Predicción del precio en el mercado de viviendas en la ciudad de Valencia mediante redes neuronales en el año 2020

Autor: Alejandro Antón Ruiz

Titulación: Doble Grado en Administración y Dirección de Empresas e Ingeniería de Tecnologías y Servicios de Telecomunicación (192) / Grado en Administración y Dirección de Empresas (158)

ERT: Facultad de Administración y Dirección de Empresas

Tutorizado por: Javier Ribal Sanchis

Curso académico: 2019-2020

Resumen

Gracias a la gran cantidad de datos sobre inmuebles recopilados y mostrados por plataformas diversas, es posible tomarlos y emplearlos para realizar modelos predictivos, los cuales resulten en conocer el valor de mercado de un inmueble dada una serie de características. En el presente trabajo se realiza un modelo para la predicción del precio de mercado de un inmueble en la ciudad de Valencia empleando Redes Neuronales Profundas. Los datos para su realización se han extraído de la web de la plataforma idealista, mediante técnicas manuales y de *Web Scraping*. La base de datos resultante se ha limpiado de datos anómalos, con el fin de no distorsionar el modelo y que, por tanto, las predicciones sean más precisas. La variable a explicar es el precio ofertado (en €) de un inmueble y, como variables explicativas, se han empleado, el barrio, el número de habitaciones, los metros cuadrados, la planta y si cuenta o no con ascensor.

Mediante el empleo del lenguaje de programación R, junto a distintas librerías, entre las que destaca h2o, ya que es la que implementa el modelaje mediante Redes Neuronales Profundas, se ha realizado un modelo de Red Neuronal Profunda con el objetivo de predecir el precio de un inmueble dadas las características antes mencionadas. Como medida del error cometido al tratar de predecir, se ha empleado el MAE (Mean Absolute Error), por la sencillez de su interpretación.

Abstract

Thanks to the vast amount of data about real estate goods obtained and displayed across several platforms, it is possible to collect and use them in order to build predictive models which give as a result the market price of a property based on certain characteristics. In the present work, the building of a model for predicting the market price of a property in Valencia city using deep neural networks is conducted. The data used for this purpose is extracted from the webpage of the *idealista* platform, using manual and *Web Scraping* techniques. The resulting database is cleaned of anomalous data, with the objective of not distorting the model and, therefore, making the prediction more accurate. The dependent variable is the price offered (in €) of a property, and the explaining variables are the neighbourhood, the number of rooms, the squared meters, the floor number and if it has or not an elevator.

Using R programming language, as well as several libraries, highlighting h2o, since it is the library which implements the Neural Networks modelling, a Deep Neural Network has been built, with the objective of predicting a property's price based in the aforementioned features. MAE (Mean Absolute Error) has been used for error measuring, because of it being easy to interpret.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ADE FACULTAD DE ADMINISTRACIÓN Y DIRECCIÓN DE EMPRESAS. UPV

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ADE
FACULTAD DE ADMINISTRACIÓN Y DIRECCIÓN DE EMPRESAS. UPV

ÍNDICE DE TABLAS

1.INTRODUCCIÓN

1.1. CONTEXTO Y OBJETO DEL TRABAJO

Es innegable la, cada vez mayor, cantidad de posibilidades que ofrece hoy en día la tecnología, y una de las áreas en las que más desarrollo reciente está habiendo es la Inteligencia Artificial, gracias, por un lado, a la investigación y mejor comprensión de sus posibilidades y maneras de uso y, por otro, al constante incremento del poder de computación, con compañías con divisiones dedicadas exclusivamente a la producción y desarrollo de hardware para ser empleado en aplicaciones de Inteligencia Artificial (NVIDIA, 2020).

La Inteligencia Artificial es una tecnología que ofrece un amplio abanico de posibilidades, siendo algunas de sus aplicaciones relativamente obvias para la mayoría de la población, como pueden ser el reconocimiento de escenas y rostros en las aplicaciones de cámara de los teléfonos móviles, en parte gracias a la gran campaña de marketing realizada por los fabricantes, o el reconocimiento de voz de los contestadores automáticos o de los propios móviles. Sin embargo, hay aplicaciones mucho menos conocidas, pero no por ello menos relevantes, como son el análisis de riesgos y tendencias en los mercados financieros, la conducción semiautónoma o autónoma de vehículos, el reescalado de imagen y vídeo a resoluciones superiores con resultados no muy distantes de los obtenidos a dichas mayores resoluciones, o la traducción entre idiomas, cada vez más refinada gracias a la, cada vez mejor, implementación de técnicas de Inteligencia Artificial.

Por otra parte, si hay algo que defina el mundo actual, no es otra cosa que la información, la cual está disponible en cantidades ingentes a través de Internet, como puede observarse en la Figura 1, de la cual se ha de tener en cuenta que, el prefijo zetta, es el penúltimo de los disponibles en el Sistema Internacional, siendo el siguiente el prefijo yotta, por lo que, en un futuro no muy lejano, es probable que, para evaluar la cantidad de datos producidos en un año, sea necesario ampliar estos prefijos.

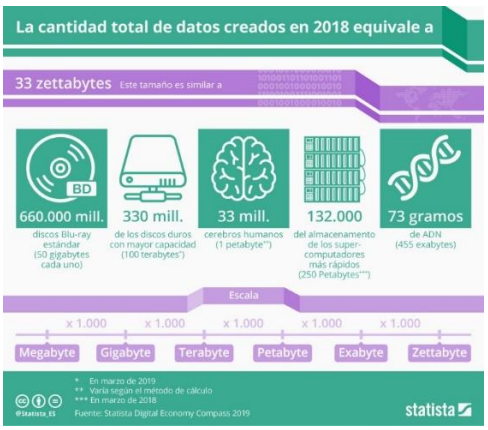


Figura 1. Cantidad de datos generada en 2018 y equivalencias. Fuente: www.es.statista.com

Así pues, dada la ingente cantidad de datos generados, existe la posibilidad de obtener valor a partir de los mismos, naciendo de esto disciplinas como el análisis de *Big Data* y la Inteligencia Artificial (IA). La primera de ellas trata de obtener, a partir de la amplia cantidad de datos disponibles, correlaciones, patrones y cualquier otra información que no es visible a primera vista (Rouse, 2012), aplicando esta información casos de uso como pueden ser el desarrollo de productos, el mantenimiento predictivo o el *Machine Learning* (Oracle, 2020). Precisamente, la última de estas aplicaciones engloba a la IA, complementándose a la perfección con el *Big Data*, obteniendo valiosos modelos con infinidad de aplicaciones, como las comentadas anteriormente.

Por otra parte, cabe contextualizar el objeto del presente trabajo, que no es otro que la predicción del precio de una vivienda en la ciudad de Valencia, lo cual implica de manera directa al mercado inmobiliario español en general y al de la ciudad de Valencia en particular. Cabe resaltar que el mercado inmobiliario, pese a ser un mercado más en cuanto a funcionamiento, tiene la peculiaridad de que el objeto con el que se comercia en él, la vivienda, es, en cierta manera, un bien esencial para las personas, aunque también cabe señalar que el mercado inmobiliario en que se encuadra el presente trabajo es el de la compraventa de vivienda, lo cual no tiene un carácter eminentemente esencial.

Respecto al mercado inmobiliario español, se puede observar (Tabla 1) que, en su conjunto, había tomado una cierta tendencia alcista respecto a 2019, habiendo en enero de 2020 una variación anual del precio por metro cuadrado de un 4,3%. Sin embargo, dicha tendencia ha sido invertida por la crisis sanitaria causada por el SARS-CoV-2, la cual ha llevado a que, en el mes en el cual se extrajeron los datos de las viviendas, junio, la variación anual del precio por metro cuadrado fuese de un -4,8%.

Mes	Precio m2	Variación mensual	Variación trimestral	Variación anual
Julio 2020	1.677 €/m2	+1,6 %	-4,3 %	-3,5 %
Junio 2020	1.650 €/m2	-4,6 %	-6,1 %	-4,8 %
Mayo 2020	1.730 €/m2	-1,3 %	-1,3 %	+1,0 %
Abril 2020	1.753 €/m2	-0,3 %	-0,3 %	+2,6 %
Marzo 2020	1.758 €/m2	+0,3 %	-0,3 %	+3,2 %
Febrero 2020	1.752 €/m2	-0,3 %	-0,5 %	+3,4 %
Enero 2020	1.758 €/m2	-0,3 %	+1,0 %	+4,3 %

Tabla 1. Histórico de variaciones de precios de venta en España. Fuente: https://www.idealista.com/

En cuanto al mercado inmobiliario de la ciudad de Valencia, como se puede observar en la Tabla 2, se ha comportado de una manera bien distinta a como lo ha hecho el mercado español en su conjunto, ya que, pese a la crisis sanitaria, la variación anual del precio del metro cuadrado se ha mantenido positiva. Este comportamiento puede ilustrar, a falta de un análisis más exhaustivo, que el aumento de la demanda de vivienda en una ciudad de considerable tamaño como es Valencia, supera al aumento de oferta hasta el punto de que ni la crisis sanitaria es capaz de revertir la tendencia alcista del mercado, aunque si la ha contrarrestado en cierto modo, viéndose variaciones mensuales negativas en marzo, junio y julio. Dicha exigua disminución de los precios por metro cuadrado contrasta con el fuerte descenso que han sufrido las transacciones de compraventa de viviendas, llegando a un -42,2% en la Comunidad Valenciana (El País, 2020).

Mes	Precio m2	Variación mensual	Variación trimestral	Variación anual
Julio 2020	1.807 €/m2	-1,9 %	+0,1 %	+3,3 %
Junio 2020	1.842 €/m2	-0,6 %	+2,6 %	+5,3 %
Mayo 2020	1.854 €/m2	+2,7 %	+2,9 %	+6,4 %
Abril 2020	1.805 €/m2	+0,6 %	+0,8 %	+3,5 %
Marzo 2020	1.795 €/m2	-0,3 %	+0,9 %	+3,4 %
Febrero 2020	1.801 €/m2	+0,6 %	+1,7 %	+2,6 %
Enero 2020	1.791 €/m2	+0,7 %	+1,1 %	+3,5 %

Tabla 2. Histórico de variaciones de precios de venta en la ciudad de Valencia. Fuente: https://www.idealista.com/

Así pues, este trabajo queda encuadrado en el uso de, en cierta medida, *Big Data*, accesible al público a través de Internet, en combinación con técnicas de IA, en concreto Redes Neuronales Profundas, que serán descritas en detalle más adelante, para la obtención de un modelo que aporte información valiosa acerca del mercado inmobiliario de la ciudad de Valencia en la forma de predicciones del valor de un inmueble, dada una serie de características del mismo.

1.2. OBJETIVOS

En cuanto a los objetivos del presente trabajo, existe un objetivo principal, que no es otro que la obtención de un modelo predictivo basado en Redes Neuronales Profundas que cumpla con la función de predecir, de la manera más precisa posible, o lo que es lo mismo, con el mínimo error, el precio de un inmueble ubicado en la ciudad de Valencia, dado el barrio en que se ubica (lo cual lleva implícita la información sobre la zona en la que se ubica), el número de habitaciones de que dispone, los metros cuadrados de superficie con los que cuenta, la planta en la que se encuentra y si cuenta o no con ascensor.

Para lograr el objetivo principal, sin embargo, ha de alcanzarse una serie de objetivos más concretos:

- Familiarizarse con la web de la plataforma Idealista para obtener los datos de la forma más eficaz y eficiente posible
- Aprender el manejo del lenguaje de programación R en general, y de una serie de librerías en particular, como rvest (empleada para el *Web Scraping*) como h2o (empleada para la construcción de modelos basados en Redes Neuronales)
- Conseguir y adecuar a un formato conveniente para alimentar a la red neuronal la base de datos de las viviendas de la ciudad de Valencia, así como detectar y eliminar datos anómalos y posibles errores
- Construir y entrenar un modelo de red neuronal, con el objetivo de que obtenga el mínimo error de predicción que sea posible, explorando para ello los distintos parámetros modificables en la librería h2o

Así pues, se puede observar que, principalmente, el trabajo consta de dos partes diferenciadas, que, en cierto modo, pueden ser vistas como fases, debiendo realizarse una tras otra para conseguir cumplir con el objetivo principal del trabajo. La primera de ellas sería la obtención y adecuación de la base de datos, y, en segundo lugar, estaría la obtención del modelo predictivo empleando Redes Neuronales Profundas, alimentadas por la base de datos creada en la fase anterior.

1.3. ESTRUCTURA DE LA MEMORIA

En esta sección se resume brevemente el contenido de cada una de las secciones principales del trabajo:

1. **Introducción**: en esta sección se define, en líneas generales, el contexto en que se enmarca el trabajo, a la par que se realiza una descripción de los objetivos, generales y específicos, así como de las fases de realización del mismo
2. **Marco conceptual**: en esta sección se describen aquellos conceptos relevantes y necesarios para la realización y comprensión del presente trabajo.
3. **Metodología**: en esta sección se detalla el procedimiento seguido para la realización del trabajo, paso a paso. Esta es, sin duda, una de las secciones más relevantes, ya que se va a mostrar el proceso que hay detrás de los resultados obtenidos.
4. **Resultados y discusión**: en esta sección, la más relevante del trabajo, se presentan los resultados obtenidos del proceso detallado en la sección de Metodología, los cuales representan la cristalización de dicho proceso. Además, se realizan los comentarios pertinentes acerca de los resultados, con fin valorativo y explicativo.
5. **Conclusiones y líneas futuras**: en esta sección se expresan las conclusiones a las que se ha llegado tras la discusión de los resultados obtenidos, teniendo en mente la finalidad del trabajo, así como las limitaciones existentes. Por otra parte, se describen propuestas para una futura mejora o aplicación en otras situaciones de lo realizado en este trabajo.
6. **Referencias**: aquí quedan ubicadas las fuentes de las cuales se ha obtenido información, debidamente referenciadas.
7. **Anexos**: en esta última sección se encuentra todo aquel material gráfico y documental de relevancia para la realización del trabajo, pero que no se encuentra en el resto de la memoria por no considerarlo oportuno.

2. MARCO CONCEPTUAL

2.1. FUNDAMENTOS DE LAS REDES NEURONALES PROFUNDAS

Las Redes Neuronales Profundas forman parte de un concepto que está sufriendo su auge en los últimos años, que no es otro que el de Inteligencia Artificial (IA). Así pues, es interesante definir en primer lugar este concepto más general para, a continuación, relacionarlo con el concepto más específico del *Machine Learning* (ML), dentro del cual se encuentra el de las Redes Neuronales.

2.1.1. INTELIGENCIA ARTIFICIAL (IA)

Por lo que respecta a la IA, se podría definir como "la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano" (Rouhiainen, 2018). Por tanto, se puede decir que, el objetivo último de la IA no es otro que el de intentar replicar los procesos mentales de los seres humanos, de manera que las máquinas puedan comportarse de la manera más similar posible. Además, las máquinas cuentan con una serie de ventajas como son el poder estar en funcionamiento ininterrumpidamente o la baja tasa de error que cometen, asumiendo que el error no sea de base.

Dentro de la IA, el enfoque por el que se ha optado de manera más intensa es el del *Machine Learning* (o aprendizaje automático), el cual podría describirse como la capacidad de que las máquinas aprendan, es decir, que puedan identificar patrones en los datos para realizar las correspondientes predicciones, sin estar explícitamente programadas para ello (BBVA, 2019).

A su vez, dentro del ML o aprendizaje automático, se encuentran otras tres variantes de aprendizaje, las cuales se pueden observar en la Figura 2.

Figura 2. Clases de aprendizaje automático. Fuente: https://www.raona.com/

La diferencia entre estos tipos de aprendizaje radica en el proceso y los elementos que conllevan a que los sistemas que los empleen, valga la redundancia, aprendan.

Por un lado, el *Supervised Learning* o aprendizaje supervisado, es aquel cuyos algoritmos emplean datos previamente clasificados, de manera que el sistema conoce cómo debe catalogar a estos datos aportados. Por tanto, se requiere que, de manera previa al aprendizaje del sistema, una persona haya clasificado los datos.

Por otra parte, se encuentra el *Unsupervised Learning* o aprendizaje no supervisado, el cual, a diferencia del anterior, no requiere de esa clasificación previa de los datos, de manera que se delega en el propio algoritmo la tarea de clasificar, sin la intervención de ninguna persona, los datos que se le proporcionan.

Por último, se encuentra el *Reinforcement Learning* o aprendizaje por refuerzo, basado en la retroalimentación que obtienen por parte de un usuario acerca de si sus predicciones son o no correctas, es decir, funcionan por el principio del refuerzo positivo y negativo.

Por lo que respecta a las Redes Neuronales Profundas, caen dentro del concepto de *Supervised Learning* o aprendizaje supervisado, ya que en su entrenamiento se emplean datos que han sido previamente clasificados, en forma del denominado *Deep Learning* o aprendizaje profundo, el cual puede definirse como un proceso en el cual no solamente se aprende la relación que tienen dos o más variables, sino que también se aprende el principio rector de la relación de dichas variables, así como aquello que le da sentido a dicha relación (Zhang et al., 2018).

Sin embargo, esta definición debe ser complementada por la de las Redes Neuronales en sí mismas, así como la concreción de las Redes Neuronales en Redes Neuronales Profundas.

2.1.2. REDES NEURONALES

Así pues, cabe comenzar por la definición general de las Redes Neuronales (realmente, Redes Neuronales Artificiales): sistemas de procesamiento de información constituidos por una serie de unidades de procesamiento básicas que imitan las características no fisiológicas de una neurona, organizadas de manera similar a como lo están en el cerebro humano (Izaurieta & Saavedra, 2000).

Esta organización consiste en que cada neurona (Figura 3) está conectada con una serie de neuronas previas a través de las dendritas, de las cuales recibe información, a la vez que se conecta a otro conjunto de neuronas posteriores, a las cuales envía información, pasando por el axón y a través de las terminaciones sinápticas, las cuales, junto con las dendritas de las neuronas siguientes, forman una interconexión denominada sinapsis (Figura 4).

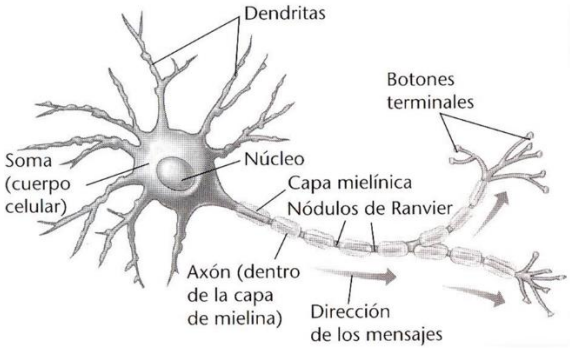


Figura 3. Esquema de una neurona. Fuente: https://sites.google.com/site/lasneuronasim/

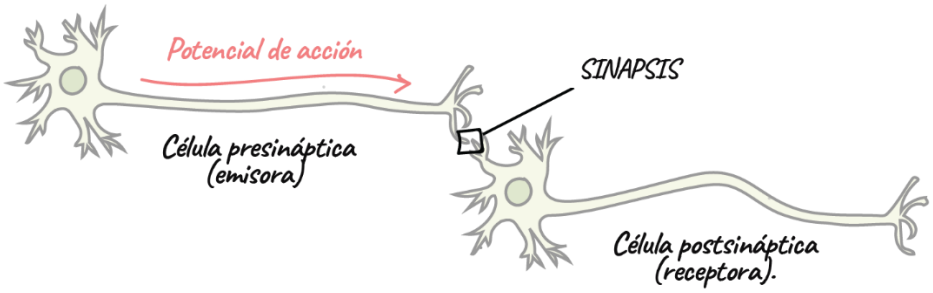


Figura 4. Esquema de interconexión de dos neuronas. Fuente: https://es.khanacademy.org/

El envío de información entre neuronas se realiza de manera que, como puede observase en las Figuras 3 y 4, una neurona recibe información de las neuronas previas o presinápticas mediante un proceso biológico denominado sinapsis química, mediado por neurotransmisores. La neurona suma, en el soma, todas las entradas de las dendritas, las cuales serán de mayor o menor magnitud en función de la cantidad de neurotransmisores que hayan sido enviados por las neuronas presinápticas. Si esta suma de las entradas es mayor que un determinado umbral, se envía un impulso eléctrico a través del axón, el cual activará la liberación de neurotransmisores en los botones terminales, los cuales serán nuevas entradas de las dendritas de las neuronas a las cuales esté conectada. Es importante reseñar que la cantidad de neurotransmisores que libera una neurona es modulada a través del proceso de aprendizaje.

Este funcionamiento de las neuronas permite a los sistemas basados en estas estructuras asemejarse, en lo que a características se refiere, a un cerebro humano, contando así con las capacidades siguientes:

- Almacenamiento de conocimiento basado en la experiencia, quedando plasmado esto en los pesos relativos de cada una de las conexiones entre neuronas
- Elevada plasticidad y capacidad de adaptación, permitiendo esto ajustarse a entornos cambiantes
- Alta tolerancia a los fallos

- Comportamiento fuertemente no lineal, lo cual los hace idóneos para el tratamiento de datos cuya fuente sea de naturaleza no lineal

Así pues, los sistemas de Redes Neuronales (Artificiales), están constituidos por un cierto número de unidades básicas de procesamiento: las neuronas, dispuestas en una serie de capas, quedando cada capa conectada con la inmediatamente anterior y con la inmediatamente posterior. Como puede observarse en la Figura 5, la primera de dichas capas recibe el nombre de capa de entrada, y es por donde se introducirá la información. Por otra parte, la última capa recibe el nombre de capa de salida, y será la que proporcione el resultado del procesado de la información introducida en la red. Por último, están las capas denominadas ocultas, que son aquellas que quedan entre la de entrada y la de salida. En función de si hay una sola de estas capas (esquema B de la Figura 5) o más (esquema A de la Figura 5), se considerará como una Red Neuronal Simple (*shallow*) o como una Red Neuronal Profunda (*deep*), respectivamente.

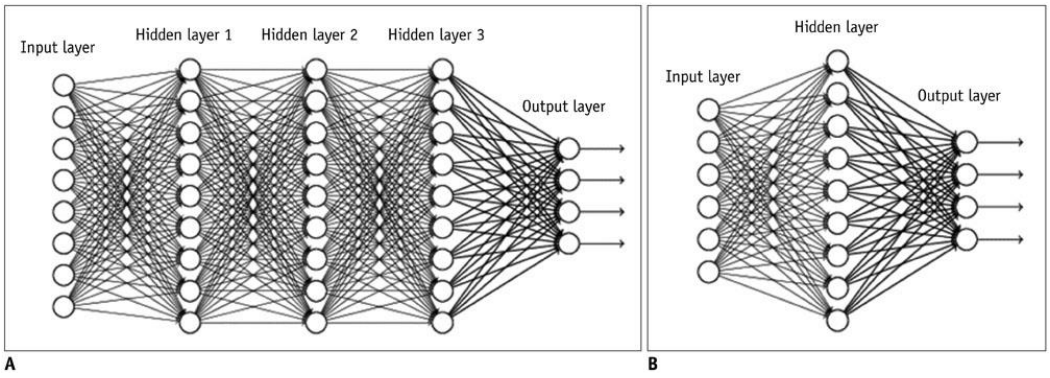


Figura 5. Esquemas de Redes Neuronales. Fuente: https://kjronline.org/

Por otra parte, para que cada una de estas neuronas artificiales se comporte de la manera más similar posible a una neurona biológica, cabe hablar de la activación de las mismas, la cual, de manera análoga a como lo hace una neurona biológica, está basada en sumar todas las entradas que le llegan y, si dicha suma supera un cierto umbral, dicha neurona se activa, o lo que es lo mismo, su salida toma un valor que será empleado por la neurona siguiente como una entrada o, en caso de ser una neurona de la capa de salida, dicha información será mostrada al exterior. Dicho valor estará relacionado con el valor que tomen las entradas, a través de lo que se conocen como funciones de activación, como las observadas en la Figura 6:

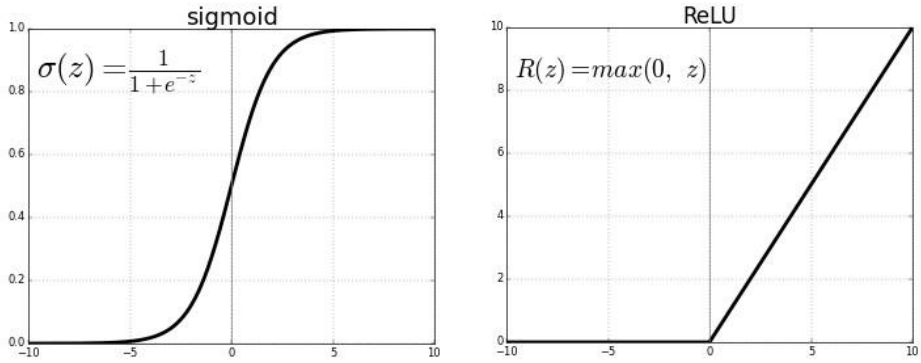


Figura 6. Funciones de activación típicas. Fuente: https://vincentblog.xyz/

Ahora bien, como se ha indicado anteriormente, el verdadero potencial de estos sistemas no es otro que el aprendizaje. Para poder describir cómo sucede dicho aprendizaje en estos sistemas, primero es necesario introducir el concepto de los pesos sinápticos, los cuales son valores por los que cada neurona multiplica cada una de sus entradas y, al haber un peso por cada una de las entradas, posibilita que se le dé una importancia relativa distinta a cada entrada, afectando pues a su propia salida y, en definitiva, a la salida global del sistema dada una serie de entradas.

En cuanto al proceso de aprendizaje en sí, está basado en una serie de algoritmos de optimización (Naranjo & Colomer, 2020), empleándose mayormente variaciones del algoritmo de descenso por gradiente, el cual, de manera resumida, consiste en un primer paso llamado *forward propagation*, en el cual, dados unos pesos iniciales, se calculan, a partir de unos datos de entrada (de los cuales se conoce de antemano qué salidas del sistema han de generar), las entradas y salidas de cada una de las neuronas (incluyendo así también la salida del sistema en sí). A continuación, se calcula el error cometido (de acuerdo con la función de pérdidas empleada) comparando las salidas del sistema con las que debían obtenerse dados los datos de entrada, y con ello se actualizan los pesos sinápticos para, a continuación, realizar un proceso denominado *back propagation*, el cual, sin entrar en más detalles, acaba con otra actualización de los citados pesos sinápticos.

A cada una de las pasadas de los datos de entrada por este proceso de *forward* y *back propagation*, se le conoce como época y, el objetivo es que, a cada época, se consiga que los pesos sinápticos se ajusten de una manera tal que el error cometido al predecir la salida a partir de los datos de entrada sea cada vez menor, lo cual es muy coherente con el concepto de aprendizaje.

Introducidos los principales conceptos de Redes Neuronales y definido el *Deep Learning* o aprendizaje profundo, así como indicado qué hace a profunda a una Red Neuronal Profunda, queda resaltar las características y posibilidades que ofrecen las Redes Neuronales Profundas. Este tipo de Redes Neuronales supera, dada una cantidad de neuronas totales (lo cual puede asemejarse a la complejidad del sistema y, por tanto,

al esfuerzo computacional), a los sistemas de tipo *shallow* en lo que a precisión de los resultados obtenidos se refiere (Mhaskar et al., 2017). Es decir, esta arquitectura goza de una mayor eficiencia computacional. Además, son capaces de aprender representaciones complejas, sin estar limitadas a ningún requisito de, por ejemplo, linealidad o continuidad de los datos (Heaton, 2017), tal y como se observa en la Figura 7.

Num Hidden Layers	Result
none	Only capable of representing linear separable functions or decisions.
1	Can approximate any function that contains a continuous mapping from one finite space to another.
2	Can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy.
>2	Additional layers can learn complex representations (sort of automatic feature engineering) for layer layers.

Figura 7. Número de capas ocultas y resultados obtenidos en Redes Neuronales. Fuente: https://www.heatonresearch.com/

2.2. ESTADO DE LA TÉCNICA

En esta sección se pretende mostrar una imagen, lo más actualizada posible, del uso de la tecnología de las Redes Neuronales, resaltando aquellas aplicaciones más relevantes en términos de valor aportado, ya que, al fin y al cabo, esta tecnología es una herramienta a través de la cual generar mayor valor, haciendo que, en los campos en los que se aplica, se llegue más allá.

Para empezar, qué mejor que algo de máxima actualidad, como es todo lo relacionado con el virus SARS-CoV-2, causante de la enfermedad COVID-19. Por ejemplo, el grupo de investigación CVBLab de la UPV, ha desarrollado un sistema que emplea Redes Neuronales como base para el reconocimiento, a través de una radiografía, si un paciente presenta o no síntomas de la enfermedad, teniendo en sus primeros resultados, una tasa de éxito del 97% a la hora de reconocer si una determinada radiografía pertenece a un paciente que sufra la citada enfermedad. (ABC, 2020).

Por su parte, una de las compañías con mayor enfoque a la aplicación de esta tecnología como es NVIDIA, tiene a disposición de los investigadores médicos una serie de aplicaciones, entre las que se encuentra Clara Parabricks (NVIDIA, 2020), cuyo objeto es la aceleración del análisis del genoma, lo cual se ha aplicado al SARS-CoV-2, además de mediante supercomputadores como los que varias empresas, como Lenovo, Intel, IBM o la propia NVIDIA, han cedido al Consorcio de Computación de Alto Rendimiento COVID-19 (Miranda, 2020), con el propio poder computación de los usuarios, de manera descentralizada con la iniciativa lanzada por NVIDIA llamada "Folding@Home" (Martí, 2020).

Sin embargo, las aplicaciones en el ámbito médico no se limitan a esta enfermedad exclusivamente, sino que hay otros ejemplos como el proyecto GALAHAD (UPV, 2017), del mismo CVBLab de la UPV, que permite, a través del diagnóstico de imagen, la detección del glaucoma. Otro ejemplo sería el trabajo de, en el cual emplean Redes Neuronales Profundas para identificar cáncer de mama metastático con un 92,5% de aciertos operando de manera autónoma, y de un 99.48% en combinación con un médico especialista en identificar este tipo de cáncer (Wang et al., 2016).

Por otra parte, desde un punto de vista más tecnológico y menos de aplicación, caben destacar las Redes Neuronales Ópticas, las cuales prometen una latencia y consumo inferiores (Fang et al., 2019), de manera que pueden considerarse una opción de futuro para la implementación de esta tecnología. Además, recientemente se han propuesto modelos de Redes Neuronales Ópticas con funciones de activación no lineales y regulables (Zuo et al., 2019).

Otra aplicación reciente de esta tecnología, de amplia visibilidad y relacionada con el tratamiento de imágenes sería el piloto automático de los vehículos de la marca

Tesla, el cual está basado en esta tecnología y se nutre de los datos recogidos por los cerca de 500.000 vehículos que tiene en circulación, para así refinar el entrenamiento recibido por estas Redes Neuronales (Eady, 2019).

Asimismo, otra aplicación de esta tecnología a una industria con un volumen de facturación de unos nada despreciables 152.100 millones de dólares en 2019 (AEVI, 2020), sería el denominado *Deep Learning Super Sampling* (DLSS) que la compañía NVIDIA ofrece en las tarjetas gráficas de la serie RTX, el cual consigue, a partir de una imagen renderizada a una determinada resolución, reescalar dicha imagen a una resolución superior, de manera tal que el resultado sea similar al obtenido si la citada imagen hubiese sido renderizada a la resolución superior, ahorrando así coste computacional a la tarjeta gráfica, lo cual le permite arrojar una mayor cantidad de imágenes por segundo para una resolución dada (Edelsten, 2019).

Por último, respecto a la aplicación concreta al caso que concierne al presente caso, como es la predicción del precio de un inmueble dada una serie de características, hay ejemplos abundantes (Varma et al., 2018) (Wang et al., 2018) (Chiarazzo et al., 2014).

2.3. *WEB SCRAPING*

Otra de las herramientas empleadas para la realización del trabajo ha sido el *Web Scraping*, también conocido como *Web Mining*, puede definirse, como el proceso consistente en la identificación y codificación automáticas de la información disponible en las páginas web (Landers et al., 2016).

Esta herramienta pertenece a la disciplina del *Big Data*, siendo uno de los primeros pasos para la extracción de valor de la información accesible a través de Internet, ya que permite recopilarla y generar una base de datos, ordenada y estructurada de manera que, tras ser debidamente procesada, sea útil a algún fin.

En cuanto a su implementación, puede realizarse a través de librerías de lenguajes de programación como R: rvest o Rselenium; y Python: Scrapy, cuya estructura puede observarse en la Figura 8, o lxml (Elite Data Science, 2019).

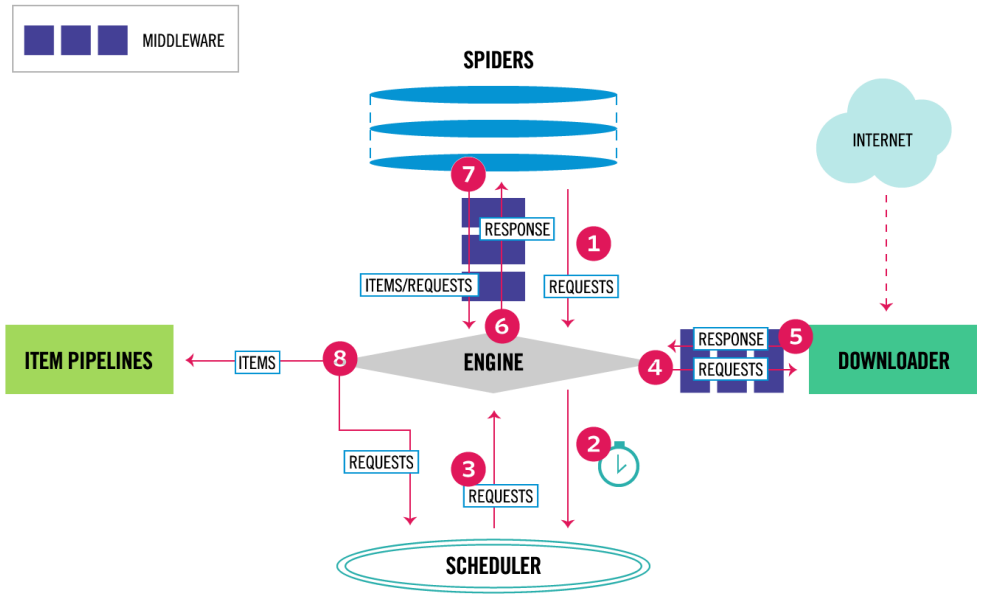


Figura 8. Esquema de *Web Scraping* con Scrapy. Fuente: https://elitedatascience.com/

En la Figura 8 pueden apreciarse los distintos elementos que conforman el proceso del *Web Scraping* empleando la librería Scrapy, aunque el esquema es similar para cualquier proceso, aunque se emplee otra librería, como rvest. En primer lugar, están los *Spiders* o *Crawlers*, los cuales se encargan de realizar las peticiones de archivos de código fuente de las páginas web, así como de, cuando reciben dichos archivos, extraer la información deseada (ITEMS en la Figura 8). Por otra parte, las peticiones de archivos se han de atender a su debido tiempo, ya que, de realizar una rápida sucesión de peticiones a una página, es muy probable que se prohíba la conexión (apareciendo así el error HTTP 403 *Forbidden*), por lo que se introduce una serie de retrasos en dichas peticiones. Dichas peticiones tienen como fin la descarga del archivo con el código fuente de la página (RESPONSE en la Figura 8).

Cabe destacar un aspecto comentado en el párrafo anterior, como es que los procedimientos de *Web Scraping* pueden ser obstaculizados por los administradores de las páginas web de las cuales se está intentando extraer información, lo cual tiene motivaciones diversas, como que se quieran evitar copias de su página web (sobre todo si el valor de dicha web son precisamente sus datos) o simplemente porque no se quiera destinar recursos de computación y red a usuarios que no sean seres humanos. De hecho, se puede comprobar si una página web no desea que sus datos sean recopilados de esta manera introduciendo la dirección web, terminada por "/robots.txt".

Estas restricciones por parte de los administradores pueden tratar de evitarse por parte de quien programa el software para el *Web Scraping* mediante una serie de técnicas, entre las que se encuentran (Yan, 2020):

- Añadir tiempos de espera entre peticiones, además de un tiempo de espera proporcional a lo que el servidor tarde en responder (*backing off*)
- Uso de identificadores de agente de usuario (*User-Agent*) idénticos a los que emplean los navegadores y dispositivos más usados. Esto se conoce como *User-Agent Spoofing*
- Uso de servidores proxy que oculten la IP de la máquina que realiza las solicitudes de descarga, siendo ideal que se vaya alternando el servidor proxy para que, de esta manera, la IP vaya cambiando cada petición o número de estas
- Empleo de librerías como polite (R) que sirven para pedir permiso al servidor para realizar las peticiones para el *Web Scraping*

3. METODOLOGÍA

3.1. OBTENCIÓN DE LA BASE DE DATOS

Para la realización de esta tarea, se recurre al proceso de *Web Scraping* de la página web de idealista. Para ello, en primer lugar, se acota qué sección de la misma se va a emplear para recopilar datos, recurriendo para ello al motor de búsqueda que poseen, seleccionando el área de la ciudad de Valencia, como se observa en la Figura 9.

Figura 9. Mapa de viviendas de la Ciudad de Valencia. Fuente: https://www.idealista.com/

Con el fin de caracterizar la ubicación de cada una de las viviendas, se accede a cada uno de los distritos en que se divide la ciudad, los cuales se aprecian en la Figura 9, para, a su vez, acceder a cada uno de los barrios (llamados zonas en la web) de que se compone cada uno de los distritos, como puede observarse en la Figura 10.

Figura 10. Barrios (zonas) de Ciutat Vella. Fuente: https://www.idealista.com/

Las urls (direcciones web) de los citados barrios conducen a un listado de las viviendas de cada barrio ordenadas en páginas, a razón de 30 viviendas por página como máximo. Quedando, en total, 17 distritos y 73 barrios, para el total de 11.522 viviendas en oferta en la ciudad de Valencia a fecha de 7 de junio de 2020, cuando se realizó la extracción de los datos para el presente trabajo.

Es en este punto en el cual se procede a emplear la herramienta elegida para la realización del *Web Scraping*. Dicha herramienta es el lenguaje de programación R, empleando, aunque no de manera exclusiva, la librería rvest.

En este punto cabe mencionar una dificultad que no ha conseguido ser superada en el presente trabajo, que no es otra que el hecho de que no se ha podido automatizar completamente el proceso del *Web Scraping*. Esto se debe a que, al tratar de ejecutar la instrucción de "html_session", mediante la cual se establece la sesión web con el servidor, la respuesta del servidor, a partir de una cierta fecha, comenzó a ser la de denegar el acceso (error HTTP 403 *Forbidden*). Se intentó aplicar los métodos descritos al final de la sección 2.3. Ninguno de ellos funcionó, y cabe reseñar un comportamiento algo extraño que se dio al aplicar las instrucciones de la librería polite, en concreto la instrucción "bow", la cual consiste en, figuradamente, hacerle una reverencia al servidor, para pedir permiso para realizar peticiones de descarga para realizar *Web Scraping*, a lo cual el servidor respondía indicando que sí permitía a nuestra máquina realizar esta práctica, para, a continuación, denegar las peticiones.

Así pues, la solución por la que se optó fue por, manualmente, descargar cada una de las páginas web de cada uno de los barrios (07/06/2020), clasificándolas con un código de numeración para suplir la codificación que se iba a realizar automáticamente de manera inicial del barrio en el que se encontraba cada una de las viviendas. Salvo por la manera de codificar la información sobre el barrio y el reemplazo del establecimiento de sesión del servidor y la correspondiente petición de descarga por la intervención manual, el resto del proceso es exactamente igual que el que se realizaría en un *Web Scraping* en condiciones, ya que, en cierto modo, lo que se ha acabado haciendo ha sido un *Web Scraping* en local.

Aclarado este punto, por lo que respecta al resto de las características de las viviendas, además del barrio en el cual se ubican, se selecciona recopilar el precio (que es la variable a explicar por el resto), el número de habitaciones, la superficie en metros cuadrados, la planta y si cuentan o no con ascensor. Se eligen estas características por ser accesibles directamente desde la página en la cual se presentan las viviendas de treinta en treinta (como máximo) (Figura 11), ya que no se considera viable, en ningún momento, el realizar 11.522 accesos web (uno por vivienda) o, en su defecto, 11.522 descargas manuales.

Para la extracción de las características mencionadas, en primer lugar, además del archivo html con el código fuente de la página (se lee con la instrucción "read_html"), es necesario conocer cómo se referencian dentro del código los

contenedores de dichas características. Para ello, se emplea la extensión del navegador web SelectorGadget, la cual funciona de manera que, como se ve en la Figura 11, al activarla, si se hace click sobre algún elemento de la página, se indica la referencia que, al aplicarse a la función "html_nodes", devuelve aquello que contiene el contenedor al que hace referencia, en este caso, el precio de las 30 viviendas que hay en la página web, lo cual se convierte en texto al emplear la instrucción "html_text".

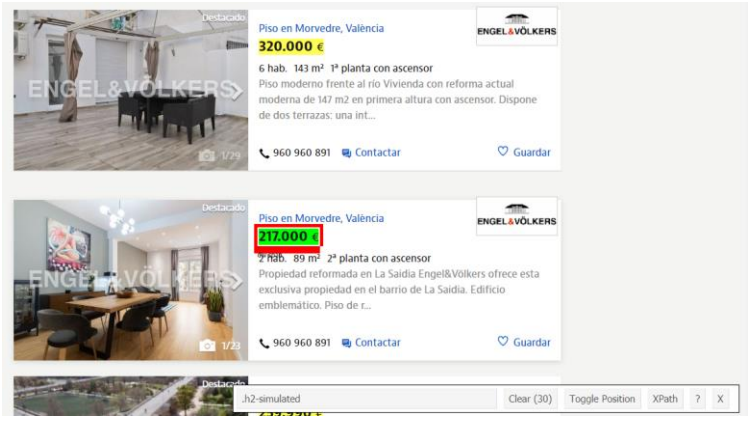


Figura 11. Empleo de la extensión SelectorGadget para conocer la referencia de los elementos deseados. Fuente: https://www.idealista.com/

Empleando las instrucciones arriba descritas, se extrae el contenido de los contenedores seleccionados para la extracción de las características, lo cual se lleva a cabo aplicando una extensa serie de instrucciones de manejo de cadenas de texto (*strings*) para eliminar espacios, símbolos indeseados y demás elementos que no sean la información deseada. Además, los contenedores en los que se encuentran el número de habitaciones, los metros cuadrados, la planta y si cuenta o no con ascensor, comparten el mismo identificador, lo cual, sumado a que no siempre están presentes los tres contenedores en cada una de las viviendas, (Figura 12), dificulta el proceso de extracción de estas características.

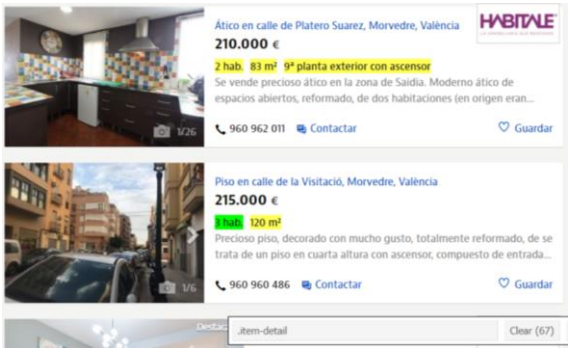


Figura 12. Identificadores de contenedores de información sobre habitaciones, metros cuadrados, planta y ascensor. Fuente: https://www.idealista.com/

Este problema se soluciona incluyendo una serie de bucles y condiciones para que detecten si cada vivienda tiene o no todos los elementos y actúen en consecuencia, rellenando la información del campo no disponible con un indicador de que no se ha especificado dicha información, así como asegurando que cada característica se almacene en el lugar que le corresponde, para que quede relacionada con la vivienda a la que pertenece.

Aplicando el proceso descrito, se finaliza obteniendo la información acerca del barrio, precio, número de habitaciones, metros cuadrados, planta y ascensor de cada una de las 11.522 viviendas, sin ninguna clase de error, debido a que se ha construido desde cero la manera de organizar y almacenar los datos, junto a que la página web los presenta de una forma consistentemente ordenada.

Para cerciorarse de que la base de datos no contiene ningún error y que contiene lo esperado, se recurre a la instrucción "makeDataReport", de la librería dataMaid, generándose así un informe de la base de datos, que puede encontrarse en la sección de anexos. Este informe contiene información relevante, como histogramas, número de valores únicos de cada variable y qué valores son estos, útil para comprobar que los datos están en el formato deseado. Además, indica si falta algún valor de alguna de las características de las viviendas. Por último, informa acerca de valores anómalos en variables de naturaleza numérica, pero esto será tratado en la sección siguiente.

3.2. DETECCIÓN Y ELIMINACIÓN DE DATOS ANÓMALOS

En primer lugar, se procede a eliminar de la base de datos todas aquellas viviendas con alguno de sus campos de datos marcado con el indicador que, anteriormente, se había empleado para señalar que la característica de una vivienda no se mostraba en la página, las cuales se elevan a 1.534 (quedando 9.988 viviendas). De esta forma se asegura que todas las viviendas de la base de datos cuentan con todos los datos por los que se ha decidido caracterizarlas.

Hecho esto, se obtiene qué viviendas tienen valores anómalos de las tres variables cuantitativas de las que se dispone, como son el precio, la superficie en metros cuadrados y el número de habitaciones. Para hacer esto, se recurre a la instrucción "boxplot", la cual devuelve, entre otras cosas, los valores anómalos o *outliers*, por lo que, a partir de estos valores, se identifica a qué viviendas pertenecen y se eliminan las 1.012 viviendas con variables con valores anómalos de la base de datos, quedando la base de datos final, con 8.972 observaciones, ya lista para realizar con ella modelos predictivos.

Para la detección de anómalos, se ha optado por el método basado en la instrucción "boxplot", el cual considera como *outliers* o anómalos aquellos valores que se distancien más de 1,5 veces el rango intercuartílico o IQ (tercer cuartil menos primer cuartil) respecto del primer cuartil o Q1 (25% de los valores) o del tercer cuartil o Q3 (75%

de los valores). De manera que serán considerados *outliers* aquellos valores que queden por debajo del Q1 menos el IQ, así como aquellos que queden por encima del Q3 más el IQ.

Sin embargo, este enfoque, el cual es de tipo univariante, al tenerse en cuenta para considerar que una observación es anómala, basándose exclusivamente en una variable a la vez, no es el único enfoque existente, ya que hay otros enfoques de naturaleza multivariante, en los cuales se toma en cuenta más de una variable a la vez, para determinar si una observación es o no anómala.

En los enfoques univariante, se considera que una observación es anómala si una de las variables sometidas al análisis está demasiado alejado de un indicador robusto de tendencia central, como puede ser la mediana o los cuartiles. Por su parte, en los enfoques multivariante, se considera que una observación es anómala si, en el conjunto de las N variables analizadas, se encuentran valores demasiado alejados de la figura de N dimensiones que conforman los indicadores robustos de tendencia central de las N variables sometidas al análisis (Leys et al., 2019).

Por lo que respecta al enfoque adoptado, ha resultado en dejar fuera de la muestra viviendas de grandes dimensiones y/o elevados precios y/o muchas habitaciones. En definitiva, viviendas que podría considerarse que pertenecen a un nicho de mercado, el cual tiene un perfil de cliente bien diferenciado, lo que repercute en el precio que está dispuesto a aceptar pagar. Así pues, su exclusión de la muestra es algo justificable y que, previsiblemente, debería mejorar los resultados obtenidos, al quedar una muestra de un segmento del mercado relativamente homogéneo. Además, la exclusión de estas viviendas porcentualmente no supone una parte de la muestra excesivamente elevada, quedándose en un 10,13% (1.012) del total de viviendas con todos sus campos especificados (9.988).

Gracias a la citada eliminación de datos anómalos, es posible conseguir predicciones más precisas, a costa, eso sí de estar sesgando artificialmente la muestra de viviendas, lo cual hace perder generalidad al modelo. Es decir, es un *trade-off* entre generalidad y rendimiento del modelo.

3.3. CONSTRUCCIÓN DEL MODELO PREDICTIVO

Obtenida la base de datos, se procede a la realización de un modelo predictivo, empleando como arquitectura las Redes Neuronales. Para su implementación, se recurre al paquete h2o para el lenguaje R. Dicho paquete es más bien una plataforma multilenguaje y de código abierto que provee las librerías necesarias para implementar algoritmos de *Machine Learning*.

En primer lugar, se ha procedido a su inicialización, ya que ha de crearse un *cluster* para que pueda funcionar. Esto se hace con la instrucción "h2o.init". En el caso del presente trabajo, los parámetros establecidos han sido un máximo de memoria RAM ("max_mem_size") de 6 GB de los 16 GB totales del equipo, así como un número de hilos ("n_threads") de 18, de los 24 totales del equipo (cuenta con un procesador de 12 núcleos físicos y 24 hilos).

Entre las diversas funcionalidades de este paquete, se ha escogido la que implementa algoritmos de *Deep Learning*, es decir, de Redes Neuronales Profundas (aunque se puede modificar para que emplee una sola capa oculta y, por tanto, implementaría una Red Neuronal de tipo *Shallow* (poco profunda). Dicha funcionalidad cae bajo la instrucción "h2o.deeplearning", y, para su manejo y comprensión, se han empleado un par de manuales (Oxdata, Inc., 2013) (Candel et al., 2018).

Ahora bien, el proceso para la obtención de un modelo basado en Redes Neuronales comienza con un paso fundamental, el cual es la segmentación de la base de datos en datos para entrenamiento, validación y test. Para ello, el paquete h2o cuenta con la instrucción "split", la cual segmenta, aleatoriamente, un conjunto de datos de acuerdo con el tanto por uno que se le indique. Así pues, como se recomienda por Naranjo y Colomer 2020, se divide la base de datos en un 80% para entrenamiento y validación, y en un 20% para test. De la parte de los datos que quedan para entrenamiento y validación, se vuelve a aplicar la instrucción "split" para dividirlos en un 80% (64% del total) para entrenamiento y un 20% (16% del total) para validación.

Aclarar que los datos de entrenamiento son aquellos a los que se aplica el algoritmo de descenso por gradiente, por el cual los pesos sinápticos van siendo modificados para disminuir el error cometido (definido por la función de pérdidas). Por su parte, tras cada época (pasada de los datos de entrenamiento por el algoritmo de descenso por gradiente), se calcula el error cometido empleando los datos de validación, almacenándose los pesos del modelo de la época en la que se hace la validación si el error cometido en los datos de validación es menor que el cometido en épocas anteriores. Por último, los datos de test se emplean, una vez ya construido el modelo, para determinar qué error se comete a la hora de tratar de predecir, en este caso, el precio de una vivienda dadas las características de la misma.

Cabe mencionar que, por lo que respecta a las variables, se ha realizado el paso previo de convertirlas a los tipos correspondientes, para su correcta interpretación por parte de la librería h2o. De esta manera, se emplea la instrucción "as.factor" para las variables que se han considerado como categóricas, a saber, el barrio, la planta y si cuenta o no con ascensor. Así, la instrucción "h2o.deeplearning", realizará, por defecto, su codificación siguiendo el esquema "OneHotInternal", el cual deja los datos como estaban, pero internamente emplea un esquema de codificación de tipo *One-Hot*, en la que, para cada variable categórica, se emplearán tantos dígitos como niveles tenga la variable. Cada nivel quedará codificado con un "1" en una posición que quedará asignada al mismo, y el resto serán todo "0".

La motivación de emplear la codificación *One-Hot* es que el algoritmo de la Red Neuronal no asuma que existe un cierto orden entre los niveles de la variable categórica si, por ejemplo, se codificase como números enteros, con un valor para cada nivel (Vasudev, 2017). Además, en la práctica se ha mostrado una mejora en los resultados obtenidos por el modelo predictivo al usar este esquema de codificación versus uno numérico.

En cuanto a los parámetros que se han ido modificando, buscando obtener un modelo lo más preciso posible, cabe destacar:

- Número de épocas ("epochs"): define el número de veces que pasan los datos de entrenamiento por el algoritmo de descenso por gradiente
- Función de activación ("activation"): afecta a la función de activación de las neuronas de las capas ocultas, la de la capa de salida no es modificable y, en este caso, es lineal. Las posibilidades incluyen "Rectifier", "Tanh", "TanhWithDropout", "Rectifier", "RectifierWithDropout", "Maxout", "MaxoutWithDropout". Las que se acompañan de "WithDropout", incluyen la funcionalidad del *Dropout*, que cosiste en que, en cada iteración del algoritmo de descenso por gradiente, se "desconecta" una cierta cantidad de neuronas de cada capa, siendo controlada esta cantidad (con tantos por uno, para cada capa) por otro parámetro: "hidden_droput_ratios"
- Función de pérdidas ("loss"): afecta a la función que se tendrá en cuenta para la medida del error en el algoritmo de descenso por gradiente. Para este problema, pueden emplearse: "Automatic", "Quadratic", "Huber", "Absolute", "Quantile"
- Distribución asumida de los datos de entrenamiento ("distribution"): van ligadas a una serie de funciones de pérdidas. Para un problema de regresión como este, solamente pueden emplearse "gaussian", "possion", "gamma", "tweedie", "quantile", "laplace"

- Parámetros de regularización, que ayudan a dar estabilidad y generalizar (evitar *overfitting*, que es que el modelo se ajuste demasiado a los datos de entrenamiento en concreto, y después funcione deficientemente con los demás datos): "l1" y "l2"
- Parámetros del método de tasa de aprendizaje adaptativa (ADADELTA) que se implementa por defecto:
    - Rho ("rho"): relacionado con la memoria que se tiene de actualizaciones de pesos realizadas anteriormente. Valores típicos entre 0.9 y 0.99
    - Épsilon ("epsilon"): determina la tasa a la que se reduce la tasa de aprendizaje. Ayuda a progresar cuando se está cerca de la solución óptima. Valores típicos entre 10^{-4} y 10^{-10}
- Tamaño del *mini-batch* ("mini_batch_size"): establece el número de filas de datos que se utilizan para realizar una actualización de pesos, la cual se hace promediando entre los pesos resultantes de la aplicación del algoritmo de descenso por gradiente con los datos de cada fila. Así pues, en una época sucederá este proceso un número de veces igual al entero superior al resultado de dividir el número de filas de datos (viviendas) de la base de datos, por el valor de este parámetro. Según se indica en Masters y Luschi 2018, el valor óptimo de este parámetro para Redes Neuronales Profundas se encuentra en 32
- Número de capas ocultas y neuronas de cada una ("hidden")
- Fracción de las características de cada fila de datos (vivienda) que no serán tomadas en cuenta al pasar los datos por el algoritmo de descenso por gradiente ("input_dropout_ratios"). Este parámetro está indicado para cuando se sufra de *overfitting*

3.4. OPTIMIZACIÓN DEL MODELO

Para esta tarea, en primer lugar, ha de definirse cómo evaluar el rendimiento del modelo, para así conocer si cualquier modificación que se le haga lo mejora o lo empeora.

De esta manera, se establece que, para evaluar el desempeño del modelo, la métrica a observar va a ser el MAE (Mean Absolute Error) o error absoluto medio, mediante la instrucción "h2o.performance", el cual se calcula como sigue (nótese que N denota el número total de viviendas y P el precio):

$$MAE = \frac{\sum_{i=1}^{N} \left| P_{predecido_i} - P_{verdadero_i} \right|}{N}$$

Se escoge esta métrica debido a que, en cierto modo, es el que mejor ilustra cuánto yerra el modelo evaluado, ya que, si el MAE es de, por ejemplo, 100.000€, significa que, de media, infravalora o sobrevalora, sin distinción, el valor de una vivienda en 100.000€. Así pues, es una métrica que ilustra de una forma directa y clara este error cometido.

Establecida la métrica para evaluar el rendimiento, el proceso seguido para la optimización del modelo ha consistido en, primero, ir modificando, de manera aislada, los parámetros que, en cierto modo, mayor influencia pueden tener en el modelo, siendo el primero de ellos la propia arquitectura, definida con el número de capas y neuronas ocultas. Hecho esto, se continuaba con las funciones de activación, así como con las funciones de pérdidas y distribuciones (las cuales tienen relación). Por último, se llegaba a la etapa de ajuste fino, modificando parámetros relacionados con la tasa de aprendizaje y el tamaño del *mini-batch*. En caso de detectarse *overfitting*, se intentaba corregir con funciones de activación con *Dropout* y modificando las ratios de *Dropout*, así como modificando los parámetros "l1" y "l2" y las ratios de *Dropout* de las entradas.

En cuanto a la detección de *overfitting* mencionada, se puede identificar por una discrepancia entre el rendimiento obtenido con los datos de entrenamiento y el obtenido con los datos de validación y test. En concreto, esto se debe a que el modelo aprende los patrones específicos de los datos de entrenamiento, patrones que no son los mismos que los de los datos de validación y test. Ello lleva a la diferencia en rendimiento cuando trabaja con datos que no son los de entrenamiento, ya que se ha especializado demasiado en los datos de entrenamiento y no es capaz de generalizar lo que ha aprendido de ellos cuando se le presentan los datos de validación y test (Salman & Liu, 2019). Gráficamente, puede detectarse si, al representar las métricas de pérdidas para entrenamiento y validación por cada época, se aprecia una separación importante entre ellas, como se aprecia en la Figura 13.

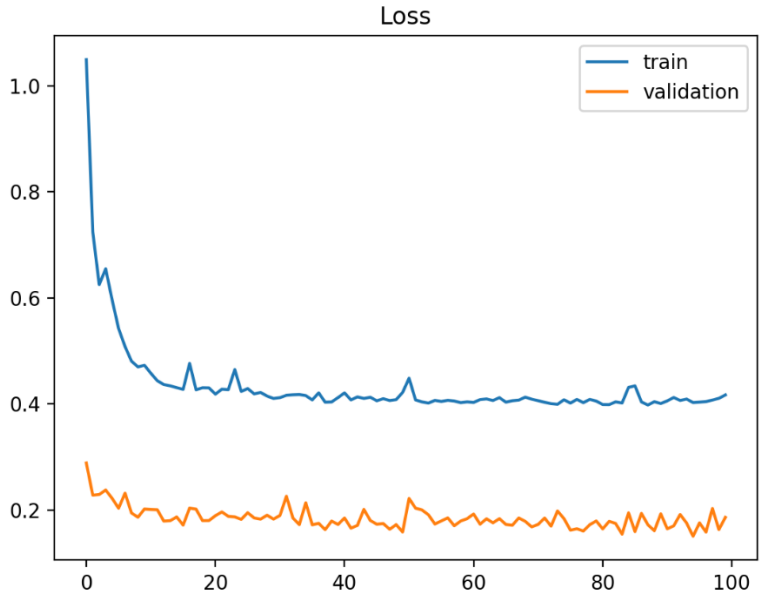


Figura 13. *Overfitting*. Fuente: https://machinelearningmastery.com/

4. RESULTADOS Y DISCUSIÓN

4.1. BASE DE DATOS

Como se ha comentado en la sección de Metodología, la base de datos se construye a partir de los datos, debidamente procesados, obtenidos de las páginas web de idealista. El resultado es una base de datos (tipo *data frame* en R), la cual puede almacenarse en un archivo csv mediante la instrucción "write.csv".

Tras la recopilación de datos de las páginas web, se obtiene una base de datos con 11.522 viviendas, con 6 campos distintos, correspondientes con las distintas características de las que se ha recogido información, como se puede en la Tabla 3.

	Precio	Barrio	Habitaciones	Metros_cuadrados	Planta	Ascensor
1	430000	sant-francesc	4	173	3	1
2	385000	sant-francesc	5	160	3	1
3	460000	sant-francesc	6	190	3	1
4	385000	sant-francesc	5	160	3	1
5	340000	sant-francesc	3	90	3	1
6	270000	sant-francesc	2	110	2	0
7	275000	sant-francesc	2	86	10	1
8	620000	sant-francesc	4	180	7	0

Showing 1 to 10 of 11,522 entries, 6 total columns

Tabla 3. Base de datos sin modificar.

Sin embargo, no todas las viviendas cuentan con todos los campos de características completos, por lo que se procede a eliminar todas aquellas que, al menos en uno de sus campos, tengan un valor sin especificar, tras lo cual quedan 9.988 viviendas con todos sus campos especificados.

Hecho esto, se procede a la detección de datos anómalos de las variables cuantitativas (precio, superficie en metros cuadrados y número de habitaciones). Para ello, se emplea la instrucción "boxplot", la cual, entre sus salidas, devuelve el valor de los datos anómalos, los cuales, gráficamente, son aquellos que quedan fuera de los "bigotes" de la representación que muestra la instrucción "boxplot", marcados con circunferencias huecas. Dichos bigotes se construyen sumando al tercer cuartil (75%) 1,5 veces el rango intercuartílico (tercer cuartil menos primer cuartil) y restando esto mismo al primer cuartil (25%). En la Figura 14, se puede apreciar el resultado del "boxplot" para el precio.

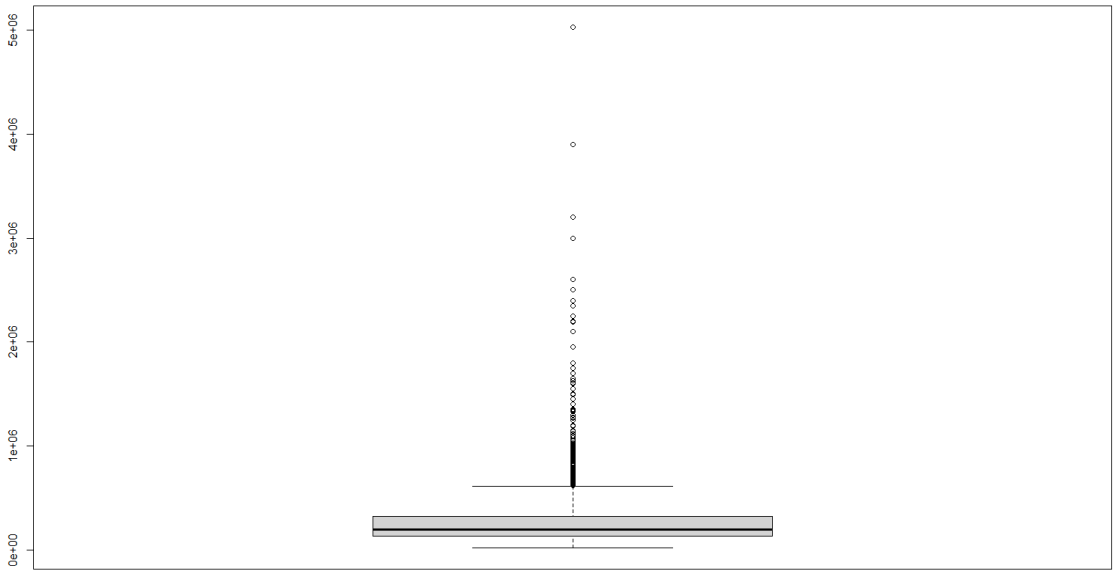


Figura 14. *Boxplot* para la variable precio (€).

Con los valores de los datos anómalos de las variables precio, superficie en metros cuadrados y número de habitaciones, se busca en la base de datos a qué viviendas pertenecen, procediéndose a su eliminación, y quedando, en total, 8.976 viviendas, con todos los campos de características con valores definidos, como puede observarse en la Tabla 4.

	Precio	Barrio	Habitaciones	Metros_cuadrados	Planta	Ascensor
1	430000	sant-francesc	4	173	3	1
2	385000	sant-francesc	5	160	3	1
3	460000	sant-francesc	6	190	3	1
4	385000	sant-francesc	5	160	3	1
5	340000	sant-francesc	3	90	3	1
6	270000	sant-francesc	2	110	2	0
7	275000	sant-francesc	2	86	10	1
9	260000	sant-francesc	2	49	3	1

Showing 1 to 10 of 8,976 entries, 6 total columns

Tabla 4. Base de datos final.

4.2. MODELO PREDICTIVO

Una vez obtenida y tratada la base de datos, se procede a la construcción y optimización del modelo predictivo basado en Redes Neuronales. Para ello, como se ha indicado en la sección de Metodología, se ha empleado la instrucción "h2o.deeplearning", perteneciente al paquete h2o, por tanto, el modelo estará basado en Redes Neuronales Profundas, aunque se puede forzar a que solo haya una capa oculta, por lo que sería una Red Neuronal de tipo *Shallow*, en vez de profunda, aunque los resultados de dicha arquitectura no han sido satisfactorios.

Así pues, el primer paso para construcción el modelo sería definir su arquitectura, es decir, el número de capas ocultas y el número de neuronas de cada una. Se toma como guía para decidir el número de capas la información de la Figura 7, la cual sugiere que un número de capas igual o mayor a 2 es capaz de aprender relaciones complejas entre las variables. Aun así, se han probado configuraciones con diversidad de número de capas y de neuronas por capa, encontrando que la solución que arroja resultados superiores al resto es la de emplear 4 capas ocultas de 100 neuronas cada una.

También se define el número de épocas por las que entrenar el modelo, seleccionándose 20.000 debido a que, en ese punto, los modelos alcanzan sobradamente lo que se podría definir como un régimen estacionario (no cambian sustancialmente los errores ni de entrenamiento ni de validación). También se establece el parámetro "stopping_rounds" a 0 para que se calculen las 20.000 épocas.

Cabe señalar que, cuando se dice que la configuración de cierto parámetro da un resultado superior a otro, significa, por lo general, que el MAE (Mean Absolute Error) o error absoluto medio obtenido respecto de los datos de test, es decir, al tratar de predecir con el modelo resultante los datos de test, ha sido menor.

Aclarado esto, el siguiente parámetro que ha tratado de optimizarse ha sido la función de activación de las neuronas de las capas ocultas, ya que la de la última capa no se puede variar, quedando además alineada con lo expresado en la Tabla 5, ya que se indica que la función de activación de dicha capa es lineal. En cuanto a la función de activación de las capas ocultas, se llega a la conclusión, tras las pruebas pertinentes, de que la que mejores resultados arroja es la de tipo "Rectifier".

Problem Type	Output Type	Final Activation Function	Loss Function
Regression	Numerical value	Linear	Mean Squared Error (MSE)
Classification	Binary outcome	Sigmoid	Binary Cross Entropy
Classification	Single label, multiple classes	Softmax	Cross Entropy
Classification	Multiple labels, multiple classes	Sigmoid	Binary Cross Entropy

Tabla 5. Funciones de activación de la última capa y funciones de pérdidas ideales según el tipo de problema. Fuente: https://towardsdatascience.com/

Después, por lo que respecta a la función de pérdidas y la distribución, las cuales guardan relación (no todas las funciones de pérdidas pueden emplearse con todas las distribuciones), se ha observado que, pese a lo indicado en la Tabla 5, la mejor combinación de función de pérdidas y distribución es "Huber" y "huber", respectivamente, lo cual, dicho sea de paso, es concordante con lo expresado en RapidMiner 2020, ya que se indica que esta es una de las combinaciones que se puede emplear para un problema de regresión, como es el que ocupa al presente trabajo.

Por lo referente al parámetro "mini_batch_size", parece, en base a las pruebas llevadas a cabo, que lo indicado por Masters y Luschi 2018, es decir, que su óptimo se sitúa en un valor de 32, es cierto.

En este punto, se procede a ajustar los parámetros relacionados con la tasa de aprendizaje ("rho" y "epsilon"), concluyendo que su configuración óptima es, para "rho", un valor de 0.999, y, para "epsilon", el valor por defecto, 10^{-8}.

En este punto, se procede a intentar resolver un comportamiento que se está detectando, que no es otro que un cierto *overfitting* del modelo. Esto puede verse en la Figura 15, donde se ve representado el RMSE (Rooted Mean Squared Error) o la raíz del error cuadrático medio, otra manera de medir el error cometido, en función del número de épocas, para la configuración con los parámetros especificados hasta ahora. Se aprecia el *overfitting* en la marcada diferencia que existe entre el RMSE de entrenamiento (en azul) y el de validación (en amarillo), el cual es análogo al obtenido para test, para el cual se obtiene un MAE de 35,260.09€.

Figura 15. Curva de aprendizaje: RMSE en entrenamiento y validación en función de la época.

Por otra parte, en la recomendación de (H2O, s.f.), se indica que se le dé al parámetro "max_w2", que se señala que mejora la estabilidad del modelo al usar la función de activación "Rectifier", el valor de 10, lo cual, *ceteris paribus* respecto al caso ilustrado en la Figura 15, que arrojaba un MAE para test de 35,260.09€, disminuye dicho MAE para test hasta los 34.277,97€.

Para abordar el problema del *overfitting*, se prueba a modificar los parámetros de regularización "l1" y "l2", siguiendo la recomendación de (H2O, s.f.), la cual indica un valor de 10^{-5} para ambos parámetros, del MAE de mejor que ambos parámetros a 0, disminuyendo el MAE cometido en el test hasta 33.204,71€, respecto de los 34.277,97€, sin embargo, a nivel de *overfitting*, gráficamente se sigue observando el mismo comportamiento.

A continuación, a fin de continuar de tratar de resolver el *overfitting*, se modifica el parámetro "input_dropout_ratio", dentro de los rangos recomendados en la propia ayuda del paquete h2o, los cuales son 0,1 a 0,2. Paralela y combinadamente, se modifica la función de activación por "RectifierWithDropout", y se varían los valores de *Dropout* para cada capa oculta (todos con el mismo valor) ("hidden_dropout_ratios").

Se observa, en ambos casos, que, si bien se reduce algo la discrepancia entre el error de entrenamiento y el error de validación, esto sucede por aumento del error de entrenamiento, el de validación y el de test (MAE), en ningún caso mejoran. Además, se observa una notable inestabilidad en la gráfica de la curva de aprendizaje, en la que se representa el RMSE de entrenamiento y validación en función de la época, como puede apreciarse en la Figura 16, donde se muestra el modelo con "input_dropout_ratio" de 0,1 y "hidden_dropout_ratios" de 0,05 (para todas las capas ocultas.

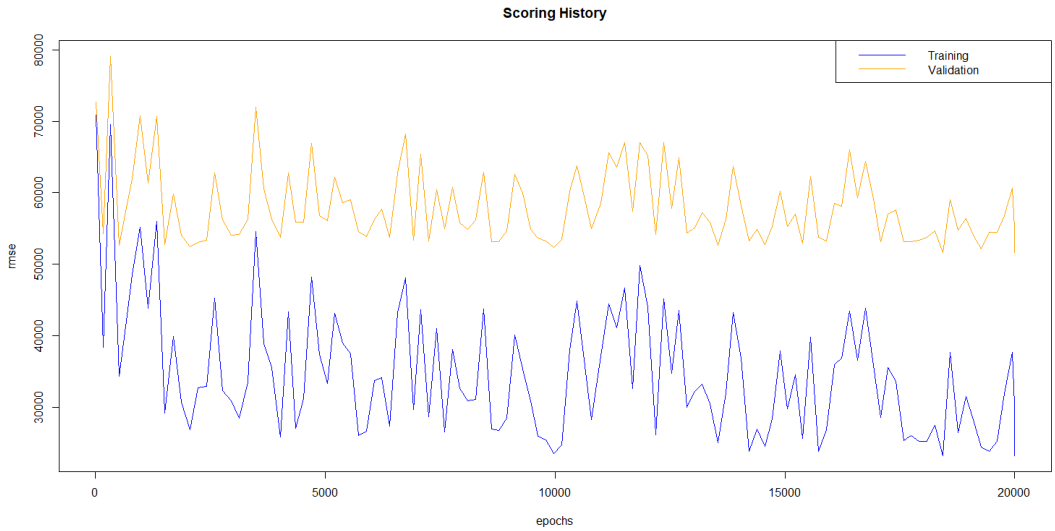


Figura 16. Curva de aprendizaje para "input_dropout_ratio" de 0,1 y "hidden_dropout_ratios" de 0,05.

Esto último implica que, al intentar solucionar el *overfitting*, se consiguen unos resultados en términos del MAE en test inferiores, lo cual resulta paradójico, ya que se suele considerar que, si se mejora el *overfitting*, debería mejorar el error en validación y test, pero esto no sucede así en este caso.

De esta manera, se llega al modelo final, cuyos parámetros relevantes distintos a los que se establecen por defecto son los siguientes:

- Función de activación ("activation"): "Rectifier"
- Número de capas y neuronas ocultas ("hidden"): c(100,100,100,100)
- Función de pérdidas ("loss"): "Huber"
- Distribución ("distribution"): "huber"
- Parámetro de regularización 1 ("l1"): 10^{-5}
- Parámetro de regularización 2 ("l2"): 10^{-5}
- Parámetro de pesos máximos ("max_w2"): 10
- Parámetro de la tasa de aprendizaje adaptativa ADADELTA ("rho"): 0,999
- Tamaño del *mini-batch* ("mini_batch_size"): 32

El resultado que arroja este modelo en términos de MAE es, como se había indicado anteriormente, de 33.204,71€. Es decir, de media, el modelo comete un error, bien de sobrevaloración, bien de infravaloración, de 33.204,71€ al tratar de predecir el precio de una vivienda perteneciente a los datos de test, a partir de su superficie en metros cuadrados, el barrio en el que se ubica, el número de habitaciones, la planta y si tiene o no ascensor. En la Figura 17, puede observarse la curva de aprendizaje del modelo, y, en la Tabla 6, los parámetros que devuelve el mismo, incluyendo la arquitectura.

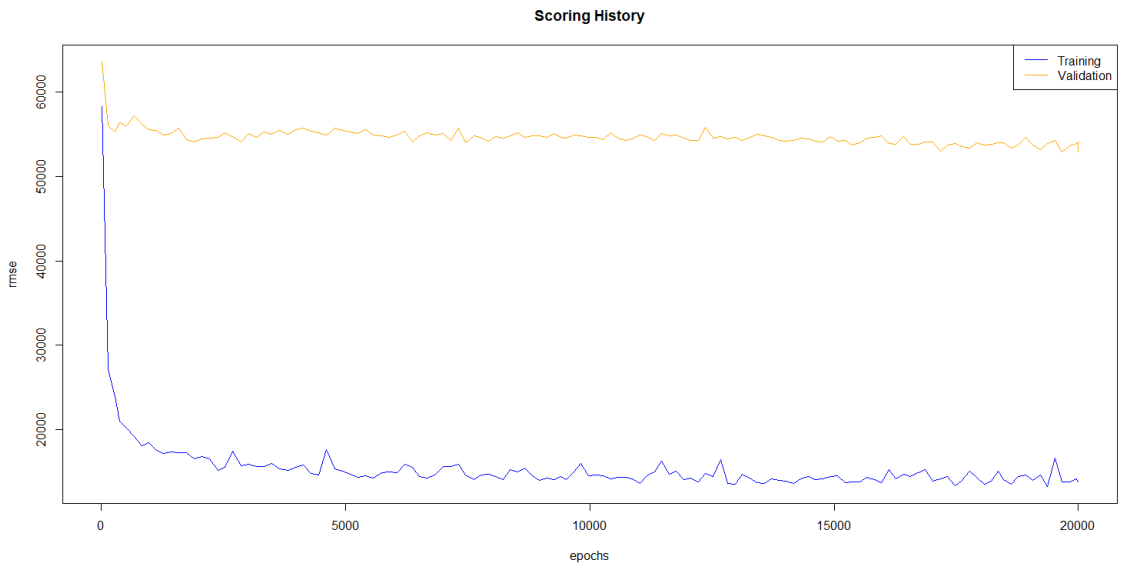


Figura 17. Curva de aprendizaje del modelo definitivo.

	layer	units	type	dropout	l1	l2	mean_rate	rate_rms	momentum	mean_weight	weight_rms	mean_bias	bias_rms
1	1	104	Input	0.00 %	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	2	100	Rectifier	0.00 %	0.000010	0.000010	0.073164	0.191899	0.000000	-0.063096	0.299760	-0.216723	0.491469
3	3	100	Rectifier	0.00 %	0.000010	0.000010	0.036722	0.030030	0.000000	-0.088791	0.298202	0.187124	0.376574
4	4	100	Rectifier	0.00 %	0.000010	0.000010	0.047942	0.034238	0.000000	-0.076659	0.292152	0.084635	0.221662
5	5	100	Rectifier	0.00 %	0.000010	0.000010	0.081462	0.081421	0.000000	-0.104951	0.287867	-0.171933	0.544793
6	6	1	Linear	NA	0.000010	0.000010	0.009675	0.007157	0.000000	0.030980	0.272810	0.154239	0.000000

Tabla 6. Información del modelo definitivo.

Por lo que respecta al error cometido, se pueden observar todas las métricas que se proporcionan con el modelo, para los tres conjuntos de datos (entrenamiento, validación y test) en la Figura 18.

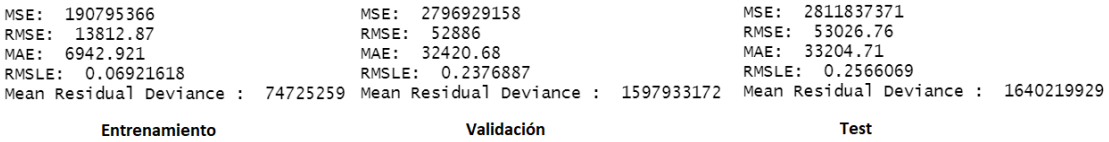


Figura 18. Métricas de error para datos de entrenamiento, validación y test del modelo definitivo.

Por otra parte, se dispone de la importancia que se le da a cada variable en la Tabla 7, obtenida como la suma del porcentaje de importancia que se le da a cada variable, calculada empleando el método Gedeon (Candel et al., 2018). Ha de notarse que, al emplear codificación *One-Hot*, por cada uno de los niveles de las variables categóricas se genera, por así decir, una nueva variable. Por tanto, como se puede observar en la Tabla 6, en la capa input hay un total de 104 entradas, correspondientes a los 73 niveles de la variable "Barrios", los 24 de la variable "Planta", los 2 de la variable "Ascensor", las variables "Metros_cuadrados" y "Habitaciones", así como tres variables llamadas "Barrios.missing(NA)", "Planta.missing(NA)" y "Ascensor.missing(NA)", para observaciones en las que falten estos campos, aunque en este caso no hay ninguna, así que estas variables no tienen relevancia alguna.

En la Tabla 7 se observa que la variable "Barrio" es la que mayor peso total tiene en el modelo, seguida, a cierta distancia de la variable "Planta". A continuación, llama la atención el exiguo peso que tiene la variable "Metros_cuadrados", ya que, intuitivamente, se había presupuesto que tendría una relevancia considerable y, de hecho, en modelos elaborados sin codificación *One-Hot*, así sucedía, siendo la variable más relevante, aunque también es cierto que dichos modelos han demostrado ser inferiores en rendimiento. Por último, con pesos aún menores, y similares entre sí, se encuentran las variables "Ascensor" y "Habitaciones".

Variable	Suma de porcentajes de importancia
Barrio	74,74%
Planta	18,83%
Metros_cuadrados	3,09%
Ascensor	1,8%
Habitaciones	1,54%

Tabla 7. Importancia de las variables para el modelo definitivo.

De hecho, en la Tabla 8, donde se puede apreciar la importancia de las variables en un modelo con los mismos parámetros pero habiendo hecho una codificación manual con enteros de las variables categóricas, se aprecian las enormes discrepancias en comparación con la importancia de las variables obtenidas en el modelo con codificación *One-Hot*, lo cual llama la atención y, a la vez, hace dudar del empleo de este indicador como verdadero indicador de cuánto afecta qué variable a la formación de precios. Nótese que el modelo sin codificación *One-Hot* ha rendido peor que el que sí cuenta con esta codificación, obteniendo un MAE de 38.512,87€.

Variable	Porcentajes de importancia
Barrio	21,78%
Planta	18,59%
Metros_cuadrados	29,59%
Ascensor	10,5%
Habitaciones	19,53%

Tabla 8. Importancia de las variables para modelo como el definitivo, pero sin codificación *One-Hot*

Por último, con el fin de contextualizar y evaluar el rendimiento que ha obtenido el modelo, se ha de comparar el MAE obtenido con cómo se distribuyen los precios en las viviendas de la base de datos obtenida. Para ello, se emplean la media y la mediana de los mismos, las cuales son de 211.874,4€ y 179.000€, respectivamente.

De esta manera, se tiene que, el MAE o error absoluto medio cometido, en tanto por cien, sobre el precio medio de las viviendas de la base de datos es de un 15,67%, siendo de un 18,55% para el caso del precio mediano de las viviendas. Así pues, pese a que no es un error despreciable, también es cierto que existen aspectos, bien no tenidos en cuenta, como el estado de la vivienda o cuándo se construyó, si está amueblada, etc., bien de naturaleza subjetiva, como cualquier apreciación estética de la fachada o el interior, o la calidad de los materiales empleados, que pueden explicar, de cierto modo, este error.

Por tanto, dadas las limitaciones a las que se ha visto sometido el modelo, se puede considerar que el resultado obtenido, pese a estar lejos de la perfección, puede tener cierta utilidad, por ejemplo, a la hora de querer poner en venta una vivienda, para hacerse una idea aproximada, teniendo en cuenta el margen de error con el que se trabaja, del precio de oferta al que anunciarla. A partir de esa idea aproximada, se procedería a modular el precio de oferta en función de aspectos que valore el propio oferente, como en qué estado considera que está la vivienda, qué calidad de acabados posee, la estética, o, mismamente, la necesidad que tenga el vendedor por conseguir cerrar la venta de la vivienda.

5. CONCLUSIONES Y LÍNEAS FUTURAS

A lo largo del presente trabajo, se ha empleado una serie de herramientas como son el *Web Scraping* y las Redes Neuronales Profundas que, hoy en día, están al alcance del usuario y, cuyo gran potencial ha quedado, de cierta manera, plasmado.

Aunque es cierto que en la parte de la recopilación automática de la base de datos no se ha gozado del éxito que se esperaba, teniendo que realizar descargas de manera manual, una vez hecho esto, se ha logrado un algoritmo que predice el precio de una vivienda, tomando en cuenta su superficie en metros cuadrados, el barrio en el que se ubica, el número de habitaciones, la planta y si tiene o no ascensor, que yerra, de media, en poco más de 33.000€ (15,66% sobre el precio medio y 18,55% sobre el precio mediano), lo cual, como se ha expuesto al final de la sección de resultados, puede considerarse como un resultado satisfactorio y de cierta utilidad, probándose así el potencial de estas herramientas.

Sin embargo, la extracción de información valiosa para la comprensión del mercado inmobiliario de la ciudad de Valencia del modelo construido es algo dudosa, ya que la importancia de cada una de las variables que se ha obtenido para el modelo con codificación *One-Hot* respecto al modelo codificado a mano con enteros es lo suficientemente amplia como para, por prudencia, no tomar por válida la información obtenida acerca de la importancia de las variables, ya que la relación de causalidad con la formación del precio de oferta no parece, en absoluto, razonable con la importancia de las variables. Por ejemplo, parece algo muy improbable que solamente el 3,09% del precio de oferta de una vivienda esté condicionado por su superficie en metros cuadrados, antojándose una influencia excesivamente escasa.

Es más, como se ha comentado en la sección de resultados, hay variables que no han sido tenidas en cuenta, como el estado de la vivienda, en qué año se construyó, si está amueblada, la estética del interior, o la calidad de los materiales empleados. Estas variables, en principio, tendrán cierta importancia sobre la formación del precio de oferta de la vivienda, por lo que, incluso aunque la importancia de cada variable pudiese tomarse como válida, debería ser tomada simplemente de manera orientativa.

Ha de reconocerse una limitación en la realización del presente trabajo, la cual es que el precio de las viviendas, tanto el obtenido de la web de idealista, como el que intenta predecirse, es el precio de la oferta en la página web, no incluyendo la posible negociación de las partes, lo cual modificaría el precio de adquisición, normalmente a la baja.

Otra limitación que afecta al trabajo realizado es la situación de crisis sanitaria en la cual se han extraído los datos (07/06/2020), situación que implica una elevada inestabilidad y que, como se ha explicado en la sección de Introducción, si bien no parece haber afectado fuertemente a la evolución de los precios por metro cuadrado, no se ha realizado el análisis de si ha afectado más a unos rangos de precios que a otros. Además, no se conoce la evolución que va a sufrir este mercado, por lo que, en definitiva, los resultados obtenidos son específicos a la fecha de recogida de los datos y, si se quisiera hacer uso de los resultados de un modelo como este, lo óptimo sería emplear los datos de la fecha más cercana posible a la fecha en que se pretenda usar.

Por otro lado, queda, como línea a seguir de este trabajo, la realización de un modelo que, en su base de datos, tenga en cuenta más variables, como, por ejemplo, el estado de la vivienda (si es nueva o no, o si está reformada), el año de construcción del edificio, si está amueblada, e incluso una segmentación por zonas más reducidas, todo con el objetivo de lograr predicciones más precisas.

También queda como línea futura de trabajo, el empleo de librerías más sofisticadas para evitar ser bloqueados en el proceso de *Web Scraping*, como, por ejemplo, Selenium.

Por último, queda pendiente la implementación de los algoritmos de *Deep Learning* mediante GPU (*Graphics Processing Unit*) o tarjeta gráfica, aumentando así considerablemente la rapidez de las pruebas de construcción de modelos, lo cual, para este trabajo en particular, ha supuesto un importante cuello de botella. Esto puede lograrse empleando la librería Keras, y Tensorflow como *backend*, por ejemplo, e incluso puede hacerse gratuitamente (o con una suscripción de pago para más recursos) y en línea mediante la herramienta Google Colaboratory.

Por la parte personal, este trabajo ha servido para reforzar y ampliar los conceptos acerca de las Redes Neuronales, así como sus fundamentos. Además, se ha experimentado el cómo afectan los cambios en los parámetros y cómo ha de organizarse el código para poder comparar y decidir qué valores son los óptimos.

En cuanto al *Web Scraping*, la experiencia, pese a la frustración por la imposibilidad de superar los bloqueos sufridos por el servidor de la página de idealista, ha sido positiva, en tanto en cuanto se han adquirido conocimientos sobre su funcionamiento específico, así como sobre diversos aspectos del lenguaje de programación R, haciendo hincapié en el manejo de cadenas y de otros tipos de datos, lo cual puede resultar útil en el futuro.

REFERENCIAS

ABC. (4 de Marzo de 2020). *Coronavirus: un nuevo sistema diagnostica Covid-19 por radiografía con un 97% de éxito*. Recuperado el 20 de Agosto de 2020, de https://www.abc.es/espana/comunidad-valenciana/abci-coronavirus-nuevo-sistema-diagnostica-covid-19-radiografia-97-por-ciento-exito-202005031142_noticia.html?ref=https:%2F%2Fes-es.facebook.com%2F

AEVI. (2020). *El videojuego en el mundo*. Recuperado el 21 de Agosto de 2020, de http://www.aevi.org.es/la-industria-del-videojuego/en-el-mundo/#:~:text=El%20mercado%20internacional%20del%20videojuego,seg%C3%BAn%20las%20estimaciones%20de%20Newzoo.

BBVA. (8 de Noviembre de 2019). *'Machine learning': ¿qué es y cómo funciona?* Recuperado el 10 de Agosto de 2020, de https://www.bbva.com/es/machine-learning-que-es-y-como-funciona/

Candel, A., LeDell, E., & Bartz, A. (Abril de 2018). *Deep Learning with H2O.* Recuperado el 6 de Agosto de 2020, de Deep Learning Booklet: https://h2o-release.s3.amazonaws.com/h2o/rel-wolpert/8/docs-website/h2o-docs/booklets/DeepLearningBooklet.pdf

Chiarazzo, V., Caggiani, L., Marinelli, M., & Ottomanelli, M. (2014). A Neural Network based Model for Real Estate Price Estimation Considering Environmental Quality of Property Location.

Eady, Y. (7 de Mayo de 2019). *Tesla's Deep Learning at Scale: Using Billions of Miles to Train Neural Networks*. Recuperado el 21 de Agosto de 2020, de https://towardsdatascience.com/teslas-deep-learning-at-scale-7eed85b235d3

Edelsten, A. (15 de Febrero de 2019). *NVIDIA DLSS: Damos respuesta a tus preguntas*. Recuperado el 21 de Agosto de 2020, de https://www.nvidia.com/es-es/geforce/news/nvidia-dlss-your-questions-answered/

El País. (11 de Junio de 2020). *Cinco Días*. Recuperado el 31 de Agosto de 2020, de La compraventa de casas baja el 39% en abril y sufre el mayor descenso en 11 años: https://cincodias.elpais.com/cincodias/2020/06/11/economia/1591861025_193423.html

Elite Data Science. (2019). *5 Tasty Python Web Scraping Libraries*. Recuperado el 22 de Agosto de 2020, de https://elitedatascience.com/python-web-scraping-libraries

Fang, M. Y.-S., Manipatrui, S., Wierzynski, C., Khosrowshahi, A., & DeWeese, M. R. (2019). Design of optical neural networks with component imprecisions. *Optical Society of America*. Obtenido de https://www.osapublishing.org/oe/fulltext.cfm?uri=oe-27-10-14009&id=411885

H2O. (s.f.). *H2O Tutorials*. Recuperado el 11 de Agosto de 2020, de https://docs.h2o.ai/h2o-tutorials/latest-stable/tutorials/deeplearning/index.html

Heaton, J. (1 de Junio de 2017). *Heaton Research*. Recuperado el 18 de Agosto de 2020, de The Number of Hidden Layers: https://www.heatonresearch.com/2017/06/01/hidden-layers.html

Izaurieta, F., & Saavedra, C. (2000). Redes Neuronales Artificiales. *Departamento de Física, Universidad de Concepción Chile*.

Landers, R. N., Brusso, R. C., Cavanaugh, K. J., & Collmus, A. B. (2016). A primer on theory-driven web scraping: Automatic extraction of big data from the Internet for use in psychological research. *American Psychological Association*. doi:http://dx.doi.org/10.1037/met0000081

Leys, C., Delacre, M., L.Mora, Y., Lakens, D., & Ley, C. (2019). How to Classify, Detect, and Manage Univariate and Multivariate Outliers, With Emphasis on Pre-Registration. *International Review of Social Psychology*. Obtenido de https://www.rips-irsp.com/articles/10.5334/irsp.289/

Martí, A. (15 de Marzo de 2020). *NVIDIA nos propone cómo poner sus gráficas al servicio de la investigación sobre el COVID-19: la iniciativa Folding@Home*. Recuperado el 20 de Agosto de 2020, de https://www.xataka.com/otros/nvidia-nos-propone-como-poner-sus-graficas-al-servicio-investigacion-covid-20-iniciativa-folding-home

Masters, D., & Luschi, C. (20 de Abril de 2018). Revisiting Small Batch Training for Deep Neural Networks. Obtenido de https://arxiv.org/pdf/1804.07612.pdf

Mhaskar, H., Liao, Q., & Poggio, T. (2017). When and Why Are Deep Networks Better than Shallow Ones? *Thirty-First AAAI Conference on Artificial Intelligence*.

Miranda, L. (6 de Abril de 2020). *NVIDIA ofrecerá 30 supercomputadoras para luchar contra el coronavirus*. Recuperado el 20 de Agosto de 2020, de https://hipertextual.com/2020/04/nvidia-supercomputadoras-coronavirus

Naranjo, V., & Colomer, A. (Febrero de 2020). Deep Learning aplicado al análisis de señales e imágenes. *Puesta en marcha de una red neuronal: aprendizaje y evaluación*. CFP, UPV.

NVIDIA. (2020). *Deep Learning AI*. Recuperado el 5 de Agosto de 2020, de https://www.nvidia.com/en-us/deep-learning-ai/

NVIDIA. (2020). *NVIDIA CLARA PARABRICKS*. Recuperado el 20 de Agosto de 2020, de https://www.nvidia.com/es-es/healthcare/clara-parabricks/

Oracle. (2020). *¿Qué es big data?* Recuperado el 5 de Agosto de 2020, de https://www.oracle.com/es/big-data/what-is-big-data.html

Oxdata, Inc. (2013). *Data Science in H2O*. Recuperado el 3 de Agosto de 2020, de Deep Learning: https://s3.amazonaws.com/h2o-release/h2o/rel-kahan/10/docs-website/datascience/deeplearning.html

RapidMiner. (2020). *Deep Learning (H2O)*. Recuperado el 27 de Agosto de 2020, de https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/neural_nets/deep_learning.html

Rouhiainen, L. (2018). Inteligencia artificial. *Alienta Editorial*.

Rouse, M. (Noviembre de 2012). *Análisis de "big data"*. Recuperado el 5 de Agosto de 2020, de https://searchdatacenter.techtarget.com/es/definicion/Analisis-de-big-data

Salman, S., & Liu, X. (2019). Overfitting Mechanism and Avoidance in Deep Neural Networks. Obtenido de https://www.researchgate.net/publication/330553571_Overfitting_Mechanism_and_Avoidance_in_Deep_Neural_Networks

UPV. (27 de Julio de 2017). *Proyecto GALAHAD*. Recuperado el 21 de Agosto de 2020, de La UPV desarrolla un sistema de análisis de imagen para mejorar y abaratar el coste de la detección de glaucoma: https://www.upv.es/noticias-upv/noticia-9268-proyecto-galaha-es.html

Varma, A., Sarma, A., Doshi, S., & Nair, R. (2018). House Price Prediction Using Machine Learning and Neural Networks. *IEEE*.

Vasudev, R. (3 de Agosto de 2017). *What is One Hot Encoding? Why And When do you have to use it?* Recuperado el 31 de Agosto de 2020, de https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f

Wang, D., Khosla, A., Gargeya, R., Irshad, H., & Beck, A. H. (2016). Deep Learning for Identifying Metastatic Breast Cancer. Obtenido de https://arxiv.org/pdf/1606.05718.pdf

Wang, J. J., Hu, S. G., Zhan, X. T., Luo, Q., Yu, Q., Liu, Z., . . . Liu, Y. (2018). Predicting House Price With a Memristor-Based Artificial Neural Network. *IEEE*.

Yan, Y. (27 de Mayo de 2020). *Web Scraping Reference: Cheat Sheet for Web Scraping using R*. Recuperado el 23 de Agosto de 2020, de https://github.com/yusuzech/r-web-scraping-cheat-sheet/blob/master/README.md

Zhang, W. J., Yang, G., Lin, Y., Ji, C., & Gupta, M. M. (2018). On Definition of Deep Learning. *World Automation Congress (WAC)*, (págs. 1-5). Stevenson, WA, USA. Obtenido de https://ieeexplore.ieee.org/abstract/document/8430387?casa_token=SUFer6oFwJoAAAAA:30pb85Ls--S9Rcd1jGG0vJO8V8TdTgHsE7eIZ7ji5IJ6rGtrqK_3T3pHuLyHWUIJhhaNIuwnDA

Zuo, Y., Li, B., Zhao, Y., Jiang, Y., Chen, Y.-C., Chen, P., . . . Du, S. (2019). All-optical neural network with nonlinear activation functions. *Optical Society of America*. Obtenido de https://www.osapublishing.org/optica/fulltext.cfm?uri=optica-6-9-1132&id=417261

Anexo 1: Informe de dataMaid de la base de datos inicial

MIS_DATOS_FRAMED

AUTOGENERATED DATA SUMMARY FROM DATAMAID

2020-08-28 23:47:01

Data report overview

The dataset examined has the following dimensions:

Feature	Result
Number of observations	11522
Number of variables	6

Checks performed

The following variable checks were performed, depending on the data type of each variable:

	character	factor	labelled	haven labelled	numeric	integer	logical	Date
Identify miscoded missing values	×	×	×	×	×	×		×
Identify prefixed and suffixed whitespace	×	×	×	×				
Identify levels with < 6 obs.	×	×	×	×				

Identify case issues	×	×	×	×				
Identify misclassified numeric or integer variables	×	×	×	×				
Identify outliers					×	×		×

Please note that all numerical values in the following have been rounded to 2 decimals.

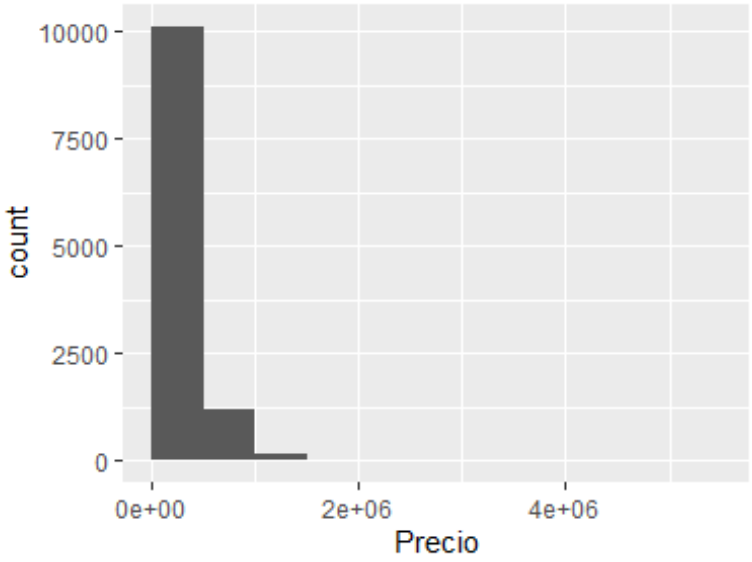
Summary table

	Variable class	# unique values	Missing observations	Any problems?
Precio	numeric	1356	0.00 %	×
Barrio	character	73	0.00 %	
Habitaciones	character	15	0.00 %	×
Metros_cuadrados	numeric	371	0.00 %	×
Planta	character	26	0.00 %	×
Ascensor	character	3	0.00 %	

Variable list

Precio

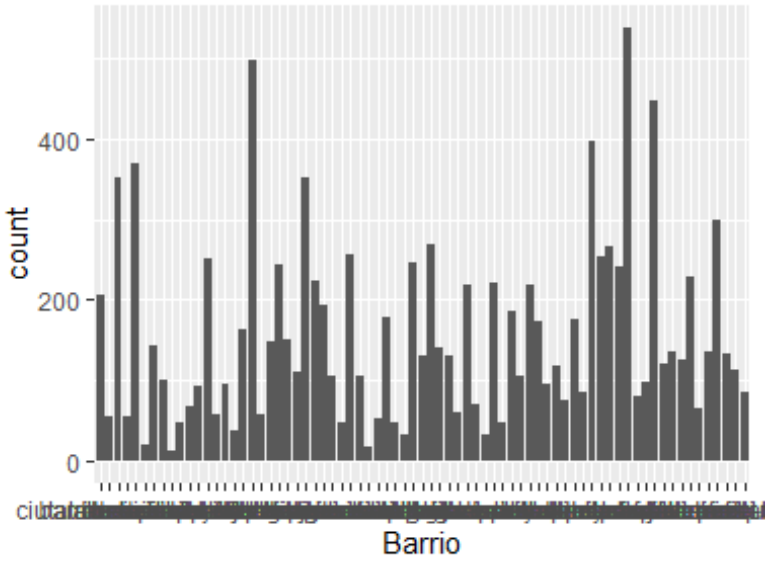
Feature	Result
Variable type	numeric
Number of missing obs.	0 (0 %)
Number of unique values	1356
Median	198000
1st and 3rd quartiles	135000; 337750
Min. and max.	20200; 5029594



- Note that the following possible outlier values were detected: "20200", "24500", "25000", "29000", "30000", "31600", "33300", "33800", "34000", "36000" (151 additional values omitted).

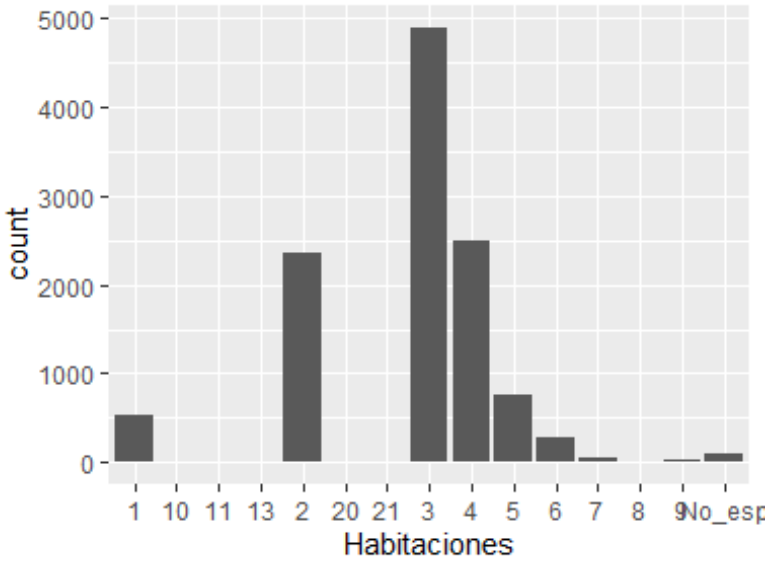
Barrio

Feature	Result
Variable type	character
Number of missing obs.	0 (0 %)
Number of unique values	73
Mode	“russafa”



Habitaciones

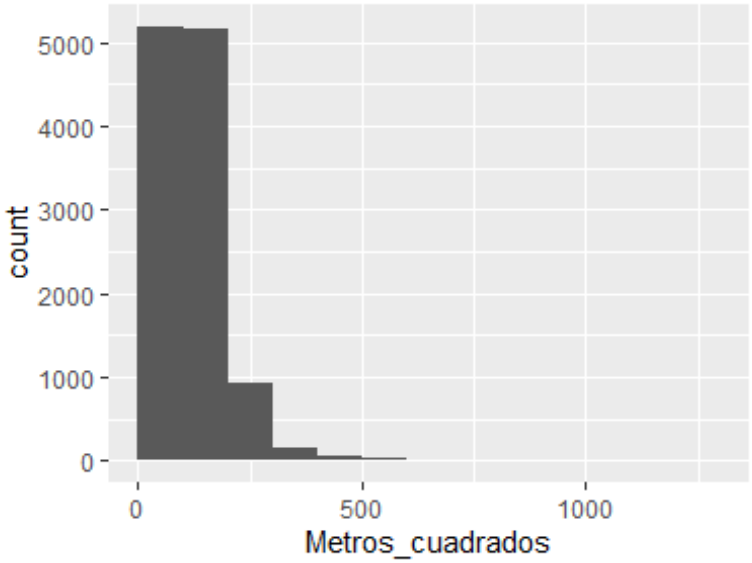
Feature	Result
Variable type	character
Number of missing obs.	0 (0 %)
Number of unique values	15
Mode	“3”



- The following suspected missing value codes enter as regular values: "8", "9".
- Note that the following levels have at most five observations: "11", "13", "20", "21".

Metros_cuadrados

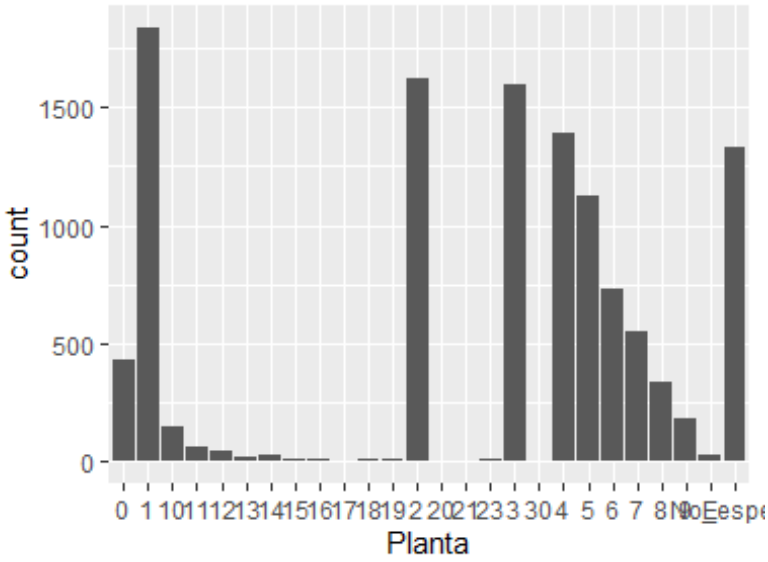
Feature	Result
Variable type	numeric
Number of missing obs.	0 (0 %)
Number of unique values	371
Median	105
1st and 3rd quartiles	81; 139
Min. and max.	18; 1265



- Note that the following possible outlier values were detected: "18", "21", "25", "26", "28", "29", "30", "33", "34", "35" (94 additional values omitted).

Planta

Feature	Result
Variable type	character
Number of missing obs.	0 (0 %)
Number of unique values	26
Mode	“1”

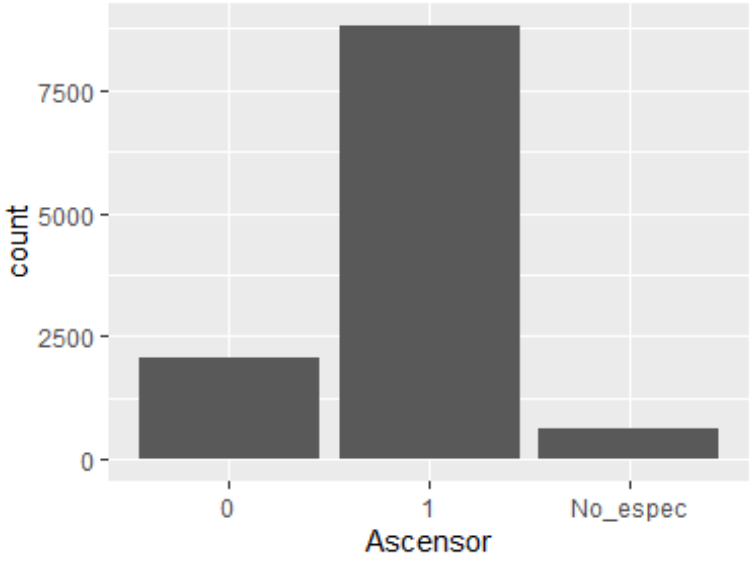


- The following suspected missing value codes enter as regular values: "8", "9".

- Note that the following levels have at most five observations: "17", "20", "21", "30".

Ascensor

Feature	Result
Variable type	character
Number of missing obs.	0 (0 %)
Number of unique values	3
Mode	“1”



Report generation information:

- Created by: Could not determine from system (username: Unknown)
- Report creation time: vi. ago. 28 2020 23:47:01
- Report was run from directory: `C:/Users/Usuario/Documents`
- dataMaid v1.4.0 [Pkg: 2019-12-10 from CRAN (R 4.0.2)]
- R version 4.0.1 (2020-06-06).
- Platform: x86_64-w64-mingw32/x64 (64-bit)(Windows 10 x64 (build 18362)).
- Function call: `makeDataReport(data = mis_datos_framed, replace = TRUE)`