, págs. 293-307. DOI: 10.4103/0256-4602.64604.

Manic, M. et al. (2016). "Intelligent Buildings of the Future: Cyberaware, Deep Learning Powered, and Human Interacting". En: *IEEE Industrial Electronics Magazine* 10.4, págs. 32-49. ISSN: 1932-4529. DOI: 10.1109/MIE.2016.2615575.

Marill, T. y Green, D. M. (1960). "Statistical Recognition Functions and the Design of Pattern Recognizers". En: *IRE Transactions on Electronic Computers* EC-9.4, págs. 472-477. ISSN: 0367-9950. DOI: 10.1109/TEC.1960.5219888.

Marr, B. y Von Scheel, H. (2015). *Big Data: Using SMART Big Data, Analytics and Metrics To Make Better Decisions and Improve Performance*. ISBN: ISBN: B00S4TBEJK.

Martínez, F. et al. (2014). "An ensemble method for time series forecasting with simple exponential smoothing". En: *International Conference Computational and Mathematical Methods in Science and Engineering (CMMSE 2014)*, págs. 871-876.

Masci, J. et al. (2011). "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction". En: *Artificial Neural Networks and Machine Learning (ICANN)*. Ed. por T. Honkela et al. Berlin, Heidelberg: Springer Berlin Heidelberg, págs. 52-59. ISBN: 978-3-642-21735-7.

Maurya, S. et al. (2018). "Fusion of Low-level Features with Stacked Autoencoder for Condition based Monitoring of Machines". En: *IEEE International Conference on Prognostics and Health Management (ICPHM)*, págs. 1-8. DOI: 10.1109/ICPHM.2018.8448969.

Mazurowski, M. A. et al. (2008). "Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance". En: *Neural Networks* 21.2. Advances in Neural Networks Research: IJCNN '07, págs. 427 -436. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2007.12.031.

Min, R. et al. (2009). "A Deep Non-linear Feature Mapping for Large-Margin kNN Classification". En: *Proceedings of the International Conference on Data Mining*. IEEE, págs. 357-366. ISBN: 978-1-4244-5242-2. DOI: 10.1109/ICDM.2009.27.

Minelli, M., Chambers, M. y Dhiraj, A. (2013). *Big Data, Big Analytics: Emerging Business Intelligence and Analytic Trends for Today's Businesses (Wiley CIO)*. 1st. Wiley Publishing. ISBN: 111814760X, 9781118147603.

Mitra, P., Murthy, C. A. y Pal, S. K. (2002). "Unsupervised feature selection using feature similarity". En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.3, págs. 301-312. ISSN: 0162-8828. DOI: 10.1109/34.990133.

Mnih, V. et al. (2015). "Human-level control through deep reinforcement learning". En: *Nature* 518, pág. 529. DOI: 10.1038/nature14236.

Mohamed, A., Dahl, G. E. e Hinton, G. (2012). "Acoustic Modeling Using Deep Belief Networks". En: *IEEE Transactions on Audio, Speech, and Language Processing* 20.1, págs. 14-22. ISSN: 1558-7916. DOI: 10.1109/TASL.2011.2109382.

Mohammadi, M. et al. (2018). "Semisupervised Deep Reinforcement Learning in Support of IoT and Smart City Services". En: *IEEE Internet of Things Journal* 5.2, págs. 624-635. ISSN: 2327-4662. DOI: 10.1109/JIOT.2017.2712560.

Moravčík, M. et al. (2017). "DeepStack: Expert-Level Artificial Intelligence in No-Limit Poker". En: *Science* 356. DOI: 10.1126/science.aam6960.

Muggleton, S. (1991). "Inductive logic programming". En: *New Generation Computing* 8.4, págs. 295-318. ISSN: 1882-7055. DOI: 10.1007/BF03037089.

Najafabadi, M. M. et al. (2015). "Deep learning applications and challenges in big data analytics". En: *Journal of Big Data* 2.1, pág. 1. ISSN: 2196-1115. DOI: 10.1186/s40537-014-0007-7.

Ng, A. et al. (2011). "Sparse autoencoder". En: *CS294A Lecture notes* 72.2011, págs. 1-19.

Olshausen, B y Field, D (1997). "Sparse coding with an overcomplete basis set: A strategy employed by V1?" En: *Vision research* 37.23, págs. 3311-3325. DOI: 10.1016/S0042-6989(97)00169-7.

Olshausen, B. A. y Field, D. J. (1996). "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". En: *Nature* 381.6583, págs. 607-609. ISSN: 1476-4687. DOI: 10.1038/381607a0.

Orriols-Puig, A. y Bernadó-Mansilla, E. (2008). "Evolutionary rule-based systems for imbalanced data sets". En: *Soft Computing* 13.3, pág. 213. ISSN: 1433-7479. DOI: 10.1007/s00500-008-0319-7.

Papernot, N. et al. (2016). "The Limitations of Deep Learning in Adversarial Settings". En: *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, págs. 372-387. DOI: 10.1109/EuroSP.2016.36.

Pearson, K. (1901). "LIII. On lines and planes of closest fit to systems of points in space". En: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11, págs. 559-572. ISSN: 1941-5982. DOI: 10.1080/14786440109462720.

Polishetty, R., Roopaei, M. y Rad, P. (2016). "A Next-Generation Secure Cloud-Based Deep Learning License Plate Recognition for Smart Cities". En: *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, págs. 286-293. DOI: 10.1109/ICMLA.2016.0054.

Provost, F. y Fawcett, T. (2013). "Data Science and its Relationship to Big Data and Data-Driven Decision Making". En: *Big Data* 1.1, págs. 51-59. DOI: 10.1089/big.2013.1508.

Pulgar, F. J. et al. (2017). "On the impact of imbalanced data in convolutional neural networks performance". En: *International Conference on Hybrid Artificial Intelligence Systems*. Springer, págs. 220-232.

Pulgar, F. J. et al. (2018a). "A First Approach to Face Dimensionality Reduction Through Denoising Autoencoders". En: *19th International Conference on Intelligent Data Engineering and Automated Learning, IDEAL 2018*, 439–447. DOI: 10.1007/978-3-030-03493-1_46.

Pulgar, F. J. et al. (2018b). "AEkNN: An AutoEncoder kNN-Based Classifier With Built-in Dimensionality Reduction". En: *International Journal of Computational Intelligence Systems* 12 (1), págs. 436-452. ISSN: 1875-6883. DOI: 10.2991/ijcis.2018.125905686.

Pulgar, F. J. et al. (2018c). "Análisis del impacto de datos desbalanceados en el rendimiento predictivo de redes neuronales convolucionales." En: *XVIII Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2018)*, págs. 1213-1218.

Pulgar, F. J. et al. (2020a). "Choosing the proper autoencoder for feature fusion based on data complexity and classifiers: Analysis, tips and guidelines". En: *Information Fusion* 54, págs. 44 -60. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2019.07.004.

Pulgar, F. J. et al. (2020b). "ClEnDAE: A classifier based on ensembles with built-in dimensionality reduction through denoising autoencoders". En:

Pumsirirat, A. y Yan, L. (2018). "Credit Card Fraud Detection using Deep Learning based on Auto-Encoder and Restricted Boltzmann Machine". En: *International Journal of Advanced Computer Science and Applications* 9.1. DOI: 10.14569/IJACSA.2018.090103.

Qi, Y. et al. (2014). "Robust feature learning by stacked autoencoder with maximum correntropy criterion". En: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, págs. 6716-6720. DOI: 10.1109/ICASSP.2014.6854900.

Quinlan, J. R. (1996). "Bagging, Boosting, and C4.S". En: *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 1*. AAAI'96. Portland, Oregon: AAAI Press, págs. 725-730. ISBN: 0-262-51091-X.

Quinlan, J. R. (1986). "Induction of decision trees". En: *Machine learning* 1.1, págs. 81-106.

Radovanović, M., Nanopoulos, A. e Ivanović, M. (2010). "Hubs in Space : Popular Nearest Neighbors in High-Dimensional Data". En: *Journal of Machine Learning Research* 11, págs. 2487-2531. ISSN: 15324435.

Ravanelli, M. et al. (2018). "Light Gated Recurrent Units for Speech Recognition". En: *IEEE Transactions on Emerging Topics in Computational Intelligence* 2.2, págs. 92-102. ISSN: 2471-285X. DOI: 10.1109/TETCI.2017.2762739.

Rifai, S. y Muller, X. (2011). "Contractive Auto-Encoders: Explicit Invariance During Feature Extraction". En: *Proceedings of the 28th International Conference on Machine Learning*. Vol. 85. 1, págs. 833-840. ISBN: 978-1-4503-0619-5.

Rifai, S. et al. (2012). "A Generative Process for Sampling Contractive Auto-Encoders". En: *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*. Vol. 2.

Rivera, A. J. et al. (2015). "An ensemble strategy for forecasting the extra-virgin olive oil price in Spain". En: *International work-conference on Time Series (ITISE 2015)*, págs. 506-516.

Robbins, H. y Monro, S. (1951). "A Stochastic Approximation Method". En: *Ann. Math. Statist.* 22.3, págs. 400-407. DOI: 10.1214/aoms/1177729586.

Rokach, L. (2010a). "A survey of Clustering Algorithms". En: *Data Mining and Knowledge Discovery Handbook*. Ed. por O. Maimon y L. Rokach. Boston, MA: Springer US, págs. 269-298. ISBN: 978-0-387-09823-4. DOI: 10.1007/978-0-387-09823-4_14.

Rokach, L. (2010b). "Ensemble-based classifiers". En: *Artificial Intelligence Review* 33.1, págs. 1-39. ISSN: 1573-7462. DOI: 10.1007/s10462-009-9124-7.

Roweis, S. T. y Saul, L. K. (2000). "Nonlinear Dimensionality Reduction by Locally Linear Embedding". En: *Science* 290.5500, págs. 2323-2326. ISSN: 0036-8075. DOI: 10.1126/science.290.5500.2323.

Rumelhart, D. E., Hinton, G. E. y Williams, R. J. (1986). "Learning representations by back-propagating errors". En: *nature* 323.6088, pág. 533. DOI: 10.1038/323533a0.

Sarikaya, R. (2017). "The Technology Behind Personal Digital Assistants: An overview of the system architecture and key components". En: *IEEE Signal Processing Magazine* 34.1, págs. 67-81. ISSN: 1053-5888. DOI: 10.1109/MSP.2016.2617341.

Sarikaya, R., Hinton, G. E. y Deoras, A. (2014). "Application of Deep Belief Networks for Natural Language Understanding". En: *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 22.4, págs. 778-784. ISSN: 2329-9290. DOI: 10.1109/TASLP.2014.2303296.

Schalkoff, R. J. (1997). *Artificial neural networks*. Vol. 1. McGraw-Hill New York.

Schmidhuber, J. (2015). "Deep learning in neural networks: An overview". En: *Neural Networks* 61, págs. 85 -117. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003.

Schwenk, H. y Bengio, Y. (1998). "Training methods for adaptive boosting of neural networks". En: *Advances in neural information processing systems*, págs. 647-653. DOI: 10.1162/089976600300015178.

Shen, Y. et al. (2014). "Learning Semantic Representations Using Convolutional Neural Networks for Web Search". En: *Proceedings of the 23rd International Conference on World Wide Web*. WWW '14 Companion. Seoul, Korea: ACM, págs. 373-374. ISBN: 978-1-4503-2745-9. DOI: 10.1145/2567948.2577348.

Sheng, C. et al. (2013). "Prediction Intervals for a Noisy Nonlinear Time Series Based on a Bootstrapping Reservoir Computing Network Ensemble". En: *IEEE Transactions on Neural Networks and Learning Systems* 24.7, págs. 1036-1048. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2013.2250299.

Shetty, B. (2015). *Curse of Dimensionality*. URL: https://towardsdatascience.com/curse-of-dimensionality-2092410f3d27.

Siam, M. et al. (2017). "Convolutional gated recurrent networks for video segmentation". En: *2017 IEEE International Conference on Image Processing (ICIP)*, págs. 3090-3094. DOI: 10.1109/ICIP.2017.8296851.

Singh, V. et al. (2019). "Feature Learning Using Stacked Autoencoder for Shared and Multimodal Fusion of Medical Images". En: *Computational Intelligence: Theories, Applications and Future Directions - Volume I*. Singapore: Springer Singapore, págs. 53-66. ISBN: 978-981-13-1132-1. DOI: 10.1007/978-981-13-1132-1_5.

Smolensky, P. (1986). "Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1". En: ed. por D. E. Rumelhart, J. L. McClelland y C. PDP Research Group. Cambridge, MA, USA: MIT Press. Cap. Information Processing in Dynamical Systems: Foundations of Harmony Theory, págs. 194-281. ISBN: 0-262-68053-X.

Specht, D. F. (1991). "A general regression neural network". En: *IEEE Transactions on Neural Networks* 2.6, págs. 568-576. ISSN: 1045-9227. DOI: 10.1109/72.97934.

Stallkamp, J. et al. (2012). "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition". En: *Neural Networks* 0. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2012.02.016.

Sun, Y., Wong, A. K. C. y Kame, M. S. (2009). "Classification of imbalanced data: a reviw". En: *International Journal of Pattern Recognition and Artificial Intelligence* 23.04, págs. 687-719. DOI: 10.1142/S0218001409007326.

Sutskever, I., Vinyals, O. y Le, Q. V. (2014). "Sequence to Sequence Learning with Neural Networks". En: *Advances in Neural Information Processing Systems 27*. Ed. por Z. Ghahramani et al. Curran Associates, Inc., págs. 3104-3112.

Sutton, R. S., Barto, A. G. et al. (1998). *Introduction to reinforcement learning*. Vol. 2. 4. MIT press Cambridge.

Szegedy, C. et al. (2015). "Going Deeper with Convolutions". En: *Computer Vision and Pattern Recognition (CVPR)*.

Szegedy, C., Ioffe, S. y Vanhoucke, V. (2016). "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". En: *AAAI Conference on Artificial Intelligence*.

Tan, P.-N., Steinbach, M. y Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0321321367.

Tan, T. et al. (2018). "Adaptive Very Deep Convolutional Residual Network for Noise Robust Speech Recognition". En: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.8, págs. 1393-1405. ISSN: 2329-9290. DOI: 10.1109/TASLP.2018.2825432.

Tang, D., Qin, B. y Liu, T. (2015). "Document Modeling with Gated Recurrent Neural Network for Sentiment Classification". En: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, págs. 1422-1432. DOI: 10.18653/v1/D15-1167.

Tenenbaum, J. B. (2000). "A Global Geometric Framework for Nonlinear Dimensionality Reduction". En: *Science* 290.5500, págs. 2319-2323. ISSN: 00368075. DOI: 10.1126/science.290.5500.2319.

Tian, Y. et al. (2018). "DeepTest: Automated Testing of Deep-neural-network-driven Autonomous Cars". En: *Proceedings of the 40th International Conference on Software Engineering*. ICSE '18. Gothenburg, Sweden: ACM, págs. 303-314. ISBN: 978-1-4503-5638-1. DOI: 10.1145/3180155.3180220.

Tieleman, T. e Hinton, G. (2012). "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". En: *COURSERA: Neural networks for machine learning* 4.2, págs. 26-31.

Toderici, G. et al. (2017). "Full Resolution Image Compression With Recurrent Neural Networks". En: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Van Der Maaten, L. J. P., Postma, E. O. y Van Den Herik, H. J. (2009). "Dimensionality Reduction: A Comparative Review". En: *Journal of Machine Learning Research* 10, págs. 1-41. DOI: 10.1.1.112.5472.

Van Hasselt, H., Guez, A. y Silver, D. (2016). "Deep reinforcement learning with double q-learning". En: *Thirtieth AAAI Conference on Artificial Intelligence*.

Venkatesan, M. (2018). *Artificial Intelligence vs. Machine Learning vs. Deep Learning*. URL: https://www.datasciencecentral.com/profiles/blogs/artificial-intelligence-vs-machine-learning-vs-deep-learning.

Vens, C. et al. (2008). "Decision trees for hierarchical multi-label classification". En: *Machine Learning* 73.2, pág. 185. ISSN: 1573-0565. DOI: 10.1007/s10994-008-5077-3.

Venugopalan, S. et al. (2015). "Translating Videos to Natural Language Using Deep Recurrent Neural Networks". En: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, págs. 1494-1504. DOI: 10.3115/v1/N15-1173.

Vincent, P. et al. (2008a). "Extracting and Composing Robust Features with Denoising Autoencoders". En: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: ACM, págs. 1096-1103. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390294.

Vincent, P. et al. (2008b). "Extracting and composing robust features with denoising autoencoders". En: *Proceedings of the 25th International Conference on Machine Learning*. ACM, págs. 1096-1103. DOI: 10.1145/1390156.1390294.

Vincent, P. et al. (2010). "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". En: *Journal of machine learning research* 11.Dec, págs. 3371-3408.

Waller, M. A. y Fawcett, S. E. (2013). "Data Science, Predictive Analytics, and Big Data: A Revolution That Will Transform Supply Chain Design and Management". En: *Journal of Business Logistics* 34.2, págs. 77-84. DOI: 10.1111/jbl.12010.

Wang, F. et al. (2017). "Residual Attention Network for Image Classification". En: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, K., Zhou, S. y He, Y. (2000). "Growing Decision Trees on Support-less Association Rules". En: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '00. Boston, Massachusetts, USA: ACM, págs. 265-269. ISBN: 1-58113-233-6. DOI: 10.1145/347090.347147.

Wang, T. et al. (2012). "End-to-end text recognition with convolutional neural networks". En: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, págs. 3304-3308.

Wang, W. et al. (2014). "Generalized autoencoder: A neural network framework for dimensionality reduction". En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, págs. 496-503. ISBN: 9781479943098. DOI: 10.1109/CVPRW.2014.79.

Wang, X. (2011). "A fast exact k-nearest neighbors algorithm for high dimensional search using k-means clustering and triangle inequality". En: *Proceedings of the International Joint Conference on Neural Networks*. IEEE, págs. 1293-1299. DOI: 10.1016/j.patcog.2010.01.003.

Wei, W. et al. (2013). "Effective detection of sophisticated online banking fraud on extremely imbalanced data". En: *World Wide Web* 16.4, págs. 449-475. ISSN: 1573-1413. DOI: 10.1007/s11280-012-0178-0.

Wilson, D. L. (1972). "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data". En: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-2.3, págs. 408-421. DOI: 10.1109/TSMC.1972.4309137.

Witten, I. H. et al. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Wolpert, D. H. (1996). "The Lack of A Priori Distinctions Between Learning Algorithms". En: *Neural Computation* 8.7, págs. 1341-1390. ISSN: 0899-7667. DOI: 10.1162/neco.1996.8.7.1341.

Wolpert, D. H. (1992). "Stacked generalization". En: *Neural Networks* 5.2, págs. 241 -259. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(05)80023-1.

Xu, J. (2017). *An Intuitive Guide to Deep Network Architectures*. URL: https://towardsdatascience.com/an-intuitive-guide-to-deep-network-architectures-65fdc477db41.

Xu, J., He, X. y Li, H. (2018). "Deep Learning for Matching in Search and Recommendation". En: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '18. Ann Arbor, MI, USA: ACM, págs. 1365-1368. ISBN: 978-1-4503-5657-2. DOI: 10.1145/3209978.3210181.

Xu, T. et al. (2018). "AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks". En: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, págs. 1316-1324. DOI: 10.1109/CVPR.2018.00143.

Yang, Y., Wu, Q. M. J. y Wang, Y. (2018). "Autoencoder With Invertible Functions for Dimension Reduction and Image Reconstruction". En: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48.7, págs. 1065-1079. ISSN: 2168-2216. DOI: 10.1109/TSMC.2016.2637279.

Yann, L (1987). "Modeles connexionnistes de lapprentissage". Tesis doct. PhD thesis, These de Doctorat, Universite Paris 6.

Yu, C. et al. (2001). "Indexing the Distance: An Efficient Method to KNN Processing". En: *International Conference on Very Large Databases*, págs. 421-430. ISBN: 1-55860-804-4.

Yu, H. y Yang, J. (2001). "A direct LDA algorithm for high-dimensional data-with application to face recognition". En: *Pattern Recognition* 34.10, págs. 2067-2070. DOI: 10.1016/S0031-3203(00)00162-X.

Yu, L. y Liu, H. (2003). "Feature Selection for High-dimensional Data: A Fast Correlation-based Filter Solution". En: *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*. ICML'03. Washington, DC, USA: AAAI Press, págs. 856-863. ISBN: 1-57735-189-4.

Yu, W. et al. (2018). "Multi-view embedded clustering with unsupervised trace ratio LDA". En: *Neurocomputing* 315, págs. 169 -176. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2018.07.014.

Yuan, Z. et al. (2014). "Droid-Sec: Deep Learning in Android Malware Detection". En: *SIGCOMM Comput. Commun. Rev.* 44.4, págs. 371-372. ISSN: 0146-4833. DOI: 10.1145/2740070.2631434.

Yue-Hei Ng, J. et al. (2015). "Beyond Short Snippets: Deep Networks for Video Classification". En: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zadeh, L. A. (1973). "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes". En: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-3.1, págs. 28-44. ISSN: 0018-9472. DOI: 10.1109/TSMC.1973.5408575.

Zadeh, L. (1965). "Fuzzy sets". En: *Information and Control* 8.3, págs. 338 -353. ISSN: 0019-9958. DOI: 10.1016/S0019-9958(65)90241-X.

Zadrozny, B. y Elkan, C. (2001). "Learning and making decisions when costs and probabilities are both unknown". En: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, págs. 204-213.

Zadrozny, B., Langford, J. y Abe, N. (2003). "Cost-sensitive learning by cost-proportionate example weighting". En: *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, págs. 435-442.

Zhang, C. y Zhang, S. (2002). *Association Rule Mining: Models and Algorithms*. Berlin, Heidelberg: Springer-Verlag. ISBN: 3-540-43533-6.

Zhang, R., Isola, P. y Efros, A. A. (2016). "Colorful Image Colorization". En: *Computer Vision – ECCV 2016*. Ed. por B. Leibe et al. Cham: Springer International Publishing, págs. 649-666. ISBN: 978-3-319-46487-9.

Zhang, S., Zhang, C. y Yang, Q. (2003). "Data Preparation for Data Mining." En: *Applied Artificial Intelligence* 17, págs. 375-381. DOI: 10.1080/713827180.

Zhao, J. et al. (2018). "Deep Reinforcement Learning for Sponsored Search Real-time Bidding". En: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '18. London, United Kingdom: ACM, págs. 1021-1030. ISBN: 978-1-4503-5552-0. DOI: 10.1145/3219819.3219918.

Zheng, Y.-J. et al. (2018). "Generative adversarial network based telecom fraud detection at the receiving bank". En: *Neural Networks* 102, págs. 78 -86. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2018.02.015.

Zong, W., Huang, G.-B. y Chen, Y. (2013). "Weighted extreme learning machine for imbalance learning". En: *Neurocomputing* 101, págs. 229 -242. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2012.08.010.

Zuheros, C. et al. (2019). "Deep recurrent neural network for geographical entities disambiguation on social media data". En: *Knowledge-Based Systems* 173, págs. 117 -127. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2019.02.030.

Apéndice A

Publicaciones

El presente anexo contiene los trabajos desarrollados a lo largo de la tesis doctoral y que han sido descritos en los capítulos precedentes. A continuación, se incluyen una serie de secciones cuyo título coincide con los Capítulos 3, 4 y 6, que incluyen los artículos de revista asociados a cada uno de ellos y que componen la presente tesis por compendio. Así mismo, la Sección **??** contiene un artículo de revista publicado durante la elaboración de la presente tesis doctoral pero que no está directamente relacionado con el contenido de la misma.

A.1. Tratamiento de alta dimensionalidad con técnicas DL

El artículo presentado en esta sección está directamente relacionado con el trabajo descrito en el Capítulo 3, sus datos son los siguientes:

- **Título:** AEkNN: An AutoEncoder kNN-Based Classifier With Built-in Dimensionality Reduction.

  Autores: Francisco J. Pulgar, Francisco Charte, Antonio J. Rivera, María J. del Jesus.

  Revista: International Journal of Computational Intelligence Systems, Volume 12, Issue 1, November 2018, Pages 436 - 452.

  ISSN (Online): 1875-6883.

  DOI: 10.2991/ijcis.2018.125905686.

  Estado: Publicado.

International Journal of Computational Intelligence Systems
Vol. **12(1)**; 2019, *pp.* 436–452
DOI: https://doi.org/10.2991/ijcis.2018.125905686; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

AEkNN: An AutoEncoder kNN–Based Classifier With Built-in Dimensionality Reduction

Francisco J. Pulgar *, Francisco Charte, Antonio J. Rivera, María J. del Jesus

Andalusian Research Institute on Data Science and Computational Intelligence (DaSCI), Department of Computer Science, University of Jaén, Jaén, Spain

ARTICLE INFO

Article History
Received 19 Jul 2018
Accepted 13 Feb 2019

Keywords
kNN
Deep learning
Autoencoders
Dimensionality reduction
High dimensionality

ABSTRACT

High dimensionality tends to be a challenge for most machine learning tasks, including classification. There are different classification methodologies, of which instance-based learning is one. One of the best known members of this family is the k-nearest neighbors (kNNs) algorithm. Its strategy relies on searching a set of nearest instances. In high-dimensional spaces, the distances between examples lose significance. Therefore, kNN, in the same way as many other classifiers, tends to worsen its performance as the number of input variables grows. In this study, AEkNN, a new kNN-based algorithm with built-in dimensionality reduction, is presented. Aiming to obtain a new representation of the data, having a lower dimensionality but with more informational features, AEkNN internally uses autoencoders. From this new vector of features the computed distances should be more significant, thus providing a way to choose better neighbors. An experimental evaluation of the new proposal is conducted, analyzing several configurations and comparing them against the original kNN algorithm and classical dimensionality reduction methods. The obtained conclusions demonstrate that AEkNN offers better results in predictive and runtime performance.

1. INTRODUCTION

Classification is a well-known task within the area of machine learning [1]. The main objective of a classifier is to find a way to predict the label to be assigned to new data patterns. To do so, usually a model is created from previously labeled data. In traditional classification, each example has a single label. Different algorithms have been proposed in order to address this work. One of the classic methodologies is instance-based learning (IBL) [2]. Essentially, this methodology is based on local information provided by the training instances, instead of constructing a global model from the whole data. The algorithms belonging to this family are relatively simple, however they have shown to obtain very good results in tackling the classification problem. A traditional example of such an algorithm is k-nearest neighbors (kNNs) [3].

The different IBL approaches, including the kNN algorithm [4], have difficulties when faced with high-dimensional datasets. These datasets are made of samples which contain a large number of features. In particular, the kNN algorithm presents problems when calculating distances in high-dimensional spaces. The main reason is that distances are less significant as the number of dimensions increases, tending to equate [4]. This effect is one of the causes of the curse of dimensionality, which occurs when working with high-dimensional data [5]. Another consequence that emerges in this context is the Hughes phenomenon. This fact implies that the predictive performance of a classifier decreases as the number of features of the dataset grows, keeping the number of examples constant [6]. In other words, more instances would be needed to maintain the same level of performance.

Several approaches have been proposed for to deal with the dimensionality reduction task. Recently, a few proposals based on deep learning (DL) [7, 8] have obtained good results while tackling this problem. The rise of these techniques is produced by the good performance that DL models have had in many research areas, such as computer vision, automatic speech processing, or audio and music recognition. In particular, autoencoders (AEs) are DL networks offering good results due to their architecture and operation [9–11].

High dimensionality is usually mitigated by transforming the original input space into a lower-dimensional one. In this paper an instance-based algorithm, that internally generates a reduced set of features, is proposed. Its objective is to obtain a better IBL method, able to deal with high-dimensional data.

The present study introduces AEkNN, a kNN-based algorithm with built-in dimensionality reduction. AEkNN projects the training patterns into a lower-dimensional space, relying on an AE to do so. The goal is to perform the classification using new features of higher quality and lower dimensionality [9]. This approach is experimentally evaluated, and a comparison between AEkNN and the kNN algorithm is performed considering predictive performance and execution time. The results obtained demonstrate that AEkNN offers better results in both metrics. In addition, AEkNN is compared with other traditional dimensionality reduction algorithms. This comparison offers an idea of the behavior of the AEkNN algorithm when facing the task of dimensionality reduction.

*Corresponding author. Email: fpulgar@ujaen.es

An important aspect of the AEkNN algorithm that must be highlighted is that it performs a transformation of the features of the input data, in contrast to other traditional feature selection algorithms that perform a simple selection of the most significant features. AEkNN performs this transformation taking into account all the characteristics of the input data, although not all will have the same weight in the new generated space.

In short, AEkNN combines two reference methods, kNN and AE, in order to take advantage of kNN in classification and reduce the effects of high dimensionality by means of AE. Summarizing, the main contributions of this study are 1. the design of a new classification algorithm, AEkNN, which combines an efficient dimensionality reduction mechanism with a popular classification method, 2. an analysis of the AEkNN operating parameters that allows selection of the best algorithm configuration, 3. an experimental demonstration of the improvement that AEkNN achieves with respect to the kNN algorithm, 4. an experimental comparison between the use of an AE for dimensionality reduction with respect to other classical methods such as principal components analysis (PCA) and linear discriminant analysis (LDA), 5. a set of guidelines for the reader when using the AEkNN algorithm, and 6. results on the use of AEkNN in real cases.

This paper is organized as follows: In Section 2 some details about the kNN algorithm, DL techniques, and AEs are provided. Section 3 describes relevant studies related to this proposal, focused on tackling the problem of dimensionality reduction in kNN. In Section 4, the proposed AEkNN algorithm is introduced. Section 5 defines the experimental framework, as well as the different results obtained from the experimentation. Finally, Section 6 provides the overall conclusions.

2. PRELIMINARIES

In general terms, machine learning is a subfield of Artificial Intelligence whose aim is to develop algorithms and techniques able to generalize behaviors from information supplied as examples. Classification is one of the tasks performed in the data mining phase. It is a predictive task that usually develops through supervised learning methods [12]. Its purpose is to predict, based on previously labeled data, the class to assign to future unlabeled patterns.

Several issues can emerge while designing a classifier, with some of them related to high dimensionality. According to the Hughes phenomenon [6], the predictive performance of a classifier decreases as the number of features increases, provided that the number of training instances is constant. Another phenomenon that particularly affects IBL methods is the curse of dimensionality. IBL algorithms are based on the similarity of individuals, calculating distances between them [2]. These distances tend to lose significance as dimensionality grows.

AEkNN, the algorithm proposed in this study, is a kNN-based classification method designed to deal with high-dimensional data. This section outlines the essential concepts AEkNN is founded on, such as nearest neighbors classification, DL techniques, and AEs. The kNN algorithm is discussed in Subsection 2.1, while DL and AEs are briefly described in Subsections 2.2 and 2.3.

2.1. The kNN Algorithm

kNN is a nonparametric algorithm developed to deal with classification and regression tasks [3]. In classification, kNN predicts the class for new instances using the information provided by the kNNs, so that the assigned class will be the most common among them. Figure 1 shows a very simple example of how kNN works with different k values. As can be seen, the prediction obtained with $k = 3$ would be B, with $k = 5$ would be A and with $k = 11$ would be A.

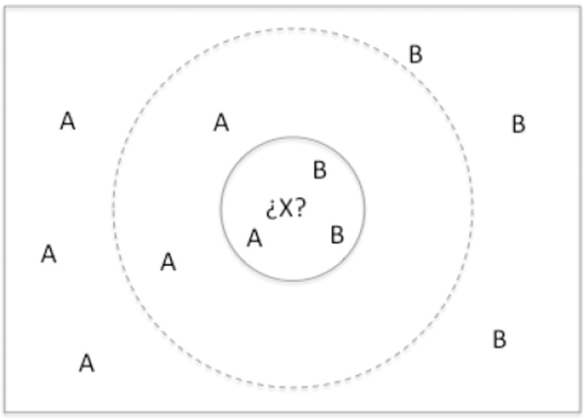


Figure 1 | kNN algorithm in a bidimensional space.

An important feature of this algorithm is that it does not build a model for accomplishing the prediction task. Usually, no work is done until an unlabeled data pattern is encountered, thus the denomination of *lazy* approach [13]. Once the instance to be classified is given, the information provided by its kNNs [14] is used as explained above.

One of kNN's main issues is its behavior with datasets which have a high-dimensional input space. Nowadays, the generation of information is growing in all fields of research. Therefore, when dealing with many real problems it is necessary to use increasingly larger datasets. This fact implies the need to deal with the problem of high-dimensional space when working with machine learning algorithms.

In particular, kNN is affected by the high dimensionality due to the loss of significance of traditional distances as the dimensionality of the data increases [4]. This fact occurs because the distances from the farthest points and closest to any data pattern do not increase as fast as the minimum of the two. This is obviously a problem, since it indicates a poor discrimination of the closest and farthest points with respect to the reference pattern [15]. In such a high-dimensional space distances between individuals tend to be the same. As a consequence similarity-/distance-based algorithms, such as kNN, usually do not offer adequate results.

However, kNN is very popular since it has a good performance, uses few resources, and it is relatively simple [16, 17]. The objective of this proposal is to present an algorithm that combines the advantages of kNN in classification with DL models to reduce dimensionality.

2.2. Deep Learning

DL [7] arises with the objective of extracting higher-level representations of the analyzed data. In other words, the main goal of DL-based techniques is to learn complex representations of data. The main lines of research in this area are intended to define different data representations and create models to learn them [18].

As the name suggests, models based on DL are developed as multilayered (deep) architectures, which are used to map the relationships between features in the data (inputs) and the expected result (outputs) [19, 20]. Most DL algorithms learn multiple levels of representations, producing a hierarchy of concepts. These levels correspond to different degrees of abstraction. The following are some of the main advantages of DL:

- These models can handle a large number of variables and generate new features as part of the algorithm itself, not as an external step [7].
- Provides performance improvements in terms of time needed to accomplish feature engineering, one of the most time-consuming tasks [19].
- Achieves much better results than other methods based on traditional techniques [21] when dealing with problems in certain fields, such as image, speech recognition, or malware detection.
- DL-based models have a high capacity of adaptation for facing new problems [20, 22].

Recently, several new methods [7, 8, 23] founded on the good results produced by DL have been published. Some of them are focused on certain areas, such as image processing and voice recognition [21]. Other DL-based proposals have been satisfactorily applied in disparate areas, gaining advantage over prior techniques [7]. Due to the great impact of DL-based techniques, as well as the impressive results they usually offer, new challenges are also emerging in new research lines [8].

There are two main reasons behind the rise of DL techniques, the large amount of data currently available and the increase in processing power. In this context, different DL architectures have been developed: AEs (Section 2.3), convolutional neural networks [24, 25], long short-term memory [26], recurrent neural networks [27], gated recurrent unit [28], deep Boltzmann machines [29], deep stacking networks [30], deep coding networks [31], deep recurrent encoder [32], deep belief networks [33], among others [7, 20].

DL models have been widely used to perform classification tasks obtaining good results [21]. However, the objective of this proposal is not to perform the classification directly with these models, but use them to the dimensionality reduction ask.

The goal of the present study is to obtain higher-level representations of the data but with a reduced dimensionality. One of the dimensionality reduction DL-based techniques that has achieved good results is the use of AEs [34]. An AE is an artificial neural network whose purpose is to reproduce the input into the output, in order to generate a compressed representation of the original information [20]. Section 2.3 describes this technique in detail.

2.3. Autoencoders

An AE is an artificial neural network able to learn new information encodings through unsupervised learning [9]. AEs are trained to learn an internal representation that allows reproduction the input into the output through a series of hidden layers. The goal is to produce a more compressed representation of the original information in the hidden layers, so that it can be used in other tasks. AEs are typically used for dimensionality reduction tasks owing to their characteristics and performance [10, 11, 35]. Therefore, this is the reason to choose AEs in this paper.

The most basic structure of an AE is very similar to that of a multilayer perceptron. An AE is a feedforward neural network without cycles, so the information always goes in the same direction. AEs are typically formed by a series of layers: an input layer, a series of hidden layers, and an output layer, with the units in each layer connected to those in the next. The main characteristic of AEs is that the output has the same number of nodes as the input, since the goal is to reproduce the latter in the former throughout the learning process [20].

Two parts can be distinguished in an AE, the encoder and the decoder. The first is made up of the input layer and the first half of hidden layers. The second is composed of the second half of hidden layers and the output layer. This is the architecture shown in Figure 2. As can be seen, the structure of an AE always is symmetrical.

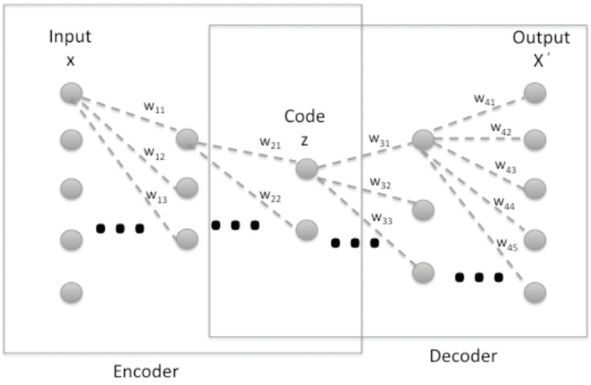


Figure 2 Architecture of autoencoder with three hidden layers.

The encoder and decoder parts in an AE can be defined as functions ω (Equation (1)) and β (Equation (2)), so that:

$$\omega : X \rightarrow F \tag{1}$$

$$\beta : F \rightarrow X \tag{2}$$

where $x \in \mathbb{R}^d = X$ is the input to the AE, and $z \in \mathbb{R}^p = F$ is the mapping contained in the hidden layers of the AE. When there is only one hidden layer (the most basic case) the AE maps the input X onto Z. To perform this, a weight vector W and a bias parameter b are used:

$$z = \gamma_1 (Wx + b) \tag{3}$$

Equation (3) corresponds to the compression function, for which an encoded input representation is obtained. Here, γ_1 is an activation function such as a rectified linear unit or a sigmoid function.

The next step is to decode the new representation z to obtain x', in the same way as x was projected into z, by means of a weight vector W' and a bias parameter b'. Equation (4) corresponds to the decoder part, where the AE reconstructs the input from the information contained in the hidden layer.

$$x' = \gamma_2 (W' z + b') \tag{4}$$

During the training process, the AE tries to reduce the reconstruction error. This operation consists of back-propagating the obtained error through the network, and then modifying the weights to minimize any error. Algorithm 1 shows the pseudocode of this process.

Algorithm 1 AE training algorithm's pseudocode.

Inputs:
TrainData ▷ Train Data
▷ For each training instance:
for each *instance* **in** *TrainData* **do**
  ▷ Do a feed-forward through AE to obtain output:
  newCod ← feedForwardAE(*aeModel, instance*)
  ▷ Calculate error:
  error ← calculateError(*newCod, instance*)
  ▷ Backpropagate the error through AE and perform weight update:
  aeModel ← backpropagateError(*error*)
end for

Learning a representation that allows reproducing the input into the output could seem useless at first sight, but in this case the output is not of interest. Instead, the concern is in the new representation of the inputs learned in the hidden layers. Such new codification is really interesting, because it can have very useful features [19]. The hidden layers learn a higher-level representation of the original data, which can be extracted and used in independent processes.

Depending on the number of units the hidden layers have, two types of AEs can be distinguished:

- Those whose hidden layer has fewer units than the input and output layers. This type of AE is called undercomplete. Its main objective is to force the network to learn a compressed representation of the input, extracting new, higher-level features.
- Those whose hidden layer has more units than the input and output layers. This type of AE is called overcomplete. The main problem in this case is that the network can learn to copy the input to the output without learning anything useful, so when it is necessary to obtain an enlarged representation of the input it is necessary to use other tools to prevent this problem.

In conclusion, AEs are a very suitable tool for generating a new lower-dimensional input space consisting of higher-level features. AEs have obtained good results in performing this task. This is the main reason to choose this technique to design the AEkNN algorithm described later. However, it is important to note that there are other methods of dimensionality reduction that produce good results, such as denoising AEs [36], restricted Boltzmann machines (RBMs) [37], or sparse coding [38]. The objective of this proposal is to present an algorithm that hybridizes kNN with AEs. This establishes a baseline that allows supporting studies with more complex methods.

3. DIMENSIONALITY REDUCTION APPROACHES

In this section, an exploration of previous studies related to the proposal made in this paper is carried out. Subsection 3.1 introduces classical proposals to tackle the dimensionality reduction problem. Some approaches for facing the dimensionality reduction task for kNN are outlined in Subsection 3.2.

In automatic learning, dimensionality reduction is the process aimed to decrease the number of considered variables, by obtaining a subset of main features. Usually two different dimensionality reduction methods are considered:

- Feature selection [39], where the subset of the initial features that provides more useful information is chosen. The final features have no transformation in the process.
- Feature extraction [40], where the process constructs from the initial features a set of new ones providing more useful and nonredundant information, facilitating the next steps in machine learning and in some cases improving understanding by humans.

3.1. Classical Proposals for Dimensionality Reduction

Most dimensionality reduction techniques can be grouped into two categories, linear and nonlinear approaches [41]. Below some representative proposals found in the literature, which can be considered as traditional methods, are shown.

Commonly, classical proposals for dimensionality reduction were developed using linear techniques, such as the following:

- PCA [42] is a well-known solution for dimensionality reduction. Its objective is to obtain the lower-dimensional space where the data are embedded. In particular, the process starts from a series of correlated variables and converts them into a set of uncorrelated linear variables. The variables obtained are known as principal components, and their number is less than or equal to the original variables. Often, the internal structure extracted in this process reflects the variance of the data used.
- Factors analysis [43] is based on the idea that the data can be grouped according to their correlations, that is, variables with a high correlation will be within the same group and variables with low correlation will be in different groups. Thus, each group represents a factor in the observed correlations. The potential factors plus the terms error are linearly combined to model the input variables. The objective of factor analysis is to look for independent dimensions.

- Classical scaling [44] consists in grouping the data according to their similarity level. An algorithm of this type aims to place all data in an N-dimensional space where the distances are maintained as much as possible.

Despite their popularity, classical linear solutions for dimensionality reduction present the problem that they cannot correctly handle complex nonlinear data [41]. For this reason, nonlinear proposals for dimensionality reduction arose. A compilation of these is presented in [45, 46]. Some of these techniques are Isomap [47], Maximum Variance Unfolding [48], diffusion maps [49], manifold charting [41], among others. These techniques allow working correctly with complex nonlinear data. This is an advantage when working with real data, which are usually of this type.

3.2. Proposals for Dimensionality Reduction in kNN

There are different proposals trying to deal with the problems of kNN when working with high-dimensional data. In this section, some of them are collected:

- A method for computing distances for kNN is presented in [50]. The proposed algorithm performs a partition of the total data set. Subsequently, a reference value for each partition made is assigned. Grouping the data in different partitions allows to obtain a space of smaller dimensionality where the distances between the reference points are more significant. The method depends on the division of the data performed and the selection of the reference.

  The experimentation presented in [50] shows an improvement in efficiency. However, the methodology used can be affected by high-dimensional data. In this type of scenario, the distances between the data are less significant, and therefore, the groupings can include instances with very different features. Thus, relevant information can be lost during the process. Likewise, performance depends on the choice of the representative element of each subset, that is, a bad choice can influence the final results.

- The authors of [51] analyze the curse of dimensionality phenomenon, which states that in high-dimensional spaces the distances between the data points become almost equal. The objective is to investigate when the different methods proposed reach their limits. To do this, they perform an experiment in which different methods are compared with each other. In particular, it is found that the kNN algorithm begins to worsen the results when the space exceeds eight dimensions. A proposed solution is to adapt the calculation of distances to high-dimensional spaces. However, this approach does not consider a transformation of the initial data to a lower-dimensional space.

  One of the most important aspects of this study is that it demonstrates how dimensionality affects the kNN method. However, the proposed solution consists of a modification in the calculation of distances. In this way, the consequences of high dimensionality can be alleviated, but solutions must be found that make it possible to reduce the input space. The fundamental reason is that the dimensionality will continue to increase in the future and other methods, like AEkNN, can be better adapted to the new data.

- The proposal in [52] is a kNN-based method called kMkNN, whose objective is to improve the search of the nearest neighbors in a high-dimensional space. This problem is approached from another point of view, with the goal of accelerating the computation of distances between elements without modifying the input space. To do this, kMkNN relies on k-means clustering and the triangular inequality. The study shows a comparison with the original kNN algorithm where it is demonstrated that kMkNN works better considering the execution time, although it is not as effective when predictive performance is taken into account.

  The main problem with the kMkNN method is that it makes a grouping of examples in different clusters. In this way, in spaces of high dimensionality where distances are less significant, instances with different characteristics can be included in the same cluster. This implies a significant loss of relevant information, which translates into worse predictive results. However, the execution time is reduced when clusters are used to classify. Despite the improvement in time, kMkNN method is not a good option as predictive performance is adversely affected.

- A new aspect related to the curse of dimensionality phenomenon, occurring while working with the kNN algorithm, is explored in [53]. It refers to the number of times that a particular element appears as the closest neighbor of the rest of elements. The main objective of the study is to examine the origins of this phenomenon as well as how it can affect dimensionality reduction methods. The authors analyze this phenomenon and some of its effects on kNN classification, providing a foundation which allows making different proposals aimed to mitigate these effects.

  The proposed solution is based on finding *hubs*, that is, popular elements that effectively represent a set of close neighbors. The fundamental problem of this approach is that it does not consider all the input data, but uses a representative for different instances. In this way, relevant information may be removed from the process. Furthermore, the loss of useful information can increase as the dimensionality increases, since the distances between elements are less significant.

- In [15], the problem of finding the nearest neighbors in a high-dimensional space is analyzed. This is a difficult problem both from the point of view of performance and the quality of results. In this study, a new search approach is proposed, where the most relevant dimensions are selected according to certain quality criteria. Thus, the different dimensions are not treated in the same way. This can be seen as an extraction of characteristics according to a particular criterion. Finally, an algorithm based on the previous approach, which tackles the problem of the nearest neighbor, is proposed.

  The proposal made in this study is based on the selection of the most relevant input features. This choice is made according to different preestablished criteria, which implies discarding some input features. Therefore, this process does not take into account all the input characteristics. In certain cases, this fact may not be relevant. However, in other cases, important information may be lost when discarding input features. This

situation has greater effect in spaces of high dimensionality. Therefore, proposals that generate new features by considering all the input information could provide better results.

- A method called DNet-kNN is presented in [54]. It consists in a nonlinear feature mapping based on Deep Neural Network, aiming to improve classification by kNN. DNet-kNN relies on RBMs to perform the mapping of characteristics. RBMs are another type of DL network [55]. The study offers a solution to the problem of high-dimensional data when using kNN by combining this traditional classifier with DL-based techniques. The experimentation carried out proves that DNet-kNN improves the results of the classical kNN.

  The main strength of this algorithm is the use of DL models to carry out dimensionality reduction. In addition, DNet-kNN combine DL model with kNN to improve predictive performance. However, it is based on RBM and requires pretraining the model to generate good results. This implies a previous phase, separated from the proposed algorithm. Another negative aspect is that the experimentation is only based on two datasets (digits and letters) and does not include datasets from other areas, such as medical.

In conclusion, different proposals have arisen to analyze and try to address the problems of IBL algorithms when they have to deal with high-dimensional data. These methods are affected by the curse of dimensionality, which raises the need to find new approaches. Among the previous proposals, there is none that presents a hybrid method based on IBL that incorporates the reduction of dimensionality intrinsically. In addition, there are not many proposals that obtains improvements both in predictive performance as well as in execution time. The present study is aimed to fulfilling these aspects.

4. AEKNN: AN AUTOENCODER KNN-BASED CLASSIFIER WITH BUILT-IN DIMENSIONALITY REDUCTION

Once the main foundations behind our proposed algorithm have been described, this section presents AEkNN. It is an instance-based algorithm with a built-in dimensionality reduction method, performed with DL techniques, in particular by means of AEs.

4.1. AEkNN Foundations

As mentioned, high dimensionality is an obstacle for most classification algorithms. This problem is more noticeable while working with distance-based methods, given that as the number of variables increases these distances are less significant [6, 51]. In such situation an IBL method could lose effectiveness, since the distances between individuals are equated. As a consequence of this problem, different methods which are able to reduce its effects have been proposed. Some of them have been previously listed in Section 3.2.

AEkNN is a new approach in the same direction, introducing the use of AEs to address the problem of high dimensionality. The structure and performance of this type of neural networks makes it suitable for this task. As explained above, AEs are trained to reproduce the input into the output, through a series of hidden layers. Usually, the central layer in AEs has fewer units than the input and output layers. Therefore, this bottleneck forces the network to learn a more compact and higher-level representation of the information. The coding produced by this central layer can be extracted and used as the new set of features in the classification stage. In this sense, there are different studies demonstrating that better results are obtained with AEs than with traditional methods, such PCA or multidimensional scaling [10, 41]. Also, there are studies analyzing the use of AEs from different perspectives, either focusing on the training of the network and its parameters [56] or on the relationship between the data when building the model [35].

AEkNN is an instance-based algorithm designed to address classification with a high number of variables. It starts working with an N-dimensional space X that is projected into an M-dimensional space Z, with $M < N$. This way M new features, presumably of higher level than the initial ones, are obtained. Once the new representation of the input data is generated, it is possible to get more representative distances. To estimate the output, the algorithm uses the distances between each test example and the training ones but based on the M higher-level features. Thus, the drawbacks of high-dimensional data in distances computation can be significantly reduced. As can be seen, AEkNN is a nonlazy instance-based algorithm. It starts by generating the model in charge of producing the new features, unlike the lazy methods that do not have a learning stage or model. AEkNN allows enhancement of predictive performance, as well as obtaining improvements in execution time, when working with data which have a large number of features.

4.2. Method Description

AEkNN consists of two fundamental phases. Firstly, the learning stage is carried out using the training data to generate the AE model that enables production of a new encoding of the data. Secondly, the classification step is performed. It uses the model generated in the first phase to obtain the new representation of the test data and, later, the class for each instance is estimated based on nearest neighbors. Algorithm 2 shows the pseudocode of AEkNN, which is discussed in detail below, while Figure 3 shows a general representation of the algorithm process.

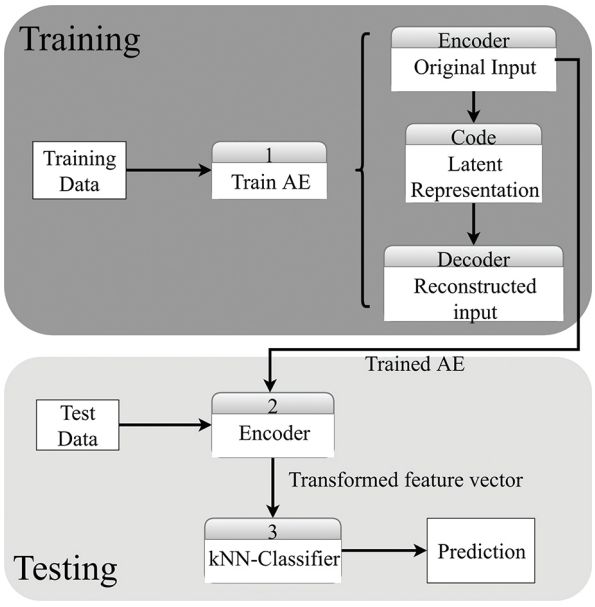


Figure 3 | Method description.

Algorithm 2 AEkNN algorithm's pseudocode.

```
Inputs:
    TrainData                    ▷ Train Data
    TestData                     ▷ Test Data
    PPL                          ▷ Percentage per layer
    k                            ▷ Number of nearest neighbors
1: ▷ Training phase:
2: modelData ← TrainData
3: aeModel ← ()
4: for each layer in PPL do
5:    sizeLayer ← getSizeLayer(modelData, layer)
6:    aeLayer ← getAELayer(modelData, sizeLayer)
7:    modelData ← applyAELayer(aeLayer, modelData)
8:    aeModel ← addAEModel(aeModel, aeLayer)
9: end for
10: ▷ Classification phase:
11: result ← classification(TrainData, TestData, k, aeModel)
12: return result
13:
14: function GetAELayer(modelData, sizeLayer)
15:    aeLayer ← initializeAE(modelData, sizeLayer)
16:    for each instance in modelData do
17:       output ← feedForwardAE(aeLayer, instance)
18:       error ← calculateDeviation(instance, outPut)
19:       aeLayer ← updateWeightsAE(aeLayer, error)
20:    end for
21:    return aeLayer
22: end function
23:
24: function CLASSIFICATION(TrainData, TestData, k, aeModel)
25:    error ← 0
26:    for each instance in TestData do
27:       newCod ← feedForwardAE(aeModel, instance)
28:       output ← distanceBased(newCod, k, TrainData)
29:       if outPut ! = realOutPut(instance) then
30:          error ← error + 1
31:       end if
32:    end for
33:    result ← error / size(TestData)
34:    return result
35: end function
```

The inputs to the algorithm are *TrainData* and *TestData*, the train and test data to be processed, *k*, the number of neighbors, and *PPL*, the percentage of elements per layer (PPL). This latter parameter sets the structure of the AE, that is, the number of layers and elements per layer. It is a vector made of as many items as hidden layers are desired in the AE, each indicating the percentage of units in that layer with respect to the number of input features. In Section 5, different configurations are analyzed to find the one that offers the best results.

The algorithm is divided into two parts. The first part of the code (lines 2–9) corresponds to the training phase of AEkNN. The second part (lines 11–12) refers to the classification phase. During training AEkNN focuses on learning a new representation of the data. This is done through an AE, using the training data to learn the weights linking the AE's units. This is a process that has to be repeated for each layer in the AE, stated by the number of elements in the *PPL* parameter. This loop performs several tasks:

- In line 5 the function *getSizeLayer* is used to obtain the number of units in the layer. This value will depend on the number of characteristics of the training set (*TrainData*) and the percentage applied to the corresponding layer, which is established by the *PPL* parameter.
- The function *getAELayer* (called in line 6 and defined in line 14) retrieves a layer of the AE model. The layer obtains a new representation of the data given as first parameter (*modelData*). The number of units in the AE layer generated in this iteration will be given by the second parameter (*sizeLayer*). Firstly, the AE is initialized with the corresponding structure (line 15). The number of units in the hidden layer is given by the variable computed in the previous step, and the weights are randomly initialized. Secondly, for each training instance the following steps take place:

  1. The AE is used to obtain the output for the given instance (line 17).
  2. The deviation of the given output with respect to the actual one is calculated (line 18).
  3. The weights of the network are updated according to the obtained error (line 19).

  Finally, the generated AE layer is returned (line 21).
- The function *applyAELayer* (line 7) obtains a new representation of the data given as second parameter (*modelData*). To do this, the previously generated AE layer, represented by the first parameter (*aeLayer*), is used.
- The last step consists in adding the AE layer generated in the current iteration to the complete AE model (line 8).

During classification (lines 11–12) the function *classification* is used (lines 24–35). The class for the test instances given as the first parameter (*TestData*) is predicted. The process performed internally in this function is to transform each test instance using the AE model generated in the training phase (*aeModel*), producing a new instance, which is more compact and representative. (line 27). This new set of features is used to predict a class with a classifier based on distances, using for each new example its kNNs (line 28). Finally, this function returns the error rate (*result*) for the total set of test instances (line 33). As can be seen, classification is conducted in a lower-dimensional space, mitigating the problems associated with a high number of variables.

At this point, it should be clarified that the update of weights (lines 16–20) is carried out using mini-batch gradient descent [57]. This is a variation of the gradient descent algorithm that splits the training dataset into small batches that are used to calculate the model error and update the model coefficients. The reason for using this technique is its improved performance when dealing with large dataset.

From the previous description is can be inferred how AEkNN accomplish the objective of addressing classification with high-dimensional data. On the one hand, aiming to reduce the effects of working with a large number of variables, AEs have been used to transform such data into a lower-dimensional space. On the other hand, the classification phase is founded on the advantages of IBL.

4.3. AEkNN Contributions with Respect to Previous Proposals

AEkNN presents significant differences with previous proposals that tackled kNN while dealing with high-dimensional spaces. In this section, the objective is to highlight the main contributions of AEkNN with respect the proposals analyzed in Section 3.2:

- AEkNN is an IBL method that integrates dimensionality reduction. For this reason, it does not need a preprocessing phase where the size of the input space is reduced. This is an improvement over classical kNN and the method presented in [50], which requires a previous grouping of the data to select the most representative patterns.
- AEkNN performs a fusion of features that implies the generation of new features from the original input space. This allows incorporation of the most relevant information of the different characteristics. This implies advantages over the proposal made in [15], where it selects a specific number of original features, discarding the rest, as well as the information that they can provide. Likewise, it is an advantage over the proposals [50, 53] where the elements are grouped into subsets and a representative element is selected.
- The parametrization of AEkNN allows to perform different degrees of dimensionality reduction, according to the characteristics of the problem and the needs of the user. This implies being able to work with a dataset with a very large dimensionality. In this sense, the authors of [51] propose a modification on the calculation of distances. However, this study does not verify its proposal with really large datasets, where the effects of high dimensionality are very significant.
- AEkNN has been proposed to improve both the predictive performance and the execution time of kNN with high-dimensional data. However, the proposal made in [52] focuses only on the improvement in time, while not effective when analyzing the predictive performance.
- The development of AEkNN aims to provide a useful algorithm for different types of datasets. Therefore, the experimentation carried out in this study covers data of different nature. Nevertheless, the proposal made in [54] is based on analyzing only two dataset. In this way, the development of the model is limited to focusing on the properties of specific data.

In this section, an analysis of the contributions of the AEkNN method with respect to other previous proposals has been performed. In order to justify the behavior of the method, extensive experimentation has been designed. To do this, Section 5 analyzes the performance of AEkNN.

5. EXPERIMENTAL STUDY

In order to demonstrate the improvements provided by AEkNN, an experimental study was conducted. It has been structured into three steps, all of them using the same set of datasets:

- The objective of the first phase is to determine how the *PPL* parameter in AEkNN influences the obtained results. For this purpose, classification results for all considered configurations are compared in Subsection 5.2. At the end, the value of the *PPL* parameter that offers the best results is selected.
- The second phase aims to verify whether AEkNN with the selected configuration improves the results provided by the classic kNN algorithm. In Subsection 5.3, the results of both algorithms are compared.
- The third phase of the experimentation aims to assess the competitiveness of AEkNN against traditional dimensionality reduction algorithms, in particular, PCA and LDA. In Subsection 5.4, the results of the three algorithms are compared.

Subsection 5.1 describes the experimental framework and the following subsections present the results and their analysis.

5.1. Experimental Framework

The conducted experimentation aims to show the benefits of AEkNN over a set of datasets with different characteristics. Their traits are shown in Table 1. The datasets' origin is shown in the column named Ref. For all executions, datasets are given as input to the algorithms applying a 2×5 folds cross validation scheme.

Table 1 Characteristics of the datasets used in the experimentation.

Dataset	Samples	Number of Features	Classes	Type	Ref
image	2310	19	7	Real	[58]
drive	58509	48	11	Real	[58]
coil2000	9822	85	2	Integer	[59]
dota	102944	116	2	Real	[58]
nomao	1970	118	2	Real	[58]
batch	13910	128	6	Real	[60]
musk	6598	168	2	Integer	[58]
semeion	1593	256	10	Integer	[58]
madelon	2000	500	2	Real	[61]
hapt	10929	561	12	Real	[62]
isolet	7797	617	26	Real	[63]
mnist	70000	784	10	Integer	[64]
microv1	360	1300	10	Real	[58]
microv2	571	1300	20	Real	[58]

In both phases of experimentation, the value of k for the classifier kNN and for AEkNN will be 5, since it is the recommended value in the related literature. In addition, to compare classification results it was necessary to compute several evaluation measures. In this experimentation, Accuracy (5), F-Score (6), and area under the ROC curve (AUC) (9) were used.

Accuracy (5) is the proportion of true results among the total number of cases examined.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

where TP stands for true positives, instances correctly identified. FP is the false positives, instances incorrectly identified. TN represents the true negatives, instances correctly rejected. FN corresponds to false negatives, instances incorrectly rejected.

F-Score is the harmonic mean of Precision (7) and Recall (8), considering Precision as the proportions of positive results that are true positive results and Recall as the proportion of positives that are correctly identified as such. These measures are defined by the Equations (6–8):

$$F - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{6}$$

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

Finally, AUC is the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. AUC is given by Equation 9:

$$AUC = \int_{\infty}^{-\infty} TPR(T)\, FPR(T)\, dT \tag{9}$$

where *TPR* stands for the true positive rate and *FPR* is false positive rate.

The significance of results obtained in this experimentation is verified by appropriate statistical tests. Two different tests are used in the present study:

- In the first part, the Friedman test [65] is used to rank the different AEkNN configurations and to establish if any statistical differences exist between them.
- In the second part, the Wilcoxon [66] nonparametric sign rank test is used. The objective is to verify if there are significant differences between the results obtained by AEkNN and kNN.

These experiments were run in a cluster made up of 8 computers, each with 2 CPUs (2.33 GHz) and 7 GB RAM. The AEkNN algorithm and the experimentation was coded in R language [67], relying on the H2O package [68] for some DL-related functions. The following is a list of the packages used to implement each part of the algorithm:

- AE: The implementation of the AE has been carried out using the resources provided by the H2O package. This allows the use of functions to generate and train the AE model.
- kNN classifier: To perform the classification based on classical kNN, *kknn* package of R has been used [69].
- R language: The kernel of the AEkNN method where the AE model joins and the classification based on kNN has been implemented using the R language. In this phase, the algorithm has been developed to allow parametrization of the architecture. Therefore, both the AE model training and the classification performed with kNN depend on the parameters of the method.

5.2. PPL Parameter Analysis

AEkNN has a parameter, named *PPL*, that establishes the configuration of the model. This parameter allows the selection of different architectures, both in number of layers (depth) and number of neurons per layer.

The datasets used (see Table 1) have disparate number of input features, so the architectures will be defined according to this trait. Table 2 shows the considered configurations. For each model the number of hidden layers, as well as the number of neurons in each layer, is shown. The latter is indicated as a percentage of the number of initial characteristics. Finally, the notation of the associated *PPL* parameter is provided.

Table 2 Configurations used in the experimentation and PPL parameter.

	Number of Hidden Layers	Number of Neurons (%)			PPL Parameter
		Layer 1	Layer 2	Layer 3	
AEkNN 1	1	25	—	—	(25)
AEkNN 2	1	50	—	—	(50)
AEkNN 3	1	75	—	—	(75)
AEkNN 4	3	150	25	150	(150, 25, 150)
AEkNN 5	3	150	50	150	(150, 50, 150)
AEkNN 6	3	150	75	150	(150, 75, 150)

PPL, percentage of elements per layer; kNN, k-nearest neighbor.

The results produced by the different configurations considered are presented grouped by metric. Table 3 shows the results for Accuracy, Table 4 for *F*-Score, and Table 5 for AUC. These results are also graphically represented. Aiming to optimize the visualization, two plots with different scales have been produced for each metric. Figure 4 represents the results for Accuracy, Figure 5 for *F*-Score, and Figure 6 for AUC.

Table 3 Accuracy classification results for test data.

Dataset	(0.25)	(0.5)	(0.75)	(1.5, 0.25, 1.5)	(1.5, 0.5, 1.5)	(1.5, 0.75, 1.5)
image	0.930	0.945	0.952	0.925	0.938	**0.956**
drive	0.779	0.791	**0.862**	0.763	0.746	0.677
coil2000	**0.929**	0.900	0.898	0.928	0.897	0.897
dota	0.509	0.516	**0.517**	0.510	0.515	0.516
nomao	**0.904**	0.894	0.890	0.902	0.896	0.894
batch	0.995	0.995	0.995	**0.996**	**0.996**	**0.996**
musk	0.974	0.979	0.983	0.982	0.980	**0.991**
semeion	0.904	0.905	**0.909**	0.898	0.905	0.896
madelon	0.532	0.540	**0.547**	0.520	0.510	0.523
hapt	0.936	0.946	**0.950**	0.943	0.947	0.948
isolet	**0.889**	0.885	0.882	0.873	0.876	0.876
mnist	**0.963**	0.960	0.959	0.950	0.954	0.944
microv1	0.857	**0.863**	0.857	0.849	0.856	0.857
microv2	0.625	0.638	0.629	**0.644**	0.639	0.629

Bold values represent the best result in each case for different datasets and metrics.

The results presented in Table 3 and in Figure 4 show the Accuracy obtained for AEkNN with different *PPL* values. These results indicate that there is no one configuration that works best for all datasets. The configurations with 3 hidden layers obtain the best

Table 4 F-score classification results for test data.

Dataset	(0.25)	(0.5)	(0.75)	(1.5, 0.25, 1.5)	(1.5, 0.5, 1.5)	(1.5, 0.75, 1.5)
image	0.930	0.945	0.952	0.925	0.938	**0.956**
drive	0.782	0.796	**0.863**	0.772	0.746	0.683
coil2000	**0.963**	0.947	0.946	0.962	0.946	0.945
dota	0.481	0.487	0.486	0.481	**0.488**	0.485
nomao	**0.905**	0.897	0.893	0.904	0.898	0.897
batch	**0.995**	**0.995**	**0.995**	**0.995**	**0.995**	**0.995**
musk	0.984	0.988	0.990	0.989	0.988	**0.995**
semeion	0.905	0.906	**0.910**	0.899	0.905	0.896
madelon	0.549	0.542	**0.567**	0.531	0.501	0.529
hapt	0.818	0.829	**0.842**	0.833	0.839	0.837
isolet	**0.890**	0.887	0.883	0.875	0.878	0.878
mnist	**0.963**	0.960	0.959	0.950	0.954	0.944
microv1	0.868	**0.872**	0.867	0.861	0.867	0.868
microv2	0.619	0.636	0.625	0.641	**0.643**	0.628

Bold values represent the best result in each case for different datasets and metrics.

Table 5 Area under the ROC curve (AUC) classification results for test data.

Dataset	(0.25)	(0.5)	(0.75)	(1.5, 0.25, 1.5)	(1.5, 0.5, 1.5)	(1.5, 0.75, 1.5)
image	0.932	0.943	0.951	0.923	0.936	**0.956**
drive	0.881	0.889	**0.922**	0.875	0.850	0.826
coil2000	0.526	**0.543**	0.541	0.526	0.538	0.539
dota	0.508	0.514	**0.515**	0.508	0.514	0.514
nomao	**0.903**	0.894	0.890	0.901	0.895	0.893
batch	**0.997**	**0.997**	**0.997**	**0.997**	**0.997**	**0.997**
musk	0.950	0.958	0.966	0.967	0.962	**0.983**
semeion	0.927	0.927	**0.928**	0.923	0.923	0.922
madelon	0.532	0.540	**0.547**	0.520	0.513	0.523
hapt	0.888	0.898	**0.917**	0.900	0.916	0.915
isolet	**0.948**	0.946	0.942	0.938	0.941	0.940
mnist	0.974	0.972	**0.975**	0.965	0.968	0.958
microv1	0.928	**0.934**	0.931	0.932	0.927	0.927
microv2	0.891	**0.897**	0.887	**0.897**	0.890	0.885

Bold values represent the best result in each case for different datasets and metrics.

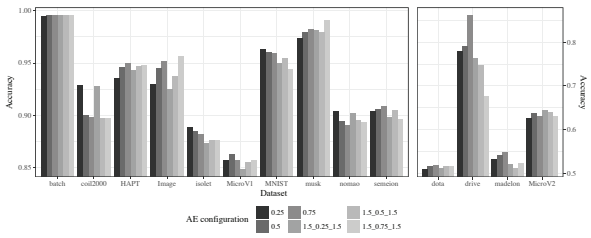


Figure 4 Accuracy classification results for test data.

results in 4 out of 14 datasets, whereas the configurations with 1 hidden layer win in 10 out of 14 datasets. This trend can also be seen in the graphs.

Table 4 and Figure 5 show the *F*-Score obtained by AEkNN with different *PPL* values. The values indicate that the configurations with one single hidden layer obtain better results in 11 out of 14 datasets. The configuration with *PPL* = (0.25) and with *PPL* = (0.75) are the ones winning more times (5). The version with *PPL* = (0.25) shows

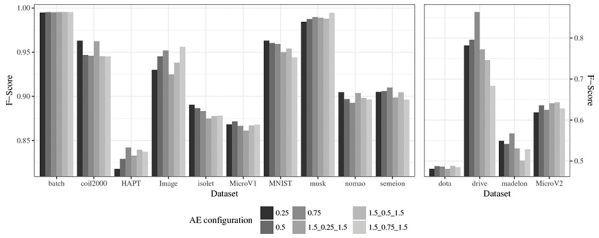


Figure 5 F-score classification results for test data.

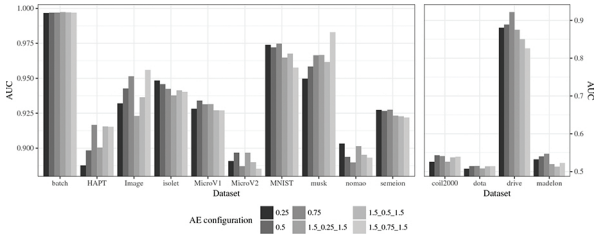


Figure 6 Area under the ROC curve (AUC) classification results for test data.

disparate results, the best values for some cases and bad results for other cases, for example, with *hapt, image,* or *microv2*. Although the version with *PPL* = (0.75) wins the same number of times, its results are more balanced.

In Table 5 the results for AUC obtained with AEkNN can be seen. Figure 6 represents those results. For this metric, it can be appreciated that single hidden layer structures work better, obtaining top results in 12 out of 14 datasets. However, a configuration that works best for all cases has not been found.

Summarizing, the results presented above show the metrics obtained for AEkNN with different *PPL* values. These results show some variability. The *PPL* value that obtains the best performance cannot be determined exactly, since for each dataset there is a setting that works best. However, some initial trends can be drawn:

- Single-layer configuration: Better results than configurations with three hidden layers. For Accuracy and *F*-Score 71% of the best results are with 1 hidden layer, and for AUC this rises to 85%.
- Configurations with *PPL* = (0.75): For AUC 50% of the best results are with this configuration. Their values are close to the best in most cases.
- Configurations with *PPL* = (0.5): 29% of the best results are obtained with this parametrization if AUC is considered. In general, results are good in many cases.
- Other configurations: Some good results but at other times they are far from the best values.

Once the results have been obtained, it is necessary to determine if there are statistically significant differences for each one of them in order to select the best configuration. To do this, the Friedman test [65] will be applied. Average ranks obtained by applying the Friedman test for Accuracy, *F*-Score, and AUC measures are shown in Table 6. In addition, Table 7 shows the different *p-values* obtained by the Friedman test.

Table 6 Average rankings of the different PPL values by measure.

Accuracy		F-Score		AUC	
PPL	**Ranking**	**PPL**	**Ranking**	**PPL**	**Ranking**
(0.75)	2.679	(0.5)	2.923	(0.75)	2.357
(0.5)	2.857	(0.75)	3.000	(0.5)	2.786
(0.25)	3.714	(0.25)	3.429	(1.5, 0.25, 1.5)	3.786
(1.5, 0.75, 1.5)	3.821	(1.5, 0.5, 1.5)	3.500	(0.25)	3.857
(1.5, 0.5, 1.5)	3.892	(1.5, 0.25, 1.5)	4.071	(1.5, 0.5, 1.5)	4.000
(1.5, 0.25, 1.5)	4.036	(1.5, 0.75, 1.5)	4.071	(1.5, 0.75, 1.5)	4.214

PPl, percentage of elements per layer; AUC, area under the ROC curve.

Table 7 Results of Friedman's test (p-values).

Accuracy	F-Score	AUC
0.236	0.423	0.049

AUC, area under the ROC curve.

As can be observed in Table 7, for AUC (which is considered a stronger performance metric) there are statistically significant differences between the considered *PPL* values if we set the *p-value* threshold to the usual range [0.05, 0.1]. However, for Accuracy and *F*-Score there are no statistically significant differences. In addition, in the rankings obtained, it can be seen that there are two specific configurations that offer better results than the remaining ones. In the three rankings presented, the results with *PPL* = (0.75) and *PPL* = (0.5) appear first, clearly highlighted with respect to the other values. Therefore, it is considered that these two configurations are the best ones. Thus, the results of AEkNN with both configurations will be compared against the kNN algorithm.

5.3. AEkNN versus kNN

This second part is focused on determining whether the results obtained with the proposed algorithm, AEkNN, improve those obtained with the kNN algorithm. To do so, a comparison will be made between the results obtained with AEkNN, using the values of the *PPL* parameter selected in the previous section, and the results obtained with kNN algorithm on the same datasets.

First, Table 8 shows the results for each one of the datasets and considered measures, including running time. The results for both algorithms are presented jointly, and the best ones are highlighted in bold. Two plots have been generated for each metric, aiming to optimize data visualization, as in the previous phase, since the range of results was very broad. Figure 7 represents the results for Accuracy, Figure 8 for F-Score, Figure 9 for AUC, and Figure 10 for runtime.

The results shown in Table 8 indicate that AEkNN works better than kNN for most datasets considering Accuracy. On the one hand, the version of AEkNN with *PPL* = (0.75) improves kNN in 11 out of 14 cases, obtaining the best overall results in 6 of them. On the other hand, the version of AEkNN with *PPL* = (0.5) obtains better results than kNN in 11 out of 14 cases, and is the best configuration in 6 of them. In addition, kNN only obtains one best result. Figure 7 confirms this trend. It can be observed that the right bars, where AEkNN results are represented, are higher in most datasets.

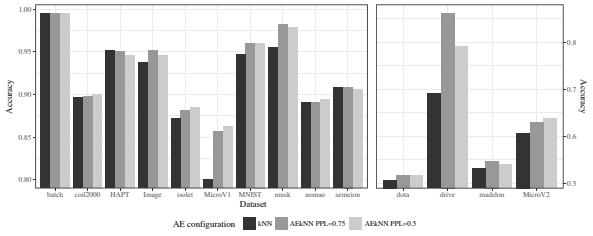


Figure 7 Accuracy results for test data.

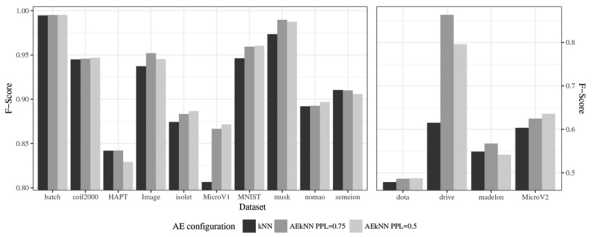


Figure 8 F-score results for test data.

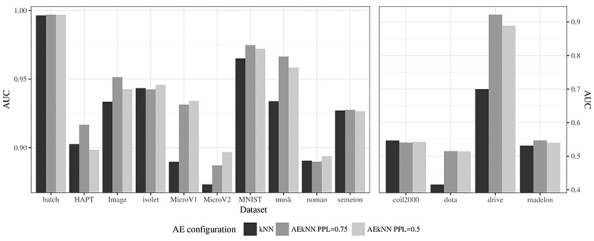


Figure 9 Area under the ROC curve (AUC) results for test data.

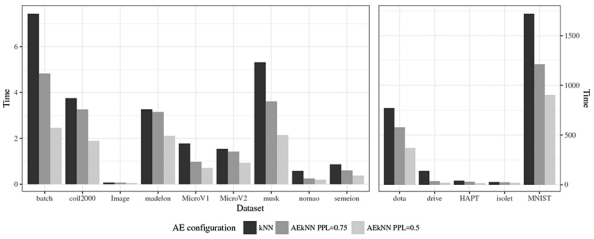


Figure 10 Time results for test data.

Analyzing the data corresponding to the metric *F*-Score, presented in Table 8, it can be observed that AEkNN produces an overall improvement over kNN. The AEkNN version with *PPL* = (0.75) improves kNN in 11 out of 14 cases, obtaining the best overall results in 5 of them. The version of AEkNN with *PPL* = (0.5) obtains better results than kNN in 10 out of 14 cases, and is the best configuration in 7 of them. kNN does not obtain any best result. How the results of the versions corresponding to AEkNN produce higher values than those corresponding to kNN can be seen in Figure 8.

The data related to AUC, presented in Table 8, also show that AEkNN works better than kNN. In this case, the two versions of AEkNN improve kNN in 11 out of 14 cases each one, obtaining the best results in 13 out of 14 cases. Figure 9 shows that the trend is increasing towards the versions of the new algorithm. kNN only obtains one best result, specifically with the *coil2000* dataset, which

may be due to the low number of features in this dataset. Nonetheless, AEkNN performs better than kNN in the rest of metrics for *coil2000* dataset.

The running times for both algorithms are presented in Table 8 and Figure 10. As can be seen, the configuration that takes less time to classify is the one corresponding to AEkNN with *PPL* = (0.5), obtaining the lowest value for all datasets. This is due to the higher compression of the data achieved by this configuration. In the same way, AEkNN with *PPL* = (0.75) obtains better results than the algorithm kNN in all cases.

Summarizing, the following conclusions can be drawn from the previous analysis:

- Accuracy: 93% of the best results are obtained with AEkNN. Both AEkNN configurations considered behave in a similar way. kNN only improves in one case.
- *F*-Score: AEkNN obtains the best results in 100% of the cases. The configurations used have a similar performance. kNN ties with AEkNN in three cases.
- AUC: The best results are obtained with AEkNN in 93% of the datasets. The configuration with *PPL* = (0.75) stands out, achieving the best results in 64% of cases. kNN improves in one case.
- Time: Both AEkNN configurations improve kNN in 100% of cases. AEkNN with *PPL* = (0.5) obtains the best results in all cases.

To sum up, AEkNN performs a transformation of the input space to reduce dimensionality. In spite of this, the quality of the results in terms of classification performance are better than those of kNN in most cases. In addition, in terms of classification time, it can be noted how AEkNN, with higher compression of information, significantly reduces the time spent on classification, without having a negative impact on the other measures.

To determine if there are statistically significant differences between the obtained results, the proper statistical test has been conducted. For this purpose, the Wilcoxon test will be performed, comparing each version of AEkNN against the results of the classical kNN algorithm. In Table 9, the results obtained for Wilcoxon tests are shown.

Table 9 Result of Wilcoxon's test (p-values) comparing kNN versus AEkNN.

	Accuracy	*F*-Score	AUC	Time
AEkNN with PPL = (0.75)	0.0017	0.0003	0.0085	0.0002
AEkNN with PPL = (0.5)	0.0023	0.0245	0.0107	0.0001

kNN, k-nearest neighbor; AUC, area under the ROC curve, PPL, percentage of elements per layer.

As can be seen the *p-values* are rather low, so statistically significant differences between the two AEkNN versions and the original kNN algorithm in all considered measures exist can be concluded, considering the *p-value* threshold within the usual [0.05, 0.1] range. On the one hand, taking into account Accuracy, *F*-Score, and AUC, the configuration with best results is that with 75% of feature reduction. Therefore, this is the optimal solution from the point of view of predictive performance. The reason for this might be that there is less compression of the data, therefore, there is less loss of information compared to the other considered configuration (50%). On the other hand, considering running time, the configuration with best results is that with 50% of feature reduction. It is not surprising that having fewer features allows to compute distances in less time.

5.4. AEkNN vs PCA/LDA

The objective of this third part is to assess the competitiveness of AEkNN against traditional dimensionality reduction algorithms. Specifically, the algorithms used will be PCA [42] and LDA [70], since they are traditional algorithms that offer good results in this

Table 8 Classification results of AEkNN (with different PPL) and kNN algorithm for test data.

	Accuracy			F-Score			AUC			Time (Seconds)		
	kNN	AEkNN		kNN	AEkNN		kNN	AEkNN		kNN	AEkNN	
Dataset		(0.75)	(0.5)		(0.75)	(0.5)		(0.75)	(0.5)		(0.75)	(0.5)
image	0.937	**0.952**	0.945	0.937	**0.952**	0.945	0.934	**0.951**	0.943	0.074	0.073	**0.052**
drive	0.691	**0.862**	0.791	0.615	**0.863**	0.796	0.700	**0.922**	0.889	139.623	36.977	**20.479**
coil2000	0.897	0.898	**0.900**	0.945	0.946	**0.947**	**0.547**	0.541	0.543	3.753	3.262	**1.886**
dota	0.507	**0.517**	0.516	0.479	0.486	**0.487**	0.416	**0.515**	0.514	772.219	578.437	**370.599**
nomao	0.891	0.890	**0.894**	0.892	0.893	**0.897**	0.891	0.890	**0.894**	0.582	0.252	**0.200**
batch	**0.995**	**0.995**	**0.995**	**0.995**	**0.995**	**0.995**	0.996	**0.997**	**0.997**	7.433	4.829	**2.459**
musk	0.956	**0.983**	0.979	0.974	**0.990**	0.988	0.934	**0.966**	0.958	5.317	3.613	**2.144**
semeion	0.908	**0.909**	0.905	**0.910**	**0.910**	0.906	0.927	**0.928**	0.927	0.865	0.601	**0.381**
madelon	0.531	**0.547**	0.540	0.549	**0.567**	0.542	0.532	**0.547**	0.540	3.267	3.150	**2.109**
hapt	**0.951**	0.950	0.946	**0.842**	**0.842**	0.829	0.903	**0.917**	0.898	41.673	30.625	**16.365**
isolet	0.872	0.882	**0.885**	0.874	0.883	**0.887**	0.943	0.942	**0.946**	27.563	24.748	**19.538**
mnist	0.947	0.959	**0.960**	0.946	0.959	**0.960**	0.965	**0.975**	0.972	1720.547	1213.168	**904.223**
microv1	0.800	0.857	**0.863**	0.806	0.867	**0.872**	0.890	0.931	**0.934**	1.776	0.977	**0.709**
microv2	0.607	0.629	**0.638**	0.603	0.625	**0.636**	0.873	0.887	**0.897**	1.542	1.425	**0.933**

PPL, percentage of elements per layer; kNN, k-nearest neighbors; AUC, area under the ROC curve.

Bold values represent the best result in each case for different datasets and metrics.

task [71]. To do so, a comparison will be made between the results obtained with AEkNN, using the values of the *PPL* parameter selected in Section 5.3, and the results obtained with PCA and LDA algorithm on the same datasets. It is important to note that the number of features selected with these methods will be the same as with the AEkNN algorithm, so there are two executions for each algorithm.

First, Table 10 shows the results for each one of the datasets and considered measures. The results for the three algorithms are presented jointly, and the best ones are highlighted in bold. One plot have been generated for each metric aiming to optimize data visualization. In this case, the graphs represent the best value of the two configurations for each algorithm in order to better visualize the differences between the three methods. Figure 11 represents the results for Accuracy, Figure 12 for *F*-Score, and Figure 13 for AUC.

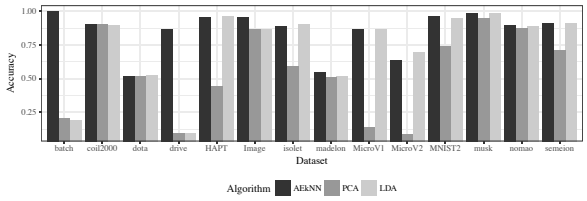


Figure 11 | Accuracy results for test data.

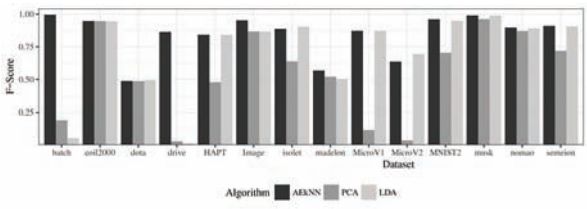


Figure 12 | F-score results for test data.

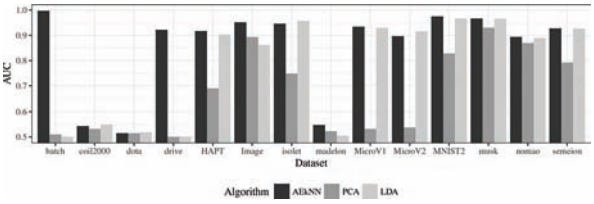


Figure 13 | Area under the ROC curve (AUC) results for test data.

The results shown in Table 10 indicate that AEkNN works better than PCA and LDA for most datasets considering Accuracy. On the one hand, the version of AEkNN with *PPL* = (0.75) improves PCA in 12 out of 14 cases and LDA in 9 out of 14 cases, obtaining the best overall results in 7 of them. On the other hand, the version of AEkNN with *PPL* = (0.5) obtains better results than PCA in 13 out of 14 cases and LDA in 8 out of 14 cases, and is the best configuration in 4 of them. In addition, LDA only obtains the best result in 4 cases. Figure 11 confirms this trend. It can be observed that the bars, where AEkNN results are represented, are higher in most datasets.

Analyzing the data corresponding to the metric *F*-Score, presented in Table 10, it can be observed that AEkNN produces an overall improvement over PCA and LDA. The AEkNN version with *PPL* = (0.75) improves PCA in 12 out of 14 cases and LDA in 10 out of 14 cases, obtaining the best overall results in 6 of them. The version of AEkNN with *PPL* = (0.5) obtains better results than PCA in all cases and LDA in 8 out of 14 cases, and is the best configuration in 4 of them. PCA does not obtain any best result and LDA obtains the best result in 4 cases. How the results of the versions corresponding to AEkNN show values higher than those corresponding to kNN can be seen in Figure 12.

The data related to AUC, presented in Table 10, also show that AEkNN works better than PCA and LDA. The AEkNN version with *PPL* = (0.75) improves PCA in 13 out of 14 cases and LDA in 10 out of 14 cases, obtaining the best overall results in 7 of them. The version of AEkNN with *PPL* = (0.5) obtains better results than PCA in 13 out of 14 cases and LDA in 8 out of 14 cases, and is the best configuration in 4 of them. LDA only obtains the best result in 4 cases. Figure 13 confirms this trend.

Summarizing, these results show some trends that are listed below:

- Accuracy: 71% of the best results are obtained by AEkNN. LDA improves AEkNN in 29% of the cases. AEkNN is always close to the best result. PCA does not surpass AEkNN in any case.
- *F*-Score: AEkNN obtains the best results in 79% of the cases. LDA generates better performance in 21% of the cases. AEkNN always improves PCA.
- AUC: The best results are obtained by AEkNN in 71% of the datasets. LDA achieves the best results in 29% of cases. PCA does not improve in any dataset.

The quality of the results with AEkNN in terms of classification performance are better than those of PCA and LDA in most cases. This means that the high-level features obtained by the AEkNN algorithm provides more relevant information than those obtained by the PCA and LDA algorithms.

Previously, the data obtained in the experimentation have been presented and a comparison between them is made. However, it is necessary to verify whether there are significant differences between the data corresponding to the different algorithms. To do so, the Friedman test [65] will be applied. Average ranks obtained by applying the Friedman test for Accuracy, *F*-Score, and AUC measures are shown in Table 11. In addition, Table 12 shows the different *p-values* obtained by the Friedman test.

As can be observed in Table 12, for Accuracy, *F*-Score, and AUC there are statistically significant differences between the different *PPL* values if we set the *p-value* threshold to the usual range [0.05, 0.1]. It can be seen that AEkNN with *PPL* = (0.75) offer better results than the remaining ones. In the three rankings presented, the AEkNN configurations with *PPL* = (0.75) and *PPL* = (0.5) appear first, clearly highlighted with respect to the other values. Therefore, it is considered that AEkNN obtains better predictive performance, since the reduction of dimensionality generates more significant features.

Table 10 Accuracy, F-Sscore, and AUC classification results of AEkNN (with different PPL), LDA, and PCA for test data.

	Accuracy						F-Score						AUC					
	AEkNN		PCA		LDA		AEkNN		PCA		LDA		AEkNN		PCA		LDA	
Dataset	(0.75)	(0.5)	(0.75)	(0.5)	(0.75)	(0.5)	(0.75)	(0.5)	(0.75)	(0.5)	(0.75)	(0.5)	(0.75)	(0.5)	(0.75)	(0.5)	(0.75)	(0.5)
image	**0.952**	0.945	0.563	0.862	0.676	0.866	**0.952**	0.945	0.580	0.867	0.672	0.866	**0.951**	0.943	0.734	0.893	0.780	0.862
drive	**0.862**	0.791	0.091	0.092	0.091	0.091	**0.863**	0.796	0.023	0.030	0.015	0.015	**0.922**	0.889	0.500	0.491	0.500	0.500
coil2000	0.898	**0.900**	0.825	0.899	0.895	0.895	0.946	**0.947**	0.861	0.946	0.944	0.944	0.541	0.543	0.523	0.532	0.545	**0.549**
dota	0.517	0.516	0.514	0.517	0.519	**0.520**	0.486	0.487	0.486	0.486	0.489	**0.493**	0.515	0.514	0.513	0.515	0.517	**0.519**
nomao	0.890	**0.894**	0.798	0.869	0.818	0.889	0.893	**0.897**	0.821	0.871	0.851	0.891	0.890	**0.894**	0.797	0.869	0.817	0.889
batch	**0.995**	**0.995**	0.195	0.206	0.189	0.152	**0.995**	**0.995**	0.176	0.190	0.053	0.044	**0.997**	**0.997**	0.502	0.510	0.500	0.500
musk	**0.983**	0.979	0.942	0.931	0.982	0.982	**0.990**	0.988	0.962	0.945	0.989	0.989	**0.966**	0.958	0.912	0.930	0.965	0.965
semeion	**0.909**	0.905	0.598	0.706	0.907	0.890	**0.910**	0.906	0.624	0.718	0.906	0.893	**0.928**	0.927	0.732	0.792	0.926	0.913
madelon	**0.547**	0.540	0.506	0.511	0.501	0.512	**0.567**	0.542	0.519	0.503	0.503	0.491	**0.547**	0.540	0.523	0.510	0.505	0.505
hapt	0.950	0.946	0.179	0.443	0.959	**0.960**	**0.842**	0.829	0.180	0.478	0.841	0.841	**0.917**	0.898	0.553	0.690	0.903	0.899
isolet	0.882	0.885	0.424	0.590	0.886	**0.902**	0.883	0.887	0.504	0.638	0.888	**0.903**	0.942	0.946	0.660	0.748	0.951	**0.957**
mnist	0.959	**0.960**	0.740	0.640	0.949	0.943	0.959	**0.960**	0.704	0.704	0.949	0.943	0.975	**0.972**	0.828	0.798	0.966	0.954
microv1	0.857	**0.863**	0.135	0.135	0.818	0.861	0.867	**0.872**	0.116	0.116	0.823	0.871	0.931	**0.934**	0.532	0.532	0.898	0.929
microv2	0.629	0.638	0.058	0.082	0.649	**0.690**	0.625	0.636	0.005	0.037	0.644	**0.695**	0.887	0.897	0.500	0.537	0.891	**0.916**

kNN, k-nearest neighbor; AUC, area under the ROC curve, PPL, percentage of elements per layer; LDA, linear discriminant analysis; PCA, principal components analysis.

Bold values represent the best result in each case for different datasets and metrics.

Table 11 Average rankings of the different dimensionality reduction algorithms by measure.

Accuracy		F-Score		AUC	
Algorithm	**Ranking**	**Algorithm**	**Ranking**	**Algorithm**	**Ranking**
AEkNN PPL = 0.75	2.286	AEkNN PPL = 0.75	2.143	AEkNN PPL = 0.75	1.929
AEkNN PPL = 0.5	2.357	AEkNN PPL = 0.5	2.214	AEkNN PPL = 0.5	2.500
LDA PPL = 0.5	3.000	LDA PPL = 0.75	3.429	LDA PPL = 0.5	2.929
LDA PPL = 0.75	3.571	LDA PPL = 0.5	3.429	LDA PPL = 0.75	3.500
PCA PPL = 0.5	4.321	PCA PPL = 0.5	4.429	PCA PPL = 0.5	4.679
PCA PPL = 0.75	5.464	PCA PPL = 0.75	5.357	PCA PPL = 0.75	5.464

kNN, k-nearest neighbor; AUC, area under the ROC curve, PPL, percentage of elements per layer; LDA, linear discriminant analysis; PCA, principal components analysis.

Table 12 Results of Friedman's test (p-values).

Accuracy	F-Score	AUC
1.266e-05	7.832e-06	7.913e-07

AUC, area under the ROC curve.

5.5. General Guidelines on the Use of AEkNN

AEkNN could be considered as a robust algorithm on the basis of the previous analysis. The experimental study demonstrates that it has good performance with the two *PPL* considered values. From the conducted experimentation some guidelines can also be extracted:

- When working with very high-dimensional datasets, it is recommended to use AEkNN with the *PPL* = (0.5) configuration. In this study, this configuration has obtained the best results for datasets containing more than 600 features. The reason is that the input data has a larger number of features and allows a greater reduction without losing relevant information. Therefore, AEkNN can compress more in these cases.
- When using binary datasets with a lower dimensionality, the AEkNN algorithm with the *PPL* = (0.5) configuration continues to be the best choice. In our experience, this configuration has shown to work better for binary datasets with a number of features around 100. In these cases, the compression may be higher since it is easier to discriminate by class.
- For all other datasets, the choice of configuration for AEkNN depends on the indicator to be enhanced. On the one hand, if the goal is to achieve the best possible predictive performance, the configuration with *PPL* = (0.75) must be chosen. In these cases, AEkNN needs to generate more features. On the other hand, when the interest is to optimize the running time, while maintaining improvements in predictive performance with respect to kNN, the configuration with *PPL* = (0.5) is the best selection. The reason is in the higher compression of the data. AEkNN needs less time to classify lower-dimensional data.

Summarizing, the configuration of AEkNN must be adapted to the data traits to obtain optimal results. For this, a series of tips have been established.

5.6. AEkNN Application to Real Cases

The main objective of this section is to present the results of applying AEkNN to real cases. To do this, two datasets with a large number of features have been used:

Table 13 AEkNN versus kNN results in real cases and %percent improvement.

	Accuracy			F-Score			AUC			Time (Seconds)		
	kNN	AEkNN	%	kNN	AEkNN	%	kNN	AEkNN	%	kNN	AEkNN	%
arcene	0.682	**0.714**	4.692	0.679	**0.712**	4.861	0.689	**0.722**	4.789	34.582	**11.834**	65.779
gisette	0.823	**0.841**	2.187	0.824	**0.844**	2.427	0.823	**0.849**	3.159	256.231	**119.723**	53.275

AUC, area under the ROC curve; kNN, k-nearest neighbor.

- Arcene: This dataset belongs to the medical field. The objective is to derive patterns corresponding to patients with cancer. The dataset have 10000 features and 900 instances [61].
- Gisette: It is an image recognition dataset. It consists in separating the digits 4 and 9. This set contains 5,000 characteristics and 13500 examples [61].

Following the recommendations established in the Section 5.5, the configuration of AEkNN used is with *PPL* = (0.5), since both are datasets with a very high dimensionality. Next, the results obtained with AEkNN and kNN are shown for both datasets.

Table 13 shows the behavior of AEkNN when applying it to real cases. In both datasets the results obtained with AEkNN obtain better performance than those obtained with the classic kNN algorithm. For Arcene, the improvement of AEkNN with respect to kNN in the three metrics considered is higher than 4% and for Gisette it is higher than 2%. In terms of execution time, the reduction obtained is very significant in excess of 50% in both cases.

In summary, the analysis of the application of AEkNN to real cases shows the improvements obtained with the method proposed in this work. Additionally, it provides the reader with an application example for other similar problems.

6. CONCLUDING REMARKS

In this paper, a new classification algorithm called AEkNN has been proposed. This algorithm is based on kNN but aims to mitigate the problem that arises when working with high-dimensional data. To do so, AEkNN internally incorporates a model-building phase aimed at achieving a reduction of the feature space, using AEs for this purpose. The main reason that has led to the design of AEkNN are the good results that have been obtained by AEs when they are used to generate higher-level features. AEkNN relies on an AE to extract a reduced representation of a higher level that replaces the original data.

In order to determine if the proposed algorithm has better behavior than the kNN algorithm, an experimentation process has been followed. Firstly, the analysis of different AE architectures have allowed to determine which structure works better. In this sense, single-layer configurations obtain the best performance in more than 71% of the datasets considering the different metrics.

Furthermore, in the second part of the conducted experimentation, AEkNN with the best configurations have been compared with classical kNN. As has been stated, the results of AEkNN improve those obtained by kNN in all metrics. Specifically, AEkNN obtains the best performance in more than 93% for all metrics. In addition, AEkNN offers a considerable improvement with respect to the time invested in the classification. In this sense, AEkNN works better than kNN in 100% of the datasets.

In addition, a comparison has been made with other traditional methods applied to this problem, in order to verify that the AEkNN algorithm improves behavior when carrying out the dimensionality reduction task. For this, AEkNN has been compared with LDA and PCA. The results show that the proposed AEkNN algorithm improves performance in classification for most of the dataset used. In more detail, AEkNN always improves PCA and exceeds at least 71% of the dataset to LDA, according to the considered metrics. This occurs because the features generated with the proposed algorithm are more significant and provide more relevant information to the classification using distance-based algorithms.

Finally, AEkNN has been applied to solve real problems. Given two datasets corresponding to different fields, AEkNN has been applied following the guidelines set out in the article. This analysis shows clear improvements in terms of predictive performance and execution time with respect to the classic kNN algorithm. It is important to highlight that all the conclusions reached throughout the experimentation have been confirmed by statistical tests where significant differences are obtained.

In conclusion, AEkNN is able to reduce the adverse effects of high-dimensional data while performing instance-based classification, improving both running time and classification performance. This paper shows that the use of AEs can be helpful to solve this kind of obstacle, opening up new possibilities of future work in which they are applied to help solve similar problems presented by other traditional models.

ACKNOWLEDGMENT

The work of F. Pulgar was supported by the Spanish Ministry of Education under the FPU National Program (Ref. FPU16/00324). This work was partially supported by the Spanish Ministry of Science and Technology under project TIN2015-68454-R.

REFERENCES

[1] R.O. Duda, P.E. Hart, D.G Stork, Pattern Classification, Wiley, New York, 1973.
[2] D.W. Aha, D.Kibler, M.K. Albert, Instance-based learning algorithms, Mach. Learn. 6(1) (1991), 37–66.
[3] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory. 13(1) (1967), 21–27.
[4] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is "Nearest Neighbor" meaningful? in International Conference on Database Theory, 7th International Conference, Jerusalem, Israel, 1999, pp. 217–235.

[5] R. Bellman, Dynamic Programming, Princeton University Press, New York, 1957.
[6] G.F. Hughes, On the mean accuracy of statistical pattern recognizers, IEEE Trans. Inf. Theory. 14(1) (1968), 55–63.
[7] L. Deng, Deep learning: methods and applications, Foundations Trends Signal Process. 7(3–4) (2014), 197–387.
[8] Y. Bengio, Deep learning of representations: looking forward, in International Conference on Statistical Language and Speech Processing, 2013, pp. 1–37.
[9] D. Charte, F. Charte, S. García, M.J. del Jesus, F. Herrera, A practical tutorial on autoencoders for nonlinear feature fusion: taxonomy, models, software and guidelines, Inf. Fusion. 44 (2018), 78–96.
[10] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science. 313(5786) (2006), 504–507.
[11] S. Mabu, K. Kobayashi, M. Obayashi, T. Kuremoto, Unsupervised image classification using multi-autoencoder and k-means++, J. Robot. Netw. Artif. Life. 5 (2018), 75.
[12] S.B. Kotsiantis, Supervised machine learning: a review of classification techniques, Informatica. 31 (2007), 249–268.
[13] C.G. Atkeson, A.W. Moorey, S. Schaalz, A.W. Moore, S. Schaal, Locally weighted learning, Artif. Intell. 11 (1997), 11–73.
[14] B.V. Dasarathy, Nearest Neighbor Norms: NN Pattern Classification Techniques, IEEE Computer Society Press, Los Alamitos, CA, 1991.
[15] A. Hinneburg, C.C. Aggarwal, D.A. Keim, What is the nearest neighbor in high dimensional spaces?, in Proceedings of the International Conference on Very Large Databases, Cairo, Egypt, 2000, p. 506–515.
[16] J. Maillo, J. Luengo, S. García, F. Herrera, I. Triguero, Exact fuzzy k-nearest neighbor classification for big datasets, in IEEE International Conference on Fuzzy Systems, IEEE, Naples, Italy. 2017, p. 1–6.
[17] Z. Deng, X. Zhu, D. Cheng, M. Zong, S. Zhang, Efficient knn classification algorithm for big data, Neurocomputing. 195 (2016), 143–148.
[18] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, Pattern Anal. Mach. Intell. IEEE Trans. 35(8) (2013), 1798–1828.
[19] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT Press, Cambridge, MA. 2016.
[20] Y. Bengio, Learning deep architectures for AI, Foundations Trends Mach. Learn. 2(1) (2009), 1–127.
[21] P. Li, J. Xie, Z. Li, T. Liu, W. Yan, Facial peculiarity retrieval via deep neural networks fusion, Int. J. Comput. Intell. Syst. 11 (2018), 58.
[22] J. Liu, Y. An, R. Dou, H. Ji, Dynamic deep learning algorithm based on incremental compensation for fault diagnosis model, Int. J. Comput. Intell. Syst. 11(1) (2018), 846–860.
[23] L. Guan, Y. Wu, J. Zhao, Scan: semantic context aware network for accurate small object detection, Int. J. Comput. Intell. Syst. 11 (2018), 936.
[24] S. Tabik, D. Peralta, A. Herrera-Poyatos, F. Herrera, A snapshot of image pre-processing for convolutional neural networks: case study of mnist, Int. J. Comput. Intell. Syst. 10 (2017), 555–568.
[25] Y. Zhang, X. Cui, Y. Liu, B. Yu, Tire defects classification using convolution architecture for fast feature embedding, Int. J. Comput. Intell. Syst. 11 (2018), 1056.
[26] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9(8) (1997, Nov.), 1735–1780.
[27] H. Sak, A. Senior, F. Beaufays, Long short-term memory recurrent neural network architectures for large scale acoustic modeling, in Proceedings of the Annual Conference of the International Speech Communication Association, 2014, Sept., Singapore, pp. 338–342.
[28] J. Chung, C. Gülçehre, K. Cho, Y. Bengio, Gated feedback recurrent neural networks, in Proceedings of the International Conference on Machine Learning, Lille, France, 2015, p. 2067–2075. http://dl.acm.org/citation.cfm?id=3045118.3045338.
[29] G. Hinton, R. Salakhutdinov, A better way to pretrain deep Boltzmann machines, in Advances in Neural Information Processing Systems, 2012, vol. 3, pp. 2447–2455. http://papers.nips.cc/paper/4610-a-better-way-to-pretrain-deep-boltzmann-machines.pdf.
[30] L. Deng, D. Yu, Deep convex net: a scalable architecture for speech pattern classification, in Proceedings of the Annual Conference of the International Speech Communication Association, Florence, Italy, 2011, pp. 2285–2288.
[31] Y. Lin, T. Zhang, S. Zhu, K. Yu, Deep coding network, in Advances in Neural Information Processing Systems, Vancouver, Canada, 2010, pp. 1405–1413. http://papers.nips.cc/paper/3929-deep-coding-network.pdf.
[32] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. Grue Simonsen, J.-Y. Nie, A hierarchical recurrent encoder-decoder for generative context-aware query suggestion, in Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, ACM, Melbourne, Australia, 2015, pp. 553–562.
[33] H. Lee, R. Grosse, R. Ranganath, A.Y. Ng, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, in Proceedings of the 26th Annual International Conference on Machine Learning, ACM Press, Montreal, Quebec, Canada, 2009, vol. 2008, pp. 609–616.
[34] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, D.-R. Liou, Autoencoder for words, Neurocomputing. 139 (2014, Sept.), 84–96.
[35] S. Rifai, X. Muller, Contractive auto-encoders: explicit invariance during feature extraction, in Proceedings of the 28th International Conference on Machine Learning, Bellevue, WA, 2011, vol. 85, pp. 833–840. https://dl.acm.org/citation.cfm?id=3104587.
[36] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in Proceedings of the 25th International Conference on Machine Learning, ACM, Helsinki, Finland, 2008, pp. 1096–1103.
[37] R. Salakhutdinov, A. Mnih, G. Hinton, Restricted boltzmann machines for collaborative filtering, in Proceedings of the 24th International Conference on Machine learning, ACM, Corvalis, OR, 2007, pp. 791–798.
[38] X. Shaohua, X. Jiwei, L. Xuegui, A sparse auto encoder deep process neural network model and its application, Int. J. Comput. Intell. Syst. 10(1) (2017), 1116–1131.
[39] H. Liu, H. Motoda, Computational Methods of Feature Selection, CRC Press, London, 2007.
[40] H. Liu, H. Motoda, Feature Extraction, Construction and Selection: A Data Mining Perspective, vol. 453, Springer Science & Business Media, Berlin, Heidelberg, Germany, 1998.
[41] L.J.P. Van Der Maaten, E.O. Postma, H.J. Van Den Herik, Dimensionality reduction: a comparative review, J. Mach. Learn. Res. 10 (2009), 1–41. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.112.5472.
[42] K. Pearson, LIII. On lines and planes of closest fit to systems of points in space, Lond. Edinb. Dublin Philos. Mag. J. Sci. 2(11) (1901, Nov.), 559–572.

[43] C. Spearman, "General Intelligence," objectively determined and measured, Am. J. Psychol. 15(2) (1904), 201–292.
[44] W.S. Torgerson, Multidimensional scaling: I. Theory and method, Psychometrika. 401(4) (1952), 401–419.
[45] C.J.C. Burges, Geometric Methods for Feature Extraction and Dimensional Reduction, Springer, Boston, MA, 2005, pp. 59–91.
[46] L.K. Saul, K.Q. Weinberger, J.H. Ham, F. Sha, D.D. Lee, Spectral Methods for Dimensionality Reduction, The MIT Press, Cambridge, MA, 2006, pp. 292–308.
[47] J.B. Tenenbaum, A global geometric framework for nonlinear dimensionality reduction, Science. 290(5500) (2000), 2319–2323.
[48] K.Q. Weinberger, L.K. Saul, An introduction to nonlinear dimensionality reduction by maximum variance unfolding, in Proceedings of the 21st National Conference on Artificial Intelligence, 2006, vol. 2, pp. 1683–1686. https://dl.acm.org/citation.cfm?id=1597471.
[49] R.R. Coifman, S. Lafon, A.B. Lee, M. Maggioni, B. Nadler, F. Warner, S.W. Zucker, Geometric diffusions as a tool for harmonic analysis and structure definition of data: diffusion maps, Proc. Natl. Acad. Sci. 102(21) (2005, May), 7426–7431.
[50] C. Yu, B. Chin Ooi, K.-Lee Tan, H.V. Jagadish, Indexing the distance: an efficient method to KNN processing, in International Conference on Very Large Databases, 2001, pp. 421–430. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.8162
[51] N. Kouiroukidis, G. Evangelidis, The effects of dimensionality curse in high dimensional kNN search, in Proceedings of the 15th Panhellenic Conference on Informatics, IEEE, Kastonia, Greece, 2011, Sept., pp. 41–45.
[52] X. Wang, A fast exact k-nearest neighbors algorithm for high dimensional search using k-means clustering and triangle inequality, in Proceedings of the International Joint Conference on Neural Networks, IEEE, San Jose, CA, 2011, pp. 1293–1299.
[53] M. Radovanović, A. Nanopoulos, M. Ivanović, Hubs in space: popular nearest neighbors in high-dimensional data, J. Mach. Learn. Res. 11 (2010), 2487–2531. https://dl.acm.org/citation.cfm?id=1953015.
[54] R. Min, D.A. Stanley, Z. Yuan, A. Bonner, Z. Zhang, A deep non-linear feature mapping for large-margin kNN classification, in Proceedings of the International Conference on Data Mining, IEEE, Miami, FL, 2009, Dec., pp. 357–366.
[55] G.E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Comput. 14(8) (2002), 1771–1800.
[56] W. Wang, Y. Huang, Y. Wang, L. Wang, Generalized autoencoder: a neural network framework for dimensionality reduction, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Columbus, OH, 2014, pp. 496–503.
[57] G. Hinton, A practical guide to training restricted boltzmann machines, Momentum. 9(1) (2010), 926.
[58] K. Bache, M. Lichman, UCI Machine Learning Repository, School of Information and Computer Sciences, University of California, Irvine, CA, 2013. http://archive.ics.uci.edu/ml.
[59] P. Van Der Putten, M. Van Someren, A bias-variance analysis of a real world learning problem: the coil challenge 2000, Mach. Learn. 57(1) (2004), 177–195.
[60] A. Vergara, S. Vembu, T. Ayhan, M.A. Ryan, M.L. Homer, R. Huerta, Chemical gas sensor drift compensation using classifier ensembles, Sens. Actuators B Chem. 166 (2012, May), 320–329.
[61] I. Guyon, S. Gunn, A. Ben-Hur, G. Dror, Result analysis of the NIPS 2003 feature selection challenge, in Proceedings of Neural Information Processing Systems, Vancouver, British Columbia, Canada, 2004, vol. 4, pp. 545–552. https://papers.nips.cc/paper/2728-result-analysis-of-the-nips-2003-feature-selection-challenge.
[62] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, D. Anguita, Transition-Aware human activity recognition using smartphones, Neurocomputing. 171 (2016, Jan.), 754–767.
[63] R. Cole, M. Fanty, Spoken letter recognition, in Proceedings of the Workshop on Speech and Natural Language, 1990, pp. 385–390.
[64] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE. 86(11) (1998), 2278–2324.
[65] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Stat. Assoc. 32(200) (1937), 675–701.
[66] D.J. Sheskin, Handbook of parametric and nonparametric statistical procedures, Technometrics. 46 (2004), 1193.
[67] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2016. http://softlibre.unizar.es/manuales/aplicaciones/r/fullrefman.pdf.
[68] S. Aiello, T. Kraljevic, Petr Maj, and with Contributions from the H2O.ai Team, H2o: R Interface for H2O, 2016. R package version 3.8.1.3. http://h2o-release. s3. amazonaws. com/h2o/master/3233/docs~···
[69] K. Schliep, K. Hechenbichler, kknn: Weighted k-Nearest Neighbors, 2016. R package version 1.3.1.
[70] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, Cambridge, MA, 2013.
[71] A.M. Martínez, A.C. Kak, PCA versus LDA, IEEE Trans. Pattern Anal. Mach. Intell. 23(2) (2001), 228–233.

A.2. Estudio sobre la relación entre *autoencoder*, método de aprendizaje y problema

La publicación principal asociada al Capítulo 4 es la siguiente:

- **Título:** Choosing the proper autoencoder for feature fusion based on data complexity and classifiers: Analysis, tips and guidelines.

  Autores: Francisco J. Pulgar, Francisco Charte, Antonio J. Rivera, María J. del Jesus.

  Revista: Information Fusion, Volume 54, February 2020, Pages 44-60.

  ISSN: 1566-2535.

  DOI: 10.1016/j.inffus.2019.07.004.

  Estado: Publicado.

Information Fusion 54 (2020) 44–60

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Choosing the proper autoencoder for feature fusion based on data complexity and classifiers: Analysis, tips and guidelines

Francisco J. Pulgar*, Francisco Charte, Antonio J. Rivera, María J. del Jesus

Andalusian Research Institute on Data Science and Computational Intelligence (DaSCI), Computer Science Dpt., University of Jaén, Jaén 23071, Spain

ARTICLE INFO

Keywords:
Classification
Deep learning
Autoencoders
Dimensionality reduction
Feature fusion

ABSTRACT

Classifying data patterns is one of the most recurrent applications in machine learning. The number of input features influences the predictive performance of many classification models. Most classifiers work with high-dimensional spaces. Therefore, there is a great interest in facing the task of reducing the input space. Manifold learning has been shown to perform better than classical dimensionality reduction approaches, such as Principal Component Analysis and Linear Discriminant Analysis. In this sense, Autoencoders (AEs) provide an automated way of performing feature fusion, finding the best manifold to reconstruct the data. There are several models and architectures of AEs. For this reason, in this study an exhaustive analysis of the predictive performance of different AEs models with a large number of datasets is proposed, aiming to provide a set of useful guidelines. These will allow users to choose the appropriate AE model for each case, depending on data traits and the classifier to be used. A thorough empirical analysis is conducted including four AE models, four classification paradigms and a group of datasets with a variety of traits. A convenient set of rules to follow is obtained as a result.

1. Introduction

Machine learning is one of the most widely studied fields of artificial intelligence, due to its extensive application in solving real problems. The generalization of behaviors from a series of training instances is the main objective of the methods developed in this field [1]. Some of the main applications of these algorithms are classification, regression and clustering [2,3]. In particular, classification is one of the most well-known and developed tasks. The main purpose of a classifier is to establish a prediction for new patterns, based on the information provided by training instances. To meet this challenge, different proposals have emerged over time that can be categorized into different methodologies according to their structure and function [4].

There are very diverse approaches among the existing methodologies that handle the classification task. Some of the most commonly used that offer the best results are: Instance-based learning (IBL), which bases the prediction on the information provided by the training data without carrying out a training process [5]; Support Vector Machines (SVMs), which generate a distribution of the examples in the space with the objective of establishing groupings of related instances [6]; Artificial Neural Networks (ANNs), which present an architecture based on the human brain, and whose objective is to establish relationships between the data through an internal training process [7]; And Decision trees (DTs), which are tree-based models, whose architecture allows different features of the data to be evaluated in order to obtain the greatest separability between classes [8].

The approaches included in the previous methodologies have difficulties when working with data that has certain characteristics. In general, these methods are created to work with real data. Therefore, they must take the traits of the input data features into account in order to give the best possible response. One of these characteristics is the high dimensionality of the data. At present, the information captured and stored is growing due to the increase in the generation, reception and storage mechanisms. In this context, the data sets generated have a growing number of features, so the classification algorithms must adapt to this fact. Traditional methodologies decrease their predictive performance when working with data that has a large number of features. This is mainly due to the curse of dimensionality [9,10].

In this context, different proposals have emerged for handling the reduction of dimensionality. The fundamental objective of these methods is to mitigate the effects of high dimensionality on the predictive performance obtained with different classifiers [11,12]. Initially, the process consisted of manual evaluation by an expert who selected the most relevant features. This task was automated in subsequent years, along with the very first methods of feature selection [13]. Some of the best-known algorithms are: Linear Discriminant Analysis (LDA) [14], Principal Component Analysis (PCA) [15], Isometric feature mapping (ISOMAP) [16] and Locally Linear Embedding (LLE) [17]. Over time,

* Corresponding author.
E-mail address: fpulgar@ujaen.es (F.J. Pulgar).

https://doi.org/10.1016/j.inffus.2019.07.004
Received 31 January 2019; Received in revised form 17 July 2019; Accepted 17 July 2019
Available online 18 July 2019

new approaches have emerged to reduce input space dimensionality that improve on aspects of previous tasks or use different methodologies to deal with the problem [18–21].

Recently, the term Feature Fusion has emerged, due to the need to work with multimedia data using machine learning algorithms. The objective of this type of method is to combine features with the aim of eliminating non-relevant information [22–24]. In order to carry out this task, proposals based on deep learning (DL) have emerged that offer effective performance [25]. These good results in certain fields have led to a rise in the use of DL methods [26,27]. Specifically, one of the most suitable models for the feature fusion task is Autoencoders (AEs), due to its operation and architecture [28–34]. In addition to the improvement in predictive performance, the use of dimensionality reduction models allows a considerable reduction in the computation time of the classification algorithms. The reason for this is that the original high-dimensional data leads to a greater cost in computing time; when the input space is reduced, a smaller amount of data is provided to the classification algorithm and the associated time is reduced [30].

The architecture of AEs allows them to learn an internal representation of the input data during the training process. This phase consists of reproducing the input in the output of the network through a series of hidden layers. When the task of reducing the input space dimensionality is tackled, the internal coding must be of smaller dimensionality than the original data. Therefore, the hidden layer that supplies the information must be of smaller dimensionality than the input layer [28,30,35]. This is, however, only an overview of the operation of the AES. There are different models that include variations in different aspects, for example, introducing noise in the input data or changing the loss function. There a numerous different variants of AEs and it is not possible to incorporate all of them in this study. Therefore, four of the most widely used AE models have been considered in this paper. These models are: basic AE [36], denoising AE [37], contractive AE [38] and robust AE [39]. The objective of this study is to carry out an exhaustive study of the performance of the different models of AEs in coping with dimensional reduction. Similarly, the study will not focus on a single classification methodology but rather will aim to evaluate the behavior of classifiers belonging to different methodologies according to the type of AEs used. In this way the reader is provided with a broad set of tests, as well as associated conclusions that allow decisions to be made when facing the task of dimensionality reduction with mediated AEs.

In summary, the objective of this paper is to guide the choice of the proper autoencoder for feature fusion according to the classifier and the data complexity. In this sense, the main contributions are: (1) a parametric analysis of the four models of AEs included in the study that allow users to select the best performing configuration, (2) an experimentation of the four classification algorithms including a comparison of their results with the four AE models and with the original data, (3) an experimental demonstration of the AE model that offers the best performance for each classification methodology, (4) a comparison between AE models and other classical methods, such as PCA, LDA, ISOMAP and LLE, and (5) a series of guidelines that allow the reader to decide which AE model to use according to the characteristics of the input data.

In conclusion, the experimentation presented in this paper shows the great performance of the AEs when facing the dimensionality reduction task. Specifically, the predictive performance of the four classifiers considered after applying the most sophisticated models of AEs clearly improves with respect to the basic AE model. In general, the results generated through any model of AE are better than those that use the raw data. In addition, experimentation shows that AEs behave better than other classic models of dimensionality reduction, such as PCA, LDA, ISOMAP and LLE.

This paper is organized as follows: Section 2 includes the main theoretical concepts that are used throughout the paper: in Section 2.1 the different classification methodologies and the algorithms used in the experimentation are presented; the theoretical foundations of the AEs are described, as well as each of the AE models involved in the experimentation, in Sections 2.3 and 2.4. In Section 3, the experimental framework is defined. The selection of the best architecture of AEs is undertaken in Section 4. Section 5 presents the results obtained after applying different classification algorithms on the data generated by the AE models. In Section 6, a comparison between AEs and classic models of dimensionality reduction is carried out. Section 7 presents a series of guidelines established according to the experience provided by previous experimentation, the objective of which is to facilitate decision-making when faced with the problem of dimensionality reduction. Finally, Section 8 includes the main conclusions reached in this study.

2. Preliminaries

The main objective of this study is to analyze the performance of different models of AEs in tackling the task of dimensionality reduction and how these new representations affect classification methods corresponding to different paradigms.

For this reason, the use of different classification algorithms is considered. In Section 2.1, the main methodologies for classification tasks are presented, as well as the main algorithms used in the subsequent experimentation. In addition, it is necessary to introduce the methods used to perform dimensionality reduction. For this, the concept of AE (2.3) and the main models used (2.4) are presented.

2.1. Classifier paradigms and algorithms

Since the 20th century, different methods for tackling classification tasks have been developed. The proposals can be grouped into several methodologies according to their architecture and operation. The existing paradigms are very diverse, which opens up a large number of possibilities when opting for a specific classification method. Some of these methodologies are: instance-based learning [5], artificial neural networks [7], Bayesian networks [40], support vector machines [6], decision trees [8], rule-based systems [41], genetic algorithms [42] and inductive logic programming [43], among others. This wide variety of options requires experts to know the characteristics of each algorithm, as well as to adapt their choice to the data used in each case. The four paradigms used in this paper are described in more detail bellow, all of which are among the most well-known paradigms:

- Instance-based learning (IBL): This type of algorithm is lazy, that is to say there is no learning process where a model is built. To make the prediction, the information provided by the training instances is used directly in the inference phase [5].
- Artificial Neural Networks (ANNs): These models are inspired by the structure of the human brain. The fundamental elements used in the construction of this type of algorithm are neurons and the connections between them. ANNs use their own experience to identify relationships between the data [7].
- Support Vector Machines (SVMs): These methods relate training instances with points in space. The process, called *kernel trick*, consists of projecting the data patterns into a space of greater dimensionality. Thus, the model manages to separate linearly instances that, initially, were not separable [6].
- Decision trees (DTs): The performance and structure of these algorithms is based on trees. The main elements of these models are the branches, where the attribute that provides more information to make a division in the samples is evaluated, and the leaves, which contain the values of the target class [8].

A specific algorithm of each of these classification methodologies has been chosen aiming to include a representative of each of the most widespread paradigms in the experimentation:

- Within the IBL methodology, one of the best known methods is k-nearest neighbors (kNN). It is a non-parametric algorithm used for classification and regression tasks [44,45]. In classification, kNN

does not build a model to undertake the prediction task. The method does not do any work until it is not necessary to predict a new instance, therefore, it is called the *lazy* approach [46]. Once the new example arrives, kNN predicts the class using the information provided by the k nearest examples, assigning the class that is the most common among the neighbors.

- Multi-layer perceptron (MLP) is the proposal used to evaluate the performance of dimensionality reduction in ANNs. MLP is a traditional model within ANNs [47]. The algorithm can model nonlinear functions and is trained to learn and generalize knowledge from new data. MLP is formed by a series of interconnected elements, known as neurons or nodes. The neurons are organized in different layers and the connections between them are weighted. During the training process, the network uses the back-propagation algorithm to transmit the error throughout the network and adjust the weights to minimize it [48]. The fundamental objective of the model is to map the input data through the layers of the network, generating the output vector [49].
- SVMs are a type of learning algorithm commonly used for classification. This algorithm was originally introduced in 1992 [50] and has been widely used and extended due to its robust performance. This type of model separates the training data provided within a hyperplane that maximizes the distances between them. In this way, elements of similar categories will be closer than examples of different classes. If there is no possible line separation, then the algorithm employs techniques to perform non-linear mapping to a feature space [6].
- There are different proposals for DTs for tackling the task of classification. In this study C4.5 has been selected because it is a classic model within this family and because it is widely used to deal with this type of problem. This algorithm represents the features of the input data as branches and the different values of the target class as leaves of the DT. Finally, the classification rules are obtained from the tree [8,51].

The methodologies and algorithms proposed for dealing with the classification task must take the characteristics of the data used into account. These data are often extracted from different sources. At present, due to the large number of devices and sensors, one of the most frequent characteristics is the high dimensionality of the data. This factor negatively affects the predictive performance of most traditional classifiers. This phenomenon is known as the curse of dimensionality [9,10]. Another consequence associated with high-dimensional data is the need to increase the number of training instances in order to maintain an acceptable level of performance, which is known as the Hughes phenomenon [52].

This factor affects the paradigms specified in this study differently. The IBL algorithms base their operation on the information provided by the nearest instances; therefore, distance is a fundamental element. In spaces of high dimensionality, distances tend to equalize, that is they become less significant, which implies less relevant results. Something similar happens with SVM, since it tries to maximize the distances between data samples of different classes that are less significant with high dimensional data. Similarly, ANNs are affected by the existence of features that do not provide information or are redundant, so that a reduction in dimensionality where more significant features are provided would produce better results. Finally, in some case, decision trees would produce enormous structures based on non-significant features when processing data with a large number of characteristics. The problem of high dimensionality has been widely studied, and in Subsection 2.2 some of the methods proposed to deal with it are detailed.

2.2. Dimensionality reduction methods

The classification methods described in 2.1 are used for prediction tasks based on real data. These data often have high dimensionality, a factor that affects their predictive performance. Therefore, different proposals have emerged with the aim of mitigating its effects [11,12,53]. The first solutions were based on the manual work of the experts who selected the most important characteristics. However, this process was soon automated and different methods of feature selection arose. Some of the most widely used traditional algorithms are: LDA [54], PCA [15,55], ISOMAP [16] and LLE [17]. In recent years, new models have appeared for tackling this task. Advances in technology and the large amount of data available have allowed DL algorithms to be developed. In particular, AEs have shown good performance due to their structure and operation [28–30,56].

One of the first methodologies that tackles the task of dimensionality reduction is feature selection [13]. The process consists of selecting the best subset of input variables. Moreover, feature selection treats each input attribute independently. However, it is well known that the information provided by the variables treated together can be more useful than if they are used independently. In order to solve this factor, other methodologies arose, such as feature extraction and feature fusion [57].

The objective of the feature extraction methodology is to generate the best representation of the input data [21]. This representation will depend on the features of this data and the characteristics of the machine learning algorithm that will be used. To generate the new representation several techniques are used: basic transformations in the data, normalization, discretization and scaling. These types of methods can be classified as supervised (LDA) [54] or non-supervised (PCA) [15,55]. In a similar manner, the new feature space can be obtained through linear combinations of input data, such as in PCA or LDA, or through non-linear combinations, such as ISOMAP [16] or LLE [17]. In the second case, the methods that apply nonlinear dimensionality reduction techniques are known as *manifold learning* [58,59]. These methods have been shown to be very effective in generating new feature spaces, but they have an extremely high computational cost.

As a result, the new term feature fusion has recently emerged [22–24,32–34], due to the need to process multimedia data, especially text, images and sound. The main objective of feature fusion algorithms is to generate new attributes by combining original variables. This way, less relevant and redundant information is eliminated, making the task of automatic learning algorithms more effective [22]. In this context, the use of different DL models has experienced an important rise, specifically regarding AEs. This type of model has shown great performance in discovering *manifolds* between the data automatically, with a much lower computational cost than traditional methods. Therefore, this study focuses on analyzing the behavior of different AE models when tackling the task of dimensionality reduction. In Subsection 2.3 the AE concept and the different models used are presented.

2.3. Autoencoder foundations

AEs are ANNs whose structure is symmetrical. The main objective of this type of model is to reconstruct the input into the output. The network learns using only the input attributes to carry out this process, without the need for any labeling or prior processing; therefore, it is about unsupervised learning. The network can simply copy the input to the output. To avoid this, certain restrictions must be verified [28,30]. Nowadays, the most widely used term used to refer to these structures is autoencoder, but they have also been referred to as diabolo networks [60], autoassociative neural networks [61] and replicator neural networks [62].

The structure of the AEs allows a coded representation of the input information to be obtained in the middle layer. From this coding, the network is able to reconstruct the original input. In cases in which this middle layer is smaller than the input, a representation of lower dimensionality can be obtained [29,56,63]. This fact means that AEs are being widely used to reduce the size of the feature space. Specifically, these models combine variables to remove redundant and irrelevant information, which is known as feature fusion.

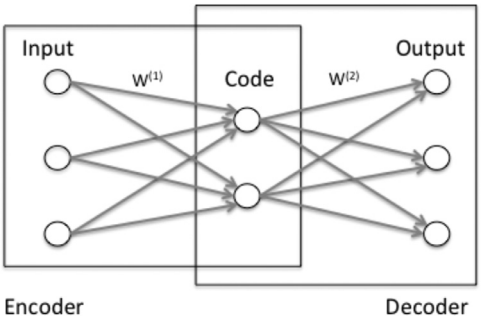


Fig. 1. Architecture of autoencoder with one hidden layer.

In general terms, the basic structure of an AE is a feed-forward neural network [47] without cycles, where information always flows in the same direction. This structure is very similar to the multilayer perceptron. The architecture of the AE is composed of a sequence of layers: an input layer, a series of hidden layers and an output layer. The input and output layers must have the same dimension in order to reproduce the input into the output through the network [64]. The middle layer must be smaller than the input when the objective is to perform feature fusion. This architecture is presented in Fig. 1.

In Fig. 1, the AE has 3 layers. The architecture is divided into two parts. In the first part, the encoder is made up of the first two layers, including the middle encoding one. In the second part the decoder starts in the middle layer and reaches the output layer. Each neuron connects with all of the previous layers, and these connections form weight matrices denoted by W.

AEs can have as many layers as necessary, often located symmetrically as seen in Fig. 1. However, this representation corresponds to the most basic architecture; there are proposals that include modifications. Some of these will be seen in Section 2.4.

In this context, the output layer in the AE can be useful depending on the task in hand. On the one hand, there are studies, such as this one, where this layer is not the central objective, since the purpose is to extract information from the hidden layers. On the other hand, the output layer acquires great importance, for example, in cleaning the noise of the input data.

According to the objectives set out in this paper the importance lies in the information located in the hidden layers. Therefore the restrictions included in the network are fundamental, allowing the model to learn an internal representation of the input data. This new coding corresponds to feature fusion with a higher level of representation. As a result, the process eliminates redundant or unnecessary information. The benefit of the model is that this code can be extracted and used in an independent process [65]. According to the size of the middle layer, there are two types of AEs:

- Undercomplete: The size of the middle layer is smaller than the input. This type of architecture forces the network to learn a coded and compressed representation of the input space. This model is used to carry out feature fusion, where new higher level variables are generated.
- Overcomplete: The size of the middle layer is larger than the input and output layer. This model can be limited by its structure to copying the entry through the network without learning anything useful. So, it is necessary to include restrictions to avoid this. Therefore, the architecture allows for a sparse representation of the input data.

This study focuses on undercomplete AEs, since the aim is to reduce the dimensionality of the original data. This architecture is a very suitable tool for performing feature fusion, obtaining new variables of a higher level and lower dimensionality.

There are several AE models that can be used to tackle the task of dimensionality reduction. However, there are no data in the literature that facilitate the task of selecting the most appropriate method depending on the type of data or the type of classification algorithm to be used. Therefore, this paper aims to provide exhaustive experimentation to facilitate this task. With this aim, the performance of different AE models is analyzed. In Section 2.4 the approaches used in this experimentation are presented.

2.4. Autoencoder models

The feature fusion task using AEs can be achieved using different approaches. In this study we focus on undercomplete models where the coding produced in the intermediate layer is of lower dimensionality. Therefore, overcomplete models are not taken into account, due to their structure. However, these models are widely used to deal with other problems. For example, sparse AEs are suitable for speech [66] and image recognition [67].

As already indicated, the objective of AEs is to find a codification of the data by learning non-linear combinations of their features. In this section, different approaches that lead to a lower-dimensional space are discussed. Specifically, four of the best-known AE models are described: basic, contractive, denoising and robust. This study focuses on these four proposals with the aim of establishing a baseline study on the use of AEs to reduce dimensionality. However, there are many other variants in the literature. In fact, new proposals for AEs arise continuously. Some of these proposals are: Correspondence AEs [68], AE Node Saliency [69] and AE With Invertible Functions [70].

2.4.1. Basic autoencoder

The main objective of the AE models analyzed in this section is to tackle the task of dimensionality reduction on a wide set of datasets with disparate characteristics. Hence, the model obtained will be able to map new examples onto the latent feature space. All AEs start from a basic model, called basic AE (BAE) [36].

In the previous section the basic AE structure has been presented. This consists of feed-forward ANN with symmetrical layer architecture. The symmetry does not necessarily have to be reflected in the weights and activation functions.

The most basic AE, when there is only one hidden layer, is composed of two weight matrices, W and W', and two bias vectors, b and b'. Therefore, where x is the input vector the functions of AE can be expressed as follows:

$$z = f(x) = \gamma_1(Wx + b) \tag{1}$$

$$x' = g(y) = \gamma_2(W'z + b') \tag{2}$$

Eq. (1) corresponds to the compression function, from which an encoded input representation is obtained. Eq. (2) corresponds to the decoder part, where the AE reconstructs the input from the information contained in the hidden layer. Here, γ_1 and γ_2 are two activation functions, which are usually nonlinear.

The objective function for AEs generally corresponds to a per-instance loss function. For example, a widely used metric is the mean square error (MSE). Similarly, the algorithms used to optimize the weights and biases in these models are stochastic gradient descent (SGD) [71] and variants, such as RMSProp [72] or AdaGrad [73].

The gradient descent technique consists of modifying the parameters in order to minimize the objective function [74]. So, by applying the back-propagation algorithm, the necessary gradients are computed [48]. Back-propagation starts by calculating terms of the last layers and transmits the value through the network.

In many cases, a regularization term is added to prevent the overfitting of the model to the training data. The following models incorporate restrictions and mechanisms on basic AE in order to increase its performance.

2.4.2. Contractive autoencoder

AEs are very sensitive to variations in the input data, and small perturbations could generate very different encodings. This is a drawback and motivates the appearance of the model known as contractive AE (CAE) [38]. This type of AE achieves local invariance to changes in the training instances and so it is able to identify lower-dimensional manifold structures more easily.

CAEs include a regularization term that allows them to stabilize the encodings in spite of disturbances in the input data. This new model can generate instances from the learning performed, adding noise at different points and computing its coding [38].

2.4.3. Denoising autoencoder

Denoising AE (DAE) [37] introduces modifications to the basic model in order to achieve greater robustness, that is, allowing the model to reconstruct the input by eliminating any noise that is present.

AEs have previously been applied to the noise elimination task [75]. However, this model has a broader objective, since it takes advantage of noise to build a new feature space that is more resistant to corrupt entries. As such, the field of application is wider and the performance of the AE increases.

The architecture of the DAE model is identical to that of the basic model. The fundamental modification is in the corruption of the entrance during the training phase. An example of this process is given in [37], where a number of input variables are randomly chosen and set to 0. However, the reconstruction is compared to the original unmodified values. Therefore, the AE training process will be adjusted in order to detect the missing values.

DAE can have several hidden layers. Similarly, the training technique can be adapted to other ways of corrupting the input data [76], for example, additive Gaussian noise or salt-and-pepper noise.

2.4.4. Robust autoencoder

Robust AEs (RAE) are networks trained to tolerate possible noise present in the training data [39], just like DAE. However, this task is handled in a different way. The alternative used is to modify the loss function used when training the network. At the time of minimizing the reconstruction error, changes are introduced that reduce the effects of noise in the calculations performed.

Robust stacked AEs incorporate this idea with the aim of being more tolerant to input noise than basic models. To accomplish this, the proposal in [77] uses an error function based on correntropy.

Correntropy allows us to measure the probability density that two events are similar. The outliers affect this measure to a lesser extent than the MSE. Therefore, RAE attempts to maximize this measurement, implying greater resilience to noise.

3. Experimental study

In order to analyze the improvements of tackling the task of feature fusion using AEs, an exhaustive experimental study has been carried out. In addition, this analysis compares different AE models with the aim of determining which of them offers better performance. Thus the experimentation performed follows the steps detailed bellow:

- Firstly, the different models of AEs are used to generate a new feature space from that of the input data. For this, different AE architectures are used in each case. In this way several subsets with different degrees of reduction are obtained from the original datasets. The models of AEs used are: basic, contractive, denoising and robust.
- Secondly, the classification with the different methods is carried out. Each classification algorithm works with the different data sets generated in the previous phase. In this way classifications are performed for the different subsets of the same dataset. This process allows us to establish comparisons between the AE models and architectures used when generating reduced subsets of the input data. The classifiers used are: kNN, MLP, SVM and C4.5.

In the following sections, the analysis of the experimentation is presented. Fundamentally, the objectives pursued are:

- Determine which structure of AEs offers better predictive performance. For this purpose, classification results for the different configurations are compared in Section 4. Due to the large number of results generated, this step is necessary in order to focus the subsequent analysis on just one architecture.
- Perform a comparison of the classification results obtained by the different classifiers in Section 5. The comparative data will be those corresponding to the classification of the subsets obtained with the four models of AEs with the configuration selected in Section 3 and the classification of the original data without reduction of dimensionality. The purposes of Section 5 are:
  - Present, in Section 5.1, the experimental framework of the classification algorithms used in this study.
  - Analyze, by type of classifier, the performance of the different models of AEs in Sections 5.2–5.5.
  - Establish a general analysis of the results supported by statistical tests in Section 5.6.
  - Study the execution time of the classification algorithms considering the different AE architectures in Section 5.7.
- Carry out a comparison of the AEs' performance when tackling the task of dimensionality reduction compared to the traditional methods outlined in Section 6.
- Propose guidelines to facilitate the task of dimensionality reduction using AEs in Section 7.

In addition, Section 3.1 presents the framework used in the experimentation.

3.1. Experimental framework

This study aims to analyze the performance of different models of AEs when dealing with the reduction of dimensionality. Similarly, the experimentation aims to determine the behavior of different classifiers depending on the type of AE. For this, a wide range of datasets with very varied characteristics has been used. Their traits are presented in Table 1, which also shows the origin of the dataset in the Ref column. When performing the executions, a 2×5 fold cross validation scheme was applied.

The datasets described in Table 1 are those used by all models of AEs to establish a comparison based on a large data set with different characteristics. In addition, establishing a form of evaluation is necessary in order to assess the predictive performance of different methods and models. In this case the area under the ROC curve (AUC) has been used. This metric offers a robust view of the predictive performance of the models evaluated. That is to say, AUC provides a reliable overview of the results, something that has not been obtained with other metrics such as Precision or Accuracy, which give a more partial view. AUC is the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. AUC is given by the Eq. (3):

$$AUC = \int_{\infty}^{-\infty} TPR(T)FPR(T)dT \tag{3}$$

where *TPR* stands for the true positive rate and *FPR* is false positive rate.

In this study, the *pROC* package for R, which contains a set of tools for displaying and analyzing ROC curves, is used to calculate AUC for both binary and multi-class datasets [87]. This package obtains multiclass AUC as defined by Hand and Till [88].

Finally, two statistical tests are used in this study to verify the significance of the results obtained. This will allow us to establish the conclusions associated with the study. The tests performed are:

- The Friedman test [89] is used to rank the different configurations. In this way, the selection of the best architecture and model can be carried out.

Table 1
Characteristics of the datasets used in the experimentation.

Dataset	Number of			Type	Field	Ref
	Samples	Features	Classes			
arcene	900	10,000	2	Real	Medical	[78]
batch	13,910	128	6	Real	Chemical	[79]
coil2000	9822	85	2	Integer	Social	[80]
dota	102,944	116	2	Real	Game	[81]
drive	58,509	48	11	Real	Motor	[81]
facial	2964	301	2	Real	Image	[81]
fashionmnist	70,000	784	10	Integer	Image	[82]
gisette	13,500	5000	2	Integer	Image	[78]
hapt	10,929	561	12	Real	Activity	[83]
image	2310	19	7	Real	Image	[81]
isolet	7797	617	26	Real	Image	[84]
letter	20,000	16	26	Integer	Image	[81]
madelon	2000	500	2	Real	Artificial	[85]
mfeat	2000	649	10	Real	Image	[81]
microv1	360	1300	10	Real	Biology	[81]
microv2	571	1300	20	Real	Biology	[81]
mnist	70,000	784	10	Integer	Image	[86]
musk	6598	168	2	Integer	Physical	[81]
nomao	1970	118	2	Real	Technology	[81]
semeion	1593	256	10	Integer	Image	[81]

- The Li post-hoc tests [90] for Friedman test are applied in order to compare all the configurations and allow us to verify whether there are significant differences between these results. The Li test is a non-parametric method appropriate when the number of samples is not very large and it is a good alternative for finding significant differences [91].

The equipment used to develop this set of executions is a cluster made up of 18 computers, with 2 CPUs (2.33 GHz) of 7 GB RAM each. The cluster is mounted using Rocks 6.1.1, a cluster Linux distribution. Rocks 6.1.1 is based upon CentOS 6.5. Therefore, CentOS 6.5 is the operating system included in each of the nodes of the cluster. The different models of AEs were coded in Python language, using the Keras library [92]. The classification methods used were programmed using R language [93] and are detailed in Subsection 5.1. Finally, it is important to note that during the executions of this study the cluster has been dedicated exclusively to them, without any other type of work executed in the system. In addition, two parallel tasks have been executed in each node, as each one has 2 CPUs.

4. Autoencoders architectures analysis

The objective of this section is to study which structure of AEs offers a better predictive performance. In this way, the subsequent analysis will focus on the best architecture obtained. Section 4.1 describes the different architectures used and Subsection 4.2 presents the results.

4.1. Autoencoders architectures framework

In this subsection, the architectures used with the different AE models to perform the feature fusion are presented. One of the fundamental characteristics of these models is that they have symmetrical architecture where the number of neurons in the input layer is equal to that of the output layer. In this work, AEs are used to reduce dimensionality, therefore, the intermediate layer must have a lower dimensionality than the input and output layers. In Table 2, the different configurations are shown.

To evaluate the different models, the same selection of several architectures were used for all models of AEs. The architectures proposed in Table 2 make a reduction in the number of original features of 75% (ARQ_75), 50% (ARQ_50) and 25% (ARQ_25), respectively. Thus, AEs will essentially have the following layers:

Table 2
Autoencoder architectures used in the experimentation.

	Number of layers	Number of neurons (% of total)		
		Input	Hidden	Output
ARQ_25	3	100	25	100
ARQ_50	3	100	50	100
ARQ_75	3	100	75	100

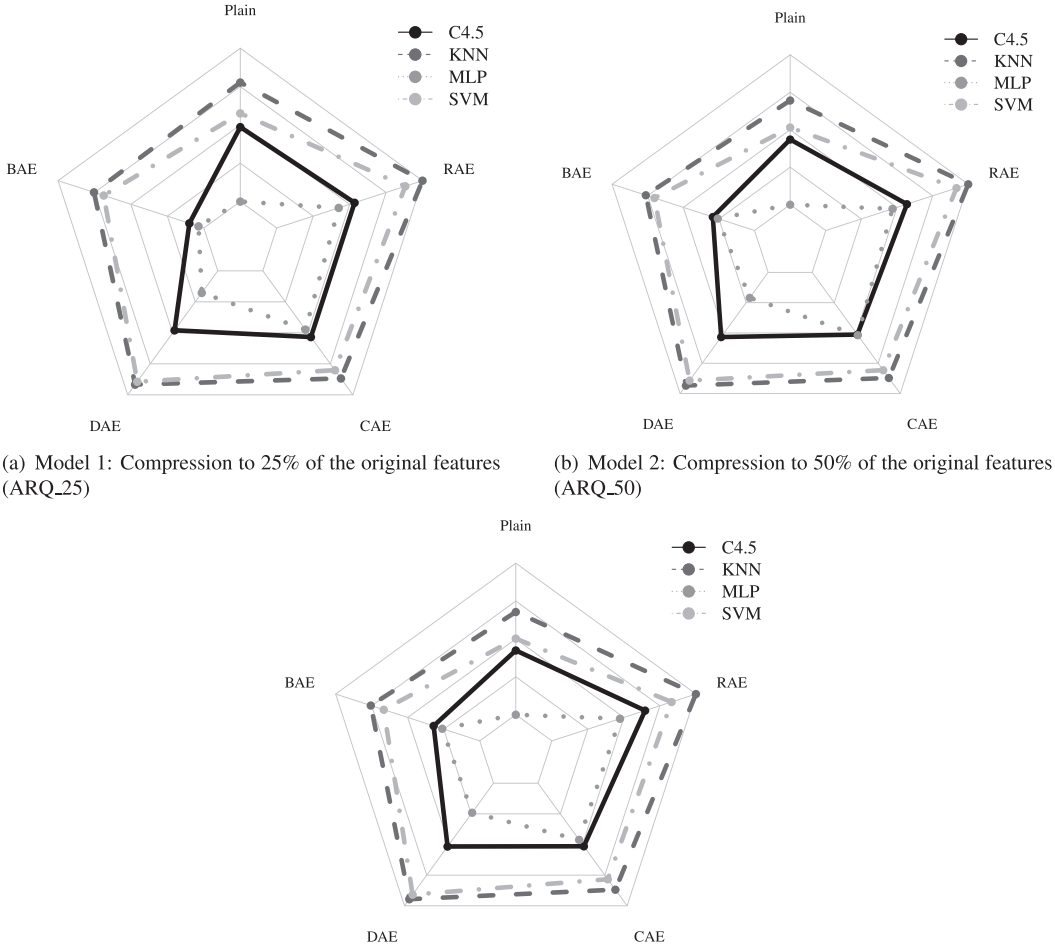
- Input layer: This part has as many neurons as the original dataset has features.
- Hidden layer: The number of units is variable in each configuration according to the reduction carried out.
- Output layer: Due to the symmetric architecture of AEs, this layer has the same number of elements as the input.

As indicated above, the architectures used have a single hidden layer. The objective is to make the evaluation of basic architectures to establish a baseline study. However, AEs can be expanded with more layers while maintaining their symmetrical structure. In this experimentation, the architectures considered have a single hidden layer. The main reason is, as has been shown in previous experiments [30], that including more hidden layers in the form of stacked AEs does not imply an improvement in predictive performance.

4.2. Autoencoders architectures analysis

The main objective of this Section is to study which AE configuration provides a better predictive performance with the different classifiers. Due to the large number of results generated, the objective is to establish one of the configurations in order to later perform a more detailed analysis of classifiers.

The visualization using plots of the results generated can provide a first approximation of their quality and of which models work better. Therefore, the first step is to graphically represent the results obtained for the different configurations. Fig. 2(a)–(c) present the results obtained by compressing 25% (ARQ_25), 50% (ARQ_50) and 75% (ARQ_75), respectively. The three plots show the aggregated results by configuration, AE model and classifier. The classifiers are represented by the different strokes, while the vertices of the plot correspond to the different AE models.



(a) Model 1: Compression to 25% of the original features (ARQ_25)

(b) Model 2: Compression to 50% of the original features (ARQ_50)

(c) Model 3: Compression to 75% of the original features (ARQ_75)

Fig. 2. Predictive performance (AUC) of the different autoencoder models and classification algorithms for each architecture.

Table 3
Average rankings of the different autoencoder architectures by classification method.

kNN		SVM		MLP		C4.5	
Architecture	Ranking	Architecture	Ranking	Architecture	Ranking	Architecture	Ranking
ARQ_75	1.850	ARQ_75	2.037	ARQ_75	1.850	ARQ_75	1.825
ARQ_50	2.075	ARQ_50	2.050	ARQ_50	2.150	ARQ_50	2.300
ARQ_25	2.750	ARQ_25	2.662	ARQ_25	2.725	Plain	2.837
Plain	3.325	Plain	3.250	Plain	3.275	ARQ_25	3.037

Plots 2(a)–(c) show the results for the different configurations proposed in this study. Each figure is a radar chart that represents the results obtained with the 4 models of AE and the model without performing compression. Similarly, there is a line for each classification algorithm and so, a global vision of the performance of the different models is generated. The representation is an aggregation of all the datasets used.

In general, RAE works better in all cases, since it can be observed that for all classification methods it tends to the maximum value. On the contrary, the worst results are obtained with BAE and with the model without compression, which is expected since they are the less sophisticated models and contain original data without processing, respectively.

Regarding the different classification methods, some patterns can be observed. On the one hand, for the kNN, SVM and MLP algorithms, the different models of AE improve in the three configurations on execution without compression. On the other hand, C4.5 performs the worst, since in some cases the execution without compression improves some of the models of AE used.

Once a global vision of the results obtained has been presented, the objective is to select the configuration with the best predictive performance. Due to the large number of executions carried out, it is necessary to perform a more detailed subsequent analysis.

To determine the best configuration, it is necessary to verify whether there are significant differences between them. To do so, first the Friedman test [89] is applied. Average ranks obtained by applying this test for AUC are shown in Table 3. These rankings have been generated by adding data from all datasets and AE models. In this case, the objective is to select the best architecture, therefore it is not necessary to separate the results by AE model. This more detailed analysis will be shown in Section 5.

As can be seen in Table 3, the configuration that provides the best predictive performance is that which reduces the number of features

Table 4
Li post-hoc Friedman test for classification algorithms by autoencoder architecture.

		ARQ_25	ARQ_50	ARQ_75	Plain
kNN	ARQ_25	-	-	-	**6.601E-03**
	ARQ_50	**1.292E-03**	-	-	**1.253E-09**
	ARQ_75	**1.423E-05**	**1.703E-01**	-	**6.817E-13**
	Plain	-	-	-	-
SVM	ARQ_25	-	-	-	**7.571E-02**
	ARQ_50	**5.229E-02**	-	-	**8.464E-08**
	ARQ_75	**4.311E-02**	9.512E-01	-	**5.837E-08**
	Plain	-	-	-	-
MLP	ARQ_25	-	-	-	**8.147E-03**
	ARQ_50	**5.617E-03**	-	-	**4.148E-08**
	ARQ_75	**2.114E-05**	**1.416E-01**	-	**3.413E-12**
	Plain	-	-	-	-
C4.5	ARQ_25	-	-	-	-
	ARQ_50	**4.497E-04**	-	-	**1.242E-02**
	ARQ_75	**4.236E-09**	**2.882E-02**	-	**1.047E-06**
	Plain	3.272E-01	-	-	-

to 75% (ARQ_75) of the total. In addition, the worst results are obtained with the execution without compression, except for C4.5 where the worst results are obtained with the configuration ARQ_25 (25% of the total).

To select a configuration based on solid results, it is necessary to verify whether there are statistically significant differences. Therefore, Li post-hoc tests [90] for Friedman test are applied in order to compare all the configurations. Table 4 shows the different *p-values* obtained by the Li test. In this table, the p-values are represented when the architecture of the row has a better ranking than the architecture of the column.

As can be seen in Table 4, there are significant differences between the different configurations proposed in this study if we set the *p-value* threshold to the usual range [0.05, 0.1]. The only cases where there are no clearly significant differences is for SVM, between the models that reduce to 50% (ARQ_50) and 75% (ARQ_75) of the total, and with C4.5, between the models that reduce to 25% and the model without reduction. However, the results presented in this section allow us to establish a solid base for selecting the architecture that reduces 75% of the total as the best in terms of predictive performance.

5. Classification performance analysis

Once the configuration that generates a better predictive performance in classification has been selected in Section 4, the next objective is to perform a more detailed analysis of the different AE models and how they affect the classification methods proposed in this study. For this, Section 5.1 presents the framework used for the different classification algorithms and Sections 5.2–5.5 present the results obtained by kNN, SVM, MLP and C4.5, respectively. In these subsections the results are presented in greater detail, showing the values of AUC for each one of the datasets and for each AE model. The different tables present the performance of the four classifiers when using the original dataset without reduction and the subsets that were generated after carrying out fusion features with the AE models. In all of the tables, the best result for each dataset is highlighted in bold. In addition, Section 5.6 establishes a general analysis where statistical tests are used to support the results and Section 5.7 studies the execution time of the different architectures considered.

5.1. Classification algorithm framework

The main objective of this paper is to carry out an exhaustive study that analyzes the performance of AE models in handling the task of dimensionality reduction. This experimentation allows us to offer useful information when using these models according to the characteristics of the data and the classification algorithms that are used. In Section 3.1, data sets with different traits have been described. In this Subsection, the classification algorithms used, their origin and their main parameters are presented.

The classification algorithms are implemented in R [93], using in each case a specific package of this platform. Another important aspect is the parameterization of the classification algorithms. Bellow, the main characteristics of the different algorithms are shown:

- For the kNN algorithm, the *kknn* package has been used [94]. Similarly, the value of parameter *k* is 5, since it is the one recommended in the literature [30].
- The SVM algorithm has been obtained from the *e1071* package [95]. The parameters used are those established by default in the package documentation including radial kernel.
- To perform the classification with the MLP algorithm, *caret* and *RSNNS* packages were used [96,97]. The structure of the network and the number of iterations are determined by the default parameters of the package.
- The C4.5 algorithm was provided by the *RWeka* package [98]. The default parameters were used in the implementation of C4.5.

The parameters used in four algorithms are those provided by default for the different algorithms, since the objective is to assess the predictive performance obtained by reducing with AEs without adjusting the methods in any other way.

5.2. kNN

Table 5 shows the results obtained using the kNN algorithm. These data show that in most cases the best results are obtained with data sets reduced by AEs. Specifically, in 10 out of 20 cases the best result is obtained with RAE, in 4 out of 20 datasets DAE generates the best predictive performance and in 5 out of 20 the best data is obtained by CAE. However, there are 2 cases where the best results are obtained with the original dataset.

These results show the improvement of predictive performance occurs when reducing dimensionality using AEs. The justification for this can be found in the functional scheme of the IBL algorithms, specifically, kNN. The behavior of this type of method is affected by the high dimensionality of the input space, because in this scenario the distances between the instances tend to equalize. Therefore, the loss of significance of this factor leads to a fall in predictive performance [99]. In this context, AEs are incorporated. The objective is to reduce the dimensionality of the input data using several AE models, aiming to obtain distances between the examples that are more significant. AEs carry out the fusion features, where new spaces are generated by adding the most relevant information from the original data.

Another aspect to note is that for 18 datasets the best results were obtained with sophisticated AE models (CAE, DAE, RAE), which is obvious since they incorporate improvements over BAE. Therefore, the generated feature fusion contains more and more relevant information and is more useful when carrying out the classification.

5.3. SVM

In Table 6 the classification results of the SVM algorithm are presented. The trend of the previous algorithm is maintained according to the data shown. The best results are obtained by classifying subsets generated by AEs in most cases. In more detail, the RAE model shows the best performance in 8 out of 20 cases, DAE works best in 5 out of 20 datasets and CAE generates better results in 5 out of 20 cases. Finally, there are no improvements for the other 3 datasets, where the best performance is obtained with the plain data.

The behavior of the SVM algorithm when decreasing the input space by means of AEs clearly improves with respect to the results with the

Table 5
kNN results for plain data and autoencoder models (AUC).

Dataset	Plain Data	BAE	CAE	DAE	RAE
arcene	0.693 ± 0.003	0.591 ± 0.011	0.605 ± 0.015	0.672 ± 0.005	**0.743** ± 0.008
batch	0.870 ± 0.005	0.877 ± 0.003	0.900 ± 0.010	0.990 ± 0.001	**0.992** ± 0.001
coil2000	0.546 ± 0.003	0.543 ± 0.002	**0.554** ± 0.004	**0.554** ± 0.004	0.545 ± 0.003
dota	0.416 ± 0.021	0.515 ± 0.002	0.519 ± 0.000	**0.522** ± 0.003	0.516 ± 0.002
drive	0.800 ± 0.009	0.819 ± 0.011	**0.877** ± 0.005	0.846 ± 0.006	0.873 ± 0.004
facial	**0.795** ± 0.011	0.668 ± 0.005	0.690 ± 0.006	0.685 ± 0.003	0.697 ± 0.002
fashionmnist	0.906 ± 0.004	0.914 ± 0.003	0.912 ± 0.001	0.920 ± 0.003	**0.922** ± 0.001
gisette	0.824 ± 0.005	0.879 ± 0.009	0.897 ± 0.004	**0.923** ± 0.002	0.900 ± 0.004
hapt	0.903 ± 0.005	0.918 ± 0.004	0.876 ± 0.005	0.931 ± 0.003	**0.939** ± 0.002
image	0.929 ± 0.005	0.952 ± 0.000	0.952 ± 0.002	0.950 ± 0.001	**0.958** ± 0.003
isolet	0.943 ± 0.003	0.942 ± 0.004	0.974 ± 0.001	0.968 ± 0.002	**0.976** ± 0.002
letter	**0.973** ± 0.015	0.818 ± 0.007	0.925 ± 0.012	0.923 ± 0.006	0.947 ± 0.005
madelon	0.526 ± 0.004	0.550 ± 0.003	0.566 ± 0.006	0.591 ± 0.003	**0.618** ± 0.005
mfeat	0.703 ± 0.004	0.982 ± 0.002	**0.986** ± 0.002	0.984 ± 0.001	0.984 ± 0.001
microv1	0.920 ± 0.005	0.933 ± 0.002	0.922 ± 0.003	**0.939** ± 0.005	0.929 ± 0.004
microv2	0.883 ± 0.006	0.896 ± 0.003	**0.955** ± 0.002	0.932 ± 0.005	0.954 ± 0.003
mnist	0.965 ± 0.003	0.958 ± 0.001	0.969 ± 0.000	0.975 ± 0.001	**0.979** ± 0.000
musk	0.829 ± 0.005	0.914 ± 0.004	0.940 ± 0.002	0.941 ± 0.001	**0.947** ± 0.002
nomao	0.891 ± 0.001	0.891 ± 0.003	**0.914** ± 0.005	0.904 ± 0.002	0.905 ± 0.003
semeion	0.927 ± 0.003	0.929 ± 0.002	0.907 ± 0.004	0.940 ± 0.001	**0.942** ± 0.000

Table 6
SVM results for plain data and autoencoder models (AUC).

Dataset	Plain Data	BAE	CAE	DAE	RAE
arcene	0.500 ± 0.000	0.500 ± 0.000	0.500 ± 0.000	**0.632** ± 0.005	0.500 ± 0.000
batch	0.923 ± 0.003	0.933 ± 0.005	0.924 ± 0.004	0.980 ± 0.002	**0.985** ± 0.002
coil2000	0.500 ± 0.000	0.500 ± 0.000	0.500 ± 0.000	0.500 ± 0.000	0.500 ± 0.000
dota	0.509 ± 0.002	0.500 ± 0.000	**0.553** ± 0.001	0.550 ± 0.002	0.534 ± 0.005
drive	0.885 ± 0.005	0.934 ± 0.007	**0.985** ± 0.003	0.941 ± 0.002	0.863 ± 0.004
facial	**0.802** ± 0.003	0.697 ± 0.006	0.701 ± 0.004	0.713 ± 0.002	0.714 ± 0.005
fashionmnist	0.932 ± 0.002	0.933 ± 0.005	0.938 ± 0.003	0.940 ± 0.000	**0.942** ± 0.001
gisette	0.803 ± 0.008	0.821 ± 0.007	0.816 ± 0.001	**0.969** ± 0.003	0.955 ± 0.006
hapt	**0.900** ± 0.001	0.895 ± 0.001	0.871 ± 0.003	0.889 ± 0.001	0.886 ± 0.004
image	0.846 ± 0.004	0.853 ± 0.003	0.886 ± 0.001	0.877 ± 0.005	**0.910** ± 0.002
isolet	0.954 ± 0.003	0.968 ± 0.002	0.971 ± 0.002	0.975 ± 0.001	**0.976** ± 0.001
letter	**0.966** ± 0.003	0.790 ± 0.006	0.841 ± 0.004	0.828 ± 0.004	0.879 ± 0.005
madelon	0.569 ± 0.004	0.569 ± 0.004	0.580 ± 0.002	**0.591** ± 0.001	0.587 ± 0.000
mfeat	0.970 ± 0.002	0.982 ± 0.001	**0.985** ± 0.002	**0.985** ± 0.001	**0.985** ± 0.001
microv1	0.500 ± 0.000	0.774 ± 0.004	0.796 ± 0.003	**0.918** ± 0.005	0.786 ± 0.002
microv2	0.500 ± 0.000	0.849 ± 0.002	**0.916** ± 0.003	0.871 ± 0.001	0.881 ± 0.001
mnist	0.981 ± 0.002	0.984 ± 0.001	0.984 ± 0.001	0.985 ± 0.001	**0.987** ± 0.000
musk	0.838 ± 0.004	0.893 ± 0.004	0.939 ± 0.005	0.948 ± 0.006	**0.964** ± 0.003
nomao	0.914 ± 0.001	0.915 ± 0.001	**0.935** ± 0.000	0.924 ± 0.002	0.932 ± 0.001
semeion	0.891 ± 0.000	0.913 ± 0.002	0.957 ± 0.003	0.957 ± 0.001	**0.962** ± 0.001

plain data. The philosophy of the SVM algorithm is based on the maximization of distances between data samples of different classes [6]. In this sense, the distance between the instances tends to equalize in spaces of high dimensionality. This leads to a considerable loss of predictive performance and justifies the use of methods to mitigate its effects [10].

In this study, the use of AEs allows the generation of a new space of features where the distances are more significant. During the process, different AE models are trained to discard irrelevant or redundant information. In this new scenario the SVM algorithm improves the distribution of instances, generating better predictive results.

The results for SVM confirm that more complex AE models offer better results in 17 datasets. These results provide a solid basis for considering the use of AEs to pre-process the data when classification is performed using SVM-based algorithms. This is especially recommended when the data has high-dimensional input space.

5.4. MLP

The classification results generated by the MLP algorithm are shown in Table 7. These data reflect that the predictive performance is improved by reducing the dimensionality using the different models of AEs. This continues to confirm the trend marked by the previous algorithms. In this case, all datasets show improvements when applying AEs. Specifically, RAE generates the best results in 10 out of 20 cases, DAE obtains the best predictive performance in 2 out of 20 datasets, CAE in 6 out of 20 sets and BAE is the best model in the 2 remaining cases.

The results shown in Table 7 confirm that the predictive performance of a basic neural network (MLP) improves by reducing the input space using different models of AEs. The MLP algorithm corresponds to a basic neural network whose objective is to learn to predict the class of a new instance from a series of input instances. During the training process, the network adjusts its parameters in order to generate the smallest possible error in the training data [47]. The information of each of the classes that the network generalizes is worse if the dimensionality of the data increases, so it is necessary to use different methods that improve this.

The AEs used in this experimentation are also ANNs. Therefore, the feature fusion produced generates information that follows a ”natural” process when used as input to another ANN. In fact, this process is very similar to classification schemes based on DL, which do not use any external models [65]. ANN convention allows us to extract higher-level information that improves the predictive performance of MLP.

Following the trend marked by previous algorithms, MLP results indicate that complex AE models offer the best performance in 18 datasets.

Table 7
MLP results for plain data and autoencoder models (AUC).

Dataset	Plain Data	BAE	CAE	DAE	RAE
arcene	0.522 ± 0.001	0.492 ± 0.003	0.465 ± 0.005	0.500 ± 0.000	**0.523** ± 0.001
batch	0.914 ± 0.003	0.920 ± 0.005	**0.929** ± 0.002	0.920 ± 0.003	0.920 ± 0.002
coil2000	0.500 ± 0.000	0.500 ± 0.000	0.500 ± 0.000	0.500 ± 0.000	0.500 ± 0.000
dota	0.500 ± 0.000	0.504 ± 0.001	0.529 ± 0.003	0.500 ± 0.000	**0.547** ± 0.005
drive	0.698 ± 0.008	0.740 ± 0.011	**0.964** ± 0.004	0.680 ± 0.010	0.812 ± 0.006
facial	0.520 ± 0.009	0.699 ± 0.005	0.706 ± 0.003	0.701 ± 0.004	**0.718** ± 0.003
fashionmnist	0.901 ± 0.001	0.908 ± 0.004	0.915 ± 0.001	0.911 ± 0.004	**0.917** ± 0.002
gisette	0.605 ± 0.004	**0.755** ± 0.003	0.591 ± 0.002	0.627 ± 0.003	0.614 ± 0.003
hapt	0.839 ± 0.005	**0.855** ± 0.002	0.838 ± 0.002	0.850 ± 0.001	0.841 ± 0.003
image	0.703 ± 0.005	0.823 ± 0.004	0.884 ± 0.003	0.872 ± 0.003	**0.898** ± 0.001
isolet	0.827 ± 0.008	0.786 ± 0.005	0.850 ± 0.006	0.842 ± 0.007	**0.880** ± 0.006
letter	0.734 ± 0.003	0.701 ± 0.005	**0.749** ± 0.003	0.740 ± 0.002	0.742 ± 0.005
madelon	0.500 ± 0.000	0.556 ± 0.002	0.590 ± 0.001	**0.637** ± 0.004	0.631 ± 0.002
mfeat	0.500 ± 0.000	0.932 ± 0.003	0.938 ± 0.003	0.968 ± 0.001	**0.976** ± 0.000
microv1	0.678 ± 0.005	0.692 ± 0.006	0.713 ± 0.004	0.690 ± 0.005	**0.714** ± 0.003
microv2	0.708 ± 0.006	0.500 ± 0.000	**0.747** ± 0.002	0.725 ± 0.004	0.742 ± 0.003
mnist	0.872 ± 0.003	0.891 ± 0.004	0.896 ± 0.000	0.905 ± 0.002	**0.909** ± 0.001
musk	0.850 ± 0.003	0.953 ± 0.004	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000
nomao	0.901 ± 0.002	0.904 ± 0.002	**0.923** ± 0.003	0.530 ± 0.011	0.913 ± 0.003
semeion	0.827 ± 0.003	0.817 ± 0.008	**0.860** ± 0.005	0.756 ± 0.002	0.793 ± 0.006

Table 8
C4.5 results for plain data and autoencoder models (AUC).

Dataset	Plain Data	BAE	CAE	DAE	RAE
arcene	0.655 ± 0.005	0.679 ± 0.009	0.565 ± 0.006	0.678 ± 0.003	**0.730** ± 0.004
batch	0.921 ± 0.003	0.941 ± 0.002	0.911 ± 0.003	0.982 ± 0.000	**0.983** ± 0.001
coil2000	0.503 ± 0.002	0.509 ± 0.003	**0.521** ± 0.001	0.516 ± 0.002	0.513 ± 0.004
dota	0.462 ± 0.004	0.498 ± 0.006	**0.548** ± 0.003	0.539 ± 0.002	0.521 ± 0.005
drive	0.869 ± 0.003	0.862 ± 0.006	**0.944** ± 0.003	0.850 ± 0.004	0.867 ± 0.005
facial	**0.789** ± 0.011	0.629 ± 0.009	0.660 ± 0.005	0.655 ± 0.013	0.683 ± 0.008
fashionmnist	0.877 ± 0.002	0.875 ± 0.000	0.883 ± 0.003	0.888 ± 0.001	**0.890** ± 0.000
gisette	0.657 ± 0.005	0.758 ± 0.008	0.714 ± 0.003	0.743 ± 0.005	**0.790** ± 0.006
hapt	0.798 ± 0.001	0.791 ± 0.002	**0.800** ± 0.001	0.799 ± 0.000	0.799 ± 0.001
image	0.832 ± 0.002	0.753 ± 0.001	0.862 ± 0.003	0.859 ± 0.003	**0.927** ± 0.001
isolet	0.870 ± 0.004	0.853 ± 0.002	**0.888** ± 0.003	0.874 ± 0.003	0.867 ± 0.004
letter	**0.922** ± 0.005	0.742 ± 0.008	0.813 ± 0.012	0.811 ± 0.008	0.820 ± 0.004
madelon	0.520 ± 0.004	0.515 ± 0.002	0.557 ± 0.006	0.524 ± 0.003	**0.565** ± 0.002
mfeat	0.911 ± 0.003	0.922 ± 0.003	0.930 ± 0.001	0.913 ± 0.004	**0.941** ± 0.002
microv1	0.838 ± 0.004	0.851 ± 0.002	0.871 ± 0.002	0.849 ± 0.001	**0.937** ± 0.003
microv2	0.736 ± 0.005	0.746 ± 0.003	0.867 ± 0.006	0.845 ± 0.002	**0.873** ± 0.003
mnist	0.965 ± 0.005	0.958 ± 0.004	0.969 ± 0.002	0.975 ± 0.001	**0.979** ± 0.002
musk	0.800 ± 0.002	0.805 ± 0.000	0.880 ± 0.002	0.883 ± 0.001	**0.885** ± 0.000
nomao	**0.886** ± 0.002	0.877 ± 0.001	0.885 ± 0.001	0.877 ± 0.003	0.882 ± 0.002
semeion	0.716 ± 0.003	0.606 ± 0.005	0.727 ± 0.002	0.739 ± 0.005	**0.759** ± 0.001

It is important to note that in all cases the best results are obtained with the use of AEs.

5.5. C4.5

In this section, the classification results obtained for the C4.5 method are presented in Table 8. The C4.5 algorithm behaves similarly to the previous algorithms while using the new input space produced by AE models. On the one hand, the best results are obtained with the data generated by the AEs in most cases. Specifically, RAE generates the best performance in 12 out of 20 cases and CAE in 5 out of 20 datasets. On the other hand, in 3 cases the results obtained with the original data are the best. Unlike the previous algorithms, in this case the DAE model never obtains the best result. However, the performance of DAE is close to that obtained with RAE in many cases. This fact is best seen in Fig. 2(c).

The predictive performance of the C4.5 algorithm is improved when AEs are used. Higher-level feature generation positively affects tree-based algorithms. The methodology of the C4.5 algorithm is based on the analysis of the attributes that provide more information to separate the instances in classes [8]. In a high dimensionality space, the high number of features prevents this type of algorithms from selecting attributes that provide little information. For this reason, the use of dimensionality reduction methods is recommended.

The process followed in this study allows information to be merged from different features using AEs and generating new attributes by combining the originals. In this new scenario, the C4.5 algorithm has new features that contain more relevant information; therefore, the divisions performed are better and the predictive performance improves.

For C4.5, the results follow the line of the previous algorithms. The most sophisticated AE models generate the best predictive performance. These results justify the use of AEs to reduce dimensionality as a previous phase to that using tree-based algorithms, such as C4.5. The fusion of features allows for the generation of more relevant attributes that provide more useful information.

5.6. Results analysis

Once the results for each of the classification algorithms with the different AE models have been presented, the objective is to perform statistical tests that support the conclusions reached. This step is essential in order to confirm the results obtained and the differences between the different models.

Table 9
Average rankings of the different autoencoder models by classification method.

kNN		SVM		MLP		C4.5	
AE model	Ranking	AE model	Ranking	AE model	Ranking	AE model	Ranking
RAE	1.750	RAE	2.025	RAE	1.750	RAE	1.700
DAE	2.400	DAE	2.200	CAE	2.400	CAE	2.450
CAE	2.700	CAE	2.850	DAE	3.100	DAE	3.000
BAE	4.000	BAE	3.850	BAE	3.500	Plain	3.800
Plain	4.150	Plain	4.075	Plain	4.250	BAE	4.050

Table 10
Li post-hoc Friedman test for classification algorithm by autoencoder model.

		BAE	CAE	DAE	RAE	Plain
kNN	BAE	-	-	-	-	7.642E-01
	CAE	**3.803E-02**	-	-	-	**1.558E-02**
	DAE	**5.794E-03**	6.993E-01	-	-	**1.969E-03**
	RAE	**2.881E-05**	**1.958E-01**	4.508E-01	-	**6.728E-06**
	Plain	-	-	-	-	-
SVM	BAE	-	-	-	-	7.046E-01
	CAE	**1.426E-01**	-	-	-	**4.961E-02**
	DAE	**3.521E-03**	4.143E-01	-	-	**6.457E-04**
	RAE	**9.573E-04**	2.655E-01	7.263E-01	-	**1.509E-04**
	Plain	-	-	-	-	-
MLP	BAE	-	-	-	-	**1.882E-01**
	CAE	**4.603E-02**	-	**1.891E-01**	-	**3.739E-04**
	DAE	4.237E-01	-	-	-	**3.588E-02**
	RAE	**8.067E-04**	2.514E-01	**1.189E-02**	-	**9.948E-07**
	Plain	-	-	-	-	-
C4.5	BAE	-	-	-	-	-
	CAE	**3.576E-03**	-	4.147E-01	-	**1.779E-02**
	DAE	**8.534E-02**	-	-	-	2.225E-01
	RAE	**6.794E-06**	2.586E-01	**2.376E-02**	-	**6.969E-05**
	Plain	6.171E-01	-	-	-	-

To support the results obtained, it is necessary to check whether there are significant differences between the models. The fundamental objective is to check the differences between the most sophisticated models and the plain data, since the previous results determined that these cases were those that significantly improved predictive performance. To do this, first average ranks are obtained by applying the Friedman test [89]. The rankings for each classification algorithm are shown in Table 9.

Table 9 shows that for the four classification algorithms the best predictive performance is obtained with RAE. The DAE and CAE models are in second and third place, alternating their position according to the classification method. In addition, the worst results are obtained with the original data, except for C4.5, which is obtained with the BAE model. These rankings confirm the conclusions drawn in the previous subsections where it was indicated that the most sophisticated models offered better performance in most cases.

However, the previous rankings must be confirmed by statistical tests that verify that there are significant differences. This is necessary in order to be able to draw conclusions based on solid results. For this reason, the Li post-hoc [90] for Friedman test is carried out, comparing the different models with each other. Table 10 presents the different *p-values* obtained with the Li test for the four classification algorithms.

Table 10 shows that there are significant differences between most of the comparisons between the different models used in this experiment, setting the *p-value* threshold to the usual range [0.05, 0.1]. In general, the most sophisticated models present clear differences with respect to the BAE model and the model with plain data. The RAE, DAE and CAE models do not always have significant differences, since each model presents better results for some of the datasets. Similarly, there are no significant differences between the plain models and the most basic, since they are generally surpassed by more sophisticated models.

These results confirm that the more sophisticated AE models improve on the predictive performance obtained by the plain model using the four selected classification algorithms. There are no significant differences among the sophisticated models, although the rankings indicate that the RAE model works better.

5.7. Running time analysis

Once each of the classification algorithms has been analyzed in detail from a predictive performance perspective, an analysis of the execution time of the different configurations for each algorithm is carried out. The objective of this subsection is to verify the time improvements in the configurations as the size of the output space is reduced by AEs.

First, it is necessary to clarify the execution time studied in this Subsection. Hence, the following analysis only considers the time of the different classifiers. The training and compression time of the AEs models is not taken into account, since this process is performed only once for each model and architecture, generating a subset that can be used by any classifier in later stages. The reason for this approach is to highlight the performance improvement in terms of execution time obtained when classifying data of lower dimensionality. Thus the analysis is focused on studying the times grouped by the different architectures and classification algorithms.

Table 11 shows a series of rankings obtained after applying the Friedman test [89] for execution time. The data of all the datasets and AE models have been grouped, since the main interest in this section is to verify the time differences of each configuration. In addition, it is important to indicate that the times considered are averages for all the executions performed considering the 2x5 fold cross validation scheme. These averages have been used to develop the rankings presented.

As can be seen in Table 11, the best performance with respect to execution time is obtained with the configuration that reduces the number

Table 11
Average rankings considering execution time of the different autoencoder architecture by classification method.

kNN		SVM		MLP		C4.5	
Architecture	Ranking	Architecture	Ranking	Architecture	Ranking	Architecture	Ranking
ARQ_25	1.031	ARQ_25	1.025	ARQ_25	1.012	ARQ_25	1.037
ARQ_50	2.056	ARQ_50	2.075	ARQ_50	2.000	ARQ_50	2.068
ARQ_75	2.912	ARQ_75	3.031	ARQ_75	3.012	ARQ_75	2.981
Plain	4.000	Plain	3.868	Plain	3.975	Plain	3.912

Table 12
Li post-hoc Friedman test for running time results by autoencoder architecture.

		ARQ_25	ARQ_50	ARQ_75	Plain
kNN	ARQ_25	-	5.128E-07	0.000E + 00	0.000E + 00
	ARQ_50	-	-	2.732E-05	0.000E + 00
	ARQ_75	-	-	-	9.949E-08
	Plain	-	-	-	-
SVM	ARQ_25	-	2.691E-07	0.000E + 00	0.000E + 00
	ARQ_50	-	-	2.805E-06	0.000E + 00
	ARQ_75	-	-	-	4.080E-05
	Plain	-	-	-	-
MLP	ARQ_25	-	1.313E-06	0.000E + 00	0.000E + 00
	ARQ_50	-	-	7.041E-06	0.000E + 00
	ARQ_75	-	-	-	2.414E-06
	Plain	-	-	-	-
C4.5	ARQ_25	-	4.370E-07	0.000E + 00	0.000E + 00
	ARQ_50	-	-	7.810E-06	0.000E + 00
	ARQ_75	-	-	-	5.063E-06
	Plain	-	-	-	-

of features to 25% of the total for all the algorithms. In a similar manner, the execution time increases as compression decreases. In this way, the worst results are obtained with the plain model, that is the model that does not use AEs to reduce dimensionality.

Finally, it is necessary to determine whether there are statistically significant differences in order to establish solid and well-founded results. To this end, Li post-hoc tests [90] for the Friedman test are applied. Table 12 presents the different *p-values* generated using the Li test.

The results obtained from the Li test, shown in Table 12, show that there are significant differences between the configurations considered in this study, setting the *p-value* threshold to the usual range [0.05, 0.1]. These differences are given in all cases compared.

Therefore, the fact that the execution time decreases as the compression of the data increases is confirmed by solid results. In conclusion, the use of AEs to reduce dimensionality entails a considerable reduction in terms of execution time. Thus, the selection of the architecture may depend on the relevance of the execution time in the problem in question. If it is important to reduce the time, an architecture with fewer units in the hidden layer can be selected, although this implies lower predictive performance.

6. Autoencoders vs classical feature extraction techniques

Finally, the objective of this section is to assess the competitiveness of AE models as compared against traditional dimensionality reduction methods. Specifically, four of the most well-known and widely-used algorithms have been selected, PCA, LDA, ISOMAP and LLE. To do so, a comparison has been made between the most sophisticated models of AE (DAE, CAE and RAE) and the results obtained with PCA, LDA, ISOMAP and LLE on the same datasets. The configuration selected in all cases corresponds to that established in Section 4.2 and provides the best performance. It is important for the number of features obtained through the different methods to be the same in order to make a reliable comparison. Similarly, this contrast has been made considering the four classification algorithms used in the experimentation.

Tables 13–16 present the classification results of the algorithms kNN, SVM, MLP and C4.5, respectively. Each table contains the data obtained after applying the 7 methods of dimensionality reduction: CAE, DAE, RAE, PCA, LDA, ISOMAP and LLE. In each case, the best result is highlighted in bold.

Observing the results presented in Tables 13–16, it can be seen that the best results are obtained with the AE models for most of the cases. Specifically, for kNN there are only three datasets (*arcene, microv1, musk*) where the best result is obtained with traditional algorithms and another dataset (*mfeat*) where CAE and ISOMAP have the same result. The SVM and MLP algorithms show similar behavior, there are two datasets (*arcene, microv1*) where the best results are obtained with traditional algorithms and another dataset (*coil2000*) where all the results obtained are similar. Finally, considering the C4.5 algorithm, it can be observed that there are three datasets (*arcene, drive, madelon*) where the best predictive performance is obtained from the data generated by traditional dimensionality reduction algorithms. Therefore, a general trend in all classification algorithms can be observed that indicates that AE-based methods work better than LDA, PCA, ISOMAP and LLE.

However, it is necessary to confirm that these differences are statistically significant. To do so, the Friedman test is applied first [89]. The average rankings obtained are presented in Table 17. Then, a series of performance classifications of the different dimensionality reduction methods are generated.

The different rankings presented in 17 show that the best models are those based on AEs, while the traditional algorithms generate the worst results from an overall point of view. This trend is maintained in all the classification algorithms used in the experiment.

The second step when verifying the differences between the models is to verify whether there are significant differences or not. In order to do so, Li post-hoc tests [90] for the Friedman test are used to compare the different models. Table 18 presents the *p-values* obtained with the Li test.

Table 18 shows that there are significant differences between most AE models with traditional models in the cases considered, setting the *p-value* threshold to the usual range [0.05, 0.1]. However, there are some AE models that do not show significant differences with LDA, ISOMAP and LLE in some classification algorithms. Nevertheless, the RAE model shows significant differences with LDA, PCA, ISOMAP and LLE in the four classification algorithms considered. This confirms the trend that indicates that the RAE model produces the best results from an overall point of view.

In summary, the data presented in this section show that dimensionality reduction models based on AEs generate a predictive performance superior to traditional models such as LDA, PCA ISOMAP and LLE. This is mainly due to the fact that models based on AEs generate more relevant features and provide the classification algorithms with more useful information. This allows us to consider this type of model as an option to take into account when tackling this task.

7. General guidelines on the use of autoencoder models

In the previous sections an exhaustive experimentation on the performance of several classifiers with data reduced using AE models and the different conclusions reached in each case have been presented. This section aims to analyze in more detail the relationship between the characteristics of the datasets and the results obtained. As a result, a series of guidelines are presented that allow the most appropriate AE model to be selected according to the classification algorithm and the traits of the data specific to each problem. To this end, the most relevant recommendation for each algorithm is presented in the following list. In addition, Table 19 presents a summary of the main ideas.

KNN:

- For a dataset with a very high dimensionality, the best results are obtained with DAE and RAE models. Therefore, we recommend using them for datasets with more than 1000 features.
- The most recommended model when using datasets with a number of features between 500 and 1000 is RAE. However, CAE model also offers good results in certain cases.
- If datasets have between 100 and 500 features, RAE model generally performs well. Similarly, DAE and CAE models obtain good results with binary datasets.
- The models that generate the best predictive performance for datasets with less than 100 features are RAE and CAE.

SVM:

Table 13
kNN classification results of CAE, DAE, RAE, PCA, LDA, ISOMAP and LLE for test data (AUC).

Dataset	CAE	DAE	RAE	PCA	LDA	ISOMAP	LLE
arcene	0.605 ± 0.015	0.672 ± 0.005	0.743 ± 0.008	0.661 ± 0.006	0.654 ± 0.009	0.738 ± 0.003	**0.747** ± 0.005
batch	0.900 ± 0.010	0.990 ± 0.001	**0.992** ± 0.001	0.861 ± 0.005	0.852 ± 0.007	0.786 ± 0.004	0.862 ± 0.006
coil2000	**0.554** ± 0.004	**0.554** ± 0.004	0.545 ± 0.003	0.523 ± 0.006	0.525 ± 0.003	0.525 ± 0.003	0.502 ± 0.002
dota	0.519 ± 0.000	**0.522** ± 0.003	0.516 ± 0.002	0.513 ± 0.003	0.517 ± 0.001	0.516 ± 0.002	0.516 ± 0.002
drive	**0.877** ± 0.005	0.846 ± 0.006	0.873 ± 0.004	0.683 ± 0.011	0.782 ± 0.006	0.693 ± 0.003	0.726 ± 0.008
facial	0.690 ± 0.006	0.685 ± 0.003	**0.697** ± 0.002	0.631 ± 0.005	0.648 ± 0.004	0.601 ± 0.004	0.602 ± 0.003
fashionmnist	0.912 ± 0.001	0.920 ± 0.003	**0.922** ± 0.001	0.831 ± 0.004	0.911 ± 0.002	0.842 ± 0.003	0.896 ± 0.001
gisette	0.897 ± 0.004	**0.923** ± 0.002	0.900 ± 0.004	0.855 ± 0.003	0.883 ± 0.006	0.877 ± 0.003	0.878 ± 0.005
hapt	0.876 ± 0.005	0.931 ± 0.003	**0.939** ± 0.002	0.553 ± 0.012	0.903 ± 0.004	0.750 ± 0.003	0.781 ± 0.006
image	0.952 ± 0.002	0.950 ± 0.001	**0.958** ± 0.003	0.734 ± 0.011	0.779 ± 0.004	0.882 ± 0.007	0.874 ± 0.002
isolet	0.974 ± 0.001	0.968 ± 0.002	**0.976** ± 0.002	0.659 ± 0.006	0.971 ± 0.002	0.962 ± 0.003	0.927 ± 0.001
letter	0.925 ± 0.012	0.923 ± 0.006	**0.947** ± 0.005	0.882 ± 0.005	0.893 ± 0.003	0.894 ± 0.003	0.883 ± 0.004
madelon	0.566 ± 0.006	0.591 ± 0.003	**0.618** ± 0.005	0.523 ± 0.002	0.505 ± 0.005	0.506 ± 0.003	0.501 ± 0.001
mfeat	**0.986** ± 0.002	0.984 ± 0.001	0.984 ± 0.001	0.963 ± 0.003	0.972 ± 0.002	0.985 ± 0.001	**0.986** ± 0.002
microv1	0.922 ± 0.003	0.939 ± 0.005	0.929 ± 0.004	0.532 ± 0.006	0.897 ± 0.002	**0.947** ± 0.000	0.946 ± 0.001
microv2	**0.955** ± 0.002	0.932 ± 0.005	0.954 ± 0.003	0.632 ± 0.005	0.891 ± 0.003	0.948 ± 0.002	0.951 ± 0.002
mnist	0.969 ± 0.000	0.975 ± 0.001	**0.979** ± 0.000	0.828 ± 0.002	0.966 ± 0.003	0.923 ± 0.001	0.944 ± 0.002
musk	0.940 ± 0.002	0.941 ± 0.001	0.947 ± 0.002	0.912 ± 0.004	**0.964** ± 0.001	0.938 ± 0.003	0.901 ± 0.005
nomao	**0.914** ± 0.005	0.904 ± 0.002	0.905 ± 0.003	0.797 ± 0.004	0.817 ± 0.006	0.804 ± 0.003	0.849 ± 0.005
semeion	0.907 ± 0.004	0.940 ± 0.001	**0.942** ± 0.000	0.732 ± 0.003	0.926 ± 0.005	0.894 ± 0.005	0.884 ± 0.003

Table 14
SVM classification results of CAE, DAE, RAE, PCA, LDA, ISOMAP and LLE for test data (AUC).

Dataset	CAE	DAE	RAE	PCA	LDA	ISOMAP	LLE
arcene	0.500 ± 0.000	0.632 ± 0.005	0.500 ± 0.000	0.531 ± 0.003	0.545 ± 0.004	0.721 ± 0.002	**0.722** ± 0.003
batch	0.924 ± 0.004	0.980 ± 0.002	**0.985** ± 0.002	0.983 ± 0.001	0.980 ± 0.002	0.983 ± 0.001	0.892 ± 0.003
coil2000	**0.500** ± 0.000	**0.500** ± 0.000	**0.500** ± 0.000	0.493 ± 0.002	**0.500** ± 0.000	**0.500** ± 0.000	**0.500** ± 0.000
dota	**0.553** ± 0.001	0.550 ± 0.002	0.534 ± 0.005	0.517 ± 0.005	0.529 ± 0.003	0.547 ± 0.002	0.548 ± 0.000
drive	**0.985** ± 0.003	0.941 ± 0.002	0.863 ± 0.004	0.978 ± 0.001	0.979 ± 0.002	0.981 ± 0.002	0.982 ± 0.001
facial	0.701 ± 0.004	0.713 ± 0.002	**0.714** ± 0.005	0.672 ± 0.005	0.703 ± 0.003	0.697 ± 0.005	0.501 ± 0.007
fashionmnist	0.938 ± 0.003	0.940 ± 0.000	**0.942 ± 0.001**	0.921 ± 0.003	0.935 ± 0.001	0.929 ± 0.001	0.934 ± 0.002
gisette	0.816 ± 0.001	**0.969** ± 0.003	0.955 ± 0.006	0.925 ± 0.002	0.941 ± 0.004	0.962 ± 0.001	0.963 ± 0.003
hapt	0.871 ± 0.003	**0.889** ± 0.001	0.886 ± 0.004	0.788 ± 0.005	0.875 ± 0.003	0.859 ± 0.003	0.869 ± 0.001
image	0.886 ± 0.001	0.877 ± 0.005	**0.910** ± 0.002	0.850 ± 0.004	0.862 ± 0.001	0.784 ± 0.005	0.620 ± 0.003
isolet	0.971 ± 0.002	0.975 ± 0.001	**0.976** ± 0.001	0.958 ± 0.003	0.965 ± 0.001	0.963 ± 0.001	0.966 ± 0.003
letter	0.841 ± 0.004	0.828 ± 0.004	**0.879** ± 0.005	0.812 ± 0.003	0.839 ± 0.005	0.842 ± 0.004	0.852 ± 0.002
madelon	0.580 ± 0.002	**0.591** ± 0.001	0.587 ± 0.000	0.557 ± 0.003	0.585 ± 0.002	0.547 ± 0.004	0.523 ± 0.004
mfeat	**0.985** ± 0.002	**0.985** ± 0.001	**0.985** ± 0.001	0.926 ± 0.003	0.962 ± 0.003	0.980 ± 0.002	0.981 ± 0.000
microv1	0.796 ± 0.003	0.918 ± 0.005	0.786 ± 0.002	0.731 ± 0.005	0.876 ± 0.004	**0.953** ± 0.002	**0.953** ± 0.002
microv2	**0.916** ± 0.003	0.871 ± 0.001	0.881 ± 0.001	0.652 ± 0.010	0.791 ± 0.008	0.832 ± 0.009	0.835 ± 0.011
mnist	0.984 ± 0.001	0.985 ± 0.001	**0.987** ± 0.000	0.971 ± 0.002	0.979 ± 0.002	0.829 ± 0.005	0.976 ± 0.001
musk	0.939 ± 0.005	0.948 ± 0.006	**0.964** ± 0.003	0.955 ± 0.003	0.958 ± 0.001	0.959 ± 0.001	0.954 ± 0.002
nomao	**0.935** ± 0.000	0.924 ± 0.002	0.932 ± 0.001	0.929 ± 0.000	0.930 ± 0.001	0.933 ± 0.001	0.925 ± 0.003
semeion	0.957 ± 0.003	0.957 ± 0.001	**0.962** ± 0.001	0.936 ± 0.005	0.957 ± 0.002	0.929 ± 0.003	0.949 ± 0.003

Table 15
MLP classification results of CAE, DAE, RAE, PCA, LDA, ISOMAP and LLE for test data (AUC).

Dataset	CAE	DAE	RAE	PCA	LDA	ISOMAP	LLE
arcene	0.465 ± 0.005	0.500 ± 0.000	0.523 ± 0.001	0.521 ± 0.004	0.512 ± 0.003	**0.691** ± 0.006	0.673 ± 0.004
batch	**0.929** ± 0.002	0.920 ± 0.003	0.920 ± 0.002	0.915 ± 0.001	0.512 ± 0.003	0.919 ± 0.004	0.566 ± 0.001
coil2000	**0.500** ± 0.000	**0.500** ± 0.000	**0.500** ± 0.000	**0.500** ± 0.000	**0.500** ± 0.000	**0.500** ± 0.000	**0.500** ± 0.000
dota	0.529 ± 0.003	0.500 ± 0.000	**0.547** ± 0.005	0.524 ± 0.002	0.525 ± 0.004	0.527 ± 0.002	0.525 ± 0.001
drive	**0.964** ± 0.004	0.680 ± 0.010	0.812 ± 0.006	0.941 ± 0.003	0.883 ± 0.004	0.802 ± 0.005	0.809 ± 0.004
facial	0.706 ± 0.003	0.701 ± 0.004	**0.718** ± 0.003	0.692 ± 0.002	0.704 ± 0.002	0.707 ± 0.001	0.697 ± 0.004
fashionmnist	0.915 ± 0.001	0.911 ± 0.004	**0.917** ± 0.002	0.902 ± 0.003	0.909 ± 0.001	0.898 ± 0.004	0.905 ± 0.004
gisette	0.591 ± 0.002	**0.627** ± 0.003	0.614 ± 0.003	0.602 ± 0.001	0.612 ± 0.003	0.609 ± 0.004	0.605 ± 0.002
hapt	0.838 ± 0.002	**0.850** ± 0.001	0.841 ± 0.003	0.834 ± 0.005	0.839 ± 0.003	0.840 ± 0.004	0.820 ± 0.005
image	0.884 ± 0.003	0.872 ± 0.003	**0.898** ± 0.001	0.881 ± 0.003	0.552 ± 0.010	0.672 ± 0.008	0.645 ± 0.009
isolet	0.850 ± 0.006	0.842 ± 0.007	**0.880** ± 0.006	0.849 ± 0.005	0.819 ± 0.006	0.791 ± 0.009	0.805 ± 0.005
letter	**0.749** ± 0.003	0.740 ± 0.002	0.742 ± 0.005	0.713 ± 0.005	0.732 ± 0.002	0.739 ± 0.006	0.722 ± 0.002
madelon	0.590 ± 0.001	**0.637** ± 0.004	0.631 ± 0.002	0.577 ± 0.008	0.582 ± 0.005	0.556 ± 0.005	0.512 ± 0.007
mfeat	0.938 ± 0.003	0.968 ± 0.001	**0.976** ± 0.000	0.921 ± 0.003	0.953 ± 0.004	0.974 ± 0.001	0.964 ± 0.002
microv1	0.713 ± 0.004	0.690 ± 0.005	0.714 ± 0.003	0.694 ± 0.003	0.701 ± 0.005	0.829 ± 0.006	**0.868** ± 0.004
microv2	**0.747** ± 0.002	0.725 ± 0.004	0.742 ± 0.003	0.731 ± 0.002	0.720 ± 0.002	0.740 ± 0.001	0.733 ± 0.003
mnist	0.896 ± 0.000	0.905 ± 0.002	**0.909** ± 0.001	0.902 ± 0.002	0.906 ± 0.001	0.893 ± 0.000	0.896 ± 0.003
musk	**0.999** ± 0.000	**0.999** ± 0.000	**0.999** ± 0.000	0.992 ± 0.002	0.991 ± 0.002	0.994 ± 0.001	0.952 ± 0.001
nomao	**0.923** ± 0.003	0.530 ± 0.011	0.913 ± 0.003	0.913 ± 0.002	0.915 ± 0.003	0.911 ± 0.005	0.914 ± 0.003
semeion	**0.860** ± 0.005	0.756 ± 0.002	0.793 ± 0.006	0.820 ± 0.004	0.849 ± 0.002	0.746 ± 0.005	0.822 ± 0.003

Table 16
C4.5 classification results of CAE, DAE, RAE, PCA, LDA, ISOMAP and LLE for test data (AUC).

Dataset	CAE	DAE	RAE	PCA	LDA	ISOMAP	LLE
arcene	0.565 ± 0.006	0.678 ± 0.003	0.730 ± 0.004	0.654 ± 0.005	0.662 ± 0.004	**0.766** ± 0.003	0.695 ± 0.004
batch	0.911 ± 0.003	0.982 ± 0.000	**0.983** ± 0.001	0.971 ± 0.003	0.973 ± 0.002	0.980 ± 0.001	0.975 ± 0.002
coil2000	**0.521** ± 0.001	0.516 ± 0.002	0.513 ± 0.004	0.509 ± 0.002	0.504 ± 0.002	0.507 ± 0.001	0.500 ± 0.000
dota	**0.548** ± 0.003	0.539 ± 0.002	0.521 ± 0.005	0.527 ± 0.003	0.531 ± 0.005	0.503 ± 0.005	0.515 ± 0.003
drive	0.944 ± 0.003	0.850 ± 0.004	0.867 ± 0.005	0.936 ± 0.003	0.938 ± 0.003	**0.954** ± 0.001	0.947 ± 0.002
facial	0.660 ± 0.005	0.655 ± 0.013	**0.683** ± 0.008	0.626 ± 0.003	0.651 ± 0.005	0.635 ± 0.007	0.632 ± 0.004
fashionmnist	0.883 ± 0.003	0.888 ± 0.001	**0.890** ± 0.000	0.872 ± 0.002	0.883 ± 0.000	0.879 ± 0.001	0.881 ± 0.001
gisette	0.714 ± 0.003	0.743 ± 0.005	**0.790** ± 0.006	0.682 ± 0.003	0.705 ± 0.005	0.692 ± 0.003	0.709 ± 0.004
hapt	**0.800** ± 0.001	0.799 ± 0.000	0.799 ± 0.001	0.797 ± 0.002	0.791 ± 0.003	0.792 ± 0.002	0.788 ± 0.004
image	0.862 ± 0.003	0.859 ± 0.003	**0.927** ± 0.001	0.913 ± 0.002	0.922 ± 0.002	0.774 ± 0.004	0.838 ± 0.003
isolet	**0.888** ± 0.003	0.874 ± 0.003	0.867 ± 0.004	0.840 ± 0.004	0.859 ± 0.004	0.851 ± 0.005	0.862 ± 0.002
letter	0.813 ± 0.012	0.811 ± 0.008	**0.820** ± 0.004	0.783 ± 0.010	0.811 ± 0.004	0.793 ± 0.005	0.805 ± 0.003
madelon	0.557 ± 0.006	0.524 ± 0.003	0.565 ± 0.002	0.626 ± 0.007	**0.677** ± 0.003	0.652 ± 0.005	0.631 ± 0.005
mfeat	0.930 ± 0.001	0.913 ± 0.004	**0.941** ± 0.002	0.889 ± 0.002	0.898 ± 0.003	0.892 ± 0.001	0.895 ± 0.002
microv1	0.871 ± 0.002	0.849 ± 0.001	**0.937** ± 0.003	0.731 ± 0.010	0.806 ± 0.003	0.685 ± 0.005	0.868 ± 0.002
microv2	0.867 ± 0.006	0.845 ± 0.002	**0.873** ± 0.003	0.835 ± 0.005	0.837 ± 0.005	0.815 ± 0.006	0.838 ± 0.003
mnist	0.969 ± 0.002	0.975 ± 0.001	**0.979** ± 0.002	0.891 ± 0.003	0.927 ± 0.002	0.897 ± 0.003	0.915 ± 0.001
musk	0.880 ± 0.002	0.883 ± 0.001	**0.885** ± 0.000	0.869 ± 0.004	0.872 ± 0.002	0.876 ± 0.004	0.871 ± 0.003
nomao	**0.885** ± 0.001	0.877 ± 0.003	0.882 ± 0.002	0.881 ± 0.001	0.862 ± 0.000	0.767 ± 0.003	0.872 ± 0.003
semeion	0.727 ± 0.002	0.739 ± 0.005	**0.759** ± 0.001	0.729 ± 0.000	0.739 ± 0.003	0.744 ± 0.003	0.731 ± 0.002

Table 17
Average rankings considering CAE, DAE, RAE, PCA, LDA, ISOMAP and LLE by classification method.

kNN		SVM		MLP		C4.5	
Architecture	Ranking	Architecture	Ranking	Architecture	Ranking	Architecture	Ranking
RAE	1.925	RAE	2.600	RAE	2.250	RAE	2.025
DAE	2.700	DAE	3.150	CAE	3.025	CAE	3.075
CAE	2.750	CAE	3.550	DAE	4.125	DAE	3.275
LDA	4.475	LDA	4.200	ISOMAP	4.350	LDA	4.275
LLE	4.775	ISOMAP	4.275	LDA	4.425	LLE	4.750
ISOMAP	4.925	LLE	4.300	LLE	4.800	ISOMAP	4.950
PCA	6.450	PCA	5.925	PCA	5.025	PCA	5.650

Table 18
Li post-hoc Friedman test for dimensionality reduction methods by classification algorithm.

		CAE	DAE	RAE	PCA	LDA	ISOMAP	LLE
kNN	CAE	-	-	-	**4.260E-07**	**1.862E-02**	**3.810E-03**	**6.360E-03**
	DAE	9.417E-01	-	-	**4.234E-07**	**1.634E-02**	**3.373E-03**	**5.558E-03**
	RAE	2.870E-01	3.067E-01	-	**7.346E-10**	**6.626E-04**	**5.908E-05**	**1.268E-04**
	PCA	-	-	-		-	-	-
	LDA	-	-	-	**7.316E-03**	-	5.650E-01	6.970E-01
	ISOMAP	-	-	-	**3.564E-02**	-	-	-
	LLE	-	-	-	**2.124E-02**	-	8.408E-01	
SVM	CAE	-	-	-	**3.549E-03**	4.219E-01	3.792E-01	3.792E-01
	DAE	6.144E-01	-	-	**5.103E-04**	1.972E-01	**1.840E-01**	**1.840E-01**
	RAE	2.517E-01	4.906E-01	-	**2.376E-05**	**5.924E-02**	**5.924E-02**	**5.924E-02**
	PCA	-	-	-	-	-	-	-
	LDA	-	-	-	**5.924E-02**	-	9.226E-01	9.072E-01
	ISOMAP	-	-	-	**5.924E-02**	-	-	9.708E-01
	LLE	-	-	-	**5.924E-02**	-	-	-
MLP	CAE	-	2.122E-01	-	**1.426E-02**	**1.027E-01**	**1.181E-01**	**2.784E-02**
	DAE	-	-	-	3.276E-01	7.165E-01	7.762E-01	4.676E-01
	RAE	4.048E-01	**2.104E-02**	-	**1.020E-03**	**1.013E-02**	**1.104E-02**	**1.986E-03**
	PCA	-	-	-	-	-	-	-
	LDA	-	-	-	4.876E-01	-	-	6.606E-01
	ISOMAP	-	-	-	4.676E-01	9.126E-01	-	6.080E-01
	LLE	-	-	-	7.762E-01	-	-	-
C4.5	CAE	-	7.860E-01	-	**8.588E-04**	**1.245E-01**	**1.806E-02**	**3.687E-02**
	DAE	-	-	-	**2.131E-03**	**1.946E-01**	**3.687E-02**	**6.366E-02**
	RAE	**1.805E-01**	**1.147E-01**	-	**2.347E-06**	**3.457E-03**	**1.947E-04**	**4.644E-04**
	PCA	-	-	-	-	-	-	-
	LDA	-	-	-	**8.257E-02**	-	3.657E-01	5.217E-01
	ISOMAP	-	-	-	**3.626E-01**	-	-	-
	LLE	-	-	-	2.388E-01	-	7.860E-01	

Table 19
Autoencoder recommended considering number of feature and classifier.

Number of features	Classifiers			
	kNN	SVM	MLP	C4.5
> 1000	RAE DAE	DAE	RAE CAE (large number of classes)	RAE
> 500 - < 1000	RAE CAE	RAE (non-binary) DAE (binary)	RAE (non-binary) DAE (binary)	RAE CAE (more 10 classes)
> 100 - < 500	RAE CAE-DAE (binary)	RAE CAE (binary)	CAE RAE (binary)	RAE CAE (binary)
< 100	RAE CAE	RAE CAE	RAE CAE	RAE CAE

- When working with a dataset with very high dimensionality we recommend using DAE model. This model has obtained the best results for datasets with more than 1000 features.
- The model that works best when starting from a binary dataset with a number of features between 500 and 1000 is DAE. However, if it is a non-binary dataset the best model is RAE.
- RAE model offers good results for datasets with between 100 and 500 features. Similarly, CAE model demonstrates good predictive performance with binary datasets.
- CAE and RAE are the most recommended models if datasets with less than 100 features are used.

MLP:

- The model with the best performance is RAE when working with a dataset with a number greater than 1000 features. However, CAE model offers good results when working with datasets with a high number of classes.
- The best results are obtained with RAE model for a non-binary dataset with a number of features between 500 and 1000. However, the best model is DAE when using a binary dataset.
- The model that generates the best predictive performance when datasets have between 100 and 500 features is CAE. However, the best model when using several binary datasets is RAE.
- We recommend using CAE and RAE models when using datasets with less than 100 features.

C4.5:

- Based on this study, RAE is recommended for a dataset with a very high dimensionality (with more than 1000 features).
- When working with datasets that has between 500 and 1000 characteristics, it is advisable to use RAE and CAE models when the number of classes is greater than 10. However, RAE is the best model when using a binary dataset.
- The model that generates the best predictive performance is RAE if the number of features of the datasets is between 100 and 500. Similarly, CAE model works well with some binary datasets.
- According to the generated results, we recommend using CAE and RAE models when using datasets with less than 100 features.

Time:

- The results presented in this study indicate that, in general, the highest predictive performance is obtained with configurations that compress the data less. However, sometimes it is important to reduce execution times. In these cases a configuration that compresses more data may be more useful, despite a loss in predictive performance.

In summary, the AE model used to carry out the reduction of dimensionality must be adapted to the data traits, as well as to the methodology of the classification algorithm that is used. Therefore, the above advice has been presented according to the experience provided by this study.

8. Concluding remarks

In this paper an exhaustive study has been carried out on the performance of AEs when tackling the task of dimensionality reduction. This task is one of the challenges faced by machine learning, due to the effects of high dimensionality of the data in many current processes. Specifically, this study focuses on the task of classification. In this context there are several different classification methodologies. Therefore, experimentation is necessary to incorporate algorithms that cover several of the most widely used methodologies in order to provide a general vision of the performance of AEs. For this reason the most well known algorithms with the best results from the most important methodologies have been selected, specifically kNN, SVM, MLP and C4.5.

In order to undertake the task of dimensionality reduction, AEs have been used. The main reason is the good results that they have obtained when generating high level features, which is fundamental to improving the subsequent predictive performance. In this case, in order to give a broader view of the performance of AEs four different models were used, starting from the most basic model and generating three more sophisticated models, which give a global vision of the purchase of AEs.

To assess the performance of the different AE models a thorough experimentation on 20 datasets was designed combining the four AE models with the four classification algorithms. New feature spaces are generated for each dataset using the different AE models, which are used as input for the different classification algorithms. In particular, in the first part of the experimentation we determined which AE architecture generated the best performance. In the second part, the results generated by the best architecture have been compared with the results obtained from the original data.

The results obtained after performing the experiments show that predictive performance improves when using AE models to reduce the dimensionality of the input space. In particular, the most sophisticated AE models significantly improve the results obtained with the most basic model and with the original data. In a similar way, the execution time is significantly improved when using the different AE models by reducing the input space and the computational load of the different classification algorithms. The conclusions reached have been supported by a series of statistical tests that confirm the existence of significant differences between the different models.

In addition, AE models have been compared with traditional dimensionality reduction methods, with the aim of verifying their performance in relation to other models that have previously performed well in this task. For this reason, AE models have been compared with LDA, PCA, ISOMAP and LLE. The conclusions obtained indicate that sophisticated models based on AEs have a better predictive performance than the traditional models analyzed. This may indicate that the features generated by methods based on AEs are more significant and provide more relevant information to the classification algorithms.

Finally, the study leads to a series of guidelines with the aim of facilitating the choice of the most appropriate AE models according to the classification algorithm and the characteristics of the input data. It is

important to consider these factors in order to maximize the predictive performance of the model used.

Acknowledgments

This study by F. Pulgar was supported by the Spanish Ministry of Education under the FPU National Program (Ref. FPU16/00324). This work was partially supported by the Spanish Ministry of Economy and Competitiveness under project TIN2015-68454-R.

References

[1] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, Wiley, New York, 1973.
[2] R. Kohavi, F. Provost, Glossary of terms, Mach. Learn. 30 (2–3) (1998) 271–274, doi:10.1023/A:1017181826899.
[3] D.E. Goldberg, J.H. Holland, Genetic algorithms and machine learning, Mach. Learn. 3 (2) (1988) 95–99, doi:10.1023/A:1022602019183.
[4] S.B. Kotsiantis, Supervised machine learning: a review of classification techniques, Informatica 31 (2007) 249–268.
[5] D.W. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1) (1991) 37–66, doi:10.1023/A:1022689900470.
[6] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, B. Scholkopf, Support vector machines, IEEE Intell. Syst. Appl. 13 (4) (1998) 18–28.
[7] R.J. Schalkoff, Artificial Neural Networks, vol. 1, McGraw-Hill, New York, 1997.
[8] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.
[9] R. Bellman, Dynamic Programming, Princeton University Press, 1957.
[10] R. Bellman, Adaptive Control Processes: A Guided Tour, Princeton University Press, 1961.
[11] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, F. Herrera, A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches, IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) 42 (4) (2012) 463–484.
[12] G.E. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD Explor. Newsletter 6 (1) (2004) 20–29.
[13] M. Dash, H. Liu, Feature selection for classification, Intell. Data Anal. 1 (3) (1997) 131–156, doi:10.3233/IDA-1997-1302.
[14] H. Yu, J. Yang, A direct lda algorithm for high-dimensional data-with application to face recognition, Pattern Recogn. 34 (10) (2001) 2067–2070, doi:10.1016/S0031-3203(00)00162-X.
[15] K. Pearson, LIII. On lines and planes of closest fit to systems of points in space, Lond. Edinb. Dublin Philos.Mag. J. Sci. 2 (11) (1901) 559–572, doi:10.1080/14786440109462720.
[16] J.B. Tenenbaum, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323, doi:10.1126/science.290.5500.2319.
[17] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326, doi:10.1126/science.290.5500.2323.
[18] P. Domingos, A few useful things to know about machine learning, Commun. ACM 55 (10) (2012) 78–87, doi:10.1145/2347736.2347755.
[19] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, Pattern Anal. Mach. Intell. IEEE Trans. 35 (8) (2013) 1798–1828, doi:10.1109/TPAMI.2013.50.
[20] S. Garca, J. Luengo, F. Herrera, Data Preprocessing in Data Mining, Springer, 2015, doi:10.1007/978-3-319-10247-4.
[21] I. Guyon, A. Elisseeff, An Introduction to Feature Extraction, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 1–25, doi:10.1007/978-3-540-35488-8_1.
[22] U.G. Mangai, S. Samanta, S. Das, P.R. Chowdhury, A survey of decision fusion and feature fusion strategies for pattern classification, IETE Tech. Rev. 27 (4) (2010) 293–307, doi:10.4103/0256-4602.64604.
[23] G. Lin, H. Zhu, X. Kang, C. Fan, E. Zhang, Feature structure fusion and its application, Inf. Fusion 20 (2014) 146–154, doi:10.1016/j.inffus.2014.01.002.
[24] Y. Du, W. Song, Q. He, D. Huang, A. Liotta, C. Su, Deep learning with multi-scale feature fusion in remote sensing for automatic oceanic eddy detection, Inf. Fusion 49 (2019) 89–99, doi:10.1016/j.inffus.2018.09.006.
[25] Q. Zhang, L.T. Yang, Z. Chen, P. Li, A survey on deep learning for big data, Inf. Fusion 42 (2018) 146–157, doi:10.1016/j.inffus.2017.10.006.
[26] L. Deng, Deep learning: methods and applications, Found. Trend. Signal Process. 7 (3–4) (2014) 197–387, doi:10.1561/2000000039.
[27] Y. Bengio, Deep learning of representations: looking forward, in: International Conference on Statistical Language and Speech Processing, 2013, pp. 1–37, doi:10.1007/978-3-642-39593-2_1.
[28] D. Charte, F. Charte, S. Garca, M.J. del Jesus, F. Herrera, A practical tutorial on autoencoders for nonlinear feature fusion: taxonomy, models, software and guidelines, Inf. Fusion 44 (2018) 78–96, doi:10.1016/j.inffus.2017.12.007.
[29] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507, doi:10.1126/science.1127647.
[30] F.J. Pulgar, F. Charte, A.J. Rivera, M.J.D. Jesus, Aeknn: an autoencoder knn-based classifier with built-in dimensionality reduction, Int. J. Comput. Intell. Syst. 12 (11) (2018) 436–452, doi:10.2991/ijcis.2018.125905686.
[31] H.F. Nweke, Y.W. Teh, G. Mujtaba, M.A. Al-garadi, Data fusion and multiple classifier systems for human activity detection and health monitoring: review and open research directions, Inf. Fusion 46 (2019) 147–170, doi:10.1016/j.inffus.2018.06.002.
[32] Z. Chen, W. Li, Multisensor feature fusion for bearing fault diagnosis using sparse autoencoder and deep belief network, IEEE Trans. Instrument. Measur. 66 (7) (2017) 1693–1702, doi:10.1109/tim.2017.2669947.
[33] V. Singh, N.K. Verma, Z.U. Islam, Y. Cui, Feature learning using stacked autoencoder for shared and multimodal fusion of medical images, in: Computational Intelligence: Theories, Applications and Future Directions - Volume I, Springer Singapore, Singapore, 2019, pp. 53–66, doi:10.1007/978-981-13-1132-1_5.
[34] S. Maurya, V. Singh, S. Dixit, N.K. Verma, A. Salour, J. Liu, Fusion of low-level features with stacked autoencoder for condition based monitoring of machines, in: IEEE International Conference on Prognostics and Health Management (ICPHM), 2018, pp. 1–8, doi:10.1109/ICPHM.2018.8448969.
[35] J. López, A.S. Garea, D.B. Heras, F. Argüello, Stacked autoencoders for multiclass change detection in hyperspectral images, in: IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2018, pp. 1906–1909, doi:10.1109/IGARSS.2018.8518338.
[36] H. Bourlard, Y. Kamp, Auto-association by multilayer perceptrons and singular value decomposition, Biol. Cybern. 59 (4–5) (1988) 291–294, doi:10.1007/BF00332918.
[37] P. Vincent, H. Larochelle, Y. Bengio, P.A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th International Conference on Machine Learning, ACM, 2008, pp. 1096–1103, doi:10.1145/1390156.1390294.
[38] S. Rifai, Y. Bengio, Y. Dauphin, P. Vincent, A generative process for sampling contractive auto-encoders, in: Proceedings of the 29th International Conference on Machine Learning, ICML, 2012. Vol. 2, 2012
[39] Y. Qi, Y. Wang, X. Zheng, Z. Wu, Robust feature learning by stacked autoencoder with maximum correntropy criterion, in: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 6716–6720, doi:10.1109/ICASSP.2014.6854900.
[40] D. Heckerman, D. Geiger, D.M. Chickering, Learning bayesian networks: the combination of knowledge and statistical data, Mach. Learn. 20 (3) (1995) 197–243, doi:10.1007/BF00994016.
[41] L.A. Zadeh, Outline of a new approach to the analysis of complex systems and decision processes, IEEE Trans. Syst. Man Cybern. SMC- 3 (1) (1973) 28–44, doi:10.1109/TSMC.1973.5408575.
[42] L. Davis, Handbook of genetic algorithms, CUMINCAD (1991).
[43] S. Muggleton, Inductive logic programming, New Generat. Comput. 8 (4) (1991) 295–318, doi:10.1007/BF03037089.
[44] N.S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, Am. Stat. 46 (3) (1992) 175–185, doi:10.1080/00031305.1992.10475879.
[45] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27, doi:10.1109/TIT.1967.1053964.
[46] C.G. Atkeson, A.W. Moorey, S. Schaalz, A.W. Moore, S. Schaal, Locally weighted learning, Artif. Intell. 11 (1997) 11–73, doi:10.1023/A:1006559212014.
[47] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[48] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533, doi:10.1038/323533a0.
[49] M.W. Gardner, S. Dorling, Artificial neural networks (the multilayer perceptron)a review of applications in the atmospheric sciences, Atmos. Environ. 32 (14–15) (1998) 2627–2636.
[50] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, ACM, 1992, pp. 144–152.
[51] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd edition, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011. ISBN 0123748569.
[52] G.F. Hughes, On the mean accuracy of statistical pattern recognizers, IEEE Trans. Inf. Theory 14 (1) (1968) 55–63, doi:10.1109/TIT.1968.1054102.
[53] W. Liu, X. Yang, D. Tao, J. Cheng, Y. Tang, Multiview dimension reduction via hessian multiset canonical correlations, Inf. Fusion 41 (2018) 119–128, doi:10.1016/j.inffus.2017.09.001.
[54] R.A. Fisher, The statistical utilization of multiple measurements, Annal. Eugen. 8 (4) (1938) 376–386, doi:10.1111/j.1469-1809.1938.tb02189.x.
[55] H. Hotelling, Analysis of a complex of statistical variables into principal components, J. Educ. Psychol. 24 (6) (1933) 417–441, doi:10.1037/h0071325.
[56] W. Wang, Y. Huang, Y. Wang, L. Wang, Generalized autoencoder: a neural network framework for dimensionality reduction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2014, pp. 496–503, doi:10.1109/CVPRW.2014.79.
[57] H. Liu, H. Motoda, Feature extraction, construction and selection, A Data Mining Perspective, Vol. 453, Springer Science & Business Media, 1998, doi:10.1007/978-1-4615-5725-8.
[58] L. Cayton, Algorithms for manifold learning, University of California at San Diego Tech. Rep2005, 12, 1–17, 1.
[59] J.A. Lee, M. Verleysen, Nonlinear Dimensionality Reduction, Springer Science & Business Media, 2007.
[60] H. Schwenk, Y. Bengio, Training methods for adaptive boosting of neural networks, in: Advances in Neural Information Processing Systems, 1998, pp. 647–653, doi:10.1162/089976600300015178.
[61] M.A. Kramer, Nonlinear principal component analysis using autoassociative neural networks, AIChE J. 37 (2) (1991) 233–243, doi:10.1002/aic.690370209.
[62] R. Hecht-Nielsen, Replicator neural networks for universal optimal source coding, Science 269 (5232) (1995) 1860–1863, doi:10.1126/science.269.5232.1860.
[63] S. Rifai, X. Muller, Contractive auto-encoders: explicit invariance during feature extraction, in: Proceedings of the 28th International Conference on Machine Learning, Vol. 85, 2011, pp. 833–840.
[64] Y. Bengio, Learning deep architectures for AI, Found. Trend. Mach. Learn. 2 (1) (2009) 1–127, doi:10.1561/2200000006.
[65] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT Press, 2016.

[66] J. Deng, Z. Zhang, E. Marchi, B. Schuller, Sparse autoencoder-based feature transfer learning for speech emotion recognition, in: 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction (ACII), Vol. 00, 2014, pp. 511–516, doi:10.1109/ACII.2013.90.
[67] B.A. Olshausen, D.J. Field, Sparse coding with an overcomplete basis set: a strategy employed by v1? Vision Res. 37 (23) (1997) 3311–3325, doi:10.1016/S0042-6989(97)00169-7.
[68] F. Feng, X. Wang, R. Li, Cross-modal retrieval with correspondence autoencoder, in: Proceedings of the 22nd ACM International Conference on Multimedia, MM '14, ACM, New York, NY, USA, 2014, pp. 7–16, doi:10.1145/2647868.2654902.
[69] Y.J. Fan, Autoencoder node saliency: selecting relevant latent representations, Pattern Recognition 88 (2019) 643–653, doi:10.1016/j.patcog.2018.12.015.
[70] Y. Yang, Q.M.J. Wu, Y. Wang, Autoencoder with invertible functions for dimension reduction and image reconstruction, IEEE Trans. Syst. Man Cybern. 48 (7) (2018) 1065–1079, doi:10.1109/TSMC.2016.2637279.
[71] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Statist. 22 (3) (1951) 400–407, doi:10.1214/aoms/1177729586.
[72] T. Tieleman, G. Hinton, Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude, COURSERA: Neural Netw. Mach. Learn. 4 (2) (2012) 26–31.
[73] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (2011) 2121–2159.
[74] A. Cauchy, Méthode générale pour la résolution des systemes déquations simultanées, Comp. Rend. Sci. Paris 25 (1847) (1847) 536–538.
[75] L. Yann, Modeles connexionnistes de lapprentissage, ph.d. thesis, These de Doctorat, volume 6, Universite Paris, 1987.
[76] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.A. Manzagol, Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion, J. Mach, Learn. Res. 11 (2010) 3371–3408. Dec.
[77] W. Liu, P.P. Pokharel, J.C. Principe, Correntropy: a localized similarity measure, in: The 2006 IEEE International Joint Conference on Neural Network Proceedings, 2006, pp. 4919–4924, doi:10.1109/IJCNN.2006.247192.
[78] I. Guyon, S. Gunn, A. Ben-Hur, G. Dror, Result analysis of the nips 2003 feature selection challenge, in: Advances in neural information processing systems, 2005, pp. 545–552.
[79] A. Vergara, S. Vembu, T. Ayhan, M.A. Ryan, M.L. Homer, R. Huerta, Chemical gas sensor drift compensation using classifier ensembles, Sensor. Actuat. 166 (2012) 320–329, doi:10.1016/j.snb.2012.01.074.
[80] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, Keel: a software tool to assess evolutionary algorithms for data mining problems, Soft Comput. 13 (3) (2009) 307–318, doi:10.1007/s00500-008-0323-y.
[81] D. Dua, C. Graff, UCI machine learning repository, 2019, http://archive.ics.uci.edu/ml.
[82] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017, http://arxiv.org/abs/cs.LG/1708.07747.
[83] J.L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, D. Anguita, Transition-aware human activity recognition using smartphones, Neurocomputing (171) (2016) 754–767, doi:10.1016/j.neucom.2015.07.085.
[84] R. Cole, M. Fanty, Spoken letter recognition, in: Proceedings of the Workshop on Speech and Natural Language, 1990, pp. 385–390, doi:10.3115/116580.116725.
[85] I. Guyon, S. Gunn, A. Ben-Hur, G. Dror, Result analysis of the NIPS 2003 feature selection challenge, in: Proceedings of Neural Information Processing Systems, Vol. 4, 2004, pp. 545–552.
[86] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, in: Proceedings of the IEEE, volume 86, 1998, pp. 2278–2324, doi:10.1109/5.726791.
[87] X. Robin, N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.C. Sanchez, M. Müller, Proc: an open-source package for r and s+ to analyze and compare roc curves, BMC Bioinf. 12 (1) (2011) 77, doi:10.1186/1471-2105-12-77.
[88] D.J. Hand, R.J. Till, A simple generalisation of the area under the roc curve for multiple class classification problems, Mach. Learn. 45 (2) (2001) 171–186, doi:10.1023/A:1010920819831.
[89] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Stat. Assoc. 32 (200) (1937) 675–701, doi:10.1080/01621459.1937.10503522.
[90] J.D. Li, A two-step rejection procedure for testing multiple hypotheses, J. Stat. Plan. Inference 138 (6) (2008) 1521–1527, doi:10.1016/j.jspi.2007.04.032.
[91] S. Garca, A. Fernndez, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (10) (2010) 2044–2064. Special Issue on Intelligent Distributed Information Systems. doi:10.1016/j.ins.2009.12.010.
[92] F. Chollet, et al., Keras, 2015, https://keras.io.
[93] R.C. Team, R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria, 2016. https://www.R-project.org/.
[94] K. Schliep, K. Hechenbichler, kknn: weighted k-nearest neighbors, r package version 1.3.1, 2016, https://CRAN.R-project.org/package=kknn.
[95] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, F. Leisch, e1071: misc functions of the department of statistics, Probability Theory Group (Formerly: E1071), TU Wien, r Package Version 1.6–7, 2015. https://CRAN.R-project.org/package=e1071.
[96] M.K. C., J. Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, R.C. Team, M. Benesty, R. Lescarbeau, A. Ziem, L. Scrucca, Y. Tang, C. Candan, caret: classification and regression training, r package version 6.0–68, 2016, https://CRAN.R-project.org/package=caret.
[97] C. Bergmeir, J.M. Benítez, Neural networks in r using the stuttgart neural network simulator: RSNNS, J. Stat. Softw. 46 (7) (2012) 1–26.
[98] K. Hornik, C. Buchta, A. Zeileis, Open-source machine learning: R meets weka, Comput. Stat. 24 (2) (2009) 225–232, doi:10.1007/s00180-008-0119-7.
[99] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is "nearest neighbor" meaningful? in: International Conference on Database Theory, 1999, pp. 217–235, doi:10.1007/3-540-49257-7_15.

A.3. *Ensembles* de arquitecturas profundas

El artículo relacionado con el trabajo descrito en el Capítulo 6 es el siguiente:

- **Título:** ClEnDAE: A classifier based on ensembles with built-in dimensionality reduction through denoising autoencoders.

  Autores: Francisco J. Pulgar, Francisco Charte, Antonio J. Rivera, María J. del Jesus.

  Revista: Information Sciences.

  ISSN: 0020-0255.

  Estado: En revisión.

ClEnDAE: A classifier based on ensembles with built-in dimensionality reduction through denoising autoencoders

Francisco J. Pulgar*, Francisco Charte, Antonio J. Rivera, María J. del Jesus

Andalusian Research Institute on Data Science and Computational Intelligence (DaSCI), Computer Science Dpt., University of Jaén, 23071 - Jaén, Spain

Abstract

High dimensionality is a problem that affects most classification algorithms. This factor implies that the predictive performance of many traditional classifiers decreases considerably as the number of features increases. Therefore, there are numerous proposals that try to mitigate the effects of this problem.

In this study, ClEnDAE, a new classifier based on ensembles whose components incorporate denoising autoencoders (DAEs) to reduce the dimensionality of the input space, is proposed. On the one hand, the use of ensembles allows to improve the predictive performance by using several components working together. On the other hand, the use of DAEs allows to generate a new higher level and smaller sized feature space reducing the effects of high dimensionality. Finally, an experimentation is conducted with the goal of evaluating the behavior of ClEnDAE. Its first part is to compare the performance of ClEnDAE to a model based on basic DAE and to the original untreated data. The second part analyzes the results of ClEnDAE and other traditional methods of dimensionality reduction in order to determine the improvement obtained with the proposed algorithm. In both parts of the experimentation, the conclusions show that ClEnDAE offers a better predictive performance than the other analyzed models.

Keywords: classification, deep learning, denoising autoencoders, dimensionality reduction, ensembles, feature fusion

1. Introduction

Classification is one of the best known and most studied tasks in machine learning [30]. Basically, a classifier analyzes a series of training instances to extract relevant information that allows it to predict properties of new patterns. Over the years, a large number of proposals to deal with this problem have emerged. These methods are based on different methodologies associated with their structure and internal functioning [52].

The extended use of classification methods applied to very diverse data causes new problems and challenges associated with the properties of these data. One of the issues that affects most of the classification systems is the high dimensionality of the aforementioned data [83]. This factor has increased considerably in recent years due to continuous evolution of the information collection system. At present, there are a large number of tools that allow generating, storing and managing huge amounts of information and, therefore, the task of analyzing them becomes more relevant. In this context, the performance of a significant amount of traditional classification algorithms is negatively affected by the high dimensionality of the data [85, 53, 66]. The main reason for this behavior lies on the curse of dimensionality [11]. Fundamentally, this phenomenon is related to the decrease in performance of traditional classifiers as the dimension of the input data increases [9, 10]. Therefore, it is essential to develop new models or adapt traditional algorithms to deal with this problem.

On this matter, there are proposals that address data dimensionality reduction. Essentially, the goal is to reduce dimensionality while preserving most of relevant information. In this way, classifier performance improves when working with fewer input features [35]. Some of the most well-known traditional dimensionality reduction methods are: Principal Component Analysis (PCA) [63], Linear Discriminant Analysis (LDA) [93], Isometric feature mapping (ISOMAP) [81] and Locally Linear Embedding (LLE) [78], among others [36].

In recent years, a new methodology, known as Feature Fusion, has emerged [58, 20, 28]. The constant growth in the dimensionality of the data and the need to work with multimedia data have influenced the appearance of this new concept [59, 58]. The main objective of Feature Fusion is to generate new features based on the information provided by the whole set of original characteristics. In this manner, the most relevant input information is used and redundant or irrelevant data is discarded. Some of the most popular methods that apply Feature Fusion are based on Deep Learning (DL) [40]. For example, Convolutional Neural Networks (CNNs) [55] are able to identify complex patterns in images, and Autoencoders (AEs) [47] concentrate all the information needed to reconstruct the patterns in a few high-level features. These types of algorithms have proven to offer good results in many fields of application, which implies that they have significantly increased their use [27, 12].

In relation to dimensionality reduction, AEs are the most

*Corresponding author.

Email addresses: fpulgar@ujaen.es (Francisco J. Pulgar*), fcharte@ujaen.es (Francisco Charte), arivera@ujaen.es (Antonio J. Rivera), mjjesus@ujaen.es (María J. del Jesus)

used models due to their operation [47]. In fact, there are several studies that show the improvement of predictive performance and execution time when using AEs to reduce input data dimensionality [20, 67, 65]. The AEs rely on nonlinear relationships of the input data to obtain a new representation of it. This is the main distinguishing feature when comparing to traditional methods such as PCA, LDA, LLE or ISOMAP, which are based on input features linear relationships.

In addition, using ensembles implies the integration of a set of components that solve the same task in a single predictive model. Some studies show that these components offer better results working together than used in isolation [38, 77]. Furthermore, there are others that relate the use of ensembles and DAEs [5]. These reasons, among others further extended in Section 2.5, justify the use of ensembles in this proposal.

Specifically, the Classifier Ensemble DAE (ClEnDAE) model is proposed in this work. This algorithm is based on an ensemble of Denoising AEs (DAEs) to reduce the dimensionality of the input space. The new variables generated by ClEnDAE in the feature fusion phase can be used as a new representation of the input data and serve as input for traditional data mining methodologies. In the second phase of ClEnDAE, different algorithms can be used to classify. Concretely, the possible classifiers are k-Nearest Neighbors (kNN) [24], Support Vector Machines (SVM) [16], Multi-Layer Perceptron (MLP) [49] and C4.5 [70]. The ClEnDAE method is parameterized so as to select the desired classifier from the previous options. Based on the results of these methods, it is possible to evaluate the quality of the features generated by the proposed model. Fundamentally, the classifiers are applied using the new representation of the input generated by the different components of the ensemble. Finally, the information provided by them is combined to generate the final response of the model.

In summary, ClEnDAE combines the information provided by a certain number of models in order to classify new data. This method addresses the reduction of dimensionality from different perspectives. Firstly, each component is trained with a subset of the original examples and features randomly sampled with replacement, and then a new representation is obtained through the use of DAEs. This process allows each model to face the reduction of dimensionality through DAEs before classifying by four classical algorithms: kNN, SVM, MLP and C4.5.

More precisely, the main contributions of this work are: 1) the development of a new classification model, ClEnDAE, based on ensembles and DAEs to reduce the input dimensionality, 2) a thorough experimentation to verify the behavior of the proposed model against traditional methods and a model based on basic DAE, 3) a comparison between ClEnDAE and four classic models of dimensionality reduction, such as PCA, LDA, ISOMAP and LLE, and 4) a general conclusion about how to use and operate the proposed algorithm according to the characteristics of the input data.

Lastly, the experimentation developed to verify the proper functioning of the ClEnDAE algorithm, based on 20 datasets, shows significant improvements over traditional classification methods. Additionally, ClEnDAE achieves better performance in most analyzed cases than traditional dimensionality reduction algorithms such as PCA, LDA, ISOMAP and LLE. These data show the value of ClEnDAE when facing the task of dimensionality reduction, considerably improving predictive performance compared to classical methods.

The article is organized as follows. In Section 2, main theoretical concepts related to the model which is developed in this article are presented. Section 3 details the proposed ClEnDAE method. In Section 4, the experimental framework is presented and the obtained results are analyzed. Finally, Section 5 presents general conclusions of this study.

2. Preliminaries

In this section all theoretical concepts related to ClEnDAE are presented. In Subsection 2.1, the problem of high dimensionality and some traditional proposals to deal with it are described. Subsection 2.2 presents the traditional classifiers applied, as well as the family of algorithms they belong to. In addition, the concept of AE and its inner workings is analyzed in Subsection 2.3. The AE model used in this paper, DAE, is outlined in Subsection 2.4. Finally, the ensemble methodology is detailed in Section 2.5.

2.1. Dimensionality reduction methods

A frequent characteristic of any classification methodology, among which are the four presented in Section 2.2, is that its performance is affected by complexity of the input data. In general, these algorithms use data from very diverse sources and varied scopes. In recent years growing dimensionality is a common factor in these data [21]. This property is one of the characteristics that negatively affects predictive performance in many classifiers, due to the phenomenon known as curse of the dimensionality [9]. This factor refers to the decrease in classifier performance from a certain high number of features. For this reason, it is necessary to generate a new feature space of lower dimensionality, by selecting the most relevant attributes or creating new features that aggregate the most important information.

Furthermore, another effect related to this implies that the performance of a classifier will be affected in a high dimensional space by a large number of redundant features. This will occur when a limited number of training instances are used, which is known as Hughes phenomenon [51]. In many cases, models need to increase the number of training examples to generate acceptable classification results. In this subsection some proposals that try to mitigate these effects are presented.

Over the years, a large number of dimensionality reduction methods have been designed. First proposals consisted in experts' direct selection of the most relevant characteristics through a manual process. Shortly, automatic methods for feature selection and extraction began to emerge [25, 41]. Some of the best known and used automatic methods in the experimentation associated with this work are: LDA [63], PCA [50], ISOMAP [81] and LLE [78]. Currently, there are some new proposals based on DL techniques. In particular, AEs have shown good results when it comes to reducing dimensionality [20, 67].

There are different paradigms that face the task of dimensionality reduction. Here are some of the most important ones:

- Feature selection: This type of method selects a subset of features of the input data. This process does not modify the original features, but rather selects those considered most relevant. The disadvantage of some basic methods is that they treat variables independently, without taking into account information provided by the complete set [25].
- Feature extraction: These algorithms try to generate a new representation for input data with lower dimensionality [43]. To do so, it is very common to perform linear data transformations. This technique will depend on the input data, as well as the subsequent use of the generated information. Some of the most popular methods within this methodology are LDA, PCA, ISOMAP and LLE.
- Feature fusion: The increasing use of multimedia data in late years has led to the emergence of a new dimensionality reduction approach, known as Feature Fusion [59, 58]. The main objective is to generate a new space of features that contains the most relevant information of the input space, discarding irrelevant and unuseful information. Therefore, the use of new features in later stages improves the performance of the models, for example, classifiers. In this context, one of the most applied techniques to develop algorithms is DL, specifically AEs.

As indicated before, there is a large number of models to tackle dimensionality reduction. However, feature fusion models based on AEs have proven to be very efficient and offer good performance [20, 66]. For this reason, the ClEnDAE model is based on DAE, a specific type of AE, to mitigate the effects of high dimensionality. In Subsection 2.3, the concept of AE is introduced and Subsection 2.4 presents the term DAE.

2.2. Classifiers

Classification is among the most important tasks in the machine learning field [30]. As a result, every year new proposals emerge to face this task from different perspectives and applying very varied methodologies. These families are categorized according to different aspects, among which the architecture and the operation of the developed model stand out. The fact that there are different types of classifiers and many methods associated with each of them allows for a wide variety of options to address the classification problem. However, this means knowing the characteristics of each algorithm is needed to use the one that best suits the data of the problem. Later on, four classifiers belonging to four of the best known and used paradigms will be considered. However, there are many other types of methods, for example, rule-based systems [94], deductive logic programming [62] or genetic algorithms [26], among others. Next, the four families of classifiers and the selected methods are described.

- Instance-Based Learning (IBL): The main characteristic of this type of methods is that it does not build a model from the training data, that is, it is a *lazy* paradigm. This type of algorithm uses the information provided by the training instances to directly infer the prediction corresponding to each new example. Normally, these algorithms base the response on a specific subset of the training space that are related to the new instance [2].

  In this context, one of the most widespread algorithms is kNN. It is a nonparametric method used for different types of tasks, including classification or regression [4, 24]. kNN is a *lazy* method that, as stated above, does not need to build a model to generate the prediction [6]. Instead, kNN uses the information provided by the k instances closest to the new example. The distances are calculated in the n-dimensional space generated by the input features. In this way, the output provided will be the most common among the neighboring k.

- Support Vector Machines (SVMs): These methods are based on the spatial representation of the instances. In particular, a relationship between the different instances and points of space is established. Once the instances are located in the space, the process consists in determining the support vectors of the hyperplane that achieves a greater separation between the points represented. Thus, the elements of different classes are separated in space and the classification of new examples only consists in determining which set is the closest [45].

  The SVM algorithm was introduced in 1992 [16] and it is commonly used due to its good performance and robustness. During the learning process, the method projects the instances in space and then determines the hyperplane that offers a greater separation between them. Lastly, the prediction is based on spatial separation. On certain occasions, it is not possible to establish a separation between the examples. In these cases, the algorithm uses nonlinear mapping techniques on the original feature space [45].

- Artificial Neural Networks (ANNs): These models are based on the functioning of the human brain. The structure of the model is based on different layers composed of a variable number of elements, called neurons. In addition, these elements are related to each other through different types of connections with weights. It is important to note that these models use nonlinear transformations on the input to determine the output, so both weights and nonlinear transformations are relevant. During the training process, the ANNs learn by themselves the high-level relationships between the data. This allows them to make predictions when they receive unlabeled data [79].

  To evaluate the ANNs in this study, MLP [49] is the model used. The main reason is that it is a simple model but offers a good classification performance. MLP allows to generalize and extract knowledge from input data. Throughout the learning process, the model adjusts the weights according to the problem data. Finally, the model generates an output vector which is the result of mapping the input data along the network [39].

- Decision Trees (DTs): The most identifying feature of this type of model is its structure, based on trees. The fundamental parts of these models are branches and leaves. The branches allow evaluating the attributes of the data that determine a greater separation of the examples in different categories. The leaves determine the value of the target class. During the prediction process, the model will evaluate each of the attributes of the instance (branches) until it reaches a target class value (leave) [70].

  There is a large number of tree-based methods that can be applied to classification. One of the classic methods within this paradigm is C4.5. This algorithm is frequently used for classification. Fundamentally, the branches allow to evaluate the different features, this evaluation guides the process towards one of the tree leaves that contains the value of the objective class. Particularly, C4.5 generates a series of classification rules to make new predictions [70, 89].

High dimensionality affects the four classification paradigms presented above. However, the reasons for the decrease in performance are different in each of them. The IBL and SVM methods base their operation on distances between the input examples. In spaces of high dimensionality, the distances tend to equalize, so the predictive performance decreases noticeably. On the one hand, IBL algorithms using less significant distances cannot correctly identify the nearest neighbors, which implies worse operation. On the other hand, SVM cannot establish valuable relationships between instances based on distances that are not significant. Similarly, the high dimensionality negatively affects the ANNs. The use of more features prevents extracting relevant high-level relationships, since, in many cases, it involves irrelevant or redundant information. Lastly, decision trees produce gigantic structures when they work with high dimensional data, which means that the evaluation time of each of the nodes is very high and the computational performance of the model decreases.

For these reasons, the four aforementioned algorithms have been selected for this work. kNN, SVM, MLP and C4.5 will be used to evaluate the performance of ClEnDAE. In this way, the behavior of the proposed algorithm is analyzed from different perspectives. This allows to obtain a much more general vision and to draw conclusions about the effects of the reduction of dimensionality with the ClEnDAE method when applying four different classification methodologies. In Subsection 3 the ClEnDAE method is presented and Section 4 details the experimentation conducted to evaluate it.

2.3. Autoencoder foundations

The dimensionality reduction model proposed in this paper is based on the use of a type of AEs. An AE is a kind of ANN that has a series of specific characteristics. The main purpose of an AE is to reproduce the input of the network at the output [47, 64]. The model uses only the input attributes to learn an internal representation that allows it to produce the appropriate output, i.e., it is a nonsupervised learning method. To achieve this goal, AEs have a specific structure that also prevents the network from merely copying the input [20, 67].

AEs have a symmetric structure. It is clear that the input and output layers must have the same size, since their objective is to reproduce the input in the output. However, it is not only these two layers, but the entire network is completely symmetric. This structure allows the network to learn an internal encoding from which it can reconstruct the input. There are different types of AEs according to the size of the hidden layer (or layers):

- Overcomplete: The hidden layer has more elements than the input and output layer. This type of models allows to obtain a sparse representation of the original data.

- Undercomplete: The hidden layer has fewer elements than the input and output layer. In this case, the network learns an internal coding of smaller dimensionality than the original data. Therefore, it is the appropriate model to face dimensionality reduction.

This study focuses on undercomplete AEs, since they are the most suitable to reduce the dimensionality of the input space thanks to their architecture. AEs of this kind generate an encoding in their hidden layer by combining the most relevant information of the input and discarding redundant or unuseful content. In this manner, this new representation of the input information constitutes a new space of characteristics of lower dimensionality, obtained from the original features [47, 88].

A basic AE is a feed-forward neural network [49] where there are no cyclic connections. The model is composed of a series of successive layers each consisting of multiple neurons. These elements are interconnected and there are weights associated with each of the connections. During the training process, the network adjusts the weights of the connections according to the input data.

The minimum architecture of an AE contains an input layer, a hidden layer, and an output layer. This architecture can be more complex by adding extra hidden layers. Nevertheless, the structure must always be symmetric according to the characteristics of the AEs [67]. Figure 1 presents an example of basic architecture of AE.

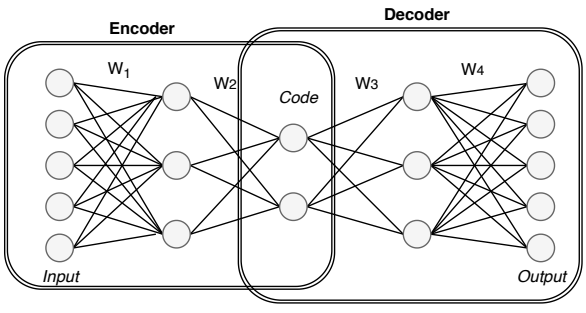


Figure 1: AE basic architecture.

Figure 1 shows that the AE is formed by two fundamental blocks. The *encoder*, where the model learns the coding of the input, and the *decoder*, where the network reconstructs the input into the output from the coding. Another important aspect is that each neuron is connected to the whole next layer through a series of weights, denoted by W_i in the figure.

The architecture described in this Section corresponds to the most basic model of AE. However, there are different types of AEs with more complex functionings, some of them are: robust AE [68], contractive AE [73] or DAE [86]. In Section 2.4, DAE model is detailed, since it is the basis of the proposal made in this study.

2.4. Denoising Autoencoder

AEs are useful models to address feature fusion. Nonetheless, this type of networks can be limited to copying input information, namely, learning the identity function. This occurs especially when the number of nodes in the hidden layer exceeds the amount of input nodes. This situation renders the work of the AE useless.

In this scenario, the reconstruction method by itself is not capable of guaranteeing the generation of useful features. There are different strategies to deal with this problem. A possibility is constraining the original data: bottleneck or sparse representations of the input. Furthermore, another alternative involves a change in the reconstruction criterion that consists in cleaning or eliminating the noise of a partially corrupt input [87, 65]. This is the foundation of DAEs.

In this context, the model learns a new representation from a corrupt input and is able to reconstruct the original one, without noise. This process generates a new representation of higher quality and robustness. In addition, this model, due to its operation, is less sensitive to the real noise that may exist in the input data.

At this point, it is important to emphasize that eliminating noise is not the objective of this type of models. Instead, the noise introduced in the first layer allows the methods to learn a higher level representation [67, 20].

DAEs are models that follow this methodology for feature fusion. There are different ways to introduce noise at the input. An example could be a DAE that changes random values from the input to the value 0. The number of elements of the input that are changed oscillates between 30% and 50%. Nevertheless, this amount will depend on the dimensionality of the input data and the architecture of the network. A fundamental aspect to keep in mind is that, when calculating the loss function, the model must compare the output obtained with the original input, without noise. As a result, the possibility of totally or partially copying the input is practically eliminated, since the corrupt input is different from the expected output.

The DAE model is similar to a basic AE, but introduces some modifications. The objective of the DAE is to reproduce the clean input from a corrupt version of it. Therefore, the first step is to perform a stochastic mapping of the input [86, 87].

The input with noise is mapped through the AE to a coding through the hidden layers. Finally, the output of the network is reconstructed from the coding generated in the hidden layers of the network. Throughout the training process, the network adjusts the parameters to generate an output as close as possible to the original input. The fundamental difference is the use of a corrupt input instead of the clean one.

The most basic DAE architecture with a single hidden layer has a series of parameters that are adjusted during the training process, most of them, except for the input modified with noise, are similar to the basic AE. Specifically, two weights matrices, W and W', and two bias vectors, b and b'. Thus, the coding and decoding functions can be expressed in the following way, where x' is the corrupted input:

$$z = f(x') = \gamma_1 (W\, x' + b) \tag{1}$$

$$y = g(z) = \gamma_2 (W'\, z + b') \tag{2}$$

Equation 1 is the compression function, where the corrupt input x' is used and the coding z is generated. Likewise, equation 2 is the reconstruction function, where the z coding is used to generate the output of the network y. In these two equations, γ_1 and γ_2 are two nonlinear activation functions.

The goal of the DAE is to minimize the reconstruction error. There are different loss functions such as mean square error (MSE) or cross entropy. Once the error is determined, it is necessary to adjust the parameters of the network. There are different methods for doing this, such as stochastic gradient descent (SGD) [75] and its variants, including RMSProp [82] and AdaGrad [29].

The main purpose of the different SGD methods is to adjust the parameters in such a way that the objective function is minimized. To do so, a backward propagation process goes through the network from the output layer to the input, transmitting the necessary adjustments to optimize the process [19]. Figure 2 presents a diagram showing the training process of a DAE.

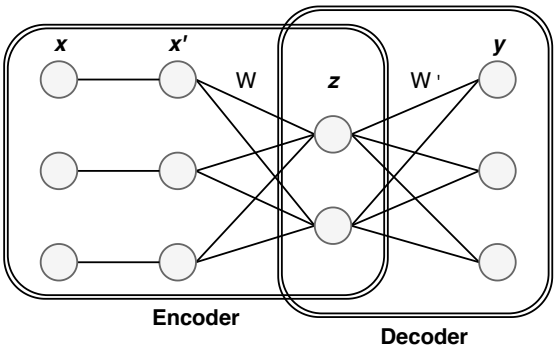


Figure 2: DAE architecture.

The training process of DAE is very similar to the one used in a basic AE. However, introducing noise in the original input makes the network more stable against noise and allows the generation of much more robust features. Because of this, the ClEnDAE model is based on this type of AE to face the task of dimensionality reduction.

2.5. Ensemble

The ClEnDAE model is based on ensembles due to their advantages and their good results in other areas [35, 54, 74, 38, 80]. An ensemble is a composition of several machine learning models. The main advantages provided by a model based on ensembles are:

- The information provided by the different models can be combined, producing a classifier profile that is not possible to obtain from a single model.
- Since each model is based on different subsets of instances and attributes obtained from the input, the ensemble is able to generalize information based on different sets, whereas a simple model generally uses the full set and all input features.
- Ensembles can be parallelized in a simpler way than a single model. The nature of the ensembles allows parallel execution to be direct. Some traditional methods need to be redesigned to work in parallel, while the architecture of the ensembles make them perfectly usable for this type of execution.

From a general point of view, the methodologies used in ensembles consist of a series of models that work on the original dataset in different ways and combine the outputs to generate a single result. There are two large groups of ensembles: sequential and parallel. Besides, there are different methodologies when developing an ensemble [8, 69], the best known and used are:

- Bagging: This methodology, also known as Bootstrap aggregation, is one of the basic paradigms used for ensembles development. This algorithm is based on the statistical method of bootstrapping [31] to jointly evaluate the execution of several models run in parallel. Bagging does not use the complete original dataset to train each of the models, but a number of input examples that are randomly selected [17].
- Boosting: This classical algorithm is commonly used in the development of ensemble-based methods [32], which is based on the reduction of bias by combining different models. Each component is added sequentially and focuses on instances misclassified by previous components. In this way, the final model is fitted more precisely to the training set, significantly improving the classification of problematic instances for individual models.
- Stacking: It is a methodology based on a sequential structure, where the outputs provided by the previous models are combined and used as input of new methods that generate a new set of predictions [90]. It is important to note that, in this case, all the models that make up the ensemble use the complete input set.

In conclusion, the methods based on ensembles allow to expand the search space of the traditional classification algorithms, since they incorporate several components that use different sets obtained from the original data. The above-mentioned advantages have a positive impact on the final performance of this type of algorithms. On account of this, the ClEnDAE proposal made in this work incorporates the use of this methodology.

3. ClEnDAE: A classifier based on ensembles with built-in dimensionality reduction through denoising autoencoders

Once main concepts of the theoretical framework associated with our proposal have been described, this section presents ClEnDAE. An ensemble-based classification method that incorporates dimensionality reduction through DAEs.

3.1. ClEnDAE foundations

In this study, the ClEnDAE model is proposed. This method is based on two fundamental pillars: DAEs and ensembles. This work is derived from the hypothesis that the an ensemble-based classifier will improve its predictive performance when incorporating DAEs to reduce dimensionality. This approach leverages the ability of DAEs to reduce dimensionality [65] and the advantages of using ensembles.

ClEnDAE trains the different components with a random subset of both instances and features obtained from the original dataset. Additionally, this method is based on the execution of several models in parallel with the aim of reducing variance. In this manner, generated predictive results are more stable and precise. Last of all, in order to combine the outputs of the different models, ClEnDAE uses a voting system, one of the most common techniques, which means the output of the ensemble corresponds to the majority result obtained by the models that form it.

The number of components of an ensemble depends on many factors, including properties of the input data and the number of output classes [15]. In our experiments, the same ClEnDAE configuration is applied to all the datasets that make up the experimental framework. The aim of this is to determine from a general perspective the performance of ClEnDAE with a large set of data with different characteristics. For this reason, the number of components is 90, since it represents a significant number of units for each of the three architectures of DAE being considered and an appropriate generic value for the different datasets. Nonetheless, when applying ClEnDAE to solve a specific problem, the number of components should be adjusted to data characteristics to obtain optimal results. Therefore, ClEnDAE algorithm has a specific parameter that defines the amount of components in the ensemble.

Specifically, each unit contains a DAE to reduce dimensionality of the input space. Subsequently, the new feature space is used by a classifier so each component uses a different feature space generated by its DAE. The final result corresponds to the majority output provided by the 90 models. On account of this, the process followed has three fundamental phases:

- Dimensionality reduction: From a subset of instances and features of the input data, each model generates a new feature space. In this phase, DAEs with different architectures are used. These architectures are adapted to generate a space of lower dimensionality.
- Classification: Data generated by each of the DAEs are used by the same type of classifier. More clearly, ClEnDAE has four possible classifiers: kNN, SVM, MLP and C4.5. The method is selected by means of one of the ClEnDAE parameters.
- Output: The final response of the model will be the majority output by different classifiers.

In this context, it is important to emphasize that there are different architectures in the DAEs used in the components of ClEnDAE. In particular, three architectures with a variable degree of dimensionality reduction are used. The architecture of the ClEnDAE model, which is made up of 90 components, has 30 models that reduce dimensionality to 75%, 30 more to 50% and the last 30 to 25%. Thus, in each group, DAEs have a hidden coding layer whose number of neurons corresponds to 75%, 50% and 25% of the original features, respectively. The use of several architectures allows the model to learn high-level characteristics associated with datasets with a different compression degree, which increases the amount of relevant information in comparison to configurations of a single type. The choice of these architectures is based on a series of previous works that show their better performance against other possible values [67, 65].

Besides, as indicated above, each component uses a different subset of instances and features obtained from the input. To achieve this, a certain percentage of both instances and attributes are randomly selected. This allows the model search space to be diversified. In this way, the ClEnDAE algorithm performs a high dimensionality treatment from different perspectives.

- In relation to training instances: Each model randomly selects a different set.
- From the perspective of the features: In the first place it makes a random sample with replacement of the variables and, later, it learns a new space of features through the DAE.

In short, ClEnDAE uses a philosophy similar to Random Forest [18], one of the most popular algorithms based on ensembles, when training each component with a random selection of instances and features of the original set. However, ClEnDAE incorporates more phases to tackle reduction of dimensionality from another perspective:

- Each component obtains a new representation of the input variables by means of DAE from different random subsets.
- Using new feature spaces, each component obtains a classification model.
- The final output is obtained by combining the outputs of each component by a majority vote.

3.2. Method description

ClEnDAE consists of three phases. Firstly, several random sets of instances and features are obtained from the original input. Particularly, each subset contains 63.2% of the training examples and 75% of the input features, in both cases a random selection with replacement is made. The random choice of 63.2% of instances ensures that all examples are selected at least once during the training process, considering a sufficiently large number of components [8, 69], like in this work. Furthermore, the selection of 75% of features is based on a series of previous works that show that this reduction degree produces better performance [67, 65]. To sum up, this random selection of both instances and characteristics implies that the search space is diversified in both directions.

Secondly, these subsets are used to train the 90 components of the ensemble. This process consists of training a DAE to obtain a new representation of lower dimensionality and, subsequently, using the new space to train a classifier to obtain the prediction. Lastly, the outputs of all components are added to generate the final output of the model. Figure 3 presents the architecture of the ClEnDAE model proposed in this paper. The image shows the three phases of random selection, reduction of dimensionality and classification, as well as different architectures of DAEs. In addition, Algorithm 1 presents the pseudocode of ClEnDAE.

The input parameters of Algorithm 1 are described below:

- *TrainData* and *TestData*: The training and test data processed by the algorithm.
- *NumComponent*: The number of models that make up the ensemble. This value has been set to 90, as explained above.
- *ArqDAE*: The configuration of the different structures of the DAEs used in the components of the ensemble. In this study, three different architectures have been considered, so there are 30 models with each configuration. This parameter sets the number of layers and elements per layer for each configuration.
- *Clas*: The classifier that the algorithm uses. In this work, four different classifiers have been considered: kNN, SVM, MLP and C4.5. All components use the same classifier.
- *Par*: The parameters associated with the selected classification algorithm.

The functioning of the method is divided into two different blocks. The first part (lines 1-14) carries out the training of the ClEnDAE model. The second part of the code (lines 16-18) performs classification of the test set to evaluate this model.

The training process has two fundamental objectives. First of all, ClEnDAE focuses on learning a new representation of the input data to face dimensionality reduction through DAEs. Secondly, the selected classification model is trained. These

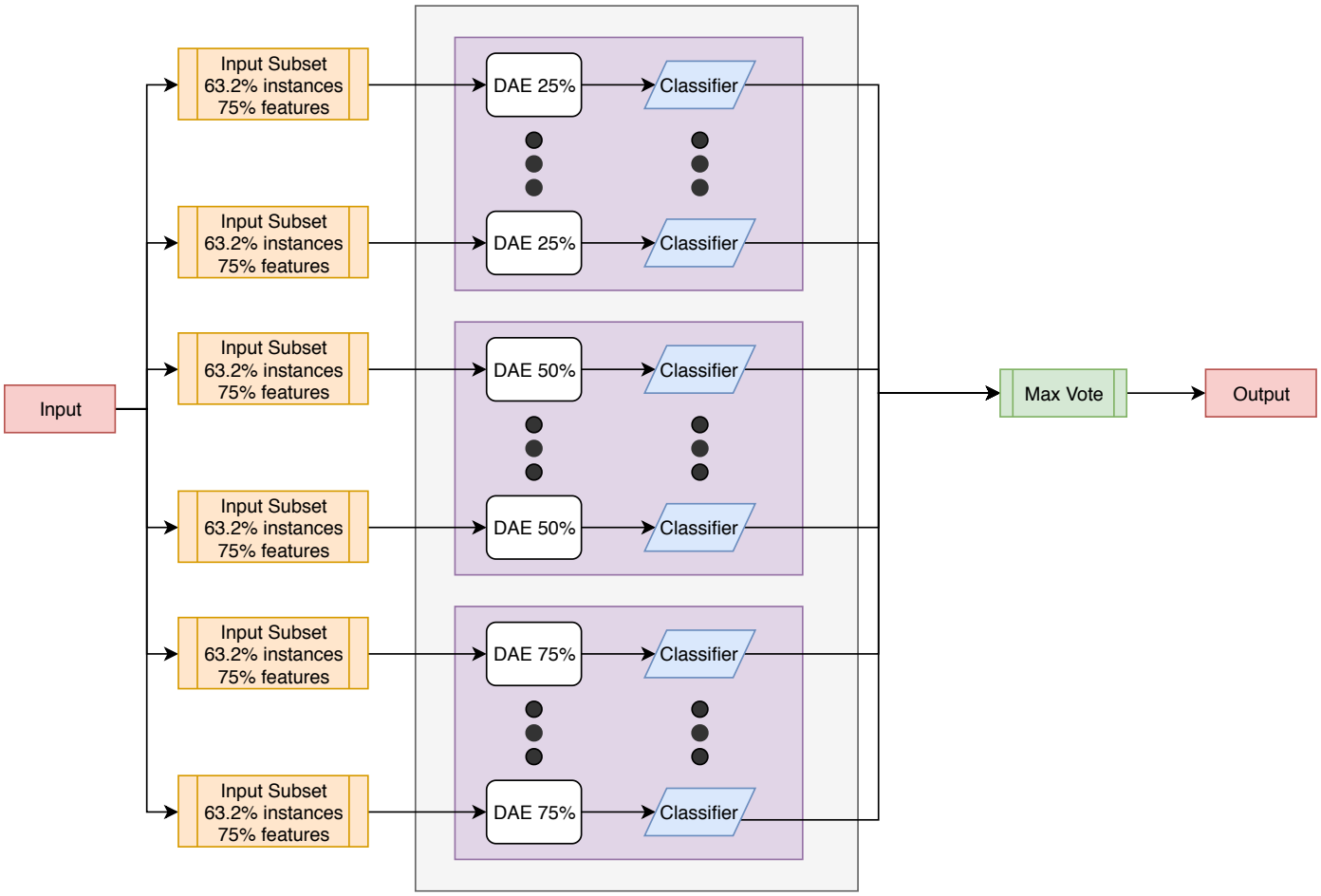


Figure 3: Architecture of the ClEnDAE model proposed in this work.

phases are done in each of the ClEnDAE components (lines 3-14). The code associated with this phase is described in more detail below:

- In line 2 the initialization of the ensemble is carried out. This process consists in the creation of the structure of the model and all the associated components. The *NumComponent* parameter defines the number of components that will be created at this point. Likewise, the *ArqDAE* parameter defines the different architectures that will have the DAEs of said components of the ensemble, which will also be initialized.
- Lines 3-14 correspond to a loop that performs the following tasks for each component of the ensemble:
  - Line 5: Random selection of 75% of the features and 63.2% of the instances of the training data. This allows to diversify the search space of the different components.
  - Line 7: Training of the DAE responsible for obtaining a new representation with less dimensionality of the input data.
  - Line 9: A new representation of the input subset is obtained using the DAE trained in the previous step.
  - Line 11: Training of the selected classifier. In this work, four different classifiers have been considered: kNN, SVM, MLP and C4.5. In this sense, it is important to emphasize that not all models need this phase (for example kNN), therefore, in these cases only the initialization of the classifier is performed.
  - Line 13: The component is updated with the DAE and classifier already trained.

Throughout the process described above, each component of the ClEnDAE conducts a DAE training by using the training data to learn the weights of the network that produce a better representation of the input. This stage corresponds to the *trainDAE* function detailed below:

- The first step is to introduce noise into the input (line 21). This is key in the operation of DAEs, which consists in learning to rebuild the input from corrupted data.
- As can be seen in the code (for loop, lines 23-26), the structure corresponding to the DAE is created, adding all the necessary layers. Each iteration of the for loop generates a new layer of the model.
- Once initialized, the DAE is trained using the training data

Algorithm 1 ClEnDAE algorithm's pseudo-code.

```
    Inputs:
        TrainData                          ▷ Train Data
        TestData                           ▷ Test Data
        NumComponent                       ▷ Number of components in ensemble
        ArqDAE                             ▷ Architecture DAE
        Clas                               ▷ Classifier
        Par                                ▷ Parameters of classifier
 1: ▷ Training:
 2: ensemble ← initEnsemble(NumComponent, ArqDAE)
 3: for each component in ensemble do
 4:     ▷ Instances and features selection:
 5:     componentData ← randomSelection(TrainData)
 6:     ▷ Train DAE:
 7:     daeModel ← trainDAE(componentData, component)
 8:     ▷ New representation through DAE:
 9:     componentData ← applyDAE(componentData, daeModel)
10:     ▷ Train classifier:
11:     clasModel ← trainClassifier(componentData, Clas, Par)
12:     ▷ Update component:
13:     ensemble ← updateComponent(component, daeModel, clasModel)
14: end for
15:
16: ▷ Classification:
17: result ← classification(TestData, ensemble)
18: return result
19:
20: function TRAINDAE(componentData, component)
21:     modelData ← corruptedInput(componentData)
22:     daeModel ← ()
23:     for each layer in component do
24:         sizeLayer ← getSizeLayer(modelData, layer)
25:         daeModel ← addDAELayer(sizeLayer)
26:     end for
27:     daeModel ← compileDAEModel(modelData, daeModel)
28:     component ← addDAEModel(daeModel)
29:     return component
30: end function
31:
32: function COMPILEDAEMODEL(modelData, daeModel)
33:     for each instance in modelData do
34:         outPut ← feedForwardDAE(daeModel, instance)
35:         error ← calculateDeviation(instance, outPut)
36:         daeModel ← updateWeightsDAE(daeModel, error)
37:     end for
38:     return daeModel
39: end function
40:
41: function CLASSIFICATION(TestData,ensemble)
42:     error ← 0
43:     for each instance in TestData do
44:         outputs ← ()
45:         for each component in ensemble do
46:             newRep ← applyDAE(instance, component)
47:             componentOutput ← applyClassifier(newRep, component)
48:             outputs ← addOutput(outputs, componentOutput)
49:         end for
50:         finalOutput ← selectMaxorityOutput(outputs))
51:         error ← updateError(instance, outPut))
52:     end for
53:     result ← calculateError(TestData, error)
54:     return result
55: end function
```

in order to update its parameters. Procedure *compileDAEModel* iteratively compiles the structure of the DAE. This function is encoded on lines 32-39. To carry out this process, the model generates a new representation for each training instance (line 33), obtains the reconstruction error of the original input (line 34) and updates the weights to minimize the error generated (line 35). Last of all, the DAE model is returned by the function (line 37).

- The DAE model obtained in this function is added to the component (line 28) and this is returned (line 29).

To carry out the classification, function *classification* (lines 41-55) is used. In this process each component obtains the class prediction for test instances (lines 45-49) and all the information is added to generate the final output, selecting the most common output (line 50). There are the following phases to carry out this prediction in each component:

1. Firstly, a new coding of the instance is obtained using the DAE model included in each component during the training phase (lines 46). This representation generates a new feature space of a higher level but lower dimensionality.
2. Secondly, the coding computed in the previous step is used as input of the classification algorithm generating the final prediction of the current component (line 47). Each component has an associated classification model produced during the training phase.
3. Thirdly, the output of each of the components is saved (line 48).

Lastly, the majority output is selected (line 50) and the error rate is determined, comparing the obtained output against the actual one (line 51). The global error value is returned by the algorithm to evaluate its behavior (lines 53-54).

As can be seen, the classification phase explained above uses data of lower dimensionality than the original input, reducing the negative effects caused by the existence of too many features.

Finally, it is necessary to clarify several aspects related to the algorithm that has just been described. For once, the DAE training process is done using mini-batch gradient descent. This method is a variation of the gradient descent that divides the dataset into small batches that are used independently to calculate the error and modify model parameters. This operation is suitable when a high-dimensional dataset is used [46].

Besides, as aforementioned, all architectures have a single hidden layer, in addition to the input and output layers. Different studies in the literature have shown that the use of a single hidden layer generates better results when facing the task of dimensionality reduction [67, 65]. Similarly, having several architectures implies generating different degrees of dimensionality reduction. Therefore, the components will obtain high level characteristics with several degrees of generalization.

3.3. ClEnDAE complexity

In this section, the objective is to analyze the computational complexity of the ClEnDAE algorithm. Algorithm 1 above described is fundamentally divided into two parts: training of the

model and classification of the test set. First, considering N instances of training data, n features of the dataset, M the number of components of the ensemble and Clt the complexity of classifier training, computational complexity for the training block would be the following:

$$\begin{gathered} C_{ClEnDAE_T} = O(1 + M(\frac{3}{4} \times n + M \times N + N + Clt + 1) = \\ O(1 + M(C \times n + M \times N + N + Clt)) = \\ O(1 + M \times n + M^2 \times N + M \times N + M \times Clt) = \\ O(M \times (n + M \times N + N + Clt)) = \\ O(M^2 \times N + M \times N + M \times n + M \times Clt) \end{gathered} \tag{3}$$

Expression 3 shows the quadratic factor M^2 being the element of greatest weight. However, this will be the case only when $M \geq N/n$. In this study, it is possible to assume that $M << N$ and $M << n$, since high-dimensional datasets are used. In consequence, M^2 would remain as an additive factor to $M \times Clt$, this being the factor of greatest weight.

Second, considering N the number of instances of the test set, M the number of components and Clc the complexity of classifier testing, the computational complexity of the classification block corresponds to:

$$C_{ClEnDAE_C} = O(N \times M \times Clc) \tag{4}$$

To conclude, the computational complexity of both parts of the algorithm depends on the parameters of ClEnDAE. Additionally, this factor also depends on the complexity of the selected classifier.

3.4. ClEnDAE contributions

ClEnDAE is an ensemble-based classifier that incorporates DAEs to reduce the dimensionality of the input data and mitigate the associated negative effects. The main contributions of this proposal are the following:

- ClEnDAE faces high dimensionality from different angles. Each component uses a random selection of both instances and features of the original set. Moreover, the model learns a new representation of the feature space through DAEs.
- The use of DAEs by ClEnDAE to tackle dimensionality reduction delivers improvements in the predictive performance of many traditional classifiers, as described in the experimentation.
- ClEnDAE is a model based on ensembles. This relies on the predictive performance of several components working together being greater than that of the models used independently.
- Each component of the ensemble uses a random selection of both instances and features to train. Because of this, the ClEnDAE search space is diversified, improving predictive performance.
- The algorithm is parameterized to use four different algorithms in the classification phase: kNN, SVM, MLP and C4.5. This allows to evaluate the behavior of the method considering four classic classifiers belonging to different families.

In addition to presenting the ClEnDAE model, this work intends to demonstrate its proper functioning when addressing classification tasks, mitigating the effects of high dimensionality. To that effect, an exhaustive experimental analysis is performed in Section 4. This study is divided into two phases: a comparison of the predictive performance of the model using different traditional classifiers, and an analysis of the results obtained using the ClEnDAE model against other traditional dimensionality reduction algorithms.

4. Experimental Study

There are studies that have shown the benefits of using AEs to deal with the feature fusion task [61, 20, 67, 95, 1, 92]. Besides, there are also works that focus on the use of DAEs [65]. The main objective of the experimentation developed in this paper is to demonstrate the improvements obtained when facing this task using ClEnDAE. To that end, it is necessary to conduct an exhaustive comparison between a basic model of DAE, a model based on the ensemble of DAEs and the results obtained from the original data. The process to execute each model is as follows:

- Basic classifier (baseline): In this case, the original raw data is used to classify by means of algorithms kNN, SVM, MLP and C4.5.
- Basic DAE: In a first stage, the task of feature fusion is performed. In this manner, different lower-dimensional subsets are obtained for each original dataset. The architecture used has a hidden layer that compresses 75% of the original characteristics. This configuration has shown the best performance in previous works [67, 66]. Then, the classification with different traditional methods is performed. Each algorithm uses the subsets generated by the basic DAE. The classifiers applied in this phase are: kNN, MLP, SVM and C4.5.
- ClEnDAE: The algorithm proposed in this work is executed for four different classification algorithms: kNN, SVM, MLP and C4.5. For each dataset, four executions are made by changing the parameter corresponding to the classifier.

Afterwards, the classification performance of the different methods can be compared, establishing which model offers the best results.

In the following subsections, the development of the experimentation explained above is presented. Fundamentally, it intends to achieve the following goals:

Table 1: Characteristics of the datasets used in the experimentation.

Dataset	Samples	Number of Features	Classes	Type	Field	Ref.
arcene	900	10000	2	Real	Medical	[43]
batch	13910	128	6	Real	Chemical	[84]
coil2000	9822	85	2	Integer	Social	[3]
dota	102944	116	2	Real	Game	[7]
drive	58509	48	11	Real	Motor	[7]
facial	2964	301	2	Real	Image	[7]
fashionmnist	70000	784	10	Integer	Image	[91]
gisette	13500	5000	2	Integer	Image	[43]
hapt	10929	561	12	Real	Activity	[72]
image	2310	19	7	Real	Image	[7]
isolet	7797	617	26	Real	Image	[23]
letter	20000	16	26	Integer	Image	[7]
madelon	2000	500	2	Real	Artificial	[42]
mfeat	2000	649	10	Real	Image	[7]
microv1	360	1300	10	Real	Biology	[7]
microv2	571	1300	20	Real	Biology	[7]
mnist	70000	784	10	Integer	Image	[56]
musk	6598	168	2	Integer	Physical	[7]
nomao	1970	118	2	Real	Technology	[7]
semeion	1593	256	10	Integer	Image	[7]

- To determine the performance of our proposal, ClEnDAE. To do so, an exhaustive comparison of the results from this model against the results from a basic DAE model and the original data is performed in Subsection 4.2. This comparison is made for each dataset and for each of the four classification algorithms considered in this work.
- To compare the results obtained with the ClEnDAE model and four traditional methods of dimensionality reduction: LDA, PCA, ISOMAP and LLE, as shown in Subsection 4.4.
- To present some general conclusions on the results obtained in this study in the Subsection 4.3 with the following purposes:
  - To analyze the performance of the different models according to each classifier.
  - To determine which model offers better performance for each case.
  - To present new ways of future work based on the results obtained in this study.

Furthermore, before presenting the results of the experimentation, Subsection 4.1 describes the framework used in this study: datasets, metric, statistical tests, AEs architecture and classification algorithms.

4.1. Experimental framework

To establish a comparison between the models considered in this study, it is necessary to use a broad set of datasets that have varied characteristics. Table 1 presents the datasets included in this experimentation, as well as their most relevant traits. The **Field** column presents the field of application and the column **Ref.** the origin of each dataset.

Additionally, to evaluate the different models, the area under the ROC curve (AUC) is used in this study. The main advantage of this metric is that it offers a robust view of the predictive performance of the analyzed methods. In other words, AUC provides a global view of the results, while other metrics such as Accuracy or Precision give a more partial view [67]. AUC is the probability that a model will classify a randomly chosen positive instance higher than a randomly chosen negative one. Equation (5) presents the equation for AUC:

$$AUC = \int_{\infty}^{-\infty} TPR(T)FPR(T)dT \tag{5}$$

where *TPR* is the true positive rate and *FPR* is the false positive rate.

In this experimentation, the AUC metric has been used to evaluate both binary and multi-class datasets. For this purpose, package *pROC* for R has been used. This framework contains a set of tools to visualize and analyze the ROC curves [76]. More precisely, the package documentation indicates that the AUC calculation for multi-class datasets is performed as defined by Hand and Till in [44].

Once the way of evaluating the results obtained has been set, it is necessary to present the statistical tests applied in this study. These tests allow to analyze if the results are statistically significant and, in this way, different conclusions can be reached. The tests used are:

- The Friedman test [33] allows to obtain a ranking of the different models. In this manner, it is possible to establish which model has a better performance from a general point of view.
- The Li post-hoc test [57] for the Friedman test is a non-parametric method. This test is applied to determine if there are significant differences between the models compared in this experimentation. To do so, it compares all the models with each other. In this study, the Li test is used because it is one of the best alternatives to find significant differences when the number of samples is not very large [37].

Lastly, it is necessary to present the hardware equipment on which the experimentation has been conducted. The objective is to make the experimentation as reproducible as possible. The experiments were performed in a cluster composed of 8 computers, with 2 CPUs (2.33 GHz) and 7 GB RAM each. Moreover, it is important to indicate that the models based on AEs (basic and ensemble) have been implemented in the Python language, using the Keras library [22]. Classification methods, that have been developed using the R language [71], are detailed in Subsection 4.1.1.

4.1.1. Classification Algorithm Framework

In this section, to verify the performance of the fusion of features, the data generated by ClEnDAE will be used by different classification algorithms. In this manner, the experimentation offers useful information about the behavior of the ensemble model according to data characteristics and the classification algorithm used. In this subsection, the classification algorithms considered in this study are presented, specifying their implementation and their main parameters.

Each classification algorithm has been obtained from a specific package of R [71]. Next, the details for each algorithm are

shown, including the parameterization used, which is a fundamental aspect for the reproducibility of the experiment.

- kNN algorithm: The *kknn* package incorporates the kNN algorithm employed in this study. Furthermore, it is important to indicate that the number of neighbors (parameter *k*) has been 5, since it is the recommended value in the literature [67].
- SVM algorithm: Package *e1071* has provided the SVM method. In this case, the algorithm has been executed with the default parameters that can be consulted in the package documentation [60].
- MLP algorithm: The execution of MLP has been performed using the packages *caret* and *RSNNS*. The parameters of this method are the default values specified in the documentation of the package [34, 13].
- C4.5 algorithm: Package *RWeka* provides method C4.5. The parameters used in the execution of the method are the default values specified in the documentation [48].

It is possible to see that the default values are used in all cases. The main reason is that this study focuses on verifying the performance of an ensemble model of DAEs over a set of traditional classification algorithms, without adjusting the parameters for each of the specific cases.

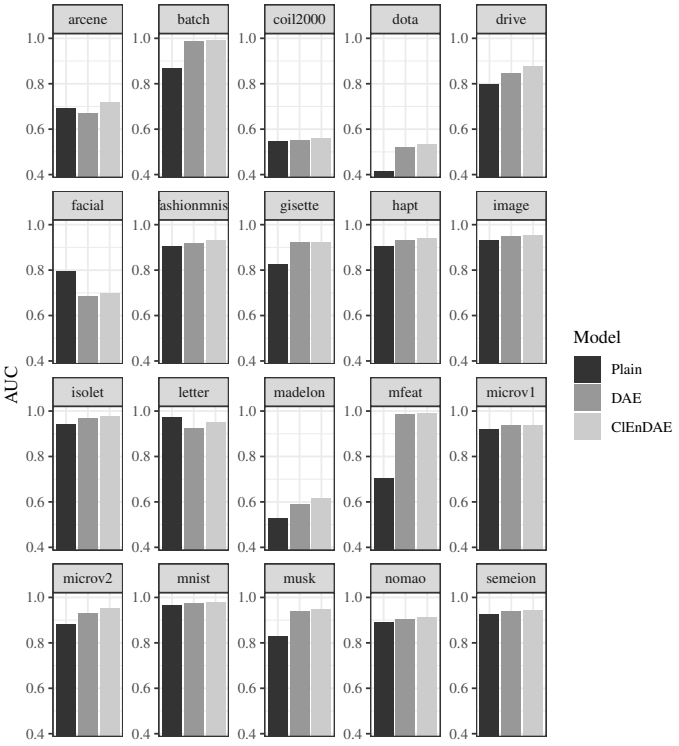


Figure 4: AUC results for kNN algorithm.

4.2. Classification Algorithms Analysis

The objective of this section is to analyze the performance of ClEnDAE with different classification methods. ClEnDAE has a parameter that allows you to select the desired classifier from four possibilities: kNN, SVM, MLP and C4.5. This makes it possible to study how feature fusion with the ClEnDAE model affects the predictive performance of these algorithms.

In this context, Subsections 4.2.1, 4.2.2, 4.2.3 and 4.2.4 describe the results achieved in the experimentation for algorithms kNN, SVM, MLP and C4.5, respectively. The structure of all the subsections is similar. First, the classification results (AUC) generated with the ClEnDAE model, the basic DAE and the raw data are presented in a figure. Subsequently, a ranking of the different models is shown and the results of the Li test are presented in order to determine whether the results are statically significant.

4.2.1. kNN

The results of the classification obtained for the kNN algorithm are presented in Figure 4. This image displays the results for each dataset obtained with the ClEnDAE model, the basic DAE and the original data. This representation allows for visual observation that the predictive performance generated with the ClEnDAE model are the best in most cases. In particular, 17 out of 20 cases the best performance is obtained with the ClEnDAE model and in 2 out of 20 it is obtained with the raw data. It is important to note that the ClEnDAE method improves upon the basic DAE model in all cases, except one in which they tie. In summary, in 85% of the cases the ClEnDAE model improves the results obtained through a basic DAE model and using the plain data.

These results determine that a clear improvement in predictive performance is obtained when the dimensionality is reduced using the ClEnDAE model. There are works that study how DAEs can obtain an improvement of the predictive performance with algorithms based on distances, in particular, kNN [65]. These methods based on distances between examples are affected by the high dimensionality of the input data, since in this context the distances are much less significant. This has a direct effect on the predictive performance produced by the IBL algorithms [14]. Incorporating DAEs to reduce the dimensionality through feature fusion allows to make the distances more significant. Likewise, the new features group the most relevant information of the input space. With all this, there is an improvement in the predictive performance of the classifier.

In this sense, the ClEnDAE model that internally trains a large number of basic DAE models allows to improve the results generated by an individual DAE. The fundamental reason is that each of the DAEs that compose the ensemble can learn certain features of the data and, thus, generalize a larger amount of relevant information than a model in an individualized way.

Finally, Table A.6 included in Appendix A.1 contains the specific results of the experimentation described above. The best results, which are obtained with the ClEnDAE method in most cases. This table confirms the conclusions reached from the previously analyzed figure.

4.2.2. SVM

In this subsection, classification results for the SVM algorithm are presented in Figure 5. The general trend shown by the kNN algorithm is maintained in this case. Generally, the predictive performance of the SVM algorithm improves by using the data generated by the ClEnDAE model. Reviewing the data in detail, the ClEnDAE model obtains the best results in 16 out of 20 cases and the model that uses the original data works best in 2 out of 20 cases. There is a dataset (*coil2000*) for which the three models generate similar results and another dataset (*mfeat*) for which the models that use DAE (ensemble and basic) have the same performance. Similar to the case of kNN, these data show that the ClEnDAE algorithm performs better with all datasets than the basic DAE model, except in two cases where they tie. In short, the ClEnDAE model produces best results in 80% of the cases (not counting the ties).

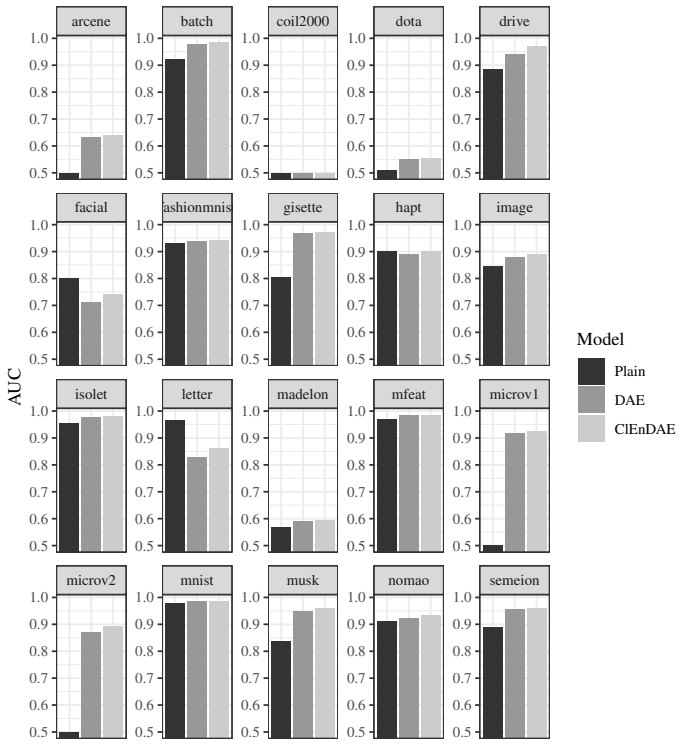


Figure 5: AUC results for SVM algorithm.

The predictive performance of the SVM algorithm when using ClEnDAE undergoes a significant improvement over the basic model and the original data. SVM is based on the maximization of the distances between instances of different classes [45]. However, these distances are equalized in spaces of high dimensionality. This implies loss of performance in this type of models and the need to apply methods that reduce the input space to mitigate the effects [10].

In this study, the proposal based on DAEs carries out a feature fusion that reduces the input space, making the distances more significant. Similarly, the model based on ensembles allows training several DAEs simultaneously, which can generate different feature spaces that prevent from discarding relevant information. Consequently, the graphs shown in this subsection confirm that the proposed model generates improvements in the behavior of the SVM algorithm.

Table A.7 (see Appendix A.2) shows the results of the different models after applying the SVM algorithm. These data allow us to verify that the ClEnDAE model produces the best perform for most of the datasets used.

4.2.3. MLP

Figure 6 represents the predictive performance of the different models. These data confirm the trend marked by the previous algorithms. The best results are obtained using the data generated by the ClEnDAE model, namely, feature fusion made with the proposed model has a positive effect on the MLP performance. A more detailed analysis of the results shows that the ClEnDAE model obtains the best performance in 16 out of 20 cases, the model based on the plain data works best in 2 out of 20 cases and the basic DAE mode does not generate the best results in any case. In addition, there is a dataset (*coil2000*) for which all models generate similar results and another dataset (*musk*) in which DAE-based models (basic and ensemble) produce the same value. This algorithm confirms the tendency in the two previous ones: the ClEnDAE model generates better results than the proposal based on basic DAE in all cases, except for two ties. In short, the ClEnDAE model obtains the best predictive performance in 80% of the analyzed cases (excluding ties).

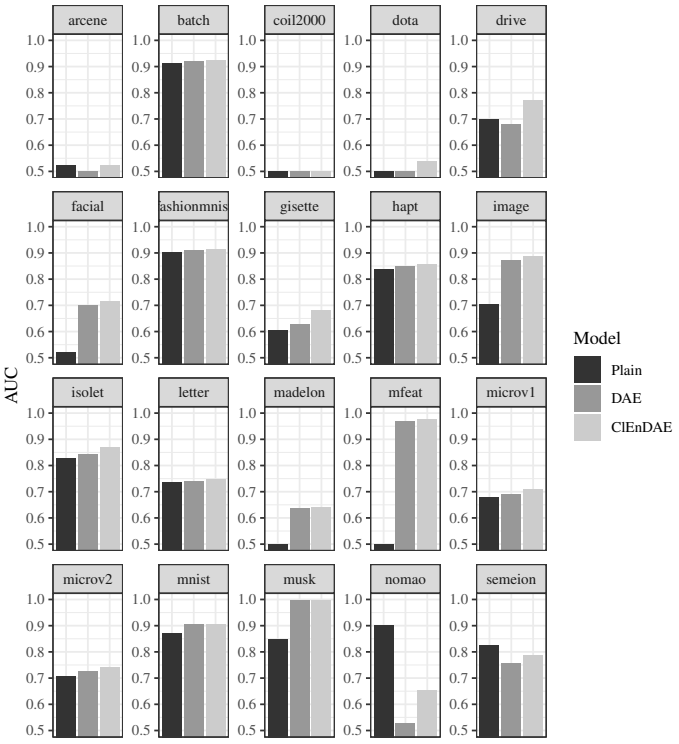


Figure 6: AUC results for MLP algorithm.

At this point, it is possible to state that the predictive performance of the MLP algorithm improves when using the ClEnDAE model to reduce the dimensionality of the original

feature space. More clearly, the MLP algorithm is a basic neural network whose fundamental objective is to classify a series of instances based on training examples [49]. The training process consists in training the network to progressively reduce the prediction error in training data. Nonetheless, in high dimensional spaces, the information of each class that the network learns is less meaningful, therefore, the predictive performance is negatively affected.

The ClEnDAE model proposed to reduce the input space is fundamentally based on ANNs. Because of that, the fusion of previously generated features can be seen as a series of previous layers that generate the input of another network. In the literature, there are classification models that incorporate the DL-based feature fusion process internally, although this phase is similar to the one performed in this study [40]. In both cases, the dimensionality reduction phase allows the generation of new features that group relevant information from the input data. In this way, the functioning of the classification algorithm, MLP, in this case, improves considerably.

To visualize the previous results minutely, Table A.8 is presented in Appendix A.2. The data represent the predictive performance of the different models using the MLP algorithm for each dataset. This confirms what has been described above, the ClEnDAE model generates the best results in most cases.

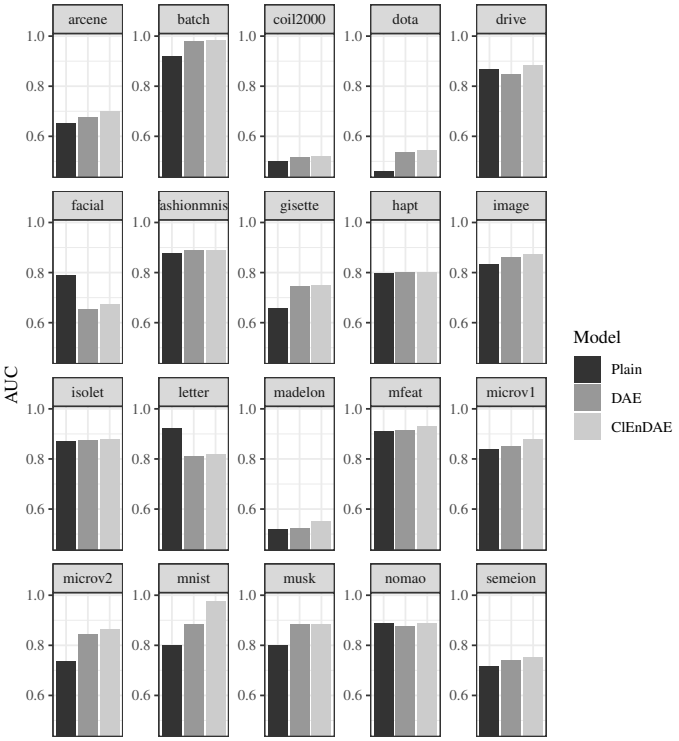


Figure 7: AUC results for C4.5 algorithm.

4.2.4. C4.5

The predictive performance data obtained for the C4.5 algorithm are presented in Figure 7. In general, the behavior in this case is very similar to that of the previous algorithms and confirms the trend marked by them. The classification made with C4.5 and the data generated by the ClEnDAE model produces the best results in most of the analyzed datasets. Specifically, the ClEnDAE model obtains the best results in 18 out of 20 cases, the model based on the plain data in 2 out of 20 and the basic DAE model does not produce the best performance in any case. Moreover, this analysis confirms that the ClEnDAE model improves the basic DAE for every case considered. In summary, the ClEnDAE model obtains the best performance in 90% of cases (excluding ties).

Thus, the performance of the C4.5 algorithm is clearly improved when feature fusion is applied through the ClEnDAE model. This fact allows us to intuit that the generation of new high-level features benefits the tree-based algorithms. Particularly, the C4.5 algorithm follows a training process where it analyzes the attributes that cause a separation of the instances in classes [70]. For this reason, the high dimensionality of the data complicates selecting the most significant features. In this circumstances, the reduction of dimensionality with ClEnDAE helps to mitigate these effects.

The proposed model generates new features that group the most relevant information of the input space. In this manner, algorithm C4.5 has a smaller and much more significant input space. This facilitates looking for the attributes that allow dividing the instances between the classes more effectively.

Finally, Table A.9 (see Appendix A.1) confirms the statements previously exposed. Different data show that the best results are obtained with ClEnDAE for most of the analyzed datasets.

4.3. Results Analysis

In the previous subsections, the results obtained with four traditional classifiers from data generated with the ClEnDAE dimensionality reduction model have been presented. Nevertheless, it is necessary to verify if these results are really significant. For this, different statistical tests will be used. These tests determine if there are actual differences between the models. The fact that these differences exist will support the results obtained and the conclusions reached.

The main objective of this phase is to corroborate if there are significant differences between the results generated with the proposed ClEnDAE model and those provided by the basic DAE and the plain dataset. The data seen in the Subsections 4.2.1, 4.2.2, 4.2.3, 4.2.4 intuitively show that the ClEnDAE model works best, now it is necessary to determine if this improvement can be supported statistically. In this sense, the first step is to apply the Friedman test [33]. This test allows to obtain an average ranking for each classifier according to the results obtained by the different analyzed models.

Table 2 presents rankings obtained for the four classification algorithms: kNN, SVM, MLP and C4.5. In all cases, the ClEnDAE model offers the best average performance, since it is the model with the best rank for the four algorithms. The model based on basic DAE is always in second place. Lastly, the model that uses the plain data is last, that is, the results generated using the original data are the lowest in general. The

Table 2: Average rankings of the different models (DAE, ClEnDAE and plain data) according to classification method.

kNN		**SVM**		**MLP**		**C4.5**	
AE Model	Ranking	AE Model	Ranking	AE Model	Ranking	AE Model	Ranking
ClEnDAE	1.150	ClEnDAE	1.150	ClEnDAE	1.150	ClEnDAE	1.100
DAE	2.100	DAE	2.150	DAE	2.200	DAE	2.200
Plain	2.750	Plain	2.700	Plain	2.650	Plain	2.700

rankings generated by the Friedman test support the conclusions reached in the previous subsections, which indicated that the ClEnDAE model reached the best predictive performance in most cases. Additionally, this statistical test confirms that there are significant differences.

Once the Friedman test has been applied, a second test must be used to confirm whether the differences presented in the rankings between the models considered are significant or not. For this reason, the Li test [57] is applied. This is a specific post-hoc test for the Friedman test that compares the different models present in the ranking. When used, the test offers a series of *p-values* that determine if there are differences between the models.

The data obtained after applying the Li test for the four classification algorithms can be seen in Table 3. This table shows that in all cases there are significant differences (values in bold) between ClEnDAE and the compared models. In general, all *p-values* are very low. Particularly, the ClEnDAE model always shows significant differences to the rest of the models, which confirms the improvements in performance discussed above. Regarding the disparities between the model with plain data and the basic DAE model, there are also significant differences in the four analyzed algorithms. This corroborates that the use of DAE for feature fusion allows to improve the performance of the classifiers, as has been described in previous studies [65].

Table 3: Li post-hoc Friedman test for classification algorithms by AE Model.

		Plain	DAE	ClEnDAE
kNN	Plain	-	-	-
	DAE	**3.983E-02**	-	-
	ClEnDAE	**4.375E-07**	**2.276E-03**	-
SVM	Plain	-	-	-
	DAE	**8.199E-02**	-	-
	ClEnDAE	**1.036E-06**	**1.702E-03**	-
MLP	Plain	-	-	-
	DAE	1.547E-01	-	-
	ClEnDAE	**2.486E-04**	**1.062E-03**	-
C4.5	Plain		-	-
	DAE	1.138E-01		-
	ClEnDAE	**4.740E-07**	**5.687E-04**	

Besides, Figure 8 represents the critical distances for the different models considered. This type of graph shows whether or not there are significant differences between the different models. Specifically, if there are no significant differences between two models, they are connected by a horizontal line in the graph. In Figure 8, it is possible to see the significant differences between ClEnDAE and the rest of the methods used for all classifiers.

In conclusion, the statistical tests conducted in this Subsection confirm the statements made in Subsections 4.2.1, 4.2.2, 4.2.3, 4.2.4. The ClEnDAE model provides a clear improvement in the predictive performance of the four classification algorithms analyzed in this study: kNN, SVM, MLP and C4.5. In this sense, the basic DAE model also produces better results than the model based on plain data. Nonetheless, the proposed ClEnDAE model improves both in all classification algorithms.

4.4. ClEnDAE vs classical feature extraction techniques

ClEnDAE is a feature fusion algorithm that aims to mitigate the negative effects of the high dimensionality of the data. In previous sections, the effectiveness of the model proposed in this work has been compared with plain data and another model based on DAEs. However, it is necessary to establish a comparison of the ClEnDAE model with other traditional methods of dimensionality reduction. In this circumstances, PCA, LDA, ISOMAP and LLE have been selected, since they are four of the best known algorithms to treat the high dimensionality of the data. To conduct the comparison, the ClEnDAE, PCA, LDA, ISOMAP and LLE methods have been used on the same datasets. The number of features generated by these algorithms will be the same in every case, the 75% of the number of original features. It is essential that this value is shared in order to establish a reliable comparison and draw appropriate conclusions. Lastly, the data generated by the ClEnDAE, PCA, LDA, ISOMAP and LLE methods will be used to classify with the algorithms: kNN, SVM, MLP and C4.5. The predictive performance obtained with the different subsets provides the possibility to determine which model generates the best space of reduced dimensionality.

Classification results for the different datasets are presented in Figures 9, 10, 11 and 12. There is an image for each classifier: kNN, SVM, MLP and C4.5, respectively. Moreover, each classifier represents the predictive performance generated from the different models: the proposal of this study, ClEnDAE, and the traditional dimensionality reduction approaches, PCA, LDA, ISOMAP and LLE. In every case, there is a bar chart for each dataset used.

The observation of the results presented in Figures 9, 10, 11 and 12 allows to make the following statements:

- kNN: The best results for 16 out of 20 datasets are obtained with the ClEnDAE model. The LDA method only reaches the best performance in one case (*musk*), the PCA algorithm in none, ISOMAP in one case (*microv1*) and

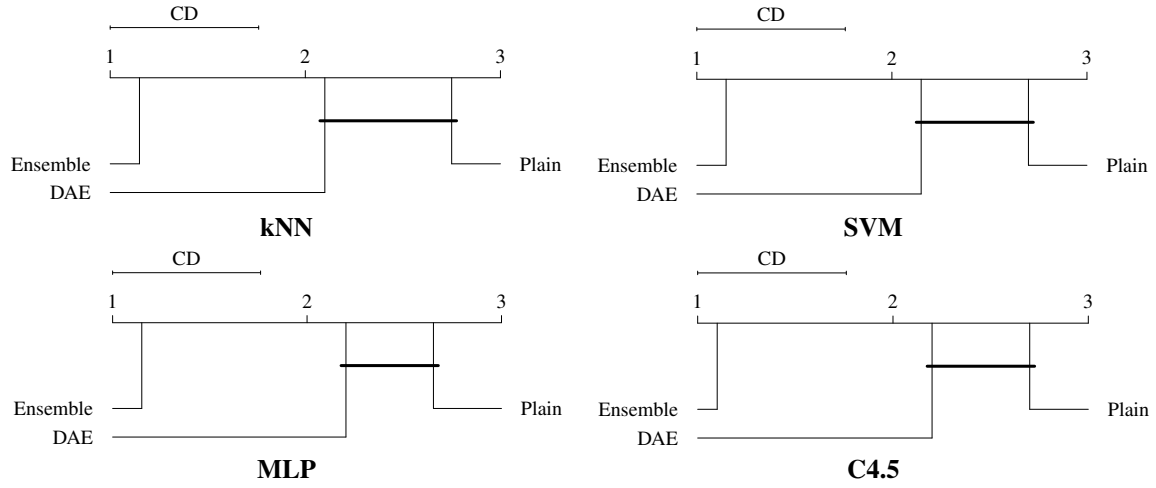


Figure 8: Critical distance between ClEnDAE, DAE and plain models for kNN, SVM, MLP and C4.5.

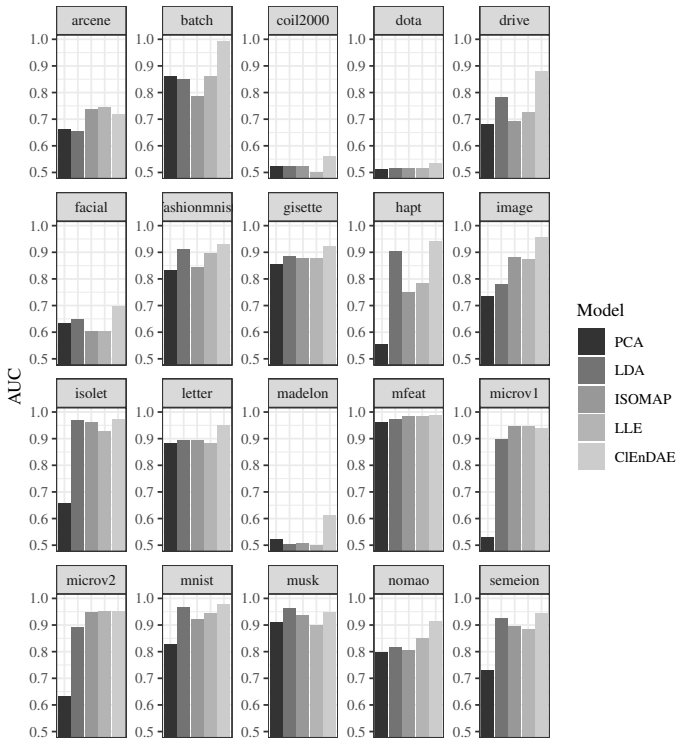


Figure 9: AUC results for kNN algorithm.

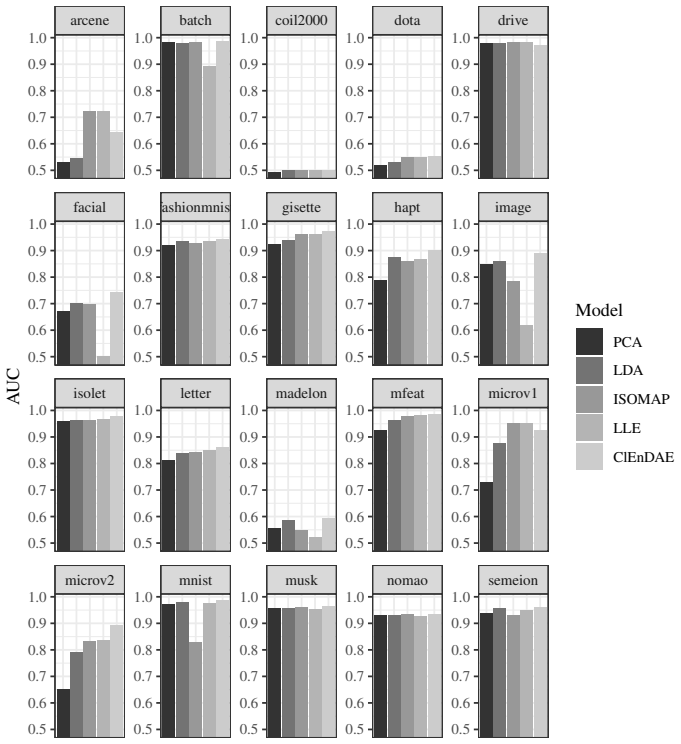


Figure 10: AUC results for SVM algorithm.

LLE in another dataset (*arcene*). There is a case in which both ClEnDAE and LLE generate the best performance. This means that the ClEnDAE representations work best in 80% of the analyzed cases (not counting the ties).

- SVM: The ClEnDAE method achieves the best predictive performance in 16 out of 20 cases, the LLE algorithm in 3 out of 20 (in *microv1* case tied with ISOMAP) and PCA and LDA do not provide the best performance in any case. There is a dataset (*coil2000*) in which the ClEnDAE, LDA, ISOMAP and LLE models produce the same result. In other words, the proposed ClEnDAE model generates better performance in 80% of cases (excluding ties).

- MLP: The ClEnDAE algorithm works best in 13 out of 20 cases. For its part, the LDA method generates better performance in 2 out of 20 datasets and PCA, ISOMAP and LLE in one case each. For one of the datasets (*coil2000*) the results of the three methods are equivalent and for another (*microv1*) the performance of LLE and ISOMAP is equivalent. The model proposed in this study produces the best results in 65% of cases (without counting ties).

- C4.5: The ClEnDAE model obtains the best predictive

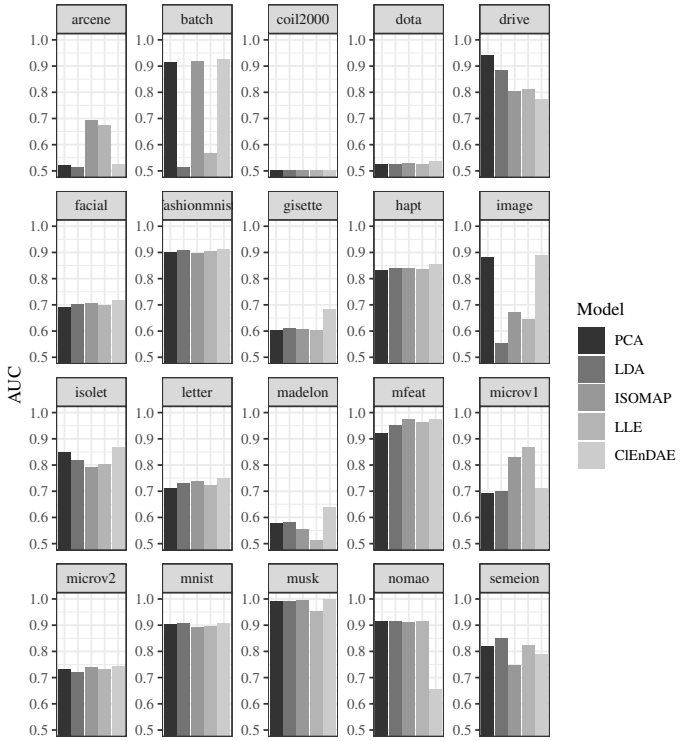


Figure 11: AUC results for MLP algorithm.

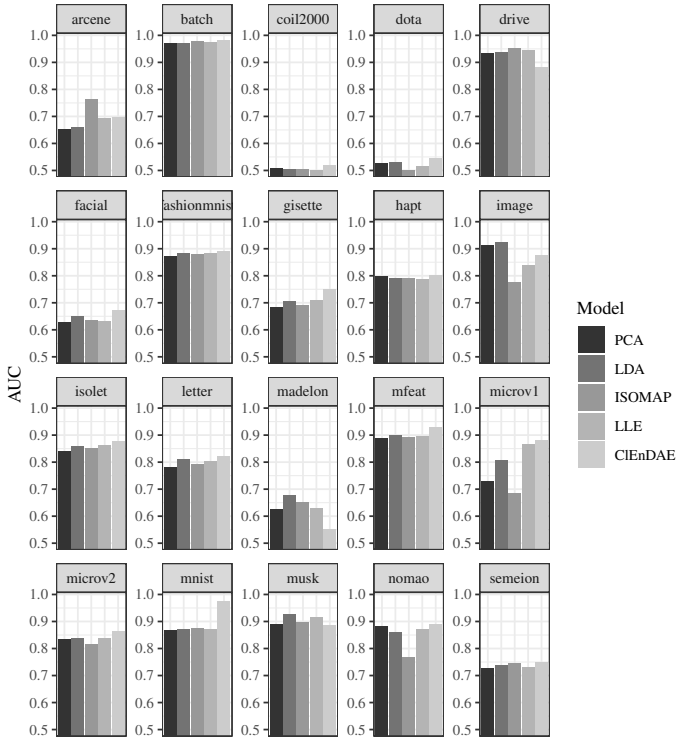


Figure 12: AUC results for C4.5 algorithm.

performance in 16 out of 20 cases. For its part, the LDA algorithm works best in 3 out of 20 cases and ISOMAP in another dataset. Last, PCA and LLE models do not generate the best results in any case. This means that the ClEnDAE method improves upon the rest of the methods in 80% of cases.

The results described above mark a common trend for all analyzed classification algorithms: the ClEnDAE model works better than PCA, LDA, ISOMAP and LLE, in general. This means the feature fusion performed with ClEnDAE builds a new feature space that provides more relevant and useful information than four of the most used models in dimensionality reduction. This pattern can be confirmed in Tables A.10, A.11, A.12 and A.13 corresponding to the algorithms kNN, SVM, MLP and C4.5, respectively. The tables highlight the best results in bold, so it is possible to see that, as a rule, ClEnDAE produces the best performance.

However, the previous results must be duly supported by statistical tests that establish a solid basis for the conclusions reached. For this, it is necessary to determine if there are significant differences between the studied proposals.

The first step consists in applying the Friedman test [33]. The different rankings for each classification algorithm are presented in Table 4. In this manner, a vision of the global operation of the different dimensionality reduction methods can be visually obtained.

The four rankings provided in Table 4 corroborate the above conclusions. The ClEnDAE model generates the best predictive performance for all classification algorithms. The LDA, LLE, ISOMAP and PCA methods perform worse for all the analyzed cases, being PCA the one that produces worst results of these four. Moreover, LLE, LDA and ISOMAP algorithms generally have equivalent performance. This is a first step to confirm that the ClEnDAE model offers a significant improvement over PCA, LDA, ISOMAP and LLE.

The second step is to apply the Li post-hoc tests [57] for the Friedman test. This allows to compare the different methods to establish whether there are significant differences between them.

In this sense, Table 5 shows the results obtained after applying the Li test. This table represents, for each pair of models, the *p-value* associated with its statistical difference. It can be observed that the ClEnDAE model presents significant differences with the LDA, PCA, ISOMAP and LLE methods in the four analyzed classification algorithms, since the *p-values* are notably low. This means there is a statistical confirmation of the improvement provided by the ClEnDAE method with respect to PCA, LDA, ISOMAP and LLE. This supports the statements that have been made after presenting the experimental data.

Finally, Figure 13, representing the critical distances for the different considered models, shows that ClEnDAE presents significant differences with all of them.

In conclusion, the objective of this section is to establish a comparison between our proposal, ClEnDAE, and four traditional dimensionality reduction algorithms: PCA, LDA, ISOMAP and LLE. The results of the experimentation have shown that the proposed method tipically works better than the four traditional models. These data have been adequately supported by several statistical tests that show that there are significant

Table 4: Average rankings considering ClEnDAE, PCA, LDA, ISOMAP and LLE by classification method.

kNN		**SVM**		**MLP**		**C4.5**	
Architecture	Ranking	Architecture	Ranking	Architecture	Ranking	Architecture	Ranking
ClEnDAE	1.275	ClEnDAE	1.475	ClEnDAE	1.875	ClEnDAE	1.750
LDA	2.825	LLE	3.000	ISOMAP	3.000	LDA	2.700
LLE	3.150	LDA	3.025	LDA	3.100	LLE	3.150
ISOMAP	3.250	ISOMAP	3.125	LLE	3.375	ISOMAP	3.350
PCA	4.500	PCA	4.375	PCA	3.650	PCA	4.050

Table 5: Li post-hoc Friedman test for dimensionality reduction methods by classification algorithm.

		ClEnDAE	PCA	LDA	ISOMAP	LLE
kNN	ClEnDAE	-	**1.118E-09**	**3.867E-03**	**3.907E-04**	**5.893E-04**
	PCA	-	-	-	-	-
	LDA	-	**2.019E-03**	-	4.668E-01	5.532E-01
	ISOMAP	-	**1.769E-02**	-	-	-
	LLE	-	**1.153E-02**	-	8.415E-01	-
SVM	ClEnDAE	-	**6.6631E-08**	**6.436E-03**	**4.824E-03**	**6.436E-03**
	PCA	-	-	-	-	-
	LDA	-	**1.188E-02**	-	8.708E-01	-
	ISOMAP	-	**1.769E-02**	-	-	-
	LLE	-	**1.188E-02**	9.691E-01	8.685E-01	-
MLP	ClEnDAE	-	**3.846E-03**	**4.683E-02**	**6.001E-02**	**1.343E-02**
	PCA	-	-	-	-	-
	LDA	-	4.099E-01	-	-	6.642E-01
	ISOMAP	-	3.497E-01	8.415E-01	-	5.779E-01
	LLE	-	6.642E-01	-	-	-
C4.5	ClEnDAE	-	**4.225E-05**	**1.115E-01**	**6.852E-03**	**1.693E-02**
	PCA	-	-	-	-	-
	LDA	-	**1.724E-02**	-	2.358E-01	3.995E-01
	ISOMAP	-	2.225E-01	-	-	-
	LLE	-	**1.169E-01**	-	6.892E-01	-

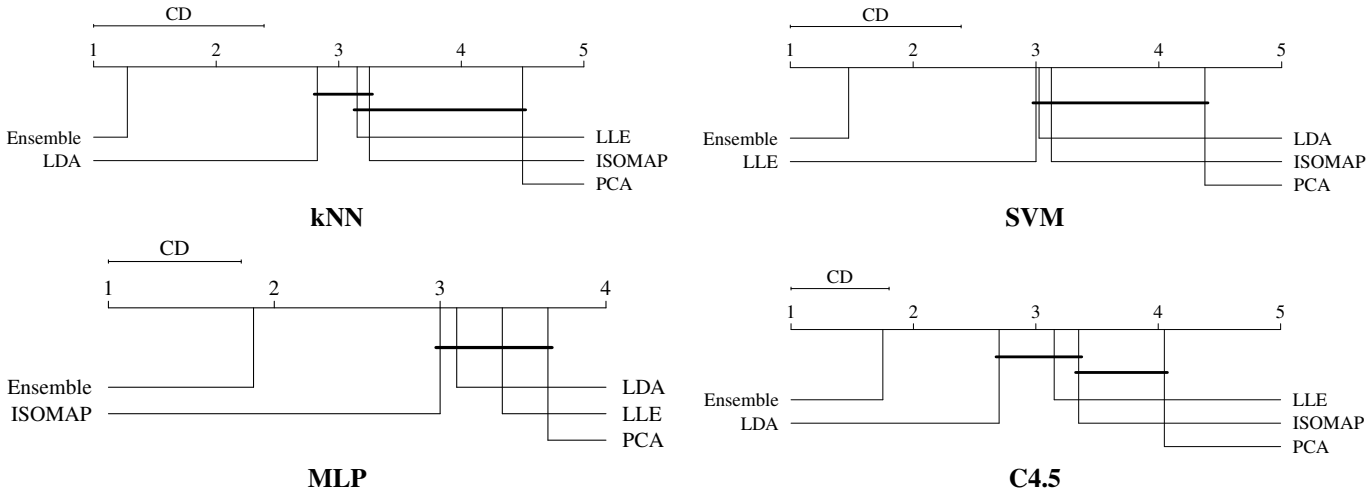


Figure 13: Critical distance between dimensionality reduction methods for kNN, SVM, MLP and C4.5.

differences between them. In short, the predictive performance of the four classifiers used with the ClEnDAE method is much higher. This suggests that the feature fusion process performed by our proposal generates higher quality and more relevant characteristics than the other traditional proposals. This experimentation allows to establish a basis to consider the ClEnDAE

model as an option to take into account to mitigate the effects of high dimensionality.

5. Concluding remarks

In this study, a model to deal with the task of dimensionality reduction has been proposed. This task is a challenge in machine learning, due to the negative effects that the high dimensionality of the data produces in many circumstances. More concretely, this study is based on the effects of high dimensional data in different classification methodologies.

Specifically, the ClEnDAE algorithm has been proposed. The fundamental objective of this method is to mitigate the effects of the high dimensionality of the data. Thus, ClEnDAE carries out a feature fusion that reduces the dimensionality of the input space. The new features generated by our method add the most relevant information obtained from the original data and discard redundant or meaningless information. The main reason that has led to develop this algorithm is the good results of AEs to deal with the task of feature fusion [61, 20, 67, 95, 1, 92], in particular, there are works that show that the models based on DAEs can improve upon traditional models [65]. The ClEnDAE method is based on two fundamental pillars: DAEs and ensembles. In this sense, the use of DAEs to address the reduction of dimensionality joins the advantages of the ensembles. Broadly, a model based on ensembles allows to train simultaneously several basic models with partitions of the input data. This helps each model generalize relevant information of the different partitions of the data. Once the information generated by the different models is merged, the result has more relevance and usefulness than when using a single basic model.

In order to determine the effectiveness of the ClEnDAE model, an exhaustive study has been developed. Fundamentally, the experimentation consists in analyzing the predictive performance of four traditional classifiers: kNN, SVM, MLP and C4.5, after performing feature and information fusion with the proposed model. This performance is compared with that obtained after using a basic DAE model and after using original data. Furthermore, a comparison with four traditional dimensionality reduction algorithms (PCA, LDA, ISOMAP and LLE) is conducted.

The first part of the experimentation has shown that the ClEnDAE model generates a better predictive performance than the basic DAE model and the plain data model for the four analyzed classifiers. The ClEnDAE proposal obtains the best results in more than 72% of the cases. Similarly, these conclusions have been duly confirmed by several statistical tests.

Next, the second part of the experimentation has analyzed the performance of ClEnDAE against PCA, LDA, ISOMAP and LLE. This part has shown that ClEnDAE works better than four of the most common algorithms for the task of dimensionality reduction. Concretely, the ClEnDAE model obtains the best results in more than 80% of the cases for kNN, SVM and C4.5 methods and more than 65% for MLP. Statistical tests have confirmed the differences between the different models as well.

Finally, this experimentation has shown that the ClEnDAE model is able to mitigate the effects caused by the high dimensionality of the data, improving the predictive performance of different classifiers corresponding to traditional methodologies. Likewise, the provided performance is better than some of the most commonly used traditional algorithms: LDA, PCA, ISOMAP and LLE.

In conclusion, ClEnDAE is a method of feature fusion based on DAEs that deals with the problem of high dimensionality. This paper describes the developed method and an exhaustive experimentation to verify its proper operation. The conclusions provided open new lines of future work where other models of AEs, for instance, robust or contractive AEs [20], are used to generate models similar to ClEnDAE. The ClEnDAE method could also be applied to solve specific problems, adapting its structure and functioning to make it more effective.

Acknowledgment

The work of F. Pulgar was supported by the Spanish Ministry of Education under the FPU National Program (Ref. FPU16/00324). This work was partially supported by the Spanish Ministry of Science and Technology under project TIN2015-68454-R.

References

[1] Manish Agnihotri, Aditya Rathod, Daksh Thapar, Gaurav Jaswal, Kamlesh Tiwari, and Aditya Nigam. Learning domain specific features using convolutional autoencoder: A vein authentication case study using siamese triplet loss network. 03 2019.

[2] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-Based Learning Algorithms. *Machine Learning*, 6(1):37–66, 1991.

[3] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3):307–318, Feb 2009.

[4] N. S. Altman. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, 46(3):175–185, aug 1992.

[5] Ricardo F Alvear-Sandoval, José L Sancho-Gómez, and Aníbal R Figueiras-Vidal. On improving cnns performance: The case of mnist. *Information Fusion*, 52:106–109, 2019.

[6] Christopher G. Atkeson, Andrew W. Moorey, Stefan Schaalz, Andrew W Moore, and Stefan Schaal. Locally Weighted Learning. *Artificial Intelligence*, 11:11–73, 1997.

[7] K. Bache and M. Lichman. UCI Machine Learning Repository, 2013.

[8] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1):105–139, Jul 1999.

[9] R Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[10] R.E. Bellman. *Adaptive control processes: A guided tour*. Princeton University Press, 1961.

[11] S. Bengio and Y. Bengio. Taking on the curse of dimensionality in joint distributions using neural networks. *IEEE Transactions on Neural Networks*, 11(3):550–557, May 2000.

[12] Yoshua Bengio. Deep Learning of Representations: Looking Forward. In *International Conference on Statistical Language and Speech Processing*, pages 1–37. 2013.

[13] Christoph Bergmeir and José M. Benítez. Neural networks in R using the stuttgart neural network simulator: RSNNS. *Journal of Statistical Software*, 46(7):1–26, 2012.

[14] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When Is "Nearest Neighbor" Meaningful? In *International conference on database theory*, pages 217–235. 1999.

[15] H. Bonab and F. Can. Less is more: A comprehensive framework for the number of components of ensemble classifiers. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2735–2745, Sep. 2019.

[16] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[17] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[18] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[19] Augustin Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

[20] David Charte, Francisco Charte, Salvador García, María J. del Jesus, and Francisco Herrera. A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines. *Information Fusion*, 44:78–96, 2018.

[21] C.L. Philip Chen and Chun-Yang Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275:314 – 347, 2014.

[22] François Chollet et al. Keras. `https://keras.io`, 2015.

[23] Ronald Cole and Mark Fanty. Spoken letter recognition. In *Proceedings of the workshop on Speech and Natural Language*, pages 385–390, 1990.

[24] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[25] Manoranjan Dash and Huan Liu. Feature selection for classification. *Intelligent data analysis*, 1(3):131–156, 1997.

[26] Lawrence Davis. *Handbook of genetic algorithms*. CUMINCAD, 1991.

[27] Li Deng. Deep Learning: Methods and Applications. *Foundations and Trends in Signal Processing*, 7(3-4):197–387, 2014.

[28] Yanling Du, Wei Song, Qi He, Dongmei Huang, Antonio Liotta, and Chen Su. Deep learning with multi-scale feature fusion in remote sensing for automatic oceanic eddy detection. *Information Fusion*, 49:89 – 99, 2019.

[29] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.

[30] R O Duda, P E Hart, and D G Stork. *Pattern Classification*. Wiley New York, 1973.

[31] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[32] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, pages 148–156, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[33] M Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[34] Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, and Can Candan. *caret: Classification and Regression Training*, 2016. R package version 6.0-68.

[35] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.

[36] Salvador García, Julián Luengo, and Francisco Herrera. *Feature Selection*, pages 163–193. Springer International Publishing, Cham, 2015.

[37] Salvador García, Alberto Fernández, Julián Luengo, and Francisco Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044 – 2064, 2010. Special Issue on Intelligent Distributed Information Systems.

[38] Diego García-Gil, Sergio Ramírez-Gallego, Salvador García, and Francisco Herrera. *On the Use of Random Discretization and Dimensionality Reduction in Ensembles for Big Data*, pages 15–26. 06 2018.

[39] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[40] Ian Goodfellow, Y Bengio, and A Courville. *Deep Learning*. The MIT Press, 2016.

[41] Isabelle Guyon and Andre Elisseeff. *An Introduction to Feature Extraction*, pages 1–25. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[42] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result Analysis of the NIPS 2003 Feature Selection Challenge. In *Proceedings of Neural Information Processing Systems*, volume 4, pages 545–552, 2004.

[43] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In *Advances in neural information processing systems*, pages 545–552, 2005.

[44] David J. Hand and Robert J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45(2):171–186, Nov 2001.

[45] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[46] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926, 2010.

[47] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, 2006.

[48] Kurt Hornik, Christian Buchta, and Achim Zeileis. Open-source machine learning: R meets weka. *Computational Statistics*, 24(2):225–232, May 2009.

[49] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[50] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

[51] Gordon F. Hughes. On the Mean Accuracy of Statistical Pattern Recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63, 1968.

[52] Sotiris B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31:249–268, 2007.

[53] Nikolaos Kouiroukidis and Georgios Evangelidis. The Effects of Dimensionality Curse in High Dimensional kNN Search. In *Proceedings of the 15th Panhellenic Conference on Informatics*, pages 41–45. IEEE, sep 2011.

[54] Bartosz Krawczyk, Michał Woźniak, and Gerald Schaefer. Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing*, 14:554 – 562, 2014.

[55] Alex Krizhevsky, Ilya Sutskever, and Hinton Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[56] Y. Lecun, L??on Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[57] Jianjun (David) Li. A two-step rejection procedure for testing multiple hypotheses. *Journal of Statistical Planning and Inference*, 138(6):1521 – 1527, 2008.

[58] Guangfeng Lin, Hong Zhu, Xiaobing Kang, Caixia Fan, and Erhu Zhang. Feature structure fusion and its application. *Information Fusion*, 20:146 – 154, 2014.

[59] Utthara Gosa Mangai, Suranjana Samanta, Sukhendu Das, and Pinaki Roy Chowdhury. A survey of decision fusion and feature fusion strategies for pattern classification. *IETE Technical review*, 27(4):293–307, 2010.

[60] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2015. R package version 1.6-7.

[61] A. Moussavi-Khalkhali, M. Jamshidi, and S. Wijemanne. Feature fusion for denoising and sparse autoencoders: Application to neuroimaging data. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 605–610, Los Alamitos, CA, USA, dec 2016. IEEE Computer Society.

[62] Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

[63] Karl Pearson. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, nov 1901.

[64] Zhendong Peng, Xi Xiao, Guangwu Hu, Arun Kumar Sangaiah, Mo-

hammed Atiquzzaman, and Shutao Xia. Abfl: An autoencoder based practical approach for software fault localization. *Information Sciences*, 510:108 – 121, 2020.

[65] Francisco J Pulgar, Francisco Charte, Antonio J Rivera, and María J del Jesus. A first approach to face dimensionality reduction through denoising autoencoders. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 439–447. Springer, 2018.

[66] Francisco J. Pulgar, Francisco Charte, Antonio J. Rivera, and María J. del Jesus. Choosing the proper autoencoder for feature fusion based on data complexity and classifiers: Analysis, tips and guidelines. *Information Fusion*, 54:44 – 60, 2020.

[67] Francisco J. Pulgar, Francisco Charte, Antonio J. Rivera, and María J. Del Jesus. AEkNN: An AutoEncoder kNN-Based Classifier With Built-in Dimensionality Reduction. *International Journal of Computational Intelligence Systems*, 12:436–452, 2018/11.

[68] Y. Qi, Y. Wang, X. Zheng, and Z. Wu. Robust feature learning by stacked autoencoder with maximum correntropy criterion. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6716–6720, 2014.

[69] J. R. Quinlan. Bagging, boosting, and c4.s. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 1*, AAAI'96, pages 725–730. AAAI Press, 1996.

[70] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[71] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016.

[72] Jorge-L Reyes-Ortiz, Luca Oneto, Albert Samà, Xavier Parra, and Davide Anguita. Transition-Aware Human Activity Recognition Using Smartphones. *Neurocomputing*, 171:754–767, jan 2016.

[73] Salah Rifai and Xavier Muller. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *Proceedings of the 28th International Conference on Machine Learning*, volume 85, pages 833–840, 2011.

[74] A. J. Rivera, M. D. Pérez-Godoy, F Charte, F. J. Pulgar, and M. J. del Jesus. An ensemble strategy for forecasting the extra-virgin olive oil price in spain. In *International work-conference on Time Series (ITISE 2015)*, pages 506–516, 2015.

[75] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951.

[76] Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez, and Markus Müller. proc: an open-source package for r and s+ to analyze and compare roc curves. *BMC Bioinformatics*, 12(1):77, Mar 2011.

[77] Juan José Rodriguez, Ludmila I Kuncheva, and Carlos J Alonso. Rotation forest: A new classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1619–1630, 2006.

[78] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[79] Robert J Schalkoff. *Artificial neural networks*, volume 1. McGraw-Hill New York, 1997.

[80] Xinmin Tao, Qing Li, Wenjie Guo, Chao Ren, Chenxi Li, Rui Liu, and Junrong Zou. Self-adaptive cost weights-based support vector machine cost-sensitive ensemble for imbalanced data classification. *Information Sciences*, 487:31 – 56, 2019.

[81] J B Tenenbaum. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, dec 2000.

[82] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[83] L J P Van Der Maaten, E O Postma, and H J Van Den Herik. Dimensionality Reduction: A Comparative Review. *Journal of Machine Learning Research*, 10:1–41, 2009.

[84] Alexander Vergara, Shankar Vembu, Tuba Ayhan, Margaret A. Ryan, Margie L. Homer, and Ramón Huerta. Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical*, 166:320–329, may 2012.

[85] M. Verleysen, D. Francois, G. Simon, and V. Wertz. On the effects of dimensionality on data analysis with neural networks. In José Mira and José R. Álvarez, editors, *Artificial Neural Nets Problem Solving Methods*, pages 105–112, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[86] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103. ACM, 2008.

[87] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010.

[88] Wei Wang, Yan Huang, Yizhou Wang, and Liang Wang. Generalized autoencoder: A neural network framework for dimensionality reduction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 496–503, 2014.

[89] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[90] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241 – 259, 1992.

[91] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[92] Xiaolu Xu, Hong Gu, Yang Wang, Jia Wang, and Pan Qin. Autoencoder based feature selection method for classification of anticancer drug response. *Frontiers in Genetics*, 10:233, 2019.

[93] Hua Yu and Jie Yang. A direct LDA algorithm for high-dimensional data-with application to face recognition. *Pattern Recognition*, 34(10):2067–2070, 2001.

[94] L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(1):28–44, Jan 1973.

[95] Changfan Zhang, Xiang Cheng, Jianhua Liu, Jing He, and Guangwei Liu. Deep sparse autoencoder for feature extraction and diagnosis of locomotive adhesion status. *Journal of Control Science and Engineering*, 2018:1–9, 07 2018.

Appendix A. Result tables

This appendix contains the tables of results associated with the experimentation presented in Section 4. In particular, Appendix A.1 includes the results tables of Subsection 4.2, and Appendix A.2 presents the data associated with the experimentation described in Subsection 4.4.

Appendix A.1. Classification Algorithms Analysis

In this appendix, Tables A.6, A.7, A.8 and A.9 are provided. The data describe the results achieved in the experimentation for the algorithms kNN, SVM, MLP and C4.5, respectively. The structure of all tables is similar. The classification results (AUC) generated with the ClEnDAE model, the basic DAE and the raw data are presented in a figure, and the best results are highlighted in bold for each dataset.

Table A.6: kNN results for plain data, DAE and ClEnDAE models (AUC).

Dataset	Plain Data	DAE	ClEnDAE
arcene	0.693	0.672	**0.719**
batch	0.870	0.990	**0.992**
coil2000	0.546	0.554	**0.562**
dota	0.416	0.522	**0.534**
drive	0.800	0.846	**0.879**
facial	**0.795**	0.685	0.698
fashionmnist	0.906	0.920	**0.929**
gisette	0.824	**0.923**	**0.923**
hapt	0.903	0.931	**0.942**
image	0.929	0.950	**0.954**
isolet	0.943	0.968	**0.975**
letter	**0.973**	0.923	0.951
madelon	0.526	0.591	**0.614**
mfeat	0.703	0.984	**0.988**
microv1	0.920	0.939	**0.939**
microv2	0.883	0.932	**0.951**
mnist	0.965	0.975	**0.978**
musk	0.829	0.941	**0.947**
nomao	0.891	0.904	**0.914**
semeion	0.927	0.940	**0.945**

Table A.7: SVM results for plain data, DAE and ClEnDAE models (AUC).

Dataset	Plain Data	DAE	ClEnDAE
arcene	0.500	0.632	**0.642**
batch	0.923	0.980	**0.985**
coil2000	**0.500**	**0.500**	**0.500**
dota	0.509	0.550	**0.553**
drive	0.885	0.941	**0.972**
facial	**0.802**	0.713	0.742
fashionmnist	0.932	0.940	**0.943**
gisette	0.803	0.969	**0.972**
hapt	0.900	0.889	**0.902**
image	0.846	0.877	**0.891**
isolet	0.954	0.975	**0.980**
letter	**0.966**	0.828	0.861
madelon	0.569	0.591	**0.594**
mfeat	0.970	**0.985**	**0.985**
microv1	0.500	0.918	**0.925**
microv2	0.500	0.871	**0.893**
mnist	0.981	0.985	**0.986**
musk	0.838	0.948	**0.962**
nomao	0.914	0.924	**0.935**
semeion	0.891	0.957	**0.959**

Table A.8: MLP results for plain data, DAE and ClEnDAE models (AUC).

Dataset	Plain Data	DAE	ClEnDAE
arcene	0.522	0.500	**0.525**
batch	0.914	0.920	**0.926**
coil2000	**0.500**	**0.500**	**0.500**
dota	0.500	0.500	**0.538**
drive	0.698	0.680	**0.772**
facial	0.520	0.701	**0.716**
fashionmnist	0.901	0.911	**0.914**
gisette	0.605	0.627	**0.682**
hapt	0.839	0.850	**0.857**
image	0.703	0.872	**0.888**
isolet	0.827	0.842	**0.869**
letter	0.734	0.740	**0.748**
madelon	0.500	0.637	**0.641**
mfeat	0.500	0.968	**0.975**
microv1	0.678	0.690	**0.711**
microv2	0.708	0.725	**0.742**
mnist	0.872	0.905	**0.906**
musk	0.850	**0.999**	**0.999**
nomao	**0.901**	0.530	0.653
semeion	**0.827**	0.756	0.789

Table A.9: C4.5 results for plain data, DAE and ClEnDAE models (AUC).

Dataset	Plain Data	DAE	ClEnDAE
arcene	0.655	0.678	**0.699**
batch	0.921	0.982	**0.984**
coil2000	0.503	0.516	**0.521**
dota	0.462	0.539	**0.546**
drive	0.869	0.850	**0.884**
facial	**0.789**	0.655	0.672
fashionmnist	0.877	0.888	**0.889**
gisette	0.657	0.743	**0.749**
hapt	0.798	0.799	**0.802**
image	0.832	0.859	**0.874**
isolet	0.870	0.874	**0.878**
letter	**0.922**	0.811	0.821
madelon	0.520	0.524	**0.553**
mfeat	0.911	0.913	**0.931**
microv1	0.838	0.849	**0.880**
microv2	0.736	0.845	**0.863**
mnist	0.965	0.975	**0.977**
musk	0.800	0.883	**0.886**
nomao	0.886	0.877	**0.889**
semeion	0.716	0.739	**0.751**

Tables A.6, A.7, A.8 and A.9 allow us to observe that the best performance is obtained with the ClEnDAE method in most cases. These results are discussed in Subsection 4.2 in more detail.

Appendix A.2. ClEnDAE vs classical feature extraction techniques

In this appendix, the classification results for the different datasets are gathered in Tables A.10, A.11, A.12 and A.13. There is a table for each classifier: kNN, SVM, MLP and C4.5, respectively. In addition, each one presents the predictive performance generated from the following models: the proposal of this study, ClEnDAE, and the traditional dimensionality reduction approaches, PCA, LDA, ISOMAP and LLE.

These results have been analyzed in Section 4.2. In general, it is possible to see that the ClEnDAE algorithm obtains the best results in almost every case, considering the four classifiers.

Table A.10: kNN classification results of ClEnDAE, PCA, LDA, ISOMAP and LLE for test data (AUC).

Dataset	PCA	LDA	ISOMAP	LLE	ClEnDAE
arcene	0.661	0.654	0.738	**0.747**	0.719
batch	0.861	0.852	0.786	0.862	**0.992**
coil2000	0.523	0.525	0.525	0.502	**0.562**
dota	0.513	0.517	0.516	0.516	**0.534**
drive	0.683	0.782	0.693	0.726	**0.879**
facial	0.631	0.648	0.601	0.602	**0.698**
fashionmnist	0.831	0.911	0.842	0.896	**0.929**
gisette	0.855	0.883	0.877	0.878	**0.923**
hapt	0.553	0.903	0.750	0.781	**0.942**
image	0.734	0.779	0.882	0.874	**0.954**
isolet	0.659	0.971	0.962	0.927	**0.975**
letter	0.882	0.893	0.894	0.883	**0.951**
madelon	0.523	0.505	0.506	0.501	**0.614**
mfeat	0.963	0.972	0.985	0.986	**0.988**
microv1	0.532	0.897	**0.947**	0.946	0.939
microv2	0.632	0.891	0.948	**0.951**	**0.951**
mnist	0.828	0.966	0.923	0.944	**0.978**
musk	0.912	**0.964**	0.938	0.901	0.947
nomao	0.797	0.817	0.804	0.849	**0.914**
semeion	0.732	0.926	0.894	0.884	**0.945**

Table A.11: SVM classification results of ClEnDAE, PCA, LDA, ISOMAP and LLE for test data (AUC).

Dataset	PCA	LDA	ISOMAP	LLE	ClEnDAE
arcene	0.531	0.545	0.721	**0.722**	0.642
batch	0.983	0.980	0.983	0.892	**0.985**
coil2000	0.493	**0.500**	**0.500**	**0.500**	**0.500**
dota	0.517	0.529	0.547	0.548	**0.553**
drive	0.978	0.979	0.981	**0.982**	0.972
facial	0.672	0.703	0.697	0.501	**0.742**
fashionmnist	0.921	0.935	0.929	0.934	**0.943**
gisette	0.925	0.941	0.962	0.963	**0.972**
hapt	0.788	0.875	0.859	0.869	**0.902**
image	0.850	0.862	0.784	0.620	**0.891**
isolet	0.958	0.965	0.963	0.966	**0.980**
letter	0.812	0.839	0.842	0.852	**0.861**
madelon	0.557	0.585	0.547	0.523	**0.594**
mfeat	0.926	0.962	0.980	0.981	**0.985**
microv1	0.731	0.876	**0.953**	**0.953**	0.925
microv2	0.652	0.791	0.832	0.835	**0.893**
mnist	0.971	0.979	0.829	0.976	**0.986**
musk	0.955	0.958	0.959	0.954	**0.962**
nomao	0.929	0.930	0.933	0.925	**0.935**
semeion	0.936	0.957	0.929	0.949	**0.959**

Table A.12: MLP classification results of ClEnDAE, PCA, LDA, ISOMAP and LLE for test data (AUC).

Dataset	PCA	LDA	ISOMAP	LLE	ClEnDAE
arcene	0.521	0.512	**0.691**	0.673	0.525
batch	0.915	0.512	0.919	0.566	**0.926**
coil2000	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**
dota	0.524	0.525	0.527	0.525	**0.538**
drive	**0.941**	0.883	0.802	0.809	0.772
facial	0.692	0.704	0.707	0.697	**0.716**
fashionmnist	0.902	0.909	0.898	0.905	**0.914**
gisette	0.602	0.612	0.609	0.605	**0.682**
hapt	0.834	0.839	0.840	0.836	**0.857**
image	0.881	0.552	0.672	0.645	**0.888**
isolet	0.849	0.819	0.791	0.805	**0.869**
letter	0.713	0.732	0.739	0.722	**0.748**
madelon	0.577	0.582	0.556	0.512	**0.641**
mfeat	0.921	0.953	0.974	0.964	**0.975**
microv1	0.694	0.701	0.829	**0.868**	0.711
microv2	0.731	0.720	0.740	0.733	**0.742**
mnist	0.902	**0.906**	0.893	0.896	**0.906**
musk	0.992	0.991	0.994	0.952	**0.999**
nomao	0.913	**0.915**	0.911	0.914	0.653
semeion	0.820	**0.849**	0.746	0.822	0.789

Table A.13: C4.5 classification results of ClEnDAE, PCA, LDA, ISOMAP and LLE for test data (AUC).

Dataset	PCA	LDA	ISOMAP	LLE	ClEnDAE
arcene	0.654	0.662	0.766	0.695	**0.699**
batch	0.971	0.973	0.980	0.975	**0.984**
coil2000	0.509	0.504	0.507	0.500	**0.521**
dota	0.527	0.531	0.503	0.515	**0.546**
drive	0.936	0.938	**0.954**	0.947	0.884
facial	0.626	0.651	0.635	0.632	**0.672**
fashionmnist	0.872	0.883	0.879	0.881	**0.889**
gisette	0.682	0.705	0.692	0.709	**0.749**
hapt	0.797	0.791	0.792	0.788	**0.802**
image	0.913	**0.922**	0.774	0.838	0.874
isolet	0.840	0.859	0.851	0.862	**0.878**
letter	0.783	0.811	0.793	0.805	**0.821**
madelon	0.626	**0.677**	0.652	0.631	0.553
mfeat	0.889	0.898	0.892	0.895	**0.931**
microv1	0.731	0.806	0.685	0.868	**0.880**
microv2	0.835	0.837	0.815	0.838	**0.863**
mnist	0.869	0.872	0.876	0.871	**0.977**
musk	0.891	**0.927**	0.897	0.915	0.886
nomao	0.881	0.862	0.767	0.872	**0.889**
semeion	0.729	0.739	0.744	0.731	**0.751**