TRABAJO DE FIN DE GRADO

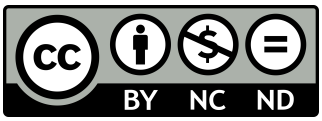
PREDICCIÓN DEL VALOR DE MERCADO DE UNA VIVIENDA EN LA CIUDAD DE BARCELONA MEDIANTE LA OBTENCIÓN DE UN CONJUNTO DE DATOS Y EL DESARROLLO DE UN ALGORITMO DE APRENDIZAJE AUTOMÁTICO

Miguel Ángel Miravé Carreño

Ingeniería Informática

Humberto Andrés Sanz

Junio de 2023



FICHA DEL TRABAJO FINAL

Título del trabajo:	Predicción del valor de mercado de una vivienda en la ciudad de Barcelona mediante la obtención de un conjunto de datos y el desarrollo de un algoritmo de aprendizaje automático.
Nombre del autor:	Miguel Ángel Miravé Carreño
Nombre del consultor:	Humberto Andrés Sanz
Fecha de entrega (mm/aaaa):	06/2023
Área del Trabajo Final:	Business Intelligence
Titulación:	Ingeniería Informática

Resumen del Trabajo (máximo 250 palabras):

El propósito de este trabajo es obtener una herramienta que permita predecir el valor de mercado de una vivienda en la ciudad de Barcelona a partir de la obtención de un conjunto de datos y el uso de un modelo de aprendizaje computacional. Se realizan diversos análisis para identificar los atributos de los inmuebles que determinan su precio, la fuente y el método de extracción de datos óptimos y el modelo de aprendizaje computacional idóneo para predecir precios de viviendas. Se obtiene el conjunto de datos del portal inmobiliario Idealista mediante una herramienta de *web scraper*. El conjunto de datos es tratado y analizado para posteriormente ser suministrado a un algoritmo de aprendizaje computacional XGBoost, que se desarrolla, optimiza y evalúa. La métrica de ajuste obtenida es el error medio absoluto relativo, y su valor es 15%. El ajuste del modelo se considera satisfactorio comparativamente, siendo similar al de las tasadoras oficiales y sustancialmente inferior al de las tasadoras en línea gratuitas. Se desarrolla una interfaz gráfica que permite al usuario obtener una predicción del valor de la vivienda a partir de los atributos introducidos.

Abstract (in English, 250 words or less):

The purpose of this project is to obtain a tool that allows predicting the market value of a residential property in the city of Barcelona by obtaining a dataset and using a machine learning model. Various analyses are conducted to identify the attributes of properties that determine their price, the optimal data source and extraction method, and the suitable machine learning model for predicting housing prices. The dataset is obtained from the real estate portal Idealista using a web scraper tool. The dataset is processed and analyzed before being provided to an XGBoost machine learning algorithm, which is developed, optimized, and evaluated. The fitting metric obtained is the relative mean absolute error, with a value of 15%. The model's fit is considered satisfactory comparatively, being similar to that of official appraisers and substantially lower than that of free online appraisers. A graphical interface is developed, allowing users to obtain a prediction of the property value based on the attributes inputted.

Palabras clave (entre 4 y 8):

Predicción de precios; Mercado inmobiliario de Barcelona; Análisis de datos inmobiliarios; Aprendizaje automático; Algoritmo XGBoost; Extracción de datos; *Web scraping*; Interfaz gráfica.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

1. Introducción

En este primer capítulo se introducen las motivaciones del trabajo, incluyendo el contexto y la justificación de este, así como los objetivos, el enfoque y métodos seguidos y la planificación temporal de su desarrollo. Finalmente, se detallan los productos obtenidos y se describen el resto de los capítulos que conforman el conjunto del trabajo.

1.1. Contexto y justificación del Trabajo

Una de las principales consideraciones a la hora de plantear este tema de trabajo fue el auge en la actividad del sector inmobiliario en la ciudad de Barcelona y, más particularmente, el auge en la actividad de compraventa y alquiler de viviendas en la ciudad. Aunque las rápidas y acusadas subidas de tipos de interés acometidas por el BCE puedan frenar en un futuro próximo la tendencia alcista en la actividad, 2022 fue el año en el que se vendieron un mayor número de viviendas desde el final de la burbuja inmobiliaria sufrida a nivel nacional[1]. En concreto, Barcelona fue en 2022 la ciudad española más cara para comprar y alquilar[2], y en ella se vendieron algo más de 4.000 viviendas[3]. Por esta razón, puede ser útil una herramienta que sirva a los diferentes agentes en el mercado para determinar o valorar un inmueble.

Si esta herramienta se acabara planteando como un servicio comercializable, parece evidente que Barcelona sería un buen lugar, o mercado, en el que ofrecerlo debido al alto volumen de operaciones realizadas. Estas operaciones se llevan a cabo tanto por particulares como por empresas, por lo que se podría plantear un modelo de negocio tanto B2B como B2C. En concreto, para los agentes particulares compradores, se podría hacer uso de esta herramienta tanto si su intención fuera la de habitar la propiedad como la de invertir para alquilar o especular. Para los agentes jurídicos compradores, la herramienta podría emplearse como método para optimizar las inversiones (como fondos inmobiliarios o empresas patrimoniales) o para intermediar y asesorar a potenciales clientes que quisieran comprar o vender (como agencias inmobiliarias). En todo caso, para aquellos agentes cuya intención fuera comprar, esta herramienta les podría permitir detectar aquellos activos cuyo precio fuera menor que su valor en el mercado, resultando en una buena oportunidad de inversión. Para

los agentes cuya intención fuera vender, esta herramienta les podría brindar una perspectiva aproximada del valor que el mercado estaría dispuesto a ofrecer por el inmueble.

Una vez obtenido el valor de compraventa, el propietario (o potencial propietario) también podría predecir con mayor facilidad la rentabilidad del inmueble. Únicamente debería calcular el precio de alquiler anual y dividirlo por el valor de mercado (y multiplicado por cien si se expresara en porcentaje). Por ello, también es una buena herramienta para comparar rentabilidades y oportunidades de inversión que tuvieran esta perspectiva. De hecho, una posible extensión que desborda el alcance de este trabajo, pero que podría plantearse en un futuro, es realizar también una predicción del precio de alquiler de un inmueble y por tanto de la rentabilidad de este. Realizar esta extensión no representaría mucho más esfuerzo a nivel algorítmico, pero se deberían obtener datos sobre alquileres, lo que incrementaría notablemente el tamaño del conjunto de datos.

Por otro lado, desarrollar una herramienta con estas características representa una modernización dentro del asesoramiento inmobiliario. Tradicionalmente, en el sector inmobiliario español, se han empleado la experiencia y el conocimiento como principales herramientas de determinación del valor de mercado de los inmuebles en los momentos de compra y de venta. Estas habilidades recaen en agentes, principalmente inmobiliarios. Se trata de un conocimiento muy limitado a zonas geográficas, como son los barrios o distritos en la ciudad de Barcelona. De hecho, las agencias inmobiliarias se suelen repartir el mercado geográficamente, encontrando cada vez más competencia a la hora de obtener una porción de la cuota de mercado. En concreto, en el año 2019 la cantidad de agencias inmobiliarias en Barcelona se había duplicado respecto a 2015[4], siendo las registradas 2.115. Por ello, el uso de esta herramienta por parte de agencias inmobiliarias puede resultar muy ventajoso para diferenciarse de la competencia, aportando un método de valoración más objetivable, que no requiere de la experiencia o la habilidad humanas para realizar predicciones o determinar precios de compra y de venta. Adicionalmente, llevar a cabo esta transformación tecnológica puede aportar una mejora sustancial en la eficiencia, tanto de los agentes compradores y vendedores, de los intermediarios y en definitiva del mercado, a través de la reducción de la asimetría de información.

Asimismo, un uso primordial de esta herramienta sería el personal, motivo por el que se pensó aplicar los conocimientos técnicos al desarrollo de un instrumento que permitiera comparar opciones de inversión sin tener especial conocimiento del sector inmobiliario.

Disponerse a realizar una compra de una vivienda en Barcelona sin tener dichos conocimientos, y sin un asesoramiento que pueda considerarse confiable, resulta imprudente dado el alto coste de dichas operaciones. Por ello, llegado el momento de adquirir o vender una vivienda a título personal, esta herramienta podría ser de gran ayuda.

Por tanto, el punto de partida de este trabajo es la necesidad manifiesta que existe en el sector inmobiliario a la hora de utilizar un método objetivo y robusto de valoración de inmuebles dedicados a la vivienda. Se trata de un sector en el que existe un grado elevado de asimetría de información e ineficiencia, en el que dos agentes inmobiliarios pueden realizar predicciones dispares respecto a un mismo inmueble. Actualmente, no existe una herramienta extendida que resuelva este problema. En consecuencia, mejorar la eficiencia de este mercado tendría un impacto positivo notable para los agentes implicados en el mercado, así como en el conjunto de la sociedad. Entre los impactos más relevantes, se puede destacar: el aumento en la actividad económica (y la inversión) producido por el aumento en la liquidez, el equilibrio de precios más justo, la mejora de la estabilidad del mercado y por tanto el impulso del crecimiento económico y la mejora del bienestar de los consumidores.

1.2. Objetivos del Trabajo

El objetivo de este Trabajo de Fin de Grado es lograr predecir con un grado satisfactorio de ajuste el precio de una vivienda en particular en la ciudad de Barcelona, con unas características dadas concretas, a partir del uso de un algoritmo que emplee el aprendizaje automático nutriéndose de un conjunto de datos que cumpla los requisitos de información necesarios. Estos requisitos son: tener una cantidad significativa de datos, tener datos de alta calidad, ser representativa y diversa, estar etiquetada y, si fuera posible, tener un formato compatible.

Por tanto, es posible desglosar el objetivo global en metas específicas:

1. Definir las características o atributos de los inmuebles destinados a vivienda necesarios para desarrollar un algoritmo de aprendizaje automático que los emplee.

2. Obtener un conjunto de datos que cumpla los requisitos comentados previamente. Para ello, se plantean dos opciones empleables indistintamente:

    a. Emplear la API del portal inmobiliario, que permite obtener datos de distintos mercados, incluido el de Barcelona, mediante solicitudes a sus servidores.

    b. Emplear una herramienta de *web scraping* que permite obtener datos de las viviendas a la venta en el municipio de Barcelona de portales inmobiliarios.

3. Analizar y tratar los datos, si fuera necesario, para que pudieran ser utilizados por el algoritmo de aprendizaje automático.

4. Seleccionar el algoritmo de aprendizaje automático idóneo para un modelo de predicción de precios.

5. Desarrollar el algoritmo de aprendizaje automático mediante el uso de librerías de Python como sickit-learn.

6. Desarrollar, entrenar y evaluar los resultados del algoritmo, obteniendo métricas de ajuste.

7. Implementar una interfaz en la que el usuario pudiera interactuar y/o interpretar fácilmente los resultados obtenidos.

8. Ofrecer un ejemplo de uso y realizar un análisis que permita concluir si se puede dar por válido, o no, el algoritmo desarrollado.

Por tanto, queda fuera del alcance de esto modelo incluir información o predicción de viviendas en el ámbito geográfico que exceda el área municipal de la ciudad de Barcelona. Tampoco se calculará ni incluirá información de rentabilidad o alquileres de las viviendas.

1.3. Enfoque y método seguido

Puesto que en este trabajo se desarrolla un algoritmo nuevo, y por tanto un producto nuevo que emplea un conjunto de datos que deberá ser obtenido previamente, es necesario incluir dentro de la propia metodología del trabajo la metodología propia del desarrollo del algoritmo.

Por tanto, para llevar a cabo el trabajo se emplea la metodología PMBOK. Se trata de una metodología que puede considerarse más bien una guía de buenas prácticas que permite seguir el proyecto y sus etapas. Sin embargo, se pueden identificar 5 fases distintas que ayudan a organizar el desarrollo satisfactorio del trabajo:

1. Iniciación, en la que se definen en qué consiste el trabajo y se argumenta su razón de ser.

2. Planificación, en la que se detallan las necesidades materiales, como suplir dichas necesidades y los posibles riesgos.

3. Ejecución, en la que se implemente el trabajo.

4. Seguimiento y control, que se hace simultáneamente con la fase de ejecución.

5. Cierre, en la que se valoran los resultados obtenidos y el desarrollo del trabajo.

De manera gráfica, es posible ver las 5 fases que permiten desarrollar este trabajo.

Figura 1. Ciclo de vida de un proyecto desarrollado con PMBOK

Por otro lado, también es necesario desarrollar el algoritmo, por lo que se debe emplear una metodología que permita llevarlo a cabo satisfactoriamente. Parece adecuado emplear un enfoque de desarrollo en cascada, que permita realizar paso a paso cada fase del ciclo de vida del algoritmo. Además, se trata de un algoritmo cuyos pasos pueden ser estructurados claramente, donde el alcance y los requisitos están bien definidos. El consumo excesivo de tiempo o la escasez de agilidad no parecen ser problemáticos dada la poca complejidad del algoritmo.

Los pasos que se pueden seguir son los siguientes:

1. Determinar los atributos necesarios para el conjunto de datos que se empleará en el algoritmo de aprendizaje automático.

2. Recopilar y cargar los datos, ya sea mediante una API como con una herramienta de *web scraping*.

3. Procesar los datos, limpiándolos y preparándolos si fuera necesario.

4. Analizar los datos.

5. Seleccionar el algoritmo de aprendizaje automático (modelo) que se considere más adecuado para realizar una predicción de precios, y diseñar su implementación.

6. Entrenar el modelo, utilizando el conjunto de datos obtenido previamente y el algoritmo escogido.

7. Validar el modelo analizando su ajuste con las métricas que se consideren adecuadas.

8. Describir los resultados y las conclusiones obtenidas.

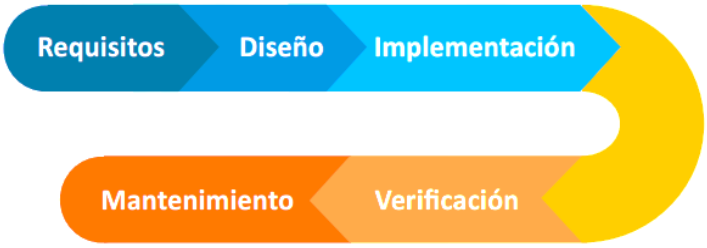


Figura 2. Ciclo de vida de un proyecto desarrollado en Cascada

1.4. Planificación del Trabajo

La planificación del trabajo consiste en las tareas principales, así como los entregables y todas las tareas que comprenden la realización de este proyecto de trabajo de fin de grado.

Se estima que se requieren un total de 275 horas de trabajo totales, dedicándose alrededor de 2 a 3 horas de trabajo diarias, en función de la necesidad y la disponibilidad. Cabe tener en cuenta que este trabajo será realizado de manera individual, por lo que la responsabilidad y carga de trabajo será también individual.

Respecto al hardware, el desarrollo de este trabajo se realizará íntegramente en un ordenador personal MAC con las siguientes características:

- Sistema operativo macOS Ventura.
- Procesador Apple M1 Pro.
- Memoria RAM de 16GB.
- Disco duro SSD de 500GB.

Respecto al software empleado se contempla utilizar, al menos, las siguientes tecnologías:

- Explorador Google Chrome.
- Extensión Web Scraper para Google Chrome.
- Extensión uBlock Origin.
- Python 3 versión 3.9.6.
- Visual Studio Code 1.76.0 (Universal).
- Microsoft Word 16.71.

El proyecto se debe llevar a cabo entre las fechas 1 de marzo de 2023 y 8 de junio de 2023, dejando al menos 10 días (o 25 horas) para posibles inconvenientes o desviaciones respecto a lo estimado hasta la fecha definitiva de entrega, que será el día 18 de junio de 2023.

Las actividades del proyecto a llevar a cabo, junto con su planificación temporal son las siguientes:

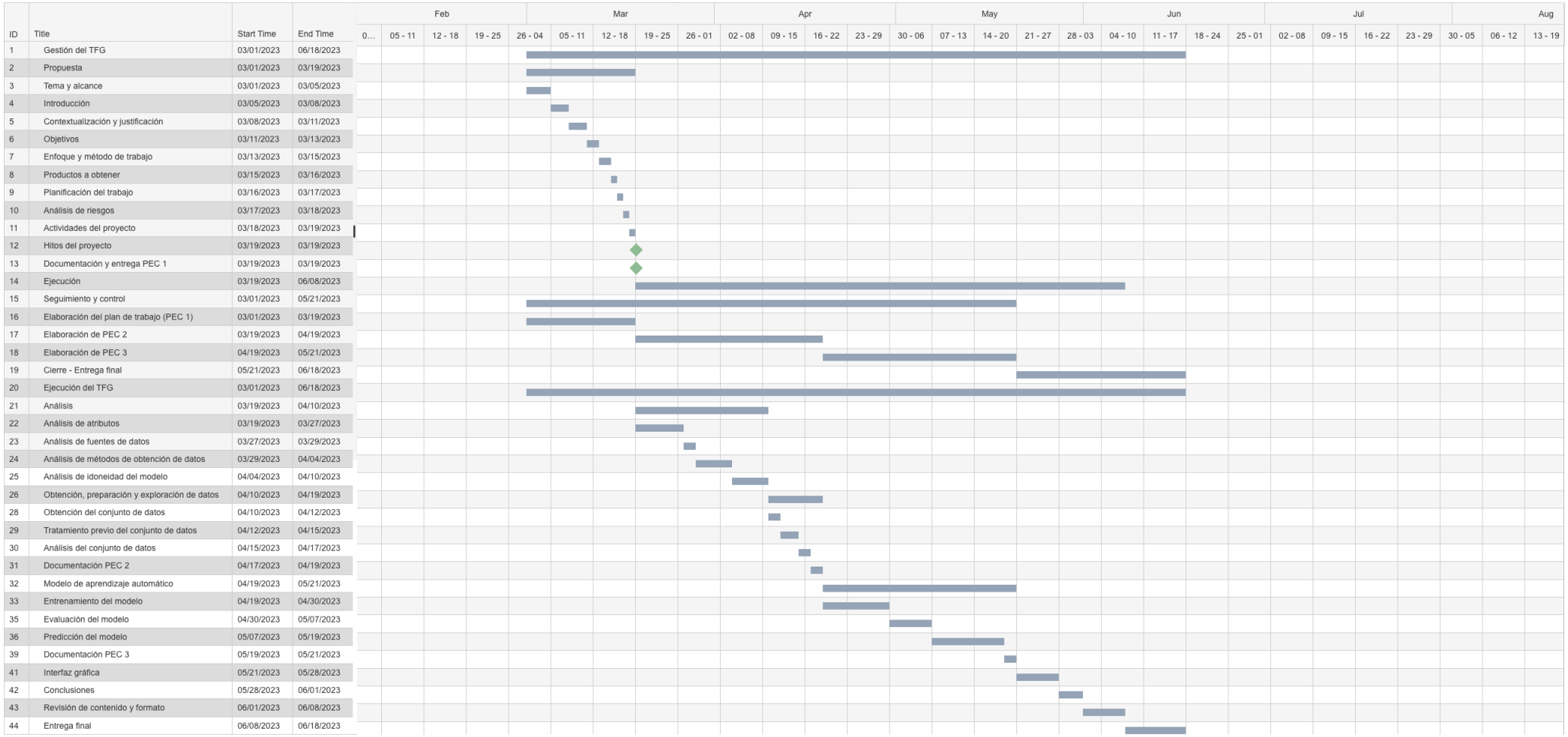
ID	Nombre	Fecha de finalización
1	Gestión del TFG	18/06/2023
2	Propuesta	19/03/2023
3	Tema y alcance	05/03/2023
4	Introducción	08/03/2023
5	Contextualización y justificación	11/03/2023
6	Objetivos	13/03/2023
7	Enfoque y método de trabajo	15/03/2023
8	Productos a obtener	16/03/2023
9	Planificación del trabajo	17/03/2023
10	Análisis de riesgos	18/03/2023
11	Actividades del proyecto	19/03/2023
12	Hitos del proyecto	19/03/2023
13	Documentación y entrega PEC 1	19/03/2023
14	Ejecución	18/06/2023
15	Seguimiento y control	21/05/2023
16	Elaboración del plan de trabajo (PEC 1)	19/03/2023
17	Elaboración de PEC 2	19/04/2023
18	Elaboración de PEC 3	21/05/2023
19	Cierre - Entrega final	18/06/2023
20	Ejecución del TFG	18/06/2023
21	Análisis	10/04/2023
22	Análisis de atributos	27/03/2023
23	Análisis de fuentes de datos	29/03/2023
24	Análisis de métodos de obtención de datos	04/04/2023

25	Análisis de idoneidad del modelo	10/04/2023
26	Obtención, preparación y exploración de datos	19/04/2023
27	Obtención del conjunto de datos	12/04/2023
28	Tratamiento previo del conjunto de datos	15/04/2023
29	Análisis del conjunto de datos	17/04/2023
30	Documentación PEC 2	19/04/2023
31	Modelo de aprendizaje automático	21/05/2023
32	Entrenamiento del modelo	30/04/2023
33	Evaluación del modelo	07/05/2023
34	Predicción del modelo	19/05/2023
35	Documentación PEC 3	21/05/2023
36	Interfaz gráfica	28/05/2023
37	Conclusiones	01/06/2023
38	Revisión de contenido y formato	08/06/2023
39	Entrega final	18/06/2023

En los hitos del proyecto, se definen aquellos considerados más importantes, como las entregas de las 3 PEC, junto con la entrega final. Además, se añaden los hitos considerados fundamentales para el desarrollo de este proyecto de trabajo de fin de grado:

Nombre	Fecha de hito	Tipo
Propuesta	19/03/2023	EJECUCIÓN
Elaboración del plan de trabajo (PEC 1)	19/03/2023	SEGUIMIENTO
Análisis de métodos de obtención de datos	04/04/2023	EJECUCIÓN
Análisis de idoneidad del modelo	10/04/2023	EJECUCIÓN
Elaboración de PEC 2	19/04/2023	SEGUIMIENTO
Entrenamiento del modelo	30/04/2023	EJECUCIÓN
Evaluación del modelo	07/05/2023	EJECUCIÓN
Predicción del modelo	19/05/2023	EJECUCIÓN
Elaboración de PEC 3	21/05/2023	SEGUIMIENTO
Interfaz gráfica	28/05/2023	EJECUCIÓN

Conclusiones	01/06/2023	EJECUCIÓN
Cierre - Entrega final	18/06/2023	EJECUCIÓN



ID	Title	Start Time	End Time
1	Gestión del TFG	03/01/2023	06/18/2023
2	Propuesta	03/01/2023	03/19/2023
3	Tema y alcance	03/01/2023	03/05/2023
4	Introducción	03/05/2023	03/08/2023
5	Contextualización y justificación	03/08/2023	03/11/2023
6	Objetivos	03/11/2023	03/13/2023
7	Enfoque y método de trabajo	03/13/2023	03/15/2023
8	Productos a obtener	03/15/2023	03/16/2023
9	Planificación del trabajo	03/16/2023	03/17/2023
10	Análisis de riesgos	03/17/2023	03/18/2023
11	Actividades del proyecto	03/18/2023	03/19/2023
12	Hitos del proyecto	03/19/2023	03/19/2023
13	Documentación y entrega PEC 1	03/19/2023	03/19/2023
14	Ejecución	03/19/2023	06/08/2023
15	Seguimiento y control	03/01/2023	05/21/2023
16	Elaboración del plan de trabajo (PEC 1)	03/01/2023	03/19/2023
17	Elaboración de PEC 2	03/19/2023	04/19/2023
18	Elaboración de PEC 3	04/19/2023	05/21/2023
19	Cierre - Entrega final	05/21/2023	06/18/2023
20	Ejecución del TFG	03/01/2023	06/18/2023
21	Análisis	03/19/2023	04/10/2023
22	Análisis de atributos	03/19/2023	03/27/2023
23	Análisis de fuentes de datos	03/27/2023	03/29/2023
24	Análisis de métodos de obtención de datos	03/29/2023	04/04/2023
25	Análisis de idoneidad del modelo	04/04/2023	04/10/2023
26	Obtención, preparación y exploración de datos	04/10/2023	04/19/2023
28	Obtención del conjunto de datos	04/10/2023	04/12/2023
29	Tratamiento previo del conjunto de datos	04/12/2023	04/15/2023
30	Análisis del conjunto de datos	04/15/2023	04/17/2023
31	Documentación PEC 2	04/17/2023	04/19/2023
32	Modelo de aprendizaje automático	04/19/2023	05/21/2023
33	Entrenamiento del modelo	04/19/2023	04/30/2023
35	Evaluación del modelo	04/30/2023	05/07/2023
36	Predicción del modelo	05/07/2023	05/19/2023
39	Documentación PEC 3	05/19/2023	05/21/2023
41	Interfaz gráfica	05/21/2023	05/28/2023
42	Conclusiones	05/28/2023	06/01/2023
43	Revisión de contenido y formato	06/01/2023	06/08/2023
44	Entrega final	06/08/2023	06/18/2023

Figura 3. Diagrama de Gantt

1.5. Breve sumario de productos obtenidos

De manera simplificada, los principales y diferentes productos obtenidos son los siguientes:

- Definición de los atributos considerados necesarios para poder realizar un algoritmo que estime los precios de viviendas en Barcelona de manera precisa.

- Análisis de idoneidad de la fuente de la que se obtiene el conjunto de datos, así como otras alternativas valoradas.

- Proceso de obtención del conjunto de datos, ya sea mediante una API o mediante una herramienta *web scraper*.

- Conjunto de datos que cumpla con los requisitos de información necesarios. Debe disponer de suficientes instancias (viviendas) cuyos atributos sean los establecidos previamente.

- Análisis de idoneidad en la selección del algoritmo de aprendizaje automático.

- Descripción del algoritmo desarrollado en Python, que emplea tanto el conjunto de datos como el algoritmo escogido, y que predice precios de viviendas en Barcelona. Este algoritmo debería incluir una interfaz que fuera comprensible para el usuario.

- Evaluación de los resultados obtenidos previamente.

- Ejemplo de uso del algoritmo implementado.

1.6. Breve descripción de los otros capítulos de la memoria

Los cuatro capítulos adicionales a la Introducción que componen esta memoria son Análisis, Obtención, preparación y exploración de datos, Modelo de aprendizaje automático,

Interfaz gráfica, Conclusiones, Glosario, Bibliografía y Anexos. Al margen de los capítulos finales de la memoria, los que pertenecen al propio desarrollo del proyecto son los de Análisis, Obtención, preparación y exploración de datos, Modelo de aprendizaje automático e Interfaz gráfica, que se describen a continuación con mayor detalle.

En primer lugar, en el capítulo de Análisis se estudian y descomponen los diferentes elementos necesarios previos a desarrollar propiamente el algoritmo que emplea aprendizaje automático para predecir la valoración de viviendas. Como primer paso, se realiza un análisis describiendo aquellos atributos que determinan las características de una vivienda y que deberían ser empleados en el modelo. Posteriormente, se realiza un estudio de criterios necesarios para el conjunto de datos, los tipos de fuente desde donde obtenerlos, y de las motivaciones por las que se ha escogido el portal inmobiliario Idealista finalmente. También se detallan los dos métodos posibles para obtener el conjunto de datos necesario, así como el motivo por el que finalmente se escoge el uso de un *web scraper*. Finalmente, se analiza la idoneidad del algoritmo de aprendizaje automático utilizado para predecir los valores de mercado de viviendas, en comparación con otros métodos. Adicionalmente, se detalla cuál es el funcionamiento de dicho algoritmo de predicción.

En el capítulo de Obtención, preparación y exploración de datos de esta memoria se detalla cómo se desarrolla el proceso de obtención del conjunto de datos, así como la ejecución de dicha obtención y las dificultades encontradas. También se comentan los diferentes procesos que se deben llevar a cabo para poder emplear el conjunto de datos con el algoritmo de aprendizaje automático escogido. Se realiza al final de este capítulo un análisis de los datos obtenidos.

Posteriormente, se comenta respecto a al Modelo de aprendizaje automático. En concreto, se desarrolla y detalla el funcionamiento de la parte del algoritmo que debe entrenarse utilizando el conjunto de los datos. Por ello, se describe la optimización del modelo de aprendizaje automático y se realiza su evaluación, detallando cómo se calculan las métricas que permiten posteriormente interpretar su grado de ajuste. Posteriormente, se desarrolla la parte de predicción del modelo de aprendizaje automático.

Una vez finalizada la parte del Modelo de aprendizaje automático, se desarrolla la Interfaz gráfica, que permite al usuario un uso más sencillo e intuitivo. Se describen finalmente los resultados obtenidos, así como las conclusiones.

2. Análisis

En el segundo capítulo se realizan distintos análisis necesarios para obtener el conjunto de datos e implementar el modelo de aprendizaje automático. Estos incluyen un análisis de los atributos empleados en el modelo, de las distintas fuentes desde las que obtener los datos, de los métodos para recabarlos y de la idoneidad del modelo de aprendizaje automático utilizado.

2.1. Análisis de atributos

Para llevar a cabo un modelo de aprendizaje automático que permita predecir el precio de una vivienda se deben emplear aquellos atributos o características que determinan el valor del inmueble. Estos atributos deben servir para que el propio modelo pueda aprender respecto al efecto que tiene cada uno de ellos en la determinación del valor de mercado, y así, pasándole los valores de un hipotético o real inmueble, predecir su valor.

Los atributos habituales que se suelen considerar en el mercado inmobiliario son aquellos que describen las características físicas y de localización del inmueble. Además, suelen estar disponibles en la mayoría de los portales inmobiliarios. Por ello, la lista de atributos que a priori se consideran puede determinar el valor de una vivienda, o pueden ser útiles para ser incluidos en el modelo, se detallarán a continuación. Cabe destacar que, aquellos que se emplean, tienen asociado el nombre de la característica usado en el modelo:

- **URL** (*url*): Aunque este atributo no está directamente relacionado con el valor de un inmueble ni su determinación del precio, puede resultar de gran utilidad para identificar la fuente de los datos (el sitio *web* desde el que se obtuvieron los datos y la fecha de obtención). Puede resultar de utilidad a la hora de limpiar y analizar los datos, pero no se emplea en el modelo de aprendizaje.

- **Barrio** (*neighborhood*): La ubicación o localización geográfica es un factor crucial en la determinación del valor de un inmueble. Esta información está estrechamente relacionada con las características de la ubicación como la seguridad, conectividad a la red de transporte público, zonas verdes, hospitales, colegios, etc. En definitiva,

características relacionadas con la calidad y comodidad de la zona.

- **Tipo de propiedad** (*property_type*): Los diferentes tipos de propiedad tienen diferentes rangos de precios y niveles de demanda, que determinan en gran medida el valor de un inmueble. Por ejemplo y por lo general, una vivienda unifamiliar tiene mayor valor que una vivienda en un bloque de pisos, dadas el resto de las características iguales.

- **Precio** (*price*): Se trata de la variable objetivo del modelo, el precio real de la propiedad. El modelo utilizará este atributo para poder aprender cómo afectan el resto de las características. Con una cantidad suficiente de viviendas y sus precios de venta, el modelo podrá determinar cuánto afecta cada atributo a dicha variable.

- **Superficie** (*surface*): La superficie de un inmueble es uno de los principales factores que determinan el valor de un inmueble. Cuanto mayor es la superficie de una vivienda, mayor es su valor.

- **Habitaciones** (*rooms*): La cantidad de habitaciones de un inmueble está estrechamente relacionada con el atributo de superficie, puesto que un mayor número de habitaciones está relacionado con una propiedad más grande y espaciosa.

- **Baños** (*bathrooms*): Al igual que en el caso del atributo habitaciones, la cantidad de baños de un inmueble está estrechamente relacionada con el atributo superficie, puesto que un mayor número de baños está relacionado con una propiedad más grande y espaciosa. Además, la proporción entre cantidad de baños y cantidad de habitaciones adquiere una relevancia especial, puesto que las viviendas que tienen una ratio superior suelen ser mejor valoradas en el mercado; manteniendo el resto de las características igual, es más valiosos tener dos baños para cuatro habitaciones que tener uno solamente.

- **Año de construcción** (*year_built*): El año de construcción de una propiedad influye directamente en el valor de la vivienda. Por lo general, los inmuebles con mayor año

de construcción (los más recientes) requieren de un menor mantenimiento que las más antiguas, lo que puede aumentar su valor. Sin embargo, se pueden dar casuísticas particulares que no cumplan estrictamente esta lógica. En el caso de Barcelona, entre los años 1950 y 1970 se construyeron una gran cantidad de pisos (entre 35.000 y 40.000), en una treintena de barrios, que padecían o padecen aluminosis[5]. Este hecho devalúa directamente a los inmuebles afectados, puesto que requieren una inversión en mantenimiento y reparación. El modelo debe poder identificar este tipo de circunstancias si resulta tener un efecto significativo.

- **Estado** (*condition*): Cuanto mejor sea el estado de un inmueble, mayor será su valor. Aquellas viviendas que estén reformadas y en mejores condiciones serán mejor valoradas, manteniendo el resto de los atributos iguales, que aquellas no lo estén.

- **Planta** (*floor*): Las viviendas que están situadas en plantas más altas suelen disponer de mejores vistas, por lo que están mejor valoradas que las que se encuentran más próximas al nivel de la calle.

- **Exterior** (*is_exterior*): Este atributo indica de manera binaria si un inmueble da a la calle o bien a un patio interior. Las propiedades que dan al exterior suelen estar mejor valoradas que aquellas que dan a patios interiores.

- **Precio del parquin** (*parking_price*): Muchas propiedades en la ciudad de Barcelona se ofertan junto con una plaza de parking, que suele venderse a parte. Disponer de plaza de parquin, y el precio de dicha plaza, influye directamente en el valor de la propiedad.

- **Balcón** (*has_balcony*): Las viviendas que disponen de balcón suelen estar mejor valoradas en el mercado que aquellas que no tienen.

- **Terraza** (*has_terrace*): Al igual que en el caso del balcón, las viviendas que disponen de terraza suelen estar mejor valoradas en el mercado que aquellas que no lo tienen. De hecho, desde la pandemia por el COVID-19 se ha producido un aumento en la

demanda, y por tanto del precio, de aquellas viviendas que tienen balcón o terraza[6].

- **Jardín** (*has_garden*): De modo similar a lo que ocurre con balcones y terrazas, disponer de jardín suele ser un elemento que aprecia las viviendas.

- **Zonas verdes** (*has_green_zones*): Al igual que en el caso de los atributos balcón, terraza y jardín, las zonas verdes comunitarias son un elemento que aprecian el valor de las viviendas.

- **Piscina** (*has_pool*): Por lo general, aquellas viviendas que posean piscina, ya sea comunitaria o privada, suelen ser más valoradas. Este hecho está relacionado con otras características como superficie, año de construcción, barrio o tipo de propiedad.

- **Trastero** (*has_storage_room*): La característica de disponer de trastero o armarios en algún lugar de la finca distinto a la propia vivienda suele apreciar la propiedad. Esto es debido a que la disponibilidad de trastero amplía el espacio total disponible de una vivienda; aquello que se puede almacenar en un trastero no ocupa el lugar dentro de la vivienda.

- **Ascensor** (*has_elevator*): Disponer de ascensor que permita acceder más fácilmente a la vivienda suele apreciar la propiedad, dado que reduce el esfuerzo y hace más cómodo el acceso, especialmente en pisos cuya planta es alta.

- **Armarios empotrados** (*has_fitted_wardrobes*): Aquellas viviendas que disponen de armarios empotrados suelen tener un valor mayor, dado que realizan un uso más eficiente maximizando el espacio de almacenamiento de manera estética.

- **Aire acondicionado** (*has_ac*): Disponer de aire acondicionado aprecia las viviendas, dado que disponen de un sistema habitualmente conveniente, o incluso imprescindible, que en caso contrario debería instalarse, incurriendo en un coste de compra e instalación.

- **Orientación norte** (*faces_north*): La orientación norte de una vivienda implica que la incidencia del sol es muy baja en el hemisferio donde se encuentra Barcelona. El valor de las viviendas con esta orientación es menor que con otras orientaciones; se requiere un mayor consumo energético y se dispone de menor luminosidad.

- **Orientación este** (*faces_east*): La orientación este de una vivienda implica que la incidencia del sol será notable hasta el mediodía en el hemisferio en el que se encuentra Barcelona. Esto hace que el valor de las viviendas con esta orientación sea mayor que con otras orientaciones, puesto que se requiere un menor consumo energético y se dispone de mayor luminosidad.

- **Orientación sur** (*faces_south*): La orientación sur de una vivienda implica que la incidencia del sol será notable durante todo el día en el hemisferio en el que se encuentra Barcelona. Esto hace que el valor de las viviendas con esta orientación sea mayor que con una orientación norte, puesto que se requiere un menor consumo energético, salvo en verano que será superior, y se dispone de mayor luminosidad.

- **Orientación oeste** (*faces_west*): La orientación oeste de una vivienda implica que la incidencia del sol será notable a partir del mediodía en el hemisferio en el que se encuentra Barcelona. Esto hace que el valor de las viviendas con esta orientación sea mayor que con orientaciones norte y sur, puesto que se requiere un menor consumo energético y se dispone de mayor luminosidad.

- **Eficiencia energética** (*energy_consumption*): La eficiencia energética de un inmueble tiene un impacto directo en su valor, debido a que una mayor eficiencia representa un ahorro en el consumo energético y por tanto un ahorro económico. Esta característica está estrechamente relacionada con el año de construcción y el estado del inmueble; los inmuebles reformados y de reciente construcción disponen de mejores calificaciones energéticas.

- **Calefacción** (*heating*): Aquellas viviendas que cuentan con un sistema de calefacción individual o centralizado suelen ser más eficientes, por lo que requieren de un menor

gasto energético. Ello hace que su valor sea superior en relación con viviendas sin sistema de calefacción.

- **Accesibilidad** (*accessibility*): Las viviendas que disponen de una adaptación para personas con movilidad reducida, en el interior de la vivienda y/o en el acceso exterior, suelen estar mejor valoradas. Dicha adaptación implica un ahorro en una eventual necesidad de adaptar la vivienda, tanto en elementos privados como comunitarios si los hubiera.

- **Gastos de IBI (y Comunidad)**: El impuesto sobre bienes inmuebles (IBI), y en algunos casos también el gasto de comunidad, representan una parte de los gastos fijos de un inmueble. Dichos gastos suelen ser un importe fijo, están estrechamente relacionados con la superficie del inmueble y suele afectar negativamente al precio de la vivienda. Sin embargo, se trata de una información de difícil acceso; no suele ofrecerse públicamente. Por esa razón, no se puede incluir definitivamente en los atributos que se incluyen en el modelo.

2.2. Análisis de fuentes de datos

Para poder llevar a cabo el modelo de aprendizaje automático que permite predecir el valor de un inmueble es necesario disponer de un conjunto de datos adecuado. El conjunto de datos debe cumplir ciertos criterios que permitan emplearlo satisfactoriamente en el modelo. Entre los criterios necesarios, se deben destacar los siguientes:

- Precisión: Los datos deben ser precisos, y deben contener el mínimo número de errores e inconsistencias posible.

- Relevancia: Los datos deben contener los atributos la mayor parte de los atributos descritos anteriormente, puesto que son aquellos que tienen un impacto significativo en el valor de las viviendas.

- Integridad: El conjunto de datos debe estar completo y ser representativo, minimizando los valores ausentes y extremos, con tal de evitar sesgos o errores en las predicciones del modelo.

- Consistencia: Los datos deben ser consistentes, sin valores atípicos significativos o anomalías que puedan sesgar los resultados.

- Tamaño: El conjunto de datos debe ser lo suficientemente grande como para proporcionar un tamaño muestral que permita al modelo aprender y hacer predicciones.

- Actualización: Los datos deben estar actualizados, de manera que la valoración que realice el modelo incluya datos recientes y pueda tener en cuenta tendencias y situaciones actuales del mercado.

- Diversidad: Se debe disponer de información diversa para cada uno de los atributos, de manera que se pueda garantizar que el modelo pondera adecuadamente cada característica que pueda determinar el precio de una vivienda.

- Variable objetivo: El conjunto de datos debe tener la variable objetivo, el precio del inmueble, bien definida para entrenar el modelo y predecir dicha variable.

Una vez definidos los criterios anteriores, se plantean dos posibles vías a partir de las que obtener un conjunto de datos que satisfaga las necesidades establecidas. Se debe encontrar información de inmuebles que se hayan vendido recientemente, o bien que estén a la venta, de manera que sea posible conocer su precio de venta (o de oferta) y sus características, ya definidas. Por un lado, se plantea la opción de emplear datos obtenidos de bases de datos abiertas, y por otro, la de obtener la información directamente desde portales inmobiliarios.

Una primera base de datos que se consulta es la del Consejo General del Notario[7]. En ella es posible consultar estadísticas completas sobre actos notariales realizados históricamente. Entre dichos actos se pueden hallar la cantidad de compraventas realizadas a

lo largo del tiempo. Sin embargo, únicamente es posible ver la evolución del número de viviendas transmitidas, pero no acceder a la información del precio de cada una de las transmisiones ni de las características del inmueble en cuestión. De hecho, no se puede discriminar entre tipos de inmuebles. Por tanto, esta base de datos es descartada.

Por otro lado, se acude al Instituto Nacional de Estadística (INE)[8], que suele disponer de bases de datos extensas y precisas. En el apartado sobre datos económicos, es posible acceder a información estadística relativa a las transmisiones de derechos de propiedad. En este caso, se puede acceder a información que puede ser filtrada por comunidades autónomas y provincias y que es relativa a viviendas transmitidas, específicamente. Al acceder, es posible seleccionar datos recientes, del primer mes del año en curso, pero la información disponible es únicamente sobre el total de viviendas transmitidas según título de adquisición. Como en el caso del Consejo General del Notario, los datos ofrecidos no cumplen los requisitos necesarios.

Finalmente, la última base de datos a la que se acude es la que ofrece el Ayuntamiento de Barcelona, llamada Open Data BCN[9]. Se trata de una iniciativa del propio consistorio cuyo objetivo es ofrecer datos al público y fomentar su utilización por parte de la ciudadanía y empresas, de manera que puedan crear valor mejorando la calidad de vida de los ciudadanos de Barcelona. Un aspecto muy positivo de esta base de datos es que los datos son confiables, puesto que, aunque su obtención no se ha realizado necesariamente por una institución pública, sí existe una revisión y supervisión de su validez. Sin embargo, tras una exhaustiva búsqueda sobre las ventas de inmuebles en Barcelona, la información encontrada es escasa. Se pueden encontrar compraventas realizadas en la ciudad, pudiendo discriminar por barrio, por distrito y por precio, y otras predicciones de precios de venta que no son de utilidad. Sin embargo, esa es toda la información disponible, lo que resulta escaso para las necesidades del modelo. Además, se trata de datos que no se actualizan desde hace un año aproximadamente, por lo que no pueden considerarse útiles; el mercado de inmuebles dedicados a vivienda puede haber sufrido muchas variaciones en el transcurso de un año.

Al descartar las bases de datos abiertas, se plantea la opción de obtener la información a partir de portales inmobiliarios. En ellos, se encuentran una gran cantidad de inmuebles a la venta, y se suelen incluir características detalladas, así como fotografías. Los dos elementos diferenciales al acudir a esta fuente de datos respecto a las bases de datos abiertas son, por un lado, la dificultad que conlleva obtener el conjunto de datos de forma sencilla y en el formato

adecuado, y por otro, la posible diferencia que puede existir entre el precio de oferta y el precio final de venta. Respecto al primer inconveniente, se plantean dos alternativas para solventarlo, o bien usar una API propia del portal o bien utilizar un *web scraper*. Respecto al segundo inconveniente, se asume que, dada la fase del mercado de alta demanda, la diferencia entre precio de oferta y precio final de venta es negligible, especialmente si se obtiene un tamaño significativo de datos. Por otro lado, no es posible obtener datos reales de compraventa junto con los atributos del inmueble. Por tanto, se deben valorar distintos portales inmobiliarios.

En primer lugar, se comprueba el portal inmobiliario Habitaclia[10], uno de los portales que recibe más visitas mensualmente. Tras realizar un filtrado geográfico y delimitar el área a la ciudad de Barcelona, se comprueba que hay alrededor de 18.000 anuncio publicados, por lo que se puede considerar una cantidad significativa de viviendas. Al acceder a algunos anuncios, se comprueba que se ofrece información básica sobre la superficie, el número de baños y habitaciones, ciertas características generales y sobre el equipamiento comunitario. Sin embargo, no parecen ser suficientes atributos para poder ser empleados en el modelo de aprendizaje automático. Además, cierta información de la vivienda se encuentra en el propio texto plano del anuncio, por lo que sería difícil de extraer. Por otro lado, este portal no parece disponer de una API que permita acceder directamente a la información de los distintos inmuebles de manera rápida. Tampoco es posible descargar información resumida en el formato adecuado.

A continuación, se realiza un análisis sobre el portal inmobiliario Fotocasa[11]. Es el segundo portal con más visitas mensuales. Para la ciudad de Barcelona, también se pueden encontrar alrededor de 18.000 anuncios publicados. Se trata pues de una cantidad significativa de viviendas ofertadas. Los anuncios contienen más información que en el caso de Habitaclia. Al margen de superficie, número de baños y habitaciones, también se encuentra una sección en cada anuncio con las características adicionales. Estas características no se encuentran en el texto del anuncio, sino en un formato que permitiría obtenerlo más fácilmente que en el caso anterior. Entre los atributos adicionales se encuentra la antigüedad, el estado del inmueble, el tipo, si tiene parquin, ascensor, si está amueblado, la planta, la orientación y su eficiencia energética. Parece por tanto una mejor opción, aunque tampoco dispone de API ni permite descargar la información global en un formato adecuado.

Finalmente, se comprueba el portal inmobiliario con más visitas mensuales, Idealista[12]. Tras realizar el filtrado geográfico, delimitando el área de la ciudad de Barcelona, se aprecia

que se ofrecen alrededor de 16.000 viviendas. Aunque se trata de una cantidad menor que en los portales vistos anteriormente, sigue siendo significativa para las necesidades del modelo a realizar. También se divide la ciudad en Distritos, y la información relativa a cada vivienda en venta es mucho más detallada. En este caso, es posible encontrar la práctica totalidad de atributos definidos previamente (salvo IBI y gastos de comunidad) distribuidos en distintas viviendas. Dichos atributos se distribuyen en cuatro grupos dentro de cada anuncio: características básicas, edificio, equipamiento y certificado energético. Por lo general, los anuncios están completados con más información que en el resto de los portales analizados. Además, la información relativa a cada inmueble está desglosada fuera del texto del comentario del anunciante para la mayoría de los anuncios, lo que permitiría recabarla más fácil y eficientemente. Un elemento muy positivo de Idealista es que dispone de una API (API de testigos[13]), que se detallará más adelante. Si no fuera posible el uso de esta herramienta, Idealista cuenta con la información más detallada y mejor distribuida, lo que facilita el uso de un *web scraper*. Por tanto, parece claro que es este el portal más indicado a partir del cual obtener el conjunto de datos que se empelarán posteriormente en el modelo.

2.3. Análisis de métodos de obtención de datos

Se plantean dos métodos principales de obtención del conjunto de datos a partir del portal inmobiliario escogido. El primero consiste en emplear la API ofrecida por Idealista. El segundo implica el uso de un software de *web scraping* que permita obtener la información desde las distintas páginas del sitio web. Se analiza en mayor detalle cada uno de estos dos métodos.

Una API, o *Application Programming Interfaces*, es un conjunto de protocolos y definiciones empleado con el fin de desarrollar e integrar software de aplicaciones, permitiendo la comunicación entre las mismas. De esta manera, y en el caso concreto de este proyecto, sería posible acceder a información concreta sobre anuncios de Idealista sin necesidad de acceder a su código, pudiendo realizar ciertas funciones sin tener conocimiento del funcionamiento interno del portal. Idealista permite realizar una consulta de datos al instante, en tiempo real y actualizados cada 24 horas. Con ella se puede obtener información de un conjunto de inmuebles, tanto en venta como en alquiler. Por otro lado, también están disponibles testigos de precios de transacciones y testigos históricos del portal. En la petición

se puede incluir una coordenada y las características del inmueble, así como la tipología. En concreto, se puede filtrar por tipo de operación, tipología y subtipología, superficie y el resto de los atributos definidos por Idealista. La API, tras la petición, devolvería 20 anuncios que cumplieran con los filtros establecidos, aportando toda la información disponible sobre dichos anuncios. También se aportarían los datos catastrales y la distancia a la localización establecida. Por tanto, este método permitiría realizar una serie de peticiones que devolverían datos sobre inmuebles en lotes de 20. Automatizar dichas peticiones posibilitaría obtener un conjunto de datos suficientemente significativo, siendo más eficiente que emplear el software de *web scraping* directamente, y lo más recientes posible. Alternativamente, Idealista cuenta con otra API de búsqueda que permite integrar la información de la propiedad publicada en otro sitio web o aplicación. De esta manera, también se podrían hacer peticiones automatizadas que facilitaran generar un conjunto de datos con la información necesaria: todas las viviendas publicadas en el portal que se encontraran en la ciudad de Barcelona, así como sus características. Para ello, a finales del mes de marzo de 2023 se contactó con el portal Idealista, rellenando el formulario habilitado para solicitar la clave API y describiendo el proyecto que justificara su uso. Sin embargo, hasta la fecha no se ha recibido ninguna respuesta al respecto. Por ello, pese a resultar la opción preferente, es necesario recurrir al método de *web scraping*.

El segundo método implica utilizar un sistema de *web scraping*, que consiste en emplear programas de software para extraer información de sitios web. En este caso, sería conveniente que el software que se utilizara realizara una navegación por todos los anuncios de viviendas a la venta en la ciudad de Barcelona listados en el portal inmobiliario de Idealista, y fuera extrayendo la información de cada uno listándolo en el formato adecuado. Una vez esta información fuera obtenida, poder exportar un fichero de tipo .CSV a partir del cual poder realizar el tratamiento adecuado de los datos para su inclusión en el modelo. Utilizando este tipo de software se evita tener que desarrollar código específico para extraer datos, así como realizar los cambios al código en caso de que el portal desde el que se obtuvieran los datos cambiara.

Para llevar a cabo este sistema para recopilar el conjunto de datos que se necesita emplear en el modelo de aprendizaje automático, se ha decidido utilizar Web Scraper[14], una extensión para distintos navegadores web. En este proyecto, se utiliza la extensión para Google Chrome. Se trata de una extensión gratuita, con una interfaz fácilmente configurable, que

permite incluso programar el *scraping* (u obtención de datos). Además, es posible extraer datos de sitios web que emplean JavaScript, que, si bien facilita la usabilidad de la interfaz, perjudica la simplicidad para realizar *scraping*. Tras crear y ejecutar el *scraper*, se pueden exportar los datos en formato .CSV desde el navegador directamente, y dispone del servicio Web Scraper Cloud que permite exportar datos en formatos .CSV, .XLSX y .JSON, acceder a ellos a través de API, *webhooks* o exportarlos a través de Dropbox, Google Sheets o Amazon S3. La configuración y uso de esta extensión se detallará en el capítulo de Implementación. Por tanto, los criterios para escoger esta extensión son la facilidad de uso, los formatos de datos soportados, el rendimiento, la velocidad y el hecho de que es gratuito. Otra extensión que se ha valorado ha sido Scraper[15], pero las funcionalidades resultan más sencillas y es poco potente para las necesidades de este proyecto. Data Scraper[16] es otra alternativa valorada positivamente, pero es gratuita hasta cierto límite de uso mensual.

2.4. Análisis de idoneidad del modelo

Una vez obtenido el conjunto de datos, se debe tratar para adecuarlo al modelo de aprendizaje automático. A continuación, es necesario seleccionar el modelo de aprendizaje automático que pueda permitir determinar el valor de una vivienda dados los atributos especificados en el primer apartado de este capítulo. Para ello, se emplean dos perspectivas que podrán ayudar a dilucidarlo. Por un lado, convendrá establecer y valorar los criterios que emplear para determinar qué modelo se ajusta a las necesidades de este proyecto. Por otro lado, será necesario realizar un breve análisis bibliográfico y comprobar aquellos métodos que han sido utilizadas en proyectos similares.

En primer lugar, se deben establecer aquellos criterios que permitan escoger un modelo de aprendizaje automático u otro. Los más destacables son los siguientes:

- Tipo de problema.

- Tamaño y calidad del conjunto de datos del que se dispone.

- Escalabilidad del modelo: La capacidad de poder gestionar un gran volumen de datos y que éstos puedan ser actualizados fácilmente.

- Interpretabilidad: La propiedad de que el modelo sea fácilmente comprensible, cómo funciona y cómo se obtiene la predicción.

- Precisión: El porcentaje de aciertos respecto al total de predicciones.

- Desempeño: El tiempo que tarda el modelo en procesar los datos y realizar las predicciones.

- Flexibilidad: La capacidad de que el modelo se adapte a distintos tipos de datos y patrones.

En el caso de este proyecto, puesto que se pretende predecir el valor de un inmueble, el tipo de problema será una regresión, por lo que el modelo deberá ser predictivo de regresión. El tamaño del conjunto de datos es sustancial, así que el modelo debe poder gestionar un volumen significativo de datos. Por otro lado, es poco probable que el conjunto de datos contenga datos ausentes, pero sí que incluya valores atípicos. En este sentido, el modelo debe ser robusto. En cuanto a la interpretabilidad, no es estrictamente necesario que el modelo sea interpretable, puesto que se debe primar la precisión de los resultados. La precisión debe ser la más alta posible, y convendría disponer de otras métricas que permitieran evaluarlo adecuadamente. Dado el volumen del conjunto de datos, conviene que el modelo sea eficiente en términos de procesamiento, pudiendo ser implementado en grandes conjuntos de datos y ofreciendo un buen desempeño. También es importante que el modelo pueda detectar patrones que sirvan para mejorar la predicción del valor de la vivienda.

Cabe mencionar que el modelo a emplear debe ser un modelo de predicción, puesto que el proceso que se debe llevar a cabo implica realizar una conjetura basada en datos muestrales sobre futuros resultados (probabilidad anterior a la ocurrencia del evento). Sin embargo, la estimación implicaría realizar un modelo que calculara un valor para un parámetro desconocido en una población utilizando datos muestrales (probabilidad posterior a la ocurrencia del evento). Por otro lado, en el transcurso de este trabajo se podría haber

aludido al término de inteligencia artificial para referirse al modelo, aunque resulta más preciso emplear el término aprendizaje automático, puesto que el modelo formará parte de este subconjunto dentro de la inteligencia artificial. Esta especialidad se centra en la capacidad de los sistemas para aprender y mejorar de forma autónoma a partir de un conjunto de datos. En concreto, este trabajo está recogido en la rama del aprendizaje automático relativa al aprendizaje supervisado, puesto que el modelo se entrena con datos de entrada y datos de salida deseados.

Una vez detalladas las características que se deben cumplir en este proyecto, es posible delimitar aquellos modelos que las cumplen. Entre ellos, cabe destacar la regresión lineal y sus variantes, que, si bien son simples y de fácil comprensión, pueden gestionar rápidamente un volumen grande de datos. Por el contrario, en ocasiones la detección de patrones es limitada. Las redes neuronales se plantean como otra opción, dado que son modelos no lineales que pueden lograr precisiones altas y detectar patrones considerados complejos. El principal inconveniente de estos tipos de modelos es el consumo de tiempo y recursos relativamente superior a otros modelos. Finalmente, la opción que parece más adecuada es alguna variante de un modelo de árboles de decisión. Son modelos no lineales capaces de gestionar un gran volumen de datos de manera eficiente y de identificar valores atípicos. El principal inconveniente es que se pueden producir errores por sobreajuste.

Si se realiza un análisis bibliográfico de trabajos o investigaciones similares al propio, es posible comprobar que el modelo XGBoost es ampliamente utilizado para predecir valores como el de la vivienda. En el estudio llevado a cabo por Zhao, Chetty y Tran (2019)[17], examinan el uso de aprendizaje profundo (*deep learning*) combinado con *extreme gradient boosting* (XGBoost) para la tasación de inmuebles, analizando los registros históricos de ventas junto con contenido visual, imágenes de casas y calificaciones de cada imagen desde un punto de vista estético. Los resultados muestran una mejora en la precisión de la predicción del precio de la vivienda al reemplazar la última capa de salida con XGBoost. Stang, Krämer, Nagl et al. (2022)[18] realizaron un estudio en el que se aplica una automatización del método de comparación de ventas mediante el uso de filtros y funciones de similitud, dos funciones de precios hedónicos (un modelo OLS y un modelo GAM), así como un enfoque de aprendizaje automático XGBoost, a un conjunto de datos de 1,2 millones de viviendas en Alemania. En los

resultados se concluye que el método de aprendizaje automático XGBoost ofrece el mejor desempeño general con respecto a la precisión de las estimaciones. En el artículo publicado por Evert, Erwin y Marten (2022)[19], se detalla cómo se implementan y comparan tres modelos de precios hedónicos (regresión lineal, regresión ponderada geográficamente y XGBoost) para modelar valores de tasación de inmuebles para cinco grandes municipios en diferentes zonas de los Países Bajos. Se concluye que, de los tres modelos implementados, el modelo XGBoost tiene la mayor precisión.

Por otro lado, si se revisan competiciones de Kaggle[20], es posible comprobar que en muchas de ellas se emplea el modelo de aprendizaje automático XGBoost para realizar tareas similares a este proyecto. Las competiciones de Kaggle son tareas de aprendizaje automático organizadas por Kaggle y otras empresas u organismos, como Google. Estas competiciones suelen ofrecer un premio económico, y varían en cuanto a tipología de problema y complejidad. Durante el año 2015, 17 de las 29 soluciones ganadores en competiciones de Kaggle usaron el modelo XGBoost[21].

Por tanto, conviene explicar en qué consiste el modelo XGBoost (*Extreme Gradient Boosting*). Se trata de una variante de un modelo de aprendizaje automático de árboles de decisión, desarrollado por Chen y Guestrin (2016)[22] en la Universidad de Washington.

En origen, este modelo consiste en un árbol de decisión, una estructura en forma de árbol que representa decisiones y consecuencias. Los nodos representan las decisiones y las ramas representan sus consecuencias. El nodo raíz simboliza la decisión inicial a tomar, que suele venir determinado por el atributo (o variable) que más efecto tiene en la variable objetivo a predecir, la que separa mejor los datos en función de dicha variable. Tras el nodo raíz, cada nodo representa una pregunta que se debe responder para llegar a una decisión, y cada respuesta configura una rama que delimita el camino hasta el nodo final, la acción a tomar. Tras realizar el proceso de separar el conjunto de datos en la raíz, se va repitiendo el proceso en los sucesivos subgrupos que se van generando hasta alcanza un criterio de parada, como un tamaño máximo de niveles, o profundidad. Las características o atributos que van separando los datos en función de la variable objetivo, el precio en este caso, se van escogiendo mediante criterios de selección como la ganancia de información o la medida de impureza de Gini. Una vez construido el árbol, éste se utiliza para predecir la variable objetivo que se ha establecido

en el modelo mediante el recorrido del camino desde la raíz hasta el nodo terminal correspondiente a la predicción.

XGBoost combina varios modelos más simples para obtener una mejor precisión en la predicción de los resultados en un modelo más complejo. En concreto, combina árboles de decisión. El proceso de *boosting*, o de impulso, consiste en entrenar un árbol de decisión simple con datos de entrenamiento y en identificar aquellos errores que el árbol no puede predecir correctamente. En el siguiente árbol se les da más peso a dichos errores, y se repite el proceso con árboles sucesivos hasta alcanzar un límite predefinido, como la mejora de precisión deseada del modelo. Además, en este modelo se utiliza la técnica de gradiente descendiente, que ajusta los parámetros del modelo en función de la magnitud de los errores, lo que permite ajustar la importancia relativa de cada atributo del conjunto de datos y mejorar la precisión general. Una particularidad de este modelo es que dispone de una serie de hiperparámetros que mejoran el ajuste de este y permiten minimizar un posible sobreajuste, o *overfitting*.

3. Obtención, preparación y exploración de datos

En este tercer capítulo del trabajo se describe como se lleva a cabo la obtención del conjunto de datos que se emplea posteriormente en el modelo de aprendizaje automático. También se detalla el tratamiento y la limpieza necesarios de los datos para poder ser empleados en el modelo. Finalmente se realiza el procedimiento efectivo del conjunto de datos, para posteriormente realizar un estudio de carácter estadístico de los mismos.

3.1. Obtención del conjunto de datos

Como se comenta anteriormente, no es posible emplear la API ofrecida por el portal inmobiliario Idealista. Por ello, la alternativa encontrada para obtener el conjunto de datos que emplear en el modelo de aprendizaje automático es utilizar una extensión de *web scraping* denominada Web Scraper. Cabe destacar que el conjunto de datos que debe ser obtenido consiste en todas las viviendas en venta en la ciudad de Barcelona cuyo anuncio esté publicado en el portal inmobiliario. Para ello, el primer paso es instalar dicha extensión en el navegador utilizado habitualmente, Google Chrome. Una vez instalada, se debe configurar adecuadamente para obtener el conjunto de datos objetivo, que se comenta posteriormente. Sin embargo, previamente a realizar una explicación sobre la configuración y obtención de los datos, conviene realizar ciertos comentarios.

Como se ha comentado, el conjunto de datos debe contener todas las viviendas en venta en la ciudad de Barcelona anunciadas en el portal. El principal inconveniente para obtener dicho conjunto de datos radica en los sistemas de seguridad de Idealista. Cuando el servidor detecta que el cliente ha sobrepasado el límite de peticiones en un período de tiempo dado, lo indica con la aparición por pantalla de la respuesta HTTP Error 429. La solución rápida para este tipo de error es, o bien esperar, o bien reiniciar el rúter para obtener una dirección IP distinta. En todo caso, este hecho detiene el *scraping* y por tanto la obtención del conjunto de datos. Por otro lado, cuando se detecta por parte de la página web algún comportamiento poco habitual (e.g. navegación y clics a una velocidad sobrehumana, bloqueo del funcionamiento de JavaScript en el ordenador del usuario o la detección de un robot en la misma red), se

redirecciona a una página que requiere la solución de un captcha. En concreto, requiere desplazar una pieza de puzle que debe ser situada en la posición correcta de una imagen. La imagen y posición correcta de la pieza cambia cada vez que se redirecciona a esta página. Un ejemplo de la página a la que se redirecciona al usuario en la que aparece el mensaje y el contenido relativo a su uso sospechoso es el que se muestra a continuación.

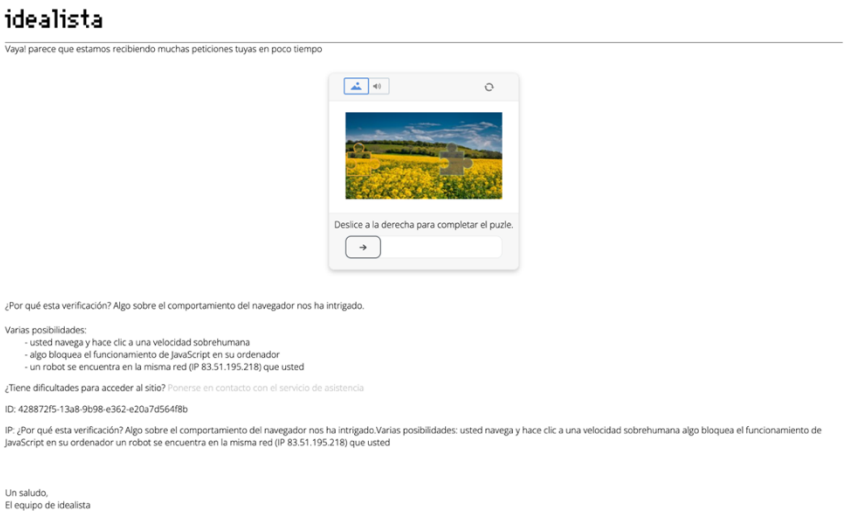


Figura 4. Página de verificación

En el caso del conjunto de datos que se quiere obtener, parece natural que se de este tipo de verificación, puesto que la navegación es demasiado rápida para ser producida por un humano y la cantidad de peticiones y acceso a distintas páginas del portal es sustancialmente superior a la que podría considerarse habitual. Este tipo de verificación suspende el *scraping* llevado a cabo por la extensión, por lo que requiere de un ser humano para poder reanudar el proceso. Por ello, no es posible automatizar el proceso de obtener la información de todas las viviendas a la venta en Barcelona a la vez. Además, no es posible reanudar el proceso desde el punto en el que suspende el procedimiento, ni determinar una cantidad precisa de anuncios visitados a partir de la cual se redirecciona a la página de verificación. Tampoco es posible obtener información relativa a estas medidas de seguridad en el portal inmobiliario. Sin embargo, la solución que se aporta y que se ha demostrado funciona es la de obtener el conjunto de datos separado por distritos. De esta manera, se obtienen *batches* (o lotes) que pueden ser posteriormente unidos obteniendo el conjunto de datos deseado final. Por tanto, en caso de que el proceso se detuviera a mita de uno de los distritos, sería posible reanudar el

proceso desde el principio de dicho distrito, pudiendo conservar el conjunto de datos de aquellos distritos que se hubieran completado anteriormente.

Los diez distritos oficiales que componen la ciudad de Barcelona[23] son los siguientes:

- Ciutat Vella (o Ciudad Vieja).
- Eixample (o Ensanche).
- Gràcia (o Gracia)
- Horta-Guinardó.
- Les Corts.
- Nou barris (o Nueve Barrios).
- Sant Andreu (o San Andrés).
- Sant Martí (o San Martín)
- Sants-Montjuïc (o Sans-Montjuich).
- Sarrià-Sant Gervasi (o Sarriá-San Gervasio).

Y se distribuyen geográficamente de la siguiente manera:

Figura 5. Distritos de Barcelona

Por tanto, cabe destacar que, tras descargar los distintos conjuntos de datos donde cada uno contiene información relativo a viviendas a la venta en uno o varios distritos de la ciudad en cuestión, es necesario agruparlos en un solo conjunto de datos. Para ello, se desarrolla un *script* de Python que permita realizar dicha agrupación, que será explicado posteriormente.

Adicionalmente, se detecta cierta lentitud realizando la extracción de datos mediante el *web scraper*. Esto es debido a la publicidad de compañías externas realizada en el portal inmobiliario. Idealista permite a empresas anunciarse con distintos *displays* en distintas páginas de su sitio web. Por ello, se considera necesario instalar una extensión de bloqueo de anuncios ofrecida en Google Chrome. Se realiza la instalación de la extensión uBlock Origin[24] para el navegador comentado. Tras su instalación y activación, la velocidad de *scraping* aumenta considerablemente, puesto que ya no es necesario cargar aquellos anuncios en forma de *display* que se van mostrando sin la extensión activada. Es importante destacar que la aparición de publicidad es frecuente, y teniendo en cuenta la cantidad de páginas a las que se debe acceder con la herramienta de *web scraping*, se reduce considerablemente el tiempo total que se requiere para obtener todo el conjunto de datos.

Una vez realizadas las valoraciones respecto a la obtención del conjunto de datos, se puede comentar la configuración y la obtención de los distintos lotes, uno para cada distrito de la ciudad de Barcelona. Una vez instalada la extensión de Web Scraper, se puede acceder a esta herramienta accediendo a las herramientas de desarrollo (o *developer tools*) del explorador Google Chrome, tal como se indica al clicar sobre la extensión. En el entorno utilizado, se abre clicando simultáneamente las teclas *command*, *option* e I.

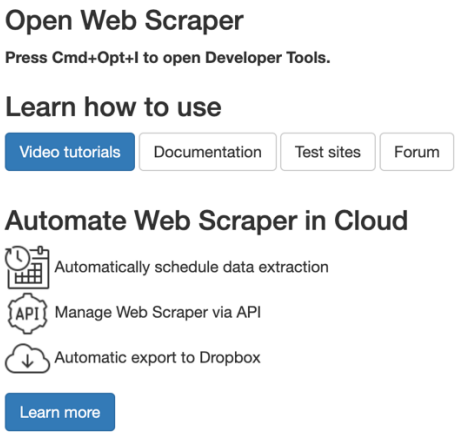


Figura 6. Indicaciones para abrir la herramienta Web Browser

Cuando se abre la sección de herramientas de desarrollo, se debe clicar en el apartado correspondiente a Web Scraper, y en el mismo clicar sobre la parte de *Create new sitemap*. Una vez abierto, es posible configurar como debe ser el *scraping* deseado. En primer lugar, se debe nombrar el *Sitemap*, que será el mapeo o recorrido que se haga en el sitio web. Se le nombra scraping-barcelona-districts. Además, se deben añadir la página o páginas desde la que se iniciará el *scraping*. En el caso de este trabajo, se incluyen las páginas relativas a la venta de viviendas en cada uno de los diez distritos que conforman la ciudad de Barcelona. Cuando cada distrito haya finalizado, se empezará con el siguiente.

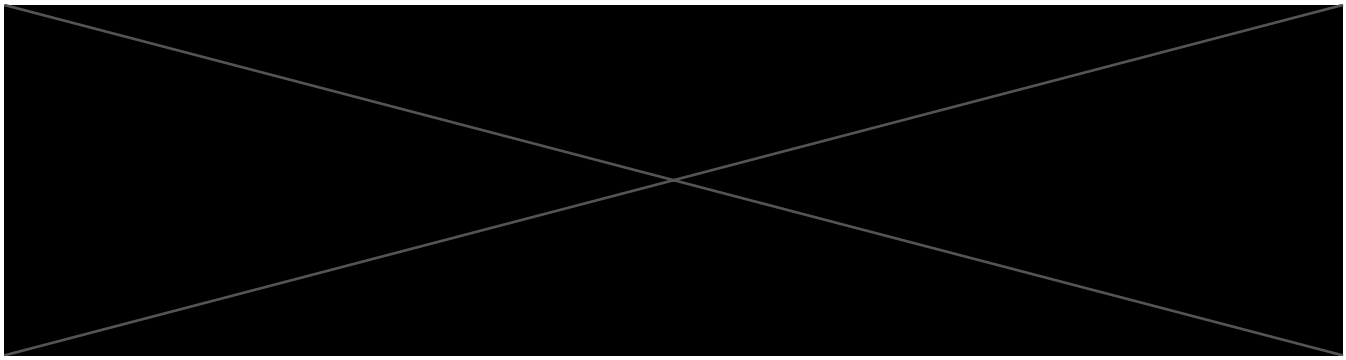


Figura 7. Creación del *Sitemap*

Una vez configuradas las URL de inicio y el nombre del *Sitemap*, es necesario configurar el árbol de navegación que debe recorrer la herramienta, de manera que pueda saber qué información extraer de cada una las páginas. Por tanto, cada vez que acabe de recorrer cada distrito volverá a empezar, tomando el siguiente enlace de la lista preestablecida que se visualiza en la Figura 6. En primer lugar, desde la raíz del árbol se origina una rama que genera el siguiente nodo, que obtiene los enlaces de los barrios del distrito de Barcelona en cuestión que se recorrerán. Con cada barrio, se debe llevar a cabo la paginación, que es el siguiente nodo del árbol. La paginación es el sistema que emplea Web Scraper para ir avanzando en las distintas páginas contenedoras de anuncios de cada barrio. En cada barrio, los anuncios están distribuidos a lo largo de una serie de páginas, con el fin de que sea más cómodo para el usuario ir avanzando a través de ellas en lugar de tener que deslizar hacia abajo de manera prolongada a través de cientos o miles de anuncios. Para cada una de las páginas del distrito se generan dos ramas. La primera tiene como nodo resultante otra paginación, de manera recursiva, para cada una de las páginas en las que se encuentren todos los anuncios que tenga el distrito. La segunda rama tiene como nodo resultante los enlaces a los distintos

anuncios de cada página. Esto quiere decir que, en cada página, la herramienta de *scraping* entrará en cada uno de los anuncios. Por tanto, para cada enlace al anuncio se generarán tres ramas y tres hojas (o nodos finales). Estos nodos finales son la información que la herramienta debe obtener, y consisten en el título del anuncio (*title*), el precio (*price*) y las características del inmueble (*features*). La estructura en forma de árbol del *Sitemap* se muestra a continuación.

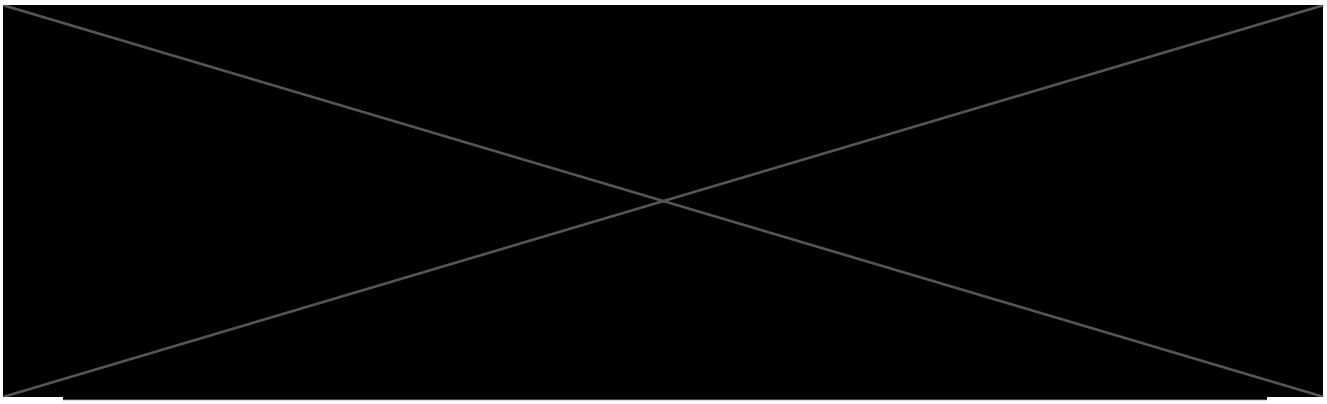


Figura 8. Estructura en forma de árbol del *Sitemap*

La extensión de *web scraping* permite configurar de manera relativamente sencilla la información que debe ser obtenida y la navegación que se debe llevar a cabo. Permite de manera visual identificar aquellos campos en los que se puede clicar y reconoce el resto de los campos que son iguales. Por ejemplo, en cada una de las páginas es suficiente clicar en dos anuncios para que la herramienta detecte que el resto de los anuncios también deben ser clicados.

Una vez se ha realizado la configuración, se debe clicar sobre el *Sitemap* generado y posteriormente darle a la opción *Scrape*, que empieza el proceso de navegación y obtención de información. Este proceso es visible en todo momento, puesto que se abre una ventana del navegador nueva, en la que se aprecia como se produce la navegación automatizada por las distintas páginas y anuncios. Se puede apreciar con más detalle el funcionamiento interno de la estructura del árbol generado, y en concreto el funcionamiento de la paginación. Para cada barrio de cada distrito, Web Scraper recorre en primer lugar todas las páginas, sin acceder a ninguno de los enlaces donde se encuentran los anuncios de los inmuebles, pero sí almacenando dichos enlaces. Una vez alcanza la última página, la herramienta dispone de todos los enlaces a todos los anuncios de cada barrio, por lo que acto seguido accede uno a uno

cada uno de ellos. Se puede visualizar por tanto el acceso directo de un anuncio tras otro, desde donde la herramienta recupera la información, sin acceder previamente al sitio web donde los anuncios están listados por páginas. En el Anexo 1 se puede encontrar el código en formato JSON que permite importar la configuración del *Sitemap* empleado. Para utilizarlo, es suficiente con clicar en *Create new sitemap* y posteriormente en *Import sitemap*. A continuación, es suficiente con copiar y pegar el contenido del Anexo 1 y opcionalmente renombrar el *Sitemap*.

Existen dos vías que finalizan el *scraping*. Por un lado, la finalización exitosa de la obtención del conjunto de datos, en la que la herramienta llega hasta el último anuncio del que debía obtener la información. A continuación, se cierra la ventana del navegador en la que se producía la navegación automática. La otra opción es cuando la finalización se produce de manera forzosa, puesto que Idealista detecta una navegación irregular y por tanto obliga a resolver un captcha o el servidor devuelve el Error 429 por exceso de peticiones en un plazo determinado de tiempo. En ambos casos es necesario clicar de nuevo sobre el *Sitemap* empleado y posteriormente en *Export*, para exportar los datos en el formato deseado. Las dos opciones posibles son en formato .XML y .CSV. En este trabajo se empleará el segundo formato. Una vez exportados los datos, si la finalización no ha sido exitosa, es necesario comprobar aquellos distritos que se han completado y retirarlos del nuevo *scraping*. También es necesario eliminar del fichero .CSV las instancias de los distritos que no se hayan completado. A continuación, se debe volver a poner en funcionamiento la herramienta con la lista de distritos actualizada en Web Scraper. En caso de haber sido un proceso exitoso, se obtiene el conjunto de datos dividido en diversos archivos .CSV, que deben ser unidos.

Para llevar a cabo la unión de los diversos archivos en formato .CSV que contienen los anuncios de los distintos distritos, se ha llevado a cabo la elaboración de un script en Python que permite realizar esta tarea. Es posible que los diez distritos se distribuyan entre uno (en el mejor de los casos) y diez archivos (en el peor de los casos) .CSV, pero es más probable que el número de archivos esté entre dos y cinco. Es necesario observar que el formato que se emplea posteriormente en el script del modelo donde se trata previamente el conjunto de datos debe contener una sola cabecera, la primera, y el resto instancias (o filas) donde cada una será una vivienda anunciada con sus características correspondientes. Puesto que las exportaciones de datos desde Web Scraper son varias, cada una contendrá una cabecera y las instancias, por lo que en la unión se debe evitar la repetición de los *headers*, de manera que se obtenga un solo

archivo como si la exportación del conjunto de datos hubiera sido una solamente. Por tanto, el script debe leer uno o más ficheros y concatenarlos en una única salida, manteniendo los encabezados del primer fichero leído. Su funcionamiento específico se comenta a continuación. En primer lugar, se importa el módulo *sys* para obtener acceso a parámetros y funciones del sistema. Posteriormente, se definen las variables *headers* y *lines*, siendo la primera la que almacenará la cabecera y la segunda una lista donde se incluirán todas las líneas de todos los archivos leídos, el conjunto de datos final, sin la cabecera. A continuación, se itera a través de cada argumento posterior al nombre del script, obtenidos desde la línea de comandos. Cada argumento será un fichero. Si el archivo está vacío, puesto que la primera línea lo está, se salta a la siguiente iteración, al siguiente fichero. En la primera iteración del bucle que va fichero por fichero se establece cuál debe ser la cabecera, y en las siguientes iteraciones, en los siguientes ficheros, se comprueba que la cabecera sea la misma que la del primer fichero para corroborar la consistencia del formato. Posteriormente, se lee línea por línea el resto del fichero, sin contar la línea de la cabecera, utilizando un bucle anidado del tipo *for*, y se añade a la variable *lines*. El fichero final que se obtiene contiene una cabecera y todas las instancias (las viviendas anunciadas). Un ejemplo de la forma de ejecutar el script será la siguiente: nombre_script fichero1 fichero2 fichero3 > fichero_final, donde el número de ficheros a unir es tres. El fichero final contendrá el conjunto de datos que podrá ser usado en el script donde se desarrolla el modelo de aprendizaje automático. El script para la unión de diversos archivos .CSV se puede encontrar en el Anexo 2 de este trabajo.

Cabe mencionar que, con el fin de velar por las buenas prácticas en programación, todo aquel código aportado en los Anexos se encuentra íntegramente comentado. Además, el tipo de programación que se lleva a cabo es modular y estructurado, de manera que resulta más sencillo reutilizar y mantener el código, así como mejorar su legibilidad.

3.2. Tratamiento previo del conjunto de datos

Previamente al uso del conjunto de datos para entrenar y testear al modelo de aprendizaje automático, conviene tratar el conjunto de datos obtenido a partir de la herramienta Web Scraper y el script para la unión de varios ficheros. Es necesario realizar este tratamiento ya que el modelo requiere de datos que sean claramente distinguibles. En concreto, el modelo requiere un *DataFrame*, una estructura de datos que organiza los datos en

una tabla bidimensional de filas y columnas. Cada atributo debe tener su mismo formato en cada instancia, y debe ser un formato aceptado por el modelo. Por tanto, cuando se realiza el entreno y testeo del modelo para posteriormente predecir, es necesario haber tratado los datos anteriormente.

En primer lugar, se ha declarado una variable que es una lista, llamada *SALE_DATA_FILES*, y almacena todos los conjuntos de datos que puedan servir para entrenar y testear el modelo. Aunque el desarrollo de este trabajo implica obtener un conjunto de datos en un día concreto, se configura el modelo de manera que se puedan emplear conjuntos de datos históricos. Por ejemplo, aunque las cerca de las 16.000 viviendas anunciadas para la venta actualmente que se encuentran en el conjunto de datos obtenido, dentro de un año gran parte de ellas se habrán vendido y habrá nuevas que estén a la venta, por lo que se podrá incluir tanto información del momento como antigua para entrenar y testear al modelo. Así, es posible emplear varios conjuntos de datos obtenidos en distintas fechas. Por otro lado, también se deja abierta la posibilidad de intentar predecir el valor de más de una vivienda, por lo que se declara una variable que es una lista, llamada *SALE_PREDICT_DATA_FILES*, en la que se almacenan aquellos inmuebles cuyo valor se desea predecir.

Una vez establecido el conjunto de datos, es necesario cargar y tratar los datos. Para ello, y como se comenta en el apartado anterior, el script se estructura mediante la definición de funciones, que permite generar un código modular más comprensible y versátil en caso de requerir realizar cambios.

La primera función definida, *read_csv*(), que es generadora, permite abrir el fichero de datos en formato .CSV que se le pasa como parámetro y devuelve cada fila como una lista de *strings*, donde cada uno corresponde a un campo en el archivo .CSV. Para ello, se utiliza la librería csv importada. Esta función se puede encontrar en el Anexo 3.

A continuación, se declara la función *read_csv_with_headers*(), que a su vez llama a la función comentada anteriormente, *read_csv*(). La función lee el fichero .CSV y verifica que la cabecera es la esperada, puesto que ésta también se le pasa como parámetro. Si los encabezados son los esperados, la función devuelve un objeto generador que puede usarse para iterar sobre las filas del fichero. Esta función sirve principalmente para comprobar fehacientemente de que la cabecera sea la correcta, y devolver todas las instancias salvo la cabecera. Se puede encontrar en el Anexo 4.

La siguiente función se denomina *load_data*(), y realiza diversas funciones, siendo la principal devolver una estructura de datos en forma de tabla en la que las columnas serán los atributos que se le pasarán al modelo de aprendizaje automático y las filas serán cada una de las instancias, o de manera equivalente, cada una de las viviendas con sus características individuales con un mismo formato válido para el modelo.. En primer lugar, se comprueba que el encabezado esperado es el mismo que aparece en el fichero y que no hay discordancia. Se llama a la función *read_csv_with_headers*() que devuelve las instancias del conjunto de datos, y por tanto no devuelve la cabecera. Cada instancia contiene información de una vivienda a la venta, con su barrio, URL, título, precio y características correspondientes. Tras comprobar la validez del encabezado, se recorre todo el conjunto de datos instancia por instancia, para poder obtener finalmente un *DataFrame*. En el proceso de recorrer todo el conjunto de datos, se genera un diccionario (o mapa) que permite asociar cada una de las columnas o atributos que se le pasan al modelo con los atributos particulares de cada vivienda. Es decir, para cada vivienda se obtiene la información de cada atributo que el modelo utilizará para realizar su predicción. Por ello, se genera una lista con todas las columnas que tendrá el *DataFrame*, que no deja de ser una estructura de datos en forma de tabla, y se va anuncio por anuncio convirtiendo su información, que está en formato *string*, al formato necesario para cada atributo, y que así pueda ser incluido en el *DataFrame*. Para ello, aunque en algunos casos la obtención de la información en el formato adecuado es directa y no es necesario nada más que asignar la característica de la instancia a su clave correspondiente, en gran parte de los atributos es indispensable emplear expresiones regulares o condicionales para aislar y asignar la información estrictamente necesaria. Una vez realizado el proceso de extraer las características de cada anuncio en el formato adecuado, para cada instancia se comprueba que todas las características de cada instancia se asocian a un atributo definido inicialmente. Finalmente se crea y devuelve la variable que contiene el *DataFrame* rellena con la información de las columnas y de las instancias. Esta función se puede visualizar en el Anexo 5.

Por último, y antes de comenzar con el entreno del modelo, una vez se cargan y obtienen los datos en el formato y estructura de datos adecuados, es necesario realizar una limpieza sobre los mismos. Para ello, se emplea la función *clean_up_data*(), que se encuentra en el Anexo 6, y que realiza las siguientes funciones de limpieza:

- Se suprimen valores de instancias que se detectan como información errónea publicada en el anuncio. En concreto, se trata el del consumo energético relacionada la eficiencia energética de la vivienda. Se aprecia que en un número considerable de anuncios se establece un número excesivamente alto de consumo, incluso hasta alcanzar 999 kWh/m^2 por año, muy alejado del consumo promedio. Esto puede ser tal vez explicado por el hecho de que sea obligatorio incluir esta información cuando se crea el anuncio, pero no se disponga de la misma. El tratamiento que se le da a este tipo de casos es sustituir los valores extremos por *None*, equivalente a *null*, de manera que el modelo no pueda utilizar el atributo en la instancia concreta y así no se produzcan sesgos.

- Se descartan aquellas viviendas que pueden considerarse *outliers*, o valores extremos. En concreto, se descartan instancias enteras si se detectan valores extremos en precio, superficie, número de habitaciones, número de baños o número de plantas del edificio. También se descartan aquellos pisos que no puedan ser considerados habitables. Según marca la ley de vivienda catalana actual, la superficie debe ser superior a los 35 m^2.

- En los casos en los que falta información para algunos atributos binarios (si la vivienda tiene balcón, terraza, aire acondicionado, etc.) se interpreta que la ausencia implica la no disposición de dichos atributos. Por ejemplo, si en el anuncio no se indica que una vivienda tiene balcón, se interpreta que la vivienda no tiene. Por ello, se deben sustituir los valores *None* de estos casos por *False*. Además, realizar esta sustitución convierte el tipo de variable almacenado en el *DataFrame* de un objeto a una variable booleana que puede tomar valores *True* y *False*. Sin embargo, se interpreta que hay ciertos atributos para los que no es tan claro que su inadvertencia implique su inexistencia, como en el caso de tener ascensor o calefacción.

- De manera similar se trata la orientación de las viviendas. En el caso de que se informe como *True* si la vivienda está orientada hacia uno o varios puntos cardinales, la orientación al resto de los puntos cardinales debe considerarse como *False*. Una vivienda en la ciudad de Barcelona puede estar orientada a más de un punto cardinal.

- Se eliminan los duplicados. Dos anuncios de viviendas no pueden tener la misma URL ni características iguales. Ocurre con frecuencia que varias empresas o agencias inmobiliarios publiquen o anuncien la misma vivienda a la venta, especialmente en los casos en los que los vendedores no hayan acordado un contrato de exclusividad. Por ello, es común encontrar el mismo inmueble con las principales características iguales, pero no todas. Por ello, el criterio para determinar si dos instancias son la misma vendrá dado por el hecho de que compartan simultáneamente barrio, tipo de propiedad, precio, superficie, baños, habitaciones, condición del inmueble, planta, si es exterior y si tiene balcón, terraza, jardín, piscina, trastero y ascensor.

Las funciones comentadas se llaman o bien desde otras funciones o bien desde la función *main*(), cuyo código correspondiente al tratamiento previo de los datos se encuentra en el Anexo 7 de este trabajo.

3.3. Análisis del conjunto de datos

Una vez descritos los procesos de obtención de conjunto de datos y de su tratamiento previo al desarrollo del modelo, se procede a la obtención y tratamiento del conjunto de datos real. En primer lugar, se configura Web Scraper para que obtenga la mayor cantidad de distritos posibles, incluyéndolos todos en la modelación de los metadatos. En un primer *scraping*, se logra obtener todos los anuncios de cuatro distritos: Horta-Guinardó, Gràcia, Eixample y Ciutat Vella. Una vez finalizados por completo estos distritos, y empezando con uno nuevo, se redirecciona al sitio web que requiere de la resolución de un captcha para poder continuar. Por ello, se detiene el proceso y se extrae un primer fichero .CSV, nombrado scraping-barcelona-districts-1.csv. Este fichero contiene 8.321 instancias (o anuncios de viviendas), en las que no se cuenta la cabecera. Para obtener el resto de los distritos, se configura de nuevo Web Browser retirando de los metadatos aquellas URL correspondientes a los distritos que ya se han obtenido. Se ejecuta el segundo *scraping*, en el que se obtienen el resto de los barrios, que son Sarrià-Sant Gervasi, Sants-Montjuïc, Sant Martí, Sant Andreu, Nou Barris y Les Corts. En este caso, la finalización es exitosa y por tanto no es forzada. Se extrae un segundo fichero .CSV, nombrado scraping-barcelona-districts-2.csv. Este fichero contiene 7.817 instancias (o anuncios de viviendas), en las que no se cuenta la cabecera. La

obtención de ambos ficheros ha conllevado un tiempo total aproximado de unas 14 horas en total, 8 horas para el primer fichero y 6 horas para el segundo.

Aunque el script está configurado para poder incluir ambos ficheros directamente en la lista *SALE_DATA_FILES*, se considera más apropiado unirlos en un solo fichero que permita disponer de todos los anuncios de todos los distritos de la ciudad en uno. Para ello, se emplea el script incluido en el Anexo 2, utilizando una terminal y ejecutando el comando adecuado desde la carpeta en la que se encuentran ambos ficheros y el script, de la siguiente manera.

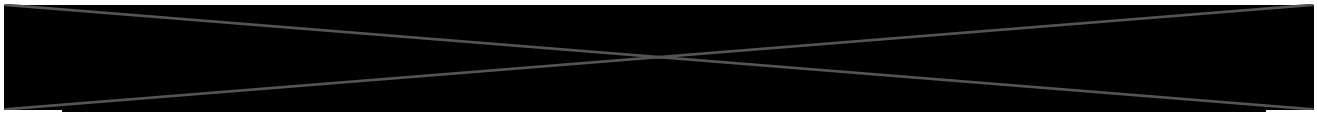


Figura 9. Ejecución del script de unión de ficheros .CSV

Tras ejecutar el script, se obtiene un solo archivo .CSV cuyo nombre es scraping-barcelona-districts.csv. Éste contiene la suma de instancias de los dos ficheros obtenidos, que son 16.138.

Este fichero .CSV, que contiene valores separados por comas, será el utilizado por el modelo de aprendizaje automático. Por tanto, se debe realizar el tratamiento del conjunto de datos. Previamente a ello, se realiza un estudio de los datos. En concreto se comprueba, en primer lugar, información sobre el *DataFrame*, que incluye los tipos de datos de cada columna, así como la cantidad de valores que no son *null*. El resultado se obtiene tras utilizar el método *.info*() de la librería Pandas, y es el que se muestra continuación.

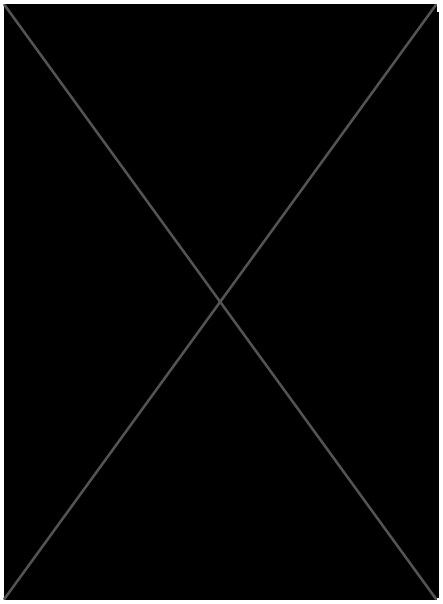


Figura 10. Resultado de emplear el método *.info*()

Como se puede comprobar, se muestran las 28 columnas del conjunto de datos, ordenadas en función de la cantidad de instancias que disponen de información para ese atributo. Los atributos que aparecen en más instancias son la URL, el barrio, el tipo de propiedad, el precio, la superficie, el número de habitaciones y el número de baños. Por otro lado, se emplea el método *.corr*(), que permite calcular la correlación entre todas aquellas variables que son numéricas. El resultado obtenido es el que se muestra a continuación.

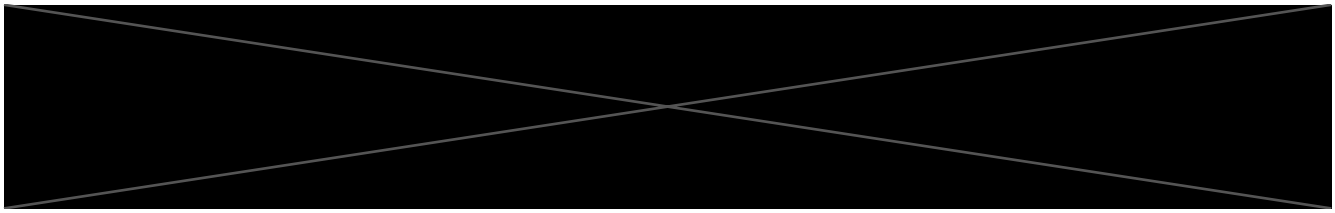


Figura 11. Resultado de emplear el método *.corr*()

Se puede visualizar claramente como la superficie, el número de habitaciones y el número de baños tienen una correlación moderada, una correlación moderada y una correlación fuerte respectivamente. El año de construcción apenas está correlacionado con el precio de la vivienda, y la planta tiene una correlación débil, mientras que el precio del parquin y el consumo energético están correlacionados negativa y débilmente con el precio. Estas correlaciones parecen razonables con la lógica económica. Por último, se muestra un resumen

estadístico de las mismas columnas numéricas mediante el uso de la función *.describe*(), como se aprecia seguidamente.

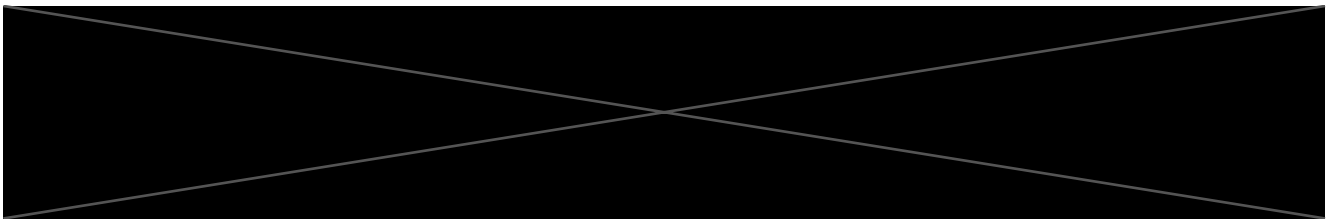


Figura 12. Resultado de emplear el método *.describe*()

Esta información permite comprobar que el precio medio se encuentra alrededor de los 570.000 de euros. Para el resto de los atributos, la superficie media es poco más de 115 m^2, el número de habitaciones casi 3, el número de baños casi 2, el año de construcción 1955, la planta casi 3, el precio del parquin alrededor de 5.000 € y el consumo energético de unos 200 kWh/m^2. También es posible apreciar que existen valores extremos. La superficie mínima dentro del conjunto de datos es de 13 m^2, lo que no puede considerarse habitable. Por otro lado, la vivienda con superficie máxima se supone tiene más de 16.000 m^2, por lo que se puede deducir que debe ser un error. Para solventar este tipo de circunstancias se realiza la limpieza de los datos.

Si se realiza la llamada a la función *clean_up_data*() comentada anteriormente es posible realizar el análisis del nuevo conjunto de datos y observar las diferencias. Los resultados se muestran a continuación.

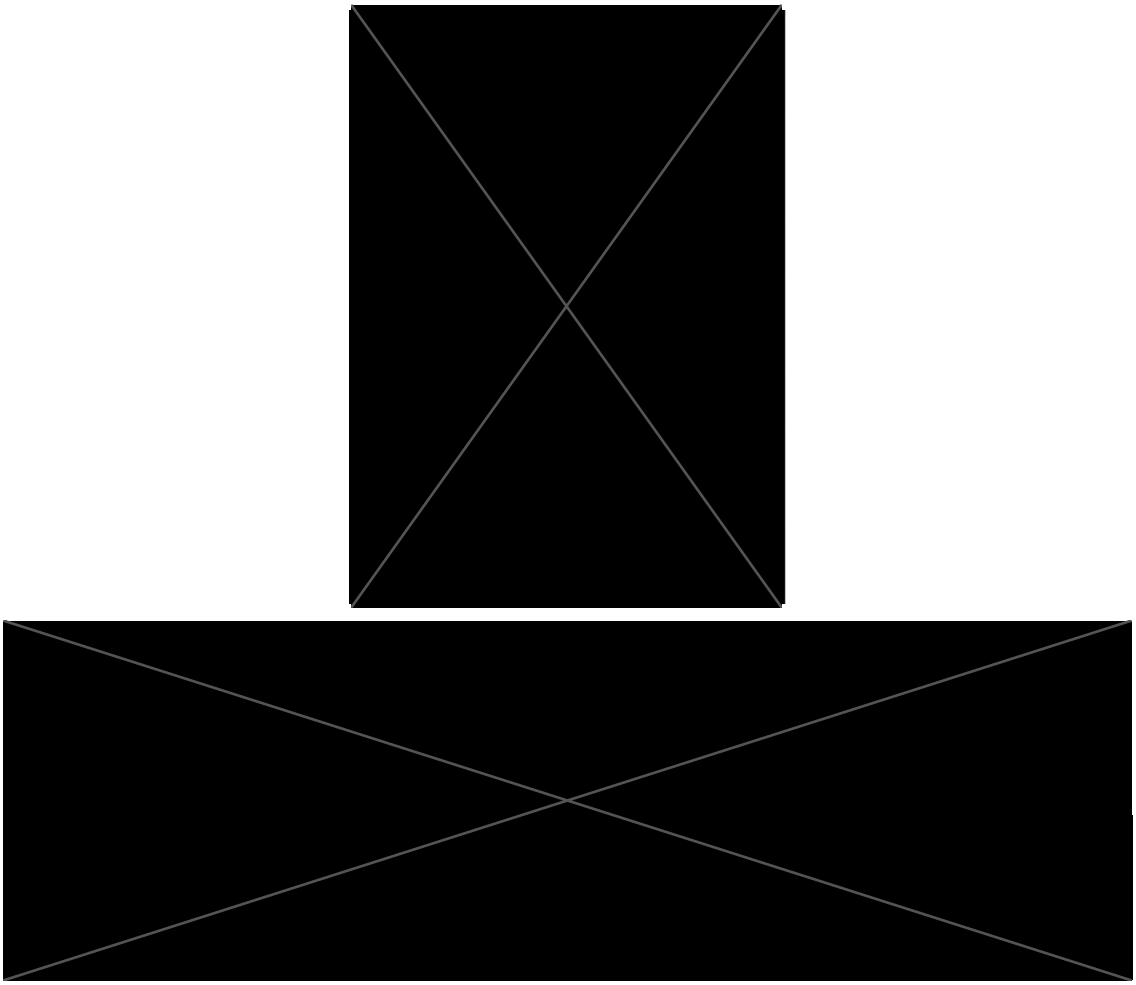


Figura 13. Resultados de emplear de nuevo los métodos *.info*(), *.corr*() y *.describe*()

Como se puede apreciar, el número de instancias se ha reducido prácticamente en un tercio. En el nuevo conjunto de datos, el número de instancias es de 12.039. Por otro lado, la correlación entre el precio de la vivienda y la superficie aumenta hasta considerarse una magnitud fuerte, mientras que el resto de las correlaciones se mantiene similar. Una vez eliminados los valores extremos y aquellos que no pueden considerarse válidos o están duplicados, las medias de los atributos disminuyen, por lo que las instancias eliminadas son principalmente de valores altos. Tanto el valor máximo como el valor mínimo se encuentran más cerca de la media que antes de la limpieza, y la desviación estándar, como es de esperar, ha disminuido.

Las líneas de código relativas a la impresión por pantalla de los métodos de análisis del conjunto de datos de tipo *DataFrame* y la llamada a la función para limpiar los datos se encuentra en el Anexo 7 de este trabajo.

4. Modelo de aprendizaje automático

El cuarto capítulo se centra en la implementación del modelo de aprendizaje automático. En primer lugar, se describe el proceso empleado para realizar su entrenamiento, que será posteriormente evaluado para determinar el grado de ajuste. Con ello, se discute mediante comparación si puede resultar una herramienta válida en relación con las disponibles actualmente. Finalmente, se detalla el procedimiento para realizar la predicción del precio de una vivienda en particular haciendo uso del modelo desarrollado.

4.1. Entrenamiento del modelo

Una vez obtenido el conjunto de datos y realizada la limpieza adecuada del mismo, se procede a realizar el entrenamiento del modelo. El entrenamiento permite al modelo adquirir el aprendizaje suficiente como para intentar predecir con el mayor nivel de confianza posible el valor de una vivienda a partir de sus características. Para ello, se describe qué proceso se sigue para realizar el entrenamiento comentado.

El entrenamiento del modelo se concentra en su totalidad en una función denominada *model_train*(), al que se le pasa el *DataFrame* del conjunto de datos tratados y que a su vez llama a otras funciones. Dicha función se encuentra en el Anexo 8 de este trabajo. El primer paso en la función de entrenamiento del modelo es realizar un *split* (o división) del conjunto de datos en dos grupos, por un lado, el conjunto de datos que se utiliza para entrenar el modelo y por otro lado el conjunto de datos que se utiliza para probar (o testear) el modelo. La proporción de instancias del conjunto de datos que se dedica para testear el modelo es del 20%. Aunque típicamente la proporción entre el conjunto de entrenamiento y el conjunto de prueba es 70/30, dada la muestra existente no parece necesario que se emplee más de un 20% del conjunto de datos para la prueba para obtener una buena estimación del desempeño del modelo. También se asigna el valor 42 para el estado de aleatoriedad, que debe mantenerse constante cada vez que se emplee el programa si se desean obtener resultados reproducibles.

A continuación, se obtiene un diccionario, también denominado mapa, en el que se relaciona cada barrio con su precio medio por metro cuadrado. Este cálculo está motivado por el hecho de que, en lugar de emplear el barrio como un atributo del conjunto de datos, que es

una variable categórica con 73 niveles, se emplea el precio por metro cuadrado del barrio, como variable numérica continua. El motivo para realizar este cambio se debe a que el precio medio por metro cuadrado de un barrio resulta más útil para el modelo a la hora de determinar el efecto en el precio de un inmueble. Según los ganadores del concurso de Zillow realizado en Kegel[25], los resultados del modelo de predicción son sustancialmente mejores que si se empleara el barrio como variable categórica. Para calcular el precio por metro cuadrado de cada barrio se emplea la función *compute_neighborhood_m2_price*(), que se puede encontrar en el Anexo 9 de este trabajo. Esta función devuelve el diccionario que relaciona cada barrio, la clave, con su precio medio por metro cuadrado, el valor. Para crear este diccionario, se recorre línea a línea un *DataFrame* creado para ser iterado. Este *DataFrame* une la columna con los barrios del conjunto de datos, para cada instancia, y la columna que computa el precio por metro cuadrado de cada inmueble. Se agrupa por barrio y se calcula su media de precio. En cada iteración del recorrido por el *DataFrame*, por cada fila, se asigna clave y valor al diccionario. De esta manera, se obtiene un diccionario que relaciona barrio con su precio promedio del metro cuadrado.

Una vez se obtiene el diccionario comentado, se puede proceder tanto a la obtención del conjunto de datos de entrenamiento como del conjunto de datos de prueba. Para ello, se deben obtener las características definitivas que se emplearán en ambos conjuntos. Para ello, se llama a la función *extract_features*(), que puede encontrarse en el Anexo 10. Esta función recibe como parámetros un *DataFrame* y el diccionario que relaciona cada barrio con su promedio del precio por metro cuadrado, y crea un *DataFrame* que contiene cada uno de los atributos que se emplean en el modelo. La mayoría de los atributos son los mismos que se pasan como parámetro. Sin embargo, hay un caso en el que no. En lugar de emplear la variable barrio como atributo, se emplea la variable precio medio por metro cuadrado. En caso de no disponer de dicha información para alguno de los barrios, se le asigna el precio medio obtenido de promediar el precio medio por metro cuadrado de todos los barrios de la ciudad de Barcelona. Finalmente, se convierte cada variable categórica en formato *one-hot-encoding* mediante el uso de *dummies*, variables que indican la presencia o ausencia da cada categoría posible del atributo. Por ejemplo, si la condición de un inmueble puede ser promoción de obra nueva, segunda mano/buen estado y segunda mano/para reformar, y el inmueble es de obra nueva, se generarán tres variables binarias cuyos valores respectivos serán 1, 0 y 0. Este proceso

es imprescindible para que el modelo pueda funcionar, puesto que no acepta variables categóricas. Posteriormente, se separan cada uno de los conjuntos de datos, de entrenamiento y de prueba, por variables predictoras y variable objetivo.

A continuación, para obtener el modelo óptimo de XGBoost, se configuran tres hiperparámetros. Estos sirven para controlar el comportamiento de aprendizaje, por lo que conviene hallar aquellos que mejoran la capacidad del modelo para ajustarse a los datos de entrenamiento y por tanto predecir mejor. La selección de los hiperparámetros surge de la experimentación iterativa; comprobando cuales son aquellos que determinan en mayor medida el ajuste del modelo. Son los siguientes:

- *learning_rate*, una ratio de aprendizaje que determina la cantidad de corrección que se hace en cada paso del algoritmo. Cuanto mayor es este valor, mayor es la velocidad de aprendizaje, pero también el riesgo de *overfitting* (o sobreajuste).

- *max_depth*, que representa la profundidad máxima de los árboles de decisión. Cuanto mayor sea este valor, mayor será la complejidad del modelo y más se ajustará a los datos de entrenamiento, por lo que también se da el riesgo de *overfitting*.

- *n_estimators*, que corresponde al número de árboles que se construyen en el modelo.

Para encontrar que combinación de hiperparámetros obtiene el mejor modelo, se realiza una búsqueda por cuadrícula, que consiste en probar combinatorias de los mismos hasta determinar el que ofrece un modelo de regresión XGBoost óptimo. Para ello, se utiliza el método de validación cruzada *k-fold*[26] (cuya k por defecto es 5), que consiste en separar el conjunto de entrenamiento en 5 particiones con el mismo número de instancias. De forma iterativa se entrena un submodelo utilizando 4 de las particiones como entrenamiento y 1 como test. Así pues, se consigue disponer de 5 valores en lugar de 1 para la métrica de optimización escogida, además de su desviación estándar.

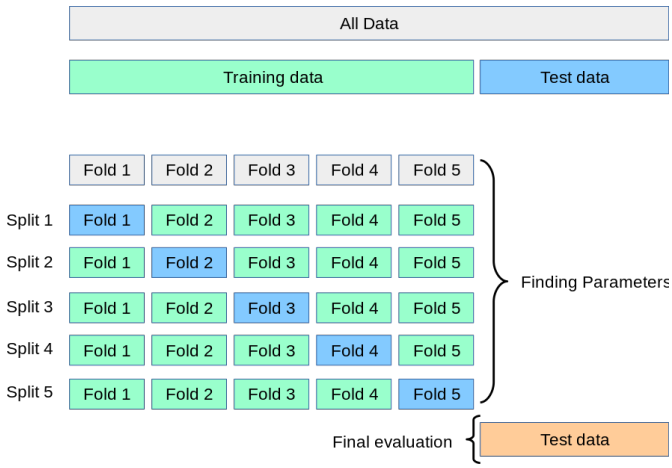


Figura 14. Método 5-*fold*

Posteriormente, se definen las métricas de evaluación que se usan para determinar el desempeño del modelo. Estas métricas son R^2, error absoluto medio negativo y error cuadrático medio negativo. La primera permite medir la proporción de la varianza en el precio explicado por el modelo, siendo de 0 a 1 el rango de valores que toma esta métrica. Cuanto más cercano a 1, mejor ajuste del modelo. El error absoluto medio negativo mide los errores absolutos de las predicciones del modelo. La última métrica es la media negativa de los errores al cuadrado de las predicciones del modelo, que permite penalizar las predicciones más extremas. La librería utilizada maximiza el valor negativo de la métrica en lugar de minimizar el valor positivo, por lo que su interpretación no cambia; los valores superiores son mejores que los inferiores. A continuación, se emplea el valor absoluto medio negativo como métrica de optimización para hallar el mejor conjunto de hiperparámetros. El motivo de emplear dicha métrica para la optimización es que, además de tener una interpretación directa, está enfocada en la magnitud de los errores. Ello resulta de especial interés teniendo en cuenta que el objetivo es la predicción de precios, puesto que conviene conocer el tamaño promedio de los errores sin ser tan relevante su dirección.

Desde el propio código se exporta en formato .CSV el *DataFrame* donde se muestran los resultados de la validación cruzada, que permite observar la combinación de hiperparámetros que optimiza el modelo. Tras ordenarlo y darle un formato que puede ser interpretado más fácilmente, se obtiene la tabla mostrada a continuación:

Params			R^2			Neg MAE			Neg MSE		
learning_rate	max_depth	n_estimators	Mean	Std	Rank	Mean	Std	Rank	Mean	Std	Rank
0.01	4	200	0,76	0,01	27	-121.650	3.427	27	-5,50E+10	6,56E+09	27
0.01	4	400	0,83	0,01	21	-103.619	2.873	24	-3,78E+10	3,43E+09	21
0.01	4	600	0,84	0,01	19	-101.053	2.432	23	-3,54E+10	2,44E+09	19
0.01	6	200	0,79	0,01	26	-113.252	2.670	26	-4,72E+10	4,93E+09	26
0.01	6	400	0,85	0,01	13	-95.293	2.311	12	-3,32E+10	2,90E+09	13
0.01	6	600	0,86	0,01	10	-93.626	2.148	11	-3,16E+10	2,64E+09	10
0.01	8	200	0,80	0,01	25	-110.649	2.244	25	-4,58E+10	4,76E+09	25
0.01	8	400	0,86	0,01	12	-90.489	2.502	7	-3,23E+10	2,96E+09	12
0.01	8	600	0,86	0,01	9	-89.264	2.342	6	-3,14E+10	2,72E+09	9
0.1	4	200	0,86	0,01	11	-96.242	1.690	13	-3,17E+10	2,23E+09	11
0.1	4	400	0,86	0,01	4	-93.491	1.597	10	-3,04E+10	2,13E+09	4
0.1	4	600	0,87	0,01	1	-91.910	1.870	9	-2,98E+10	2,31E+09	1
0.1	6	200	0,86	0,01	5	-90.601	1.794	8	-3,06E+10	2,54E+09	5
0.1	6	400	0,87	0,01	3	-88.806	1.633	5	-3,02E+10	2,63E+09	3
0.1	6	600	0,87	0,01	2	-88.218	1.510	4	-3,01E+10	2,62E+09	2
0.1	8	200	0,86	0,01	8	-87.674	1.406	3	-3,12E+10	2,46E+09	8
0.1	8	400	0,86	0,01	6	-86.918	1.221	2	-3,11E+10	2,36E+09	6
0.1	8	600	0,86	0,01	7	-86.911	1.212	1	-3,12E+10	2,36E+09	7
0.5	4	200	0,84	0,01	14	-99.271	1.043	17	-3,49E+10	1,96E+09	14
0.5	4	400	0,84	0,01	18	-99.414	1.643	18	-3,53E+10	2,39E+09	18
0.5	4	600	0,84	0,01	20	-99.588	1.861	19	-3,55E+10	2,44E+09	20
0.5	6	200	0,84	0,01	15	-97.386	1.940	14	-3,50E+10	2,35E+09	15
0.5	6	400	0,84	0,01	16	-97.769	1.886	15	-3,52E+10	2,50E+09	16
0.5	6	600	0,84	0,01	17	-97.986	1.880	16	-3,53E+10	2,51E+09	17
0.5	8	200	0,83	0,01	22	-99.759	2.392	20	-3,89E+10	2,84E+09	22
0.5	8	400	0,83	0,01	23	-99.803	2.397	22	-3,89E+10	2,84E+09	23
0.5	8	600	0,83	0,01	24	-99.801	2.396	21	-3,89E+10	2,85E+09	24

Tabla 1. Resultados de la validación cruzada

Como se puede comprobar, la tabla muestra las 27 combinaciones posibles de la validación cruzada, en la que cada hiperparámetro contiene 3 valores posibles. Puesto que se emplea el valor absoluto medio negativo (*Neg MAE*) como métrica de optimización, la combinación de hiperparámetros que optimiza el modelo es *learning_rate* = 0.1, *max_depth* = 8, *n_estimators* = 600, marcada con el 1 en la columna *Rank* para esta métrica. Cabe destacar que, si se empleara otra métrica de optimización del modelo como R^2 o la media negativa de los errores al cuadrado (*Neg MSE*), la combinación de hiperparámetros óptima en la validación cruzada sería distinta. Sin embargo, tras evaluar el modelo con dichas combinaciones alternativas, no se aprecia mejor optimización. Los valores medios de R^2, valor absoluto medio negativo y error cuadrático medio negativo con los hiperparámetros óptimos son 0,86, -86.911 y $-3{,}12 \times 10^{10}$. Estos valores son útiles para comparar en la evaluación, que se lleva a cabo previamente a la finalización de la función que devuelve el modelo optimizado.

4.2. Evaluación del modelo

Una vez se ha optimizado el modelo, es posible obtener métricas de su ajuste a partir de la partición del conjunto de datos destinado a realizar la prueba. Para ello, se llama a la función *evaluate*(), que puede encontrarse en el Anexo 11, desde la función *model_train*(). La función en la que se evalúa el modelo recibe como parámetros el modelo con el mejor estimador, que ha sido optimizado anteriormente, así como el conjunto de prueba, incluyendo tanto los atributos como el precio a predecir. Será posible gracias a ello realizar una evaluación final no sesgada del modelo. Con los atributos del conjunto de prueba se realiza la predicción de los precios empleando el modelo, para posteriormente compararlos con los reales y comprobar así la calidad del ajuste. Cabe destacar que estos datos no se han utilizado en el entrenamiento del modelo. Se imprimen por pantalla las tres métricas mostradas a continuación.

```
R2 score: 0.8797668794818977
MAE score: 80335.1413468776
MSE score: 26527901921.428032
Relative MAE: 0.15121994496056518
```

Figura 15. Métricas de ajuste del modelo

En primer lugar, se obtiene un valor de R^2 de 0,879 aproximadamente, lo que implica que el modelo desarrollado explica casi el 88% de la variabilidad de los datos. Expresado de otro modo, el 88% de la variabilidad en la variable dependiente, que es el precio de los inmuebles, puede ser explicada por las variables independientes, o atributos, incluidas en el modelo. A priori, se trata de una métrica que indica un buen ajuste del modelo, aunque debe considerarse juntamente con otras; R^2 puede ser sensible tanto al número de variables como a los valores atípicos. El error absoluto medio y el error cuadrático medio calculan la diferencia entre los valores predichos y los reales, aunque el segundo penaliza especialmente los errores mayores, por lo que es más sensible a los valores atípicos o extremos; la diferencia entre el valor predicho y el valor real se eleva al cuadrado previamente a realizar la media. El valor obtenido en la evaluación para esta métrica es $2{,}65 \times 10^{10}$. El error absoluto medio, por el contrario, es más fácilmente interpretable, puesto que representa el tamaño promedio de los errores en la misma escala que la variable objetivo, que en este caso son unidades monetarias (euros). Cuanto menor es dicha métrica, menores son los errores en las predicciones, por lo que el

ajuste del modelo es mejor. El valor de la métrica obtenido es 80.335,14€, que puede ser expresado en unidades monetarias. Cabe precisar que las métricas de ajuste son consistentes con las obtenidos en la optimización empleando la validación cruzada, puesto que apenas divergen de estas últimas. Debido a que las métricas obtenidas en la optimización se extraen a partir del conjunto de datos de entrenamiento, se aprecia consistencia en la aplicación del modelo entre el conjunto de entrenamiento y el conjunto de prueba.

Sin embargo, para poner en contexto la medida de errores absolutos, conviene considerar el cálculo del MAE relativo. Para ello, se debe dividir la métrica de ajuste por la media de los precios observados del conjunto de prueba, lo que facilita la comparación con otros métodos de tasación, puesto que relaciona la métrica con el rango de los valores observados. Al dividir por el conjunto de datos que no se ha empleado en el entrenamiento se obtiene un enfoque de cómo se comportaría el modelo con nuevos datos. Como se puede comprobar en la Figura 15, el valor del MAE relativo es del 15% aproximadamente. Ello permite interpretar que, de media, las predicciones del modelo se desvían aproximadamente un 15% del valor real de los precios. Para determinar si esta desviación puede considerarse aceptable, es necesario realizar una comparación con otros modelos u herramientas de tasación.

Si en lugar de emplear un modelo de aprendizaje automático para tasar una vivienda, lo presumible es recurrir a empresas tasadoras oficiales. Aunque la información respecto al error de dichas compañías es escasa, se calcula que el valor de tasación y el precio real de la vivienda no coincide en aproximadamente el 80% de los casos[27]. Si bien este dato no ofrece información sobre la magnitud de la desviación del error, se estima que las tasaciones oficiales de viviendas se encuentran, de media, un 10% por encima del precio real de venta[27]. Por tanto, se puede considerar que el error del modelo es más que aceptable, dado que existe una mínima diferencia del 5% respecto al error medio de las tasaciones oficiales. Además, estas empresas destinan recursos de manera individualizada, como las visitas físicas, para cada una de las viviendas tasadas, por lo que gozan de carácter oficial homologado por el Banco de España y se consideran el método más riguroso de tasación.

Por otro lado, también resulta procedente realizar una comparación con las tasadoras en línea gratuitas, que permiten obtener el valor de una vivienda en cuestión de segundos o minutos. La información que se dispone al respecto de este tipo de tasaciones y su ajuste señala un margen de error que puede alcanzar el 85%, aunque se sitúa de media en el 42,2%[28]. El

modelo de aprendizaje automático desarrollado excede claramente el ajuste de este tipo de tasadoras que, a diferencia de las tasadoras oficiales, no tienen validez a efectos de garantía hipotecaria y no están sujetas a la supervisión del Banco de España.

Una vez impresas por pantalla las métricas de ajuste, se procede a realizar la representación gráfica de la dispersión. También se representa la línea de regresión entre los valores predichos, en el eje de las ordenadas, y los valores reales, en el eje de las abscisas. La figura se exporta en formato .PNG, y su visualización es la que se muestra a continuación.

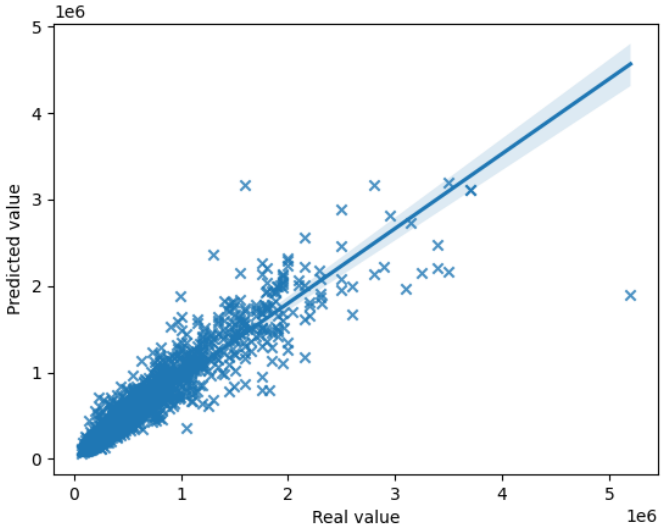


Figura 16. Representación gráfica de la dispersión

Cada inmueble en el conjunto de prueba se representa con una *x* en el gráfico, y permite visualizar cuán alejado está cada predicción de su valor real. Una manera simplificada de interpretarlo es considerar que la proximidad de cada instancia a la recta de 45° indica su grado de ajuste; cuanto más cerque esté cada punto de datos a la línea de igualdad, mayor será su ajuste. Para aquellos puntos que estén por encima de la diagonal se habrá predicho un valor superior al real, y para aquellos que estén por debajo se habrá predicho un valor inferior al real. Se puede apreciar como la dispersión es mayor cuanto mayor es el valor real del inmueble. Este hecho resulta razonable dado que la cantidad de inmuebles a la venta con un valor alejado de la mediana es menor, por lo que el modelo dispone de menos observaciones con las que entrenar. En el rango hasta el millón de euros (1e6) la dispersión es menor en valor absoluto; el modelo dispone de un número de observaciones mayor para este rango.

4.3. Predicción del modelo

Una vez se ha entrenado y evaluado el modelo, se puede emplear para que realice la predicción del valor de una vivienda con las características deseadas. Para ello, en primer lugar es necesario obtener la lista de ficheros que contienen el conjunto de datos empleado en el entrenamiento y la evaluación, del mismo modo que se hace anteriormente empleando la función *load_data*() y creando un objeto de tipo *DataFrame*. Este hecho es importante al ser necesario para poder realizar la codificación de variables categóricas en binarias. Además, aunque no se realiza en este trabajo, es posible incluir uno o más ficheros en *SALE_PREDICT_DATA_FILES* que contengan aquellos inmuebles cuyos precios de mercado hubiera que predecir, siempre y cuando mantuvieran el mismo formato que el conjunto de datos obtenido desde Idealista.

El siguiente paso es definir las características del inmueble que se desea tasar, que en un primer momento se realiza creando un *DataFrame* que contiene las columnas pertinentes y pasándole un diccionario en el que se incluyen el nombre de la columna como clave y el atributo del inmueble a tasar como valor. El precio ficticio que se establecerá en las características del inmueble a tasar será especialmente bajo, de manera que sea posible realizar un filtrado de manera sencilla que permita obtener como salida la tasación realizada por el modelo. El objeto creado se concatena con el *DataFrame* creado anteriormente. Cabe destacar que, aunque en este trabajo se realiza la predicción de precios de manera individual, se deja abierta la opción de incluir las características de varios inmuebles simultáneamente. Como ejemplo, se introduce un hipotético inmueble en el barrio de El Poblenou, cuyas características se muestran a continuación.

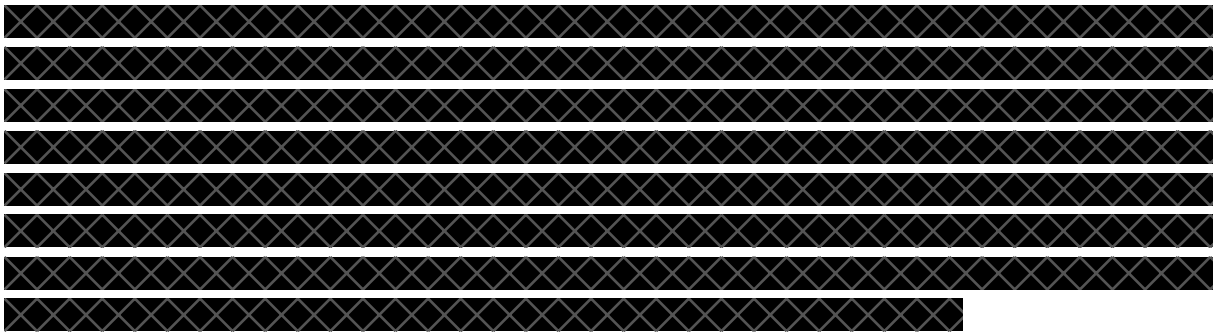


Figura 17. Atributos de la vivienda de ejemplo a predecir

Una vez obtenidas las características del inmueble cuyo precio se desea predecir, se debe realizar la limpieza de los datos mediante la función descrita en el Anexo 6, *clean_up_data*(). También es necesario realizar el cálculo del precio por m^2 medio de los barrios llamando a la función descrita en el Anexo 9, *compute_neighborhood_m2_price*(). Una vez se dispone del diccionario con los precios medios por cada barrio, el conjunto de datos a predecir y el modelo entrenado, se realiza la llamada a la función en la que se realizará la predicción. Esta función se denomina *predict_price*() y puede hallarse en el Anexo 12. En primer lugar, la función llama a *extract_features*(), descrita en el Anexo 10, que extrae del conjunto de datos que se le pasa los atributos en el formato adecuado, sustituyendo la variable categórica que es el barrio por el precio medio del m^2 del mismo y convirtiendo el resto de variables categóricas en binarias mediante *one-hot-encoding*. Una vez se dispone de los atributos, se elimina la columna correspondiente al precio, puesto que es la que se debe predecir, y se devuelve la predicción del modelo pasándole dicho conjunto de atributos en formato *DataFrame*.

A continuación, se genera una lista donde se incluye, una por una, cada vivienda que se encuentra en el *DataFrame* devuelto por la función *predict_price*(). Para cada vivienda predicha, se incluye el precio ficticio establecido. De esta manera se puede realizar una ordenación ascendente y así obtener el primer elemento fácilmente, cuyos atributos se han introducido en el código. Además, se incluye el precio predicho y la posición del inmueble en la lista. Finalmente, se obtiene e imprime por pantalla el precio predicho por el modelo para el inmueble cuyas características se han introducido. Respecto al ejemplo de vivienda ofrecido anteriormente, y previamente a realizar una interfaz gráfica, el resultado obtenido por consola se muestra de la siguiente manera:

```
321310€
```

Figura 18. Precio predicho de la vivienda de ejemplo

5. Interfaz gráfica

Para un potencial usuario, puede resultar complejo realizar la inserción de los datos del inmueble directamente en el código. Tener que identificar el lugar donde sustituir los atributos y el formato que deben tomar puede conducir a errores, modificaciones de código indeseadas y complejidad de uso para el usuario. Por ello, en este capítulo se plantea realizar una interfaz del programa que pueda ser utilizada por cualquier usuario de manera relativamente sencilla que le permita incluir las características del inmueble cuyo precio desee predecir.

Un primer planteamiento en el desarrollo de una interfaz gráfica es que los valores se introduzcan por pantalla, en la consola del propio ordenador. Sin embargo, el desarrollo y despliegue de esta opción, aunque resulta menos costoso a nivel de tiempo y esfuerzo, puede seguir resultando complejo para algunos usuarios no habituados a emplear una terminal o consola. Por ello, se decide crear un GUI, o *graphic user interface*, que no deja de ser una interfaz gráfica para el usuario. De esta manera, el usuario medio puede emplear la interfaz, que permite de manera intuitiva introducir o rellenar la información necesaria para realizar la tasación. Esta interfaz también puede controlar para ciertos errores, y obtener la información necesaria para la consecución de la tasación.

Para llevar a cabo la interfaz, se ha empleado la librería *Tk Interface* de Python, que permite de manera sencilla introducir opciones desplegables, botones de opciones, campos de entrada y *checklists*, entre otros. La información extraída se puede almacenar y emplear en realizar los cálculos necesarios o realizar predicciones, como es el caso.

Por tanto, en primer lugar, es necesario crear una instancia nueva de *Tk Inter*, que se representa como una ventana nueva, y que tiene como título *Real state appraisal*. A continuación, se crean las listas de aquellos atributos categóricos que se ofrecerán en listas desplegables. Estos son el barrio, el tipo de propiedad, la condición del inmueble, el tipo de calefacción y la accesibilidad de la que disponen. Es importante destacar que se ofrecen como opciones los únicos valores que se encuentran en alguna instancia del conjunto de datos obtenido previamente, puesto que si se introdujera un inmueble que tuviera una característica desconocida para el modelo no sería posible predecir adecuadamente su precio. Por ejemplo, sería extremadamente difícil para el modelo tasar correctamente una propiedad que fuera un

chalet adosado si en el conjunto de datos con el que se entrena no apareciera ninguno. Por otro lado, también se añade la opción de que no se dispongan datos ni para el tipo de calefacción ni para la disponibilidad de accesibilidad, puesto que se trata de atributos categóricos para los que, de manera frecuente, no se dispone de información. Se crean entonces las listas desplegables, que permiten seleccionar entre una de varias opciones para cada uno de los cinco atributos categóricos. Las opciones escogidas se guardan en una lista a la que se accede posteriormente. A continuación, se muestra un ejemplo de las listas desplegables en el que se ha presionado en las opciones de accesibilidad.

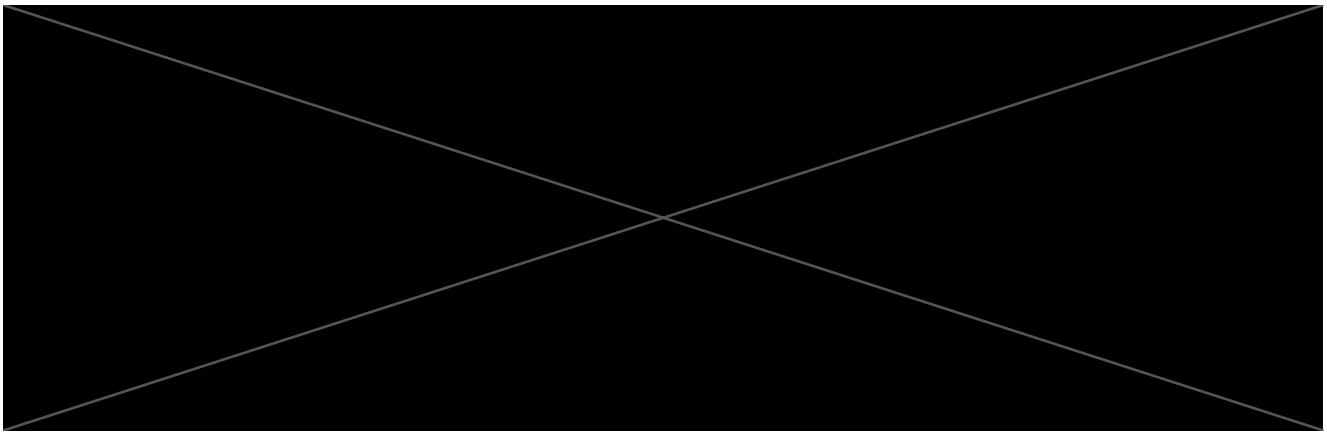


Figura 19. Ejemplo de listas desplegables

Seguidamente, se generan los siete campos de entrada que permiten introducir los valores numéricos para las variables numéricas de superficie, número de habitaciones, número de baños, año de construcción, planta, precio del parquin y consumo de energía. Adicionalmente, se ofrece una anotación en la que se indica que, con el objetivo de indicar que no se dispone información para alguno o varios de los campos, se debe introducir el valor -99. Es frecuente que para muchos inmuebles no se disponga de información relativa al consumo energético, año de construcción o directamente no tenga plaza de parquin asociado. Los valores resultantes de cada campo se guardan en una lista para poder ser empleados posteriormente. Como se explicará más adelante, es necesario en este caso controlar que se introduce un valor con caracteres numérico, y no de otro tipo. También es necesario asegurar que la información ha sido rellenada para cada uno de los atributos, ya sea mediante la indicación de que no se dispone de datos al respecto, insertando el valor -99, o mediante el valor numérico que corresponde. El conjunto de campos queda de la siguiente manera.

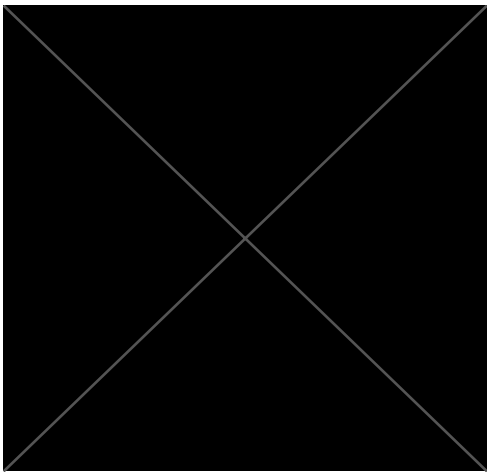


Figura 20. Ejemplo de campos de entrada

Por tanto, únicamente queda incluir botones de opciones, que permiten escoger una entre varias opciones. Para aquellos atributos originalmente binarios, que pueden tomar valores de *True* o *False* (y *None*), se aplican los mismos botones para cada uno, en forma de lista. Los atributos en cuestión son, si el inmueble es exterior, tiene balcón, terraza, jardín, zonas verdes, piscina, trastero, ascensor, armarios empotrados, aire acondicionado y si tiene orientación norte, este, sur u oeste. En este caso también se almacena el resultado de cada atributo en una lista, codificado de 1 a 3 para cada uno, que se decodificará posteriormente. El valor de 1 corresponde a *True*, 2 *False* y 3 a *None*. Aunque el atributo es originalmente binario, se ofrece la posibilidad de informar respecto a la no disposición de información. El ejemplo tratado anteriormente toma forma de la siguiente manera.

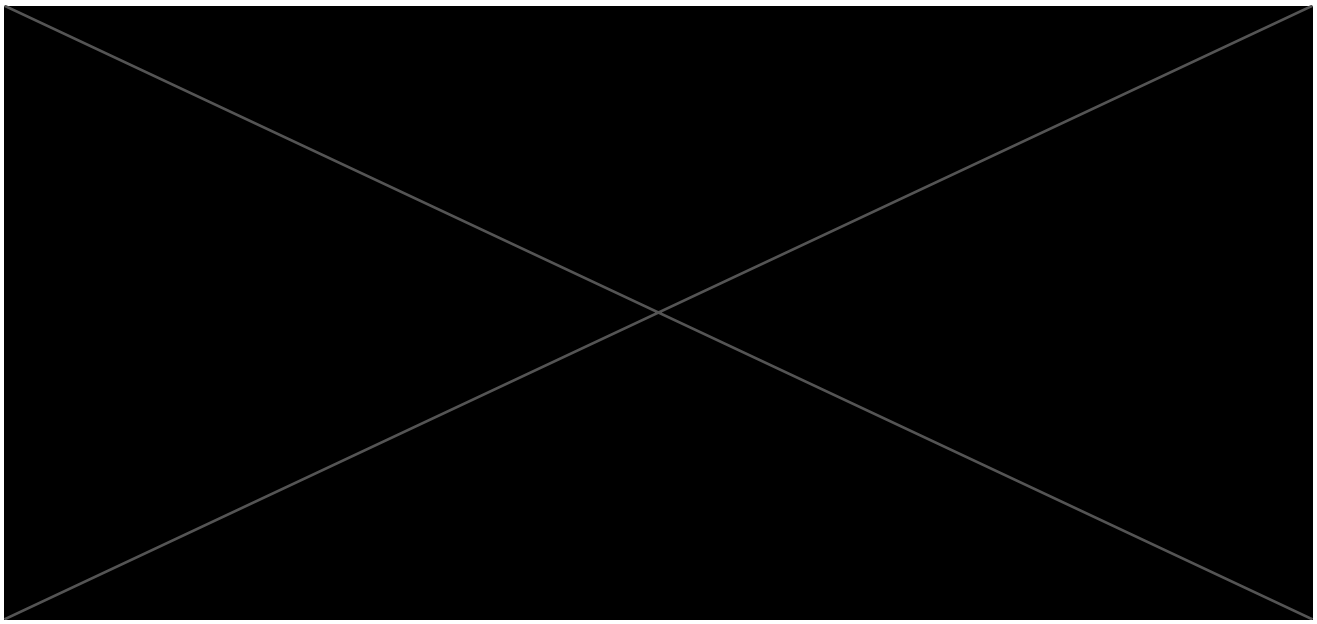


Figura 21. Ejemplo de botones de opciones

Una vez creadas todas las herramientas para obtener la información de la vivienda cuyo precio se desea predecir, se crea el botón que accionará la predicción y la muestra por pantalla. Este botón tiene el texto *Appraise*, o tasar en castellano. Al accionarlo, se llama a la función *calculate*(), que puede encontrarse en el Anexo 13 de este trabajo. En la función se ha incluido parte del código que se encontraba en la función *main*(), disponible en el Anexo 14, como la formación del *DataFrame* que incluye los ficheros con los posibles inmuebles a predecir. Posteriormente, se obtienen los datos de cada atributo que se han introducido en la interfaz para poder darles un formato que el modelo pueda usar. En el caso de los atributos categóricos, conviene convertir los valores obtenidos en formato *string*. Para la información numérica, conviene hacer la conversión a *float*, siempre y cuando el valor obtenido sea distinto a -99, que implica que no se dispone del valor. Finalmente, para los atributos originalmente booleanos se debe decodificar en función del valor de 1 a 3 asignado anteriormente. En esta parte del código también se incluyen métodos de control. En caso de no seleccionar alguno de los elementos de cada desplegable, se muestra por pantalla un mensaje de aviso, en el que se indica que el elemento en cuestión debe ser seleccionado para poder realizar la tasación. De manera similar, si alguno de los campos de entrada para valores numéricos queda vacío, o si se introduce algún carácter no numérico, se muestra por pantalla la información con el campo que debe ser introducido o corregido. Lo mismo ocurre con los botones de opciones. En caso de no seleccionar alguno, se muestra por pantalla. A continuación, se muestran tres mensajes de error, para cada uno de los *widgets*.

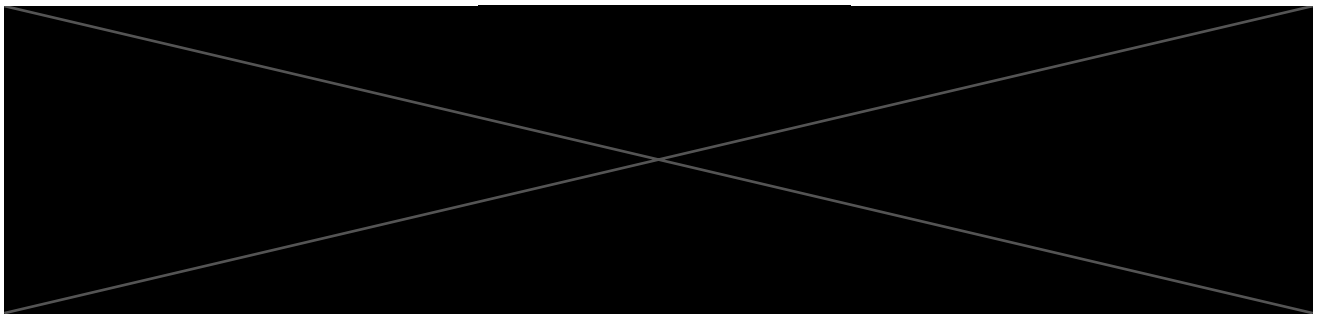


Figura 22. Ejemplos de mensajes de error

Es necesario formatear las características de la información obtenida para poder realizar la limpieza de los datos, cuya llamada se incluye en esta función *calculate*(). También se realiza la predicción del modelo, o más exactamente la llamada a la función de predicción, lo que permite mostrar un mensaje informando con el precio predicho de la siguiente manera.

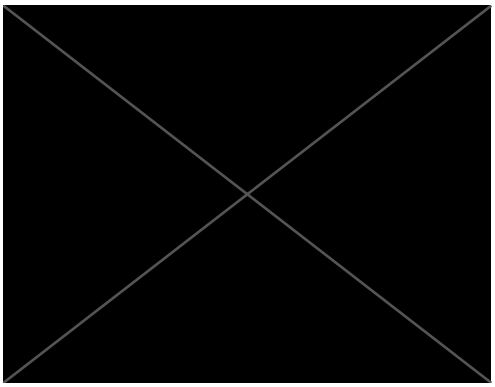


Figura 23. Ejemplo de mensaje con el resultado

Tras finalizar la función encomendada al botón, se produce un bucle que actualiza y realiza la predicción con los datos obtenidos cada vez que se vuelve a presionar. Por ello, es posible cambiar las características del inmueble tantas veces como el usuario desee, obteniendo la predicción realizada por el modelo. La petición de tasación con diferentes características se puede realizar indefinidamente hasta que se cierra la ventana de la interfaz. La vista general tiene el siguiente aspecto empleando el ejemplo definido en el apartado anterior, y que, en este caso concreto, da como resultado una tasación de 321.309,97€.

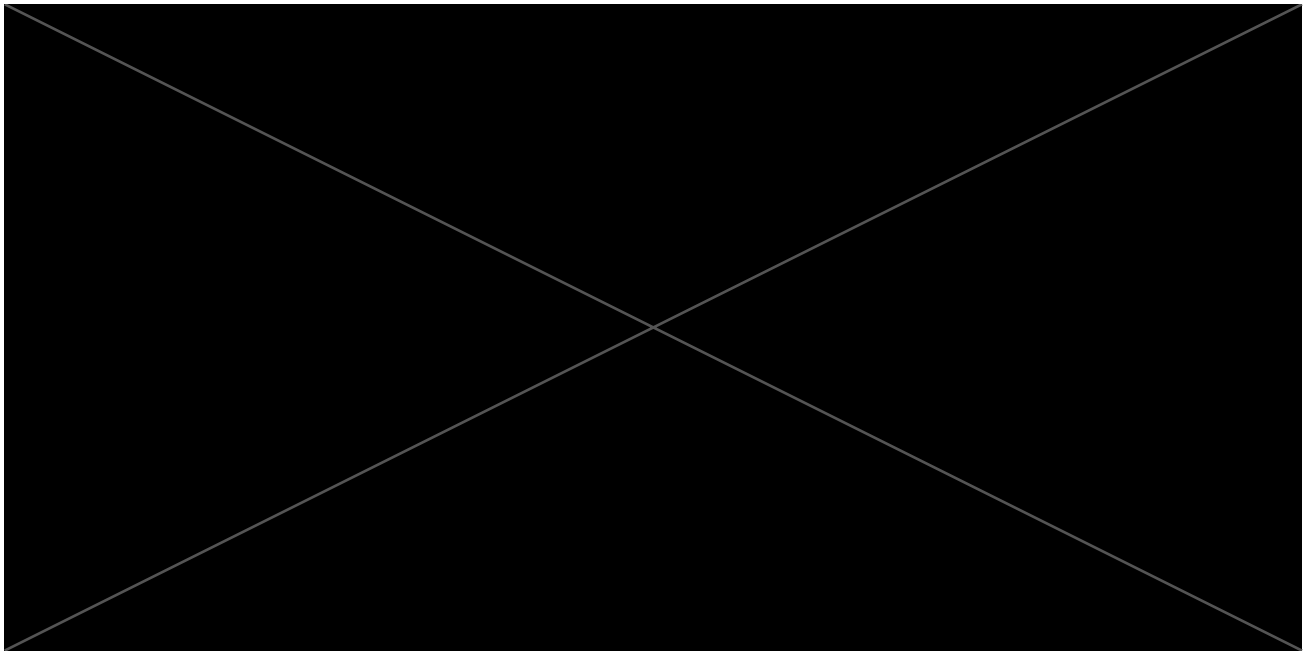


Figura 24. Vista general de la interfaz del programa

El código previo, las librerías importadas, y posterior, el bloque principal, a las funciones descritas en este trabajo se pueden encontrar en el Anexo 15.

6. Conclusiones

El objetivo principal de este trabajo ha sido lograr desarrollar una herramienta de predicción del valor viviendas situadas en la ciudad de Barcelona que pudiera resultar de utilidad para los distintos agentes del mercado inmobiliario. Para ello, ha sido necesario desarrollar un algoritmo basado en inteligencia artificial, y más concretamente en aprendizaje automático, con el grado de optimización suficiente como para ser considerado útil. Además, ha sido necesario obtener un conjunto de datos que contuviera los atributos de los inmuebles necesarios para poder determinar su valor, así como desarrollar una interfaz gráfica sencilla pero intuitiva para el usuario.

El proceso seguido para desarrollar este trabajo ha consistido, primeramente, en realizar una serie de análisis que permitieran concretar los pasos a seguir. En primer lugar, se han identificado los atributos necesarios para predecir el precio de los inmuebles. A continuación, se ha determinado la fuente más adecuada desde la que obtener el conjunto de datos, que ha sido el portal inmobiliario de Idealista. Una vez identificada la mejor fuente, se ha tenido que definir el método más conveniente para obtener dicho conjunto de datos, resultando ser una herramienta de *web scraping*. Una vez analizado qué, cómo y de dónde obtener los datos, se ha realizado un análisis del modelo de aprendizaje automático más adecuado para la tarea fijada como objetivo. El algoritmo idóneo que ha sido seleccionado es XGBoost, que cuenta con extensa literatura al respecto y ha sido ampliamente utilizado para predecir valores como el de la vivienda.

Tras realizar los análisis pertinentes y definir el proceso a efectuar, se ha configurada la herramienta para extraer el conjunto de datos de todas las viviendas a la venta en la ciudad de Barcelona del portal Idealista. Para ello, ha sido necesario realizar la extracción en dos partes, por lo que se ha utilizado un código que permitía juntar dos ficheros .CSV en uno conservando una única cabecera. A continuación, se ha realizado el tratamiento previo de los datos, así como su limpieza, para poder ser empleados en el modelo escogido. También se ha realizado un breve análisis estadístico sobre el conjunto de datos.

Con la información sobre los inmuebles ya tratada, se ha podido desarrollar el modelo de aprendizaje automático. Para ello, se ha creado, optimizado y evaluado, siendo las métricas

resultantes más que satisfactorias; la métrica más relevante, el error medio absoluto relativo, ha sido de un 15% aproximadamente. Este error es muy cercano al error medio de las tasadoras oficiales, un 10%, y sustancialmente inferior al de las tasadoras en línea gratuitas, un 42,2%. Tras obtener una métrica que permite validar el modelo, se ha llevado a cabo la implementación para realizar las predicciones. En un primer momento y a modo de ejemplo, se ha realizado la predicción del valor de un inmueble introduciendo sus atributos directamente en el código.

Finalmente, se ha desarrollado una interfaz gráfica en Python, que permite a un usuario medio obtener una predicción a partir de los atributos del inmueble introducidos por pantalla. Dicha interfaz dispone de mecanismos de control en caso de no completar información requerida, y puede ejecutarse de manera continua hasta que el usuario cierre la ventana.

Por tanto, se pueden dar por cumplidos los objetivos de este trabajo al obtener una interfaz gráfica que permite predecir el valor de un inmueble a partir de un modelo de aprendizaje computacional con un error medio absoluto relativo próximo al de los métodos de tasación más confiables. De hecho, este trabajo puede tener implicaciones prácticas directas, ya que podría ser empleado de aquí en adelante por agentes del mercado inmobiliario que desearan valorar una vivienda. Ello tendría un impacto positivo directo en la eficiencia del mercado, puesto que reduciría la desinformación de agentes que quizá no disponen del conocimiento suficiente para valorar inmuebles, ya se su intención comprar o vender. Además, puede resultar de gran utilidad para el uso personal, dado el desconocimiento específico sobre el sector. Sin embargo, aunque los objetivos pueden considerarse cumplidos, existen varias vías tanto de mejora del propio trabajo como de investigación y desarrollo. Respecto al primer caso, sería positivo realizar una evaluación del modelo en mayor profundidad. Aunque se ha procurado desarrollar un modelo en el que no se produjera *overfitting*, o sobreajuste, y no se han detectado elementos que indiquen su existencia, convendría confirmar que no se produce dicha circunstancia. En caso de darse, sería necesario tomar las medidas adecuadas para minimizarlo en la medida de lo posible. Además, parece factible minimizar el error obtenido en la optimización del modelo, dado que pueden darse combinaciones de hiperparámetros que no han sido contempladas o incluso emplear algoritmos de aprendizaje computacional cuyo ajuste para predecir valores de viviendas sea empíricamente mejor. En cuanto a las posibles vías de desarrollo, existen multitud de oportunidades a partir de este trabajo. En primer lugar,

se podría obtener un conjunto de datos histórico, extrayendo la información de los anuncios del momento de manera periódica y que permitiera tener en cuenta la fecha de la extracción. De esta manera, el conjunto de datos iría aumentando y sería posible identificar tendencias que mejoraran la calidad de la predicción. Adicionalmente, podría modificarse el modelo para ser empleado no solo en valoraciones de inmuebles, sino también de alquileres, así como adaptarse a otros lugares geográficos, ya fuera dentro o fuera de España. Estos servicios, llegado el momento y con el desarrollo adecuado, podría comercializarse. Una de las opciones sería vender la licencia de uso a tasadoras oficiales o empresas y agentes inmobiliarios, que podrían usarlo para corroborar o mejorar sus tasaciones individualizadas. Otra alternativa sería ofrecerlo como un servicio en una página web propia, en la que cualquier usuario podría hacer uso de la aplicación de predicción en línea a cambio de un precio determinado, u otros modelos de negocio. Idealmente, con el suficiente desarrollo y con la garantía de un error mínimo sostenido, podría plantearse obtener la homologación oficial otorgada por el Banco de España, y que por tanto otorgara validez de garantía hipotecaria a cada tasación.

A título personal, este trabajo ha resultado uno de los principales retos del Grado de Ingeniería Informática. Pese a tener una idea desde antes de empezar el semestre, la información y conocimiento respecto al tema tratado y como desarrollarlo era mínima. Ello ha implicado una sobrecarga de trabajo que no había sido planificada, dado que a la propia ejecución del trabajo se le debía sumar el tiempo invertido en aprender y averiguar sobre ámbitos muy diversos. Por ello, la gestión del tiempo y la aplicación metodológica han resultado de las cualidades más reforzadas. A nivel práctico, los dos principales retos han sido la obtención del conjunto de datos y su tratamiento. Dicha extracción ha sido diseñada específicamente para este trabajo, sin disponer de excesivas herramientas externas ya desarrolladas además del *web scraper*. Del mismo modo, el tratamiento de los datos para poder ser integrados en el modelo ha requerido generar código totalmente individualizado. Sin embargo, la perspectiva positiva de desarrollar un trabajo de programación de este alcance, y con el plazo de tiempo disponible, es la profundización en el conocimiento técnico. Este conocimiento práctico y teórico abarca ámbitos tan útiles y de interés como el aprendizaje computacional, o *Data Science* de manera amplia, o las interfaces gráficas.

7. Glosario

- *Outliers*: En estadística también se denominan valores atípicos o extremos y son valores excepcionalmente altos o bajos. El origen de la desviación del resto de valores puede ser debido tanto a la medición de los datos como a las propias características de estos. Suelen requerir un tratamiento especial debido al efecto distorsionador que tienen.
- *Hash map*: Una estructura de datos empleada para almacenar pares clave-valor eficientemente. Las claves son únicas y sirven para localizar el valor que almacena, lo que se puede realizar rápidamente, en tiempo constante. También se les denomina diccionarios o tablas de hash.
- Variable categórica: Variable que representa un atributo que puede tomar un solo valor, la categoría, de un conjunto de valores posibles.
- Variable binaria: Variable que puede tomar únicamente dos valores, 1 o 0, y suelen indicar la existencia, o no, de un atributo.
- *One-hot-encoding*: Técnica de procesamiento de datos que permite convertir variables categóricas en numéricas. Con esta técnica se crea una columna para cada categoría, que toma el valor 1 en caso de corresponder a esa fila y 0 en el resto de las columnas. Muchos algoritmos de aprendizaje automático requieren de este tipo de técnica al no poder procesar variables categóricas.
- Hiperparámetro: Parámetros ajustables de un modelo de aprendizaje que no se aprenden del conjunto de datos, por lo que se establecen antes y pueden ser optimizados.
- *Overfitting*: Circunstancia en la que un modelo de aprendizaje se ajusta en exceso a los datos de entrenamiento, de manera que no es posible generalizar. Cuando se da, el modelo predice bien los datos de entrenamiento, pero no datos nuevos. También se le denomina sobreajuste.
- *Widget*: Elemento básico de la interfaz que permite interaccionar al usuario con la aplicación.

8. Bibliografía

[1] Salvador, R. (2023) *La Venta de vivienda cayó un 10,2% en diciembre tras 21 Meses Consecutivos de aumentos.* La Vanguardia. Disponible en: https://www.lavanguardia.com/economia/20230217/8764054/ventas-vivienda-cayeron-10-2-diciembre-tres-21-meses-consecutivos-aumentos.html(Acceso: 19 de marzo de 2023).

[2] Idealista. (2022) *Barcelona es la Ciudad Más cara para comprar y Alquilar Vivienda, según ey.* idealista/news. Disponible en: https://www.idealista.com/news/inmobiliario/vivienda/2022/12/07/800465-barcelona-es-la-ciudad-mas-cara-para-comprar-y-alquilar-vivienda-segun-ey (Acceso: 19 de marzo de 2023).

[3] Iván. (2022) *El Mercado Inmobiliario de Barcelona en el Tercer Trimestre de 2022.* Bcn Advisors. Disponible en: https://www.bcn-advisors.com/el-mercado-inmobiliario-de-barcelona-en-el-tercer-trimestre-de-2022 (Acceso: 19 de marzo de 2023).

[4] Castán, P. (2019) *Las agencias inmobiliarias se duplican en Barcelona en apenas cuatro años*. El Periódico. Disponible en: https://www.elperiodico.com/es/barcelona/20190114/agencias-inmobiliarias-han-doblado-barcelona-apenas-cuatro-anos-7239340 (Acceso: 12 de abril de 2023).

[5] Pauné, M.M. (2015) *Un cuarto de siglo de aluminosis en Barcelona*. La Vanguardia. Disponible en: https://www.lavanguardia.com/local/barcelona/20151111/54438761913/aluminosis-barcelona-25-anos.html (Acceso: 12 de abril de 2023).

[6] Redacción. (2020) *El covid-19 impulsa la demanda de viviendas con terraza o estancias al aire libre*. La Razón. Disponible en: https://www.larazon.es/familia/20200607/26zfxr5eofctvm7jhm77c6kdvu.html (Acceso: 13 de abril de 2023).

[7] Consejo General del Notario. Disponible en: https://www.notariado.org/liferay/web/cien/inicio (Acceso: 14 de abril de 2023).

[8] Instituto Nacional de Estadística. Disponible en: https://www.ine.es/ (Acceso: 14 de abril de 2023).

[9] Open Data BCN. Disponible en: https://opendata-ajuntament.barcelona.cat/ (Acceso: 14 de abril de 2023).

[10] Habitaclia. Disponible en: https://www.habitaclia.com/ (Acceso: 15 de abril de 2023).

[11] Fotocasa. Disponible en: https://www.fotocasa.com/ (Acceso: 15 de abril de 2023).

[12] Idealista. Disponible en: https://www.idealista.com/ (Acceso: 15 de abril de 2023).

[13] Api de testigos. Disponible en: https://www.idealista.com/data/productos/desarrollo/api-de-testigos (Acceso: 15 de abril de 2023).

[14] Web Scraper. Disponible en: https://www.webscraper.io/ (Acceso: 16 de abril de 2023).

[15] Scraper. Disponible en: https://chrome.google.com/webstore/detail/scraper/mbigbapnjcgaffohmbkdlecaccepngjd (Acceso: 16 de abril 2023).

[16] Data Scraper. Disponible en: https://chrome.google.com/webstore/detail/data-scraper-easy-web-scr/nndknepjnldbdbepjfgmncbggmopgden (Acceso: 16 de abril de 2023).

[17] Y. Zhao, G. Chetty y D. Tran. "Deep Learning with XGBoost for Real Estate Appraisal". *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, Xiamen, China, 2019, pp. 1396-1401, doi: 10.1109/SSCI44817.2019.9002790.

[18] Stang, M., Krämer, B., Nagl, C. *et al.* "From human business to machine learning—methods for automating real estate appraisals and their practical implications". *Z Immobilienökonomie*, 2022, doi: 10.1365/s41056-022-00063-1.

[19] Guliker E, Folmer E y van Sinderen M. "Spatial Determinants of Real Estate Appraisals in The Netherlands: A Machine Learning Approach". *ISPRS International Journal of Geo-Information*, 2022, 11(2):125, doi: 10.3390/ijgi11020125.

[20] Kaggle Competitions. Disponible en: https://www.kaggle.com/competitions (Acceso: 17 de abril de 2023).

[21] Greenfield, Y. (2020) *Four ways teams win on Kaggle, Medium*. Towards Data Science. Disponible en: https://towardsdatascience.com/four-ways-teams-win-on-kaggle-50e62acb87f4 (Acceso: 17 de abril de 2023).

[22] Chen, T. y Guestrin, C. "XGBoost: A Scalable Tree Boosting System". *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785-794, doi: 10.1145/2939672.2939785.

[23] Distritos de Barcelona. (2023) *Wikipedia*. Disponible en: https://es.wikipedia.org/wiki/Distritos_de_Barcelona (Acceso: 3 de junio de 2023).

[24] uBlock Origin. Disponible en: https://ublockorigin.com/es (Acceso: 19 de abril 2023).

[25] Research Triangle Analysts. *Analytics Forward 2019 Keynote by Jordan Meyer – Research Triangle Analysts* [Archivo de vídeo]. Youtube. https://m.youtube.com/watch?v=kUHEDuwhZHM (Acceso: 28 de abril de 2023).

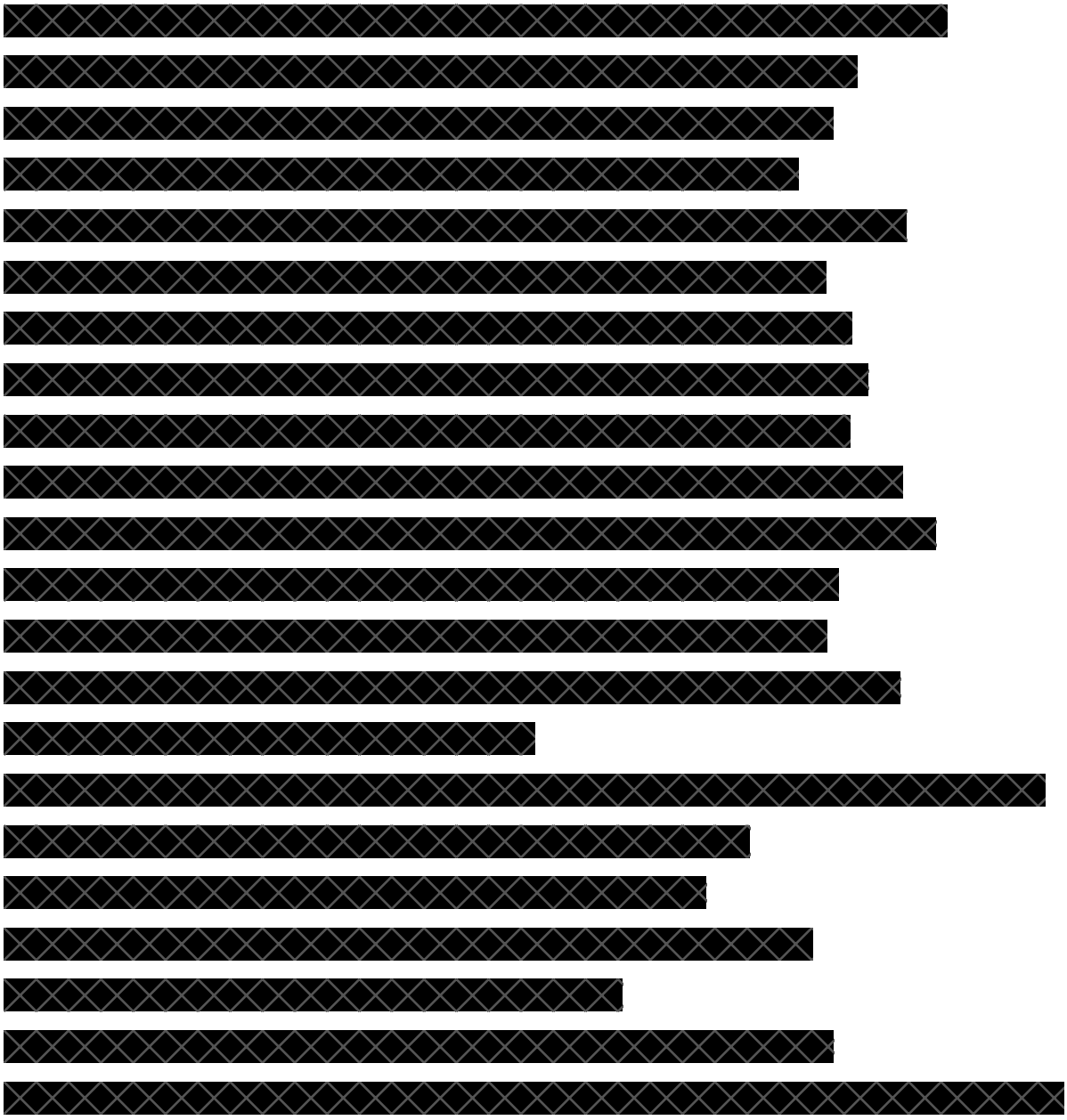
[26] Pedregosa, F. *et al.* "Scikit-learn: Machine Learning in Python". *JMLR 12*, 2011, pp. 2825-2830, 12(85):2825–2830.

[27] Sanz, E. (2019) *Al Comprar Casa, qué es mejor, ¿Una tasación superior o inferior al precio de venta?*. El Confidencial. Disponible en: https://www.elconfidencial.com/vivienda/2019-02-18/tasacion-banco-de-espana-hipoteca-compraventa_1832434/ (Acceso: 5 de junio de 2023).

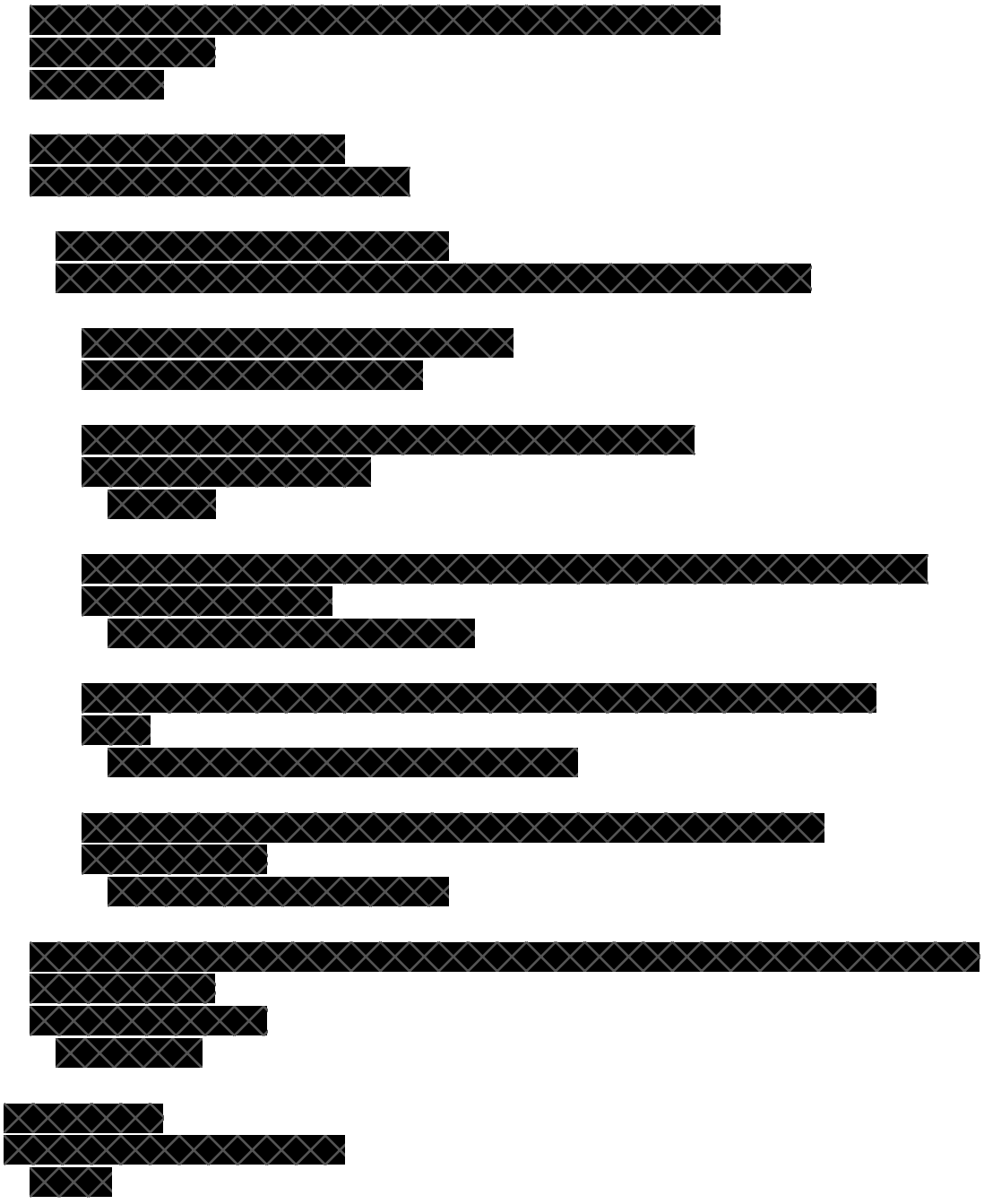
[28] Arquitasa. (2023) *Simulación tasación de inmuebles gratis*. Arquitasa. Disponible en: https://arquitasa.com/tasaciones-inmuebles-online/ (Acceso: 5 de junio de 2023).

9. Anexos

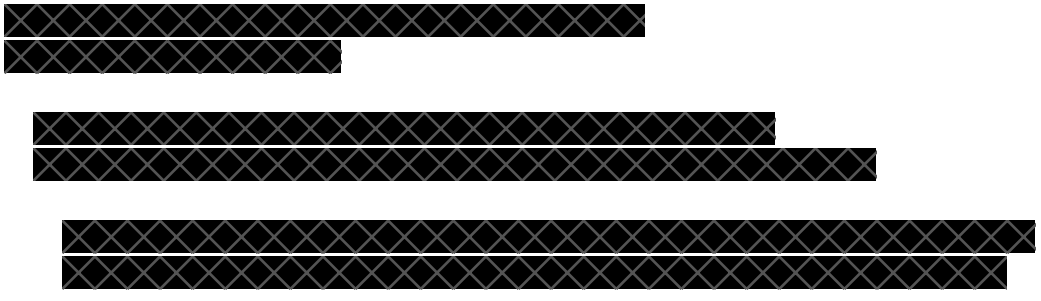
9.1. Anexo 1. Configuración del *Sitemap*



9.2. Anexo 2. Script unión-csv.py para la unión de ficheros .CSV

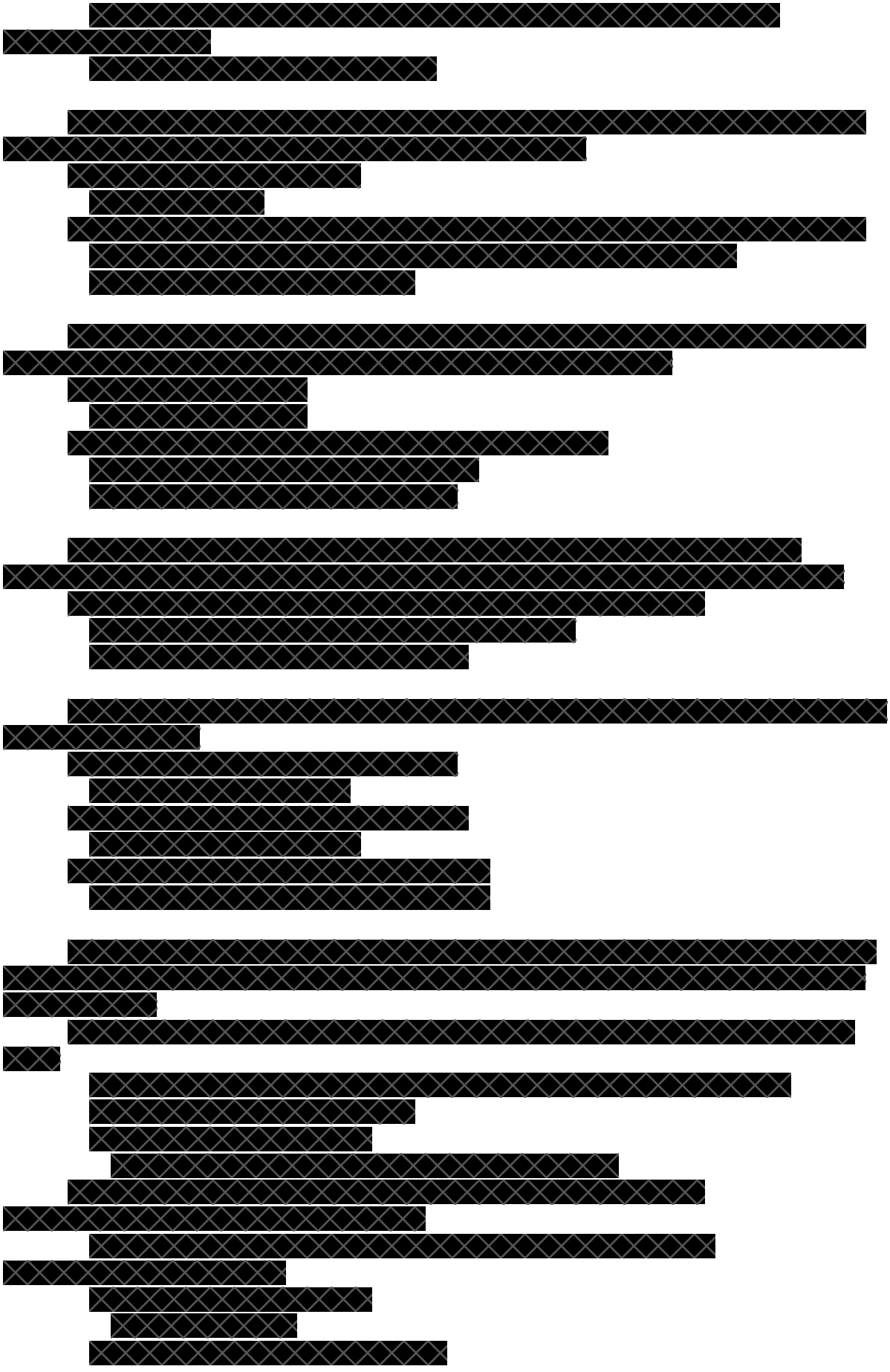


9.3. Anexo 3. Función *read_csv*()

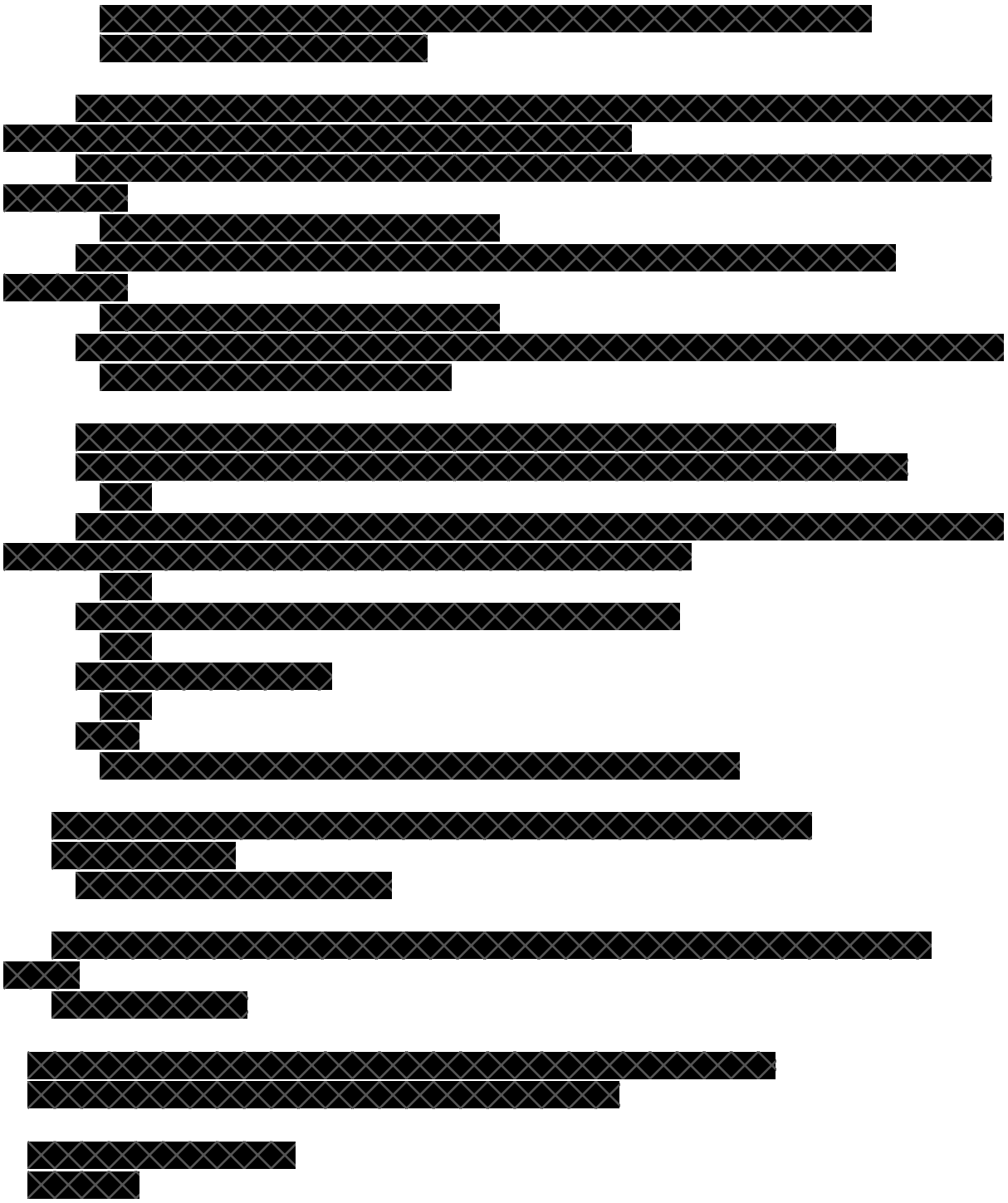


9.4. Anexo 4. Función *read_csv_with_headers*()

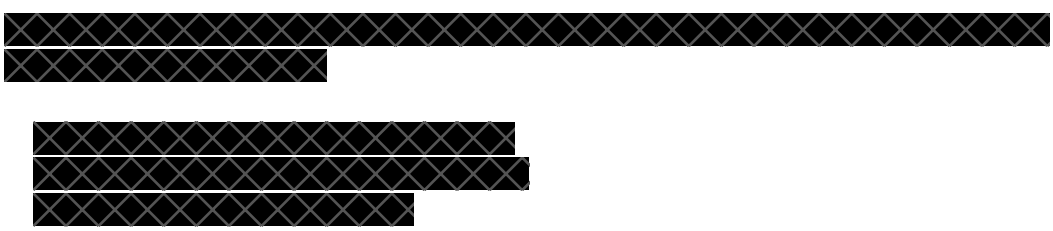
9.5. Anexo 5. Función *load_data*()

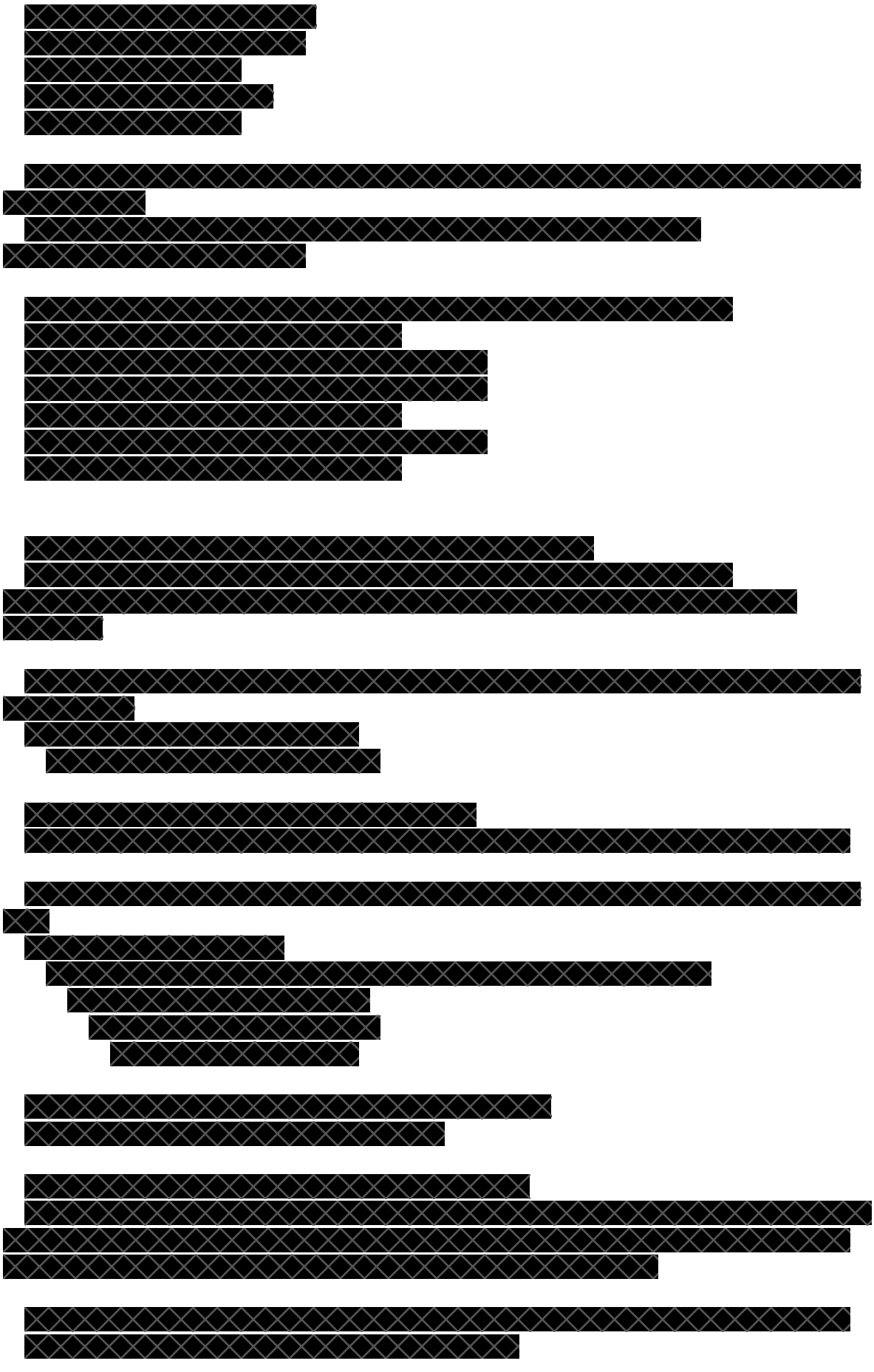


[redacted]



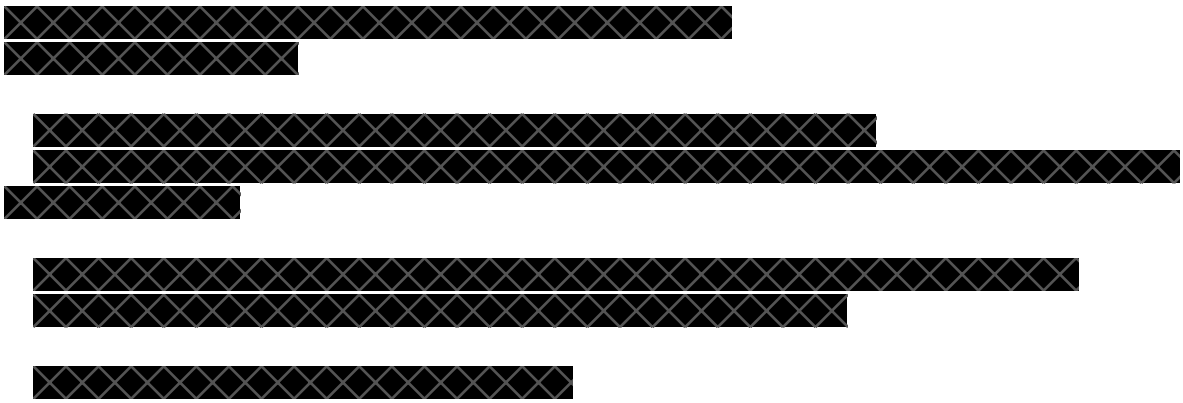
9.6. Anexo 6. Función *clean_up_data*()



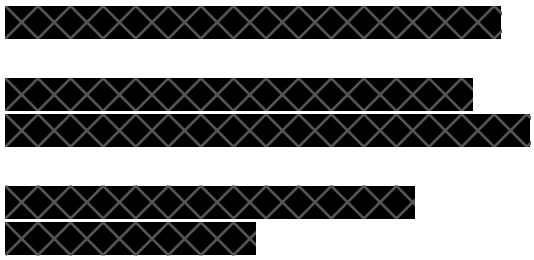


9.7. Anexo 7. Métodos de análisis y llamada a la función *clean_up_data*()

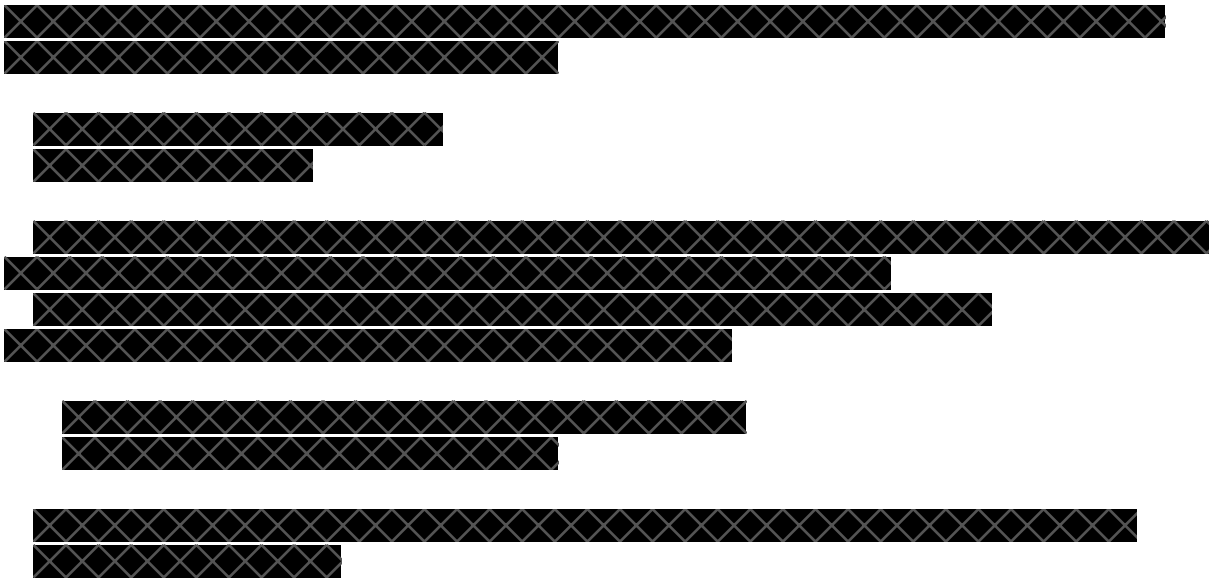
9.8. Anexo 8. Función *model_train*()



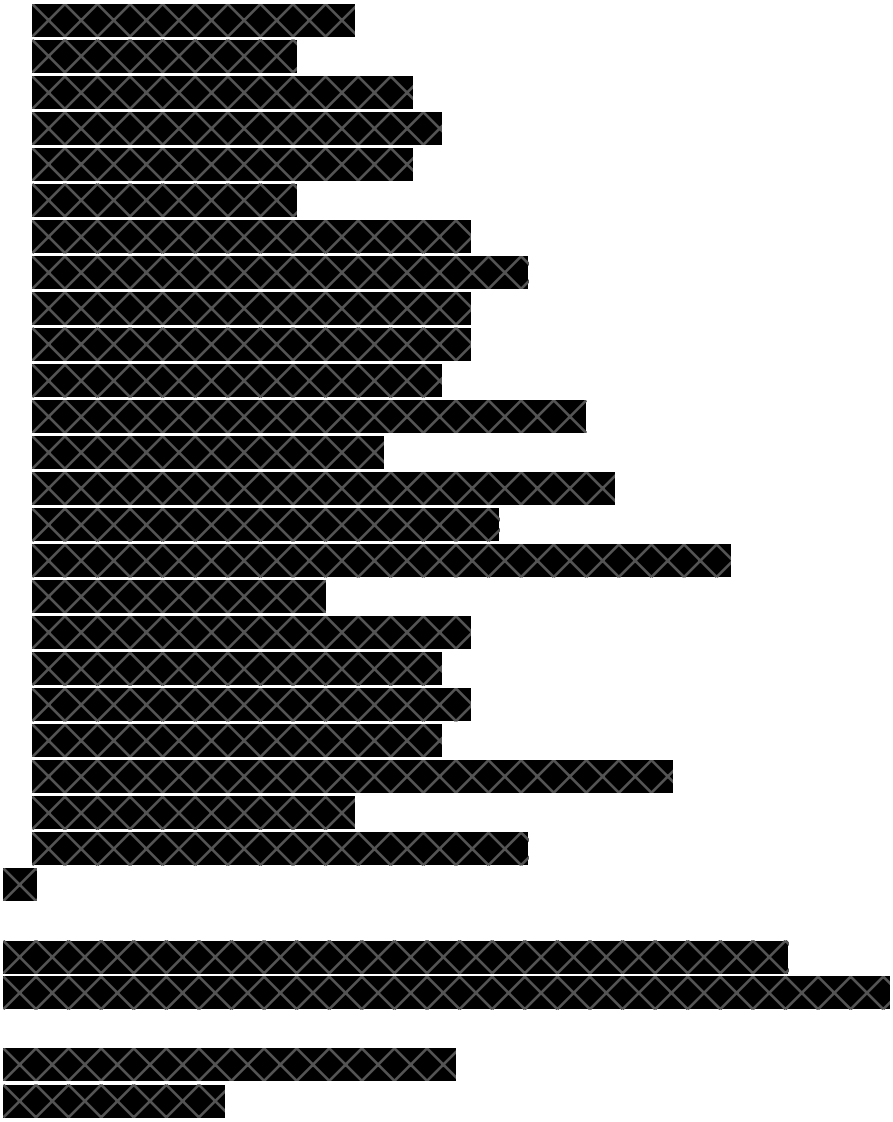
[redacted]



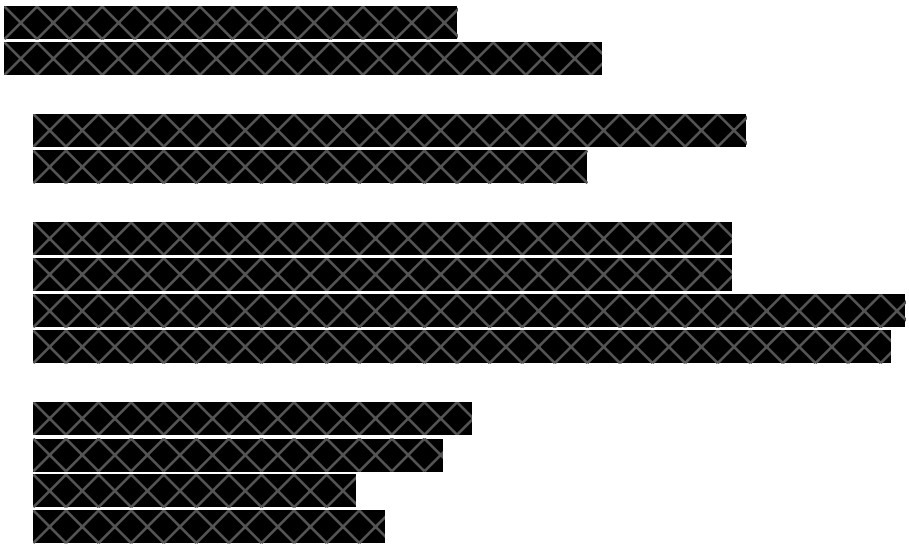
9.9. Anexo 9. Función *compute_neighborhood_m2_price*()



9.10. Anexo 10. Función *extract_features*()



9.11. Anexo 11. Función *evaluate*()



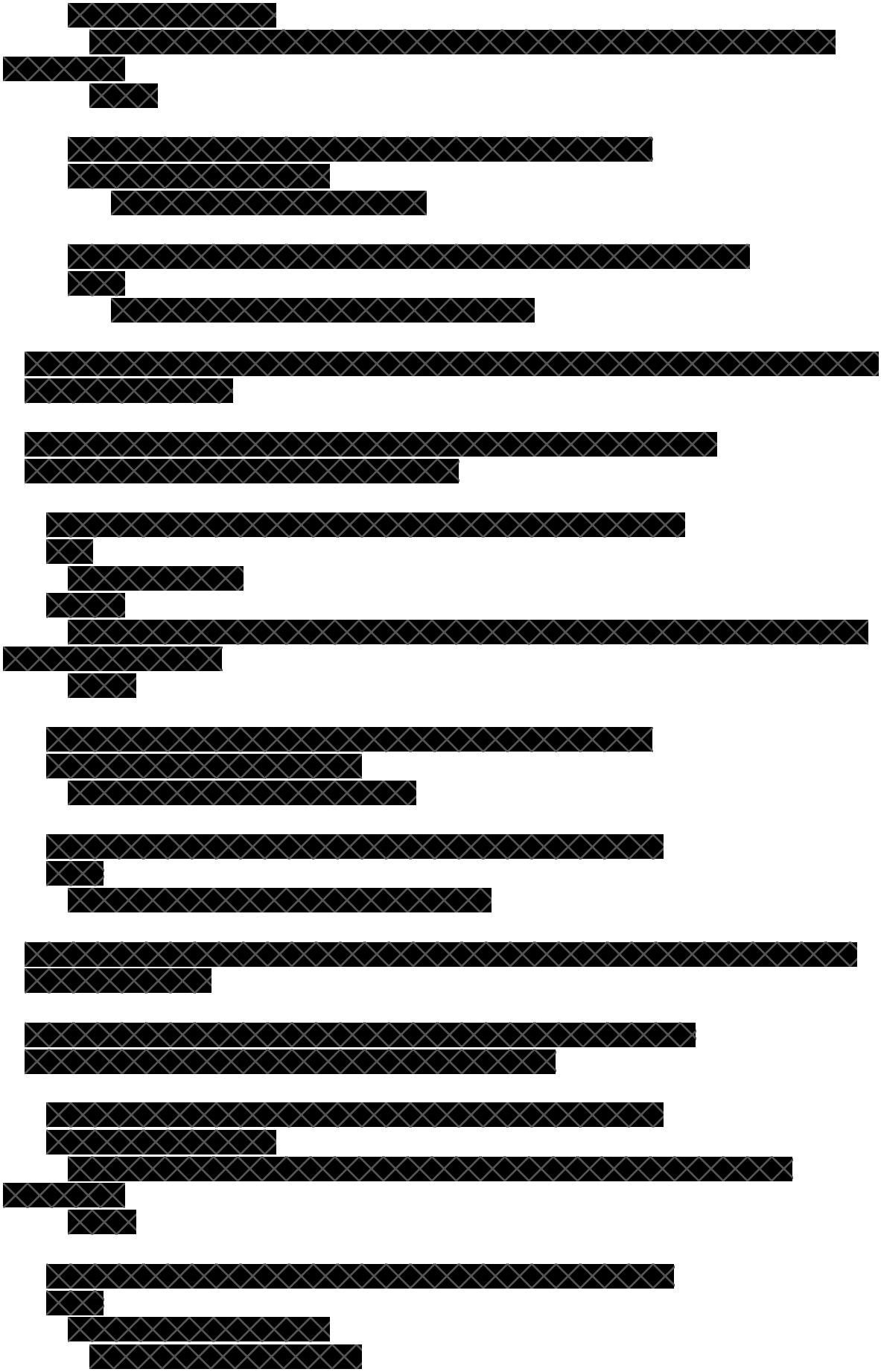
[redacted]

9.12. Anexo 12. Función *predict_price*()

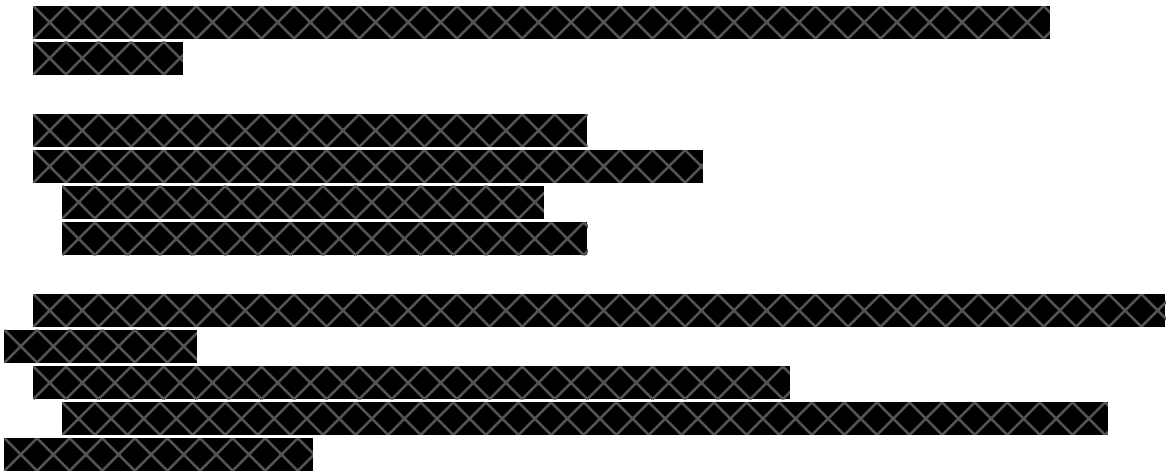
[redacted]

9.13. Anexo 13. Función *calculate*()

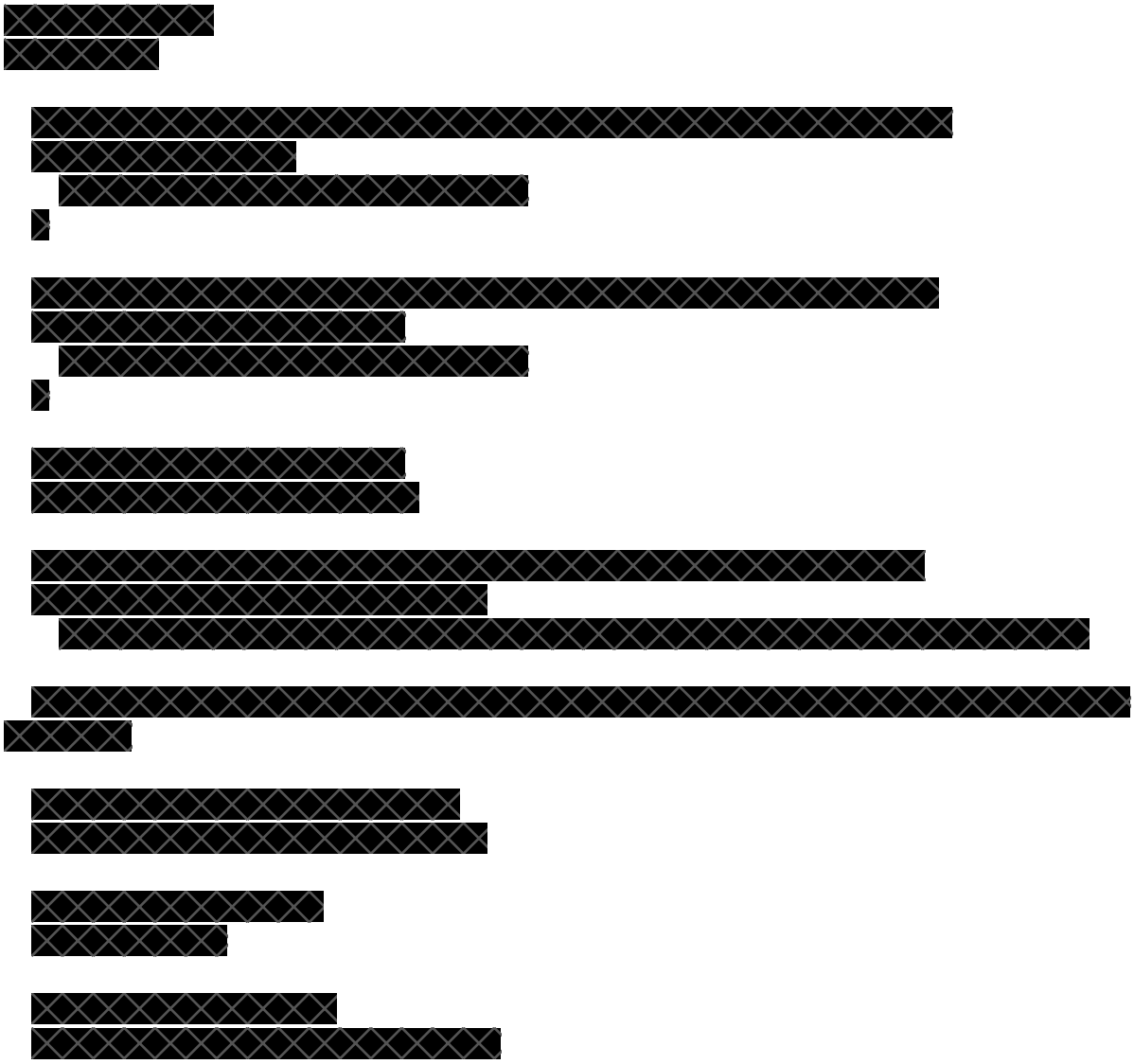
[redacted]



[redacted]

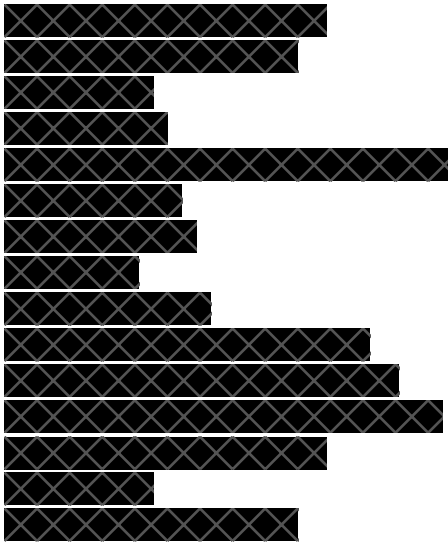


9.14. Anexo 14. Función *main*()



[redacted]

9.15. Anexo 15. Librerías y bloque principal



[redacted]

[redacted]

[redacted]

[redacted]