TRABAJO DE FIN DE GRADO

Reducción de dimensionalidad en problemas de regresión

Grado en Ingeniería Informática

Título: Reducción de dimensionalidad en problemas de regresión.

Autor: Covadonga Olivera Fernández-Cortés

Tutores: Ricardo Aler Mur y José María Valls Ferrán

Titulación: Grado en Ingeniería Informática

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Trabajo de Fin de Grado el día ___ de _____ de 20____ en Colmenarejo, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO **PRESIDENTE**

AGRADECIMIENTOS

Quiero agradecer y hacer una pequeña referencia a todas aquellas personas que han estado apoyándome, no solo durante la realización de este trabajo de fin de grado, si no a lo largo de todos estos años. Quiero pedir disculpas de antemano por si olvido mencionar a alguna persona, ya que en estos momentos no tengo la cabeza demasiado ordenada, pero todas aquellas personas que me quieren GRACIAS.

En primer lugar, quiero agradecérselo a mis padres Luis y Covadonga, que me han apoyado incondicionalmente durante toda mi vida, me lo han dado todo, han creído en mí y han tenido la paciencia necesaria para aguantarme siempre. Quiero que sean los primeros en aparecer en estos agradecimientos, porque sé que nadie va a estar más orgulloso cuando diga que he terminado la carrera que ellos y además se lo merecen.

A mi hermana Lucía, que desde pequeñas hemos estado juntas para lo bueno y para lo malo. Que, aunque la vida nos lleve por caminos distintos, siempre estaremos juntas y sé que siempre podré contar con ella cuando lo necesite.

A Rafa, aunque no siempre ha estado en mi vida, en los últimos años ha formado y formará una gran parte de ella. Te agradezco haber estado estos años conmigo, viéndome como la mejor persona del mundo, aunque no lo sea.

A Nena, María e Inma que, aunque con algunas no me una la sangre, son mi familia y han estado siempre. Viendo algunos de los libros de la carrera les han entrado dolor de cabeza, pero siempre han creído que podía hacerlo y al final aquí estoy.

A Javi, que nos une una amistad muy grande y ahora un gran futuro laboral. Nos conocemos desde hace muchos años y a día de hoy todavía no hemos discutido ni una sola vez, no de todo el mundo se puede decir los mismo.

A los peludos que ahora y siempre me acompañan, Bruce y Nora, y a los que han estado a lo largo de mi vida y que por desgracia ya no están. No dan muchos consejos, pero siempre los tengo a mi alrededor. Durante todos los meses que he estado sentada delante de este ordenador realizando este trabajo, han estado tumbados a mis pies y es su forma de apoyarme en la recta final.

Al resto de mi familia, sobre todo a mis abuelas, aunque ya no estén, este momento les habría hecho mucha ilusión y estarían muy orgullosas de mí.

También he de agradecérselo a mis profesores, algunos me lo han hecho pasar mejor que otros a lo largo de la carrera, pero al final todo se saca con dedicación. Sobre todo, agradecérselo a mis tutores que han tenido mucha paciencia conmigo, pero al final estoy escribiendo mis últimas palabras del trabajo.

A todos, muchas GRACIAS por vuestro apoyo y cariño.

RESUMEN

El presente trabajo está basado en un estudio de investigación, donde lo que se pretende es demostrar que se puede reducir la dimensionalidad de los atributos de entrada en dominios de regresión, mediante técnicas de inteligencia artificial basadas en aprendizaje supervisado. Es un estudio interesante debido al crecimiento de la inteligencia artificial en la actualidad y a la progresión que se espera que tenga en los próximos años. Por lo tanto, a nivel personal se cree que todo lo que sea investigación en esta rama de la informática es sinónimo de avance en el futuro.

Para llevar a cabo el estudio, se realizan una serie de experimentos sobre unos conjuntos de datos, donde se realiza una comparación entre una técnica basada en aprendizaje supervisado con una técnica cuyo objetivo es reducir la dimensionalidad de los atributos de entrada.

La técnica de aprendizaje supervisado escogida son las Redes de Neuronas Artificiales (RNA), es un paradigma de aprendizaje y procesamiento automático muy interesante inspirado en el cerebro. Se trata de un sistema de interconexión de neuronas que colaboran entre sí para producir un estímulo de salida. La estructura de las redes de neuronas artificiales está formada por una capa de entrada, una o varias capas ocultas, en este proyecto únicamente se cuenta con una capa oculta; y una capa de salida; cada una de estas capas consta de neuronas conectadas entre sí. Utilizan un tipo de aprendizaje supervisado, las neuronas situadas en la capa de entrada, propagan los atributos hacia la última capa, generando un estímulo salida, el cual se compara con una salida deseada, calculando y ajustando de esta manera el error que se haya producido.
La forma en la que en el presente estudio se reduce la dimensión de los atributos de entrada mediante RNA, es obteniendo las activaciones que calculan las neuronas que forman la capa oculta de la red y las entradas netas a estas neuronas; de este modo se reducen los N atributos de entrada originales a M neuronas ocultas.

El algoritmo con el que se va a comparar el comportamiento de las RNA, es una técnica que en la actualidad es una de las más empleadas para reducir la dimensión de los datos, denominada Principal Component Analysis (PCA). Esta técnica realiza una transformación de los atributos de entrada obteniendo un listado con los componentes principales ordenados de mayor a menor relevancia; de este modo escogiendo los primeros componentes del listado se consigue reducir la dimensión de los atributos de entrada.

Para analizar y comparar el comportamiento de estas dos técnicas, se emplea el modelo de aprendizaje automático K-Nearest Neighbors (KNN), es un algoritmo empleado tanto para clasificación como para regresión, y es de esta última manera del modo en el que se va a utilizar, ya que los dominios sobre los que se realizan los experimentos contienen salidas continuas. La forma de aprender es, que para cada patrón de entrada nuevo se escogen los K vecinos más cercanos a él del conjunto de datos de entrenamiento introducidos previamente, y se calcula la salida del nuevo dato realizando la media de las salidas de los K vecinos más cercanos.

Para lograr los objetivos estipulados en el presente trabajo, se realiza para cada dominio un total de ochenta y ocho experimentos, donde los datos se van a ver reducidos en siete ocasiones de forma progresiva, desde un 50% menos de atributos de entrada hasta un 7,14% (50/7) y analizados mediante KNN teniendo en cuenta 1, 3, 5 y 7 vecinos más cercanos. De este modo, para las RNA se realizan 7 arquitecturas modificando el número de neuronas ocultas y obteniendo las activaciones de dos maneras distintas; estos dos subconjuntos de datos reducidos serán analizados mediante KNN calculando el coeficiente de determinación logrado variando el valor del parámetro K. Con el método PCA, se obtienen 7 subconjuntos de componentes principales y se aplica KNN para obtener el coeficiente de determinación logrado. Por último, se aplica KNN sobre los datos originales.

Una vez realizados todos los experimentos y habiendo obtenido el coeficiente de determinación de los mismos, se realiza una comparación de los resultados para poder determinar que reducir la dimensión de los atributos de entrada mediante RNA alcanza, en la mayoría de los casos, mejores resultados que el algoritmo PCA e incluso que los propios datos originales.

Se concluye, para los conjuntos de datos estudiados, que es favorable reducir la dimensión de los atributos de entrada mediante esta técnica de aprendizaje supervisado (RNA) y que en la mayoría de los casos funciona mejor que un algoritmo creado específicamente para ello PCA.

ÍNDICE DE CONTENIDO

CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

1. INTRODUCCIÓN Y OBJETIVOS

1.1. Introducción

En el presente trabajo se ha realizado un estudio de investigación, para determinar si es posible reducir la dimensión de un conjunto de datos mediante técnicas basadas en la inteligencia artificial y realizar una comparación con técnicas especializadas en efectuar reducciones sobre dichos datos.

Se ha considerado un tema de estudio interesante, debido a la relevancia que tiene el "Big Data" en la actualidad y al incesante desarrollo de las ciencias de la computación.

La Inteligencia artificial es la rama de las ciencias de la computación que nos interesa para el presente estudio. Su finalidad es elaborar teorías y modelos que manifiesten el funcionamiento y la organización de la inteligencia. En la actualidad, el mayor esfuerzo en la búsqueda de inteligencia artificial se centra en el desarrollo de sistemas de procesamiento de datos que sean capaces de simular a la inteligencia humana, desempeñando tareas que requieran aprendizaje, solución de problemas y toma de decisiones. La inteligencia artificial se centra en dos áreas de estudio: el cuerpo humano y el ordenador; ya que su objetivo final es producir la inteligencia, dando paso a un estudio más exhaustivo del aprendizaje humano y, por lo tanto, a una mejor comprensión del mismo. Si esto se consiguiera se daría un nuevo enfoque a la computación y se crearían aplicaciones nuevas.

Según la Real Academia Española (RAE) la inteligencia artificial se define como: *"Disciplina científica que se ocupa de crear programas informáticos que ejecutan operaciones comparables a las que realiza la mente humana, como el aprendizaje o el razonamiento lógico"*. Lo que otorga las capacidades de aprender y razonar al ser humano, es el cerebro, que está compuesto por redes de neuronas. Inspiradas en el funcionamiento del sistema nervioso de los organismos vivos, surgen las redes de neuronas artificiales utilizadas en este trabajo. Las redes de neuronas artificiales se han hecho muy populares debido a la facilidad en su uso e implementación y a la habilidad para aproximar cualquier función matemática; poseen una marcada habilidad para obtener resultados de datos complicados e imprecisos, se emplean para obtener patrones y detectar tramas que son complicadas de apreciar por los humanos u otras técnicas computacionales.

Las redes de neuronas artificiales logran un modelo no explicito, donde relacionan un conjunto de atributos de entrada con un conjunto de atributos de salida. Realizan un aprendizaje supervisado, por lo que se debe partir de un conjunto de datos que la red toma como entrenamiento y que hacen que sea capaz de aprender por medio de un ajuste de sus parámetros según el error que cometa la red, para predecir las salidas de nuevas observaciones con el mayor ajuste posible.

Para entender el modo en el que se pretende reducir la dimensionalidad de los datos de entrada mediante redes de neuronas artificiales, se debe comprender el funcionamiento de las mismas. Las redes de neuronas están formadas por neuronas o nodos, que se encuentran en niveles o capas dentro de la arquitectura de la red; existen tres tipos de capas o niveles: capa de entrada cuyos nodos se encargan de introducir la información o los atributos de entrada a la red, capas ocultas que transmiten la información hacia la salida, y capa de salida donde las neuronas de salidas generan la salida producida por la red. Todas las neuronas están conectadas con las neuronas de la capa siguiente y estas conexiones tienen unos pesos, que se irán modificando según el error que cometan para ajustar la red a los patrones de entrenamiento, de este modo es cómo la red aprende.

Lo que se va a realizar para reducir la dimensión de los atributos de entrada es obtener las salidas que producen las neuronas de la capa oculta de la red y obtener las entradas netas a dichas neuronas,

reduciendo de este modo el número de atributos de entrada en el número de neuronas ocultas que contenga la red.

Con el fin de comparar con un método clásico de reducción de dimensionalidad, se va a aplicar el modelo PCA (Principal Component Analysis). Es un modelo que transforma los atributos de entrada y los reordena según el grado de relevancia, por lo tanto, se puede enmarcar como una técnica no supervisada, ya que únicamente se parte de los atributos de entrada.

En este estudio nos interesan los modelos supervisados, puesto que se quiere aplicar una transformación de los atributos de entrada y una reducción de dimensionalidad a problemas de regresión, que son problemas supervisados. Por este motivo se quiere realizar una comparación entre reducir la dimensión de los atributos de entrada mediante redes de neuronas artificiales y mediante PCA, que es una técnica especializada para tal fin. De esta manera se realiza una comparación entre un método supervisado y uno no supervisado, para reducir la dimensionalidad de los datos.

Para llevar a cabo estas comparaciones se empleará el algoritmo de aprendizaje automático: KNN (K nearest neighbours) o K-vecinos. Se trata de un modelo que es sensible a la presencia de atributos irrelevantes y que se puede emplear tanto para clasificación y como para regresión, que es como se ha empleado en este trabajo, al tratarse con conjuntos de datos cuya salida es continua. Con la ayuda de este modelo y aplicando los datos reducidos por las dos técnicas anteriormente descritas, se comprobará la eficacia de ambas a la hora de reducir la dimensionalidad de los datos en problemas de regresión.

1.2. Objetivos

El objetivo del trabajo es determinar si es posible realizar una reducción de la dimensionalidad de varios conjuntos de datos en problemas de regresión mediante diferentes métodos. Lo que se pretende es aplicar diferentes técnicas que reduzcan la dimensión del conjunto de datos de entrada mediante una transformación, compararlas entre sí y analizar si se alcanzan buenos resultados empleando un menor número de atributos de entrada. También se va a analizar si es más eficaz reducir la dimensión de los datos mediante técnicas basadas en aprendizaje automático, como es el caso del Perceptrón Multicapa o mediante el modelo estadístico PCA.

Se trata por lo tanto de un trabajo de investigación en el que no se sabe previamente si los resultados que se alcanzarán son favorables o no. En caso de que los resultados sean positivos, se da pie a realizar investigaciones más exhaustivas en el campo y a empezar a desempeñar estas técnicas de reducción en problemas que cuenten con un número elevado de atributos.

Resulta un ámbito de investigación interesante, ya que en la actualidad el término “Big Data” está en auge, y poder reducir el volumen de información, es decir, poder reducir la dimensión de los datos sin perder información relevante, es un avance muy importante.

En conclusión, los objetivos del presente trabajo son:

1. Entrenar una red de neuronas capaz de ajustarse a los patrones de entrada nuevos y extraer la transformación realizada por las neuronas de la capa oculta.
2. Aplicar el modelo KNN sobre las activaciones obtenidas por la red de neuronas, este es el método de transformación y reducción supervisado.
3. Aplicar el modelo KNN sobre los datos originales.
4. Realizar una transformación de los datos mediante PCA y aplicar el modelo KNN, este es el método no supervisado.

5. Realizar una comparación entre los resultados logrados tras las transformaciones y, por lo tanto, las reducciones llevadas a cabo por las redes de neuronas artificiales y por el algoritmo PCA; para poder determinar la técnica más eficaz a la hora de reducir la dimensionalidad.
6. Analizar si es viable realizar estas reducciones sin perder información relevante.

1.3. Estructura del documento

El presente trabajo consta de 9 capítulos, el primero de ellos es una introducción y objetivos del trabajo, se expone un planteamiento del núcleo central del estudio; donde se expone el propósito y los objetivos del mismo.

El segundo capítulo es el denominado estado del arte, en él se incluyen todos los conocimientos necesarios para llevar a cabo el trabajo y su finalidad es poner en situación al lector.

El tercer capítulo está formado por el análisis, diseño e implementación del sistema; En este capítulo se muestra el diseño que se lleva a cabo para desarrollar el trabajo tras analizar el problema a resolver, así como las diferentes implementaciones realizadas y una descripción del entorno de desarrollo que se ha empleado.

En el cuarto capítulo se muestran todos los estudios realizados en el presente trabajo; partiendo de las metodologías llevadas a cabo, así como los dominios utilizados y concluyendo con los resultados alcanzados y un análisis de dichos resultados.

El capítulo 5 contiene la planificación seguida para la realización del proyecto, así como un desglose del presupuesto necesario para llevarlo a cabo.

El capítulo 6 muestra el marco regulador que rige este trabajo.

El capítulo 7 cuenta con las conclusiones obtenidas tras la realización del proyecto, también cuenta con las conclusiones propias del autor y con futuros trabajos que se pueden desempeñar basándose en los resultados obtenidos en el presente trabajo.

El capítulo 8 contiene la bibliografía que se ha consultado para realizar el trabajo, ya que se ha requerido una documentación a lo largo de todo el trabajo.

El capítulo 9 contiene el glosario de acrónimos que aparecen a lo largo del presente documento.

2. ESTADO DEL ARTE

2.1. Inteligencia Artificial

2.1.1. Introducción

Puede decirse que una de las áreas de las ciencias de la computación más atrayente y con más desafíos es la Inteligencia Artificial (IA). Surgió como un mero estudio filosófico y racional de la inteligencia humana, combinada con la inquietud del hombre por imitar su propia naturaleza.

La idea de algo semejante a la inteligencia artificial se remonta hasta hace millones de años. El primer hombre primitivo que fue consciente de su existencia y de su capacidad de pensar y, por lo tanto, de que poseía inteligencia; probablemente se preguntó del funcionamiento de su propio pensamiento e inteligencia, llegando a la conclusión de un "creador superior". De este modo, la idea de que un ser inteligente cree a otro, la idea de un diseño virtual para la inteligencia, es tan antigua como la toma de conciencia del ser humano.

Debido a la gran amplitud del área de estudio de la Inteligencia Artificial, ésta tuvo numerosos padres, y por ello, no existe un consenso para definir el concepto de Inteligencia Artificial; pero se puede indicar que es el área de las ciencias de la computación encargada de modelar en sistemas computacionales la inteligencia humana.

La Inteligencia Artificial es el área de la ciencia que, desde un punto de vista global, se encomienda al estudio de la inteligencia en elementos artificiales y, visto desde el punto de la ingeniería, se centra en la creación de componentes que posean un comportamiento inteligente. Citado de otra forma, la Inteligencia Artificial pretende construir sistemas y máquinas que sean capaces de tener un comportamiento que sea inteligente, es decir, intentan imitar el comportamiento inteligente de un ser humano. Se considera un comportamiento inteligente, aquel en el que se produce un aprendizaje, el que tiene capacidad de adaptarse a entornos cambiantes, etc. Además, en la Inteligencia Artificial cooperan muchas disciplinas, por lo que sus campos de aplicación son muy amplios y están interrelacionados; intervienen disciplinas tan variadas como la neurociencia, la psicología, las tecnologías de la información, la física, las matemáticas, etc.

Desde un punto de vista más práctico, se muestran cuatro definiciones de Inteligencia Artificial, de distintos autores, desde cuatro enfoques diferentes. Algunas de estas definiciones hacen referencia a procesos de razonamiento y mentales (como son las definiciones de sistemas que piensan como humanos y sistemas que piensan racionalmente), mientras que otras se refieren a la conducta (como sistemas que actúan como humanos y sistemas que actúan racionalmente). Para medir el éxito, unas definiciones lo regulan en términos de similitud a la forma de actuar humana; otras toman como referencia el concepto ideal de inteligencia, que se denominará racionalidad. Se entiende que un sistema es racional, si realiza "lo correcto" basándose en su conocimiento.

Las definiciones de Inteligencia Artificial desde cuatro enfoques distintos son:

- **Sistemas que piensan como humanos**
  "*El nuevo y excitante esfuerzo de hacer que los computadores piensen… máquinas con mentes, en el más amplio sentido literal*". (Haugeland, 1985)

"*[La automatización de] actividades que vinculamos con procesos de pensamiento humano, actividades como la toma de decisiones, resolución de problemas, aprendizaje…".* (Bellman, 1978)

- **Sistemas que piensan racionalmente**
  "*El estudio de las facultades mentales mediante el uso de modelos computacionales*". (Charniak y McDermott, 1985)
  "*El estudio de los cálculos que hacen posible percibir, razonar y actuar*". (Winston, 1992)
- **Sistemas que actúan como humanos**
  "*El arte de desarrollar máquinas con capacidad para realizar funciones que cuando son realizadas por personas requieren de inteligencia*" (Kurzweil, 1990).
  "*El estudio de cómo lograr que los computadores realicen tareas que, por el momento, los humanos hacen mejor*". (Rich y Knight, 1991)
- **Sistemas que actúan racionalmente**
  "*La Inteligencia Computacional es el estudio del diseño de agente inteligentes*". (Poole et al., 1998)
  "*IA… está relacionada con conductas inteligentes en artefactos*". (Nilsson, 1998)

(Russell & Norvig, 2004)

Como se ha mencionado anteriormente, la Inteligencia Artificial es interdisciplinar, cooperan infinidad de disciplinas; por lo tanto, puede llegar a tener infinidad de aplicaciones. A continuación, se citan algunos de los campos más destacados de la IA:

1. **Aprendizaje Automático (Machine Learning):** su finalidad es elaborar programas que sean capaces de aprender, es decir, de otorgar la capacidad de aprender a las computadoras mediante una serie de técnicas. Poseen diversas aplicaciones, como motores de búsqueda, diagnósticos médicos, detección de fraudes en tarjetas de crédito, etc.
2. **Ingeniería del conocimiento (Knowledge Engineering):** se trata de una rama de la IA más moderna, cuya finalidad es diseñar y desarrollar Sistemas Expertos, también conocidos como Sistemas Basados en el Conocimiento. Para ello, deben obtener, enunciar e informatizar el conocimiento de un experto.
3. **Lógica difusa (Fuzzy Logic):** son capaces de otorgar a los problemas de una forma natural, eficiente y robusta, la información borrosa o imprecisa.
4. **Redes neuronales artificiales (Artificial Neural Networks):** son técnicas inspiradas en el funcionamiento del cerebro humano, pero forman parte del campo más amplio del Aprendizaje Automático.
5. **Sistemas reactivos (Reactive Systems):** son sistemas cuyo comportamiento se basa en una secuencia de estímulos-respuestas en relación con el medio. Con el tiempo, se han expandido a sistemas de tiempo real, donde dependen de las magnitudes de los retardos temporales.
6. **Sistemas multi-agente (Multi-Agent Systems):** como su propio nombre indica, son sistemas formados por una gran cantidad de agentes inteligentes, que interactúan entre sí. Al estar compuesto por varios agentes inteligentes, son capaces de solucionar problemas que son imposibles de resolver por un único agente o sistema monolítico.
7. **Sistemas basados en reglas (Rule-Based Systems**): son sistemas que funcionan aplicando reglas; aplican una serie de reglas dependiendo del estado en el que se encuentre en ese momento y aplica nuevas reglas según el estado tras realizar las reglas anteriores.
8. **Razonamiento basado en casos (Case-Based Reasoning):** son capaces de emplear situaciones de un problema pasado concreto, que se denominan casos, y adaptarlos para utilizarlos en el problema actual.

Este trabajo está centrado en dos técnicas pertenecientes al campo del aprendizaje automático: el algoritmo KNN y las Redes de Neuronas; para el estudio de la reducción de dimensionalidad en problemas de regresión. Estas técnicas desarrolladas serán explicadas más adelante con más detalle.

(ITAM, 1987)

2.2. Aprendizaje Automático

2.2.1. Historia del Aprendizaje Automático

El Aprendizaje Automático (AA) surgió en el momento en el que la IA comenzó a centrarse en el tratamiento simbólico de la información, es en esta etapa, cuando se desarrollaron los primeros programas que enfrentaban el problema de aprendizaje desde un punto de vista simbólico. En este periodo se desarrolla un programa capaz de jugar a las damas, creado por Samuel en 1963, que fue el primer programa de aprendizaje. A lo largo del tiempo, de este primer programa original, se han realizado diferentes modificaciones que han dado lugar a un gran número de soluciones distintas.

Se puede decir que el aprendizaje automático ha evolucionado a lo largo de la historia en función de cuatro fases: inicial, intermedia, de asentamiento y actual.

Etapa Inicial (1955 – 1965)
Inicialmente todos los trabajos que se realizaron en esta época, se centraron en lo que otorga la facultad de pensar al ser humano; por lo tanto, se basaron en el conocimiento que poseían del funcionamiento biológico del cerebro humano, que tiene la capacidad de memorizar, autoorganizarse y aprender. De este modo se desarrollaron las primeras versiones de las redes de neuronas artificiales. De cualquier forma, todos los modelos de aprendizaje desarrollados en esta etapa eran esencialmente numéricos.

Etapa intermedia (1962 – 1976)
El principal avance que se realiza en este periodo, es que se comienza a introducir el uso del cálculo simbólico frente al exclusivamente numérico de la primera etapa; también se comienza a proporcionar conocimiento inicial del dominio a los sistemas, dándoles la capacidad de adquirir conocimiento.
En cualquier caso, el principal objetivo de esta época era crear mecanismos generales de aprendizaje partiendo de muy poco conocimiento inicial.

Etapa de asentamiento (1976 – 1988)
En un primer momento, se dejó de realizar trabajos e investigar en el área del aprendizaje automático, causado principalmente por dos motivos; uno de ellos fue el libro publicado por Minsky y Papert, “Perceptrons”, donde se demostraba la limitación del Perceptrón para aprender, y otro gran motivo fue la limitación de las capacidades de las computadoras de la época para afrontar la complejidad que suponían los problemas planteados de AA.
Pese a todo ello, a finales de los años 70 se retomó el estudio de esta rama; y supuso el desarrollo y la divulgación de las principales técnicas en el campo. Aparecen las primeras aplicaciones reales y se comienzan a realizar conferencias a nivel mundial sobre el tema, lo que logra un asentamiento del aprendizaje automático en todo el mundo.

Etapa actual (1988 – actualidad)
Se comienza a evolucionar en los conceptos de conocimiento y en la preocupación por los aspectos metodológicos en los diferentes campos. Todo ello acompañado por un desarrollo de aplicaciones de interés comercial, lo que da lugar a mayor cantidad de investigaciones en este campo.

A principios de los años 90, se produce un gran crecimiento del interés por llevar a cabo propuestas de carácter metodológico que aclaren el campo y que propongan distintas áreas de investigación. Todo ello ha provocado que, a día de hoy, exista una gran variedad de aplicaciones de aprendizaje automático.

(Borrajo Millán, González Boticario, & Isasi Viñuela, 2006)

2.2.2. Conceptos del Aprendizaje Automático

El aprendizaje automático es una rama de la inteligencia artificial, que surge con el objetivo de ayudar al hombre a solucionar problemas complejos mediante la creación de técnicas que permitan a las computadoras aprender. Por lo tanto, se centran en crear programas que, partiendo de información no estructurada, sean capaces de generalizar comportamientos considerados inteligentes. Es, por lo tanto, un proceso que incita o instiga al conocimiento.

El principal objetivo que el ordenador debe realizar de manera completamente automática, es disponer de una forma de aprendizaje que sea capaz de resolver problemas por sí mismo. Por lo que debe tener la capacidad, de adquirir de conocimientos para solucionar nuevos problemas o incluso mejorar en la mejor medida su comportamiento en base a la experiencia. Para ello, el ordenador debe conocer tres elementos: espacio de aprendizaje, finalizad u objetivo a cumplir y soluciones que sea capaz de ofrecer.

El propósito principal del aprendizaje automático se basa en poder proporcionar a los ordenadores de adquisición de conocimiento, habilidades y hábitos a través de experiencias, estudio de diversas situaciones, capacidad de razonamiento y observación del entorno. El aprendizaje automático dispone de un gran número de tareas y metodologías cuyo principal objetivo es diseñar programas que puedan ajustarse a las necesidades que posee el entorno del problema, así como medrar en la realización de sus tareas, ser capaz de hallar nuevos problemas para buscar una solución y, de esta manera, brindar la posibilidad de adquirir nuevo conocimiento.

En resumen, el aprendizaje automático persigue el objetivo de, partiendo de distintos ejemplos en diferentes situaciones, sea capaz de adquirir nuevo conocimiento para la realización de nuevos problemas o tareas y, de especial importancia, tener la capacidad de reducir los recursos para realizar estas tareas en el mejor tiempo posible, reduciendo costes.

(Borrajo Millán, González Boticario, & Isasi Viñuela, 2006)

Existen distintos tipos de problemas que se pueden solucionar mediante aprendizaje automático, se va a hacer una distinción por el tipo de objetos que intentar predecir; ya que los programas de aprendizaje automático en función de unas entradas proporcionadas, son capaces de obtener unas salidas en base a una técnica de aprendizaje. Las principales clases de problemas que se encuentran son:

1. **Regresión:** intentan predecir un valor de una variable de respuesta que toma un valor real y continuo.
2. **Clasificación:** intentan predecir la clase a la que pertenecen los objetos sobre un conjunto de clases prefijadas.

La principal diferencia entre estos dos tipos de problemas es la respuesta que requiere el problema a resolver, puede ser un número cualquiera o una clase (categoría). Las técnicas de aprendizaje supervisado basadas en problemas de regresión, intentan modelar una función que se aproxime a las salidas del problema; mientras que las técnicas basadas en clasificación crean modelos que asignen una clase a los datos dados.

En la siguiente ilustración, se aprecia la diferencia entre clasificación y regresión; en la primera gráfica el objetivo es ser capaces de diferenciar la clase a la que pertenecen los datos, representada mediante colores (azul, rojo o verde); y en la segunda gráfica se intenta predecir la función correspondiente con las salidas de los datos.

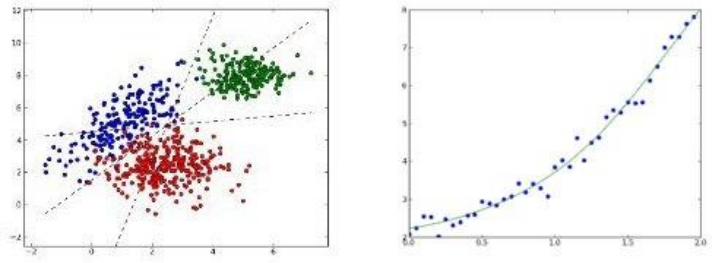


Ilustración 1: Clasificación VS Regresión

Normalmente, lo primero que se debe hacer, es enmarcar el problema presentado dentro de alguna de las clases anteriormente mencionadas, ya que, dependiendo del tipo de problema al que pertenezca, será la forma en la que se pueda medir el error cometido entre la predicción y la realidad.

Una de las principales tareas a la hora de aplicar aprendizaje automático, es fomentar la experiencia de la máquina por medio de objetos con los que entrenarse, que se denominan ejemplos, para posteriormente aplicar los patrones que haya reconocido sobre otros objetos distintos.

Sabiendo que para que una máquina aprenda se debe partir de una serie de ejemplos con los que entrenarla, se puede clasificar diferentes tipos de aprendizaje automático en función del tipo de salida que se produzca y del tratamiento que se les otorgue a los ejemplos. En este trabajo se va a realizar una distinción entre aprendizaje supervisado y aprendizaje no supervisado; aunque existen más tipos de aprendizajes.

- **Aprendizaje supervisado:** Este tipo de aprendizaje se realiza a través de un proceso de entrenamiento, el cual está supervisado por un agente externo que es capaz de determinar una respuesta o salida en base a uno o varios datos de entrada. Este conjunto de ejemplos conocidos de antemano se denomina como ejemplos “a priori”.
- **Aprendizaje no supervisado:** Por el contrario, en este tipo de aprendizaje no se establece un supervisor externo, y, por lo tanto, no se recibe de antemano ningún tipo de información del exterior que ayude a indicar si la salida obtenida es necesariamente la salida precisa, por lo que es necesario buscar coherencias, patrones y características comunes en los ejemplos de entrada.

(Sancho Caparrini, 2015)

Hoy en día, todo el mundo ha escuchado en alguna ocasión el término “Big Data”, consiste en el estudio y gestión de una enorme cantidad de datos, que debido a su dimensionalidad no pueden ser tratados de manera convencional. Existe un gran problema en la actualidad con la gran cantidad de información almacenada en las bases de datos, en muchas ocasiones es conveniente e incluso necesario reducir la dimensionalidad de las mismas empleando diversas técnicas de aprendizaje automático, capaces predecir patrones o generalidades en los mismos.

2.3. Redes de Neuronas Artificiales

2.3.1. Fundamentos biológicos de las Redes Neuronales

Las redes de neuronas artificiales están inspiradas en la estructura y el funcionamiento del sistema nervioso humano; esto es así debido a que el cerebro humano es el encargado de proporcionar inteligencia al hombre. Se va a exponer de manera breve, el funcionamiento del sistema de comunicación neuronal para poder comprender el desarrollo y el funcionamiento de las redes de neuronas artificiales.

El sistema de comunicación neuronal está compuesto por tres partes:

- Los receptores: que como su nombre indica, reciben la información que el ser humano recoge a partir de estímulos, esta información puede provenir tanto del interior como del exterior del organismo.
- El sistema nervioso: está en comunicación tanto con los receptores, encargados de transmitirle la información, como con los órganos efectores, así como con otras partes del propio sistema nervioso. De este modo, reciben la información, la transmiten y la elaboran; teniendo la capacidad de almacenar esa información para poder enviarla posteriormente.
- Órganos efectores o diana: son los encargados de recibir la información elaborada por el sistema nervioso e interpretarla en acciones.

El sistema nervioso está compuesto por un conjunto de neuronas conectadas entre sí, formando una malla encargada de elaborar y transmitir la información. El componente principal es la neurona que, cuenta con una serie de componentes esenciales que se muestran en la siguiente ilustración. Donde se aprecia el axón, que es la ramificación de salida de las neuronas, a través de él se propaga la información. Además, cuenta con las dendritas, que son las ramificaciones de entrada encargadas de propagar las señales al interior. Las sinapsis que como puede apreciarse en la imagen, son las encargadas de recoger la información procedente de las células vecinas; esta información llega al núcleo, donde se procesa y genera una respuesta que es propagada por el axón.

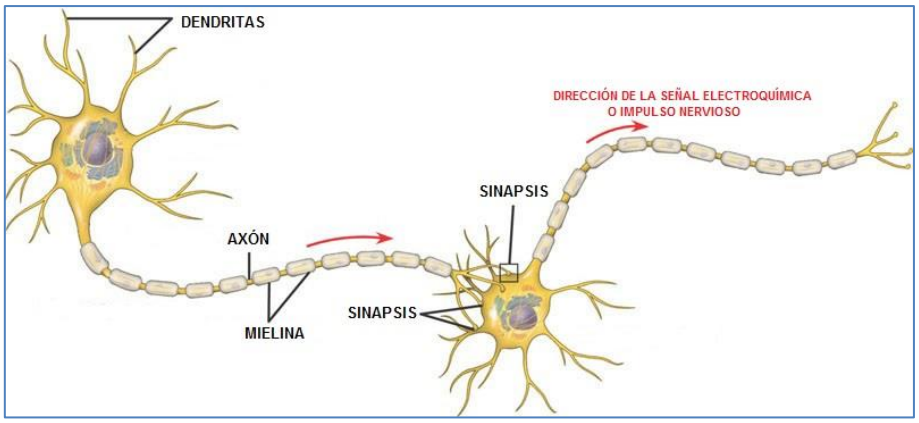


Ilustración 2: Diagrama de una neurona biológica

De forma simplificada y esquemática, las neuronas realizan cinco funciones esenciales:

1. Las neuronas pueden recibir información procedente de los receptores o de otras neuronas de la red en forma de impulsos, tienen la función de recoger dicha información.
2. Una vez que poseen la información, la deben tratar para integrarla dentro de la propia célula, este tratamiento se entiende como una codificación de la información que han recibido en forma de estímulo.
3. Con la información ya codificada, la transmiten a través de la red por medio del axón.
4. El axón se ramifica para distribuir la información a través de la red.

5. Los terminales son los encargados de transmitir la información a la célula con la que esté conectada, ya sea otra neurona de la red o una célula receptora.

En cuanto al funcionamiento general, será una enorme red compuesta por neuronas conectadas unas con otras, que reciben información del exterior, la procesan y la transmiten. Basándose en este funcionamiento se crean las redes de neuronas artificiales.

(Isasi Viñuela & Galván León, 2004)

2.3.2. Historia de las Redes Neuronales

Sigmund Freud fue un médico austriaco dedicado a la neurología y de él constan las primeras investigaciones a principios del siglo XX sobre las Redes de Neuronas Artificiales (RNA). Hasta a mediados de siglo, sobre la década de los 40 no se comenzó a mostrar especial interés en este campo, debido a la fuerte implicación de los científicos y el gran avance tecnológico.

A continuación, se describen las diferentes etapas más importantes en la historia sobre el avance en el campo de las redes de neuronas artificiales.

Década de los 40 y 50:
El primer modelo matemático con redes de neuronas artificiales fue creado en la década de los 40 por Warren McCulloch y Walter Pitts. Este primer ejemplar se basaba en los impulsos binarios de las neuronas mediante una función de paso y un umbral a la neurona, que posteriormente sería utilizada en otros muchos modelos de RNA.
Más adelante, en la década de los 50, se obtuvieron los primeros resultados funcionales con Marvin Minsky, cofundador del laboratorio de inteligencia artificial del Instituto Tecnológico de Massachusetts. Junto con la ayuda de Edmons, Minsky diseñó una máquina compuesta por 40 neuronas, las cuales se interconectaban y se ajustaban según los valores que iban dando lugar a medida que se realizaban ciertas tareas. La máquina, en realidad simulaba el comportamiento propio de una rata buscando comida.
Es ya a finales de los años 50, cuando se comienzan a desarrollar paradigmas en RNA. Albert Uttley diseñó una máquina la cual reajustaba sus parámetros dependiendo de la entrada recibida a través de emplear la medida de Shannon. Posteriormente se crearon el Perceptrón y Adaline, diseñados por Frank Rosenblatt y Bernard Widrow respectivamente. El Perceptrón se diseñó como una generalización de las interconexiones de células creados por Minsky. Ambos modelos, tanto el Perceptrón como Adaline, eran capaces de ajustar sus valores en base a los valores de entrada en función de los valores de salida obtenidos con los valores de salida esperados.

Década de los 60:
Los primeros métodos de codificación de la información en RNA, creados por Steinbuch, se aplicaron en modelos de reconocimiento de escritura a mano. Se intentaron combinar en una sola teoría los procesos mente-cerebro, y se combinaron las redes de neuronas biológicas con modelos complejos de RNA para el tratamiento de aprendizaje competitivo y memoria asociativa. A raíz de esto, surgió el modelo “Holophone”, una relación entre hologramas y memoria asociativa que se basa en un paradigma, el cual es capaz de generar una señal completa mediante el almacenamiento de señales de entrada.
Posteriormente se desarrolló el “Neocognitron”, más adelante mejorado al introducir el método basado en “*backpropagation*” (retropropagación) en el cual, cada neurona recibe una señal del error que describe su contribución relativa al error total.
También, a partir de 1969 se estudiaron las relaciones entre la biología del cerebro y la psicología de la mente en búsqueda de nuevos objetivos.

Década de los 70 y 80:
A principio de los años 70 comenzaron las investigaciones sobre las RNA con conexiones aleatorias (Mapas de Kohonen) y se creó el llamado "OLAM", Asociador Óptimo de Memoria Lineal el cual necesitaba vectores linealmente independientes para obtener buenos resultados y fue mejorado en la búsqueda de las óptimas en vectores linealmente dependientes.
Más adelante apareció "la máquina de Boltzmann", que fue la primera RNA capaz de reconocer un algoritmo de aprendizaje para una red de tres niveles de neuronas.
En 1982, se creó el algoritmo de Hopfield, un sistema de memoria asociativa con unidades binarias diseñadas para converger en un mínimo local.
Por último, Bart Kosko extendió las RNA asociativas de un nivel en dos niveles, mediante aprendizaje no supervisado, dada una matriz binaria poder converger en una solución mínima.

(Isasi Viñuela & Galván León, 2004)

2.3.3. Elementos de una Red Neuronal Artificial

El funcionamiento básico de una red de neuronas artificial, consiste en propagar una información de entrada, transformándola hasta conseguir una salida. Es importante tener en consideración todos los elementos de una red de neuronas y su funcionamiento, ya que dependiendo de las características con las que se doten a la red, influirá en su comportamiento final y será vital para resolver una determinada tarea.

Las redes de neuronas artificiales están compuestas por neuronas, cada neurona es una unidad de proceso. Las funciones que deben cumplir son recoger la información del exterior, transmitirla y procesarla; es por ello, que dependiendo del funcionamiento que adquiera la neurona dentro de la red, se pueden diferenciar tres tipos:

1. Unidades de entrada: son las encargadas de recoger la información procedente del exterior y transmitirla al interior.
2. Unidades ocultas: son las encargadas del procesamiento de la información. Pueden recibir la información directamente de las unidades de entrada o de otras unidades ocultas.
3. Unidades de salida: son las encargadas de dar una respuesta al sistema.

La red se divide en capas o niveles, cada capa está compuesta por todas las neuronas con el mismo comportamiento, es decir, pertenecen a la misma clase; y cuyas salidas se dirigen al mismo destino.

(Hilera González & Martínez Hernando, 1995)

2.3.3.1. La Neurona Artificial

La neurona es la unidad de procesamiento de la información de una red de neuronas, esta información debe ser integrada, computada y emitida por la misma; la información puede proceder de otros nodos o del exterior de la red.

Al igual que las neuronas biológicas mediante la sinapsis, la información que recibe la neurona procedente de otro nodo, puede adquirir mayor o menor fuerza, es lo que se conoce como fuerza sináptica. A cada información que recibe la neurona, se le asigna un peso, el cual hará que la información altere en mayor o menor medida a la neurona receptora, y por lo tanto a la salida que esta produzca. Estos pesos son los que otorgan a la red el recurso de memoria a largo plazo, y normalmente el aprendizaje de la red depende del ajuste de estos pesos. Todas las informaciones que recibe la neurona como entrada son ponderadas por los pesos correspondientes, este valor se denomina Net. Una vez integradas las entradas que recibe la neurona, esta información debe ser computada en la misma, para calcular su estado interno; para ello se realiza la función de activación

(F) sobre el conjunto de entradas recibido. Este estado interno será la información que la neurona transmita como salida.

Es importante destacar en qué consiste el estado interno de una neurona, denominado estado o nivel de activación $a_i(t)$ de la neurona U_i en el instante de tiempo t; que se corresponderá con la salida que transmita la neurona en ese momento. El cálculo del estado de activación de una neurona depende de dos componentes, por un lado la función lineal correspondiente al cálculo de la entrada que recibe Net_i en ese momento y explicada en el apartado *2.3.3.2.Estructura de una Red Neuronal y conexiones entre neuronas*; y por otro componente no lineal calculado mediante la función de activación F, explicada con detalle en *2.3.3.3.Función o regla de activación*. El conjunto de todos los estados de activación de las neuronas que forman la red se representa mediante el vector $A(t)$ de N números reales, correspondientes a las N neuronas de la red en el instante t.

$$A(t) = (a_1(t), a_2(t), \dots, a_i(t), \dots, a_N(t))$$

Por lo tanto, el procesamiento que realiza la red se puede entender como la evolución del vector que contiene las activaciones de las neuronas a través del tiempo.

La evolución de la red, es decir, el sistema que sigue la red para actualizar los estados internos de sus neuronas, puede ser de dos tipos: modo asíncrono y modo síncrono.

Modo asíncrono: cada neurona modifica su estado interno de forma independiente, según les va llegando nueva información; esta información llega de forma continua.
Modo síncrono: De la misma manera que en el caso anterior, la información llega de forma continua, la diferencia se encuentra en que, en este caso, las modificaciones de los estados internos se realizan de forma simultánea; es decir, todas las neuronas se modifican al mismo tiempo.

En la siguiente imagen se puede observar el concepto y el comportamiento de neurona artificial, y cómo se asemeja al funcionamiento de una neurona biológica.

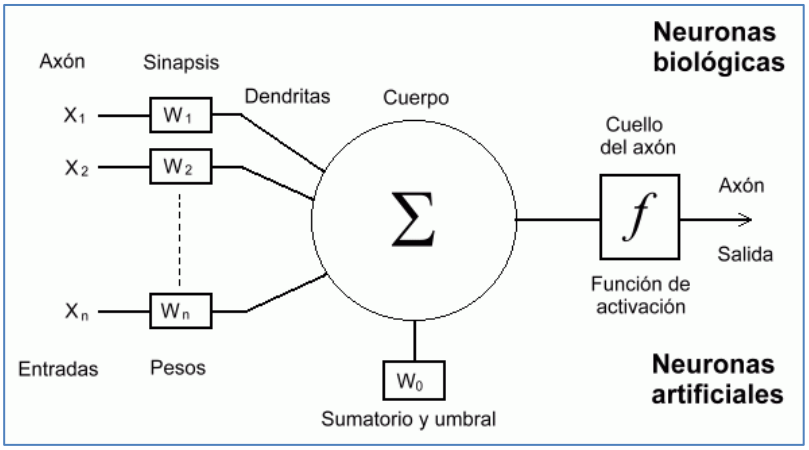


Ilustración 3: Neurona Artificial simulando a una Neurona Biológica

De este modo se puede ver gráficamente como las neuronas reciben unas entradas a través del axón, que mediante los pesos sinápticos son transmitidas al interior de la neurona calculando el valor Net; este valor debe ser procesado, calculando la función de activación, para adquirir el estado interno de la neurona en ese momento $a(t)$; que será el valor que el axón transmita como salida.

(Gestal Pose)
(Hilera González & Martínez Hernando, 1995)
(Sociedad Andaluza de Educación Matemática THALES, s.f.)

2.3.3.2. Estructura de una Red Neuronal y conexiones entre neuronas

En primer lugar, se describe de forma más detallada, la estructura de una red de neuronas artificiales. Como se ha mencionado anteriormente, existen tres tipos de neuronas según la función que desempeñen: de entrada, oculta o de salida; Cada una de estas neuronas se estructuran dentro de la red formando capas o niveles, cada nivel tiene un número determinado de neuronas. Dependiendo de la funcionalidad y la situación que ocupen las neuronas, se distinguen tres tipos de capas:

- Capa de entrada: se encuentra en contacto con las fuentes externas de la red y se encarga de recibir la información procedente del exterior.
- Capas ocultas: componen la estructura central de la red, es decir, son internas a la misma y no están en contacto con el exterior. El número de niveles ocultos no está predefinido, puede ser desde cero hasta un número elevado. Las conexiones entre las neuronas de estos niveles pueden ser de diferentes formas. Estos dos factores, se corresponden a lo que se denomina topología de la red y serán decisión del diseñador de la misma.
- Capa de salida: se corresponde con el último nivel de la red y es la encargada de transmitir el resultado de la red al exterior.

De acuerdo con el número de capas, las redes neuronales se clasifican como monocapa o multicapa.

En la siguiente figura se puede observar la estructura de una red neuronal artificial.

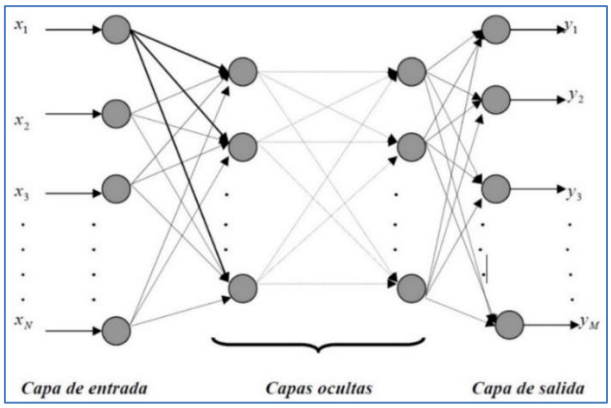


Ilustración 4: Capas de una red neuronal

A continuación, se explican las conexiones que unen a las neuronas que constituyen la red. Estas conexiones tienen asociado un peso, comentado anteriormente como fuerza sináptica, mediante el ajuste de este peso se consigue que la red adquiera conocimiento.

Consideremos y_i como el valor que recibe una neurona j en un instante determinado, esta información puede proceder del exterior, en el caso de que la neurona sea de entrada, o de la salida producida por una neurona i. Independientemente de la capa en la que se encuentre la neurona, ésta recoge un conjunto de señales que le proporcionan información o bien del exterior, en el caso de que se encuentre en la capa de entrada; o bien del estado de activación de todas las neuronas con las que esté conectada. Cada una de estas conexiones entre dos neuronas, i y j, está ponderada por un pesow_{ji}; para obtener el valor total o la entrada neta, de las todas las señales que recibe una neurona j (Net_j), se calcula el sumatorio del producto de cada señal de forma individual por el valor del peso asociado a la conexión entre las dos neuronas, es decir:

$$Net_j = \sum_i^N w_{ji}\ y_i$$

Ecuación 1: Entrada neta de una neurona

Esta forma de emitir y divulgar las señales a través de la red, es conocida como regla de propagación.

Los pesos que ponderan las conexiones entre neuronas, pueden adquirir valores negativos, positivos o nulos. Dependiendo del valor que tomen indicará el efecto que tiene una neurona sobre la otra; en el caso de que el peso entre las neuronas i y j, w_{ji}, sea positivo indica que siempre que la neurona j esta activada tiende a activar a la neurona i, sinapsis excitadora. Si por el contrario el peso toma un valor negativo la sinapsis será inhibidora, la neurona j mandará una señal para desactivar la neurona i. Finalmente, si el valor es nulo, indica que entre esas dos neuronas no existe conexión.

Una vez explicada la estructura y la existencia de conexiones, se procede a explicar las posibles formas de conexión entre neuronas. Existen dos tipos de conexión: el primero de ellos y el más habitual, es cuando la señal de salida de un nodo es la entrada de otra neurona; el segundo se conoce como conexión autorrecurrente, donde la salida que genera la neurona pasa a ser una entrada para sí misma.

Según la dirección de las conexiones entre niveles, se puede decir que existen dos tipos:

- Feedforward o propagación hacia delante: las salidas de las neuronas pertenecientes a un nivel son entradas de niveles siguientes, es decir, ninguna salida de las neuronas pasa a ser entrada de neuronas de niveles anteriores o del mismo nivel.
- Backpropagation o propagación hacia atrás: las salidas de las neuronas pueden ser transmitidas como entradas de neuronas de niveles anteriores, del mismo nivel o incluso de ellas mismas.

(Hilera González & Martínez Hernando, 1995)

2.3.3.3. Función o regla de activación

Mediante la regla de propagación, las entradas que recoge una neurona se combinan con los pesos de las conexiones; se debe realizar una regla que sea una combinación de las entradas con el estado interno o de activación actual de la neurona, para producir un nuevo estado de activación. Esta función F, se conoce como función de activación y es la encargada de calcular un nuevo estado interno o de activación de la neurona i en función del estado de activación de la misma a_i y el valor neto de las entradas Net_i.

El siguiente estado de activación $a_i(t+1)$ de una neurona U_i, se calcula aplicando la función de activación F, sobre el estado de activación de la neurona $a_i(t)$ en el momento t y la entrada total que llega a ella (Net_i).

$$a_i(t+1) = F(a_i(t), Net_i)$$

Usualmente la función de activación F es la función identidad y no se suele tener en cuenta el estado de activación anterior de la misma; por lo tanto, el parámetro que llega a la función de salida, f, de la neurona es directamente el Net_i. Debido a estos motivos, la salida de la neurona i, representada mediante y_i, se calculará según la expresión:

$$y_i = f(Net_i) = f(\sum_{j=1}^{N} w_{ij}\ y_j(t))$$

Ecuación 2: Salida de la neurona i

Por todo esto, únicamente se tiene en consideración la función f, que se denomina indistintamente función de transferencia o de activación. Por lo tanto, la función de transferencia o de activación queda de la siguiente manera:

$$y_i(t+1) = f(Net_i) = f(\sum_{j=1}^{N} w_{ij}\ y_j(t))$$

Ecuación 3: Función de transferencia o activación de la neurona i

Las funciones de transferencia o de activación típicas que se pueden dar, se observan en la siguiente tabla; donde el parámetro θ_i representa el umbral de activación de la neurona i, este umbral se considera un desplazamiento debido a las características internas de la propia neurona.

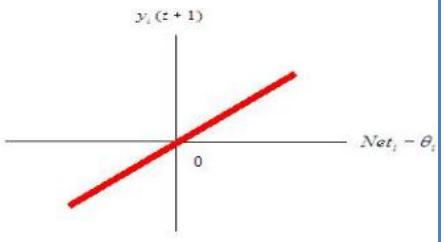
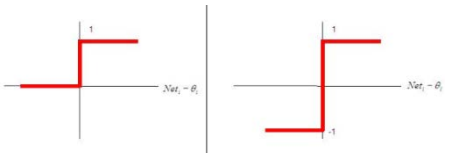
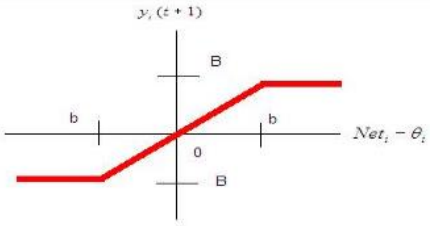
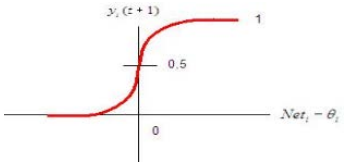
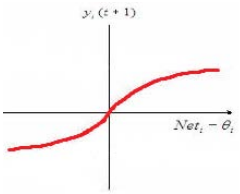
NOMBRE	FUNCIÓN	RANGO	GRÁFICA
IDENTIDAD O LINEAL	$y_i(t+1) = Net_i - \theta_i$	[-∞, ∞]	
ESCALÓN	$y_i(t+1) = \begin{cases} 1\ si\ Net_i > \theta_i \\ y(t) si\ Net_i = \theta_i \\ 0\ si\ Net_i < \theta_i \end{cases}$	(0,1)	
	$y_i(t+1) = \begin{cases} 1\ si\ Net_i > \theta_i \\ y(t) si\ Net_i = \theta_i \\ -1\ si\ Net_i < \theta_i \end{cases}$	(-1,1)	
LINEAL-MIXTA	$y_i(t+1) = \begin{cases} b\ si\ Net_i > b + \theta_i \\ Net_i - \theta_i\ si\ b - \theta_i < Net_i < B + \theta_i \\ B\ si\ Net_i \geq B \end{cases}$	(b, B)	
SIGMOIDAL	$y_i(t+1) = \frac{1}{1 + e^{-(Net_i - \theta_i)}}$	[0,1]	
FUNCIÓN TANGENTE HIPERBÓLICA	$y_i(t+1) = \frac{1 - e^{-(Net_i - \theta_i)}}{1 + e^{-(Net_i - \theta_i)}}$	[-1,1]	

Tabla 1: Funciones de transferencia o de activación

2.3.3.4. Regla de aprendizaje

El tipo de problemas que será capaz de resolver una red de neuronas artificiales, dependerá del esquema de aprendizaje de la misma; por lo tanto, el aprendizaje es la parte más importante de una red de neuronas artificiales.

El aprendizaje de las RNA está basado en ejemplos o patrones, de este modo, la capacidad de aprender depende fundamentalmente del tipo de ejemplos de los que se disponga. Para que la red aprenda de forma adecuada se debe disponer de ejemplos que cumplan las siguientes características:
Ser significativos: es necesario que se disponga de una cantidad de ejemplos considerables.
Ser representativo: el conjunto de todos los ejemplos debe ser variado. Es importante que el conjunto de ejemplos contenga toda la información que caracteriza todas las regiones del espacio de estados posibles.

El aprendizaje se realiza mediante la determinación de los valores óptimos de los pesos de todas las conexiones existentes en la red. Este proceso consiste en ajustar el valor de los pesos de las conexiones de la red, siguiendo un esquema de aprendizaje específico, en función de los ejemplos que se introducen como conjunto de aprendizaje. Este ajuste o modificación de los pesos se puede realizar tras introducir cada ejemplo o patrón del conjunto de aprendizaje o tras introducir el conjunto en su totalidad. El proceso de aprendizaje se repite hasta que se cumple un cierto criterio de convergencia.

El criterio de convergencia debe seleccionarse dependiendo del tipo de problema que se quiera resolver; el proceso de aprendizaje de la red finaliza cuando se cumple alguno de los siguientes criterios:

1. Cuando el número de ciclos de aprendizaje ha alcanzado el valor fijo determinado a priori.
2. Cuando el error producido por la red se encuentre por debajo del error establecido previamente.
3. Cuando la modificación del valor de los pesos sea irrelevante.

Se van a distinguir dos tipos de aprendizaje, dependiendo del esquema de aprendizaje y del problema que se desea resolver:

Aprendizaje supervisado

En este tipo de aprendizaje, los ejemplos o patrones que forman el conjunto de aprendizaje se componen de dos tipos de atributos: los datos propiamente dichos, que representan los atributos de entrada; e información que determina la solución al problema o la salida deseada. La forma de aprender es modificar y ajustar el valor de los pesos en función de la diferencia que se produce entre la salida obtenida por la red, al procesar el ejemplo o patrón introducido como entrada; y la salida deseada, que se corresponde con la salida disponible en el ejemplo introducido.
Se dice que, para este tipo de aprendizaje, se cuenta con profesor o experto externo que indica si el comportamiento de la red es adecuado o no, ya que modifica el valor de los pesos de las conexiones, en función de la comparación entre la salida obtenida y la deseada.

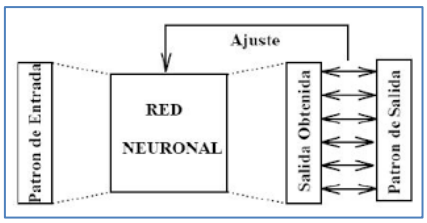


Ilustración 5: Aprendizaje supervisado

Aprendizaje no supervisado

En este tipo de aprendizaje, los ejemplos del conjunto de datos de aprendizaje o entrenamiento únicamente cuentan con los datos propiamente dichos, no existe ningún tipo de información referente a la salida que se debe alcanzar. Por lo tanto, la forma en la que la red modifica y ajusta los pesos es en función a información interna; esta información se obtiene a partir de las particularidades existentes en el conjunto de datos de aprendizaje: rasgos significativos,

regularidades o redundancias. Los modelos con este tipo de sistema de aprendizaje también son conocidos como sistemas autoorganizados, ya que no se posee ningún experto externo que indique la solución a alcanzar, sino que deben modificar y ajustar el valor de los pesos únicamente con la información presente en los ejemplos recibidos como entrada.

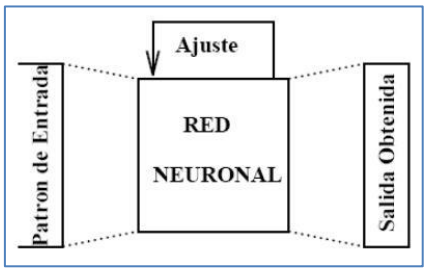


Ilustración 6: Aprendizaje no supervisado

(Isasi Viñuela & Galván León, 2004)

2.3.4. Diseño, creación y desarrollo de una Red de Neuronas Artificiales

El diseño, la creación y el desarrollo de una red se divide en fases, que comienza con el diseño de la topología de la red y finaliza con la ejecución de la red sobre datos con los que no ha entrenado. Estas fases se pueden dividir en:

- Diseño de la arquitectura de la red.
- Entrenamiento.
- Test o prueba.

2.3.4.1. Diseño de la arquitectura de la red

Esta fase también es conocida como diseño de la topología de la red, que implica la determinación de la función de activación a emplear, el número de neuronas y el número de capas de la red. Esta etapa es crítica, ya que de ella depende la cantidad de conocimiento que va albergar la red.

La elección de la función de activación se suele hacer basándose en el recorrido que se desea que realice la red. El hecho de elegir una u otra, generalmente no influye en la capacidad de la red para resolver el problema.

En lo que respecta al número de neuronas y capas, algunos de estos parámetros vienen dados por el problema, como son el número de neuronas de la capa de entrada y de la capa de salida que lo determinan las variables del conjunto de datos; otros parámetros deben ser elegidos por el diseñador, como son el número de capas ocultas y la cantidad de neuronas en cada una de ellas. No existe un método o regla que determine el número óptimo de neuronas ocultas para resolver un problema dado, por lo que el diseñador debe recurrir a la experiencia y a la técnica de prueba y error.

2.3.4.2. Entrenamiento

Una vez diseñada la arquitectura de la red y las funciones que operará, se debe entrenar la red para que aprenda a dar la respuesta adecuada al conjunto de datos proporcionados.

El objetivo de la fase de entrenamiento, es conseguir que la red recoja la suficiente información de los datos empleados para entrenar y obtener la capacidad de generalizar; es decir, que frente a un conjunto de datos no conocidos (con los que no haya entrenado) sea capaz de obtener salidas adecuadas.

Esta capacidad de generalización y obtención de resultados adecuados, se obtiene mediante el ajuste de los pesos de las conexiones de la red.

2.3.4.3. Test o prueba

Tras la fase de entrenamiento viene la fase de test, en la que se pedirá a la red que responda a estímulos (datos) diferentes a los presentados en la fase de entrenamiento. De este modo se podrá comprobar si la red ha sido capaz de generalizar y conseguir buenos resultados, frente a datos desconocidos.

El proceso de test es el mismo que el realizado en el de entrenamiento, solo que en esta fase no se realiza ningún ajuste en los pesos de las conexiones.

Se pueden distinguir diferentes tipos de redes neuronales en función de la forma en la que se organizan las neuronas, como aprenden o el número de capas: Perceptrón Simple, Adaline, Perceptrón Multicapa, redes de base radial, etc. En este trabajo se ha empleado el Perceptrón Multicapa que se explica con más detalle en la siguiente sección, donde se comentan las técnicas utilizadas para llevar a cabo la investigación.

2.4. Técnicas utilizadas

2.4.1. Perceptrón Multicapa

Debido a las limitaciones del Perceptrón simple para solucionar problemas no lineales, surge el Perceptrón Multicapa, que es una generalización del Perceptrón simple. En 1969, Minsky y Papert, mostraron que la combinación de varios perceptrones simples podrían alcanzar una solución frente a problemas no lineales, pero aún no existía un mecanismo automático para adaptar los pesos de las capas ocultas. Sin embargo, esta idea de combinar varios perceptrones sirvió de base para estudios posteriores realizados por Rumelhart, Hinton y Willians en 1986. Fueron estos autores los que presentaron la regla delta generalizada para adaptar los pesos propagando el error cometido hacia atrás (retropropagación), para funciones de activación no lineales y redes multicapa.

El Perceptrón Multicapa (MLP) está capacitado para aproximar cualquier función continua sobre el espacio de $\mathcal{R}^n$, lo que indica que el MLP es un aproximador universal.

Hoy en día, el MLP es una de las arquitecturas más empleadas en resolución de problemas dentro del marco de las redes neuronales; debido principalmente a su facilidad de uso, a su aplicabilidad y a su capacidad como aproximador universal. Por todo ello, se ha decidido emplear esta técnica en el presente trabajo, para reducir la dimensionalidad de los datos originales.

Las principales características por las que se ha decidido aplicar este modelo son:

- La salida puede tomar cualquier valor; ya sea discreta, real o un vector de valores tanto reales como discretos.
- Los ejemplos o patrones pueden ser independientes o estar correlados.
- El MLP es muy robusto al ruido, por lo que los ejemplos de entrenamiento pueden contener errores.

Aunque sea una de las redes más empleadas también posee una serie de limitaciones, para problemas complejos dependientes de una gran cantidad de variables el proceso de aprendizaje es largo; en ocasiones codificar problemas reales mediante valores numéricos es complicado; entre algunos otros.

En los siguientes apartados, se exponen las particularidades de este tipo de redes de neuronas artificiales.

2.4.1.1. Arquitectura del Perceptrón Multicapa

La arquitectura del MLP está dividida en niveles o capas, cada una de estas capas cuenta con un número determinado de neuronas y se diferencian tres tipos de capas diferentes: capa de entrada, capas ocultas y capa de salida; explicadas anteriormente en el apartado *2.3.3.2.Estructura de una Red Neuronal y conexiones entre neuronas*.

Las neuronas de la capa de entrada no actúan como neuronas propiamente dichas, se les denomina en muchas ocasiones nodos de entrada, ya que son las encargadas de recibir las entradas procedentes del exterior y propagarlas a todas las neuronas de la capa siguiente. Las neuronas que se encuentran en la última capa, las neuronas de salida, proporcionan al exterior la respuesta que obtiene la red para un determinado patrón de entrada. Las encargadas de realizar un procesamiento no lineal de los patrones recibidos, son las neuronas de las capas ocultas.

La característica de las conexiones entre las neuronas del Perceptrón Multicapa es que siempre está dirigidas hacia delante, es decir, las neuronas pertenecientes a una capa están conectadas con todas las neuronas de la siguiente capa, es por eso que también reciben el nombre de redes conectadas hacia delante o redes "feedforward". Debido a que todas las neuronas de una capa están conectadas con todas las neuronas de la capa siguiente, se dice que existe una conectividad total o que la red está totalmente conectada. Todas las conexiones entre neuronas llevan asociado un peso de la conexión, que toma como valor un número real; y todas las neuronas de la red, a excepción de las neuronas de entrada, llevan asociado un umbral.

2.4.1.2. Propagación de los patrones de entrada

El Perceptrón Multicapa define una relación entre las variables que recoge como entrada y las variables que obtiene como salida la red, esta relación se obtiene propagando los valores de entrada que recibe la red hacia delante. Para realizar esta propagación, cada neurona de la red recibe y procesa la información que recoge por sus entradas y produce una respuesta o activación que transmite hacia las neuronas agrupadas en la siguiente capa. A continuación, se muestran las expresiones concretas empleadas por el Perceptrón para calcular las activaciones de las neuronas de la red.

Partiendo de un Perceptrón Multicapa con C capas (una capa de entrada, $C-2$ capas ocultas y una capa de salida) y n_c neuronas en la capa c, para $c = 1, 2, \dots, C$. Sea $W^c = (w_{ij}^c)$ la matriz de pesos asociada a las conexiones existentes entre la capa c y la siguiente capa $c+1$, donde w_{ij}^c representa el peso de la conexión entre la neurona i de la capa c y la neurona j de la capa $c+1$; y sea $U^c = (u_i^c)$ el vector que contiene los umbrales de las neuronas de la capa c, para $c = 2, 3, \dots, C$, ya que las neuronas de entrada no actúan como neuronas normales y por lo tanto no tienen umbral asociado. Se denota por a_i^c a la activación de la neurona i de la capa c; estas activaciones se calculan de distinta manera dependiendo de la capa en la que se encuentre la neurona:

- Activación de las neuronas de la capa de entrada a_i^1. Las neuronas de la capa de entrada se encargan de propagar hacia la red las señales recibidas del exterior. Por lo tanto:

  $$a_i^1 = x_i \, para \; i = 1, 2, \dots, n_1$$

  donde $X = (x_1, x_2, \dots, x_{n_1})$ representa el vector o patrón de entrada de la red.

- Activación de las neuronas de la capa oculta c a_i^c. Las neuronas ocultas de la red se encargan de procesar la información, para ello, calculan la función de activación f al sumatorio de los productos de las activaciones que recibe por sus correspondientes pesos más el umbral, es decir:

$$a_i^c = f\left(\sum_{j=1}^{n_c-1} w_{ji}^{c-1}\ a_j^{c-1} + u_i^c\right) para\ i = 1,2,\dots,n_c\ y\ c = 2,3,\dots,C-1$$

Ecuación 4: Activación de las neuronas de la capa oculta c

donde a_j^{c-1} son las activaciones que reciben de las neuronas de la capa anterior, en el caso de que la capa anterior sea la capa de entrada se corresponde con el valor de entrada.

- Activación de las neuronas de la capa de salida a_i^C. Realizan el mismo calculo que ejecutan las neuronas de las capas ocultas, la diferencia es que en este caso la activación se corresponde con la salida de la red y reciben las activaciones de la última capa oculta, por lo tanto:

$$y_i = a_i^C = f\left(\sum_{j=1}^{n_C-1} w_{ji}^{C-1}\ a_j^{C-1} + u_i^C\right) para\ i = 1,2,\dots,n_C$$

Ecuación 5: Activación de las neuronas de la capa de salida

donde $Y = (y_1, y_2, \dots, y_{n_c})$

La función f, es la llamada función de activación explicada en el apartado *2.3.3.3.Función o regla de activación*. Para el Perceptrón Multicapa, las funciones de activación más empleadas son la función sigmoidal y la función tangente hiperbólica. Estas poseen como imagen un rango continuo de valores dentro de los intervalos [0,1] y [-1,1] respectivamente y son funciones crecientes con dos niveles de saturación: el máximo y el mínimo. Como ya se ha indicado vienen dadas por las siguientes expresiones:

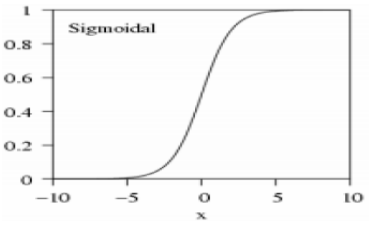
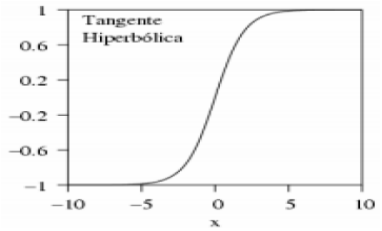
	FUNCIÓN	RANGO	GRÁFICA
SIGMOIDAL	$f_{sigm} = \left(\frac{1}{1+e^{-x}}\right)$	[0,1]	
FUNCIÓN TANGENTE HIPERBÓLICA	$f_{tag} = \left(\frac{1-e^{-x}}{1+e^{-x}}\right)$	[-1,1]	

Tabla 2: Funciones de activación MLP

La elección de la función de activación queda a cargo del diseñador de la red, esta elección suele realizarse basándose en los valores de activación que se deseen que obtengan las neuronas. Lo habitual es que todas las neuronas de la red realicen la misma función de activación.

Por todo lo visto en este apartado, se entiende que el MLP define, a través de sus conexiones y neuronas, una función continua no lineal del espacio $\mathcal{R}^{n_1}$ (espacio de ejemplos o patrones de entrada) al espacio $\mathcal{R}^{n_C}$ (espacio de patrones de salida).

2.4.1.3. Aprendizaje/ Algoritmo de retropropagación

El aprendizaje que realiza el Perceptrón Multicapa es un proceso iterativo supervisado. Es iterativo porque mientras la salida que obtiene la red para cada patrón de entrenamiento no sea próxima a la salida deseada, se realiza un ajuste sucesivo de los parámetros de la red (pesos y umbrales). Es un algoritmo de aprendizaje supervisado, ya que, para cada patrón de entrada a la red es necesario proporcionar un patrón de salida deseada.

El aprendizaje que realiza el MLP, es equivalente a minimizar el error existente entre la salida obtenida por la red y la salida deseada proporcionada; este aprendizaje se realiza mediante el error medio producido por todos los patrones de la red (E), que se calcula con el error producido por cada patrón n($e(n)$). Sus expresiones son:

$$E = \frac{1}{N}\sum_{n=1}^{N} e(n)$$

Ecuación 6: Error medio producido por la red

$$e(n) = \frac{1}{2}\sum_{i=1}^{n_C} (s_i(n) - y_i(n))^2$$

Ecuación 7: Error cuadrático medio producido por cada patrón

donde N representa el número de patrones o muestras, $e(n)$ es el error cometido por la red para el patrón n. Siendo $Y(n) = (y_i(n), \dots, y_{nC}(n))$ el vector con las salidas producidas por la red y $S(n) = (s_i(n), \dots, s_{nC}(n))$ el vector de salidas deseadas que han sido proporcionadas.

De esta forma y basándose en hallar el mínimo de la función de error, es como el Perceptrón multicapa realiza el aprendizaje. El Perceptrón multicapa realiza su aprendizaje mediante el ajuste de sus parámetros (pesos y umbrales) en base al mínimo de la función de error. Las salidas que genera la red como respuestas son no lineales debido a la presencia de funciones de activación no lineales, por lo que se emplean técnicas no lineales para ajustar los parámetros. La técnica que emplea el algoritmo de retropropagación del MLP para encontrar el mínimo de la función es el conocido como método de descenso de gradiente, pues tomará siempre la dirección en la que decrece la función de error.

Aunque lo normal es que el aprendizaje de la red se realice para minimizar el error total cometido por la misma, E; la forma habitual desempeñada por el MLP es basándose en métodos del gradiente estocástico, los cuales van minimizando el error para cada patrón $e(n)$. De este modo, aplicando el método de descenso de gradiente estocástico, para cada patrón de entrada n se modifica cada parámetro w de la red en base a la siguiente ley de aprendizaje:

$$w(n) = w(n-1) - \alpha \frac{\partial e(n)}{\partial w}$$

Ecuación 8: Ley de aprendizaje (descenso de gradiente estocástico)

donde $e(n)$ es el error cometido por el patrón n y α es la razón o tasa de aprendizaje; el parámetro α controla la proporción de desplazamiento sobre la superficie del error siguiendo la dirección negativa del gradiente. Otorgando valores altos a la tasa de aprendizaje, avanza rápidamente en la superficie del error, pero puede saltarse un mínimo; por otro lado, valores pequeños evitan estos problemas, pero la convergencia es más lenta.

Como se puede ver en la ilustración 7, es complicado encontrar el mínimo absoluto o global en la función de error. Por ese motivo se emplea el descenso de gradiente, que va adaptando los parámetros, siguiendo la dirección negativa del gradiente de la función de error. Se puede ver como si fuera "saltando" por la función de error hasta encontrar un mínimo local con la ayuda de la tasa de aprendizaje; es decir, busca pendientes pronunciadas, pero no siempre encontrará el mínimo absoluto o global de la función de error.

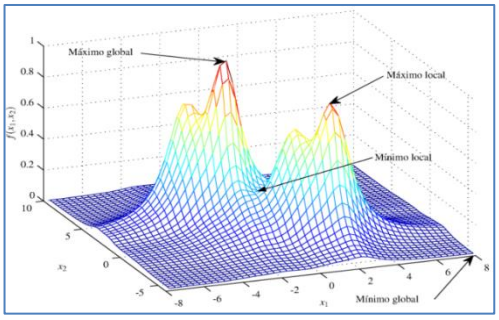


Ilustración 7: Función de error

La arquitectura del MLP cuenta con neuronas agrupadas en capas, por lo que el algoritmo de descenso de gradiente debe aplicarse de forma eficiente, que llegue a todas las neuronas de todas las capas; es lo que se conoce algoritmo de retropropagación o regla delta generalizada, explicada a continuación. El término retropropagación se emplea a la forma de aplicar el método del gradiente, ya que el error cometido por la red se propaga hacia atrás de capa en capa y lo transforma en un error para cada una de las neuronas ocultas de la red.

Regla delta generalizada

La arquitectura del Perceptrón Multicapa cuenta con la presencia de neuronas ocultas para las que no se conoce el error que han cometido, ya que se encuentran en el interior de la red; por ello, es necesario generalizar la regla delta para poder propagar hacia atrás el error cometido por la red en la salida. La regla delta generalizada funciona del siguiente modo:

La red obtiene una salida de la cual se sabe el error cometido, ya que al tratarse de un algoritmo de aprendizaje supervisado se realiza una comparación entre la salida obtenida y la deseada; cada neurona de salida propaga hacia atrás su error cometido, este error se transmite a todas las neuronas de la capa anterior con las que esté conectada y se pondera por el valor del peso de la conexión. De esta forma las neuronas de la capa oculta reciben un valor de error cometido por cada neurona de salida ponderado por el peso de la conexión, el sumatorio de todos los errores cometidos pasa a ser el error de la neurona oculta. Esta técnica se realiza sucesivamente hasta llegar a la primera capa oculta, dotando a todas las neuronas de las capas ocultas el error cometido por cada una de ellas.

Proceso de aprendizaje del Perceptrón Multicapa

Como ya se ha indicado, el aprendizaje o entrenamiento que realiza el Perceptrón Multicapa consiste en ir ajustando los parámetros de la red, tanto los pesos como los umbrales, con el objetivo de minimizar la función de error E; es decir, que para las entradas introducidas la red sea capaz de producir las salidas deseadas. A continuación, se exponen los pasos que sigue el proceso de aprendizaje completo, para así entender el funcionamiento del mismo. Los pasos que componen el proceso de aprendizaje son los siguientes:

Paso 1: Se inicializan los pesos y umbrales de la red, generalmente de forma aleatoria y con valores próximos a cero.
Paso 2: Se presenta un patrón n de entrenamiento, con su vector de entrada y su vector de salida; el vector de entrada se propaga hacia la salida, calculándose de este modo la respuesta de la red para dicha entrada (vector de salidas obtenidas por la red para el patrón n).
Paso 3: Se calcula para el patrón n el error cuadrático $e(n)$ cometido por la red.
Paso 4: Para ajustar los parámetros de la red en función del error cometido, se aplica la regla delta generalizada. Como ya se ha mencionado, se deben seguir los siguientes pasos:

> **Paso 4.1:** Se calcula el error cometido para todas las neuronas de la capa de salida.
> **Paso 4.2:** Se calcula el error producido para el resto de neuronas de la red, empezando desde la última capa oculta y retropropagando dichos valores hasta la capa de entrada.

Paso 4.3: Se modifican pesos y umbrales en función del error cometido por cada neurona.

Paso 5: Se repiten los pasos 2, 3 y 4 para todos los patrones de entrenamiento, completando así un ciclo de aprendizaje.

Paso 6: Se evalúa el error total E cometido por la red.

Paso 7: Se repiten los pasos 2, 3, 4, 5 y 6 hasta alcanzar un mínimo del error de entrenamiento o hasta cumplir el criterio de convergencia especificado; para lo cual se realizan m ciclos de aprendizaje.

De forma intuitiva, partiendo de un punto aleatorio del espacio de pesos, el proceso de aprendizaje desplaza el vector de pesos siguiendo la dirección negativa del gradiente del error en dicho punto; alcanzando un nuevo punto que estará más próximo al mínimo de la función de error que el punto anterior. Como se muestra en la siguiente ilustración.

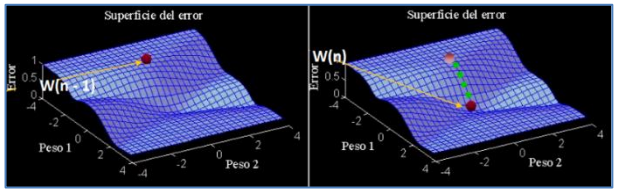


Ilustración 8: Aprendizaje MLP, actualización de los pesos en función del error

Desde un punto de vista teórico, el aprendizaje se finaliza cuando $\frac{\partial E}{\partial w} \approx 0$, que se corresponde con el momento en el que los parámetros de la red no cambian de una iteración a otra. Sin embargo, desde un punto de vista práctico se suelen fijar un número de ciclos de aprendizaje. El análisis se centra únicamente en observar si el error cometido por la red se mantiene prácticamente constante de una iteración a otra o si sigue descendiendo.

Capacidad de generalización

A la hora de evaluar el comportamiento de una red no sólo es importante saber si la red ha aprendido con éxito los patrones de entrenamiento utilizados en la dicha fase; sino que es imprescindible conocer el comportamiento de la red ante patrones que no se han utilizado en la fase de aprendizaje, patrones de test. De nada sirve que la red haya adaptado a los patrones de entrenamiento y no responda adecuadamente ante patrones nuevos. Es necesario que durante el aprendizaje la red extraiga las características de las muestras, para responder correctamente a nuevos patrones. Esto se conoce como la capacidad de generalización de la red.

En la siguiente ilustración se puede observar un ejemplo de una red con buena capacidad de generalización (A) y un ejemplo de mala generalización (B). La función representada mediante la línea discontinua es la función real, mientras que la función obtenida por la red se representa mediante la línea continua.

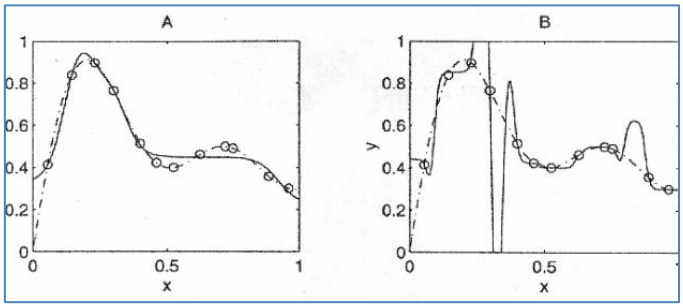


Ilustración 9: Capacidad de Generalización de una red

Se puede observar como en el caso de que la función tenga buena capacidad de generalización, gráfica A, se dan casos donde los puntos deseados y obtenidos no se corresponden, pero la función obtenida es muy similar a la función deseada. Por el contrario, en el caso de la mala generalización,

gráfica B, la red suele pasar por los puntos con los que ha entrenado pero la función obtenida no se asemeja a la deseada.

Por este motivo es imprescindible evaluar la capacidad de generalización de una red. Para ello es necesario disponer de dos conjuntos de patrones, uno para entrenar y modificar los pesos y umbrales (conjunto o patrones de entrenamiento) y otro para medir la capacidad de la red para responder correctamente ante patrones nuevos (conjunto o patrones de test). Es conveniente exigir un menor aprendizaje de la red sobre los patrones de entrenamiento, con el objetivo de obtener mejores propiedades de generalización.

Cuando una red se ajusta muy bien a los patrones de entrenamiento, pero no se adapta adecuadamente a los patrones de test se dice que se produce un sobreaprendizaje, sobreajuste u overfitting. Este hecho puede producirse por un número elevado de ciclos de entrenamiento y también por una gran cantidad de neuronas ocultas.

Deficiencias del algoritmo de aprendizaje a tener en cuenta

Mínimos locales:

Como puede apreciarse en la *Ilustración 8: Aprendizaje MLP, actualización de los pesos en función del error*, la superficie del error E en función de los parámetros de la red, es compleja y llena de valles y colinas. El método de descenso de gradiente actualiza el valor de los pesos según la dirección negativa del error, se corre el riesgo de que el proceso finalice en un mínimo local en lugar de en el mínimo global o absoluto.

Para solucionar este problema se pueden desempeñar diferentes técnicas: añadir un mayor número de neuronas ocultas para otorgar a la red de una mayor cantidad de parámetros libres (pesos y umbrales), dotándola de una mayor representatividad del problema; otra posible solución es emplear una tasa de aprendizaje que vaya decreciendo a lo largo del aprendizaje, de este modo al principio del aprendizaje el desplazamiento sobre la función de error sería grande, dando mayor posibilidad a alcanzar el mínimo global, e iría reduciéndose el desplazamiento según avanza el aprendizaje; otra solución es volver a inicializar los parámetros de la red para partir el entrenamiento desde otro punto o añadir ruido al descenso de gradiente.

Parálisis o saturación:

Esto es causado cuando la entrada total o entrada neta a una neurona de la red adquiere valores elevados, tanto negativos como positivos; ya que las funciones de activación poseen asíntotas horizontales, como puede verse en la *Tabla 2: Funciones de activación MLP*. Si el valor de la entrada neta de la neurona toma valores muy altos, se satura, y se alcanza un valor de activación máximo; como el ajuste de los parámetros de la red es proporcional a las activaciones de las neuronas de salida, si la neurona de salida está saturada la modificación de los parámetros es casi inexistente y la suma de los errores locales se mantiene constante por un periodo largo de tiempo; pudiendo dar a entender que la red a alcanzado un mínimo local cuando lo que se está produciendo es una invariación en el ajuste de los parámetros. Para solucionar este problema se suele inicializar los parámetros iniciales con valores próximos a cero, ya que si adquieren valores muy altos se saturan las neuronas.

Por lo tanto, como puede verse en la siguiente imagen, mientras el valor de las activaciones sea máximo, no se produce ningún cambio relevante en los parámetros de la red y por lo tanto el error no disminuye, dando pie a pensar que se ha encontrado un mínimo local, pero como puede verse según se van alejando las activaciones del valor máximo, el error va decrementando.

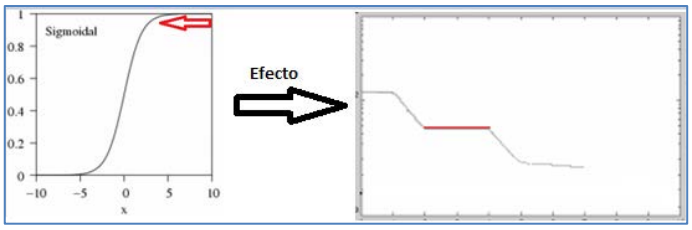


Ilustración 10: Fenómeno de saturación o parálisis

(Isasi Viñuela & Galván León, 2004)

2.4.2. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) o en español Análisis de Componentes Principales es una técnica estadística para sintetizar la información, o de reducción de la dimensionalidad de un conjunto de datos, es decir, reducción del número de variables que representan un dato. Por lo tanto, la finalidad de esta técnica es que, partiendo de un conjunto de datos representados por una gran cantidad de atributos o variables, reducir el número de variables que representan cada dato al menor número viable, perdiendo la menor cantidad de información relevante posible.

Fue primeramente introducido por Pearson en 1901 y desarrollado en 1933 por Hotelling, y la primera implementación computacional se dio en los años 60. Fue aplicado para analizar encuestas de opinión pública por Jean Pages. Como ya se ha mencionado el objetivo es construir un número inferior de nuevas variables, denominadas componentes, en las cuales se concentre la mayor cantidad posible de información.

Los componentes principales son linealmente independientes y se calculan como una combinación lineal de las variables originales que representan cada dato. Mediante esta transformación lineal, lo que se está realizando es un cambio a un nuevo sistema de coordenadas para el conjunto de datos original; en este nuevo sistema, la varianza[1] de mayor tamaño del conjunto de datos se establece en el primer eje (que se corresponde con el primer componente principal), la segunda varianza con mayor valor en el segundo eje, y así sucesivamente hasta completar la dimensión de los datos originales.

Lo realmente interesante de este modelo, es que realiza un estudio de las relaciones que existen entre las variables *m* correlacionadas, es decir, variables que miden o representan información común; y las transforma linealmente en otro conjunto de variables nuevas, ordenadas según la cantidad de información relevante, llamado conjunto de componentes principales; Las nuevas variables o componentes principales se transforman de tal manera que están incorreladas entre sí, no poseen redundancia en la información, y la varianza en cada uno de ellos esta maximizada.

Es importante destacar que el concepto de mayor información está relacionado con el de mayor varianza o variabilidad, cuanto mayor sea la variabilidad de la variable que representa a un dato (varianza) se considera que posee mayor cantidad de información. Es por esto, que este método basa su funcionamiento en las covarianzas[2] de los datos. Realiza el análisis de componentes principales tiene sentido si existen altas correlaciones entre las variables, ya que esto indica que existe información redundante y, por lo tanto, que la información se puede agrupar en un menor número de variables.

A continuación, se va a proceder a explicar el algoritmo empelado por PCA, cuyo objetivo es transformar un conjunto de datos dados, **X**, de dimensión *n x m*; a otro conjunto de datos **Y** de igual

[1] La varianza es el promedio de los cuadrados de las desviaciones de los datos con respecto a la media, miden la cantidad de variación existente en un conjunto de datos.
[2] La covarianza indica el grado de variación entre variables, determina las relaciones entre las variables que representan un conjunto de datos.

dimensión *n x m*, donde los primeros componentes principales de la lista, contendrán mayor cantidad de información; de este modo se podrá seleccionar un número inferior de variables *l*, siendo *l* menor que *m* lógicamente, donde se produzca la menor pérdida de información útil posible utilizando para ello la matriz de covarianza.

Se parte de un conjunto de datos para el análisis formado por *n* filas que representan las *n* muestras o ejemplos, cada una de las cuales está constituida por *m* columnas que forman las variables que describen cada ejemplo; se pretende que cada una de esas muestras, se describa con *l* variables correspondiente a los *l* primeros componentes principales, siendo *l* menor que *m*. Además, el número de componentes principales *l* seleccionados debe que ser inferior a la menor de las dimensiones de **X**, ya que lo que se busca es reducir la dimensión de los datos. Estos datos deben de estar centrados a media 0 (esto se calcula restándoles la media de cada columna) y preferentemente autoescalados (centrados en media cero y dividiendo cada columna por su desviación estándar[3]), esto se realiza mediante la siguiente ecuación:

$$X = \sum_{a=1}^{l} t_a \, p_a^T + E$$

Ecuación 9: Ecuación para el análisis de datos PCA

Donde se cuenta con dos conjuntos de vectores ortogonales[4], uno conocido como scores, t_a, que contiene la información de las relaciones existentes entre las muestras o ejemplos; y otro conocido como loadings, p_a, que informan de la relación existente entre las variables de dichas muestras. Como se están seleccionando menos componentes principales que variables, se produce un error en el ajuste del modelo con los datos que se acumula en la matriz **E**.

El modelo PCA está basado en la descomposición de la matriz de covarianza en vectores propios; los vectores propios son vectores no nulos que, al transformarlos por el operador dan lugar a un múltiplo escalar de sí mismos, por lo que no cambian su dirección. Esto se calcula del siguiente modo:

$$cov(X) = \frac{X^T X}{n-1}$$

$$cov(X) p_a = \lambda_a \, p_a$$

$$\sum_{a=1}^{m} \lambda_a = 1$$

Ecuación 10: Descomposición de la matriz en vectores propios

Donde se dice que p_a es el vector propio de la matriz de covarianza $cov(X)$ asociado al valor propio λ_a. Para concluir, en la siguiente ecuación, se entiende que t_a son las proyecciones de **X** en p_a:

$$t_a = \boldsymbol{X} \, p_a$$

Ecuación 11: Proyecciones de X en la matriz p_a

Los valores propios λ_a, recopilan la información que representa a cada uno de los componentes, es decir, miden la cantidad de varianza capturada para cada componente. Esta cantidad de información capturada para cada componente, va disminuyendo según su número; por lo que el componente principal número uno representa más información que el dos y así repetidamente.

Se puede ver en la siguiente ilustración el funcionamiento del modelo PCA de una forma gráfica; cómo se calculan los componentes principales (CP1 que es componente que maximiza la varianza y

[3] La desviación estándar es la raíz cuadrada de la varianza de la variable.
[4] Dos vectores son ortogonales si su producto escalar es cero.

CP2 que recoge el resto de la variabilidad), de una nube de puntos que se corresponden con el conjunto de datos dados (matriz **X**), en este caso de dos dimensiones; se tendrán tantas componentes principales como variables tenga cada dato.

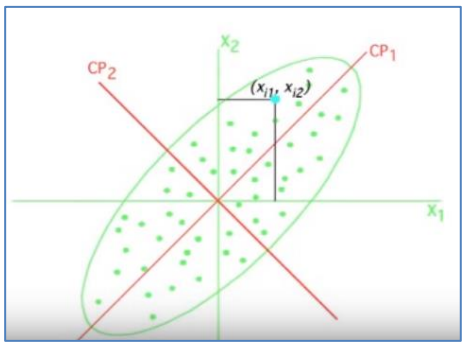


Ilustración 11: Obtención de los componentes principales

Una vez que se obtienen los componentes principales se realiza un cambio al nuevo sistema de coordenadas marcado por los componentes principales, habiendo realizado una transformación lineal en los datos originales como se ve en la siguiente ilustración.

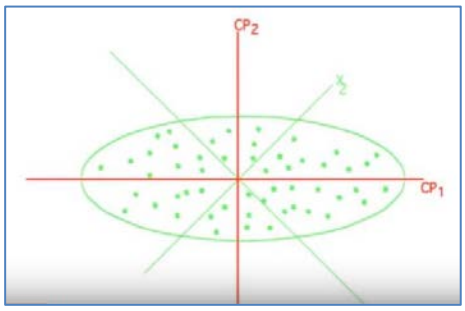


Ilustración 12: Rotación y transformación lineal

(Vicente Villardón, 2013)
(González García & Taborda Londoño, 2015)
(Gorgas & Cardiel, 2010/2011)

2.4.3. K-Nearest Neighbors (KNN)

EL algoritmo K-Nearest Neighbors o K-vecinos más cercanos, más conocido como KNN, fue desarrollado en 1951 por Fix y Hodges. Ha sido el algoritmo seleccionado en este trabajo para evaluar el comportamiento de los datos tras la reducción de la dimensionalidad de los mismos.

El algoritmo KNN es una de las aproximaciones más populares basadas en criterios de vecindad. Al estar basado en la vecindad, es necesario establecer cierta medida de similitud o distancia entre los diferentes elementos de representación, es decir, para comparar la distancia entre los distintos valores u objetos es imprescindible definir una métrica determinada.

La ventaja más inmediata que presentan las técnicas basadas en la vecindad, es su simplicidad conceptual, que se puede resumir del siguiente modo: el valor de salida que se otorgará a un nuevo punto del espacio de representación, se calcula en función de los valores de los puntos más cercanos a él. Otra peculiaridad, es que es un modelo no paramétrico, es decir, no realiza suposiciones observando la distribución de los datos, sino que son los propios datos los que la determinan. Otra característica a destacar es que se trata de un método local, asume que la salida de un nuevo dato únicamente depende de los k vecinos de entrenamiento más próximos.

2.4.3.1. Algoritmo

El modelo KNN es un método de aprendizaje inductivo supervisado, considerado perteneciente a la familia del aprendizaje perezoso, ya que no necesita realizar una fase de entrenamiento; los datos de partida, que se denominan indistintamente datos de entrenamiento, aunque no realicen esta fase, se encuentran almacenados a priori; y los nuevos datos de entrada o datos de test se valoran buscando las k muestras de los datos almacenados a priori más cercanas a él.

El modelo KNN se trata de un algoritmo sencillo, que puede ser utilizado tanto para tareas de clasificación como de regresión, que es el objetivo de este trabajo. En lugar de determinar la clase a la que pertenece la nueva instancia en función de la clase mayoritaria de los vecinos más cercanos del conjunto de entrenamiento (clasificación); debe determinar el valor del nuevo dato como el valor medio de los k ejemplos de entrenamiento más cercanos, es decir, siguiendo la siguiente ecuación:

$$Valor(P_{entrada}) = \frac{1}{K}\sum_{i=1}^{K} Valor(P_i)$$

Ecuación 12: Valor del nuevo patrón de entrada (KNN)

Para determinar cómo de cercanos se encuentran los ejemplos unos de otros, debe utilizarse cierta medida de similitud o distancia. Este cálculo debe realizarse para todos los ejemplos nuevos de entrada contra todo el conjunto de entrenamiento. Por lo que el tiempo de respuesta es elevado. Además, se necesita que la función para determinar la distancia entre los ejemplos sea la adecuada, generalmente la ecuación empleada es la distancia Euclídea, definida mediante la siguiente ecuación:

$$d(p,q) = \sqrt{\sum_{i}^{n}(p_i - q_i)^2}$$

Ecuación 13: Distancia euclídea

Por lo tanto, se determina que el algoritmo KNN sigue los siguientes pasos:

Paso 1: Se almacena el conjunto de datos de entrenamiento, compuesto por un vector de entrada y un vector de salida.
Paso 2: Se establece el valor del parámetro k.
Paso 3: Se presenta un patrón n nuevo o de test, únicamente teniendo en cuenta el vector de entrada de este nuevo dato.

- **Paso 3.1:** Se calcula la distancia euclídea del nuevo patrón n con todos los datos del conjunto de entrenamiento.
- **Paso 3.2:** Se calcula la salida del nuevo dato como la media de las salidas de los k datos más cercanos a él.

Paso 4: Se repite el paso 3 para todos los patrones de test.

Para concluir con la explicación del funcionamiento del método KNN, se muestra un ejemplo gráfico de cómo se buscan los tres vecinos más cercanos (**I**, **D** y **N**) al nuevo punto introducido, representado en la imagen como una **X**. El valor de salida que el modelo le dará al nuevo punto será la media de los valores de los puntos vecinos **I**, **D** y **N**.

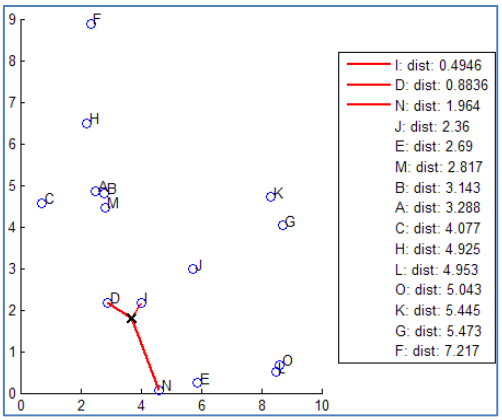


Ilustración 13: Ejemplo distancia Euclídea en KNN

2.4.3.2. Elección del parámetro k

El punto clave del modelo KNN es elegir el valor que se le va a dar a la variable K, es decir, el número de vecinos más cercanos con los que se va a realizar la media para obtener la salida del nuevo patrón. Se considera un punto crítico, ya que, si el valor de K es demasiado pequeño, el resultado que producirá el modelo es muy sensible a puntos ruidosos. Pero si por el contrario se le otorga a K un valor demasiado grande, el ruido va perdiendo influencia, pero se pierde la idea de localidad y los vecinos que se encuentran alejados comienzan a influir sobre un dato con el que puede que no tengan relación.

2.4.3.3. Deficiencias del algoritmo

Sensible al ruido y a datos irrelevantes:

Una de las desventajas que presenta, es en el momento que los datos poseen ruido o son irrelevantes, ya que el algoritmo se muestra muy sensible y no produce buenos resultados. Es importante tenerlo presente y está relacionado con el objetivo del proyecto, ya que, al reducir la dimensión de los datos iniciales se pretende eliminar los datos con ruido e irrelevantes del conjunto.

Lentitud:

Otro inconveniente que presenta es la lentitud cuando se cuentan con un conjunto de datos grande, ya que para cada patrón de entrada nuevo se debe calcular la distancia a todos los datos de entrenamiento almacenados previamente.

(Sierra Araujo, 2006)
(Bedoya Puerta, 2011)

2.5. Procesado del conjunto de datos

Todas las técnicas empleadas en este proyecto, tanto para reducir la dimensionalidad de los datos como para analizar el comportamiento, parten de un conjunto de datos representativos del problema, también denominados muestras, patrones o instancias. Dependiendo del tipo de aprendizaje los patrones están compuestos por:

- Patrones o variables de entrada y variables de salida deseada; en el caso de que los modelos empleen aprendizaje supervisado, como es el caso del Perceptrón Multicapa y el modelo KNN.
- Únicamente patrones, variables o datos de entrada, para aquellos modelos que desempeñan un aprendizaje no supervisado. El método PCA pese a no realizar un

aprendizaje, se puede enmarcar dentro de este tipo, ya que reduce la dimensión de los datos de entrada sin tener en cuenta las salidas deseadas de los mismos.

Los patrones tanto de entrada como de salida si los hubiera, están compuestos por un conjunto de valores que toman el nombre de atributos. Los modelos que se van a desarrollar en el presente trabajo, sólo trabajan con atributos numéricos, que son aquellos que adquieren valores reales o enteros.

La calidad de los resultados que obtengan los modelos puede depender en gran medida de la calidad de los datos proporcionados. Por lo tanto, después de recopilar los datos, es necesario realizar un análisis de la información que se le proporciona como entrada a los modelos, así como la información de salida. Por ello, es necesario transformar los datos de los que se dispone, para que los modelos puedan trabajar con esos valores. Esta transformación se divide en dos fases, no son obligatorias, pero si recomendables:

- **Normalización de los datos:** para que los datos se encuentren en un intervalo de valores correcto, ya que los valores deben de ser adecuados para evitar problemas en el aprendizaje; por ejemplo, evita la saturación de las neuronas en el caso del Perceptrón Multicapa.
- **Aleatorización de los datos:** para evitar sesgos en el aprendizaje.

Una vez recopilados y transformados los datos de los que se dispone, estos deben ser divididos en dos subconjuntos; uno para la obtención y construcción de los modelos, denominado conjunto de entrenamiento; y otro para medir la calidad del modelo, denominados datos o conjunto de test. Es imprescindible no utilizar los mismos datos para la fase de entrenamiento y test, ya que se debe medir la capacidad de responder correctamente ante situaciones (patrones) diferentes a las que ha empleado para entrenar, pero representadas en el conjunto de entrenamiento.

2.6. Análisis de los resultados: Coeficiente de determinación

Al tratarse de problemas de regresión, el comportamiento de los modelos ha decidido analizarse mediante el coeficiente de determinación, también conocido como R^2 o determinación múltiple (en la regresión lineal múltiple).

El R^2 es una medida estadística de la bondad del ajuste o fiabilidad del modelo estimado a los datos. Indica cuál es la proporción de la variación total en la variable dependiente (salida deseada), que es explicada por el modelo de regresión estimado; por lo tanto, mide la capacidad explicativa del modelo estimado. El R^2 determina la calidad del modelo para replicar los resultados. Por lo general, mientras mayor sea el R^2, mejor será el ajuste del modelo a sus datos. También se puede entenderse como el porcentaje de varianza explicada por la recta de regresión y su valor siempre estará entre 0 y 1, pero existen casos dentro de la definición computacional del mismo, donde este resultado puede tomar valores negativos, en el caso de que los resultados obtenidos por el modelo sean extremadamente malos.

Es una medida de la proximidad o del ajuste de la recta de regresión a la nube de puntos. También se le denomina bondad del ajuste. Para calcular el coeficiente de determinación se aplica la siguiente ecuación:

$$R^2 = 1 - \frac{Error\ cuadrático}{varianza}$$

Ecuación 14: Coeficiente de determinación

El error cuadrático producido por todos los patrones de la red, se calcula con el error producido por cada patrón n ($e(n)$).

$$Error\ cuadrático = \sum_{n=1}^{N} e(n)$$

Ecuación 16: Error cuadrático total

$$e(n) = \sum_{i=1}^{n} (s_i(n) - y_i(n))^2$$

Ecuación 15: Error cuadrático por cada patrón

donde N es el número de patrones o muestras, $e(n)$ es el error cometido por el modelo para el patrón n. Siendo $Y(n) = (y_i(n), \dots, y_n(n))$ el vector que contiene las salidas producidas por el modelo y $S(n) = (s_i(n), \dots, s_n(n))$ el vector de salidas deseadas.

La varianza se calcula mediante la siguiente ecuación:

$$varianza = \sum_{i=1}^{N} (s_i(n) - \overline{S(n)})^2$$

Ecuación 17: Varianza

siendo $\overline{S(n)}$ la media de las salidas deseadas del conjunto de datos de entrenamiento.

(Vila , Sedano, López , & Juan)
(García Centeno, s.f.)

2.7. Herramientas empleadas

2.7.1. MATLAB

Para la realización de los experimentos de este trabajo se ha empleado la herramienta Matlab. La primera versión de Matlab data de los años 70, y fue diseñada como herramienta de apoyo para los cursos de teoría de matrices, álgebra lineal y análisis numérico; el nombre Matlab es un acrónimo: "MATrix LABoratory". Hoy en día, Matlab es un programa muy potente, con un entorno agradable y con una gran versatilidad; que incluye herramientas de cálculo científico y técnico y de visualización gráfica, así como un lenguaje de programación de alto nivel. Se utiliza para aprendizaje automático, procesamiento de señales, procesamiento de imágenes, visión artificial, comunicaciones, finanzas computacionales, diseño de control, robótica y muchos otros campos. Todo esto hace que sea un programa utilizado por un gran número de científicos e ingenieros alrededor de todo el mundo.

Matlab proporciona una gran cantidad de "toolbox" o cajas de herramientas, que proporcionan un grupo de instrucciones especializadas con cierta finalidad en común. Entre ellas cuenta con una específica para las redes de neuronas, que hace que Matlab sea la herramienta perfecta para realizar este estudio.

El paquete que contiene las instrucciones especializadas en redes de neuronas se llama "Neural Nework Toolbox"; nos ofrece una implementación genérica de redes neuronales, así como de redes concretas (Perceptrón, som, etc…). Matlab utiliza una estructura única que nos da acceso a todas las propiedades de la red, independientemente del tipo que esta sea, de manera que utilizando esta propiedad se puede modificar las entradas, capas, conexiones, etc. De este modo una vez configurada la red según las necesidades, se podrá manipular fácilmente con las funciones disponibles (simulación, entrenamiento, inicialización, etc).

2.7.2. Microsoft Excel

Excel es un programa distribuido por Microsoft, conocido como una de las herramientas de software más utilizadas y útiles para realizar cálculos. Permite realizar cálculos sobre un conjunto de datos grande, por lo que ha sido utilizado para el tratamiento de los datos en este trabajo.

Más concretamente ha sido utilizado para normalizar, aleatorizar y separar el conjunto de datos; ya que los métodos empleados para el trabajo no pueden trabajar directamente con información real. Se han normalizado los datos para poder trabajar en un intervalo cerrado de valores y así poder manejarlos en los modelos desarrollados. También se han aleatorizado para eliminar la presencia de patrones en los conjuntos de datos y así poder obtener mejores resultados. Y por último para realizar una separación en dos subconjuntos, uno de entrenamiento y otro de test.

3. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

En este apartado del trabajo se describe el diseño escogido y la implementación de los modelos empleados para la reducción de la dimensionalidad de los datos en problemas de regresión.

Para realizar un estudio de si es más eficaz reducir la dimensionalidad de los datos mediante RNA frente a PCA, y si es posible reducir la dimensión de los datos en problemas de regresión sin una gran pérdida de información relevante; se han desarrollado varios experimentos, de todos ellos se ha realizado un análisis de los resultados para poder determinar si la reducción de la dimensionalidad es exitosa o no, y qué técnica es más eficaz. Los experimentos que se van a desarrollar y analizar son los siguientes:

- Aplicar Redes de Neuronas Artificiales sobre los datos originales, más concretamente el Perceptrón Multicapa.
- Obtener las activaciones de la capa oculta del Perceptrón ya entrenado, estas activaciones son obtenidas mediante dos fórmulas distintas que se explicarán más adelante, y aplicar a las activaciones el algoritmo KNN.
- Emplear KNN sobre los datos originales.
- Mediante el método de PCA, obtener los componentes principales y realizar el KNN sobre los primeros componentes principales.

Para poder realizar los experimentos, se parte de un conjunto de datos original, que como ya se ha explicado en el apartado *2.5.Procesado del conjunto de datos* deben ser tratados, ya que los datos deben estar normalizados, aleatorizados y separados en dos subconjuntos (uno de entrenamiento y otro de test). Para ello se ha empleado la herramienta de Microsoft Excel especificada en el apartado *2.7.2.Microsoft Excel*.

Se ha decidido desarrollar todas estas implementaciones en el entorno de Matlab, explicado en el apartado *2.7.1.MATLAB*, ya que dispone de un lenguaje de alto nivel y posee un paquete de herramientas destinado a las redes de neuronas. Además de todas las características detalladas anteriormente.

A continuación, se explica de forma más detallada los pasos que se han seguido para el desarrollo de este estudio, así como una descripción de las implementaciones realizadas para llevarlo a cabo.

3.1. Introducción

El propósito general de esta experimentación es ser capaces de reducir la dimensión de los datos en problemas de regresión, y realizar una comparación entre las RNA que emplean un aprendizaje supervisado y el modelo PCA. Para ello se han utilizado tres técnicas diferentes (RNA, KNN y PCA) combinándolas entre ellas, de manera que mediante un análisis de los resultados obtenidos se puede determinar si el estudio ha sido exitoso o no.

Previamente se ha realizado un diseño de los experimentos y posteriormente se han implementado en el entorno de Matlab. El esquema gráfico del sistema a realizar se muestra en la siguiente ilustración, donde se aprecia que se han ejecutado cuatro análisis distintos, de los cuales se ha hecho una observación de los resultados que ha obtenido cada uno de ellos mediante el coeficiente de determinación explicado en el apartado *2.6. Análisis de los resultados: Coeficiente de determinación*.

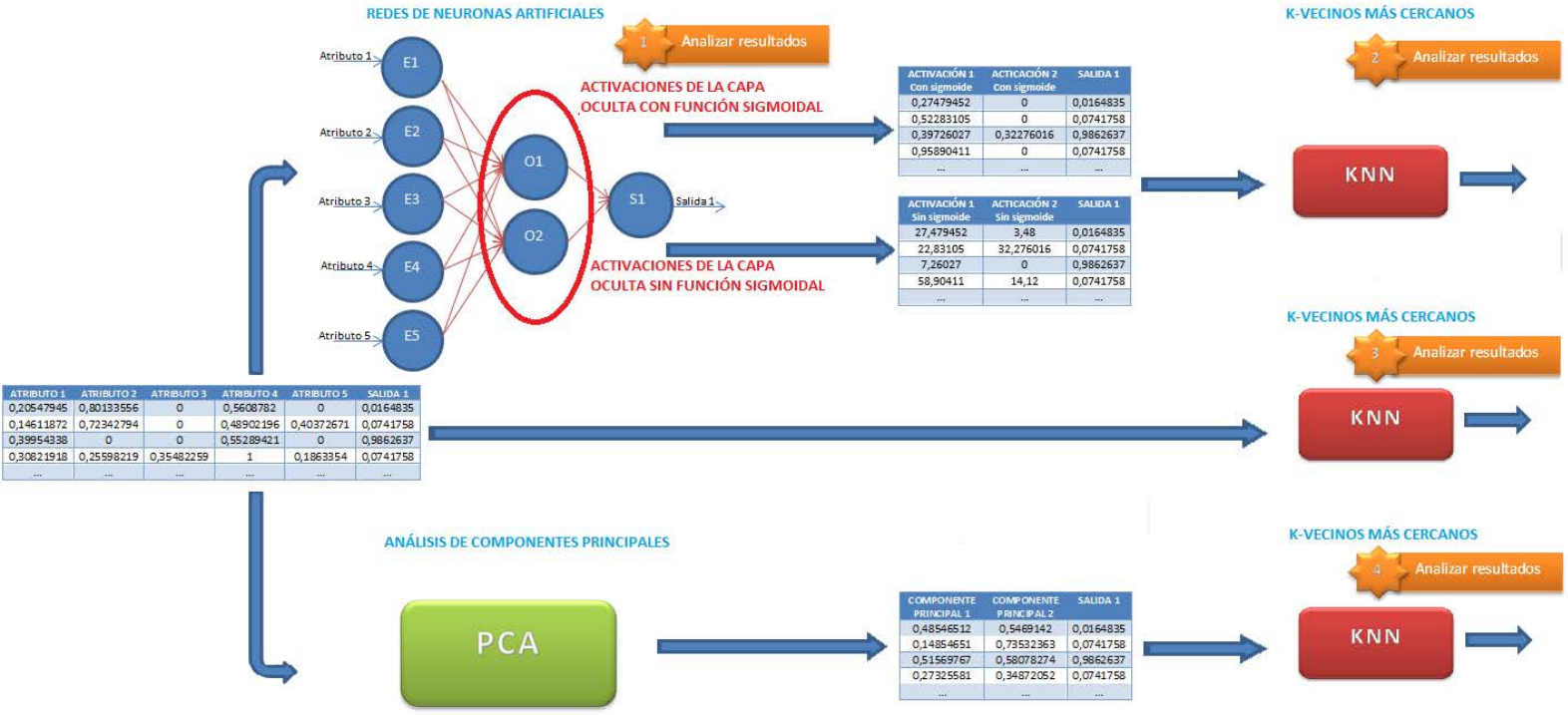


Ilustración 14: Esquema del sistema

Se parte siempre del conjunto de datos originales, los cuales se quiere reducir; los datos de partida, como ya se ha mencionado, están procesados, es decir, se han sido normalizados, aleatorizados y separados en dos subconjuntos. El primer estudio realizado es aplicar al conjunto de datos originales tratados, el Perceptrón Multicapa, analizando el comportamiento de la red y así poder determinar cómo de bien se ajusta a los patrones de entrada. Una vez que la red esta entrenada, se obtienen las activaciones de la capa oculta reduciendo así la dimensión de los datos originales. Estas activaciones se van a obtener aplicando dos fórmulas distintas, por un lado, se calculan las activaciones con la fórmula habitual expuesta en el apartado *2.4.1.2.Propagación de los patrones de entrada*, es decir, calculando la función sigmoidal del sumatorio de las entradas por los pesos más el umbral; y por otro lado, se obtendrán las activaciones sin calcular la función sigmoidal, obteniendo únicamente la entrada neta a las neuronas ocultas, como el sumatorio de las entradas por los pesos más el umbral. Se obtendrán de este modo dos reducciones de los datos diferentes, estos dos conjuntos de activaciones van a ser empleados en el algoritmo KNN, donde las activaciones pasan a ser las entradas del algoritmo y las salidas se corresponden con las mismas salidas con las que se contaba en el conjunto de datos originales. Como puede verse en la imagen de ejemplo, los datos originales parten de un total de cinco atributos de entrada y una salida; las cinco entradas originales son reducidas a dos activaciones o entradas netas, correspondientes a las dos neuronas de la capa oculta y la misma salida original. Una vez formado los nuevos conjuntos de datos reducidos, se aplica el algoritmo KNN y se analiza su comportamiento frente a estos nuevos datos; este proceso se corresponde con un segundo análisis, tanto de los resultados sobre las activaciones calculadas de la forma habitual como de las activaciones sin calcular la función sigmoidal. El conjunto de estos dos análisis, forman el primer experimento correspondiente a analizar RNA+KNN, compuesto por RNA_con_función_sigmoidal+KNN y RNA_sin_función_sigmoidal+KNN.

Si se sigue el guion de la imagen 14, se puede ver que el siguiente experimento consiste en aplicar el conjunto de datos originales tratados sobre el algoritmo KNN; este algoritmo como ya se ha mencionado, se emplea para regresión y no para clasificación como es su comportamiento habitual. Una vez que se obtienen los resultados se puede determinar cómo de correcto es el comportamiento

de este algoritmo para un conjunto de datos dado calculando el coeficiente de determinación de los resultados.

Para concluir, se aplica al conjunto de datos originales tratados el método PCA, para obtener un listado de los componentes principales; es decir, una lista con los componentes ordenados de mayor a menor relevancia. De este listado se escoge un número de componentes principales relevantes que sea inferior al número de atributos de entrada, produciéndose así una reducción en los datos originales. Como se ve en el ejemplo, se parte de los cinco atributos originales y se escogen los dos primeros componentes principales de la lista obtenida por el PCA. Por lo tanto, se obtiene un conjunto reducido de los datos que cuenta con los dos primeros componentes principales del listado y la salida se corresponde con la salida del conjunto de datos original. Sobre este nuevo conjunto se aplica el algoritmo KNN, correspondiente al experimento PCA+KNN, y se analizan los resultados obtenidos.

Para llevar a cabo el estudio, se han manejado e implementado tres funciones distintas de una forma genérica para poder aplicarlas con los distintos datos de partida, ya sean originales o reducidos:

1. Utilización una red mediante el Perceptrón e implementación para obtener las activaciones de la capa oculta.
2. Implementar el algoritmo KNN
3. Utilización del algoritmo PCA y obtención del listado con los componentes principales.

3.2. Utilizar Perceptrón Multicapa e implementar la obtención de las activaciones de la capa oculta

Se puede decir que esta función tiene dos objetivos principales, el primero de ellos es ser capaces de crear una red de neuronas y entrenarla de tal manera que se ajuste adecuadamente al conjunto de datos originales, tanto con el conjunto de entrenamiento como con el de test. Una vez que se ha cumplido el primer objetivo, se obtendrán las activaciones de la capa oculta de la red para cumplir el segundo objetivo; que es ser capaces de reducir las N dimensiones de los datos originales a M dimensiones de las activaciones de las neuronas que forman la capa oculta de la red, siendo M menor que N lógicamente. Estas activaciones se obtienen mediante dos cálculos diferentes, el primer método es mediante la forma tradicional y el otro sin realizar el cálculo de la función sigmoidal, es decir, las activaciones serán el sumatorio de las entradas por los pesos más el umbral que se corresponde con la entrada neta a la neurona oculta.

En primer lugar, una vez que los datos han sido procesado (normalizados, aleatorizados y divididos en dos subconjuntos) se procede a cargar los dos subconjuntos, el conjunto de entrenamiento y el conjunto de test. Cuando los dos bloques de datos están cargados, se dividen en atributos de entrada y atributos de salida, ya que como se ha mencionado anteriormente, el Perceptrón es un algoritmo de aprendizaje supervisado, va modificando el valor de sus parámetros según el error cometido por el mismo.

Una vez que se dispone tanto de los datos de entrada como de salida para los dos conjuntos de entrenamiento y test. Se procede a decidir la topología de la red, es decir, se establece el número de neuronas ocultas que tendrá la red, se crea y se inicializa el valor de sus parámetros.

A continuación, la red debe ser entrenada, se establecen una serie de características para dicho entrenamiento, como el número de ciclos o iteraciones que se desea que realice, el valor de la tasa de aprendizaje, la función de transferencia que empleará la red, entre otras; y se procede a entrenar la red.

Cuando la red está entrenada se obtienen las salidas producidas por la misma y se comparan con las salidas deseadas para obtener el error cometido por la red; y calculando el coeficiente de determinación se puede analizar lo bien que se ajusta la red a los datos de entrenamiento. El siguiente paso es obtener de dos maneras diferentes las activaciones de la capa oculta para los datos de entrenamiento. Para cada patrón de entrada se recorre cada neurona oculta calculando así sus activaciones como se ha visto en el apartado *2.4.1.2.Propagación de los patrones de entrada* y sus activaciones sin realizar el cálculo de la función sigmoidal. De este modo se obtendrán dos nuevos conjuntos de datos de entrenamiento, donde el número de atributos de entrada inicial (N) se ve reducido a M neuronas ocultas.

Una vez que se dispone de la red entrenada, del coeficiente de determinación cometido durante el entrenamiento y de los dos conjuntos de activaciones de las neuronas ocultas para los datos de entrenamiento; se debe probar la red con patrones nuevos, para ello se simula la red entrenada con el conjunto de datos de test. Al igual que con el entrenamiento, se obtiene el coeficiente de determinación cometido por la red con un conjunto de datos con los que no ha entrenado y las activaciones de las neuronas de la capa oculta para los patrones de test calculadas de las dos formas expuestas anteriormente.

Consecuentemente con esta función se obtiene el R^2 cometido por la red tanto en la fase de entrenamiento como en la de test y cuatro conjuntos de datos; dos que contienen las activaciones de las neuronas de la capa oculta para el conjunto de datos de entrenamiento con y sin el cálculo de la función sigmoidal; y otros dos que contienen las activaciones de las neuronas ocultas para los patrones de test con el cálculo y sin el cálculo de la función sigmoidal; que pasarán a ser las entradas en el modelo KNN y donde se han reducido el número de atributos con respecto a los datos originales.

3.3. Implementar el KNN

El objetivo de esta función es crear un modelo KNN para problemas de regresión, capaz de predecir las salidas de los nuevos patrones presentados, cometiendo el error más bajo posible. Para ello la salida de un nuevo patrón será la media de las salidas de los K vecinos más cercanos a él.

Se ha implementado una sola función para el algoritmo KNN de una forma genérica para que esta misma implementación sirva para todos los conjuntos de datos de partida, ya sean los datos originales, las activaciones de la capa oculta o un conjunto de componentes principales. El único cambio que se debe realizar es la carga en la función de los datos correspondientes. Se ha decidido realizar de este modo para no tener que implementar una función por cada experimento que se va a realizar.

En primer lugar, se cargan los datos tratados, el conjunto de entrenamiento y el conjunto de test. Una vez que los datos han sido cargados, se divide cada conjunto en dos bloques, uno con los datos de entrada y otro con los datos de salida. Por lo tanto, se tendrán cuatro conjuntos de datos: entradas de entrenamiento, salidas de entrenamiento, entradas de test y salidas de test.

A continuación, se define el único parámetro que hay que determinar en este algoritmo, que es el valor de K; donde se establece el número de vecinos que se tendrán en cuenta a la hora de aplicar el algoritmo.

En el momento que se tienen todos los datos y parámetros necesarios para llevar a cabo el algoritmo, lo que debe hacerse es calcular por cada patrón de entrada del conjunto de test la distancia con todos los patrones de entrada del conjunto de entrenamiento. Para ello, se recorre por cada patrón de test todos los patrones de entrenamiento y se calcula la distancia euclídea con cada uno de ellos.

Para calcular la distancia euclídea de cada patrón de test, se restan las entradas de todos los patrones de entrenamiento menos las entradas del patrón de test y se elevan al cuadrado. Una vez que se tiene la diferencia del patrón de test con todos los patrones de entrenamiento al cuadrado, se suman sus filas y se realiza la raíz cuadrada para obtener la distancia euclídea del patrón de test con todos los patrones de entrenamiento.

Teniendo la distancia del patrón nuevo de entrada con todos los patrones de entrenamiento se deben ordenar de menor a mayor, para obtener un listado de todos los patrones de entrenamiento ordenados de menor distancia a mayor con respecto al nuevo patrón.

Para concluir la salida que producirá el modelo para el nuevo patrón será la media de las salidas de los K primeros patrones de entrenamiento de la lista anterior, que se corresponden con los K vecinos más próximos a dicho patrón.

Se debe realizar todo el procedimiento anterior para cada patrón de test, y se calcula el coeficiente de determinación, que es el resultado de restar a la unidad, la división del error cuadrático medio entre la varianza de las salidas deseadas de test.

Esta función será aplicada tanto a los datos originales, como a las activaciones de la capa oculta, como a un conjunto de componentes principales. La única diferencia entre unos y otros es el conjunto de datos de entrada, tanto de test como de entrenamiento. Con los resultados obtenidos en los distintos experimentos se podrá comparar si la reducción de la dimensionalidad de los datos es exitosa y qué modelo consigue reducir la dimensión de los datos perdiendo la menor información relevante posible, si el Perceptrón con las activaciones de la capa oculta o el PCA escogiendo los primeros componentes principales.

3.4. Utilizar el PCA y obtener los componentes principales

El objetivo de esta función es obtener un listado con los componentes principales ordenados de mayor a menor relevancia. De esta lista se obtendrán los M mejores componentes, que se corresponden con los M primeros componentes principales de la lista; tratando de reducir así el número de patrones de entrada originales N, en los M mejores componentes principales. Lógicamente M siempre debe ser menor que N para poder logar la reducción de los datos.

En primer lugar y como en todas las implementaciones anteriores, se deben cargar el conjunto de entrenamiento y test; separando los atributos de entrada y de salida de ambos conjuntos.

En el momento que se cuenta con los datos cargados, se debe aplicar el modelo PCA al conjunto de entradas, tanto de entrenamiento como de test; PCA se puede encuadrar dentro de los modelos basados en aprendizaje no supervisado, ya que las salidas de los datos no entran en juego para calcular los componentes principales. En primer lugar, se deben centrar los datos, tanto del conjunto de test como de entrenamiento, esto se realiza restando a cada columna de ambos conjuntos, la media de las columnas del conjunto de entradas (las columnas son los atributos que caracterizan a cada dato); ambos conjuntos de datos deben ser centrados de la misma forma. Mediante la función propia de Matlab (pca), se obtiene la matriz de coeficientes de correlación.

Contando ya con los datos centrados y con los coeficientes de correlación, para obtener los dos listados de componentes principales, de entrenamiento y test; únicamente se debe multiplicar los datos centrados por los coeficientes. De este modo, los componentes principales del conjunto de entradas de entrenamiento se corresponden con los atributos de entrada centrados multiplicado por los coeficientes obtenidos mediante la función pca y los componentes principales de test se calculan multiplicando los coeficientes por las entradas de test centradas.

El siguiente paso es decidir el número de componentes principales que se desean utilizar tanto en el conjunto de entrenamiento como en el de test, este número debe ser el mismo en ambos conjuntos y debe ser menor que el número de atributos de entrada que poseían los datos originales. El número de componentes principales seleccionados, pasarán a ser entradas del modelo KNN y las salidas serán las mismas salidas que en el conjunto de datos originales, habiendo reducido así la dimensión de los datos originales.

3.5. Entorno de desarrollo

Tras haber explicado los modelos que se van a emplear, en la presente sección se van a describir las infraestructuras técnicas empleadas para el desarrollo del presente estudio.

3.5.1. Hardware utilizado

Todo el desarrollo del trabajo se ha realizado en un mismo equipo portátil que presenta las siguientes características:

- Fabricante: msi
- Modelo: GP72 6QF-492XES
- Procesador: Intel® Core™ i7-6700HQ (2.60 GHz)
- Memoria RAM 16GB DDR4 2133MHz (8GB*2)
- Disco duro 1TB (7200 RPM)

3.5.2. Software utilizado

En cuanto al conjunto de programas empleados para llevar a cabo este estudio, se pueden dividir en cuatro secciones dependiendo de la fase del proyecto en la que se trabaje. Por un lado, el software elegido para la implementación del código de los distintos modelos, en lenguaje de alto nivel, ha sido Matlab R2015a junto con el contenedor de herramientas “Neural network toolbox” especializada en redes de nueras artificiales.

Para el tratamiento de los datos, normalización, aleatorización y división en subconjuntos; se ha empleado el software basado en hojas de cálculo, Microsoft Excel 2016 perteneciente al paquete de ofimática de Microsoft Office 2016.

Para la redacción de la documentación del trabajo de fin de grado, se ha utilizado la aplicación de procesamiento de texto Microsoft Word 2016 perteneciente también al paquete Microsoft Office 2016.

Por último, para realizar la presentación del proyecto se ha utilizado Power Point del paquete de Microsoft Office 2016, que es un editor de diagramas por excelencia de Microsoft.

Todas las aplicaciones citadas anteriormente han sido gestionadas por el sistema operativo Windows 10 de 64 bits.

3.6. Alternativa de diseño

Para concluir con el apartado se procede a razonar las decisiones tomadas a la hora de elegir los programas utilizados para llevar a cabo el proyecto de fin de carrera.

En cuanto a la utilización de una aplicación para implementar los modelos en un lenguaje de programación de alto nivel, en un inicio se analizaron todos los posibles programas a escoger; de todos ellos se tuvo dudas si emplear R o Matlab.

Pese a que ambos programas son bastante parejos en un examen rápido, ya que ambos tienen un lenguaje de programación de alto nivel propio, acceso a funciones matemáticas, un enfoque al análisis estadístico y una comunidad de usuarios; Matlab presenta ciertas mejoras y ventajas para el desarrollo del presente estudio, que hacen que nos decantemos a su favor.

A diferencia de R, Matlab proporciona un entorno de desarrollo de alta productividad, los algoritmos de Matlab presentan una gran calidad y han sido comprobados de manera práctica debido a su uso por millones de ingenieros y científicos. Se han realizado pruebas a tiempo completo que validan la calidad y precisión de los productos, además de asegurar que el software pasa un conjunto de pruebas exhaustivas antes de su publicación.

Matlab también proporciona una gran cantidad de toolboxes y apps adaptados a tareas científicas y de ingeniería. A diferencia de R, Matlab proporciona una caja de herramientas específica para redes de neuronas y para aprendizaje automático; que han hecho que sea uno de los puntos más importantes por lo que se ha decidido utilizarla.

Otro punto a favor es que posee un lenguaje fácil de aprender en comparación con R, que fue desarrollado para estadísticos, el lenguaje de Matlab es fácil de aprender y recordar ya que su sintaxis es simple y coherente, además que se ha utilizado en alguna ocasión durante la carrera.

Matlab adquiere un rendimiento más rápido que ha sido probado, es entre tres y ciento veinte veces más rápido que R cuando se ejecuta en una prueba de ejemplos estadísticos; por lo que está comprobado que su ejecución es más rápida. En caso de tareas con alta carga computacional es posible ejecutar varios motores de Matlab en paralelo gracias a “Parallel Computing Toolbox”.

Para concluir Matlab ofrece respuestas más rápidas gracias a la documentación exhaustiva, el servicio de soporte y la comunidad de usuarios. En cuanto a la documentación se pueden realizar búsquedas globales online y dentro del propio escritorio de Matlab. MathWorks cuenta con más de 200 expertos de soporte técnico con dedicación exclusiva para responder preguntas y solucionar problemas. La gran documentación y soporte que ofrece es una de las razones definitivas por las que se ha decidido recurrir a Matlab.

Por todos estos motivos se ha decidido optar por Matlab en lugar de R a la hora de desarrollar el código de los modelos y realizar los experimentos.

En cuanto a las aplicaciones de ofimática se ha decidido tratar los datos mediante Microsoft Excel, ya que es una aplicación muy potente basada en hojas de cálculo. Puede realizar operaciones de manera sencilla y rápida sobre cientos de miles de datos numéricos.

Además de los motivos expresados, la Universidad posee licencia para dichos programas, por lo que no supone ningún gasto adicional y se han elegido frente a programas de licencia gratuita.

4. ESTUDIO REALIZADO

En el siguiente apartado se expone el estudio que se ha elaborado y los resultados obtenidos por los distintos experimentos realizados; analizando dichos resultados y obteniendo las conclusiones que se pueden observar tras aplicar las técnicas para reducir la dimensionalidad de los datos en problemas de regresión. Se explicará también la metodología y topología escogidas para efectuar los experimentos, así como los pasos desempeñados para realizarlos.

4.1. Metodología y topología empleada

Al tratarse de un proyecto de investigación es importante especificar la metodología que se va a llevar a cabo a la hora de realizar los experimentos, para que todas las pruebas realizadas puedan ser comparadas entre sí y poder analizar el comportamiento general del sistema elaborado. En este caso, se deben establecer las metodologías que se van a realizar a la hora de obtener los conjuntos de datos reducidos y a la hora de aplicar el KNN para comprobar el comportamiento del sistema.

Como se ha explicado anteriormente, se van a comparar dos modelos diferentes para llevar a cabo la reducción de los datos originales; cada uno de estos modelos presenta una serie de parámetros necesarios y de los que depende el comportamiento de los sistemas, es decir, dependiendo de los parámetros que se especifiquen, los modelos se ajustarán de mejor o peor manera a los datos presentados y por lo tanto obtendrán mejores o peores resultados.

Como la finalidad del proyecto es determinar si es posible reducir la dimensionalidad de los atributos de entrada en problemas de regresión desempeñando diferentes modelos, los parámetros escogidos para cada modelo no son relevantes para llevar a cabo el estudio; por lo que han sido ajustados, pero no se verán reflejados en este trabajo, ya que la finalidad es analizar el coeficiente de determinación logrado por los experimentos.

En este punto se puede dividir la metodología seleccionada en dos grupos, por un lado se debe especificar la metodología para decidir cómo se han reducido los datos presentados para aquellos modelos encargados de reducir la dimensionalidad, como son el caso del Perceptrón y el PCA; y por otro lado se debe establecer la metodología utilizada por el modelo KNN para determinar el comportamiento del sistema, es decir, se debe detallar el valor del parámetro K que representa los vecinos más cercanos a tener en cuenta.

Por lo tanto, en primer lugar, se especificará la técnica escogida para llevar a cabo la reducción de los atributos de entrada. Se ha decidido que los modelos encargados de reducir la dimensionalidad de los datos, lo hagan a partir del 50% en adelante; es decir, los atributos de entrada se verán reducidos como poco un 50% sobre los atributos de entrada originales. Estos experimentos son considerados agresivos, en el caso de que el modelo se ajuste correctamente y obtenga buenos resultados con los datos reducidos, estaría totalmente justificado realizar la reducción de los datos originales; ya que se quitaría mucho peso computacional, que implica una reducción considerable en el tiempo de ejecución; y además, ambas técnicas de reducción realizan una transformación de los atributos originales, por lo que si los resultados son favorables, se puede concluir que los nuevos atributos concentran en un menor número de ellos, la información relevante contenida en los atributos originales.

Se ha dictaminado que como mínimo los atributos de entrada originales se van a ver reducidos un 50%, pero se van a realizar siete reducciones sobre los datos originales, en las cuales, se va decrementando el porcentaje de reducción un 7,14% (50/7). Por lo tanto, se comienza reduciendo al 50% los datos originales, posteriormente al 42,86% (50%-7,14); y así sucesivamente.

Para una mejor comprensión del modo de reducir la dimensión de los datos se va a realizar un ejemplo, si se parte de un dominio que cuenta con setenta atributos de entrada originalmente, se van a realizar siete reducciones que van desde el 50% y decrementando de manera progresiva; es decir, el número de atributos originales se verá reducido a: 35, 30, 25, 20, 15, 10 y 5 atributos.

En el caso del Perceptrón Multicapa estos valores representan el número de neuronas ocultas en la topología de la red, por lo que se realizarán siete arquitecturas diferentes para un mismo conjunto de datos de partida, en todas las arquitecturas se contará únicamente con una capa oculta; y se obtendrán catorce conjuntos de datos reducidos, ya que los datos reducidos lo forman las activaciones de la capa oculta que se han calculado de dos maneras diferentes (calculando la función sigmoidal y sin el cálculo de la función sigmoidal, entradas netas a las neuronas ocultas). En el caso de que el modelo sea PCA, representan el número de los primeros componentes principales que deben seleccionarse de la lista de componentes principales ordenados de mayor a menor relevancia obtenidos por este modelo, obteniéndose siete conjuntos de datos reducidos.

Una vez establecida la metodología para reducir la dimensión de los atributos de entrada, se debe determinar la metodología que se llevará a cabo con el modelo KNN, que determinará el comportamiento general del sistema, comprobando si la reducción de la dimensionalidad es exitosa o no y la eficacia de cada modelo a la hora de realizar dicha reducción.

El parámetro que se debe fijar para realizar el modelo KNN, es el número de vecinos que entran en juego a la hora de estipular la salida de un nuevo patrón de entrada. Por lo tanto, para cada conjunto de datos que contiene los atributos de entrada reducidos se efectuarán cuatro análisis mediante el modelo KNN, teniendo en cuenta sus 1, 3, 5 y 7 vecinos más cercanos; en este caso se han escogido números impares, aunque no es relevante para nuestro estudio, ya que al tratarse de problemas de regresión (salidas continuas) no se producen empates como en el caso de clasificación.

En conclusión, para un conjunto de datos original dado y siguiendo el esquema *Ilustración 14: Esquema del sistema* se realizarán un total de ochenta y ocho experimentos. En primer lugar, con el Perceptrón Multicapa se deben realizar siete arquitecturas distintas obteniendo de dos formas diferentes las activaciones de las neuronas de la capa oculta, obteniendo un total de catorce conjuntos de datos reducidos; para cada uno de estos conjuntos de datos reducidos debe aplicarse el KNN teniendo en cuenta 1, 3, 5 y 7 vecinos; que hacen un total de cincuenta y seis experimentos. En el caso de aplicar KNN sobre los datos originales se deben realizar cuatro modelos distintos teniendo en cuanta 1, 3, 5 y 7 vecinos más cercanos. Y para concluir, mediante el PCA se obtienen siete conjuntos de datos reducidos a los que deben aplicarse los cuatro modelos de KNN, haciendo un total de veintiocho experimentos.

Una vez determinada las diferentes pruebas a realizar en cada uno de los procesos, se expone la metodología general del sistema; que consta de los siguientes pasos:

1. Procesado del conjunto de datos: tratamiento de los datos y separación en dos subconjuntos (entrenamiento y test).
2. Experimento 1: ejecución del Perceptrón Multicapa con el conjunto de datos de entrenamiento y test, extracción de las activaciones de la capa oculta para ambos conjuntos y de las dos formas especificadas; y aplicación del modelo KNN sobre el conjunto de datos formado por las activaciones de la capa oculta.
3. Experimento 2: aplicación del modelo KNN sobre los datos originales.
4. Experimento 3: aplicación del modelo PCA para la obtención de los primeros componentes principales de la lista obtenida por el modelo, tanto del conjunto de entrenamiento como del conjunto de test; y aplicación del modelo KNN sobre los conjuntos formados por los primeros componentes principales.

5. Comparación de RNA+KNN (tanto RNA_con_función_sigmoidal+KNN como RNA_sin_función_sigmoidal+KNN), KNN y PCA+KNN.

1. Procesado del conjunto de datos

Cada conjunto de datos utilizados en este estudio ha sido procesado para un mejor comportamiento de los modelos frente a dichos datos, como se ha explicado en el apartado *2.5.Procesado del conjunto de datos*.

El procesado de los datos consta de dos partes: en primer lugar, el tratamiento de los datos y posteriormente la separación de los mismos en dos subconjuntos (entrenamiento y test).

Debido a que gran parte de los conjuntos de datos empleados en este trabajo pertenecen a dominios del mundo real, todos excepto dos, sus atributos presentan características muy dispares unos de otros. Por este motivo todos los conjuntos de datos han sido tratados para ser unificados y poder así trabajar con ellos.

El primer paso para unificar los datos es realizar una normalización de los mismos, con ello se consigue que la base de datos con la que se va a trabajar esté comprendida en un rango concreto [0,1]. Como ya se ha mencionado, esta normalización de los datos ha sido realizada con el programa Excel mediante la siguiente fórmula:

$$Norma. = \frac{dato - MIN(TotalDatos)}{MAX(TotalDatos) - MIN(TotalDatos)}$$

Ecuación 18: Normalización de los datos

Una vez que los datos han sido normalizados y por lo tanto comprendidos en el rango de valores [0,1]; estos deben ser aleatorizados para evitar que se produzcan sesgos en el aprendizaje, ya que la base de datos puede estar almacenada de tal manera que sus datos estén creando estándares inútiles y que afectan al comportamiento de los modelos desarrollados. Para llevar a cabo esta tarea se ha empleado la herramienta Excel, que mediante la función ALEATORIO () se ha añadido una nueva columna con valores aleatorios que posteriormente han sido utilizados como referencia para reordenar la base de datos ya normalizada.

Una vez concluida la fase de tratamiento de los datos, estos deben ser separados en dos subconjuntos; un conjunto de entrenamiento y un conjunto de test. Se ha decidido que el subconjunto de datos de entrenamiento este formado por el 70% del conjunto total, por lo que 30% restante de los datos pasan a formar el subconjunto de test.

2. Experimento 1

El experimento 1 está compuesto por la ejecución del Perceptrón Multicapa con el conjunto de datos de entrenamiento y test, extracción de las activaciones de la capa oculta para ambos subconjuntos, mediante los dos métodos distintos especificados (con y sin función sigmoidal); y aplicación del modelo KNN sobre el conjunto de datos formado por las activaciones de la capa oculta.

Una vez que los datos han sido procesados se ejecuta el Perceptrón para el conjunto de datos de entrenamiento, obteniendo de dos formas las activaciones de las neuronas de la capa oculta una vez que la red ha sido entrenada, habiendo reducido de este modo la dimensión de los datos de entrada originales. Posteriormente se simula la red con los patrones de test, se analiza el comportamiento de la misma mediante el coeficiente de determinación y se obtienen las activaciones de las neuronas de la capa oculta para los datos de test con y sin calcular la función sigmoidal. Seguidamente se procede a crear un modelo KNN que tome como partida el conjunto de datos formados por las activaciones de la capa oculta con los datos de entrenamiento habiendo calculado la función sigmoidal y cuyas salidas se corresponden con las salidas de los datos originales; dicho modelo, debe predecir las salidas del conjunto de datos formado por las activaciones con cálculo de función

sigmoidal de las neuronas de la capa oculta, para el conjunto de datos de test. Para determinar el comportamiento del modelo, se analiza mediante el coeficiente de determinación. A continuación, se procede a realizar el mismo proceso, pero tomando los datos obtenidos por las activaciones sin el cálculo de la función sigmoidal, es decir, tomando las entradas netas (sumatorio de los pesos por las entradas más el umbral) que reciben las neuronas ocultas de cada patrón.

3. Experimento 2

El segundo experimento se basa en aplicar el modelo KNN tomando como base el conjunto de datos de entrenamiento original procesado y analizando; y analizando el comportamiento del modelo cuando se presenta el conjunto de datos de test procesado.

4. Experimento 3

El tercer y último experimento consta de aplicar a los datos originales procesados el modelo PCA, obtener un número inferior de entradas mediante los componentes principales y aplicar el algoritmo KNN.

Para ello se aplica el PCA sobre las entradas de los datos originales del conjunto de entrenamiento y test, obteniéndose de este modo dos listados que contienen los componentes principales de entrenamiento y test, ordenados de mayor a menor relevancia. De dicho listado de componentes principales de entrenamiento, se seleccionan los M primeros componentes con los que se formará el modelo KNN, siendo las salidas las mismas del conjunto de datos original. Una vez formado el modelo KNN, se introducen los M primeros componentes principales de test para que el modelo prediga sus salidas. Mediante el cálculo del coeficiente de determinación producido por el modelo KNN, se podrá determinar si la reducción de los atributos de entrada originales a M atributos reducidos, es exitosa o no.

5. Comparación de RNA+KNN, KNN y PCA+KNN

Para concluir, se debe realizar una comparación general de todos los experimentos anteriores. Para ello, en primer lugar, se compara el coeficiente de determinación obtenido por el modelo KNN aplicado a los dos conjuntos de activaciones obtenidos por el Perceptrón Multicapa, es decir, se compara RNA_con_función_sigmoidal+KNN y RNA_sin_función_sigmoidal+KNN. Posteriormente se realiza una comparación de los resultados de estos dos experimentos con los resultados obtenidos por el modelo KNN sobre los componentes principales, es decir, RNA_con_función_sigmoidal+KNN y PCA+KNN; y RNA_sin_función_sigmoidal+KNN y PCA+KNN. Mediante el coeficiente de determinación se comprueba cómo de bien se ajustan los modelos a los datos y mediante una diferencia de estos coeficientes obtenidos entre modelos, se determina qué técnica es más eficaz a la hora de reducir la dimensionalidad de los datos (RNA con función sigmoidal, RNA sin función sigmoidal o PCA). El coeficiente de determinación obtenido por el modelo KNN puede adquirir valores negativos, en caso de que el modelo se no se ajuste de forma adecuada a los datos; para realizar la comparación de resultados, se ha asumido como nulo (R2 = 0) todos aquellos coeficientes de determinación que alcancen valores negativos.

4.2. Conjuntos de datos

En este apartado se exponen los conjuntos de datos seleccionados para realizar el presente estudio; la gran mayoría de estos datos han sido seleccionados del siguiente repositorio: http://funapp.cs.bilkent.edu.tr/DataSets/. Este es un repositorio de conjuntos de datos recogidos principalmente por la búsqueda de recursos en la web. La mayoría de los conjuntos de datos de este repositorio se utilizan para el análisis experimental de técnicas de aproximación de funciones, para la ejecución de máquinas de aprendizaje y análisis estadísticos.

Los dominios seleccionados para la presente investigación son:

- Housing
- House_16H
- Elevators
- Elevators con datos girados
- Dominio sintético paraboloide con los datos girados

4.2.1. Housing

Housing es una base de datos que contiene información real referente a las viviendas de los suburbios de Boston. Este conjunto de datos fue tomado de la biblioteca StatLib perteneciente a la universidad de Carnegie Mellon y su objetivo es determinar el valor medio de las casas habitadas por sus propietarios en dólares.

Contiene un total de 506 instancias, es decir, que posee información de 506 viviendas; cada una de estas instancias consta de 14 atributos (13 de ellos con valor continuo y 1 con valor binario) que representan la información de las viviendas. La información que representa cada atributo es:

1. Tasa de criminalidad de la ciudad por cápita.
2. Porción de suelo residencial dividido en zonas.
3. Proporción de acres de negocios no minoristas presentes en la ciudad.
4. Variable ficticia con valor binario, que es 1 si la vivienda está en los límites del rio y 0 en caso contrario.
5. Concentración de óxido de nitrógeno.
6. Número promedio de habitantes por vivienda.
7. Proporción de viviendas construidas antes de 1940 que están habitadas.
8. Distancias ponderadas a los cinco centros de empleo de Boston.
9. Índice de accesibilidad a las carreteras radiales.
10. Tasa de impuestos de la propiedad.
11. Número de alumnos por profesor en la ciudad.
12. El valor de 1000*(Bk – 0,63) ^2, donde Bk es la proporción de gente de raza negra por municipio.
13. Porcentaje de estatus más bajo de la población.
14. Valor medio de las casas habitadas en valores de 1000$.

4.2.2. House_16H

Esta base de datos ha sido diseñada sobre la base de datos proporcionados por el censo de los EEUU. Los datos fueron recogidos como parte del censo de Estados Unidos en 1990. El objetivo es predecir la mediana del precio de la vivienda en la región, en base a la composición demográfica y el estado del mercado de la vivienda en dicha región.

Contiene un total de 22.784 instancias, cada una de estas instancias consta de 16 atributos continuos.

4.2.3. Elevators

Este conjunto de datos se obtiene de la tarea de controlar un avión F16, la variable objetivo está relacionada con la acción tomada por los elevadores de la aeronave que toma un valor continuo. Consta de 16.599 instancias, cada una de las cuales cuenta con 18 atributos continuos.

4.2.4. Elevators girado

Elevators girado es un dominio creado de manera artificial, partiendo del dominio Elevators al que se le han añadido 8 atributos de manera aleatoria con valores comprendidos en el rango [0,1] y posteriormente los datos han sido rotados también de forma aleatoria. Como resultado se obtiene un conjunto de datos que consta de 16.599 instancias, pero en este caso, a diferencia del dominio anterior, cada instancia está representada por 26 atributos; el atributo objetivo es el mismo que en el dominio anterior y está relacionada con la acción tomada por los elevadores de la aeronave.

Este domino es interesante utilizarlo en el proyecto, ya que se podrá determinar si los modelos son capaces de eliminar aquellos atributos que han sido introducidos de forma aleatoria y que no proporcionan información relevante para alcanzar el objetivo; es decir, se podrá comprobar si los modelos empleados para reducir la dimensionalidad de los atributos (Perceptrón Multicapa y PCA) son capaces de pasarle al algoritmo KNN la información realmente relevante.

4.2.5. Paraboloide con datos girados

Este es un domino sintético de regresión generado a partir de un paraboloide que consta de 2 entradas y 1 salida; se le han añadido 8 dimensiones aleatorias en un rango entre [0,1] y se han multiplicado las entradas por una matriz de giro aleatoria para mezclar los atributos de entrada y hacer el problema más complicado. Por lo tanto, se cuenta con un dominio con 10 atributos de entrada y 1 atributo de salida que consta de 10.000 instancias.

Al igual que en domino anterior, resulta interesante realizar el estudio con este conjunto de datos para determinar si los modelos son capaces de desechar los atributos innecesarios que no proporcionan información relevante.

A continuación, se presenta la imagen del domino paraboloide original, es decir, sin añadirle los atributos aleatorios girados y, por lo tanto, con los dos atributos de entrada de partida.

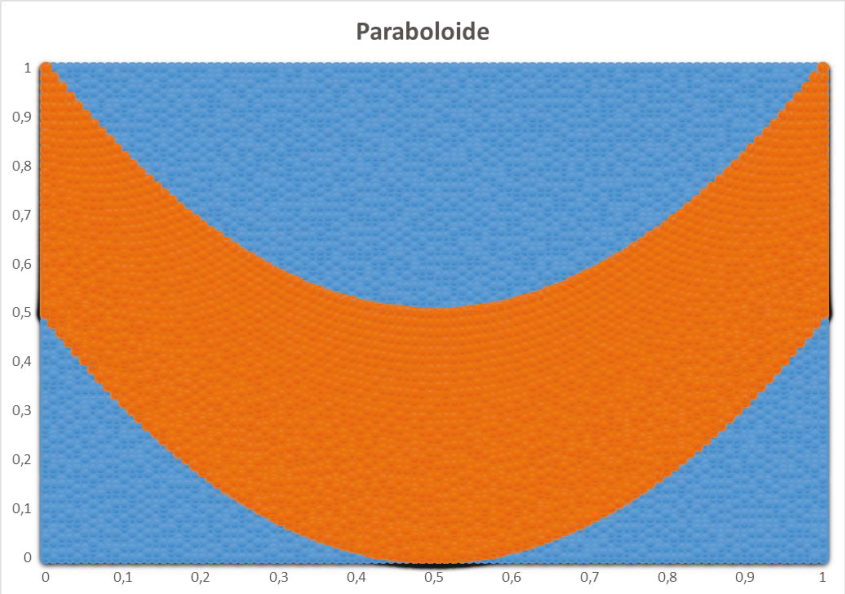


Ilustración 15: Dominio Paraboloide

4.3. Pruebas realizadas

En este apartado se exponen los resultados obtenidos en los estudios realizados para la reducción de la dimensionalidad de los datos en problemas de regresión, sobre cada conjunto de datos descrito anteriormente.

4.3.1. Resultados obtenidos por el conjunto de datos Housing

En esta sección se encuentran los resultados obtenidos tras realizar los estudios de reducción de dimensionalidad con el conjunto de datos "Housing".

4.3.1.1. R2 Perceptrón Multicapa

Tras aplicar el Perceptrón Multicapa al conjunto de datos de entrenamiento y test del dominio "Housing", los resultados obtenidos durante la fase de test, medidos mediante el coeficiente de determinación son los que se muestran en el siguiente gráfico, donde los datos se corresponden con *Tabla 1: Coeficiente de determinación redes de neuronas (Housing)* mostrada en el apéndice. Cada barra del diagrama representa el coeficiente de determinación alcanzado en la fase de test por el Perceptrón Multicapa, según el número de neuronas ocultas en su capa oculta, que se corresponde con la reducción que se va a realizar.

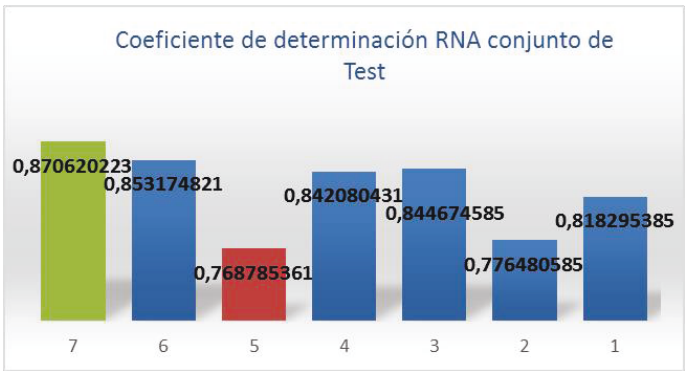


Gráfico 1: Diagrama de barras con R2 de Test RNA (Housing)

Como puede observarse el mejor resultado obtenido para aquellos patrones con los que no se ha entrenado, es decir, con el conjunto de test, lo ha alcanzado la red de neuronas que cuenta con una arquitectura con 7 neuronas ocultas cuyo coeficiente de determinación es de 0,8762 y el peor de los resultados es aquella red que cuenta con 5 neuronas ocultas y obtiene un coeficiente de determinación de 0,768785. Se puede determinar que los resultados obtenidos por la red en el conjunto de test son bastante buenos, por lo tanto, se puede decir que la red se ajusta bastante bien a patrones con los que no ha entrenado, es decir, ha adquirido una buena capacidad de generalización.

4.3.1.2. R2 KNN

Seguidamente se muestran los resultados obtenidos tras aplicar el modelo KNN sobre el conjunto de datos originales (tratado y procesado), es decir, sobre los datos de partida sin que hayan sido transformados y reducidos por ningún modelo. Se muestran mediante el siguiente gráfico que contiene los resultados que se muestran en la *Tabla 2: Coeficiente de determinación KNN (Housing)* presente en el apéndice. Cada barra representa el coeficiente de determinación alcanzado por el modelo KNN sobre los datos originales dependiendo del parámetro k.

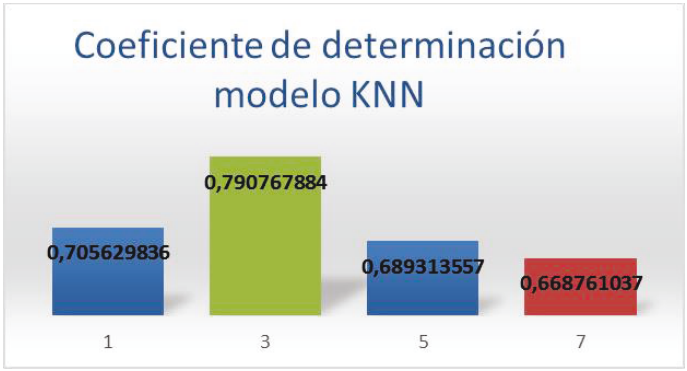


Gráfico 2: Diagrama de barras con R2 del modelo KNN (Housing)

Puede verse que el mejor resultado es aquel que tiene en cuenta 3 vecinos más cercanos y el peor de los resultados lo obtiene el modelo que obtiene las salidas contando con 7 vecinos más próximos.

4.3.1.3. R2 Perceptrón Multicapa + KNN

En esta sección se muestran los resultados obtenidos tras reducir la dimensión del conjunto de datos originales con el Perceptrón Multicapa mediante las activaciones de las neuronas de la capa oculta y aplicar a dichas activaciones el modelo KNN. Para abarcar mejor el estudio de la reducción de la dimensionalidad, por cada red de neuronas se han obtenido dos subconjuntos de datos reducidos: por un lado se obtienen las activaciones de las neuronas de la capa oculta calculadas de forma habitual, como el sumatorio de las entradas por los pesos más el umbral y calculando la función sigmoidal del valor resultante; y por otro lado calculando las activaciones únicamente como el sumatorio de las entradas por los pesos más el umbral, es decir, sin realizar la función sigmoidal obteniendo únicamente las entradas netas de las neuronas ocultas.

4.3.1.3.1. R2 Perceptrón Multicapa aplicando función sigmoidal + KNN

El primer lugar se muestran los resultados obtenidos (coeficiente de determinación) tras aplicar el modelo KNN sobre las activaciones de la capa oculta aplicando la función sigmoidal, la *Tabla 3: Coeficiente de determinación de aplicar KNN sobre las activaciones con función sigmoidal (Housing*) del apéndice, contiene los resultados representados en el siguiente diagrama. Cada color representa el número de atributos de entrada y las diferentes columnas están agrupadas según el número de vecinos a tener en cuenta por el modelo KNN.

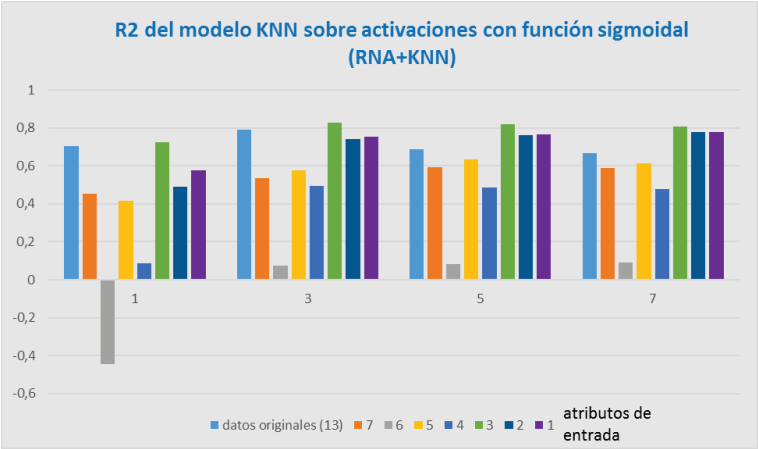


Gráfico 3: Diagrama de barras del modelo KNN sobre las activaciones con función sigmoidal (Housing)

Se observa que en muchos casos los resultados obtenidos por el modelo KNN sobre datos reducidos mediante RNA, superan a los resultados obtenidos sobre los datos originales. El mejor resultado se obtiene al reducir la dimensión de los datos a 3 atributos de entrada y aplicar el modelo KNN teniendo en cuenta las salidas de los 3 vecinos más cercanos; este valor supera incluso al mejor de los resultados obtenidos por KNN sobre los datos originales. También es destacable que al contar

únicamente con el vecino más cercano en el modelo KNN sobre los datos reducidos a 6 atributos de entrada, el resultado es pésimo y se encuentra muy por debajo del resto de resultados obtenidos.

4.3.1.3.2. R2 Perceptrón Multicapa sin aplicar función sigmoidal + KNN

El siguiente gráfico muestra los resultados alcanzados tras aplicar el modelo KNN sobre las entradas netas de las neuronas ocultas, los valores de estos resultados se encuentran en la *Tabla 4: Coeficiente de determinación de aplicar KNN sobre las activaciones sin función sigmoidal (Housing)* del apéndice.

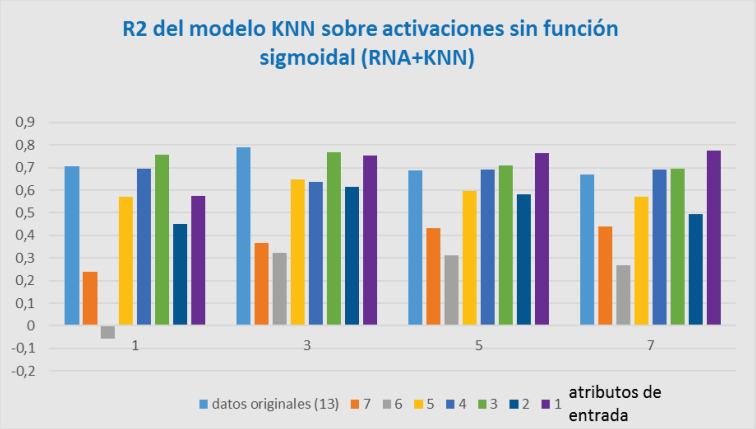


Gráfico 4: Diagrama de barras del modelo KNN sobre las activaciones sin función sigmoidal (Housing)

Puede observarse que, mediante este método de obtener las activaciones, el mejor resultado lo alcanza el modelo KNN aplicado sobre los datos originales con K igual a 3. En cuanto al mejor de los resultados obtenidos tras reducir la dimensión de los datos, lo alcanza el coeficiente de determinación obtenido por el modelo KNN con K igual a 7 y donde los datos han sido reducidos a 1 atributo de entrada. El peor de los resultados, lo obtiene el modelo que tiene en cuenta el vecino más próximo sobre los datos que han sido reducidos a 6 atributos de entrada, ya que toma un valor negativo.

4.3.1.3.3. Comparación entre RNA con función sigmoidal + KNN y RNA sin función sigmoidal + KNN

A continuación, se realiza una comparación entre reducir la dimensionalidad con las activaciones de la capa oculta aplicando la función sigmoidal y sin aplicar la función sigmoidal. Para llevar a cabo dicha comparación, se realiza la diferencia de los resultados obtenidos tras aplicar el modelo KNN sobre los datos reducidos mediante estas dos técnicas. Se ha tomado la decisión de únicamente tener en cuenta aquellos coeficientes de determinación que adquieran valores positivos; por lo tanto, todos aquellos resultados que sean negativos se han considerado como valor nulo para realizar la comparación.

En el siguiente gráfico se muestra la diferencia entre reducir la dimensionalidad con las activaciones de la capa oculta aplicando la función sigmoidal y sin función sigmoidal, cuyos valores se encuentran en la *Tabla 5: Diferencia del R2 de aplicar función sigmoidal y no aplicar la función sigmoidal(Housing)* del apéndice. Cada casilla de dicha tabla representa la diferencia de aplicar el modelo KNN sobre las activaciones calculadas de forma habitual o sobre las entradas netas que recibe cada neurona oculta. Cuanto mayor sea el valor de esta diferencia, significará que, si se aplica la función sigmoidal, los resultados obtenidos por el modelo KNN son mejores, por lo tanto, que la reducción de la dimensionalidad aplicando la función sigmoidal es más eficaz que sin realizar el cálculo de esta función. El valor de cada barra del gráfico es la diferencia de los resultados obtenidos por el modelo KNN sobre los datos reducidos obtenidos por el Perceptrón mediante las dos técnicas elaboradas (activaciones de las neuronas ocultas y entradas netas a las neuronas ocultas); donde cada color representa el número de atributos de entrada introducidos en el modelo KNN (que se corresponde

con el número de neuronas ocultas del Perceptrón) y las columnas se encuentran agrupadas por el valor del parámetro k empleado en el modelo KNN.

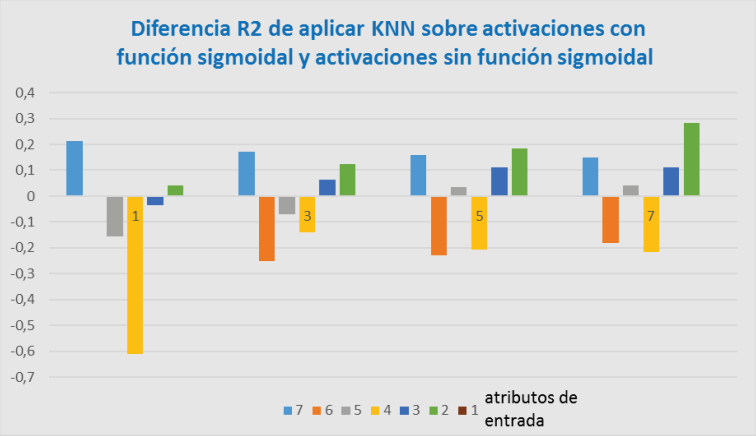


Gráfico 5: Diagrama de barras de la diferencia entre el R2 de aplicar KNN sobre activaciones con función sigmoidal y sin función sigmoidal (Housing)

Como puede observarse, dependiendo del método empleado para la reducción del número de atributos de entrada es más eficaz una técnica que otra; en caso de que el valor sea positivo es mejor obtener las activaciones de manera tradicional y en caso de que el valor sea negativo es mejor no aplicar la función sigmoidal para reducir la dimensionalidad de los atributos de entrada. En el caso de que los datos se vean reducidos a 6 o a 4, es más eficaz no calcular la función sigmoidal para realizar la transformación que reduce los datos. Por el contrario, en el caso de que los datos se reduzcan a 7 o a 2 atributos de entrada, es más eficaz calcular la función sigmoidal para reducir los datos. También se perciben casos en los que aplicar una técnica u otra es prácticamente igual, como cuando se reducen los datos a un atributo de entrada.

4.3.1.4. R2 PCA + KNN

En esta sección se muestra el coeficiente de determinación obtenido tras aplicar el modelo KNN a los datos reducidos mediante el método PCA, es decir, tras aplicar una transformación a los datos originales, obtener un listado de los componentes principales ordenados de mayor a menor relevancia y seleccionar los primeros componentes principales de la lista como atributos de entrada del modelo KNN. Los valores obtenidos se encuentran en la *Tabla 6: Coeficiente de determinación de aplicar KNN sobre los componentes principales (Housing)* del apéndice y cuyos resultados se ven reflejados en el siguiente gráfico.

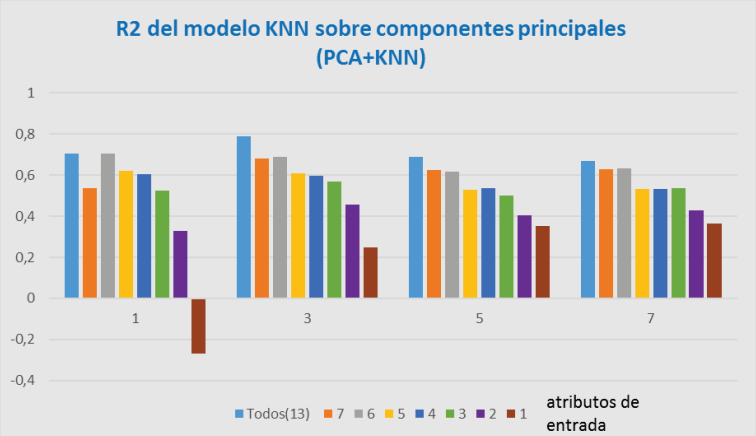


Gráfico 6: Diagrama de barras del modelo KNN sobre los componentes principales (Housing)

Como era de esperar se observa que tras aplicar el modelo KNN sobre todo el listado de componentes principales, se obtiene los mismos resultados que al aplicar el modelo KNN sobre los datos originales; ya que, los componentes principales son una trasformación lineal de los datos

originales. También puede observarse que, tras obtener los componentes principales y aplicar el modelo KNN, no se consigue mejorar el coeficiente de determinación logrado sobre los datos originales. El mejor resultado lo obtiene el modelo KNN con k=3 sobre todos los componentes principales. El mejor resultado obtenido tras aplicar una reducción a los datos es el que cuenta con 6 atributos de entrada y se tiene en cuenta el vecino más cercano en el modelo KNN. El peor de los resultados es el que se reducen los datos hasta un atributo de entrada y se observa únicamente el vecino más próximo que alcanza un valor negativo.

4.3.1.5. Comparación entre RNA+KNN y PCA+KNN

En esta sección se realiza una comparación sobre los resultados obtenidos tras aplicar la reducción de dimensionalidad de los datos mediante el Perceptrón multicapa y aplicar el modelo KNN; y reducir los datos mediante PCA y aplicar el modelo KNN. Para llevar a cabo dicha comparación se asume que todos los resultados negativos toman el valor cero. Como en este trabajo se han realizado dos formas de reducir la dimensión de los datos mediante las redes de neuronas artificiales: por un lado, calculando las activaciones de la capa oculta de forma habitual y por otro lado sin aplicar la función sigmoidal al sumatorio de las entradas por los pesos más el umbral (entrada neta); se presentan dos comparaciones.

4.3.1.5.1. Comparación entre RNA con función sigmoidal + KNN y PCA + KNN

En primer lugar, la *Tabla 7: Diferencia entre R2 cometido por el Perceptrón con función sigmoidal y PCA (Housing)*, representada mediante el siguiente gráfico, contiene el resultado de la diferencia del coeficiente de determinación obtenido por el modelo KNN con las activaciones de la capa oculta aplicando la función sigmoidal, menos el coeficiente de determinación obtenido tras aplicar el KNN a los componentes principales adquiridos mediante PCA. De este modo se muestra la diferencia de realizar la reducción de la dimensionalidad de los datos mediante el Perceptrón o mediante PCA, cuanto mayor sea el valor de la diferencia indica que la red de neuronas es capaz de reducir la dimensión de los atributos de manera más eficaz que el PCA. Por lo tanto, todos los valores positivos indican que la RNA es más eficaz, mientras que todos los valores negativos indican que el PCA es más eficaz para llevar a cabo la reducción de la dimensionalidad de los atributos de entrada.

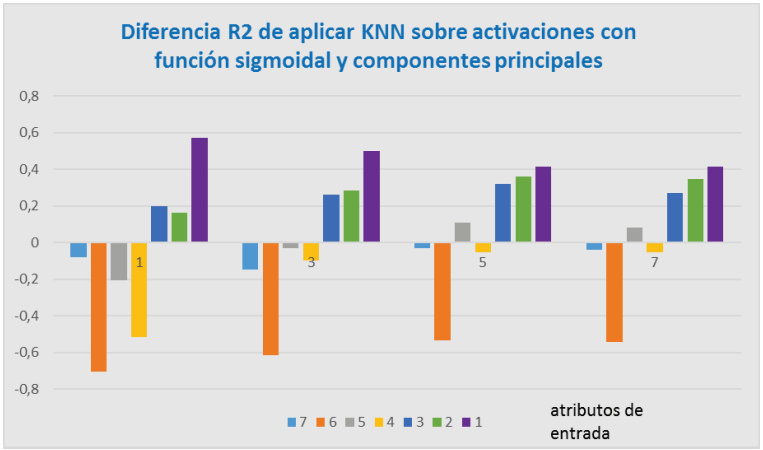


Gráfico 7: Diagrama de barras diferencia de R2 mediante RNA con función sigmoidal y PCA (Housing)

Se puede ver como en el caso de que los atributos se vean reducidos a 7,6, 5 o 4 atributos de entrada, es más eficaz emplear PCA que RNA; por el contrario, en caso de reducir a 3,2 o 1 atributo de entrada es más eficaz RNA que PCA, independientemente del número de vecinos que tenga en cuenta el modelo KNN. En resumen, se observa que cuanto más agresiva es la reducción realizada sobre los atributos de entrada, más merece la pena emplear RNA que PCA.

4.3.1.5.2. Comparación entre RNA sin función sigmoidal + KNN y PCA + KNN

Para concluir con las pruebas realizas sobre el conjunto de datos "Housing" se presenta el siguiente diagrama, la *Tabla 8: Diferencia entre R2 cometido por el Perceptrón sin función sigmoidal y PCA*

(Housing) del apéndice contiene los resultados de la diferencia del coeficiente de determinación tras aplicar el modelo KNN sobre los datos reducidos con el Perceptrón Multicapa mediante las entradas netas de las neuronas ocultas y sobre los componentes principales obtenidos por el modelo PCA. De este modo se está evaluando cuál de las dos técnicas de reducción de dimensionalidad es más eficaz, cuanto mayor sea el valor de la diferencia del coeficiente de determinación implicará que la red de neuronas sin realizar la función sigmoidal a las activaciones de la capa oculta es capaz de reducir la dimensión de los atributos de forma más eficaz que el modelo PCA con los componentes principales.

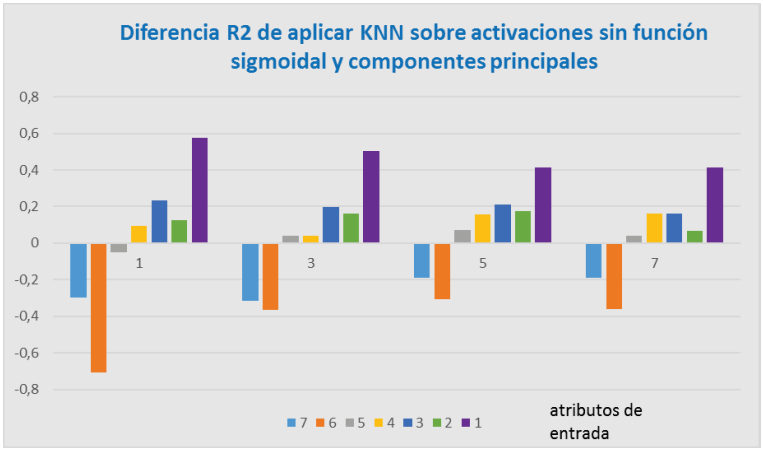


Gráfico 8: Diagrama de barras diferencia de R2 mediante RNA sin función sigmoidal y PCA (Housing)

Se puede deducir que, reducir la dimensionalidad de los atributos de entrada mediante RNA sin aplicar la función sigmoidal es más eficaz en caso de que los atributos se reduzcan a 4, 3, 2 y 1, independientemente del número de vecinos. En el caso de que se reduzcan a 7 o 6 atributos es más eficaz utilizar PCA que RNA sin función sigmoidal. En cambio, si los atributos de entrada se reducen a 5, depende del número de vecinos que tenga en cuenta el modelo KNN es más eficaz una técnica que otra. Al igual que en el caso anterior, cuanto más se ven reducidos los atributos de entrada, más compensa utilizar RNA frente a PCA.

4.3.2. Resultados obtenidos por el conjunto de datos House_16H

En esta sección se muestran los resultados obtenidos tras realizar los estudios de reducción de dimensionalidad con el conjunto de datos "House_16H".

4.3.2.1. R2 Perceptrón Multicapa

Tras realizar siete arquitecturas diferentes modificando el número de neuronas ocultas, se obtienen los resultados que se muestran en el siguiente diagrama y cuyos valores se encuentran en la *Tabla 9: Coeficiente de determinación redes de neuronas (House_16H)* del apéndice.

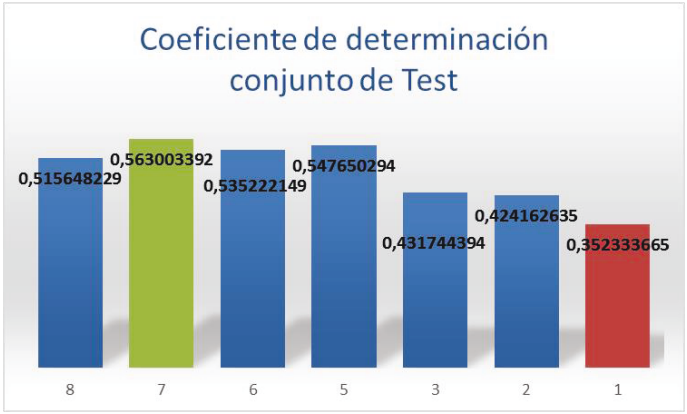


Gráfico 9: Diagrama de barras con R2 de Test RNA (House_16H)

El resultado que realmente interesa es el que se obtiene con el conjunto de test, ya que es el que indica cómo de bien se ajusta la red frente a datos nuevos. Se puede observar que el mejor resultado lo obtiene la red de neuronas que cuenta con 7 neuronas en su capa oculta, con un coeficiente de determinación de 0,563003; es decir, aquella RNA que reduce la dimensión de los atributos de entrada a 7. Por otro lado, el peor resultado 0,352334 lo obtiene la arquitectura que cuenta con 1 neurona oculta.

4.3.2.2. R2 KNN

Tras aplicar el modelo KNN sobre los datos "House_16H" originales se obtienen los coeficientes de determinación mostrados en la gráfica para cada valor del parámetro K, la *Tabla 10: Coeficiente de determinación KNN (House_16H)* del apéndice contiene estos valores.

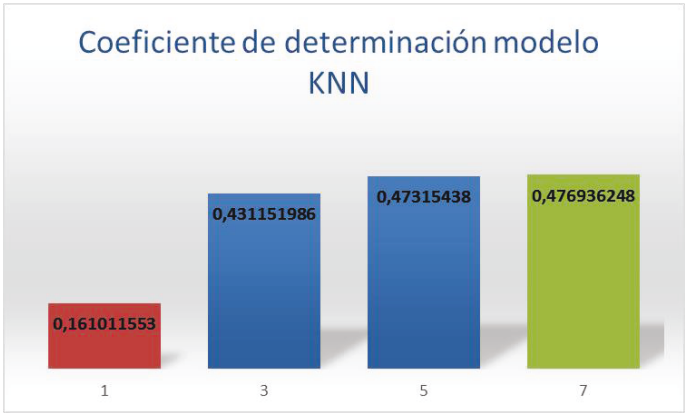


Gráfico 10: Diagrama de barras con R2 del modelo KNN (House_16H)

Teniendo en cuenta un único vecino el coeficiente de determinación es de 0,1610 que es el peor de los resultados; por el contrario, contando con los 7 vecinos más cercanos se obtiene el mejor resultado, que es de 0,4769. Se observa como los resultados van incrementando a medida que incrementa el número de vecinos a tener en cuenta.

4.3.2.3. R2 Perceptrón Multicapa + KNN

En esta sección se presentan los resultados obtenidos tras realizar la reducción de la dimensionalidad mediante redes de neuronas y aplicar a dichos datos el modelo KNN.

4.3.2.3.1. R2 Perceptrón Multicapa aplicando función sigmoidal + KNN

Tras realizar siete arquitecturas variando el número de neuronas de la capa oculta y obteniendo las activaciones de dichas neuronas de forma usual (calculando la función sigmoide); y aplicando el modelo KNN a dichas activaciones, se obtienen los coeficientes de determinación mostrados en la siguiente gráfica, cuyos valores se encuentran en la *Tabla 11: Coeficiente de determinación de aplicar KNN sobre las activaciones con función sigmoidal (House_16H)* del apéndice.

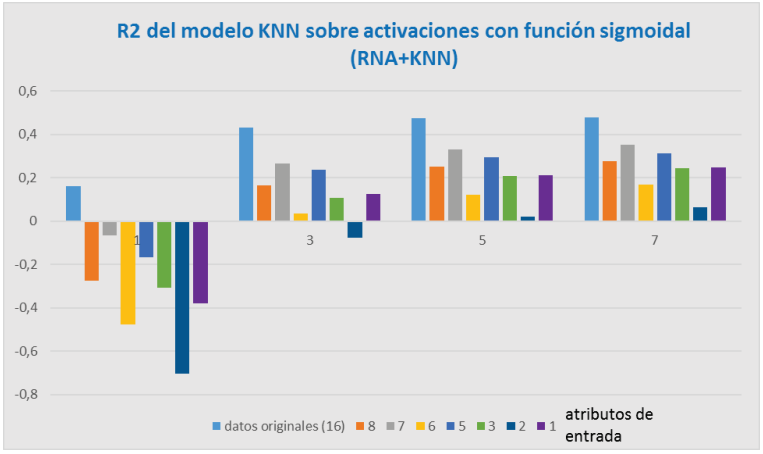


Gráfico 11: Diagrama de barras del modelo KNN sobre las activaciones con función sigmoidal (House_16H)

Donde se puede observar que los mejores resultados los alcanza el modelo KNN sobre los datos originales, es decir, en este caso la reducción nunca mejora los resultados originales. El mejor de los resultados tras reducir la dimensión de los atributos de entrada, lo obtiene el modelo KNN teniendo en cuenta 7 vecinos y con 7 atributos de entrada, cuyo coeficiente de determinación es de 0,3512. El peor resultado con bastante diferencia, es el que obtienen aquellos atributos que han sido reducidos a 2 atributos de entrada y se ha aplicado el modelo KNN teniendo de referencia el único vecino más cercano, cuyo valor es -0,7046. Es interesante destacar que, cuando el modelo KNN cuenta sólo con el vecino más próximo, el coeficiente de determinación siempre es negativo, independientemente del número de atributos de entrada a los que se haya reducido. También puede verse como los resultados no sobrepasan el valor 0,4.

4.3.2.3.2. R2 Perceptrón Multicapa sin aplicar función sigmoidal + KNN

En esta sección se revelan los resultados obtenidos tras realizar la reducción mediante RNA sin aplicar la función sigmoidal a las activaciones de la capa oculta y posteriormente aplicar a estos datos el modelo KNN. La siguiente gráfica representa estos valores recogidos en la *Tabla 12: Coeficiente de determinación de aplicar KNN sobre las activaciones sin función sigmoidal (House_16H)* del apéndice.

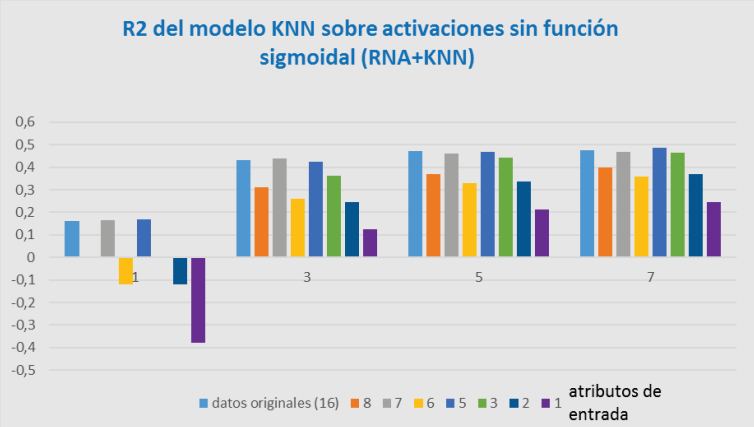


Gráfico 12: Diagrama de barras del modelo KNN sobre las activaciones sin función sigmoidal (House_16H)

El mejor resultado lo obtiene el modelo KNN que tiene en cuenta los 7 vecinos más cercanos y que se aplica a un conjunto de datos con 5 atributos de entrada, con un coeficiente de determinación de 0,48676. Por el contrario, el peor resultado es -0,37985 que lo obtiene el modelo KNN con el vecino más próximo sobre los datos que se han reducido a un único atributo de entrada. En comparación con los resultados obtenidos por el modelo KNN sobre los datos originales, se dan casos en los que se obtienen valores que superan ligeramente el coeficiente de determinación al aplicar una reducción en los atributos de entrada, por lo tanto, esté método en algunos casos es capaz de reducir la dimensionalidad de forma considerable y mantener los resultados que se obtienen con todos los atributos de partida.

4.3.2.3.3. Comparación entre RNA con función sigmoidal + KNN y RNA sin función sigmoidal + KNN

Para concluir con el experimento de RNA+KNN se realiza una comparación entre los resultados obtenidos tras reducir la dimensionalidad mediante redes de neuronas calculadas de forma usual o sin realizar la función sigmoidal (entradas netas a las neuronas ocultas). El siguiente gráfico muestra la diferencia entre el coeficiente de determinación obtenido por el modelo KNN sobre los datos adquiridos mediante las activaciones, menos el resultado del modelo KNN obtenido con los datos alcanzados por la RNA sin realizar la función sigmoidal, estos valores se encuentran en la *Tabla 13: Diferencia del R2 de aplicar función sigmoidal y no aplicar la función sigmoidal (House_16H)* existente en el apéndice.

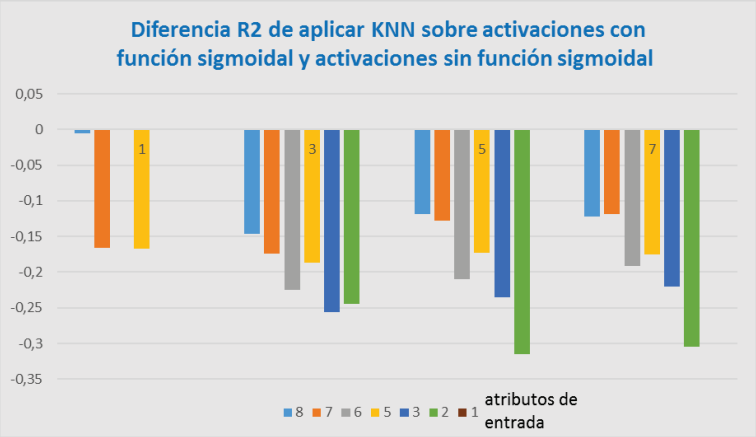


Gráfico 13: Diagrama de barras de la diferencia entre el R2 de aplicar KNN sobre activaciones con función sigmoidal y sin función sigmoidal (House_16H)

Cuanto mayor es el resultado, indica que es más eficaz reducir la dimensión de los atributos de entrada mediante RNA con función sigmoidal en lugar de realizarse mediante RNA sin función sigmoidal. Todos los valores que están por debajo de 0 (valores negativos) indican que es más eficaz calcular las activaciones sin aplicar la función sigmoidal. En este caso, en casi todas las reducciones de atributos y en todos los modelos KNN desarrollados, puede verse que es más eficaz no aplicar la función sigmoidal a la hora de obtener las activaciones de la capa oculta; en el resto de los casos, como al reducir los datos a 1 atributo de entrada, es indiferente realizar una técnica u otra.

4.3.2.4. R2 PCA + KNN

En este apartado se muestran los resultados obtenidos tras reducir la dimensionalidad de los atributos de entrada mediante PCA y aplicar el modelo KNN a dichos datos. El siguiente gráfico contiene el coeficiente de determinación obtenido por el modelo KNN con los componentes principales adquiridos con el método PCA, cuyos valores se encuentran en la *Tabla 14: Coeficiente de determinación de aplicar KNN sobre los componentes principales (House_16H)* del apéndice.

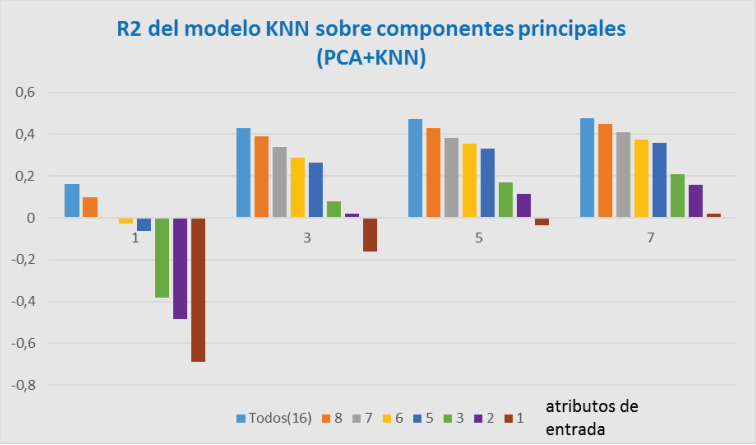


Gráfico 14: Diagrama de barras del modelo KNN sobre los componentes principales (House_16H)

En ningún caso, reducir los datos mediante PCA supera a los resultados obtenidos con los datos originales. Pero el mejor resultado tras reducir la dimensionalidad de los atributos de entrada mediante componentes principales y aplicar el modelo KNN es de 0,45089, que se corresponde con los datos que cuentan con los 8 primeros componentes principales de la lista y el modelo KNN tiene en cuenta los 7 vecinos más próximos. Se puede observar que, a medida que se va decrementando el número de componentes principales utilizados como atributos de entrada, va disminuyendo el valor del coeficiente de determinación obtenido por el modelo KNN, por lo que se va perdiendo información relevante.

4.3.2.5. Comparación entre RNA+KNN y PCA+KNN

Para concluir, se expone una comparación entre reducir la dimensionalidad de los datos mediante RNA y PCA. Como se han analizado dos técnicas diferentes para obtener las activaciones de la capa oculta mediante RNA se deben realizar dos comparaciones.

4.3.2.5.1. Comparación entre RNA con función sigmoidal + KNN y PCA + KNN

En primer lugar, en la siguiente gráfica se muestran los resultados obtenidos tras realizar la diferencia entre los resultados del modelo KNN aplicados a las activaciones de la capa oculta de la RNA con función sigmoidal, menos los resultados del modelo KNN sobre los componentes principales obtenidos mediante PCA; expuesto en la *Tabla 15: Diferencia entre R2 cometido por el Perceptrón con función sigmoidal y PCA (House_16H)* del apéndice. Como en todas las comparaciones realizadas en este trabajo, los coeficientes de determinación que tomen valores negativos, se asume que el valor es igual a 0.

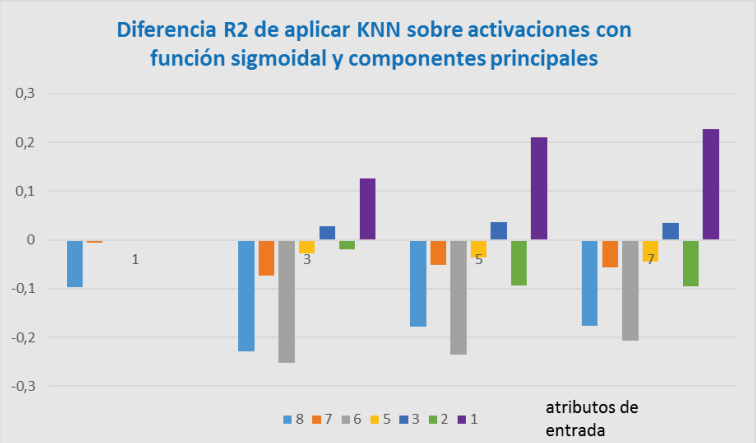


Gráfico 15: Diagrama de barras diferencia de R2 mediante RNA con función sigmoidal y PCA (House_16H)

Todos los valores que se encuentren por debajo del 0, indican que es más eficaz reducir la dimensionalidad de los datos mediante PCA; y los valores positivos revelan de forma contraría, que es más eficaz realizar la reducción mediante RNA con función sigmoidal. De este modo se puede concluir que en la mayoría de los casos esta diferencia alcanza valores negativos, por lo que es más eficaz reducir la dimensionalidad de los datos mediante PCA para este dominio; a excepción de que se reduzcan los atributos de entrada a uno sólo, en tal caso es más eficaz emplear RNA.

4.3.2.5.2. Comparación entre RNA sin función sigmoidal + KNN y PCA + KNN

Para concluir con las pruebas realizadas con el dominio "House_16H", en el siguiente diagrama se muestra la diferencia entre reducir los datos mediante RNA con las entradas netas de las neuronas ocultas y reducir los datos mediante componentes principales; los resultados se encuentran en la *Tabla 16: Diferencia entre R2 cometido por el Perceptrón sin función sigmoidal y PCA (House_16H)* del apéndice.

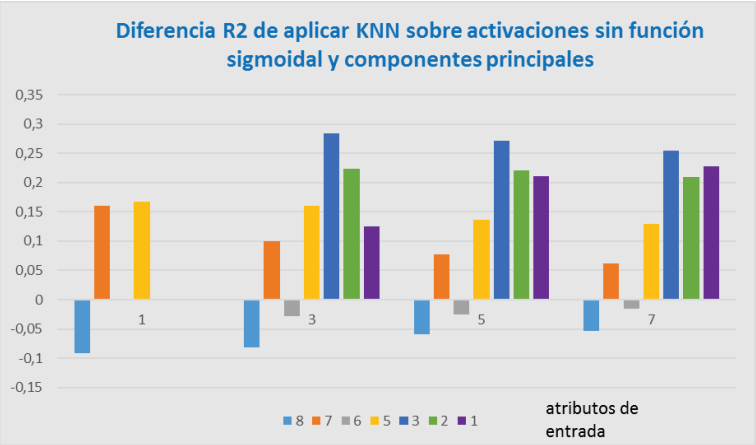


Gráfico 16: Diagrama de barras diferencia de R2 mediante RNA sin función sigmoidal y PCA (House_16H)

Los datos positivos indican que es más eficaz desempeñar RNA sin aplicar la función sigmoidal para reducir la dimensionalidad de los datos, que aplicar el modelo PCA. En este caso se ve como es más eficaz el modelo RNA sin aplicar la función sigmoidal para reducir la dimensionalidad de los datos que PCA, a excepción de que los datos se vean reducidos a 8 o 6 atributos donde es más eficaz el modelo PCA.

4.3.3. Resultados obtenidos por el conjunto de datos Elevators

En esta sección se muestran los resultados obtenidos al realizar los experimentos para reducir la dimensionalidad de los datos del dominio "Elevators".

4.3.3.1. R2 Perceptrón Multicapa

La siguiente gráfica muestra el coeficiente de determinación obtenido por el Perceptrón Multicapa al simular la red de neuronas con el subconjunto de test obtenido del dominio "Elevators", para conseguir así reducir el número de atributos de entrada mediante las activaciones de las neuronas de la capa oculta; estos resultados se muestran en la *Tabla 17: Coeficiente de determinación redes de neuronas (Elevators)* del apéndice.

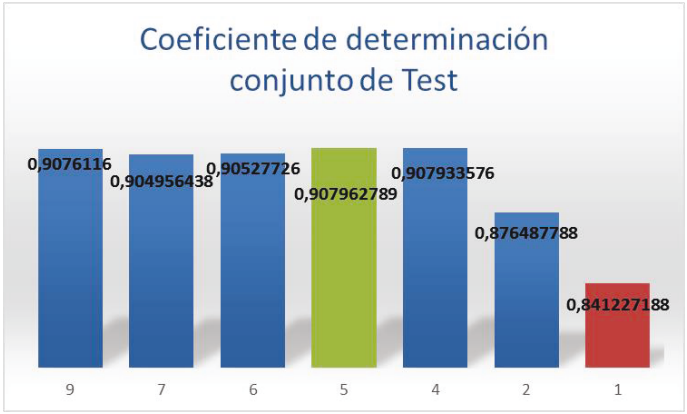


Gráfico 17: Diagrama de barras con R2 de Test RNA (Elevators)

Se puede observar como el mejor de los resultados en la fase de test son bastante buenos, ya que el mejor de los resultados es 0,90796 con 5 activaciones ocultas; y el peor es aquella red que cuenta con 1 neurona oculta cuyo coeficiente de determinación es de 0,841223. Es interesante destacar que, contando con 9, 7, 6, 5 y 4 neuronas ocultas, los resultados de los coeficientes de determinación son bastante similares y no bajan del valor 0,904.

4.3.3.2. R2 KNN

En la *Tabla 18: Coeficiente de determinación KNN (Elevators)* mostrada en el apéndice, se observan los coeficientes de determinación obtenidos tras aplicar el modelo KNN sobre el conjunto de datos "Elevators" original; cuyos datos se ven representados en el siguiente diagrama.

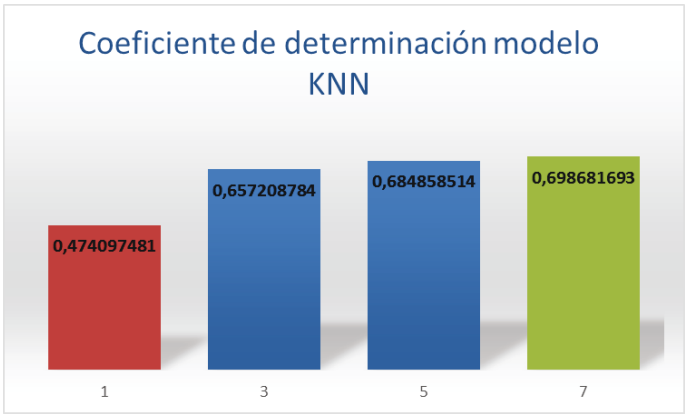


Gráfico 18: Diagrama de barras con R2 del modelo KNN (Elevators)

El peor resultado se obtiene con el modelo que solamente tiene en cuenta el vecino más cercano, con un valor de 0,4741; y el mejor de los resultados lo obtiene el modelo que cuenta con los 7 vecinos más cercanos a él, con un valor de 0,6987. Se puede ver cómo va incrementando el valor del coeficiente de determinación a medida que se incrementa el número de vecinos a tener en cuenta por el modelo KNN.

4.3.3.3. R2 Perceptrón Multicapa + KNN

Los resultados que se muestran en esta sección son los resultados obtenidos por el modelo KNN tras haber realizado una reducción de los datos mediante RNA.

4.3.3.3.1. R2 Perceptrón Multicapa aplicando función sigmoidal + KNN

En primer lugar, se muestran los resultados mediante el siguiente diagrama, del modelo KNN tras reducir la dimensionalidad de los datos con las activaciones de las neuronas de la capa oculta del Perceptrón Multicapa. Estas activaciones han sido calculadas de forma habitual, computando la función sigmoidal. Los resultados numéricos se muestran en el apéndice en la *Tabla 19: Coeficiente de determinación de aplicar KNN sobre las activaciones con función sigmoidal (Elevators)*.

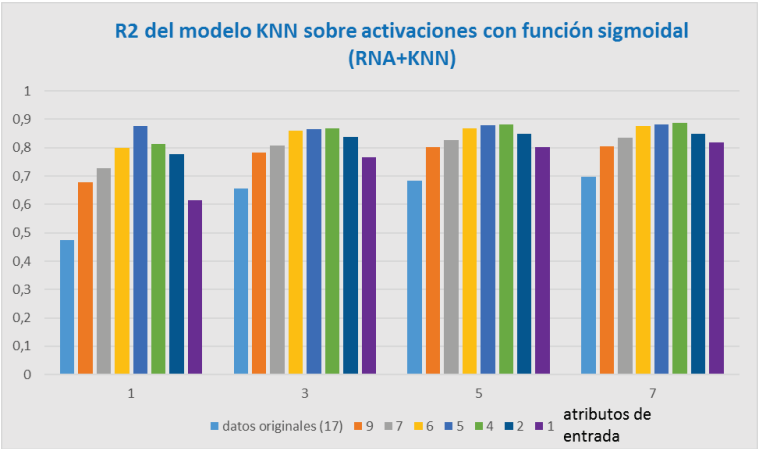


Gráfico 19: Diagrama de barras del modelo KNN sobre las activaciones con función sigmoidal (Elevators)

El mejor resultado obtenido por el modelo KNN es el que tiene en cuenta los 7 vecinos más próximos y donde los datos originales se han visto reducidos a 4 atributos de entrada, con un coeficiente de determinación igual a 0,8865. El peor resultado tras reducir la dimensión de los atributos, es de 0,6141 donde el modelo KNN cuenta con el único vecino más cercano y los atributos han sido reducidos hasta un solo atributo de entrada. Es notorio que, en todos los casos se obtienen mejores

resultados al reducir los datos mediante RNA aplicando la función sigmoidal, que al aplicar KNN directamente sobre los datos originales. Para este dominio en concreto los coeficientes de determinación alcanzados por el modelo KNN tras reducir la dimensión de los atributos de entrada mediante las activaciones de la capa oculta del Perceptrón multicapa, alcanza resultados muy buenos.

4.3.3.3.2. R2 Perceptrón Multicapa sin aplicar función sigmoidal + KNN

Los resultados obtenidos tras reducir la dimensionalidad de los atributos de entrada con RNA mediante las entradas netas de las neuronas ocultas y aplicar el modelo KNN se presentan en el siguiente diagrama, que se ha representado mediante la *Tabla 20: Coeficiente de determinación de aplicar KNN sobre las activaciones sin función sigmoidal (Elevators)* del apéndice.

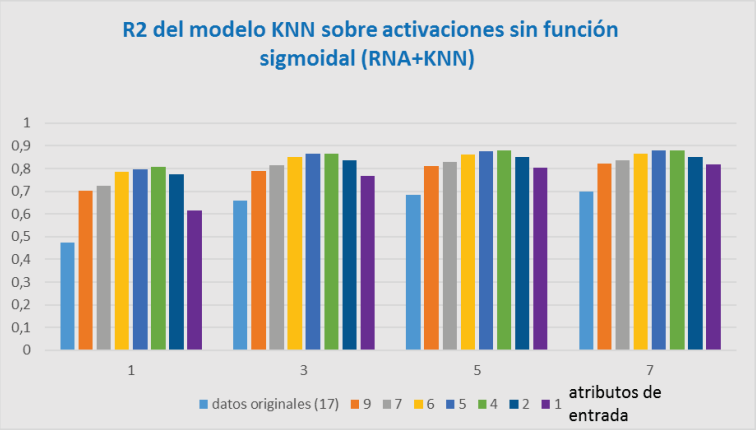


Gráfico 20: Diagrama de barras del modelo KNN sobre las activaciones sin función sigmoidal (Elevators)

Del mismo modo que en los resultados anteriores, el mejor de los casos se produce contando con 7 vecinos más cercanos en el modelo KNN y reduciendo los atributos a 4, correspondientes a las activaciones sin calcular la función sigmoidal (sumatorio de las entradas por los pesos más el umbral) de las 4 neuronas de la capa oculta. Hay que destacar que, en todos los casos, el coeficiente de determinación adquiere valores más altos cuando se reduce la dimensión de los atributos mediante RNA que aplicando al modelo KNN sobre los datos originales.

4.3.3.3.3. Comparación entre RNA con función sigmoidal + KNN y RNA sin función sigmoidal + KNN

Para concluir con los resultados obtenidos tras realizar la reducción de la dimensionalidad de los datos mediante el Perceptrón, se realiza una comparación entre las dos técnicas desarrolladas a la hora de reducir la dimensión. La comparación es la diferencia de los resultados obtenidos por el modelo KNN sobre los datos reducidos por RNA con función sigmoidal, menos los resultados obtenidos por KNN con los datos reducidos mediante RNA sin aplicar la función sigmoidal.

Esta comparación puede verse en el siguiente diagrama, cuyos valores están reflejados en la *Tabla 21: Diferencia del R2 de aplicar función sigmoidal y no aplicar la función sigmoidal (Elevators)* que aparece en el apéndice.

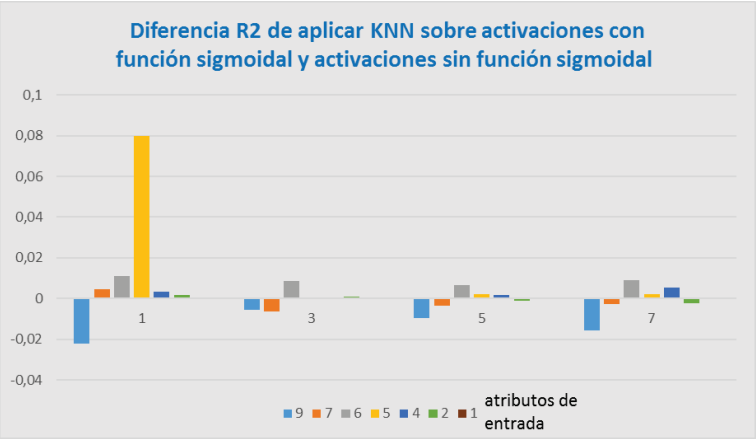


Gráfico 21: Diagrama de barras de la diferencia entre el R2 de aplicar KNN sobre activaciones con función sigmoidal y sin función sigmoidal (Elevators)

De este modo, se observa que todos los valores que se encuentren por encima del valor 0, indican que es más eficaz reducir la dimensión de los datos mediante RNA aplicando la función sigmoidal; por el contrario, en caso de adquirir valores negativos será más eficaz reducir la dimensionalidad de los datos sin aplicar la función sigmoidal. En este caso, es prácticamente igual de eficaz desempeñar una técnica u otra. Destaca del resto de diferencias el caso de que los datos se vean reducidos a 5 atributos de entrada y el modelo KNN tenga en cuenta el vecino más cercano, donde aplicar la función sigmoidal es más eficaz a la hora de reducir la dimensión de los datos.

4.3.3.4. R2 PCA + KNN

En esta sección se representan los resultados obtenidos tras reducir la dimensionalidad de los datos mediante PCA y posteriormente aplicar estos datos al modelo KNN; mediante el siguiente diagrama, cuyos datos se encuentran en la *Tabla 22: Coeficiente de determinación de aplicar KNN sobre los componentes principales (Elevators)* del apéndice.

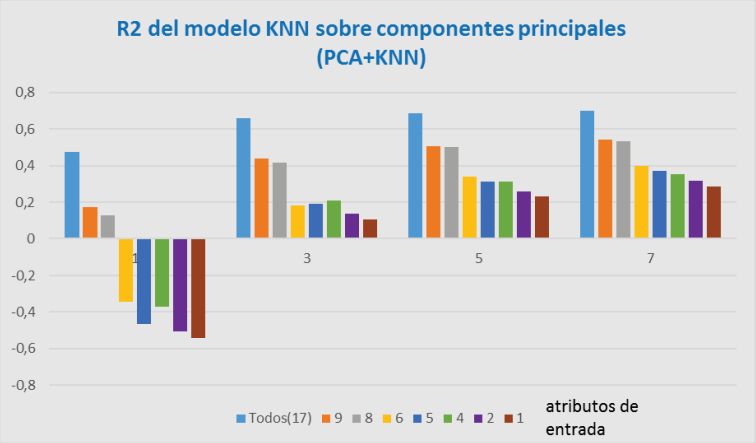


Gráfico 22: Diagrama de barras del modelo KNN sobre los componentes principales (Elevators)

Los mejores resultados alcanzados por el modelo KNN, independientemente del parámetro k, se obtienen sobre todos los componentes principales; que es el mismo resultado que obtiene el modelo sobre los datos originales. El mejor resultado se obtiene con todos los componentes principales y contando con los 7 vecinos más cercanos en el modelo KNN. El mejor resultado obtenido habiendo reducido la dimensión de los atributos de entrada, es 0,5401 partiendo con 9 atributos de entrada y aplicando el modelo KNN con los 7 vecinos más próximos. Por el contrario, el resultado más bajo lo adquiere el modelo KNN que únicamente tiene en cuenta el vecino más cercano y donde los datos se han reducido a un atributo de entrada. Se observa de forma muy clara como los peores resultados se obtienen con el modelo KNN que tiene en cuenta el vecino más cercano. Se aprecia como los mejores resultados se obtiene al reducir la dimensión de los atributos de entrada a 9,

correspondientes a los 9 primeros componentes principales obtenidos mediante PCA. También se observa que, en casi todos los casos, independientemente del valor del parámetro K, a medida que disminuye el número de atributos, desciende el coeficiente de determinación. Esto se debe a que según se reduce el número de componentes principales seleccionados, disminuye la información relevante.

4.3.3.5. Comparación entre RNA+KNN y PCA+KNN

En el siguiente apartado se muestra una comparación entre reducir la dimensión de los datos mediante RNA aplicando la función sigmoidal y el modelo PCA; y reducir la dimensionalidad de los atributos de entrada con RNA sin calcular la función sigmoidal y el método PCA. Cabe recordar que en todas las comparaciones entre modelos los coeficientes de determinación alcanzados por el modelo KNN sobre los datos que sean negativos, se asumen que toman el valor nulo.

4.3.3.5.1. Comparación entre RNA con función sigmoidal + KNN y PCA + KNN

En primer lugar, se realiza la diferencia entre reducir los datos mediante el Perceptrón Multicapa con las activaciones correspondientes a las neuronas de la capa oculta calculando la función sigmoidal y reducir los datos con los componentes principales obtenidos mediante el modelo PCA. Esta diferencia se ve reflejada en el siguiente diagrama de barras, cuyos datos se encuentran en la *Tabla 23: Diferencia entre R2 cometido por el Perceptrón con función sigmoidal y PCA (Elevators)* del apéndice.

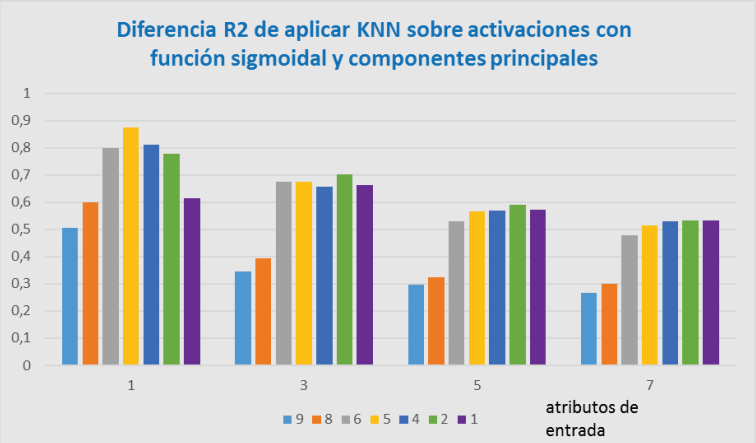


Gráfico 23: Diagrama de barras diferencia de R2 mediante RNA con función sigmoidal y PCA (Elevators)

En este gráfico se observa que todos los resultados son positivos y elevados, por lo que se determina que el modelo RNA calculando la función sigmoidal es mucho más eficaz que el modelo PCA, a la hora de reducir la dimensionalidad del conjunto "Elevators".

4.3.3.5.2. Comparación entre RNA sin función sigmoidal + KNN y PCA + KNN

Para concluir, se calcula la diferencia de los coeficientes de determinación obtenidos por el modelo KNN sobre las activaciones sin función sigmoidal, menos los resultados obtenidos por KNN sobre los componentes principales; estos resultados se ven reflejados en el siguiente diagrama, cuyos valores se encuentran en el apéndice, en la *Tabla 24: Diferencia entre R2 cometido por el Perceptrón sin función sigmoidal y PCA (Elevators)*.

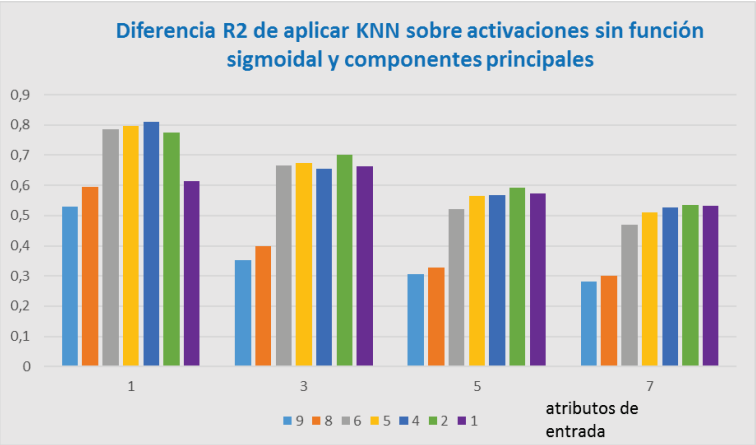


Gráfico 24: Diagrama de barras diferencia de R2 mediante RNA sin función sigmoidal y PCA (Elevators)

Del mismo modo que en la comparación anterior, todas las diferencias son positivas y elevadas, ya que, ningún valor baja de 0,28. Por lo tanto, se puede determinar que, en cualquier caso, es más eficaz reducir la dimensión del conjunto de datos "Elevators" mediante RNA sin calcular la función sigmoidal que mediante PCA.

4.3.4. Resultados obtenidos por el conjunto de datos Elevators girados

En este apartado se muestran los resultados obtenidos en todos los experimentos realizados con el dominio "Elevators girados", creado de manera artificial a partir del dominio anterior "Elevators".

4.3.4.1. R2 Perceptrón Multicapa

Mediante el siguiente diagrama, se muestran los coeficientes de determinación obtenidos al simular el Perceptrón Multicapa con diferentes neuronas en su capa oculta. La *Tabla 25: Coeficiente de determinación redes de neuronas (Elevators girado)* situada en el apéndice, contiene estos resultados.

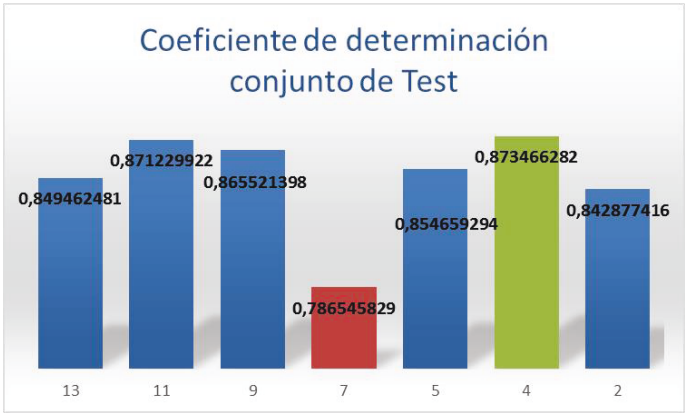


Gráfico 25: Diagrama de barras con R2 de Test RNA (Elevators girado)

El mejor resultado que obtiene la red es con una arquitectura que cuenta con 4 neuronas en su capa oculta y el valor del coeficiente de determinación alcanzado por esta red es de 0,873466. El peor valor obtenido es de 0,786546 con una arquitectura con 7 neuronas en su capa oculta. Se puede determinar que los valores alcanzados por el Perceptrón son bastante buenos.

4.3.4.2. R2 KNN

Seguidamente se muestran los resultados obtenidos por el modelo KNN con los datos originales del dominio "Elevators girado" mediante el siguiente diagrama, estos resultados se encuentran en la *Tabla 26: Coeficiente de determinación KNN (Elevators girado)* del apéndice.

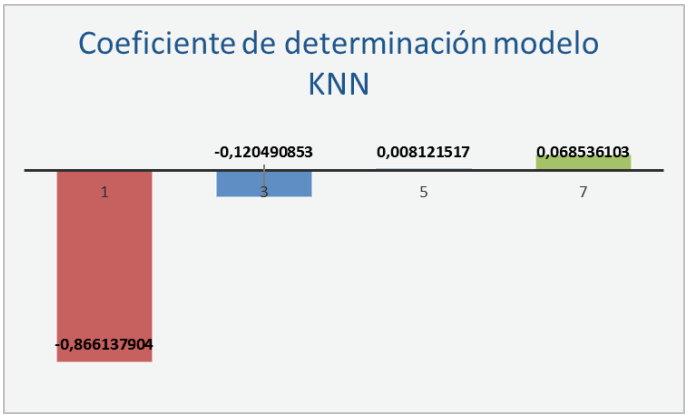


Gráfico 26: Diagrama de barras con R2 del modelo KNN (Elevators girado)

Se observa como los resultados obtenidos son bastante malos, debido a que es un dominio creado artificialmente a partir del conjunto de datos "Elevators" al que se le ha introducido nuevos atributos de forma aleatoria y han sido rotados, lo que ha introducido información irrelevante en el dominio.

4.3.4.3. R2 Perceptrón Multicapa + KNN

Al haberse realizado dos métodos diferentes para reducir los atributos de entrada mediante el Perceptrón Multicapa, en este apartado se muestran los resultados obtenidos con cada uno de ellos y una comparación entre los mismos.

4.3.4.3.1. R2 Perceptrón Multicapa aplicando función sigmoidal + KNN

En primer lugar, se muestran de forma gráfica, los resultados obtenidos por el modelo KNN con los datos reducidos mediante la obtención de las activaciones de las neuronas de la capa oculta de forma tradicional. Todos estos resultados se encuentran en el apéndice, en la *Tabla 27: Coeficiente de determinación de aplicar KNN sobre las activaciones con función sigmoidal (Elevators girado)*.

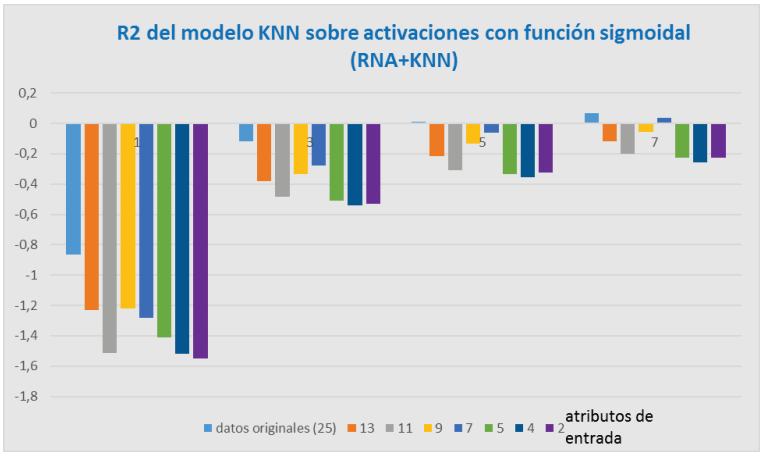


Gráfico 27: Diagrama de barras del modelo KNN sobre las activaciones con función sigmoidal (Elevators girado)

Se observa que los resultados obtenidos son extremadamente malos y casi todos los coeficientes de determinación son negativos; esto indica que las activaciones de las neuronas de la capa oculta que han sido utilizadas como entradas del modelo KNN, no han podido abstraer la información relevante, es decir, no ha sido capaz de ignorar los atributos irrelevantes e innecesarios que han sido introducidos al crear el dominio de manera artificial. Los mejores resultados alcanzados son por el modelo KNN sobre los datos originales pese a que son bastante bajos.

4.3.4.3.2. *R2 Perceptrón Multicapa sin aplicar función sigmoidal + KNN*

El segundo método para adquirir las activaciones de las neuronas ocultas, es obtener el valor de la entrada neta que recibe cada neurona oculta, sin calcular la función sigmoidal. Con estos nuevos datos reducidos se aplica el modelo KNN y se obtienen los resultados que se muestran en el siguiente diagrama, cuyos datos se encuentran en la *Tabla 28: Coeficiente de determinación de aplicar KNN sobre las activaciones sin función sigmoidal (Elevators girado)*.

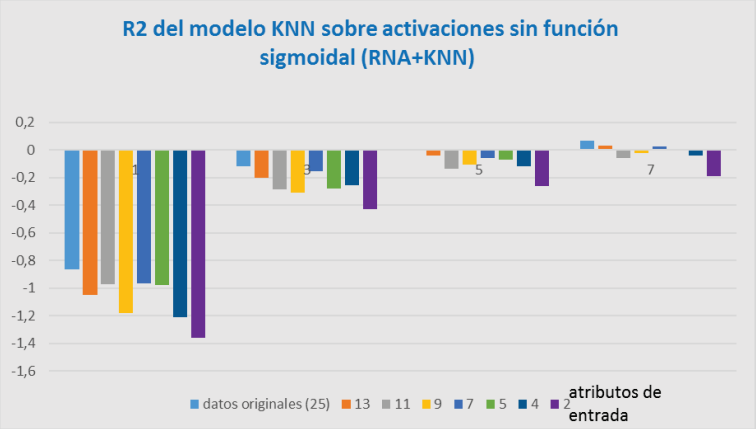


Gráfico 28: Diagrama de barras del modelo KNN sobre las activaciones sin función sigmoidal (Elevators girado)

Se observa que los resultados obtenidos son malos y casi todos los resultados adquieren valores negativos; por lo que demuestra que esta técnica de reducir la dimensionalidad de los datos no ha conseguido abstraer la información relevante e ignorar los atributos que han sido introducidos de manera aleatoria. Los mejores resultados se alcanzan sin realizar una reducción de los atributos de entrada. El mejor de los resultados obtenidos aplicando una reducción a los atributos de entrada, es 0,031138 que lo obtiene el modelo KNN teniendo en cuenta los 7 vecinos más próximos y donde los datos se han visto reducidos a 13 atributos de entrada.

4.3.4.3.3. *Comparación entre RNA con función sigmoidal + KNN y RNA sin función sigmoidal + KNN*

Para ultimar con los resultados obtenidos tras reducir el número de atributos de entrada mediante RNA, se realiza una comparación entre las dos técnicas llevadas a cabo para obtener las activaciones. Se asume como 0, aquellos coeficientes de determinación que tengan valor negativo. Para efectuar la comparación, se realiza la diferencia de resultados obtenidos por el modelo KNN sobre las activaciones obtenidas por el Perceptrón calculando función sigmoidal, menos los resultados sobre las activaciones sin calcular la función sigmoidal. Esta diferencia se ve reflejada en el siguiente gráfico, la *Tabla 29: Diferencia del R2 de aplicar función sigmoidal y no aplicar la función sigmoidal (Elevators girado)* contiene estos resultados.

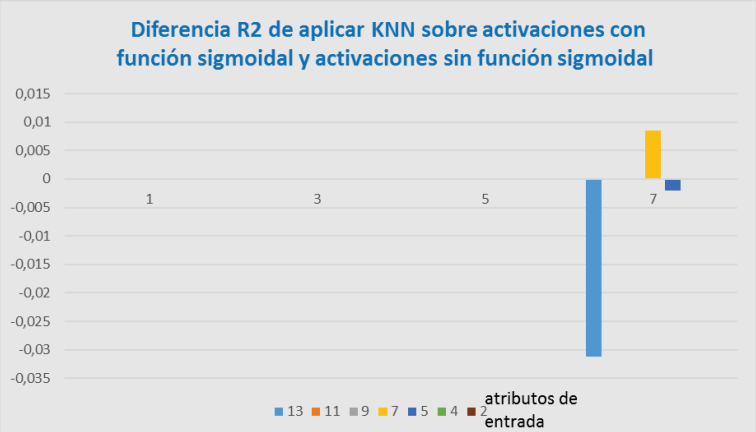


Gráfico 29: Diagrama de barras de la diferencia entre el R2 de aplicar KNN sobre activaciones con función sigmoidal y sin función sigmoidal (Elevators girado)

Como en ambas técnicas la mayoría de los resultados alcanzaban valores negativos y se ha asumido que toman el valor 0 para realizar la comparación, únicamente se compara aquellos resultados que han sido positivos, pero como puede verse la diferencia entre un modelo u otro es prácticamente inexistente. Por lo que se determina que es prácticamente igual de eficaz reducir los datos mediante RNA calculando la función sigmoidal o sin calcular la función sigmoidal, pese a que los resultados logrados para este dominio sean malos.

4.3.4.4. R2 PCA + KNN

En esta sección se exponen los resultados del último experimento realizado con el conjunto de datos artificial "Elevators girado", este último experimento consiste en obtener el listado de componentes principales mediante PCA y posteriormente, escogiendo los primeros componentes de la lita (los más relevantes) y aplicar el algoritmo KNN. Estos resultados se hallan en el apéndice en la *Tabla 30: Coeficiente de determinación de aplicar KNN sobre los componentes principales (Elevators girado)* y se reflejan en el siguiente diagrama.

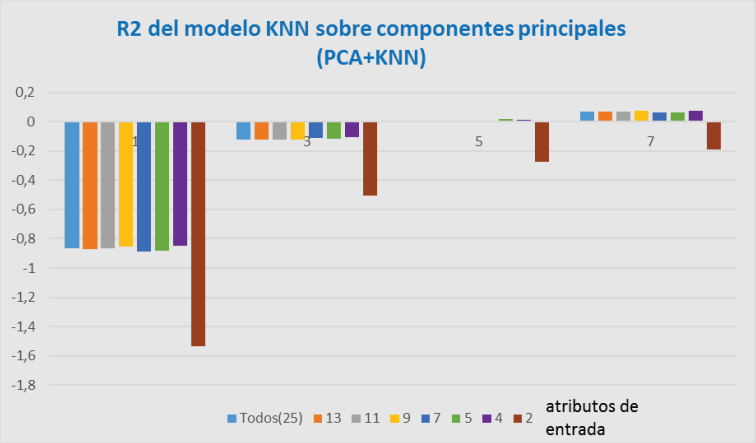


Gráfico 30: Diagrama de barras del modelo KNN sobre los componentes principales (Elevators girado)

Se aprecia que los resultados son malos y la mayoría de los resultados alcanzan valores negativos. El mejor resultado es el obtenido por el modelo KNN con K igual a 7 y donde los datos se han visto reducidos a 4 atributos de entrada, que supera el valor obtenido por el modelo KNN sobre los datos originales. Por lo que se deduce que PCA tampoco ha sido capaz de ignorar los atributos agregados de forma aleatoria a este dominio.

4.3.4.5. Comparación entre RNA+KNN y PCA+KNN

Para concluir con el conjunto de datos creado de manera artificial "Elevators girado", se muestra una comparación entre reducir la dimensión de los datos mediante RNA aplicando la función sigmoidal y el modelo PCA; y reducir la dimensionalidad de los atributos de entrada con RNA sin calcular la función sigmoidal y el método PCA. En todas las comparaciones entre modelos, los coeficientes de determinación alcanzados por el modelo KNN sobre los datos que sean negativos, se asumen que toman el valor nulo.

4.3.4.5.1. Comparación entre RNA con función sigmoidal + KNN y PCA + KNN

En primer lugar, se realiza la diferencia entre los resultados obtenidos por el modelo KNN sobre las activaciones adquiridas mediante RNA aplicando la función sigmoidal, menos los resultados del modelo KNN sobre los componentes principales calculados con PCA. Estos resultados se encuentran en la *Tabla 31: Diferencia entre R2 cometido por el Perceptrón con función sigmoidal y PCA (Elevators girado)* del apéndice y se reflejan en el siguiente gráfico.

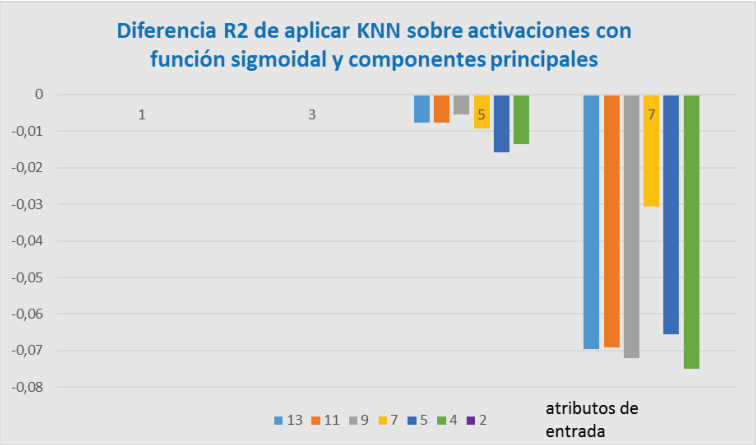


Gráfico 31: Diagrama de barras diferencia de R2 mediante RNA con función sigmoidal y PCA (Elevators girado)

Todos los valores por debajo de cero, indican que es más eficaz reducir la dimensión de los datos mediante PCA. En el caso de que K tome el valor 5 o 7, es más eficaz utilizar los componentes principales para reducir el número de atributos de entrada, pero la diferencia entre un método y otro es muy baja.

4.3.4.5.2. Comparación entre RNA sin función sigmoidal + KNN y PCA + KNN

Para concluir con los resultados obtenidos en el domino "Elevators girado" se calcula la diferencia entre reducir los datos mediante RNA sin calcular la función sigmoidal y reducir los datos mediante componentes principales obtenidos con PCA. En el apéndice, en la *Tabla 32: Diferencia entre R2 cometido por el Perceptrón sin función sigmoidal y PCA (Elevators girado)* se encuentran estas diferencias, que se muestran en el siguiente diagrama.

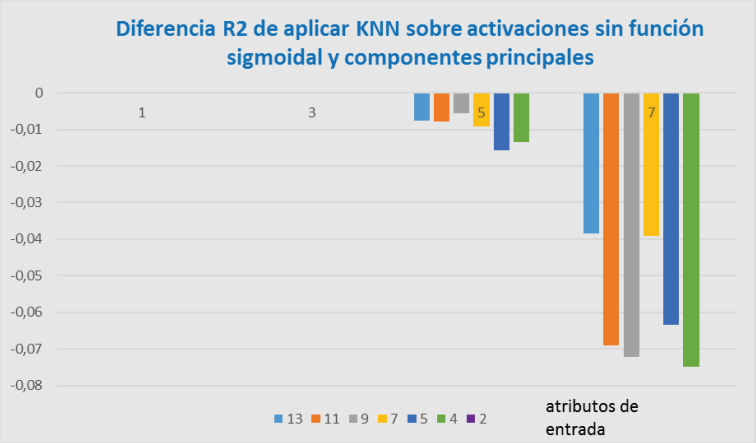


Gráfico 32: Diagrama de barras diferencia de R2 mediante RNA sin función sigmoidal y PCA (Elevators girado)

En este caso en particular la diferencia en la mayoría de los casos es nula, pero en algunos casos, como cuando K es 5 o 7, es un poco más eficaz reducir la dimensión mediante PCA.

4.3.5. Resultados obtenidos por el conjunto de datos Paraboloide girado

En el presente apartado se exponen los resultados obtenidos tras realizar los estudios con el conjunto de datos artificial "Paraboloide girado".

4.3.5.1. R2 Perceptrón Multicapa

El primer estudio realizado sobre el domino "Paraboloide girada", ha sido aplicar RNA para obtener las activaciones de las neuronas de la capa oculta. Los coeficientes de determinación al entrenar y simular la red, variando el número de neuronas en la capa oculta se muestran en la *Tabla 33: Coeficiente de determinación redes de neuronas (Paraboloide girado)* del apéndice, los resultados obtenidos en la fase de test se ven reflejados en el siguiente gráfico.

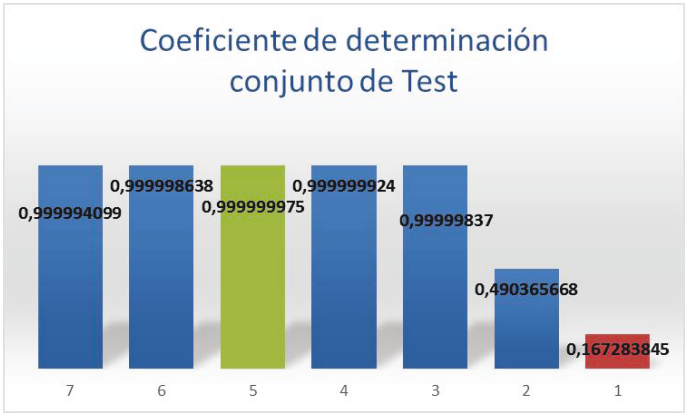


Gráfico 33: Diagrama de barras con R2 de Test RNA (Paraboloide girado)

Se aprecia como con 7, 6, 5, 4 y 3 neuronas ocultas, el coeficiente de determinación alcanzado por el Perceptrón en la fase de test es bastante similar y bueno, ya que, no desciende de 0,99999. En el momento que el número de neuronas en la capa oculta es 2 o 1, el resultado desciende de forma considerable, llegando a alcanzar 0,16728.

4.3.5.2. R2 KNN

Seguidamente se muestran los resultados obtenidos tras aplicar el modelo KNN sobre el conjunto de datos originales, tratado y procesado. Se muestran mediante el siguiente gráfico que contiene los resultados que se exponen en la *Tabla 34: Coeficiente de determinación KNN (Paraboloide girado)* presente en el apéndice.

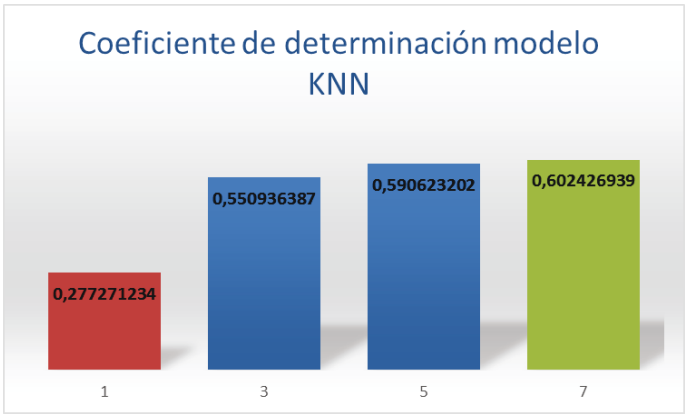


Gráfico 34: Diagrama de barras con R2 del modelo KNN (Paraboloide girado)

Se observa como a medida que se incrementa el número de vecinos, se incrementa el valor del coeficiente de determinación. Alcanzando el mayor coeficiente de determinación con K igual a 7, con un valor de 0,60243.

4.3.5.3. R2 Perceptrón Multicapa + KNN

En esta sección se presentan los resultados obtenidos tras realizar la reducción de la dimensionalidad mediante redes de neuronas y aplicar a dichos datos el modelo KNN. Para realizar la reducción mediante RNA se han desempeñado dos técnicas expuestas a continuación.

4.3.5.3.1. R2 Perceptrón Multicapa aplicando función sigmoidal + KNN

El siguiente diagrama muestra el coeficiente de determinación obtenido por el modelo KNN sobre las activaciones de la capa oculta del Perceptrón multicapa calculando la función sigmoidal. La *Tabla 35: Coeficiente de determinación de aplicar KNN sobre las activaciones con función sigmoidal (Paraboloide girado)* del apéndice, contiene estos resultados.

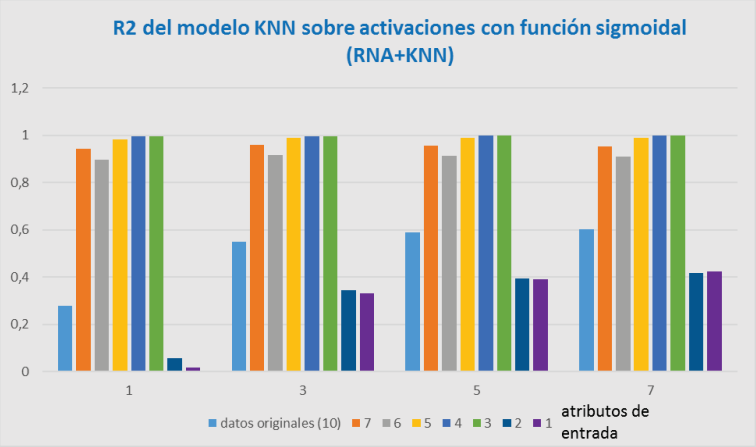


Gráfico 35: Diagrama de barras del modelo KNN sobre las activaciones con función sigmoidal (Paraboloide girado)

Se observa que, en la mayoría de los casos, a excepción de reducir a 2 o 1 atributo de entrada, los resultados obtenidos por el modelo KNN sobre conjunto de datos reducidos son mejores que los obtenidos por el modelo sobre los datos originales. Los resultados son muy buenos en caso de que los atributos de entrada se hayan reducido a 7, 6, 5, 4 o 3.

4.3.5.3.2. R2 Perceptrón Multicapa sin aplicar función sigmoidal + KNN

En este apartado se muestra una gráfica con los coeficientes de determinación obtenidos por el modelo KNN tras reducir la dimensión de los datos mediante RNA, esta reducción se obtiene realizando el sumatorio de las entradas por los pesos de las conexiones más el umbral sin aplicar función sigmoidal a este valor. La *Tabla 36: Coeficiente de determinación de aplicar KNN sobre las activaciones sin función sigmoidal (Paraboloide girado)* del apéndice contiene estos resultados.

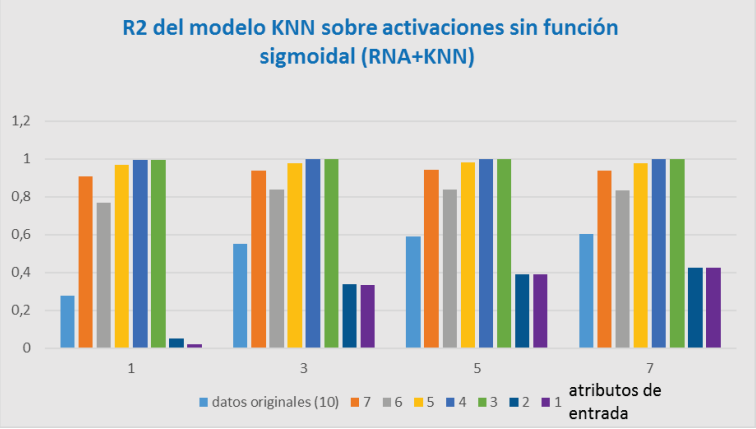


Gráfico 36: Diagrama de barras del modelo KNN sobre las activaciones sin función sigmoidal (Paraboloide girado)

Se percibe que reduciendo los datos a 7, 6, 5, 4 o 3 atributos de entrada en todos los modelos de KNN, se obtienen mejores resultados que con los datos originales. Sólo en el caso de que los datos

se vean reducidos a 2 o a 1 atributo de entrada los resultados son peores que con los datos de partida.

4.3.5.3.3. Comparación entre RNA con función sigmoidal + KNN y RNA sin función sigmoidal + KNN

En esta sección se realiza una comparación entre reducir la dimensionalidad de los datos con RNA mediante las activaciones de la capa oculta aplicando la función sigmoidal y sin aplicar la función sigmoidal. Para llevar a cabo esta comparación, se realiza la diferencia de los resultados obtenidos tras aplicar el modelo KNN sobre los datos reducidos mediante estas dos técnicas. En el siguiente diagrama se muestran los resultados, que se encuentran de forma detallada en la *Tabla 37: Diferencia del R2 de aplicar función sigmoidal y no aplicar la función sigmoidal (Paraboloide girado)* del apéndice.

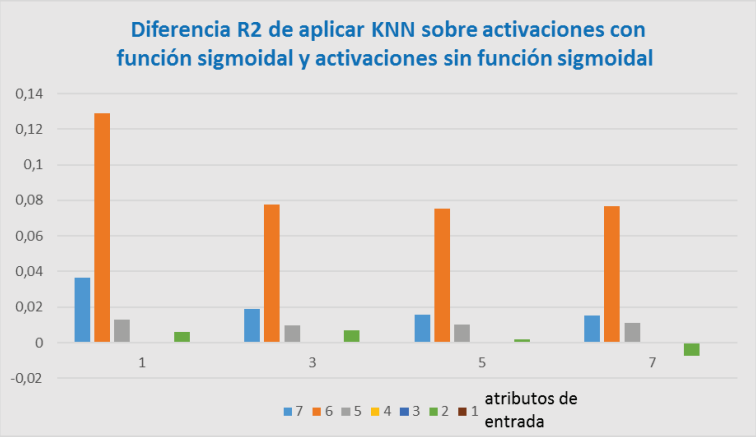


Gráfico 37: Diagrama de barras de la diferencia entre el R2 de aplicar KNN sobre activaciones con función sigmoidal y sin función sigmoidal (Paraboloide girado)

Todos los valores son positivos o nulos, lo que indica que es más eficaz, en el caso de los valores positivos, para reducir la dimensión de los datos emplear RNA calculando la función sigmoidal que sin calcularla; y que son igual de eficaces en el caso de que el valor sea nulo. Cabe destacar que donde más diferencia se nota, es cuando los datos se ven reducidos a 6 atributos de entrada, pese a que la diferencia no es elevada.

4.3.5.4. R2 PCA + KNN

En esta sección se representan los resultados obtenidos tras reducir la dimensionalidad de los datos mediante PCA y posteriormente aplicar estos datos al modelo KNN; mediante el siguiente diagrama, cuyos datos se encuentran en la *Tabla 38: Coeficiente de determinación de aplicar KNN sobre los componentes principales (Paraboloide girado)* del apéndice.

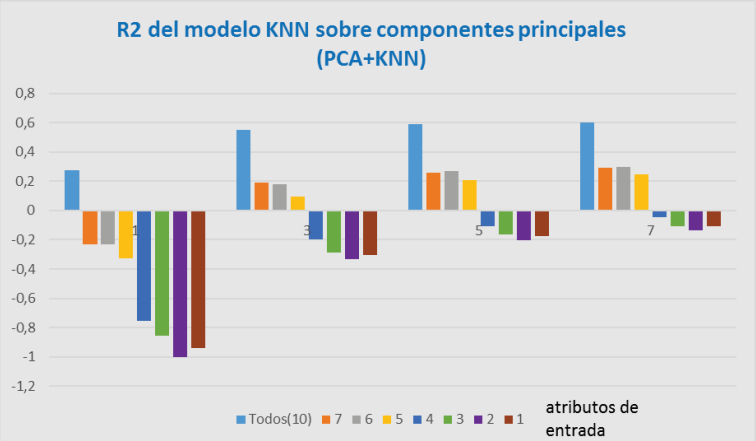


Gráfico 38: Diagrama de barras del modelo KNN sobre los componentes principales (Paraboloide girado)

Se ve como en todos los resultados obtenidos por el modelo KNN los mejores coeficientes alcanzados los adquiere con los atributos originales. En el caso de que el modelo únicamente tenga en cuenta al vecino más cercano, los resultados con los datos reducidos son muy malos. También se observa que al disminuir los datos a 4 o menos atributos de entrada el modelo KNN funciona mal.

4.3.5.5. Comparación entre RNA+KNN y PCA+KNN

Para concluir con el estudio realizado con el dominio "Paraboloide girado" se realiza una comparación entre reducir la dimensión de los datos mediante RNA o mediante PCA. Como en este estudio se han realizado dos reducciones diferentes con RNA, se realiza una comparación entre estas dos técnicas y el modelo PCA. Las comparaciones entre modelos se realizan realizando la diferencia entre resultados, asumiendo como nulos todos los coeficientes de determinación negativos.

4.3.5.5.1. Comparación entre RNA con función sigmoidal + KNN y PCA + KNN

En primer lugar, se realiza la comparación entre reducir los datos mediante Perceptrón Multicapa obteniendo las activaciones calculando la función sigmoidal y reducir los datos mediante PCA. Para ello se realiza la diferencia entre el primer modelo menos el segundo, de este modo cuanto mayor sean los resultados obtenidos más eficaz será utilizar RNA para disminuir la dimensión de los atributos de entrada. El siguiente diagrama muestra esta diferencia, la *Tabla 39: Diferencia entre R2 cometido por el Perceptrón con función sigmoidal y PCA (Paraboloide girado)* del apéndice contiene estos valores.

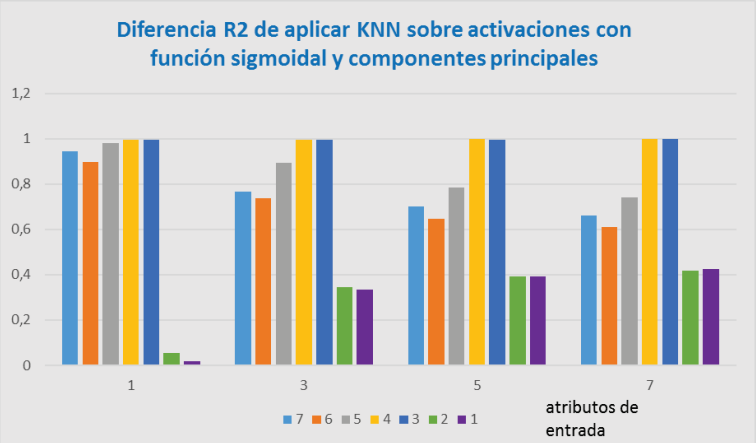


Gráfico 39: Diagrama de barras diferencia de R2 mediante RNA con función sigmoidal y PCA (Paraboloide girado)

En todos los casos es más eficaz reducir la dimensión de los datos mediante RNA, obteniendo las activaciones de las neuronas ocultas de la red. También hay que destacar que donde menos diferencia se aprecia es cuando los datos se ven reducidos a 2 o a 1 atributo de entrada. Hay casos en los que la diferencia casi alcanza la unidad, por lo que se considera una diferencia de eficacia muy notoria.

4.3.5.5.2. Comparación entre RNA sin función sigmoidal + KNN y PCA + KNN

Para concluir con las pruebas realizas sobre el conjunto de datos "Paraboloide girado" se presenta el siguiente diagrama, donde los resultados se encuentran en la *Tabla 40: Diferencia entre R2 cometido por el Perceptrón sin función sigmoidal y PCA (Paraboloide girado)* del apéndice, que contiene los resultados de la diferencia del coeficiente de determinación tras aplicar el modelo KNN sobre los datos reducidos mediante el Perceptrón Multicapa sin realizar la función sigmoidal a las activaciones de las neuronas de la capa oculta y el modelo PCA.

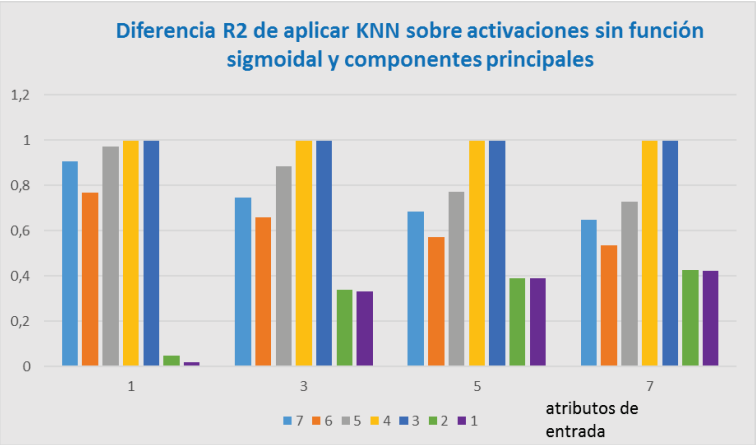


Gráfico 40: Diagrama de barras diferencia de R2 mediante RNA sin función sigmoidal y PCA (Paraboloide girado)

En todos los casos y con una diferencia bastante grande en la mayoría de ellos, es más eficaz reducir la dimensión de los datos mediante el Perceptrón Multicapa sin calcular la función sigmoidal que mediante el PCA. Únicamente en el caso de reducir los datos a 2 o 1 atributo de entrada y con un modelo KNN que solamente tiene en cuenta el vecino más cercano la diferencia entre las dos técnicas de reducción es más insignificante.

4.4. Análisis de los resultados

La intención de este apartado es realizar un análisis de los resultados obtenidos tras realizar todos los experimentos anteriores, y poder determinar, si es posible realizar una reducción de la dimensionalidad del conjunto de datos originales desempeñando estas técnicas y determinar si es más exitosa reducir la dimensión de los atributos de entrada con técnicas de aprendizaje supervisado frente a técnicas estadísticas desarrolladas para reducir dicha dimensión (PCA).

Los análisis se realizarán de forma individual, es decir, se realizará un análisis por cada dominio utilizado; y posteriormente se realizará un análisis general de todos los conjuntos de datos para obtener así una visión global del estudio. Todos estos análisis se realizarán siguiendo la *Ilustración 14: Esquema del sistema*.

4.4.1. Análisis de los resultados de Housing

Se realiza un análisis de los resultados obtenidos con el dominio real "Housing"; todas las tablas con los resultados obtenidos se encuentran en el apartado *1.Resultados obtenidos por el conjunto de datos Housing* del apéndice.

En primer lugar, se realiza un análisis de los resultados alcanzados por el modelo KNN sobre las reducciones obtenidas mediante RNA calculando la función sigmoidal, para ello se realiza una comparación entre los resultados obtenidos por el modelo KNN sobre los datos reducidos mediante esta técnica y sobre los datos originales.

Nº vecinos / Nº neuronas	1	3	5	7
7	-0,251514175	-0,254926812	-0,098072616	-0,079122051
6	-0,705629836	-0,71755023	-0,606195065	-0,579278587
5	-0,289551177	-0,212743919	-0,055473265	-0,056084025
4	-0,619887044	-0,295108447	-0,204754096	-0,19294542
3	0,017114934	0,038675511	0,131144118	0,138644101
2	-0,213658948	-0,050280505	0,074139731	0,107866461
1	-0,131084733	-0,038791407	0,074641211	0,111368969

Tabla 3: Diferencia RNA con función sigmoide + KNN y KNN sobre datos originales (Housing)

Cada una de las celdas de la tabla representa la diferencia entre los resultados obtenidos tras reducir los datos originales mediante RNA con función sigmoidal menos los resultados sobre los datos originales. Cuanto mayor sea el valor alcanzado, indicará que es más eficaz realizar una reducción de los datos mediante RNA que no reducir los datos. Se observa como en 8 de los experimentos realizados es más eficaz reducir los datos, también se aprecia como el modelo mejora independientemente del valor de k, cuando los datos se ven reducidos a 3 atributos de entrada. Si se selecciona correctamente el número de vecinos, es posible concentrar la información en unos pocos atributos (entre 1 y 3) e incluso mejorar el resultado obtenido sobre los datos originales. En el mejor de los casos, el resultado mejora un 13,86%, reduciendo los atributos de entrada a 3 con K igual a 7 frente a no realizar una reducción sobre los datos. Los peores resultados se obtienen al reducir a 6 el número de atributos de entrada.

Seguidamente se realiza el análisis de los resultados obtenidos por el modelo KNN tras reducir los datos con RNA mediante las entradas netas a las neuronas ocultas (sin calcular la función sigmoidal), para ello se realiza una comparación entre los resultados alcanzados tras reducir los datos con esta técnica y no reducir los datos.

Nº vecinos / Nº neuronas	1	3	5	7
7	-0,465826108	-0,425259166	-0,256215141	-0,229370344
6	-0,705629836	-0,467295911	-0,378064175	-0,399533755
5	-0,133725541	-0,141866148	-0,091454352	-0,097793575
4	-0,009079388	-0,153821516	0,002614377	0,022873255
3	0,051243384	-0,023534703	0,021187662	0,02664716
2	-0,25429463	-0,175446585	-0,108891645	-0,174051249
1	-0,131084733	-0,038791407	0,074363478	0,108316241

Tabla 4: Diferencia RNA sin función sigmoide + KNN y KNN sobre datos originales (Housing)

Se observa que, eligiendo correctamente el número de vecinos, es posible concentrar toda la información en unos pocos atributos de entrada, incluso en algunos casos, produciéndose una ligera ganancia. El mejor de los casos es aquel donde se obtiene un resultado de 0,108 (10,8%) de diferencia al reducir la dimensión a 1 atributo de entrada frente a no reducir la dimensionalidad de los atributos de entrada. La pérdida de información relevante en muchos casos es baja, mientras que la dimensión de los atributos de entrada se ve reducida de forma considerable.

Para concluir con las observaciones, se realiza el análisis de los resultados obtenidos por el modelo KNN tras reducir la dimensión de los datos mediante PCA, para ello se realiza la diferencia entre los resultados obtenidos tras realizar la reducción con esta técnica, menos los resultados obtenidos sobre los datos originales.

Nº vecinos / Nº componentes	1	3	5	7
7	-0,1701016	-0,10921643	-0,06620897	-0,04049551
6	0,00051334	-0,10133319	-0,07138231	-0,03719009
5	-0,08468686	-0,18342483	-0,16277534	-0,13599493
4	-0,10135867	-0,19526405	-0,15307071	-0,13793291
3	-0,18015228	-0,22302257	-0,18924593	-0,13412515
2	-0,37810602	-0,33589461	-0,28558661	-0,2401123
1	-0,70562984	-0,54093749	-0,3384949	-0,30324533

Tabla 5: Diferencia PCA+ KNN y KNN sobre datos originales (Housing)

Se puede apreciar como solamente se da un caso en el que se mejore el resultado del modelo KNN tras realizar una reducción de los datos mediante PCA, cuando se obtienen los 6 primeros componentes principales y el modelo KNN tiene en cuenta el vecino más cercano; y es una mejora insignificante 0,0513%. Para este dominio en concreto, PCA no es capaz de reducir la dimensionalidad de una manera razonable, al menos comparada con los resultados anteriores logrados por las RNA.

Como conclusión para RNA+KNN, se puede determinar que seleccionando adecuadamente el número de atributos de entrada y el número de vecinos a tener en cuenta por el modelo KNN, se pueden mantener o incluso mejorar los resultados obtenidos con los atributos de entrada originales, pero habiendo reducido considerablemente la dimensión de los datos (50% o más); esto indica que la información relevante se está concentrando en los atributos de entrada al realizar las transformaciones para reducir la dimensión, mediante las activaciones de las neuronas ocultas o mediante las entradas netas a dichas neuronas.

Por otro lado, mediante el experimento PCA+KNN, los resultados alcanzados no superan a los resultados alcanzados sobre los datos originales, a excepción de un caso. Pero dependiendo del número de componentes principales que se seleccione y del modelo KNN, se produce menor pérdida de información.

La siguiente tabla muestra la diferencia entre los resultados obtenidos tras reducir la dimensión de los atributos de entrada mediante el mejor de los resultados alcanzados por el Perceptrón Multicapa (ya sea calculando la función sigmoidal o sin calcularla), que emplea aprendizaje supervisado; frente a los resultados alcanzados por el modelo KNN sobre los componentes principales obtenidos por PCA. De esta manera, se puede determinar uno de los objetivos del trabajo, es posible mejorar los resultados de algoritmos creados específicamente para reducir la dimensionalidad de los atributos de entrada con algoritmos que emplean aprendizaje supervisado. Si la celda de la tabla adquiere un porcentaje positivo, significará el porcentaje de mejora de aplicar RNA frente a PCA; si por el contrario toma un valor negativo, indicará el porcentaje de mejora de PCA frente a RNA.

Nº vecinos / Nº atributos	1	3	5	7
7	-8,14125717%	-14,5710381%	-3,18636424%	-3,86265407%
6	-70,6143175%	-36,5962725%	-30,668187%	-36,2343662%
5	-4,90386799%	4,15586864%	10,7302073%	7,99109094%
4	9,22792828%	4,14425306%	15,5685083%	16,0806162%
3	23,1395662%	26,1698082%	32,0390045%	27,2769255%
2	16,4447075%	28,5614105%	35,972634%	34,7978757%
1	57,4545103%	50,2146086%	41,3136107%	41,4614296%

Tabla 6: Porcentaje de la diferencia entre RNA+KNN y PCA+KNN (Housing)

Las celdas de la tabla de color naranja representan el porcentaje de mejora de los resultados obtenidos tras reducir la dimensión de los atributos de entrada mediante PCA y las celdas azules indican el porcentaje de mejora tras reducir el número de atributos de entrada mediante el Perceptrón Multicapa. Se observa como es mayor el número de casos en los que el modelo KNN obtiene mejores resultados sobre los datos reducidos mediante RNA. Cuanto más se reduce el número de atributos, más eficaz es emplear RNA frente a PCA para reducir la dimensionalidad.

Por lo tanto, se puede concluir que para este dominio en particular se obtienen mejores resultados reduciendo la dimensionalidad mediante RNA que mediante PCA; es decir, se obtienen mejores resultados aplicando una técnica de aprendizaje supervisado que reduciendo la dimensionalidad de los datos mediante PCA que está especializado en ello. De este modo, se puede determinar que las neuronas de la capa oculta del Perceptrón Multicapa están concentrando la información relevante de los atributos de entrada originales en un porcentaje menor de atributos. Por otro lado, se puede determinar que reduciendo la dimensión de los atributos de entrada a un número concreto y escogiendo el valor del parámetro k del modelo KNN de forma adecuada, es viable reducir los atributos de entrada mediante estás técnicas; ya que en algunos casos la pérdida de información es pequeña y en otros, se llega incluso a mejorar el resultado alcanzado sobre los datos originales.

4.4.2. Análisis de los resultados de House_16H

En este apartado se realiza un análisis de los resultados obtenidos tras realizar los experimentos con el dominio real "House_16H", cuyos resultados se encuentran en el apartado *2.Resultados obtenidos por el conjunto de datos House_16H* del apéndice.

Para comenzar, al igual que en el apartado anterior, se realiza una comparación entre los resultados obtenidos tras realizar una reducción de los datos mediante las activaciones del Perceptrón Multicapa aplicando la función sigmoidal y los resultados obtenidos sobre los datos originales; como en todas las comparaciones del trabajo, todos los coeficientes de determinación que alcancen valores negativos se tomarán como cero.

Nº vecinos / Nº neuronas	1	3	5	7
8	-0,161011553	-0,26790491	-0,221628233	-0,201441293
7	-0,161011553	-0,167109862	-0,142300098	-0,125766842
6	-0,161011553	-0,395678391	-0,353368698	-0,308428377
5	-0,161011553	-0,192821616	-0,178014924	-0,164882318
3	-0,161011553	-0,324175642	-0,26578661	-0,231548317
2	-0,161011553	-0,431151986	-0,452263368	-0,414193536
1	-0,161011553	-0,306117486	-0,262178123	-0,23095256

Tabla 7: Diferencia RNA con función sigmoide + KNN y KNN sobre datos originales (House_16H)

Se observa que en todos los casos se obtienen mejores resultados sin reducir la dimensionalidad de los atributos de entrada, ya que todos los valores son negativos. Es necesario comentar, que no siempre se puede esperar una mejora en los resultados tras reducir la dimensionalidad de los atributos de entrada. En el mejor de los casos, la diferencia de no reducir la dimensión de los atributos no supera el 13%, frente a haber reducido los datos a 7 atributos de entrada con K igual a 7; la pérdida de información no es demasiado elevada frente a una reducción del 42,86% sobre los atributos de entrada.

La siguiente tabla muestra la comparación entre los resultados obtenidos tras reducir los datos mediante RNA sin calcular la función sigmoidal y los resultados obtenidos con los datos originales.

Nº vecinos / Nº neuronas	1	3	5	7
8	-0,15539112	-0,12180502	-0,102528	-0,07971149
7	0,00461196	0,00639293	-0,01396053	-0,00739036
6	-0,16101155	-0,1711967	-0,14373455	-0,11734254
5	0,00588239	-0,00550155	-0,00490761	0,00981923
3	-0,16101155	-0,06854003	-0,03031304	-0,01180357
2	-0,16101155	-0,1871987	-0,13728784	-0,10913232
1	-0,16101155	-0,30610512	-0,2625311	-0,23051837

Tabla 8: Diferencia RNA sin función sigmoide + KNN y KNN sobre datos originales (House_16H)

Se aprecia como dependiendo del número de neuronas que se otorguen a la capa oculta de la red y el número establecido para el parámetro K, se dan casos en los que se mejoran los resultados obtenidos sobre los datos originales. Esta mejora no es significativa, pero la reducción de la dimensionalidad es considerable. La mayor diferencia de no reducir los datos es de un 30,61% mejor que reduciendo los atributos de entrada; por lo que se puede determinar que, en la mayoría de los casos, compensa reducir la dimensión de los atributos de entrada ya que la pérdida de información no es muy significativa en comparación con la reducción de la dimensionalidad.

Para concluir con las comparaciones, se realiza la diferencia entre los resultados alcanzados tras reducir los atributos de entrada mediante PCA y los resultados obtenidos sobre los datos originales.

Nº vecinos / Nº componentes	1	3	5	7
8	-0,06358204	-0,04000641	-0,04390007	-0,02604803
7	-0,15543903	-0,09337956	-0,09127977	-0,06858327
6	-0,16101155	-0,14282026	-0,11858782	-0,10207775
5	-0,16101155	-0,16550899	-0,14157338	-0,11973753
3	-0,16101155	-0,35243599	-0,30173012	-0,2662252
2	-0,16101155	-0,4113607	-0,35858071	-0,31846203
1	-0,16101155	-0,43115199	-0,47315438	-0,45789208

Tabla 9: Diferencia PCA+ KNN y KNN sobre datos originales (House_16H)

Se aprecia como en ninguno de los casos se mejoran los resultados tras realizar la reducción de los atributos de entrada con los componentes principales obtenidos por el modelo PCA. En el peor de los casos, el resultado es un 47,31% mejor con los atributos originales frente a la reducción obtenida con PCA. Se aprecia como a medida que disminuye el número de atributos de entrada, la mejora de no reducir la dimensionalidad de los datos va incrementando, por lo tanto, según se reduce el número de componentes principales seleccionados como entradas del modelo KNN, se va perdiendo información relevante.

Se puede concluir que para el domino "House_16H" dependiendo del número al que se reduzcan los atributos de entrada y el valor que se establezca para el parámetro K, se produce una mayor o menor pérdida de información relevante; únicamente en el caso de reducir los datos mediante las entradas netas de las neuronas ocultas del Perceptrón (sin calcular la función sigmoidal), se mejoran los resultados obtenidos en algunos casos. En la mayoría de los casos la pérdida de información es pequeña, por lo que habría que estudiar dependiendo del objetivo, si compensa reducir la dimensionalidad de los datos en un 50% o más frente a una pérdida pequeña de información relevante. Los resultados obtenidos tras la reducción obtenida por el Perceptrón Multicapa para este dominio en concreto son peores que los logrados sobre otros dominios, los coeficientes de determinación alcanzados por el Perceptrón tanto en la fase de entrenamiento como en la fase de test son los más bajos en comparación con el resto de dominios; por lo tanto la red de neuronas no se ha ajustado adecuadamente a los patrones, por lo que era de esperar que los resultados alcanzados por el modelo KNN sobre estos datos fueran inferiores.

Para analizar otro de los objetivos del trabajo, se realiza una comparación entre el mejor de los resultados obtenidos por el modelo KNN sobre las reducciones obtenidas por el Perceptrón (adquiriendo las activaciones de las neuronas ocultas u obteniendo las entradas netas a estas neuronas) y los resultados obtenidos por el modelo KNN sobre los componentes principales obtenidos por el modelo PCA. Para ello se realiza la diferencia de estos resultados, mostrados en la siguiente tabla.

Nº vecinos / Nº atributos	1	3	5	7
8	-9,18090763%	-8,17986132%	-5,86279367%	-5,36634561%
7	16,005099%	9,97724934%	7,73192408%	6,11929067%
6	0	-2,83764377%	-2,51467268%	-1,52647917%
5	16,6893944%	16,0007441%	13,6665769%	12,9556762%
3	0	28,3895959%	27,1417074%	25,4421632%
2	0	22,4162003%	22,129287%	20,9329707%
1	0	12,5046865%	21,0976258%	22,7373708%

Tabla 10: Porcentaje de la diferencia entre RNA+KNN y PCA+KNN (House_16H)

Se aprecia como el porcentaje de mejora de reducir los datos mediante PCA es muy bajo frente a reducir los datos mediante RNA, ya que el mejor resultado es de 9,18%; mientras que la mayor diferencia alcanzada con el Perceptrón Multicapa es de 28,39%. También se aprecia que, al realizar una reducción de los atributos de entrada más agresiva, funcionan mejor las RNA que el modelo PCA para reducir los datos.

Se concluye que los mejores resultados, son los alcanzados por el modelo KNN sobre las activaciones de la capa oculta del Perceptrón. Por lo tanto, aunque este no sea el dominio en el que la reducción mediante RNA se comporte mejor, sigue siendo cierto que reduce la dimensión de los atributos de entrada mejor que el modelo PCA.

4.4.3. Análisis de los resultados de Elevators

En esta sección de realiza el análisis de los resultados alcanzados con el dominio "Elevators", las tablas que contienen los resultados se encuentran en el apéndice, en el apartado *3.Resultados obtenidos por el conjunto de datos Elevators*.

En primer lugar, se hace un análisis mediante la comparación entre reducir los datos mediante RNA, que emplea aprendizaje supervisado, calculando la función sigmoidal frente a los resultados obtenidos sobre los datos originales. Esta diferencia se presenta en la siguiente tabla.

Nº vecinos / Nº neuronas	1	3	5	7
9	0,20424045	0,1269394	0,11799806	0,10719653
7	0,25462992	0,14950998	0,14092183	0,13559875
6	0,32401201	0,20214565	0,18425978	0,17707988
5	0,40166409	0,20937549	0,1938282	0,18446451
4	0,3385301	0,21028876	0,19586309	0,18783799
2	0,30302355	0,18126092	0,16310006	0,15019698
1	0,13997671	0,10949551	0,11854659	0,11931191

Tabla 11: Diferencia RNA con función sigmoide + KNN y KNN sobre datos originales (Elevators)

Se aprecia como en todos los casos, independientemente del número de neuronas ocultas y del valor otorgado al parámetro K, los resultados mejoran hasta en un 40% a los alcanzados con los datos originales. Por lo tanto, en este dominio en particular, reducir la dimensionalidad de los atributos de entrada con esta técnica es muy favorable, ya que se reduce de forma considerable la dimensión del conjunto de datos y se mejoran los resultados obtenidos tras aplicar el modelo KNN sobre los datos originales. Se puede determinar que las activaciones de las neuronas de la capa oculta realizan una transformación sobre los datos originales capaces de concentrar la información relevante en un menor número de atributos.

La siguiente tabla muestra la diferencia de los resultados alcanzados por el modelo KNN sobre los datos reducidos, obtenidos por el Perceptrón sin calcular la función sigmoidal y sobre los datos originales.

Nº vecinos / Nº neuronas	1	3	5	7
9	0,22649508	0,13226162	0,12747334	0,12285291
7	0,2501983	0,15602208	0,14429029	0,13820205
6	0,3129598	0,19367945	0,17765635	0,16810475
5	0,3218123	0,20863033	0,19146667	0,18214918
4	0,33499983	0,20956744	0,19400289	0,18243583
2	0,30117794	0,18044304	0,16417266	0,15248363
1	0,13997671	0,10949551	0,11854659	0,11931038

Tabla 12: Diferencia RNA sin función sigmoide + KNN y KNN sobre datos originales (Elevators)

En este caso, como en el anterior, independientemente del porcentaje de reducción aplicado a los atributos de entrada y del valor determinado para el parámetro K, los resultados alcanzados mejoran los resultados sobre los datos originales hasta en un 33,50%. Por lo que se puede determinar que las entradas netas a las neuronas ocultas de la red, agrupan la información relevante de los datos originales tras realizar la transformación. Una observación interesante es que la diferencia entre realizar la reducción con esta técnica y no reducir los datos, aumenta según va disminuyendo el número de atributos de entrada desde 9 hasta 4; es curioso debido a que, cuanto menor es el número de atributos, más concentra la información relevante.

Una vez finalizadas las comparaciones entre el comportamiento de las RNA frente a los datos originales; se realiza una comparación entre los resultados alcanzados por el modelo KNN sobre los componentes principales adquiridos mediante PCA, menos los resultados sobre los datos de partida.

Nº vecinos / Nº componentes	1	3	5	7
9	-0,3022617	-0,21891472	-0,17963717	-0,15860494
8	-0,34458198	-0,24270021	-0,18438811	-0,16345525
6	-0,47409748	-0,47302481	-0,34424634	-0,30245725
5	-0,47409748	-0,46632553	-0,37281296	-0,32951336
4	-0,47409748	-0,44612992	-0,37355893	-0,34346297
2	-0,47409748	-0,52053499	-0,42830018	-0,38326623
1	-0,47409748	-0,55334227	-0,45461043	-0,41327186

Tabla 13: Diferencia PCA+ KNN y KNN sobre datos originales (Elevators)

En ninguno de los casos, el resultado alcanzado mejora los resultados logrados sobre los datos originales; únicamente se obtienen mejores resultados dependiendo del número de componentes escogidos y del valor estipulado al parámetro K. Los resultados empeoran de forma considerable frente a los resultados sobre los datos originales, no bajando del 30% y llegando a superar el 50%.

Por lo tanto, se puede concertar que en el caso de realizar la reducción mediante RNA, es decir, con una técnica de aprendizaje supervisado en la que se tiene en cuenta los atributos de salida para realizar la transformación de los datos; se obtienen mejores resultados aplicando el modelo KNN que sobre los datos originales, habiendo reducido la dimensión de los datos de entrada de forma notoria, del 50% en adelante. Por lo tanto, esta técnica para este dominio en concreto funciona de forma muy favorable. Por el contrario, reducir la dimensión de los atributos de entrada mediante PCA no obtiene ninguna mejora frente a los datos originales; depende del número de componentes y del valor del parámetro K, que la pérdida de información sea menor.

A continuación, se realiza una comparación del mejor de los resultados obtenidos tras reducir la dimensionalidad de los datos mediante la técnica de aprendizaje supervisado RNA y mediante PCA. De este modo, todos los valores positivos indican la mejora de reducir los datos mediante RNA frente a reducirlos con el algoritmo PCA.

Nº vecinos / Nº atributos	1	3	5	7
9	52,8756778%	35,1176346%	30,7110504%	28,1457844%
7	59,9211899%	39,8722286%	32,8678398%	30,1657297%
6	79,8109488%	67,517047%	52,8506118%	47,9537131%
5	87,5761572%	67,5701023%	56,664116%	51,3977876%
4	81,2627578%	65,6418682%	56,9422023%	53,1300953%
2	77,712103%	70,179591%	59,2472844%	53,5749865%
1	61,4074188%	66,2837783%	57,3157024%	53,258377%

Tabla 14: Porcentaje de la diferencia entre RNA+KNN y PCA+KNN (Elevators)

Se observa como en todos los casos el porcentaje de la diferencia de los resultados de reducir la dimensionalidad mediante RNA o mediante PCA son positivos y elevados, ninguno baja del 28%; por lo tanto, en cualquier caso, es más eficaz reducir la dimensión de los datos de este dominio mediante RNA. Se deduce que las activaciones de las neuronas de la capa oculta (calculando la función sigmoidal o sin calcularla) agrupan la información relevante de los atributos de entrada originales mejor que los componentes principales, que no tienen en cuenta los atributos de salida para realizar la transformación de los mismos.

4.4.4. Análisis de los resultados del dominio artificial Elevators girado

En este apartado se realiza el análisis de los resultados obtenidos por el conjunto de datos creado de manera artificial "Elevators girado" tras realizar los experimentos pertinentes. Todos estos resultados se encuentran en las tablas del apartado *4.Resultados obtenidos por el conjunto de datos Elevators girado* del apéndice.

En primer lugar, se realiza una comparación entre los resultados del modelo KNN sobre los atributos reducidos mediante las activaciones de la capa oculta del Perceptrón calculando la función sigmoidal, menos los resultados del modelo KNN sobre los datos originales. Los resultados de esta comparación se muestran en la siguiente tabla, se recuerda que aquellos resultados que tomen valores negativos, se consideran como nulos.

Nº vecinos / Nº neuronas	1	3	5	7
13	0	0	-0,00812152	-0,0685361
11	0	0	-0,00812152	-0,0685361
9	0	0	-0,00812152	-0,0685361
7	0	0	-0,00812152	-0,03366597
5	0	0	-0,00812152	-0,0685361
4	0	0	-0,00812152	-0,0685361
2	0	0	-0,00812152	-0,0685361

Tabla 15: Diferencia RNA con función sigmoide + KNN y KNN sobre datos originales (Elevators girado)

Se observa para este dominio creado de manera artificial, que los resultados son bastante malos ya que el coeficiente de determinación alcanzado por el modelo KNN en muchos casos es negativo. De todos modos, el empeoramiento no es grande. Sin embargo, el objetivo de este dominio es comprobar si el método de transformación y selección de atributos basado en RNA es capaz de deshacer el añadido de atributos aleatorios y el giro. Con estos resultados se determina que el Perceptrón Multicapa no ha sabido abstraer los atributos agregados al dominio de manera artificial.

La siguiente tabla muestra la diferencia existente entre los resultados obtenidos por el modelo KNN sobre las activaciones de la capa oculta sin calcular la función sigmoidal, menos los resultados del modelo sobre los datos originales.

Nº vecinos / Nº neuronas	1	3	5	7
13	0	0	-0,00812152	-0,03739771
11	0	0	-0,00812152	-0,0685361
9	0	0	-0,00812152	-0,0685361
7	0	0	-0,00812152	-0,04221999
5	0	0	-0,00812152	-0,06646958
4	0	0	-0,00812152	-0,0685361
2	0	0	-0,00812152	-0,0685361

Tabla 16: Diferencia RNA sin función sigmoide + KNN y KNN sobre datos originales (Elevators girado)

Este caso es muy similar al caso anterior, no se produce ninguna mejora al reducir la dimensión de los atributos de entrada mediante las activaciones de la capa oculta del Perceptrón multicapa sin calcular la función sigmoidal. Al igual que en el caso anterior, el Perceptrón Multicapa no ha desechado la información irrelevante e innecesaria que se ha introducido en este dominio de manera artificial.

A continuación, se realiza la diferencia entre los resultados del modelo KNN sobre los componentes principales menos los resultados sobre los datos originales.

Nº vecinos / Nº componentes	1	3	5	7
13	0	0	-0,00048716	0,00101544
11	0	0	-0,00032019	0,0004752
9	0	0	-0,00259029	0,00356049
7	0	0	0,00115158	-0,00301105
5	0	0	0,00765413	-0,00294536
4	0	0	0,00537845	0,00646884
2	0	0	-0,00812152	-0,0685361

Tabla 17: Diferencia PCA+ KNN y KNN sobre datos originales (Elevators girado)

Se aprecia que dependiendo del número de componentes principales que se seleccionen y del número de vecinos a tener en cuenta, los resultados del modelo KNN sobre las reducciones obtenidas por PCA mejoran a los resultados sobre los datos originales. Las mejoras producidas no son demasiado significativas, pero la reducción de la dimensionalidad si lo es.

Aunque para este dominio, el método basado en RNA no consigue los resultados esperados, la propia red de neuronas si obtiene buenos resultados (*Tabla 25: Coeficiente de determinación redes de neuronas (Elevators girado)*; por lo que se deja para investigaciones futuras el comprender por qué la transformación basada en la capa oculta de la red no es utilizable por KNN para este dominio en particular.

Para concluir con el análisis de los resultados de este domino, se hace un estudio de la diferencia de aplicar una técnica basada en aprendizaje supervisado o no, para reducir la dimensionalidad de los atributos de entrada. Para ello se calcula el porcentaje de la diferencia entre el mejor de los resultados alcanzados por el modelo KNN sobre los datos logrados por el Perceptrón, menos los resultados alcanzados por PCA.

Nº vecinos / Nº atributos	1	3	5	7
13	0	0	-0,76343616%	-3,84131446%
11	0	0	-0,7801325%	-6,90113022%
9	0	0	-0,55312237%	-7,20965907%
7	0	0	-0,92730924%	-3,06549216%
5	0	0	-1,57756516%	-6,35242259%
4	0	0	-1,34999704%	-7,50049388%
2	0	0	0	0

Tabla 18: Porcentaje de la diferencia entre RNA+KNN y PCA+KNN (Elevators girado)

Se observa como en los casos de que se tengan en cuenta 1 o 3 vecinos los resultados de ambos modelos son igual de malos, como en el caso de que los datos se vean reducidos a 2 atributos de entrada (toma valores nulos, ya que ambos modelos logran coeficientes de determinación negativos). En el resto de los casos, obtiene mejores resultados el modelo KNN tras reducir la dimensionalidad de los atributos mediante PCA, aunque el porcentaje no supere en ningún caso el 8%.

Por lo tanto, para el dominio artificial “Elevators girado” funciona mejor el modelo PCA para reducir los datos que las RNA, aunque en ningún caso funciona demasiado bien.

4.4.5. Análisis de los resultados del dominio artificial paraboloide girada

El último análisis que se realiza es el obtenido por el conjunto de datos creado artificialmente "Paraboloide girada", los resultados obtenidos se encuentran en el apéndice, en las tablas del apartado *5.Resultados obtenidos por el conjunto de datos Paraboloide girado*.

La siguiente tabla muestra la diferencia de los resultados alcanzados por el modelo KNN sobre las activaciones de la capa oculta calculando la función sigmoidal, menos los resultados obtenidos sobre los datos originales.

Nº vecinos / Nº neuronas	1	3	5	7
7	0,66592661	0,40707311	0,36691025	0,35131416
6	0,62055387	0,36610459	0,32287493	0,30631509
5	0,70512796	0,43818542	0,39959071	0,38727859
4	0,71793819	0,44618371	0,40679713	0,39505699
3	0,71786766	0,4462018	0,40677222	0,39506847
2	-0,22203728	-0,20492121	-0,19794542	-0,18492257
1	-0,25898017	-0,21806859	-0,19912083	-0,17850565

Tabla 19: Diferencia RNA con función sigmoide + KNN y KNN sobre datos originales (Paraboloide girado)

Se aprecia que, en la mayoría de los casos, se produce una mejora considerable tras reducir la dimensión de los atributos de entrada; excepto en el caso de que los atributos se vean reducidos a 2 o a 1, en este caso obtiene mejores resultados el modelo KNN sobre los datos originales, pero la más elevada de las mejoras no alcanza el 26%, en cambio la mejora que se produce tras reducir la dimensionalidad supera en algunos casos el 71%. Se concluye que, mediante esta técnica el Perceptrón es capaz de transformar los datos de tal forma, que las activaciones agrupan la información relevante de los atributos de entrada originales.

A continuación, se muestra la diferencia entre reducir la dimensión de los atributos de entrada mediante las activaciones de la capa oculta sin calcular la función sigmoidal, frente a no reducir la dimensión de los atributos de entrada.

Nº vecinos / Nº neuronas	1	3	5	7
7	0,62927446	0,38808546	0,35097275	0,33594643
6	0,4915595	0,28832268	0,24746503	0,22942997
5	0,69225699	0,42861637	0,38923769	0,37626967
4	0,71792395	0,4461794	0,4067932	0,39505192
3	0,71787079	0,44621018	0,40676064	0,3950573
2	-0,22811811	-0,21202762	-0,19971391	-0,1775246
1	-0,25898017	-0,21759391	-0,19925243	-0,17835304

Tabla 20: Diferencia RNA sin función sigmoide + KNN y KNN sobre datos originales (Paraboloide girado)

En este caso, ocurre lo mismo que en el caso anterior, la diferencia es que la mejora que se produce al disminuir la dimensión de los atributos de entrada es inferior que, en el caso anterior, donde se calculaba la función sigmoidal. Independientemente de esto, se concluye que dependiendo del número de atributos de entrada al que se reduzcan, a excepción de 2 y 1 atributo; las RNA agrupan la información relevante de los atributos de partida en un menor número de elementos.

Por último, para analizar el objetivo de si es posible realizar una reducción de la dimensionalidad de los atributos de entrada, se realiza la comparación entre los resultados alcanzados por el modelo KNN sobre los componentes principales menos los resultados alcanzados sobre los datos originales.

Nº vecinos / Nº componentes	1	3	5	7
7	-0,27727123	-0,35883543	-0,33274142	-0,3114865
6	-0,27727123	-0,37035056	-0,32354091	-0,30394348
5	-0,27727123	-0,45403385	-0,3836039	-0,35272226
4	-0,27727123	-0,55093639	-0,5906232	-0,60242694
3	-0,27727123	-0,55093639	-0,5906232	-0,60242694
2	-0,27727123	-0,55093639	-0,5906232	-0,60242694
1	-0,27727123	-0,55093639	-0,5906232	-0,60242694

Tabla 21: Diferencia PCA+ KNN y KNN sobre datos originales (Paraboloide girado)

Se observa que en ningún caso se produce una mejora en el coeficiente de determinación, al reducir la dimensión de los atributos de entrada mediante los componentes principales obtenidos por el modelo PCA. Por lo tanto, la transformación producida por el modelo no es capaz de agrupar la información relevante de los atributos originales.

Como conclusión, se determina que, en el caso de reducir la dimensionalidad mediante RNA se produce, en la mayoría de los casos, una mejora en los resultados alcanzados por el modelo KNN sobre los datos originales. Por lo tanto, reducir la dimensión de los atributos de entrada con esta técnica está totalmente justificado.

Para analizar otro de los objetivos marcados por el trabajo, se realiza una comparación entre el mejor de los resultados alcanzados por el modelo KNN sobre las reducciones obtenidas por las activaciones de la capa oculta del Perceptrón (calculando la función sigmoidal y sin calcularla), menos los resultados logrados por el modelo sobre los componentes principales. Se muestran los resultados en la siguiente tabla, donde los valores positivos indican el porcentaje de mejora que se produce al reducir la dimensionalidad mediante una técnica de aprendizaje supervisado frente al modelo PCA, que no tiene en cuenta las salidas para realizar las transformaciones. Los resultados negativos, por el contrario, indican el porcentaje de mejora al reducir los atributos de entrada mediante PCA frente a RNA.

Nº vecinos / Nº atributos	1	3	5	7
7	94,3197846%	76,5908544%	69,9651674%	66,2800656%
6	89,7825109%	73,6455149%	64,6415846%	61,0258571%
5	98,2399191%	89,2219271%	78,3194611%	74,0000852%
4	99,5209429%	99,7120096%	99,7420336%	99,7483932%
3	99,5142021%	99,7146563%	99,7395424%	99,7495404%
2	5,52339559%	34,6015175%	39,2677779%	42,4902343%
1	1,82910638%	33,3342472%	39,1502376%	42,4073898%

Tabla 22: Porcentaje de la diferencia entre RNA+KNN y PCA+KNN (Paraboloide girado)

En todos los casos, con mayor o menor porcentaje, se produce una mejora en los resultados obtenidos por el modelo KNN sobre las reducciones alcanzadas con el Perceptrón Multicapa frente a las reducciones logradas mediante PCA. Las mejoras alcanzadas por las RNA, son en la mayoría de los casos muy elevadas, por lo que, se determina que para este dominio funciona mucho mejor la técnica de aprendizaje supervisado empleada que el modelo PCA.

4.4.6. Resumen del análisis de datos

A continuación, se muestra una tabla que contiene el resumen de los mejores resultados alcanzados para cada domino por cada experimento realizado.

Método / Dominio	KNN (Sobre datos originales)	RNA+KNN (Con función sigmoidal)	RNA+KNN (Sin función sigmoidal)	PCA+KNN
Housing	0,790767884333 (13)	0,82944339575 (3)	0,77707727826 (1)	0,70614317489 (6)
House_16H	0,47693624784 (16)	0,35116940605 (7)	0,48675548077 (5)	0,45088821573 (8)
Elevators	0,69868169261 (17)	0,88651967834 (4)	0,88111752199 (4)	0,54007675494 (9)
Elevators girado	0,06853610343 (25)	0,03487012953 (7)	0,03113839522 (13)	0,07500493877 (4)
Paraboloide girado	0,60242693882 (10)	0,99749540447 (3)	0,99748423756 (3)	0,29848345883 (6)

Tabla 23: Mejores resultados alcanzados en los experimentos

La tabla contiene los mejores coeficientes de determinación alcanzados en cada uno de los experimentos realizados por cada dominio, independientemente del valor otorgado al parámetro k; entre paréntesis se muestra el número de atributos de entrada con el que se ha logrado dicho resultado; Por lo tanto, cada fila representa cada uno de los dominios con los que se ha realizado el presente estudio y cada columna representa el resultado alcanzado por los experimentos realizados expuestos en *Ilustración 14: Esquema del sistema*, coeficiente de determinación alcanzado por el modelo KNN sobre los datos originales, sobre las activaciones de la capa oculta calculando la función sigmoidal, sobre las activaciones de la capa oculta sin cálculo de la función sigmoidal (entradas netas a las neuronas ocultas) y sobre los componentes principales.

Se observa que, para todos los casos, excepto uno, se obtienen mejores resultados realizando una reducción de la dimensionalidad mediante RNA de los atributos frente a los datos originales, por lo que se concluye que, al menos considerando los dominios estudiados en este proyecto, el método basado en RNA para reducir la dimensión de los atributos de entrada, merece ser tenido en cuenta. También se aprecia que, los resultados alcanzados por KNN sobre los componentes principales, en el caso del dominio “House_16H” produce una pequeña mejora, pero la reducción en el número de atributos de entrada es grande.

Para realizar un análisis más exhaustivo de los resultados logrados, se muestra la siguiente tabla, que presenta para cada dominio empleado el número de pruebas en las que es mejor realizar una técnica u otra (RNA o PCA) para reducir la dimensión de los datos, en el caso del Perceptrón Multicapa se comparará el mejor de los resultados obtenidos, ya sea la reducción mediante las activaciones o mediante las entradas netas de las neuronas ocultas. Para cada dominio, los atributos de entrada se han reducido en siete subconjuntos y se han empleado cuatro modelos KNN variando el parámetro k; por lo tanto, se han de comparar 28 pruebas diferentes sobre el mismo conjunto de datos.

Método / Dominio	Veces que la capa oculta mejora a PCA	Veces que PCA mejora a la capa oculta	Veces que no se mejoran los resultados
Housing	19 de 28	9 de 28	0 de 28
House_16H	17 de 28	7 de 28	4 de 28
Elevators	28 de 28	0 de 28	0 de 28
Elevators girado	0 de 28	12 de 28	16 de 28
Paraboloide girado	28 de 28	0 de 28	0 de 28

Tabla 24: Mejoras entre RNA y PCA

Como se puede apreciar, para la gran mayoría de los casos, todos excepto para el dominio "Elevators girado", las activaciones de la capa oculta (independientemente de la técnica empleada) obtienen mejores resultados que los componentes principales. Por lo tanto, se puede concluir que el objetivo de comprobar que la reducción de la dimensión de los atributos de entrada mediante técnicas de aprendizaje supervisado funciona mejor que técnicas que no tienen en cuenta las salidas para realizar las transformaciones, es positivo. Se puede decir, que las neuronas de la capa oculta del Perceptrón Multicapa agrupan mejor la información relevante que los componentes principales obtenidos por el modelo PCA.

En cuanto al Perceptrón Multicapa, de todas las pruebas realizadas y de todas las tablas mostradas, tanto en el *APÉNDICE* como en apartado *4.ESTUDIO REALIZADO*, se concluye que realizar la reducción de la dimensionalidad de los atributos de entrada mediante RNA+KNN es para casi todos los casos una opción efectiva, puesto que alcanzan, en la mayoría de ellos, un coeficiente de determinación superior al resto; y en las que no, la diferencia en casi todas las pruebas es cercana y por lo tanto merece la pena aplicar la reducción, puesto que el número de atributos de entrada se ve reducido de forma considerable.

En lo referido al PCA, según los resultados recogidos para todos los dominios, no mejora los resultados tras aplicar el modelo KNN sobre los datos originales como se puede observar en la *Tabla 23: Mejores resultados alcanzados en los experimentos*; únicamente lo mejora en el caso del dominio creado de manera artificial "Elevators girado".

Como conclusión final, tras realizar y analizar los resultados se puede afirmar que, compensa realizar una reducción de los atributos de entrada mediante RNA, ya que, en la mayoría de los casos se obtienen mejores resultados que sobre los datos originales y la dimensión se ve reducida considerablemente; y por otro lado, se deduce que para realizar una reducción de la dimensionalidad de los datos, es más recomendable trasformar y reducir los atributos de entrada empleando la capa oculta del Perceptrón Multicapa que empleando los componentes principales obtenidos por PCA.

5. PLANIFICACIÓN Y PRESUPUESTO

En este apartado se expone la planificación seguida para la realización del presente trabajo de fin de grado, así como el desglose del coste estimado para llevarlo a cabo.

5.1. Planificación

El presente trabajo de fin de grado se ha desarrollado a lo largo de los últimos 3 meses: junio, julio y agosto del 2016; el número de días invertidos es de 66 días laborales, habiendo dedicado una media de 6 horas al día. Se ha considerado como si fuera un proyecto real, en el que han entrado en juego diferentes etapas a lo largo de los 3 meses que ha llevado a cabo realizarlo. A continuación, se desglosan las fases o etapas en las que se ha dividido el proyecto y en tiempo invertido en cada fase.

5.1.1. Proposición y reconocimiento del problema

Esta es la fase que se corresponde con el planteamiento y estudio inicial del problema a resolver. Se realiza un estudio de las posibles tecnologías y herramientas a utilizar para el desarrollo, así como las mejores técnicas para resolverlo.

5.1.2. Estudio del estado del arte

En esta sección se realiza un estudio de todas las técnicas que se van a desarrollar a lo largo del trabajo.

5.1.3. Análisis

En esta fase se realiza un análisis de las técnicas y algoritmos que se van a desarrollar, así como un estudio de los mismos. Mediante reuniones con los tutores se realiza un planteamiento del problema y una toma de requisitos que se deben cumplir.

5.1.4. Diseño

Durante esta fase de diseño se organiza la información recogida en la fase de análisis, generando de este modo el esquema del sistema a desarrollar.

5.1.5. Implementación

En esta fase de desarrolla el código establecido en las fases anteriores.

5.1.6. Pruebas

Durante la fase de pruebas, se realiza una batería de pruebas para comprobar errores en la fase de implementación y así poder solventarlos.

5.1.7. Obtención de resultados

En esta fase se aplica el sistema anteriormente generado a conjuntos de datos específicos.

5.1.8. Documentación

En esta fase se introduce toda la documentación generada y obtenida en el resto de fases, y se realiza la memoria del trabajo. Se ha llevado a cabo una documentación durante todas las fases del proyecto, pero en esta fase también se desarrolla la memoria del trabajo, realizada después de haber concluido todas las fases anteriores.

Para clarificar de forma gráfica la planificación de todas las fases llevadas a cabo durante el estudio, se incluye el siguiente diagrama de Gantt.

5.1.9. Diagrama de Gantt

	Nombre	Duración	Inicio	Fin	Predecesoras
1	**TRABAJO FIN DE GRADO**	66d?	**01/06/2016**	**31/08/2016**	
2	Proposición y reconocimiento del problema	3d?	01/06/2016	03/06/2016	
3	Estudio del estado del arte	12d?	03/06/2016	20/06/2016	
4	**Análisis**	5d?	**21/06/2016**	**27/06/2016**	**2,3**
5	Planteamiento del problema	5d?	21/06/2016	27/06/2016	
6	Toma de requisitos	3d?	23/06/2016	27/06/2016	
7	**Diseño**	8d?	**28/06/2016**	**07/07/2016**	**4**
8	Obtener las activaciones de la capa oculta	4d?	28/06/2016	01/07/2016	
9	Algoritmo KNN	2d?	04/07/2016	05/07/2016	
10	Algoritmo PCA	2d?	05/07/2016	06/07/2016	
11	Aplicar KNN sobre las diferentes técnicas	1d?	07/07/2016	07/07/2016	8,9,10
12	**Implementación**	11d?	**08/07/2016**	**22/07/2016**	**3,4,7**
13	Aplicar RNA y obtener las activaciones	5d?	08/07/2016	14/07/2016	
14	Implementar KNN	2d?	15/07/2016	18/07/2016	
15	Aplicar PCA	3d?	20/07/2016	22/07/2016	
16	Pruebas	9d?	11/07/2016	21/07/2016	
17	**Obtención resultados**	4d?	**25/07/2016**	**28/07/2016**	**4,7,12,16**
18	Resultados Housing	1d?	25/07/2016	25/07/2016	
19	Resultados House_16H	2d?	26/07/2016	27/07/2016	
20	Resultados Elevators	1d?	27/07/2016	27/07/2016	
21	Resultados Elevators girado	2d?	27/07/2016	28/07/2016	
22	Resultados Paraboliode girado	1d?	28/07/2016	28/07/2016	
23	Análisis resutados	3d?	29/07/2016	02/08/2016	17
24	**Documentación**	66d?	**01/06/2016**	**31/08/2016**	
25	Documentación	66d?	01/06/2016	31/08/2016	
26	Redacción de la memoria	21d?	03/08/2016	31/08/2016	

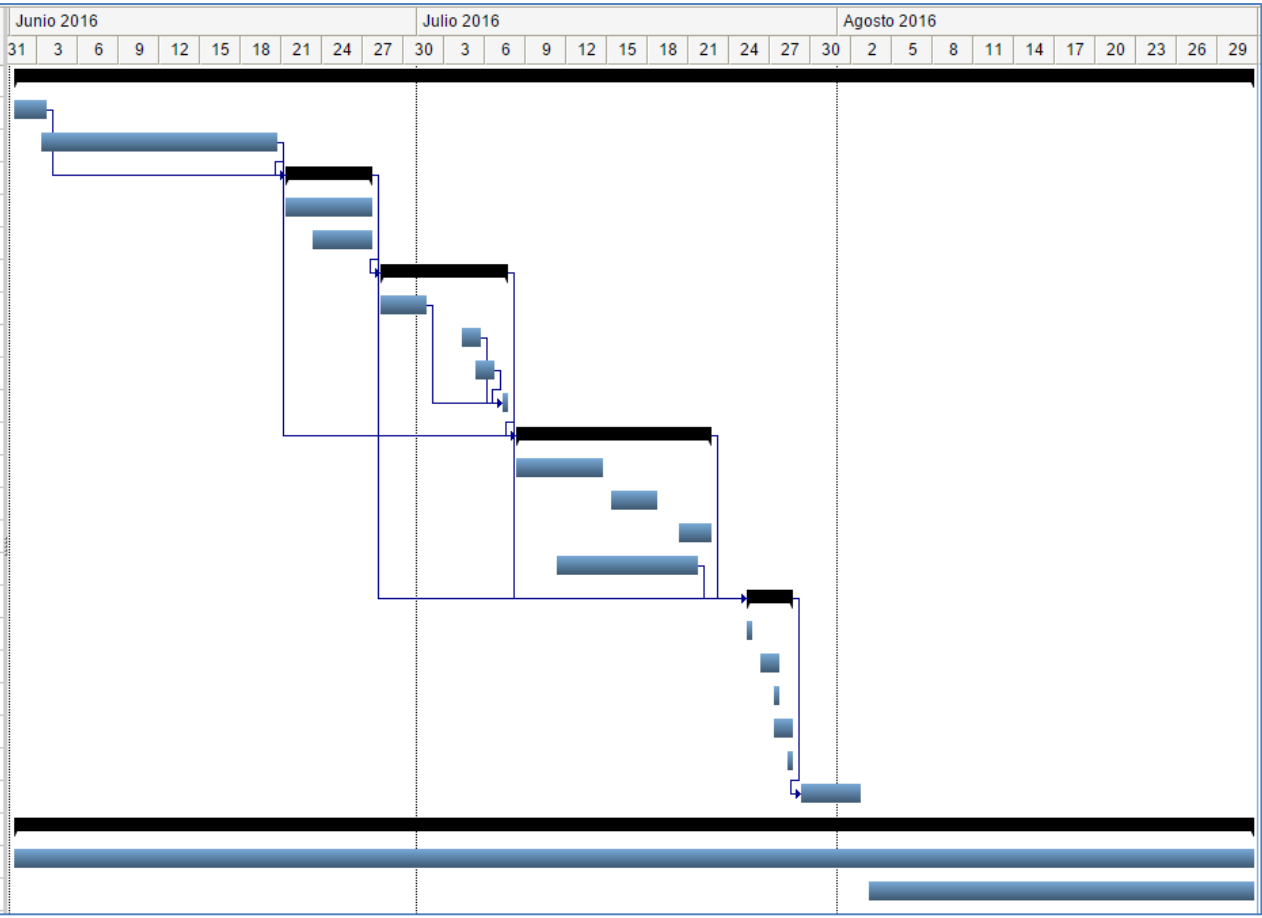


Ilustración 16: Diagrama de Gantt

Como resumen del tiempo dedicado a cada una de las fases anteriormente descritas, se presenta la siguiente tabla.

Fases	Días	Horas/Días	Horas Totales
Fase de estudio	13	6	60
Fase de análisis	7	6	42
Fase de diseño	8	6	48
Fase de Implementación	11	6	66
Fase de pruebas	9	6	54
Fase de documentación	21	6	126
		TOTAL	**396 Horas**

Tabla 25: Tiempo dedicado a cada fase

5.2. Presupuesto

En esta sección se calcula el coste que implicaría llevar a cabo el proyecto, en base a la planificación realizada en el apartado anterior. Para realizar este cálculo, se suman todos los costes producidos por el proyecto. En este caso se tienen en cuenta los costes de personal, de hardware, software y posibles costes añadidos.

5.2.1. Costes de personal

Todo el proyecto ha sido realizado por una misma persona, pero ha de tenerse en cuenta que se han adquirido diferentes roles dependiendo de la fase del proyecto desempeñada. Para las fases de análisis y diseño se debe tomar el rol de analista; para las fases de estudio, desarrollo y documentación es necesario tomar el rol de programador y finalmente para la fase de pruebas se debe adquirir un rol de tester. Al no contar con una experiencia de 2 años o superior, los tres roles adquiridos son considerados como junior.

Partiendo de que un mes cuenta con 22 días laborables y de los cuales se ha tenido una dedicación de 6 horas al día, se parte con que un mes tiene 132 horas laborables.

Para calcular el precio por hora de los servicios prestados por el personal se han seguido los siguientes puntos:

Determinar el salario anual bruto

Se denomina salario anual bruto a la suma total de dinero que percibirá un trabajador sin descontar las correspondientes retenciones y cotizaciones que se practican en cada nómina. Para calcular el sueldo anual bruto se ha realiza una media de las remuneraciones consultadas en el estudio de remuneración que despliega todos los campos tecnológicos (PagePersonnel, 2016) y se ha calculado el sueldo correspondiente a 6 horas de trabajo diarias en lugar de 8.

Se obtiene el siguiente listado de sueldos anuales:

- Sueldo Analista Junior: 25.500 €/año.
- Sueldo Programador Junior: 14.625 €/año.
- Sueldo Tester Junior: 18.000 €/año.

Estimar horas efectivas de trabajo al año

La jornada laboral en este caso es de 30 horas semanales que por 52 semanas con las que cuenta un año, suponen un total de 1560 horas de trabajo al año. A esta cantidad se le debe restar las vacaciones, festivos, asuntos personales o enfermedad; por norma general se cuentan con 21 días de vacaciones y 14 días festivos al año, que equivalen a 210 horas/año. Por lo tanto, se concluye que las horas efectivas de trabajo totales a lo largo del año son de 1350 horas/año.

Estimar horas facturables de trabajo al año

Se debe realizar una distinción entre horas de trabajo facturables y no facturables, las facturables son todas aquellas horas que generan ganancias al contrario que las no facturables, que son importantes para que el negocio funcione con fluidez. Por este motivo, en ningún caso el número de horas trabajadas es facturable al 100%, ya que se realizan tareas implícitas como consultas, búsqueda de información, llamadas telefónicas, etc. Esto obliga a contemplar un escenario de horas facturables en torno al 60% - 80% de las horas trabajadas al año. Contando con el cálculo anterior de 1350 horas/año pasan a ser facturables 945 horas/año.

Calcular el precio/horas base al año

Teniendo en cuenta los cálculos de salarios brutos estimados anteriormente para cada rol, se obtienen los siguientes precios por hora para cada uno de ellos.

- El Analista Junior cuenta con un sueldo bruto base anual de 25.500 € con 945 horas facturables/año, se obtiene un sueldo de 26,98 €/hora.
- El programador Junior cuenta con un sueldo bruto de 14.625 €/año con 945 horas facturables al año, se calcula un precio por hora de 15,47 €.
- Finalmente, para el Tester Junior con un sueldo bruto anual de 18.000 € entre las 945 horas facturables que tiene un año, se obtiene un valor de 19,05 € la hora.

El cálculo total del coste de personal se muestra en la siguiente tabla.

Personal	Horas	Precio/Hora	Coste total €
Analista Junior	90	26,98	2.428,2
Programador Junior	252	15,47	3.898,44
Tester Junior	54	19,05	1.028,7
			7.355,34 €

Tabla 26: Coste de personal

5.2.2. Costes de hardware

En este apartado se muestran los gastos producidos por la utilización del hardware necesario para el desarrollo del trabajo de fin de grado.

Como se ha especificado en el apartado *3.5.1.Hardware utilizado* el único hardware utilizado ha sido un ordenador portátil.

Para calcular el coste generado por la utilización del hardware se emplea la siguiente fórmula, donde entran en juego el número de meses desde la fecha de facturación del equipo utilizado, el periodo de depreciación, el coste del equipo y el porcentaje de uso que se le dedica al proyecto.

$$Costes\ hardware = \frac{semanas\ utilizado}{periodo\ depreciación}\ x\ costes\ x\ \%uso$$

Ecuación 19: Gastos de hardware

Hardware	Uds.	Coste (sin IVA)	Dedicación (semanas)	Periodo de depreciación	% de uso	Coste imputable
Msi GP72 6QF-492XES	1	954,14	13	52 semanas	90	214,68
				Coste (sin IVA)		214,68 €
				Coste total (con IVA)		**259,76 €**

Tabla 27: Costes de Hardware

El coste total de hardware obtenido es de 214,68 € sin IVA, por lo tanto, aplicando el 21% de IVA correspondiente, el coste total de hardware es de 259,76 €.

5.2.3. Costes de software

Seguidamente se muestran los gastos producidos por el uso de software necesario para la realización de este trabajo.

Mediante la plataforma MSDNAA y los equipos que se tienen a disposición de los alumnos de la Universidad Carlos III de Madrid, ninguno de los softwares necesarios para la realización del trabajo (*3.5.2.Software utilizado*) ha tenido que ser adquirido y por lo tanto en este proyecto tiene un coste 0. Pero al estar tratando el trabajo como un proyecto real se va a proceder a realizar los costes de software necesario para desarrollar el estudio.

Software	Uds.	Coste (sin IVA)	Coste Total
Windows 10 Professional	1	230,58	230,58
Microsoft Office 2016	1	81,82	81,82
Matlab R2015a (Academic Use)	1	413,22	413,22
		Coste (sin IVA)	725,62 €
		Coste total (con IVA)	**878 €**

Tabla 28: Coste de software

El coste total de software obtenido es de 725,62 € sin IVA, por lo tanto, aplicando el 21% de IVA correspondiente, el coste total de hardware es de 878 €.

5.2.4. Costes añadidos

Además de los costes calculados anteriormente, hay que incluir otro tipo de gastos presentes durante el proyecto, como pueden ser: costes de material fungible, márgenes de error y beneficio.

Los gastos de material fungible se presentan en la siguiente tabla.

Material fungible	Uds.	Coste (sin IVA)	Coste total
Cuaderno	1	3,23	3,23
Bolígrafo	2	0,8058	1,61
Lapicero	1	0,36	0,36
		Coste (sin IVA)	5,2 €
		Coste total (con IVA)	**6,29 €**

Tabla 29: Coste material fungible

El coste total de material fungible obtenido es de 5,2 € sin IVA, por lo tanto, aplicando el 21% de IVA correspondiente, el coste total de hardware es de 6,29 €.

5.2.5. Costes totales

Para concluir, se muestra un resumen con todos los costes obtenidos, a los que se ha añadido un 15% de costes indirectos, que representan los costes de luz, conexión a internet, alquiler de oficina, etc. Por lo que el coste total del trabajo es el siguiente:

CONCEPTO	COSTE €
Costes directos	**8.500,39**
Costes de personal	7.355,34
Costes de hardware	259,76
Costes de Software	878
Costes de material fungible	6,29
Costes indirectos	**1.275,06**
Costes indirectos (15%)	1.275,06
TOTAL PRESUPUESTO (con IVA)	**9.775,45**

Tabla 30: Total presupuesto

6. MARCO REGULADOR

Para la realización del presente trabajo de fin de grado, se ha tenido en cuenta el ámbito legal que a éste pudiera afectar.

En primer lugar, el software Matlab ha sido utilizado bajo la licencia que la UC3M posee para dicho programa, la cual permite el uso de este software de manera concurrente en cualquiera de los equipos presentes en la universidad, tanto en las aulas como en los laboratorios; se permite el uso de este software para la realización de trabajos del curso y realización de investigaciones académicas en instituciones que otorgan títulos, por el contrario no está disponible para uso comercial, gubernamental u otros usos para diferentes organizaciones.

Las bases de datos empleadas para la realización de los experimentos son propiedad de la universidad de Bilkent. Esta página permite la descarga y manejo de todas las bases de datos que se encuentran en el repositorio para análisis experimentales de aproximación de funciones, análisis del comportamiento de modelos de aprendizaje y usos estadísticos.

7. CONCLUSIONES Y FUTUROS TRABAJOS

En este apartado se presentan las conclusiones obtenidas tras finalizar el presente trabajo de fin de grado.

La finalidad del trabajo es realizar un estudio de si es viable reducir la dimensión de los datos en problemas de regresión sin que se produzca pérdida de información relevante y comparar si funciona mejor realizar la reducción con una técnica de aprendizaje supervisado, como son las RNA, frente al método PCA. Para ello, se ha implementado un programa capaz de obtener los resultados necesarios para llevar a cabo dicho estudio.

El programa realizado para obtener los resultados necesarios para el estudio efectuado en el proyecto, ha sido realizado de manera correcta y cumple con los objetivos estipulados:

- Reducir la dimensión de los atributos de entrada realizando una transformación de los mismos mediante las neuronas de la capa oculta del Perceptrón Multicapa, obteniendo las activaciones de estas neuronas de dos maneras diferentes, calculando la función sigmoidal y sin calcular esta función (entradas netas a las neuronas ocultas). Adquiriendo de este modo, dos subconjuntos de datos reducidos.
- Transformar y reducir la dimensionalidad de los datos mediante el método PCA.
- Aplicar el algoritmo KNN sobre los datos originales, sobre las transformaciones y reducciones realizadas por el Perceptrón Multicapa y sobre el conjunto de datos obtenidos por el modelo PCA tras realizar la transformación y reducción de los datos originales.

Por lo tanto, el programa cumple de manera satisfactoria con todos los objetivos propuestos para el estudio.

Otro de los objetivos fijados en el trabajo es realizar un estudio de si es posible reducir la dimensión de los atributos de entrada sin perder información relevante y obtener mejores resultados que con los propios datos originales. Este objetivo, en la mayoría de los experimentos se ha cumplido (dependiendo del número de atributos a los que se reduzcan los atributos de entrada y del parámetro k establecido), ya que, en muchos de los casos los resultados alcanzados con la reducción realizada, superan a los resultados logrados sobre los datos originales; y en muchos de los casos donde los resultados originales obtienen mejores resultados que con las reducciones, la diferencia no es demasiado grande, por lo que habría que estudiar si compensa la pérdida de información frente a reducir la dimensión de los atributos de entrada de forma considerable.

El último objetivo fijado es realizar una comparación entre reducir la dimensión de los datos mediante aprendizaje supervisado, con RNA obteniendo las activaciones de la capa oculta o las entradas netas a las neuronas ocultas; y reducir la dimensionalidad mediante PCA, algoritmo estadístico creado para esta finalidad que se puede enmarcar como no supervisado, ya que no tiene en cuenta las salidas para realizar las transformaciones sobre los datos. En el apartado *4.ESTUDIO REALIZADO* se encuentran todos los resultados logrados en los estudios; gracias a estos resultados, se concluye que al realizar la reducción de los atributos de entrada mediante la capa oculta del Perceptrón Multicapa, refleja mejores resultados en la mayoría de los casos, que reducir los atributos de entrada mediante los componentes principales del método PCA. Observando los dominios empleados, se aprecia que únicamente en el caso del dominio creado de forma artificial “Elevators girado” PCA+KNN obtiene mejores resultados que las RNA+KNN, pero en este dominio los resultados alcanzados son bastante malos en cualquiera de los dos casos. Por otro lado, en los dominios “Elevators” y “Paraboloide girado” los resultados que se obtiene mediante RNA+KNN superan de forma considerable a los logrados mediante PCA+KNN independientemente del número de vecinos a tener en cuenta y del número de atributos a los que se ven reducidos los atributos de entrada; y

para los dominios “Housing” y “House_16H” esto no se da, hay casos en los que PCA+KNN funciona mejor que RNA+KNN, pero las diferencias alcanzadas no suelen ser demasiado elevadas.

Por todo esto, con la realización de este trabajo se ha logrado alcanzar, de forma correcta, todos los objetivos fijados inicialmente; además se ha descubierto una técnica que se puede emplear para reducir la dimensión de los atributos de entrada: emplear la capa oculta de una red de neuronas artificiales. Se ha demostrado que, en muchos casos, esta técnica funciona mejor que una de las técnicas más utilizadas en la actualidad para reducir la dimensionalidad, que es PCA.

7.1. Conclusiones personales

A nivel personal, la realización de este trabajo ha sido satisfactorio, ya que, se han aplicado de forma más exhaustivas técnicas estudiadas durante la carrera.

Se ha profundizado en un tema, como es la inteligencia artificial, que a nivel personal resulta muy interesante y del que pienso que queda mucho por recorrer todavía.

Por otro lado, se ha llevado a cabo un trabajo de investigación, donde planteas algo innovador sobre lo que no se ha estudiado antes; desarrollarlo y obtener buenos resultados es algo emocionante y satisfactorio para el autor.

7.2. Futuros trabajos

Al tratarse de un trabajo de investigación, las ampliaciones que se pueden realizar sobre el mismo y los futuros trabajos que partan de él, son muy extensos.

Se puede ampliar el estudio realizando los mismos experimentos sobre nuevos dominios, ya sean originales o creados de manera artificial, para poder continuar observando los resultados que alcanzan. O incluso profundizar aún más y realizar un estudio de cuando funciona mejor una técnica que otra (RNA o PCA) y a que se debe esta mejora, si es independiente del conjunto de datos o no.

Resultaría interesante realizar un estudio del caso particular del dominio “Elevators girado”, donde la red obtiene muy buenos resultados, pero la capa oculta no es utilizable por KNN. Investigar a qué se deben los resultados alcanzados para este dominio creado de manera artificial.

Por otro lado, se puede realizar el mismo estudio, desempeñando diferentes técnicas; ya sea comparar el comportamiento de las RNA con otras técnicas que no sean el PCA o incluso probar otra técnica basada en aprendizaje supervisado para reducir la dimensionalidad de los datos. Otro futuro trabajo, podría ser comparar la transformación y la reducción de los atributos de entrada con otra técnica que no sea el algoritmo KNN.

8. BIBLIOGRAFÍA

Bedoya Puerta, J. A. (2011). *Aplicación de distancias entre términos para datos planos y jerárquicos.* Obtenido de http://users.dsic.upv.es/~flip/papers/TFM-JorgeBedoya.pdf

Borrajo Millán, D., González Boticario, J., & Isasi Viñuela, P. (2006). *Aprendizaje Automático.* Madrid: SANZ Y TORRES, S.L.

García Centeno, M. (s.f.). *COEFICIENTE DE DETERMINACIÓN*. Obtenido de http://www.expansion.com/diccionario-economico/coeficiente-de-determinacion.html

Gestal Pose, M. (s.f.). *Introducción a las Redes de Neuronas Artificiales.* Obtenido de http://sabia.tic.udc.es/mgestal/cv/RNAtutorial/TutorialRNA.pdf

González García, N., & Taborda Londoño, A. (2015). *ANÁLISIS DE COMPONENTES SPARSE Formulación, algoritmos e implicaciones en.* Obtenido de http://gredos.usal.es/jspui/bitstream/10366/126046/1/TFM_Gonz%C3%A1lezyTaborda_Analisisdecomponentes.pdf

Gorgas, J., & Cardiel, N. (2010/2011). *Análisis de componentes principales (PCA).* Obtenido de http://pendientedemigracion.ucm.es/info/Astrof/POPIA/asignaturas/ana_dat_est/tema09_x2.pdf

Hilera González, J., & Martínez Hernando, V. (1995). *REDES DE NEURONAS ARTIFICIALES. FUNDAMENTOS, MODELOS Y APLICACIONES.* Madrid: ra-ma.

Isasi Viñuela, P., & Galván León, I. (2004). *Redes de Neuronas Artificiales. Un Enfoque Práctico.* Madrid: PEARSON EDUCATION, S.A.

ITAM. (1987). Obtenido de http://biblioteca.itam.mx/estudios/estudio/estudio10/sec_13.html

PagePersonnel. (2016). *Estudio de remuneración 2016*. Obtenido de http://www.pagepersonnel.es/sites/pagepersonnel.es/files/er_tecnologia16.pdf

Russell, S., & Norvig, P. (2004). *INTELIGENCIA ARTIFICIAL. UN ENFOQUE MODERNO.* MADRID: PEARSON EDUCATION. S.A.

Sancho Caparrini, F. (2015). *Introducción al Aprendizaje Automático*. Obtenido de http://www.cs.us.es/~fsancho/?e=75

Sierra Araujo, B. (2006). *Aprendizaje Automático: Conceptos básicos y avanzados.* Madrid: PEARSON EDUCATION, S.A.

Sociedad Andaluza de Educación Matemática THALES. (s.f.). *Fundamentos de las redes neuronales*. Obtenido de http://thales.cica.es/rd/Recursos/rd98/TecInfo/07/capitulo2.html

Vicente Villardón, J. (2013). *Análisis de Componentes Principales (ACP).* Obtenido de https://www.youtube.com/watch?v=Dru4gDLFRyI

Vila , A., Sedano, M., López , A., & Juan, Á. (s.f.). *Análisis de regresión y correlación lineal.* Obtenido de http://www.uoc.edu/in3/emath/docs/RegresionLineal.pdf

9. GLOSARIO DE ACRÓNIMOS

IA → Inteligencia Artificial

KNN → k-nearest neighbours

AA → aprendizaje automático

RNA → Redes de Neuronas Artificiales.

PCA → Principal Component Analysis

RAE → Real Academia Española

OLAM → Asociador Óptimo de Memoria Lineal

MLP → Perceptrón Multicapa

APÉNDICE

Contenido

Índice de tablas

1. Resultados obtenidos por el conjunto de datos Housing

Número neuronas	Coeficiente de determinación test	Coeficiente de determinación Entrena.
7	0,870620222889797	0,973685039066590
6	0,853174821328871	0,962558494281822
5	0,768785360760572	0,942075456307062
4	0,842080430950940	0,927457552341917
3	0,844674584634634	0,901700760221853
2	0,776480584537687	0,908091935973379
1	0,818295385449651	0,802295123623161

Tabla 1: Coeficiente de determinación redes de neuronas (Housing)

Número vecinos	Coeficiente de determinación
1	0,705629835902325
3	0,790767884333843
5	0,689313556693272
7	0,668761037276393

Tabla 2: Coeficiente de determinación KNN (Housing)

Nº vecinos / Nº neuronas	1	3	5	7
7	0,454115660933390	0,535841072719170	0,591240940769762	0,589638986773615
6	-0,444338044826763	0,073217654526962	0,083118491928369	0,089482450030766
5	0,416078658994914	0,578023965831891	0,633840291972823	0,612677012308170
4	0,0857427916108853	0,495659437814505	0,484559461125906	0,475815617594245
3	0,722744770263239	0,829443395751160	0,820457674577642	0,807405138657512
2	0,491970887987328	0,740487378943769	0,763453287236826	0,776627497887605
1	0,574545103194387	0,751976477221214	0,763954767721275	0,780130005884642
Resultados KNN (datos originales)	0,705629835902325	0,790767884333843	0,689313556693272	0,668761037276393

Tabla 3: Coeficiente de determinación de aplicar KNN sobre las activaciones con función sigmoidal (Housing)

Nº vecinos / Nº neuronas	1	3	5	7
7	0,239803727493870	0,365508718125097	0,433098415489959	0,439390692828230
6	-0,058102857103011	0,323471973070228	0,311249381196514	0,269227282549991
5	0,571904294795768	0,648901736260843	0,597859204498376	0,570967461804288
4	0,696550448274230	0,636946368129691	0,691927933538217	0,691634292035159
3	0,756873219761556	0,767233180903707	0,710501218475345	0,695408196939827
2	0,451335206047474	0,615321298905742	0,580421911955199	0,494709788383098
1	0,574545103194387	0,751976477221214	0,763677034258243	0,777077278259596
Resultados KNN (datos originales)	0,705629835902325	0,790767884333843	0,689313556693272	0,668761037276393

Tabla 4: Coeficiente de determinación de aplicar KNN sobre las activaciones sin función sigmoidal (Housing)

Nº neuronas \ Nº vecinos	1	3	5	7
7	0,214311933	0,170332355	0,158142525	0,150248294
6	0	-0,250254319	-0,228130889	-0,179744833
5	-0,155825636	-0,07087777	0,035981087	0,041709551
4	-0,610807657	-0,14128693	-0,207368472	-0,215818674
3	-0,034128449	0,062210215	0,109956456	0,111996942
2	0,040635682	0,12516608	0,183031375	0,28191771
1	0	0	0,000277733	0,003052728

Tabla 5: Diferencia del R2 de aplicar función sigmoidal y no aplicar la función sigmoidal(Housing)

Nº componentes \ Nº vecinos	1	3	5	7
Todos (13)	0,705629835902325	0,790767884333843	0,689313556693272	0,668761037276393
7	0,535528232625550	0,681551454149194	0,623104583120576	0,628265527465132
6	0,706143174888997	0,689434698386700	0,617931251255846	0,631570944146669
5	0,620942974717826	0,607343049893029	0,526538218797802	0,532766102877429
4	0,604271165509909	0,595503837565862	0,536242850194113	0,530828130267782
3	0,525477557957604	0,567745314134499	0,500067629782524	0,534635883902741
2	0,327523812879775	0,454873273605100	0,403726947268124	0,428648741362688
1	-0,268576452025218	0,249830390922105	0,350818660370811	0,365515709551146

Tabla 6: Coeficiente de determinación de aplicar KNN sobre los componentes principales (Housing)

Nº atributos \ Nº vecinos	1	3	5	7
7	-0,081412572	-0,145710381	-0,031863642	-0,038626541
6	-0,706143175	-0,616217044	-0,534812759	-0,542088494
5	-0,204864316	-0,029319084	0,107302073	0,079910909
4	-0,518528374	-0,0998444	-0,051683389	-0,055012513
3	0,197267212	0,261698082	0,320390045	0,272769255
2	0,164447075	0,285614105	0,35972634	0,347978757
1	0,574545103	0,502146086	0,413136107	0,414614296

Tabla 7: Diferencia entre R2 cometido por el Perceptrón con función sigmoidal y PCA (Housing)

Nº atributos \ Nº vecinos	1	3	5	7
7	-0,295724505	-0,316042736	-0,190006168	-0,188874835
6	-0,706143175	-0,365962725	-0,30668187	-0,362343662
5	-0,04903868	0,041558686	0,071320986	0,038201359
4	0,092279283	0,041442531	0,155685083	0,160806162
3	0,231395662	0,199487867	0,210433589	0,160772313
2	0,123811393	0,160448025	0,176694965	0,066061047
1	0,574545103	0,502146086	0,412858374	0,411561569

Tabla 8: Diferencia entre R2 cometido por el Perceptrón sin función sigmoidal y PCA (Housing)

2. Resultados obtenidos por el conjunto de datos House_16H

Número neuronas	Coeficiente de determinación test	Coeficiente de determinación Entrena.
8	0,515648228730294	0,579036125779326
7	0,563003392184964	0,616048854487280
6	0,535222149479037	0,535222149479037
5	0,547650293874695	0,577209622149296
3	0,431744393687003	0,507917975860303
2	0,424162634940672	0,425538006483946
1	0,352333664897138	0,345860300414299

Tabla 9: Coeficiente de determinación redes de neuronas (House_16H)

Número vecinos	Coeficiente de determinación
1	0,161011552929144
3	0,431151986159046
5	0,473154380318491
7	0,476936247842199

Tabla 10: Coeficiente de determinación KNN (House_16H)

Nº vecinos / Nº neuronas	1	3	5	7
8	-0,274347570268687	0,163247075881852	0,251526147499056	0,275494955184424
7	-0,066954190135437	0,264042123670251	0,330854282489056	0,351169406046309
6	-0,477877807575688	0,0354735950709811	0,119785682325771	0,168507870705609
5	-0,165349187956715	0,238330369739708	0,295139456136659	0,312053930228810
3	-0,308394144960169	0,106976344652163	0,207367770526230	0,245387931150828
2	-0,704555148921671	-0,076117010774933	0,020891012053989	0,0627427118093125
1	-0,380442526044813	0,125034500473495	0,210976257532070	0,245983687933226
Resultados KNN (datos originales)	0,161011552929144	0,431151986159046	0,473154380318491	0,476936247842199

Tabla 11: Coeficiente de determinación de aplicar KNN sobre las activaciones con función sigmoidal (House_16H)

Nº vecinos / Nº neuronas	1	3	5	7
8	0,0056204329642678	0,309346962347123	0,370626376046378	0,397224759630286
7	0,165623517805471	0,437544914617029	0,459193848572631	0,469545886636967
6	-0,118584948875643	0,259955283705875	0,329419831484224	0,359593704571859
5	0,166893944059981	0,425650436788056	0,468246769404755	0,486755480768089
3	-0,001643530897098	0,362611954685685	0,442841337903556	0,465132678716575
2	-0,121279100362525	0,243953286560638	0,335866539509365	0,367803924400225
1	-0,379852724688824	0,125046865292028	0,210623285155246	0,246417873531125
Resultados KNN (datos originales)	0,161011552929144	0,431151986159046	0,473154380318491	0,476936247842199

Tabla 12: Coeficiente de determinación de aplicar KNN sobre las activaciones sin función sigmoidal (House_16H)

Nº neuronas \ Nº vecinos	1	3	5	7
8	-0,005620433	-0,146099886	-0,119100229	-0,121729804
7	-0,165623518	-0,173502791	-0,128339566	-0,118376481
6	0	-0,224481689	-0,209634149	-0,191085834
5	-0,166893944	-0,187320067	-0,173107313	-0,174701551
3	0	-0,25563561	-0,235473567	-0,219744748
2	0	-0,243953287	-0,314975527	-0,305061213
1	0	-1,23648E-05	0,000352972	-0,000434186

Tabla 13: Diferencia del R2 de aplicar función sigmoidal y no aplicar la función sigmoidal (House_16H)

Nº componentes \ Nº vecinos	1	3	5	7
Todos (16)	0,161011552929144	0,431151986159046	0,473154380318491	0,476936247842199
8	0,0974295092292895	0,391145575518008	0,429254312704973	0,450888215728456
7	0,0055725278945869	0,337772421201005	0,381874607782090	0,408352979922277
6	-0,026599687149251	0,288331721358374	0,354566558296943	0,374858496234417
5	-0,063476557421472	0,265642996212258	0,331581000719710	0,357198719195790
3	-0,380542690865024	0,0787159960390012	0,171424263623067	0,210711046617272
2	-0,485026647414307	0,0197912838420030	0,114573669318413	0,158474217557516
1	-0,690080221569136	-0,159383874297275	-0,036019621342069	0,0190441652007781

Tabla 14: Coeficiente de determinación de aplicar KNN sobre los componentes principales (House_16H)

Nº atributos \ Nº vecinos	1	3	5	7
8	-0,097429509	-0,2278985	-0,177728165	-0,175393261
7	-0,005572528	-0,073730298	-0,051020325	-0,057183574
6	0	-0,252858126	-0,234780876	-0,206350626
5	0	-0,027312626	-0,036441545	-0,045144789
3	0	0,028260349	0,035943507	0,034676885
2	0	-0,019791284	-0,093682657	-0,095731506
1	0	0,1250345	0,210976258	0,226939523

Tabla 15: Diferencia entre R2 cometido por el Perceptrón con función sigmoidal y PCA (House_16H)

Nº atributos \ Nº vecinos	1	3	5	7
8	-0,091809076	-0,081798613	-0,058627937	-0,053663456
7	0,16005099	0,099772493	0,077319241	0,061192907
6	0	-0,028376438	-0,025146727	-0,015264792
5	0,166893944	0,160007441	0,136665769	0,129556762
3	0	0,283895959	0,271417074	0,254421632
2	0	0,224162003	0,22129287	0,209329707
1	0	0,125046865	0,210623285	0,227373708

Tabla 16: Diferencia entre R2 cometido por el Perceptrón sin función sigmoidal y PCA (House_16H)

3. Resultados obtenidos por el conjunto de datos Elevators

Número neuronas	Coeficiente de determinación test	Coeficiente de determinación Entrena.
9	0,907611599630006	0,940027023183544
7	0,904956437590145	0,934366424973537
6	0,905277260058626	0,935782364871933
5	0,907962788868612	0,934439961306348
4	0,907933575665052	0,933107372838163
2	0,876487787875366	0,903645040416568
1	0,841227187827691	0,839573574999962

Tabla 17: Coeficiente de determinación redes de neuronas (Elevators)

Número vecinos	Coeficiente de determinación
1	0,474097480897623
3	0,657208783711064
5	0,684858514099779
7	0,698681692615750

Tabla 18: Coeficiente de determinación KNN (Elevators)

Nº vecinos / Nº neuronas	1	3	5	7
9	0,678337928304348	0,784148181215901	0,802856570822779	0,805878222019302
7	0,728727396603460	0,806718761841474	0,825780345144357	0,834280438582972
6	0,798109487981237	0,859354438669866	0,869118292278422	0,875761571566939
5	0,875761571566939	0,866584273380481	0,878686711488520	0,883146204804413
4	0,812627578084037	0,867497540747247	0,880721604710993	0,886519678339875
2	0,777121029880980	0,838469706662030	0,847958578592344	0,848878677325947
1	0,614074187532680	0,766704297065642	0,803405103358202	0,817993601588999
Resultados KNN (datos originales)	0,474097480897623	0,657208783711064	0,684858514099779	0,698681692615750

Tabla 19: Coeficiente de determinación de aplicar KNN sobre las activaciones con función sigmoidal (Elevators)

Nº vecinos / Nº neuronas	1	3	5	7
9	0,700592561441101	0,789470407826052	0,812331851046714	0,821534599297228
7	0,724295784465089	0,813230859068354	0,829148799539355	0,836883739700325
6	0,787057283471587	0,850888235380283	0,862514868334830	0,866786445737521
5	0,795909777356596	0,865839113919725	0,876325188017062	0,880830869801043
4	0,809097310749534	0,866776226419842	0,878861400698848	0,881117521985169
2	0,775275419074571	0,837651819676168	0,849031178925321	0,851165325212039
1	0,614074187532680	0,766704297065642	0,803405103358202	0,817992068689335
Resultados KNN (datos originales)	0,474097480897623	0,657208783711064	0,684858514099779	0,698681692615750

Tabla 20: Coeficiente de determinación de aplicar KNN sobre las activaciones sin función sigmoidal (Elevators)

Nº neuronas \ Nº vecinos	1	3	5	7
8	-0,022254633	-0,005322227	-0,00947528	-0,015656377
7	0,004431612	-0,006512097	-0,003368454	-0,002603301
6	0,011052205	0,008466203	0,006603424	0,008975126
5	0,079851794	0,000745159	0,002361523	0,002315335
3	0,003530267	0,000721314	0,001860204	0,005402156
2	0,001845611	0,000817887	-0,0010726	-0,002286648
1	0	0	0	1,5329E-06

Tabla 21: Diferencia del R2 de aplicar función sigmoidal y no aplicar la función sigmoidal (Elevators)

Nº componentes \ Nº vecinos	1	3	5	7
Todos (17)	0,474097480897623	0,657208783711064	0,684858514099779	0,698681692615750
9	0,171835783271443	0,438294061367694	0,505221347377345	0,540076754936732
8	0,129515497487211	0,414508572663657	0,500470401650305	0,535226442371351
6	-0,345332275162956	0,184183969119157	0,340612174124072	0,396224440873264
5	-0,464288332116053	0,190883250386299	0,312045551916768	0,369168328853743
4	-0,371095714765765	0,211078859074953	0,311299581736518	0,355218725549112
2	-0,507434851197112	0,136673797085458	0,256558334934964	0,315415460507893
1	-0,544046619073388	0,103866514427733	0,230248079295397	0,285409831237392

Tabla 22: Coeficiente de determinación de aplicar KNN sobre los componentes principales (Elevators)

Nº atributos \ Nº vecinos	1	3	5	7
9	0,506502145	0,34585412	0,297635223	0,265801467
8	0,599211899	0,392210189	0,325309943	0,299053996
6	0,798109488	0,67517047	0,528506118	0,479537131
5	0,875761572	0,675701023	0,56664116	0,513977876
4	0,812627578	0,656418682	0,569422023	0,531300953
2	0,77712103	0,70179591	0,591400244	0,533463217
1	0,614074188	0,662837783	0,573157024	0,53258377

Tabla 23: Diferencia entre R2 cometido por el Perceptrón con función sigmoidal y PCA (Elevators)

Nº atributos \ Nº vecinos	1	3	5	7
9	0,528756778	0,351176346	0,307110504	0,281457844
8	0,594780287	0,398722286	0,328678398	0,301657297
6	0,787057283	0,666704266	0,521902694	0,470562005
5	0,795909777	0,674955864	0,564279636	0,511662541
4	0,809097311	0,655697367	0,567561819	0,525898796
2	0,775275419	0,700978023	0,592472844	0,535749865
1	0,614074188	0,662837783	0,573157024	0,532582237

Tabla 24: Diferencia entre R2 cometido por el Perceptrón sin función sigmoidal y PCA (Elevators)

4. Resultados obtenidos por el conjunto de datos Elevators girado

Número neuronas	Coeficiente de determinación test	Coeficiente de determinación Entrena.
13	0,849462480744751	0,911782244693114
11	0,871229922144145	0,910328448179184
9	0,865521397670469	0,902284924190709
7	0,786545829127442	0,776463946787079
5	0,854659293829194	0,879037705083259
4	0,873466282419514	0,896791914868480
2	0,842877416196205	0,825125011624805

Tabla 25: Coeficiente de determinación redes de neuronas (Elevators girado)

Número vecinos	Coeficiente de determinación
1	-0,866137903805311
3	-0,120490852595520
5	0,00812151704335651
7	0,0685361034308060

Tabla 26: Coeficiente de determinación KNN (Elevators girado)

Nº vecinos / Nº neuronas	1	3	5	7
13	-1,23180491790676	-0,380830477139241	-0,214292187669175	-0,119843871509236
11	-1,51429068015925	-0,484060689813392	-0,306988636150055	-0,202967091936263
9	-1,21969041360248	-0,332701524661152	-0,132411447845159	-0,054324686503503
7	-1,27969422753188	-0,276776415959119	-0,061015065211491	0,0348701295282324
5	-1,40855434755333	-0,510911466705297	-0,332668809127760	-0,226707104700320
4	-1,51715566983498	-0,538863886690202	-0,354956491480640	-0,255042749604686
2	-1,55140677329979	-0,529515414623359	-0,325713860544945	-0,223901022760258
Resultados KNN (datos originales)	-0,866137903805311	-0,120490852595520	0,0081215170433565	0,0685361034308060

Tabla 27: Coeficiente de determinación de aplicar KNN sobre las activaciones con función sigmoidal (Elevators girado)

Nº vecinos / Nº neuronas	1	3	5	7
13	-1,05229780283749	-0,203350195564294	-0,041141699568969	0,0311383952153383
11	-0,974288553367747	-0,285979433111409	-0,136614290098114	-0,056831852884445
9	-1,18016000509887	-0,309264174454476	-0,105045331275110	-0,021073033406284
7	-0,967313861026681	-0,153206331832942	-0,060634354603561	0,0263161127258236
5	-0,979932688782239	-0,280905195277422	-0,069091973330806	0,0020665208307311
4	-1,20888500687895	-0,254451439920196	-0,116458502954611	-0,038891893707816
2	-1,36072942000809	-0,430553474988667	-0,258949102766911	-0,189093476707171
Resultados KNN (datos originales)	-0,866137903805311	-0,120490852595520	0,0081215170433565	0,0685361034308060

Tabla 28: Coeficiente de determinación de aplicar KNN sobre las activaciones sin función sigmoidal (Elevators girado)

Nº neuronas \ Nº vecinos	1	3	5	7
13	0	0	0	-0,031138395
11	0	0	0	0
9	0	0	0	0
7	0	0	0	0,008554017
5	0	0	0	-0,002066521
4	0	0	0	0
2	0	0	0	0

Tabla 29: Diferencia del R2 de aplicar función sigmoidal y no aplicar la función sigmoidal (Elevators girado)

Nº componentes \ Nº vecinos	1	3	5	7
Todos (25)	-0,866137903805311	-0,120490852595520	0,0081215170433565	0,0685361034308060
13	-0,870644627977622	-0,120190404316390	0,0076343616194911	0,0695515397719284
11	-0,863337296645702	-0,119776691815863	0,0078013250211975	0,0690113022369245
9	-0,854195084720641	-0,121591304017029	0,0055312236843857	0,0720965907067488
7	-0,889004163717211	-0,111800505377485	0,0092730923844406	0,0655250510846112
5	-0,881385653873378	-0,114149247829909	0,0157756515525676	0,0655907467561369
4	-0,850632626514045	-0,105064264172266	0,0134999704393063	0,0750049387721625
2	-1,53535425128634	-0,504197878409382	-0,273183054600667	-0,191477135340003

Tabla 30: Coeficiente de determinación de aplicar KNN sobre los componentes principales (Elevators girado)

Nº atributos \ Nº vecinos	1	3	5	7
13	0	0	-0,007634362	-0,06955154
11	0	0	-0,007801325	-0,069011302
9	0	0	-0,005531224	-0,072096591
7	0	0	-0,009273092	-0,030654922
5	0	0	-0,015775652	-0,065590747
4	0	0	-0,01349997	-0,075004939
2	0	0	0	0

Tabla 31: Diferencia entre R2 cometido por el Perceptrón con función sigmoidal y PCA (Elevators girado)

Nº atributos \ Nº vecinos	1	3	5	7
13	0	0	-0,007634362	-0,038413145
11	0	0	-0,007801325	-0,069011302
9	0	0	-0,005531224	-0,072096591
7	0	0	-0,009273092	-0,039208938
5	0	0	-0,015775652	-0,063524226
4	0	0	-0,01349997	-0,075004939
2	0	0	0	0

Tabla 32: Diferencia entre R2 cometido por el Perceptrón sin función sigmoidal y PCA (Elevators girado)

5. Resultados obtenidos por el conjunto de datos Paraboloide girado

Número neuronas	Coeficiente de determinación test	Coeficiente de determinación Entrena.
7	0,999994098942415	0,999994188361680
6	0,999998637647662	0,999998685718297
5	0,999999974660205	0,999999974452503
4	0,999999924221793	0,999999923425313
3	0,999998370236796	0,999998387789332
2	0,490365668225286	0,504381609396474
1	0,167283844800218	0,188152025362696

Tabla 33: Coeficiente de determinación redes de neuronas (Paraboloide girado)

Número vecinos	Coeficiente de determinación
1	0,277271233813161
3	0,550936386890993
5	0,590623202343617
7	0,602426938818712

Tabla 34: Coeficiente de determinación KNN (Paraboloide girado)

Nº vecinos / Nº neuronas	1	3	5	7
7	0,943197845522933	0,958009498345982	0,957533455027367	0,953741095254616
6	0,897825108560632	0,917040978295777	0,913498135777919	0,908742029429829
5	0,982399191137344	0,989121808867779	0,990213911189263	0,989705533777590
4	0,995209428597426	0,997120096221245	0,997420335668333	0,997483932487376
3	0,995138890772908	0,997138186115418	0,997395424153308	0,997495404465250
2	0,0552339558678568	0,346015174625472	0,392677778685701	0,417504364769163
1	0,0182910637665402	0,332867799968627	0,391502375949183	0,423921285805858
Resultados KNN (datos originales)	0,277271233813161	0,550936386890993	0,590623202343617	0,602426938818712

Tabla 35: Coeficiente de determinación de aplicar KNN sobre las activaciones con función sigmoidal (Paraboloide girado)

Nº vecinos / Nº neuronas	1	3	5	7
7	0,906545691792729	0,939021847146922	0,941595948104758	0,938373364139348
6	0,768830733438775	0,839259062112878	0,838088236962075	0,831856904953808
5	0,969528219248026	0,979552759745427	0,979860897291976	0,978696605729835
4	0,995195188671088	0,997115789356982	0,997416401620149	0,997478855637070
3	0,995142021223461	0,997146563030453	0,997383837475700	0,997484237556987
2	0,0491531199273059	0,338908764696468	0,390909295905100	0,424902343236478
1	0,0182910637665402	0,333342472340150	0,391370775268270	0,424073898366292
Resultados KNN (datos originales)	0,277271233813161	0,550936386890993	0,590623202343617	0,602426938818712

Tabla 36: Coeficiente de determinación de aplicar KNN sobre las activaciones sin función sigmoidal (Paraboloide girado)

Nº neuronas \ Nº vecinos	1	3	5	7
7	0,036652154	0,018987651	0,015937507	0,015367731
6	0,128994375	0,077781916	0,075409899	0,076885124
5	0,012870972	0,009569049	0,010353014	0,011008928
4	1,42399E-05	4,30686E-06	3,93405E-06	5,07685E-06
3	-3,13045E-06	-8,37692E-06	1,15867E-05	1,11669E-05
2	0,006080836	0,00710641	0,001768483	-0,007397978
1	0	-0,000474672	0,000131601	-0,000152613

Tabla 37: Diferencia del R2 de aplicar función sigmoidal y no aplicar la función sigmoidal (Paraboloide girado)

Nº componentes \ Nº vecinos	1	3	5	7
Todos (10)	0,277271233813161	0,550936386890993	0,590623202343617	0,602426938818712
7	-0,231394089022654	0,192100954706391	0,257881781386786	0,290940439573688
6	-0,233183411726287	0,180585829143865	0,267082290008288	0,298483458832220
5	-0,327718639982060	0,096902538164552	0,207019300501080	0,249704681928046
4	-0,751823631289964	-0,199291774151694	-0,105161180323242	-0,046943941569718
3	-0,853246356770159	-0,286823184478759	-0,161153514489072	-0,105463869754050
2	-1,00163426672353	-0,332857585928119	-0,202539635155713	-0,136728936099729
1	-0,939302054680535	-0,301673865198381	-0,175011366187609	-0,108179705796565

Tabla 38: Coeficiente de determinación de aplicar KNN sobre los componentes principales (Paraboloide girado)

Nº atributos \ Nº vecinos	1	3	5	7
7	0,943197846	0,765908544	0,699651674	0,662800656
6	0,897825109	0,736455149	0,646415846	0,610258571
5	0,982399191	0,892219271	0,783194611	0,740000852
4	0,995209429	0,997120096	0,997420336	0,997483932
3	0,995138891	0,997138186	0,997395424	0,997495404
2	0,055233956	0,346015175	0,392677779	0,417504365
1	0,018291064	0,3328678	0,391502376	0,423921286

Tabla 39: Diferencia entre R2 cometido por el Perceptrón con función sigmoidal y PCA (Paraboloide girado)

Nº atributos \ Nº vecinos	1	3	5	7
7	0,906545692	0,746920892	0,683714167	0,647432925
6	0,768830733	0,658673233	0,571005947	0,533373446
5	0,969528219	0,882650222	0,772841597	0,728991924
4	0,995195189	0,997115789	0,997416402	0,997478856
3	0,995142021	0,997146563	0,997383837	0,997484238
2	0,04915312	0,338908765	0,390909296	0,424902343
1	0,018291064	0,333342472	0,391370775	0,424073898

Tabla 40: Diferencia entre R2 cometido por el Perceptrón sin función sigmoidal y PCA (Paraboloide girado)