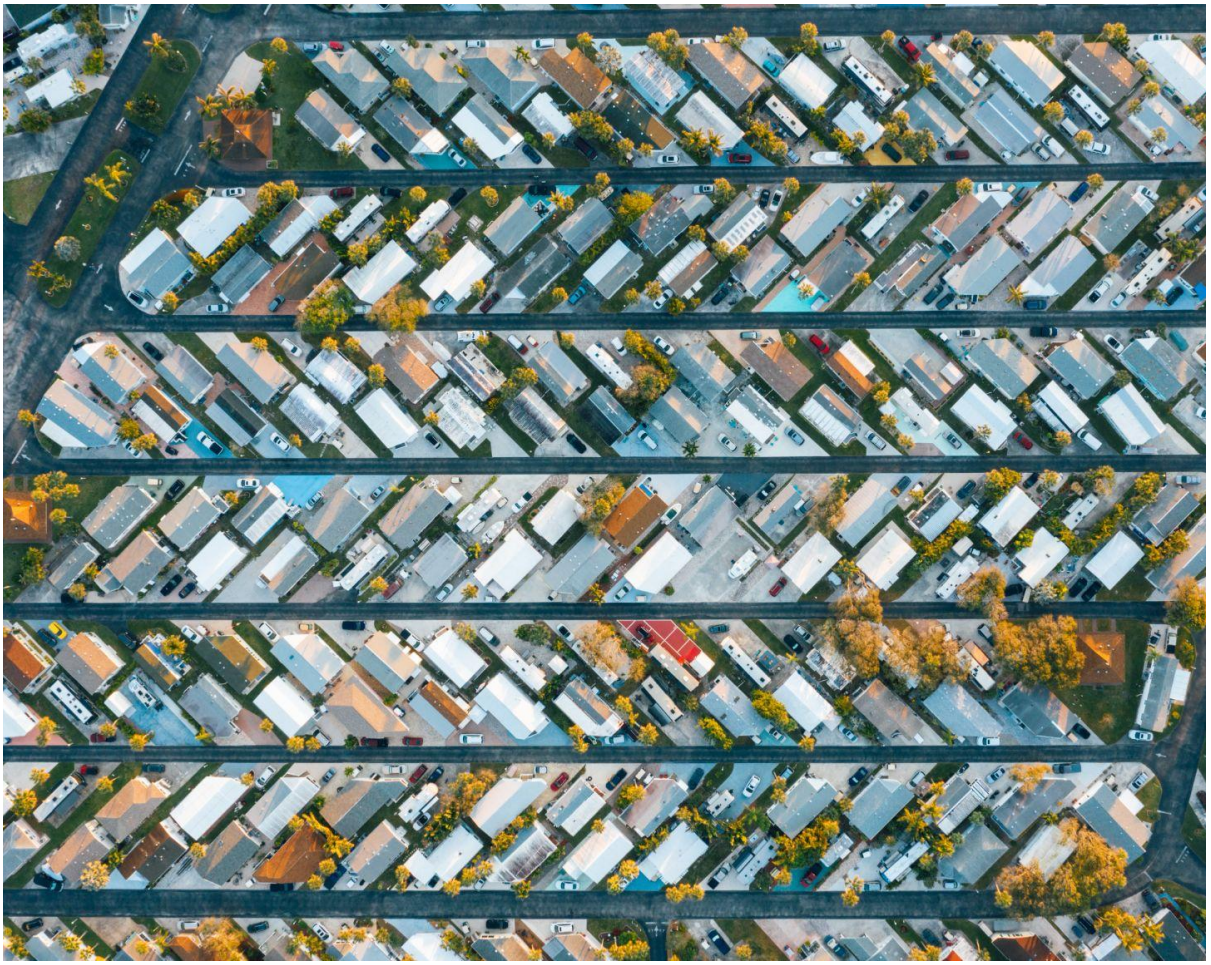
TFM

HOUSE PRICE PREDICTOR



Carmen Muñoz López

Master Data Science

KSCHOOL

ÍNDICE

INTRODUCCIÓN

Este proyecto surge con intereses personales para dar respuesta a dos preguntas:
¿Cuáles son las viviendas más rentables de Churriana?
¿Qué precio estimado debería tener mi vivienda si quiero ponerla a la venta?

A partir de datos recabados de viviendas a la venta de Churriana Málaga del portal de Idealista, se realiza un modelo de machine learning para predecir el precio de las viviendas a partir del cual se obtiene un índice de rentabilidad por vivienda.

Toda la información se muestra de forma interactiva en una aplicación web dónde el usuario puede consultar y filtrar el contenido en base a sus preferencias: precio máximo, número de habitaciones, índice de rentabilidad, viviendas con jardín, con piscina, garaje, nueva construcción, etc.

En resumen, este proyecto busca proporcionar una herramienta valiosa y fácil de usar para aquellos interesados en invertir en el sector inmobiliario.

Este proyecto es relevante para mí, ya que es una herramienta que me gustaría usar en un futuro para mis decisiones a la hora de invertir. Por otro lado, ha sido todo un reto, ya que es el primer proyecto de data science que llevo a cabo, del cual he aprendido mucho y me ha motivado a interesarme más por esta rama de estudio.

DATOS

Para llevar a cabo este proyecto, fue necesario recopilar información y características de viviendas en venta. Tras explorar diferentes opciones se decide usar como fuente de datos el portal de Idealista (compañía española fundada el 4 de octubre de 2000 que ofrece a través de Internet entre otros los servicios de portal inmobiliario en España, Italia y Portugal). A día de hoy, es uno de los portales web más usados para la compra venta de viviendas.

Este portal ofrece una API para poder extraer información de los inmuebles, por lo que este fue otro de los motivos que llevó a seleccionarla como fuente de datos. Tras explorar y probar la API encontramos diferentes barreras para el proyecto, en primer lugar sólo permite hacer 100 llamadas por usuario al mes de las cuales en cada una como máximo devuelve información de 50 viviendas. Además, la API presentaba restricciones para extraer ciertas características de las viviendas. Esto limitaba en gran parte el proyecto, por lo que tras valorar otras opciones y webs objetivo, se decide aplicar técnicas de scraping sobre Idealista ya que es la que proporciona mayor volumen de datos.

En la web se puede consultar información de las viviendas disponibles. Una vez que se hace la búsqueda de la zona deseada se muestra información en diferentes páginas de todas las viviendas, y al hacer clic sobre ellas se puede observar toda la información y características de interés que a partir del scraping se pueden extraer para generar un dataset bruto en formato .csv.

Para este proyecto en primer lugar se decide recabar datos de viviendas a la venta en Churriana, Málaga. Esta zona es elegida por intereses propios ya que es una zona en la que pienso invertir en un futuro cercano y dónde vivo actualmente. Estos datos son extraídos el día 05/11/2022 y sobre ellos trabajaremos para finalmente poder realizar y aplicar técnicas de Machine Learning.

EXTRACCIÓN DE DATOS - SCRAPPING

La extracción de datos a partir de web scraping es realizada usando las librerías Request y BeautifulSoup. En primer lugar con la librería request hacemos la solicitud al portal y posteriormente una vez que tenemos el acceso usamos BeautifulSoup para extraer datos de la página web y extraer información específica de las características de cada inmueble.

La zona objetivo elegida es Churriana y los datos extraídos como se ha mencionado corresponden al día 05/11/2022.

Los datos que se deciden extraer de cada inmuebles son los siguientes:

- Título
- Localización
- Latitud
- Longitud

- Price (variable a predecir)
- Certificado energético
- Número de habitaciones
- Número de baños
- Si dispone o no de jardín
- Si dispone o no de terraza
- Si dispone o no de parking
- Si dispone o no de piscina
- Si dispone o no de ascensor
- Metros cuadrados construidos
- Si es de nueva construcción
- Si necesita ser renovado
- Si está en buenas condiciones

Tras definir la zona de estudio y las características a obtener procedemos con la importación de librerías que vamos a necesitar:

- requests
- BeautifulSoup as bs
- time
- random
- pandas
- json

Se define una serie de encabezados para simular una consulta real y evitar así que el portal nos bloquee. Entre ellas el tipo de navegador, sistema operativo, idioma preferido, etc.

[2]:
```
headers = {
    "accept": 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9',
    "accept-encoding": 'gzip, deflate, br',
    "accept-language": 'es-ES,es;q=0.9,en;q=0.8',
    "cache-control": 'max-age=0',
    "dnt": "1",
    "sec-ch-ua": '"Chromium";v="94", "Google Chrome";v="94", ";Not A Brand";v="99"',
    "sec-ch-ua-mobile": '?0',
    "sec-ch-ua-platform": '"Windows"',
    "sec-fetch-dest": 'document',
    "sec-fetch-mode": 'navigate',
    "sec-fetch-site": 'same-origin',
    "sec-fetch-user": '?1',
    "upgrade-insecure-requests": '1',
    "user-agent": 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.81 Safari/537.36'
}
```

Además, en la función definida para realizar cada consulta se usa time.sleep para establecer un tiempo de pausa de entre 1 y 3 segundos para cada una con el mismo fin que el anterior, humanizar la consulta y evitar que la bloqueen.

A pesar de establecer estas pautas, aunque conseguimos extraer una lista con todos los ids de las viviendas a la venta en Churriana (un total de 342), en el proceso de consulta para extraer las características de cada inmueble solo conseguimos información de 186 viviendas, una representatividad del 54,39%.
Código usado para la extracción de los ids:

[13]:
```
busqueda = "churriana malaga"
busqueda = busqueda.replace(" ","_")

x = 1
ids = []

while True:
    url = f"https://www.idealista.com/buscar/venta-viviendas/{busqueda}/pagina-{x}.htm"

    r = requests.get(url, headers = headers)
    soup = bs(r.text, "html.parser")

    pagina_actual = int(soup.find("main", {"class":"listing-items"}).find("div", {"class":"pagination"}).find("li", {"class":"selected"}).text)

    if x == pagina_actual:
        articles = soup.find("main", {"class":"listing-items"}).find_all("article")

    else:
        break

    x = x+1

    for article in articles:
        id_inmueble = article.get("data-adid")

        ids.append(id_inmueble)

    time.sleep(random.randint(1,3)*random.random())
```

Código usado para definir las características a extraer de cada inmueble. Para hacer este código se procede a inspeccionar el documento html de Idealista para obtener los parámetros y etiquetas de cada una de las características a recopilar.

[7]:
```
soup = bs(r.text, "html.parser")
```

[8]:
```
def info_inmueble():

    scripts = soup.find_all("script")

    for script in scripts:
        if "var config" in str(script):
            break
    latitude = float(str(script).split("latitude")[1].split(",")[0].split()[1].replace("'", ""))
    longitude = float(str(script).split("latitude")[1].split(",")[1].strip().split()[1].replace("'", ""))

    scripts = soup.find_all("script")

    for script in scripts:
        if "var utag_data" in str(script):
            break

    data = json.loads(str(script).split("var utag_data =")[1].split(";")[0].strip())

    titulo = soup.find("h1").text.strip()
    localizacion = soup.find("span",{"class":"main-info__title-block"}).find("span").text.strip()
    price = data["ad"]["price"]
    energy_certification = data["ad"]["energyCertification"]["type"]
    basic_characteristics = data["ad"]["characteristics"]
    room_number = data["ad"]["characteristics"]["roomNumber"]
    bath_number = data["ad"]["characteristics"]["bathNumber"]
```

```
    try:
        has_garden = data["ad"]["characteristics"]["hasGarden"]
    except:
        has_garden = None

    try:
        has_terrace = data["ad"]["characteristics"]["hasTerrace"]
    except:
        has_terrace = None

    try:
        has_parking = data["ad"]["characteristics"]["hasParking"]
    except:
        has_parking = None

    try:
        has_swimmingpool = data["ad"]["characteristics"]["hasSwimmingPool"]
    except:
        has_swimmingpool = None

    try:
        has_lift = data["ad"]["characteristics"]["hasLift"]
    except:
        has_lift = None

    constructed_area = data["ad"]["characteristics"]["constructedArea"]
    is_new_development = data["ad"]["condition"]["isNewDevelopment"]
    is_needs_renovating = data["ad"]["condition"]["isNeedsRenovating"]
    is_goog_condition = data["ad"]["condition"]["isGoodCondition"]

    lista = [titulo, localizacion, latitude, longitude, price, energy_certification, basic_characteristics,
            room_number, bath_number,has_garden, has_terrace, has_parking, has_swimmingpool, has_lift, constructed_area,
            is_new_development, is_needs_renovating, is_goog_condition]

    time.sleep(random.randint(1,3)*random.random())
    return lista
```

Código para la extracción de las características de cada inmueble:

[30]:
```
lista_info_inmuebles = []

for id_inmueble in ids:
    r = request_piso(id_inmueble= id_inmueble, headers=headers)
    soup = bs(r.text, "html.parser")
    info_inmueble1 = info_inmueble()
    lista_info_inmuebles.append(info_inmueble1)
    time.sleep(random.randint(1,3)*random.random())

print(id_inmueble)
```

Finalmente usamos pandas para transformar los datos en un DataFrame y guardarlos en formato csv.

[24]:
```
columns = ["titulo", "localizacion", "latitude", "longitude", "price", "energy_certification", "basic_characteristics",
           "room_number", "bath_number", "has_garden", "has_terrace", "has_parking", "has_swimmingpool", "has_lift", "constructed_area",
           "is_new_development", "is_needs_renovating", "is_goog_condition"]
```

[25]:
```
df_inmuebles = pd.DataFrame(lista_info_inmuebles, columns= columns)
```

[26]:
```
df_inmuebles.head()
```

[26]:

	titulo	localizacion	latitude	longitude	price	energy_certification	basic_characteristics	room_number	bath_number	has_garden	has_terrace	has_parking	has_swimmingpool	has_lift	co
0	Chalet adosado en venta en Guadalmar	Churriana, Málaga	36.666638	-4.464576	395000	inProcess	{'roomNumber': '4', 'bathNumber': '3', 'hasPar...	4	3	1	1	1	1	None	
1	Casa o chalet independiente en venta en calle ...	Churriana-El Pizarrillo-La Noria-Guadalsol, Má...	36.663719	-4.503897	640000	inProcess	{'roomNumber': '4', 'bathNumber': '2', 'hasPar...	4	2	1	1	1	1	None	
2	Casa o chalet independiente en venta en calle ...	Cortijo de Maza-Finca Monsalvez-El Olivar, Málaga	36.656894	-4.502930	567000	inProcess	{'roomNumber': '5', 'bathNumber': '4', 'hasPar...	5	4	1	1	0	1	None	
3	Ático en venta en calle Bangladesh, 25	Churriana-El Pizarrillo-La Noria-Guadalsol, Má...	36.671190	-4.516900	375900	unknown	{'roomNumber': '4', 'bathNumber': '2', 'hasLif...	4	2	1	1	1	1	1	
4	Piso en venta en calle Bangladesh, 25	Churriana-El Pizarrillo-La Noria-Guadalsol, Má...	36.671190	-4.516900	249900	unknown	{'roomNumber': '2', 'bathNumber': '2', 'hasLif...	2	2	0	1	1	1	1	

Este volumen de datos es muy pequeño para poder realizar modelos de predicción exitosos y que sea un modelo fiable para el uso real. Aún así, conociendo estas limitaciones para el proyecto se decide continuar ya que el objetivo del mismo es tener conocimiento de esta zona específica "Churriana" por intereses propios de invertir en la zona.

Esto es muy importante a la hora de decidir qué modelo aplicar para que no sean tan sensibles a la falta de datos. Para ello aplicaremos diferentes modelos junto a sus respectivas evaluaciones de precisión y confiabilidad y seleccionaremos aquel que mejores resultados reporte.

En las conclusiones del proyecto se plantean puntos de mejora para adquirir más datos y mejorar el modelo en el futuro.

EXPLORACIÓN Y ANÁLISIS DE LOS DATOS

Finalmente tras la realización del web scraping obtenemos un dataset que consta de 18 columnas y 186 datos de viviendas de Churriana, Málaga.

Analizamos las características del mismo:
dtypes: float64(7), int64(7), object(4)

#	Columna	Non-Null Count	Dtype
0	titulo	187 non-null	object
1	localizacion	187 non-null	object
2	latitude	187 non-null	float64
3	longitude	187 non-null	float64
4	price	187 non-null	int64
5	energy_certification	187 non-null	object
6	basic_characteristics	187 non-null	object
7	room_number	187 non-null	int64
8	bath_number	187 non-null	int64
9	has_garden	164 non-null	float64
10	has_terrace	170 non-null	float64
11	has_parking	169 non-null	float64
12	has_swimmingpool	158 non-null	float64
13	has_lift	81 non-null	float64
14	constructed_area	187 non-null	int64
15	is_new_development	187 non-null	int64
16	is_needs_renovating	187 non-null	int64
17	is_goog_condition	187 non-null	int64

Como se puede observar, tres de las columnas del dataset son variables categóricas, las cuales no se pueden usar para alimentar un modelo de predicción por lo que trabajaremos posteriormente aplicando técnicas de Data Engineer para transformar las mismas en variables numéricas y así poder nutrir el análisis predictivo con las mismas.

Datos estadísticos:

	count	mean	std	min	25%	50%	75%	max
latitude	187.0	36.692094	0.113276	36.610730	36.659499	36.663815	36.671115	3.714505e+01
longitude	187.0	-4.449519	0.202287	-4.686328	-4.506868	-4.501821	-4.490153	-3.641632e+00
price	187.0	482993.689840	512781.940307	54900.000000	249900.000000	299990.000000	567000.000000	5.000000e+06
room_number	187.0	3.625668	1.629362	1.000000	3.000000	3.000000	4.000000	1.400000e+01
bath_number	187.0	2.556150	1.395385	1.000000	2.000000	2.000000	3.000000	8.000000e+00
has_garden	164.0	0.695122	0.461766	0.000000	0.000000	1.000000	1.000000	1.000000e+00
has_terrace	170.0	0.947059	0.224578	0.000000	1.000000	1.000000	1.000000	1.000000e+00
has_parking	169.0	0.899408	0.301681	0.000000	1.000000	1.000000	1.000000	1.000000e+00
has_swimmingpool	158.0	0.867089	0.340558	0.000000	1.000000	1.000000	1.000000	1.000000e+00
has_lift	81.0	0.975309	0.156150	0.000000	1.000000	1.000000	1.000000	1.000000e+00
constructed_area	187.0	243.433155	270.325987	50.000000	108.500000	152.000000	256.000000	2.500000e+03
is_new_development	187.0	0.229947	0.421928	0.000000	0.000000	0.000000	0.000000	1.000000e+00
is_needs_renovating	187.0	0.053476	0.225585	0.000000	0.000000	0.000000	0.000000	1.000000e+00
is_goog_condition	187.0	0.716578	0.451870	0.000000	0.000000	1.000000	1.000000	1.000000e+00

Analizando estos datos nos damos cuenta que hay grandes diferencias entre los valores mínimos y máximos de algunas de las características del dataset. Esto puede sesgar el modelo por lo que en los siguientes apartados realizaremos un análisis más profundo que nos permita conocer mejor los datos y cómo tratarlos para conseguir un modelo con la mayor efectividad posible.

LIMPIEZA DE DATOS

En este apartado analizaremos en profundidad nuestro conjunto de datos con el fin de limpiar el mismo, detectar valores atípicos y ver las correlaciones existentes.

En primer lugar se elimina la columna “basic characteristics” ya que la misma contiene un diccionario con todas las características de la viviendas las cuales ya tenemos desglosadas en diferentes columnas.
Para eliminar dicha columna usamos la función de la librería pandas “.drop”, y volvemos a guardar en la variable data nuestro dataset sin dicha columna.

En segundo lugar, analizamos todos los valores nulos del conjunto.

Columna	NaN
titulo	0
localizacion	0
latitude	0
longitude	0
price	0
energy_certification	0
room_number	0

bath_number	0
has_garden	**23**
has_terrace	**17**
has_parking	**18**
has_swimmingpool	**29**
has_lift	**106**
constructed_area	0
is_new_development	0
is_needs_renovating	0
is_goog_condition	0
Total	**193**

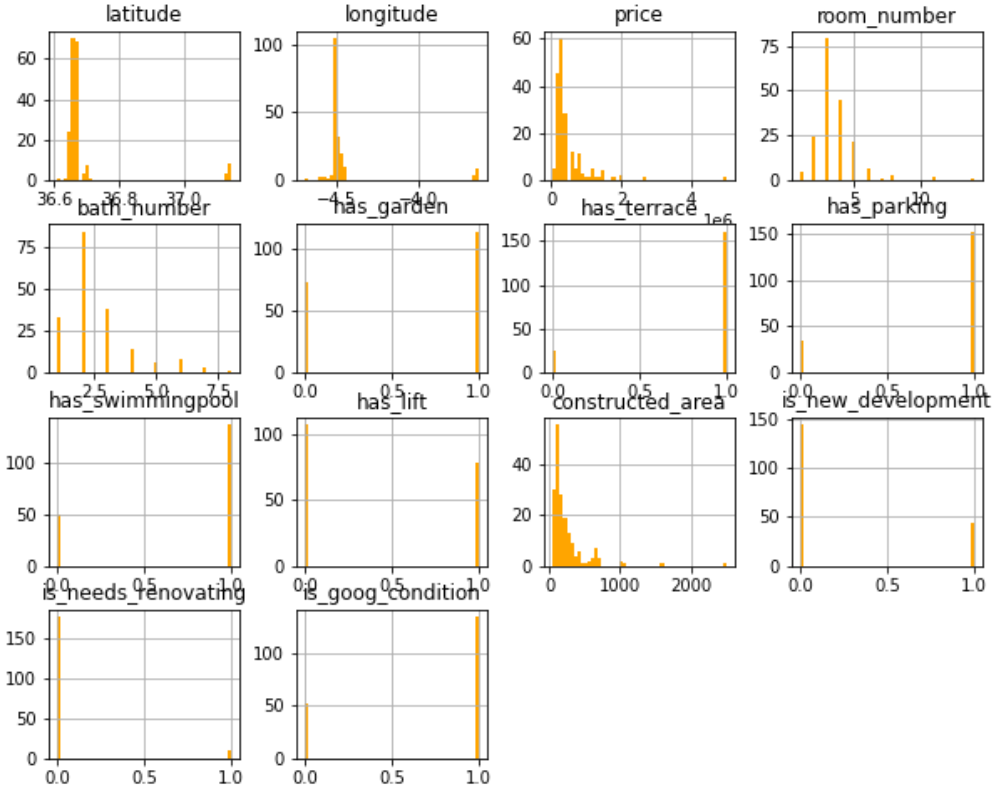
Se registran un total de **193 valores nulos** procedentes de las columnas: has_garden, has_terrace, has_parking has_swimmingpool y has_lift. Estas columnas presentan valores binarios: 0 si la vivienda no tiene esa característica y 1 si la tiene. Por lo que se procede a convertir todos los valores NaN en valor 0. Para ellos se aplica la función de pandas .fillna(0).

Una vez realizada esta limpieza guardamos el dataset en formato csv denominándose: clean_properties_churriana.csv

Una vez tenemos el dataset con las columnas que necesitamos y limpio de valores nulos procedemos a explorar los datos:

Partiremos con la visualización de los histogramas de las diferentes características de las viviendas. En la gráfica se observa que 'price' y 'constructed_area' presentan skew positivo, es decir, hay algunas viviendas que poseen mayores precios y metros cuadrados que el resto y esto hace que la gráfica presente una cola hacia la derecha.

Esta información será útil para determinar qué variables serán transformadas para que su distribución se ajuste a la de la normal y, tal vez, ayude a mejorar los resultados de la predicción. Analizaremos posibles outliers en las distintas variables.



Hacemos un **gráfico heatmap** para estudiar las correlaciones de la variable a predecir "price" con el resto del conjunto. Ver cuales presentan las correlaciones más altas y las más bajas y si son positivas o negativas. Esto nos ayuda a identificar qué variables son más relevantes para el modelo predictivo.

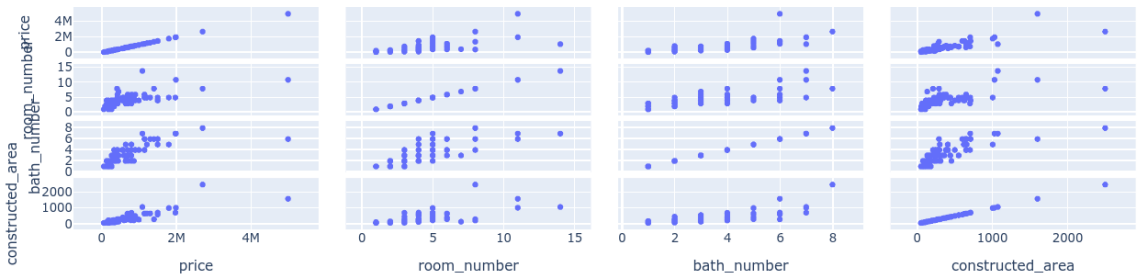
- Las variables 'constructed_area', 'room_number', 'bath_number' son las que mayor relación positiva tienen con respecto a la variable precio. 'has_lift' y 'is_need_renovating' presentan una relación negativa.
- Para el modelo valoraremos la exclusión de la variable 'is_need_renovating' ya que ya disponemos de la información 'is_good_condition' que ya nos está indicando si la vivienda está en buenas condiciones.

Analizamos nuevamente como se comporta la variable precio con respecto a las variables con mayor correlación a través de un gráfico de dispersión.

- Presenta una correlación lineal. A mayor cantidad de metros construidos, habitaciones y baños mayor precio.
- Podemos observar algunos valores extremos en precio y en metros construidos.

```
[19]: fig = px.scatter_matrix(data,
                             dimensions= ['price', 'room_number', 'bath_number', 'constructed_area']

                             )
      fig.show()
```

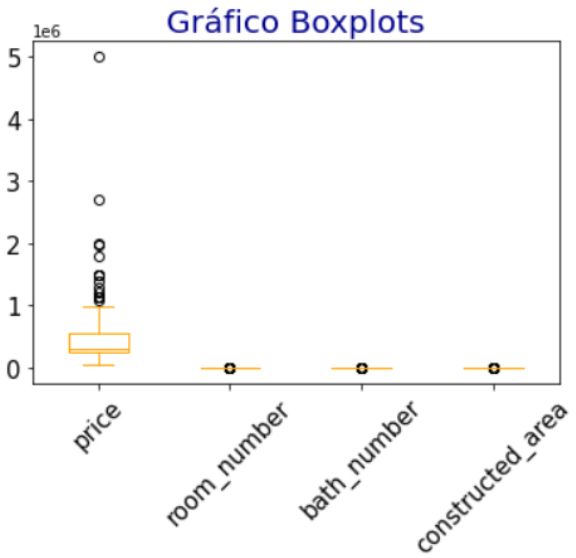


DATA ENGINEERING

Exploración de posibles outliers

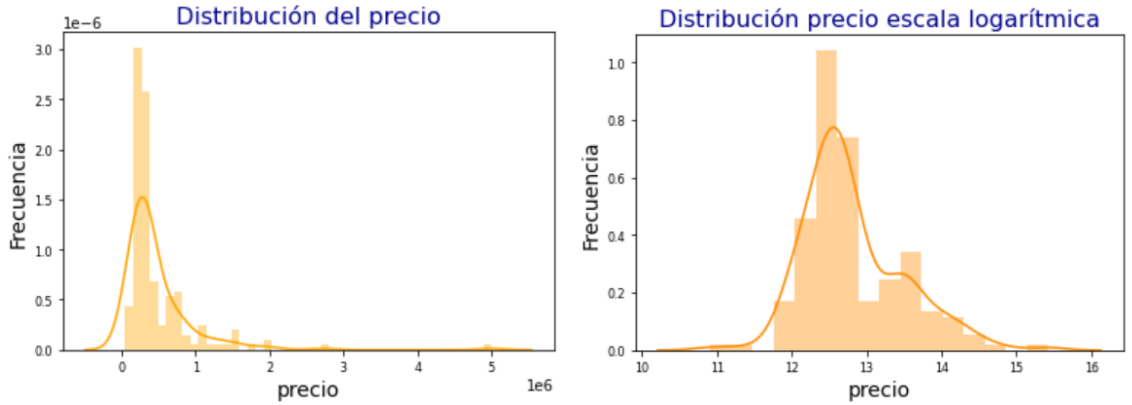
Realizamos diferentes análisis para identificar estos valores "Outliers" y estudiar si se debe a valores atípicos que pueden afectar a nuestro modelo o si se debe a valores extremos y por tanto valorar si son o no excluidos del conjunto ya que pueden aportar información relevante para el aprendizaje del modelo.

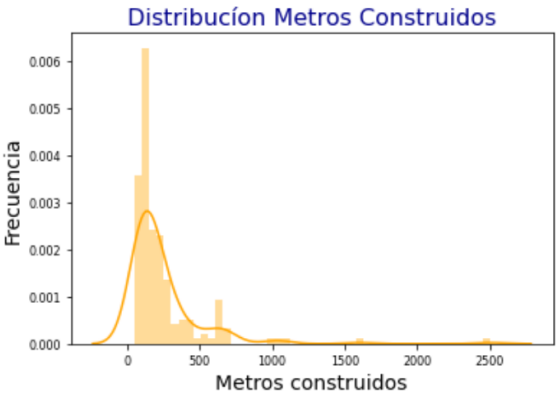
Seleccionamos del conjunto de datos las columnas de interés para analizar estos posibles outliers:

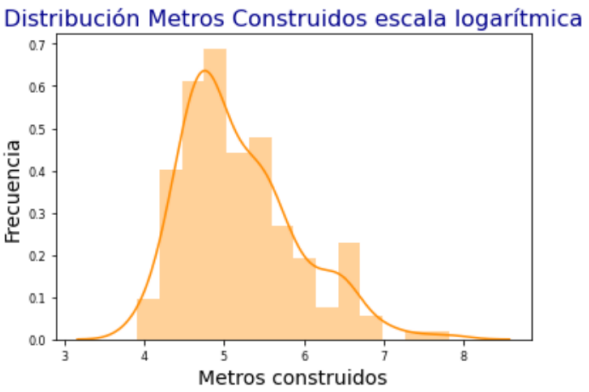


Podemos ver como la variable precio presenta outliers. Analizaremos estos valores identificados ya que es importante determinar si son valores atípicos o valores extremos. Los valores atípicos se deben a errores de medición o captura de datos, mientras que los valores extremos pueden indicar una situación especial o un comportamiento anómalo.

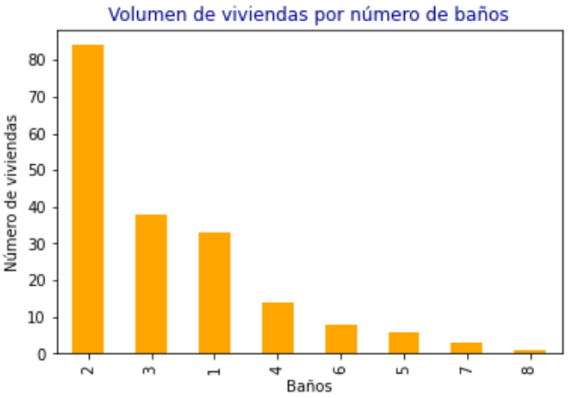
Analizamos la distribución de la variable precio la cual representa una curva sesgada a la derecha. Además, analizamos la misma en escala logarítmica para ver mejor la distribución de los datos.





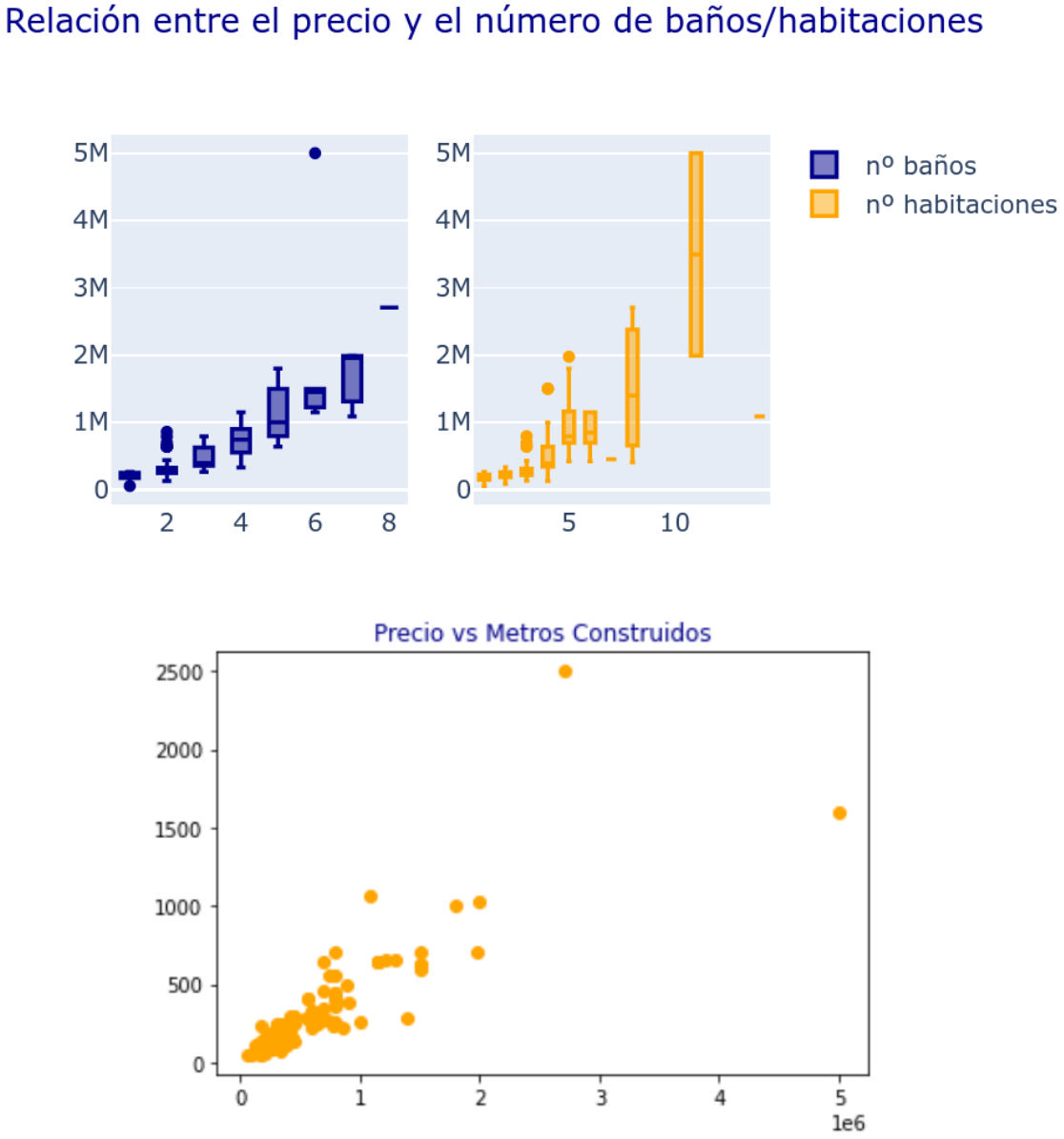


Analizamos el volumen de viviendas según el número de habitaciones.
La mayoría de las viviendas presenta 3 y 4 habitaciones y entre 2 y 3 baños. Con esto podemos concluir que las viviendas en Churriana son amplias y con gran volumen de habitaciones.



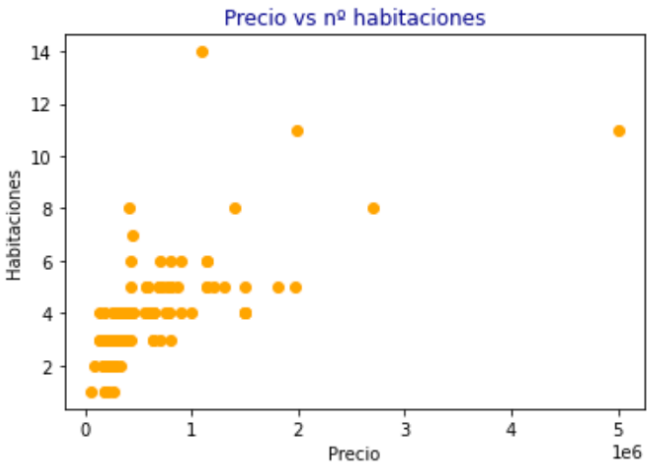
Podemos observar que hay viviendas con 14 y 11 habitaciones. Un comportamiento atípico que podría sesgar los datos y afectar al modelo ya que estas viviendas no son comparables con el resto de inmuebles del dataset que son viviendas estándar de la zona, estas son más parecidas a lo que podría ser un cortijo o una vivienda de lujo (mansión). Valoraremos si para el modelo es beneficioso o no incluir estos inmuebles ya que el volumen de ellos es muy bajo para que el modelo pueda realizar un aprendizaje exitoso y consiga ajustar bien la predicción a nuevos datos.

Analizamos la relación del precio con el número de habitaciones y el número de baños.
Se observan algunos Outlier en función del precio en viviendas de dos baños y en viviendas con 6 baños, el precio de esta es de 5 millones.
En Viviendas de 3, 4 y 5 habitaciones también se observan Outliers.



Análisis del precio en función del número de habitaciones

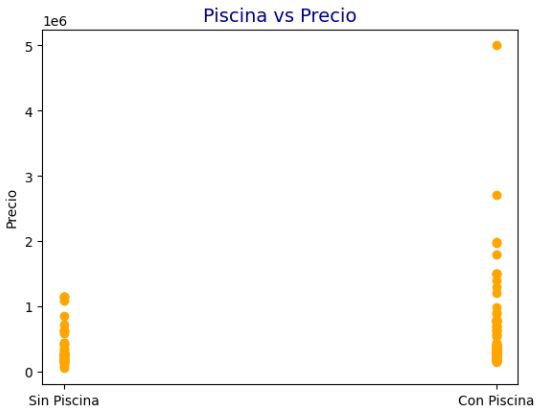
Podemos afirmar que los valores extremos de precio identificados en el estudio corresponden con viviendas que presentan un gran número de habitaciones.
Esto es algo normal y podemos decir que los datos no son atípicos sino que son valores extremos.
A pesar de afirmar esto, identificamos estos puntos para guardar un nuevo dataset con estos valores excluidos.
Posteriormente entrenaremos los modelos tanto con todo el conjunto de datos como con el dataset sin outliers para quedarnos con el que mejor rendimiento aporte al modelo de predicción.

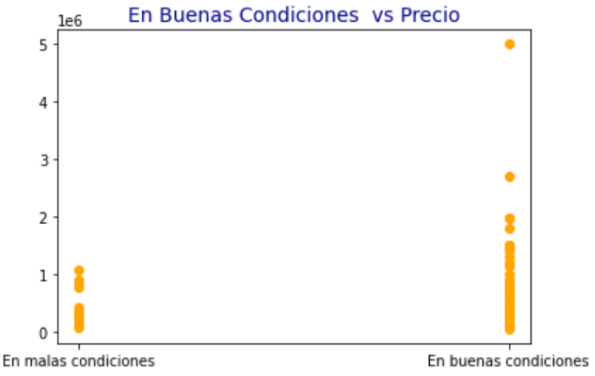


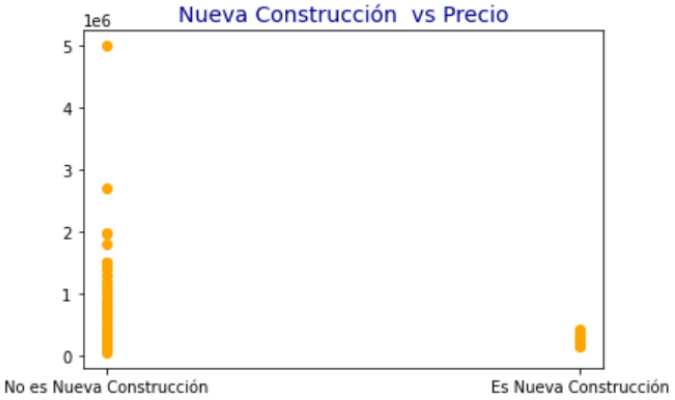
Realizamos algunos análisis adicionales en función del resto de características de la base de datos:

- Relación del precio con piscina.
- Relación del precio con el jardín.
- Relación del precio con si es nueva construcción.
- Relación del precio con si está en buenas condiciones.
- Relación del precio con si necesita reforma.
- Relación del precio con sí tiene terraza.

Las viviendas que presentan estas características presentan precios más elevados a excepción de si son de nueva construcción que no siempre tienen precios superiores al resto.







En Churriana la gran mayoría de viviendas a la venta tienen piscina, un 73,65% y una gran parte de ellas jardín, el 61,29%. Además casi todas están en buenas condiciones, tan solo el 5,38% necesita reforma.

Un 23,12% de las viviendas a la venta son de nueva construcción por lo que podemos concluir que es una zona en auge de construcción.

Una vez realizado el análisis afirmamos que existen valores extremos ya que hay inmuebles con precios muy superiores a la mayoría, pero no podemos decir que sean valores atípicos ya que esos precios vienen definidos por las características de la vivienda, ya que son mucho más grandes y con mayor volumen de habitaciones.

Se calcula el valor intercuartílico IQR para poder delimitar el extremo inferior y superior en función de la variable precio para ver qué viviendas se encuentran con precios superiores o inferiores a los límites y proceder al tratamiento de datos excluyendo estos valores.

El límite inferior nos da un valor negativo y el límite superior nos da un valor de 1.042.650€ por lo que todas las viviendas con precios superiores al límite superior sería valores extremos. En total se encuentran 17 viviendas con estas características.

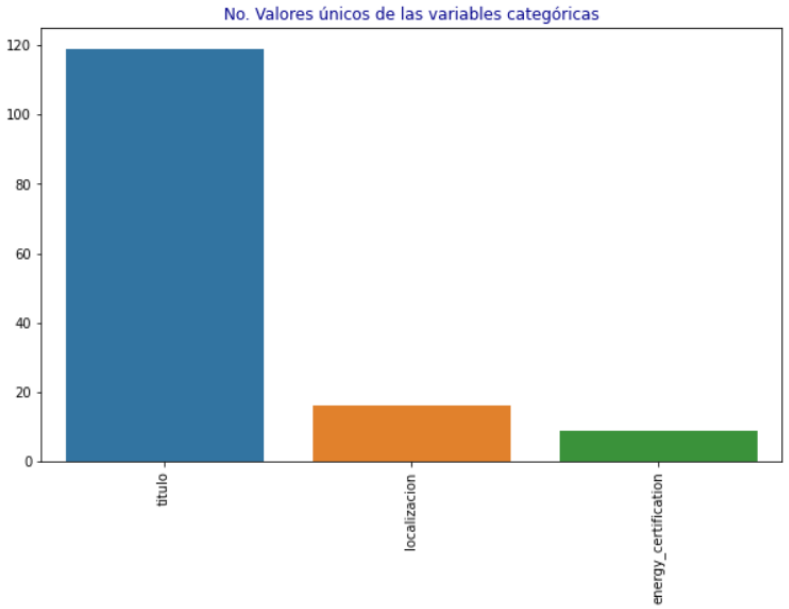
Procedemos a excluirlas del dataset, aunque estas viviendas podrían aportar información valiosa para el modelo ya que están representando una tipología de vivienda más lujosa. Debido a que representan un volumen muy bajo del conjunto lo cual no es lo suficientemente representativo para que el modelo pueda realizar un aprendizaje exitoso, se procede a excluirlas creando un nuevo dataset: clean_properties_churriana_without_outliers.csv para ver el rendimiento de los modelos usando ambos dataset, el completo y el que excluye estos valores extremos.

Hay que tener en cuenta que usando el dataset completo puede que nuestro modelo para nuevos datos de entrada sea más preciso para viviendas estándar que para viviendas de lujo.

Preprocesamiento de variables Categóricas a numéricas.

Para hacer modelos de predicción no podemos usar variables categóricas por lo que vamos a realizar un preprocesamiento de datos con técnicas de Data Engineering en el que convertiremos las variables categóricas a numéricas para poder trabajar y nutrir al modelo con ellas.

Las variables categóricas obtenidas son: Título, Localización y Certificación energética.



La variable título no va a ser usada en el modelo. La misma presenta valores diferentes en todas las viviendas ya que cada propietario pone el título que desea para promocionar su viviendas en el portal.
La variable Localización no va a ser usada tampoco en el modelo. La zona de estudio es Churriana por lo que no aporta valores significativos, además disponemos de las columnas latitud y longitud que ya marcan un punto en el mapa con lo que ya se estaría teniendo en cuenta la localización en el modelo.

Para la variable certificación energética se realiza un preprocesamiento de datos para transformar los valores categóricos en numéricos. Para ello usaremos Label Encoder.

Una vez realizados todos estos pasos ya tendríamos los datos listos para aplicar los modelos. Por un lado tenemos los datos completos limpios y preprocesados y por otro lado estos mismos datos pero sin los valores extremos. Observaremos cual presenta mejor rendimiento.

MACHINE LEARNING

Evaluación de Modelos

En este apartado del proyecto, se pretende estudiar y aplicar diferentes modelos de machine learning para predecir el precio de las viviendas con el fin de encontrar el modelo más óptimo. En este caso al tener un volumen muy bajo de datos no esperamos tener un modelo realista, sino más bien, obtener unos resultados aceptables con los que poder seguir trabajando y mejorando.

Se aplican los siguiente modelos ya que suelen ser los más usados en predicciones similares a la realizada en este proyecto. Se analizan las métricas de evaluación de cada uno para ver cuál obtiene mejor rendimiento con diferentes parámetro. Para ello se elabora una tabla que irá recogiendo los datos de evaluación de cada uno, así de un simple vistazo se puede identificar los mejores.

Estos modelos los aplicaremos tanto al dataset completo como al dataset sin outliers. Finalmente desarrollamos, aplicamos y visualizamos el modelo que mejor rendimiento tenga.

Modelos evaluados:

- **Regresión Lineal:** Es un modelo simple y fácil de entender que asume una relación lineal entre las variables predictoras y la variable objetivo. Es rápido y fácil de implementar, pero puede ser inadecuado para conjuntos de datos con relaciones complejas.

- **SVM Lineal:** Es un modelo de aprendizaje automático supervisado que se utiliza a menudo para clasificación. Sin embargo, también se puede utilizar para regresión, especialmente con conjuntos de datos pequeños. Es rápido y eficiente en términos de recursos, pero puede ser inadecuado para manejar relaciones no lineales en los datos.

- **Regresión de Puente:** Es una técnica de regresión que se utiliza para estimar modelos en datos con valores atípicos o outliers. Se utiliza a menudo en problemas de regresión en los que se espera que los valores atípicos tengan un efecto importante en los resultados.

- **XGBoost Regression:** Es un algoritmo de gradient boosting para la regresión que se utiliza a menudo para problemas complejos. Es uno de los modelos más utilizados en competiciones de ciencia de datos y es conocido por su precisión y capacidad para

manejar relaciones complejas entre variables. Sin embargo, puede requerir más tiempo de entrenamiento y ajuste de hiperparámetros que otros modelos.

- **Gradient Boosting Regressor:** Es un modelo de aprendizaje automático enfocado en la regresión que se basa en la combinación de muchos modelos más simples para obtener una solución más precisa. Es conocido por su capacidad para manejar relaciones complejas entre variables y puede ser muy preciso, pero puede requerir una gran cantidad de tiempo de entrenamiento.

- **Árbol de Regresión:** Es un modelo de aprendizaje automático que se basa en la construcción de un árbol de decisión para estimar la relación entre las variables predictoras y la variable objetivo. Es fácil de entender y visualizar, pero puede ser propenso a sobre ajustarse a los datos de entrenamiento.

Definimos las diferentes funciones de los modelos en un notebook, que posteriormente importamos para su uso con los diferentes datos, así evitamos estar todo el tiempo usando el mismo código repetidamente.

División de datos en train y test

Esta división es totalmente aleatoria para eliminar cualquier parcialidad en el ordenamiento del conjunto de datos. Además, para poder entrenar todos los modelos con la misma división de datos entre train y test y que la comparación entre ellos sea lo más igualada posible y no se vea sesgada por los datos de entrenamiento, generamos un notebook en el que usamos **%store** para guardar esta separación de datos. La separación usada es 20% traint 80% test.

Definimos una clase "DataSplit" que contendrá los valores que se aplicarán a los distintos modelos. El objetivo es facilitar el manejo de estos valores mediante un contenedor. Esto ha sido necesario para evitar llamar para los diferentes datos de entrada, en este caso datos completos y sin outliers de forma diferente a las variables train y test y evitar que en caso de necesitar usarlo para más datos tener que ponerles nombre diferentes para que no se sobre escriban unas con otras.

En segundo lugar definimos el store dónde aplicamos la clase definida dentro del mismo.

[1]:
```
import pandas as pd
```

Guardamos variables para el dataset completo

[2]:
```
file_path = "../data/data_model_churriana_malaga.csv"
data = pd.read_csv(file_path, index_col = 0)
```

[3]:
```
import sklearn


features = data.drop('price',axis=1)
target = data['price']

from sklearn.model_selection import train_test_split


features_train,features_test,target_train,target_test = train_test_split(features,target,test_size=0.2)

#Guardamos variables en el store para poder llamarlas y usarlas en otros notebooks.
```

[4]:
```
%store target
%store features
%store features_train
%store features_test
%store target_train
%store target_test
```

```
Stored 'target' (Series)
Stored 'features' (DataFrame)
Stored 'features_train' (DataFrame)
Stored 'features_test' (DataFrame)
Stored 'target_train' (Series)
Stored 'target_test' (Series)
```

Esto nos permite poder llamar en los notebooks deseados a dicha variable y de esta forma todos los modelos se entrenan con los mismos datos.

1.- Funciones_modelos_ML: Contiene las funciones que se ejecutarán posteriormente en los notebooks de evaluaciones.
2.- classes: Define las clases que se usarán como contenedor de datos para el entrenamiento y test de los modelos.
3.- store: Recupera los datos de scrapping y los almacena en el store ayudándose de "classes".

Por último, creamos las funciones para los diferentes modelos en el notebook funciones_modelos_ML.

Definición métricas de rendimiento usadas:

Como **métricas de evaluación usamos R2 (coeficiente de determinación)** para medir la calidad de ajuste del modelo así como proporción de la variabilidad total en la variable dependiente que puede ser explicada por el modelo. Es importante evaluar tanto el R2 de entrenamiento como el R2 de prueba para tener una idea más precisa de la calidad del modelo ya que un buen R2 en train y un rendimiento pobre en test puede indicar que existe overfitting o underfitting. El coeficiente de determinación de un modelo es una estadística útil en los análisis de regresión, ya que describe a menudo “como es de bueno” el modelo haciendo predicciones.

- Los valores de R^2 de 0 a 1 indican que el porcentaje de correlación cuadrática entre los valores de predicción y reales de la variable objetivo.
- Un modelo con un valor 0 de R^2 no es mejor que un modelo que predice siempre la media de la variable objetivo.

- Con un valor 1 de R^2 predice perfectamente la variable objetivo.
- Cualquier valor entre 0 y 1 indica qué porcentaje de la variable objetivo puede ser explicada por las características, usando este modelo.
- Un modelo puede también tener una R2 negativa, lo que indica que el modelo es claramente peor que uno que siempre prediga la media de la variable objetivo.

Por otro lado, usamos como métrica de evaluación MAE para tener una idea de cuánto debería estar equivocado en promedio el modelo con respecto a los precios reales. Ambas métricas se aplican tanto en el conjunto train como en el test.

Pasos de ejecución previos a los modelos

Para comenzar a entrenar los modelos es necesario previamente ejecutar tanto nuestros notebooks de store como el de las funciones de los modelos:

- Classes
- Store
- Funciones_modelos_ML

Tras ejecutar estos notebooks procedemos a ejecutar y evaluar los diferentes modelos.

Evaluación de modelos

- Datos completos
    - Sin procesar
    - Datos escalados
- Datos sin Outliers
    - Sin procesar
    - Datos escalados

Datos completos

Evaluamos todos los modelos con los datos completos del dataset sin procesar y escalando los datos para normalizarlos y comprobar de qué forma el modelo logra aprender mejor y generar los mejores resultados.

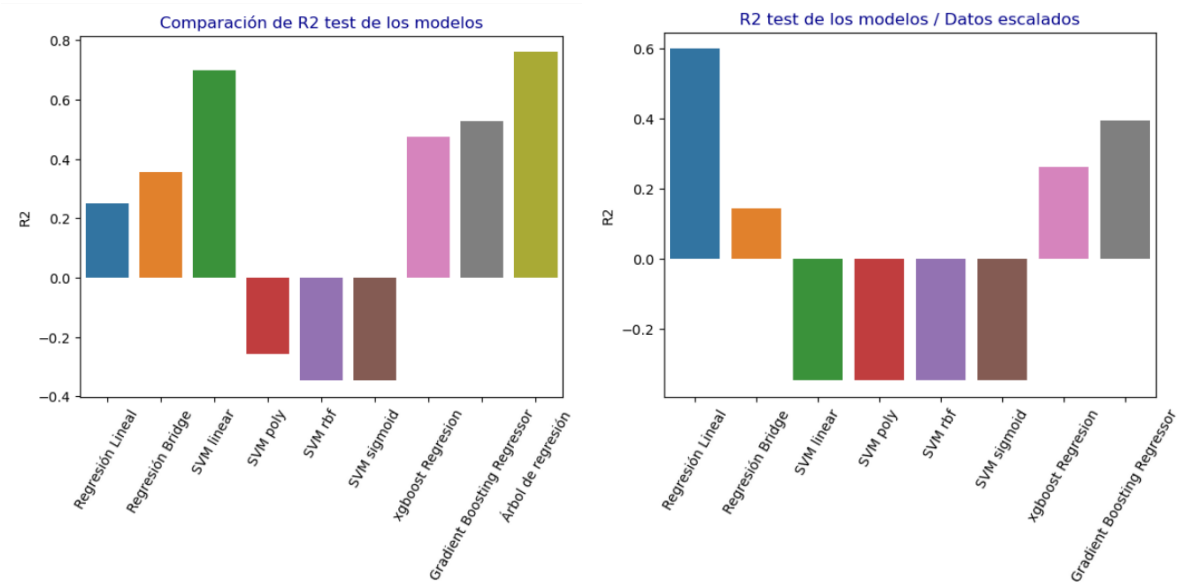
Para escalar los datos se ha usado MinManScaler de la librería de skickit-learn.

Una vez ejecutados todos los modelos esta es la tabla de resultados obtenida ordenada por R2 test:

	Modelo	Detalles	R2 train	R2 test	MAE train	MAE test
16	Árbol de regresión	Sin procesar	0.5774	0.7596	95455.722670	172703.538059
4	SVM linear	Sin procesar	0.7639	0.6973	107439.937805	138592.268673
1	Regresión Lineal	Datos escalados	0.8226	0.5991	111651.180321	241667.780677
17	Árbol de regresión	Datos sscalados	0.5774	0.5314	95455.722670	261597.541502
14	Gradient Boosting Regressor	Sin procesar	0.9958	0.5268	21729.779228	167876.985202
12	xgboost Regresion	Sin procesar	1.0000	0.4733	352.283242	175229.470806
15	Gradient Boosting Regressor	Datos escalados	0.9958	0.3944	21729.779228	239136.647191
2	Regresión Bridge	Sin procesar	0.8065	0.3546	111347.069136	184121.811797
13	xgboost Regresion	Datos escalados	1.0000	0.2626	352.283242	299370.102590
0	Regresión Lineal	Sin procesar	0.8226	0.2512	111651.180321	193646.602994
3	Regresión Bridge	Datos escalados	0.2052	0.1434	207906.719051	330167.148842
7	SVM sigmoid	Sin procesar	-0.0895	-0.3481	215891.619898	386467.649973
10	SVM rbf	Datos escalados	-0.0895	-0.3480	215886.098365	386463.530930
11	SVM sigmoid	Datos escalados	-0.0895	-0.3480	215891.608199	386467.432367
8	SVM linear	Datos escalados	-0.0894	-0.3480	215867.385009	386447.723129
6	SVM rbf	Sin procesar	-0.0895	-0.3480	215883.421059	386456.616433
9	SVM poly	Datos escalados	-0.0893	-0.3478	215824.622849	386399.463979
5	SVM poly	Sin procesar	-0.0693	-0.2575	214909.500188	379279.223971

Como era de esperar los resultados no son muy buenos, se obtiene una precisión baja y un error absoluto medio muy elevado.

Comparamos los resultados gráficamente para ver la comparación de forma más visual.



Los modelos que obtienen en este caso los mejores resultados son SVM linear y xgboost Regresion para datos sin procesar.

SVM Linear obtiene un **R2 de entrenamiento de 0.7639**, lo que significa que el 76,39% de la variabilidad en el precio de la vivienda puede ser explicado por las variables independientes en el modelo. Por otro lado, el **R2 de prueba es de 0.6963**, lo que indica que el modelo es capaz de explicar el 69,63% de la variabilidad en los datos de prueba, que son diferentes a los datos de entrenamiento. Además este modelo puede estar equivocado en promedio con respecto a los precios reales en más de 138 mil euros.

Por otro lado, **el Árbol de Regresion** obtiene un **R2 de entrenamiento del 57,74%** y un **R2 test del 75,96%**. Además este modelo puede estar equivocado en promedio con respecto a los precios reales en más de 172 mil euros

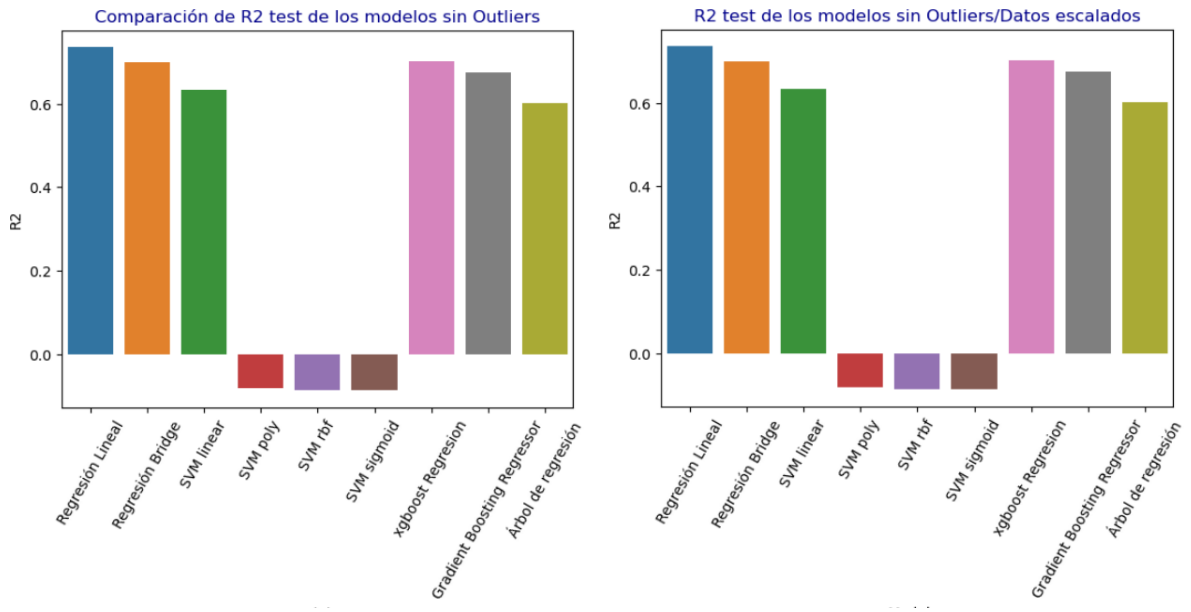
En definitiva, se obtienen unos primeros resultados bastante malos. Los modelos no presentan un buen ajuste y no son capaces de generalizar bien a nuevos datos.

Datos sin Outliers:

Procedemos a analizar los modelos para los datos excluyendo outliers. Se obtienen los siguientes resultados ordenados por R2 test.

	Modelo	Detalles	R2 train	R2 test	MAE train	MAE test
0	Regresión Lineal	Sin procesar/Sin outliers	0.7928	0.7354	61043.839656	75237.052916
12	xgboost Regresion	Sin procesar/Sin outliers	1.0000	0.7013	218.786305	68433.148438
2	Regresión Bridge	Sin procesar/Sin outliers	0.7402	0.7010	69056.248992	75928.876317
14	Gradient Boosting Regressor	Sin procesar/Sin outliers	0.9925	0.6748	12069.541356	66372.415119
4	SVM linear	Sin procesar/Sin outliers	0.6527	0.6331	74947.345882	84178.377877
1	Regresión Lineal	Datos escalados/Sin outliers	0.7928	0.6329	61043.839656	114598.333061
16	Árbol de regresión	Sin procesar/Sin outliers	0.8325	0.6017	52274.111943	86346.884334
15	Gradient Boosting Regressor	Datos escalados/Sin outliers	0.9923	0.5453	12104.364506	116695.455029
17	Árbol de regresión	Datos escalados/Sin outliers	0.8325	0.5291	52274.111943	107570.727323
13	xgboost Regresion	Datos escalados/Sin outliers	1.0000	0.4943	218.786305	116890.324449
3	Regresión Bridge	Datos escalados/Sin outliers	0.3230	0.3019	118907.210966	140214.262904
7	SVM sigmoid	Sin procesar/Sin outliers	-0.1107	-0.0881	132274.251154	143042.550448
10	SVM rbf	Datos escalados/Sin outliers	-0.1106	-0.0880	132267.497600	143036.728412
11	SVM sigmoid	Datos escalados/Sin outliers	-0.1107	-0.0880	132272.865944	143041.491491
6	SVM rbf	Sin procesar/Sin outliers	-0.1105	-0.0879	132262.552613	143032.003255
8	SVM linear	Datos escalados/Sin outliers	-0.1105	-0.0878	132250.642532	143017.931102
9	SVM poly	Datos escalados/Sin outliers	-0.1097	-0.0870	132191.334323	142955.928096
5	SVM poly	Sin procesar/Sin outliers	-0.1033	-0.0827	131935.423900	142763.350407

Los resultados tampoco son demasiado buenos, pero si podemos apreciar mejoras con respecto al análisis realizado para los datos completos. En primer lugar obtenemos valores MAE más bajos, lo que indica que el error en el precio predicho es menor.



En este caso los datos sin procesar vuelven a tener mejores resultados que los datos escalados. La **regresión lineal es la que obtiene los mejores resultados** con un R2 train del 79,28% y un R2 test del 73,54%. El error medio del precio predicho es de aproximadamente 75 mil euros, muy inferior a los presentados en los datos completos antes de excluir outliers.

Otro modelo interesante es el **xgboost Regresion**, a pesar de tener un R2 test inferior, del 70,13%, presenta un **MAE** de aproximadamente 68 mil euros, inferior al de la regresión lineal en unos 7 mil euros.

Tomaremos ambos modelos como relevantes para seguir analizando y trabajandolos para obtener mejoras en el rendimiento.

Modelo final

Tras realizar todo el estudio y evaluación de modelos finalmente se decide optar por un modelo de regresión lineal para los datos excluyendo los outliers.

En este caso, seguimos el mismo procedimiento, importamos las librerías necesarias, hacemos una lectura del cvs correspondiente, definimos las funciones que evaluaran el modelo, dividimos el dataset en train y test y generamos la instancia (lr).
Procedemos a entrenar el modelo y a aplicarlo a los datos de prueba.

En este caso no hemos llamado a las funciones definidas en el notebook con los datos de división de train y test guardados. Se realiza todo paso a paso para que la división de los datos se realizara aleatoriamente y así poder ver el rendimiento del modelo con otros datos a los analizados hasta ahora para comprobar si los resultados siguen siendo similares o mejores.

Se obtienen los siguientes resultados:

	Modelo	Detalles	R2 train	R2 test	MAE train	MAE test
0	Regresión Lineal	Sin procesar/Sin Outliers	0.769808	0.831022	64890.018077	59129.715869

Nuestro modelo tiene una precisión del 83,10% y los precios predichos pueden presentar un error de media de aproximadamente 59 mil euros.

Estos resultados son un poco mejores que los obtenidos en el notebook de evaluación de modelos.

Procedemos a realizar una validación cruzada de 3 folds para ver mejor el rendimiento del modelo. Lo aplicamos en primer lugar a train y obtenemos lo siguiente:

```
from sklearn.model_selection import cross_val_score

# Ejecución de la validación cruzada con 3 folds
scores = cross_val_score(lr, x_train, y_train, cv=3)

# Imprimir la precisión promedio y la desviación estándar
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
Accuracy: 0.67 (+/- 0.19)
```

Los resultados no son todo lo buenos que nos gustaría pero son aceptables, obtenemos una precisión del 67% y una desviación estándar del 19%.

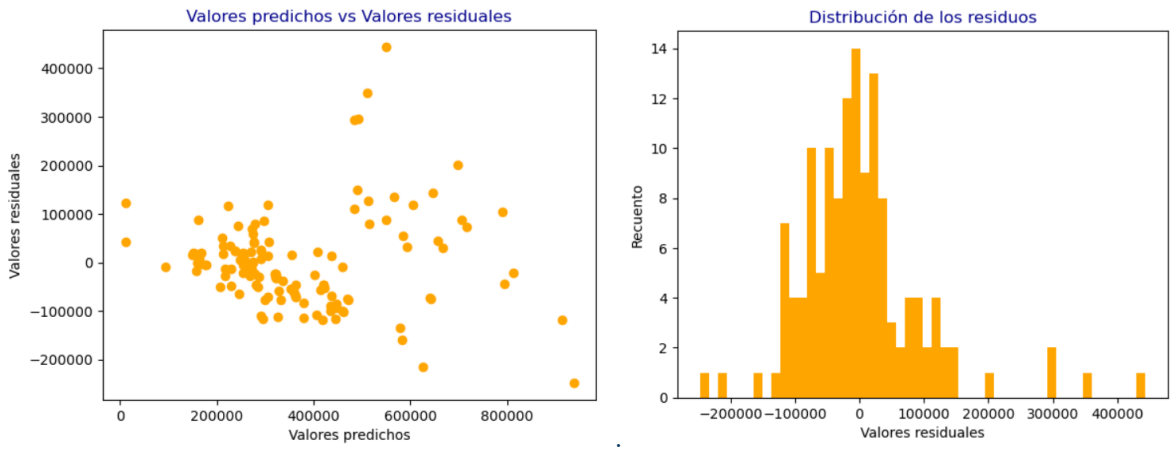
Aplicándolo a test:

```
# Ejecución de la validación cruzada con 3 folds
scores = cross_val_score(lr, x_test, y_test, cv=3)

# Imprimir la precisión promedio y la desviación estándar
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
Accuracy: 0.34 (+/- 0.86)
```

En este caso los resultados son bastante malos, una precisión inferior al 40% y una desviación muy elevada, del 86%.
Para poder analizar con más detalle la calidad del modelo procedemos a representar gráficamente una nube de puntos para visualizar la relación entre las predicciones y los valores residuales, entendiéndose como valor residual la diferencia entre los valores reales y los predichos.

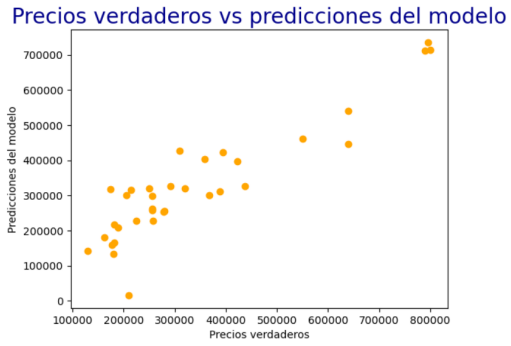


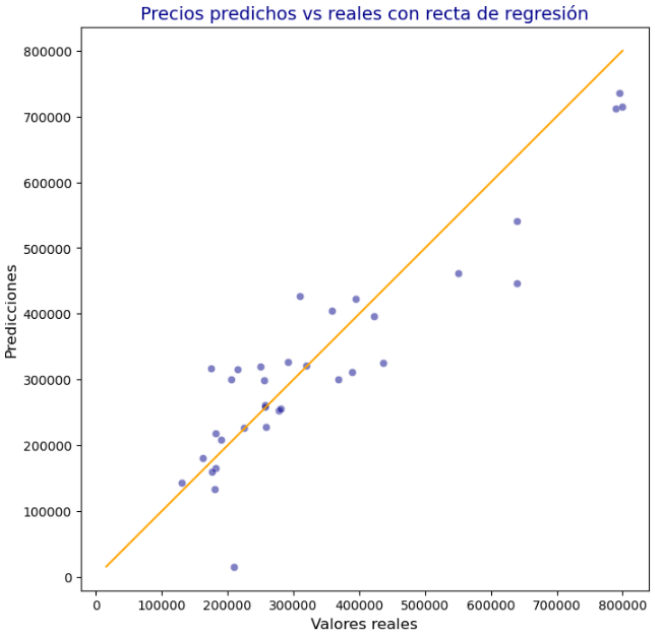
Aunque podemos apreciar como la mayoría de los puntos se encuentran cerca de 0, que indica que no hay gran diferencia entre los precios predichos y los reales, si podemos ver otros puntos que presentan diferencias significativas, tanto en valores positivos como en los negativos tomando como referencia el eje y. Esto puede ser indicativo de que el modelo no está capturando completamente la complejidad de la relación entre las variables. Además puede deberse a la poca cantidad de datos usados en el proyecto.

Los valores negativos indican que en algunos casos el precio predicho es inferior al real. En otras palabras, el modelo está subestimando los valores en algunos casos.

Por otro lado representamos los mismos datos con un histograma para ver la distribución de los mismos. Podemos decir que se distribuyen de forma normal lo que indica que el modelo es adecuado y se ajusta bien a la mayoría de datos ya que se puede apreciar algunas excepciones de residuos con valores muy altos y por otro lado, muy bajos.

Analizamos con gráfica la distribución de los precios predichos con los reales.





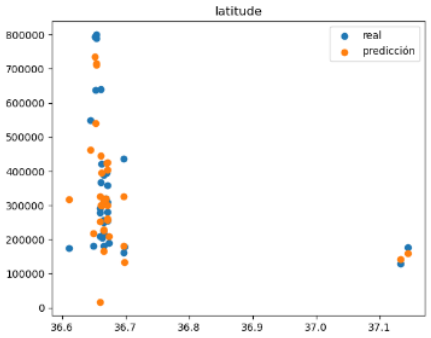
Sacamos las siguientes conclusiones:

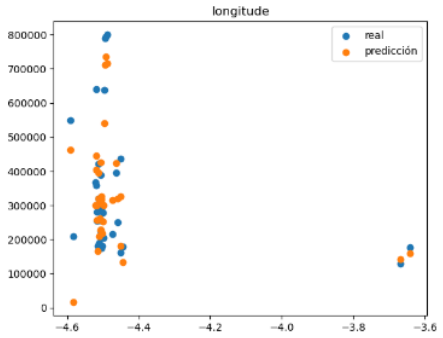
Cuanto más cerca estén los puntos de dispersión de la línea de regresión, mejor será el modelo. El R^2 obtenido fue de 0,83, que es un puntaje relativamente aceptable. Esto da como resultado que todos los puntos estén cerca de la línea diagonal.

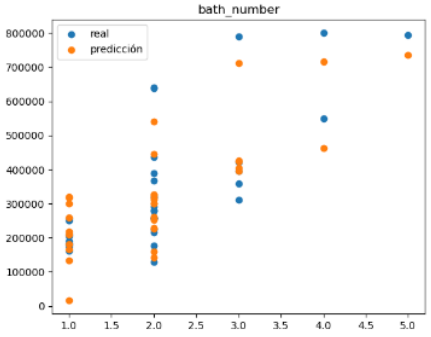
Si analizamos más a fondo los valores, podemos concluir que cuanto más alto es el precio de la vivienda, más disperso es el modelo. Esta afirmación puede deberse a la poca cantidad de datos para viviendas con precios superiores, por lo que era de esperar que el modelo ajustara mejor los precios de viviendas medias y no de lujo: superiores al millón de euros. A pesar de excluir los outliers (17 viviendas), la mayor cantidad de viviendas del conjunto de datos presentan precios entre los 200.000 y 300.000 euros.

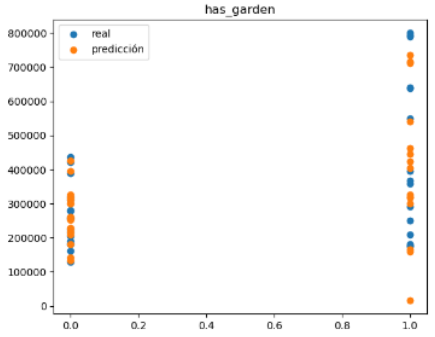
En cualquier caso, el rendimiento de nuestro modelo será más que aceptable ya que los puntos se ajustan a la recta de regresión y el valor de R^2 es cercano al 84% de efectividad.

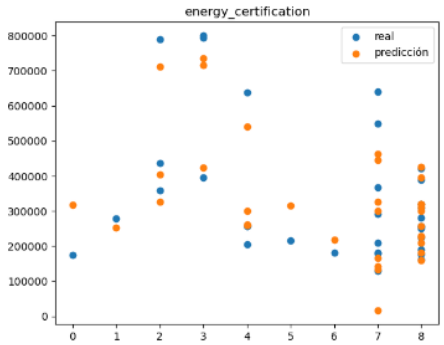
Por otro lado analizamos los valores predichos en función del resto de variables del conjunto, y volvemos a apreciar que a mayores precios peor ajuste.

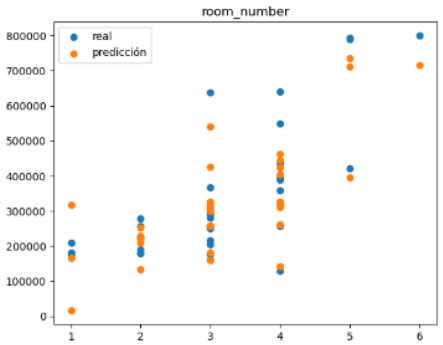

latitude
real
predicción

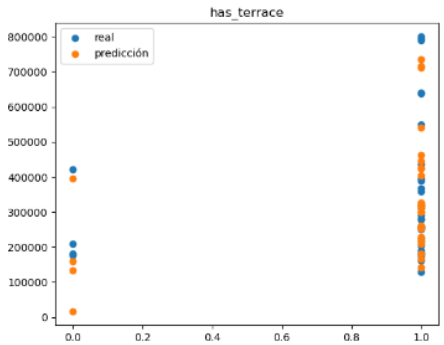

longitude
real
predicción

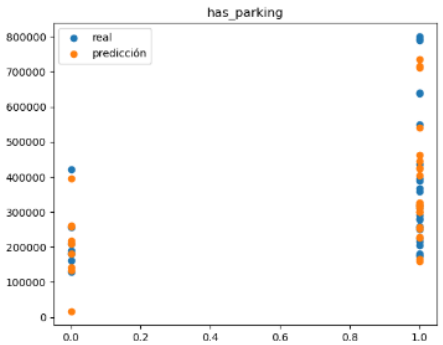

bath_number
real
predicción

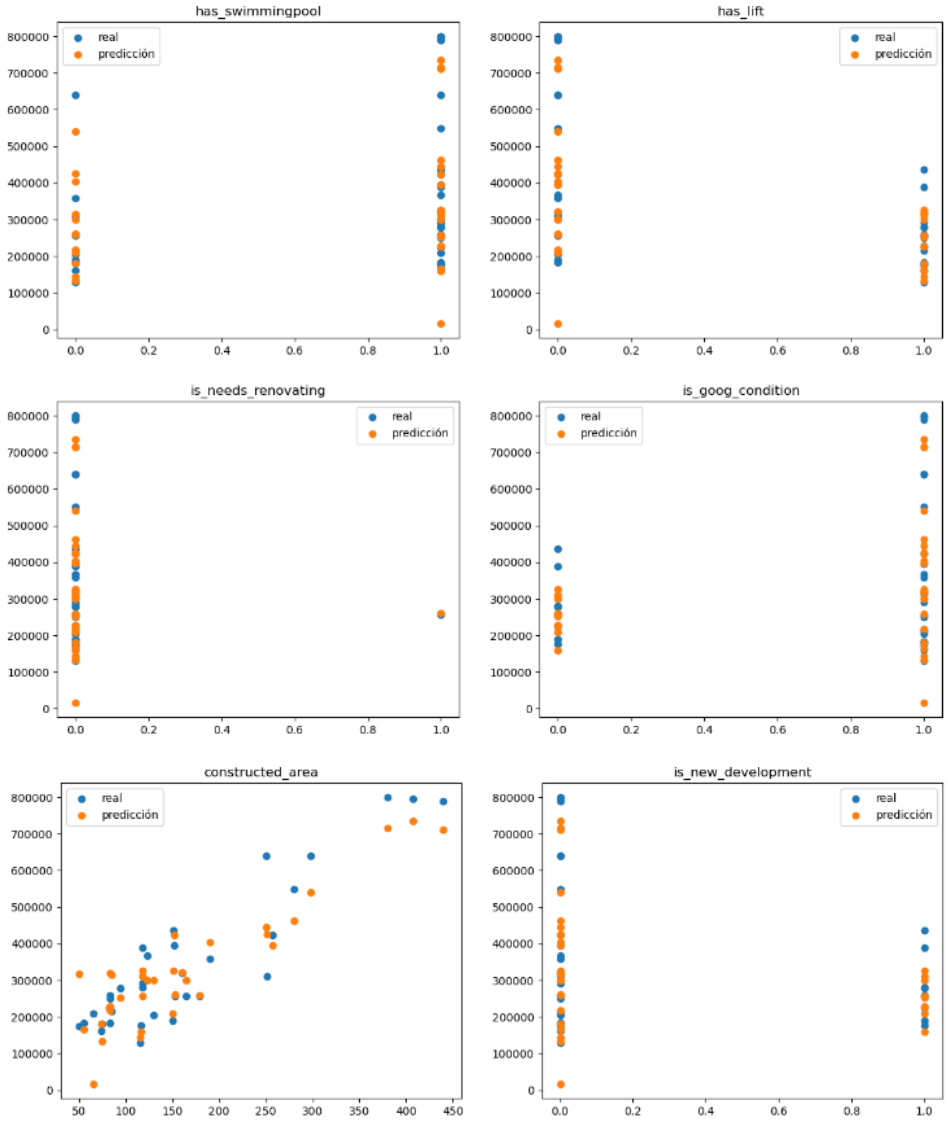

has_garden
real
predicción


energy_certification
real
predicción


room_number
real
predicción


has_terrace
real
predicción


has_parking
real
predicción



Aplicación a nuevos datos

Una vez analizado el modelo aplicamos el mismo a nuevos datos, en este caso ya que no disponemos de más datos de la zona lo aplicaremos al dataset inicial. Esto no es lo más adecuado pero procederemos así ya que no disponemos de otros datos.

Generación dataset final front-end

Generamos la variable data_pred con los precios predichos para todo el dataset aplicando lr sobre todas las features. En total, un conjunto de 169 inmuebles a la venta y 17 columnas.

Generamos las siguientes columnas al dataset:

- Columna [price_predict] con los precios predichos.
- Columna [Var_Price] con la diferencia entre precios predichos y reales.
- Columna [%_rentabilidad] con la rentabilidad de las viviendas.

Además para completar el dataset volvemos a unir las columnas del dataset inicial previamente eliminadas para el entrenamiento del modelo: [titulo] y [localizacion]. Para ellos aplicamos la función de pandas concat y reordenarmos las columnas para que aparezcan en primera posición.

Finalmente nuestro dataset final para la visualización front-end tiene las siguientes características:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 170 entries, 0 to 169
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   titulo                170 non-null    object 
 1   localizacion          170 non-null    object 
 2   latitude              170 non-null    float64
 3   longitude             170 non-null    float64
 4   price                 170 non-null    int64  
 5   energy_certification  170 non-null    int64  
 6   room_number           170 non-null    int64  
 7   bath_number           170 non-null    int64  
 8   has_garden            170 non-null    float64
 9   has_terrace           170 non-null    float64
 10  has_parking           170 non-null    float64
 11  has_swimmingpool      170 non-null    float64
 12  has_lift              170 non-null    float64
 13  constructed_area      170 non-null    int64  
 14  is_new_development    170 non-null    int64  
 15  is_needs_renovating   170 non-null    int64  
 16  is_goog_condition     170 non-null    int64  
 17  Price_predict         170 non-null    float64
 18  Var_Prices            170 non-null    float64
 19  %_rentabilidad        170 non-null    float64
dtypes: float64(10), int64(8), object(2)
memory usage: 26.7+ KB
```

FRONT END

Para este apartado se ha creado una app usando la librería de Streamlit con dos opciones, una primera que da respuesta a lo planteado en el proyecto, ¿cuáles son las viviendas más rentables de la zona? y se incorpora otra funcionalidad, que es un calculador del precio de la viviendas, ¿cuánto vale mi vivienda?

Para reflejar todo esto se prepara un front end interactivo en el que el usuario puede ver con un simple vistazo cuáles son las oportunidades de inversiones actuales en el mercado. A partir de un filtro de rentabilidad, el usuario puede elegir qué rentabilidad mínima quiere que tengan las viviendas a mostrar, además de otros filtro que puede usar adicionalmente para hacer que la búsqueda y la consulta de información se adapte a las necesidades de cada persona.

Se establecen filtros por precio, número de habitaciones, número de baños, metros cuadrados, si dispone de algunas características como piscina, garaje, jardín, etc. Además, se añade la opción de ordenar las viviendas según los siguientes criterios:
Viviendas con mayor rentabilidad.
Viviendas con menor precio.
Viviendas con mayor precio.

El usuario podrá consultar toda esta información de la zona de Churriana, que es en la que está basado el proyecto. El proyecto podría nutrirse como mejora futura de mayor cantidad de datos de más zonas para aportar mayor utilidad a los usuarios que la consultan, además mayor volumen de datos mejoraría la capacidad de entrenamiento del modelo y se podría ajustar mejor para unos resultados más precisos y acertados.

Una aplicación muy sencilla e intuitiva que facilita las decisiones de inversiones.

App:

Primera parte de la app, Buscador de viviendas rentables:

🔑 BUSCADOR DE VIVIENDAS CON RENTABILIDAD

BARRIO DE CHURRIANA, MÁLAGA

Buscardor de viviendas rentables | ¿Cuál es el precio de mi vivienda?

Aplicar Filtros

Rentabilidad mínima: -92.48 (-92.48 / 81.05)

Precio Máximo: 995000 (54900 / 995000)

Selecciona el número de habitaciones: 1 (1 / 8)

Selecciona el número de baños: 1 (1 / 5)

Mínimo Metros cuadrados: 50 (50 / 702)

- [ ] Piscina
- [ ] Jardín
- [ ] Ascensor
- [ ] parking
- [ ] Terraza
- [ ] Nueva Construcción
- [ ] Necesita Reforma
- [ ] En Buena Condición

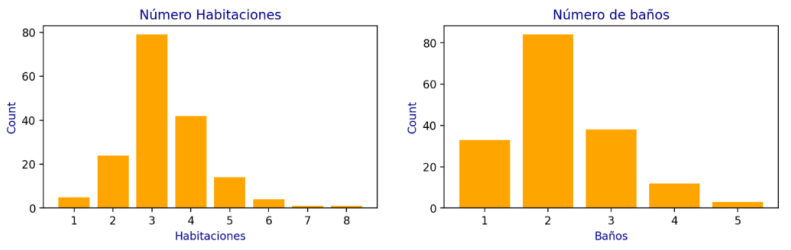
Rentabilidad positiva: **95** Viviendas en Churriana

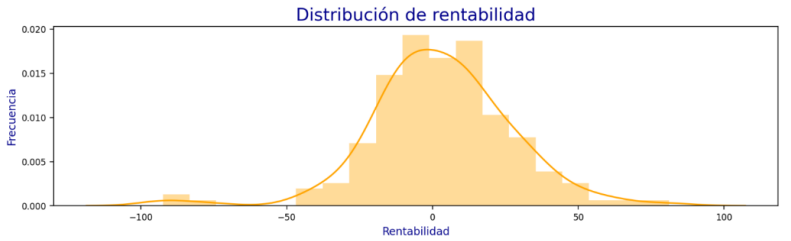
Número de viviendas: **170**

Precio medio: **360,341 €**

Area construida media: **181 m2**

	localizacion	latitude	longitude	price	ene
Piso en venta en calle Miami, 7	La Carihuela - Los Nidos, Torremolinos	36.6107	-4.5038	175000	
Chalet adosado en venta en RIBERA	San Francisco - El Chorrillo, Las Gabias	37.1329	-3.6706	179999	
Chalet adosado en venta en calle Avempace, 10	Cortijo de Maza-Finca Monsalvez-El Olivar, Málaga	36.6487	-4.5018	182100	
Casa o chalet independiente en venta en Churriana-El Pizarrillo-La Noria-Guadalsol	Churriana, Málaga	36.6656	-4.4723	410000	
Casa o chalet independiente en venta en calle Decano Juan Antonio Rando	Churriana-El Pizarrillo-La Noria-Guadalsol, Málaga	36.6689	-4.4762	215000	
Piso en venta en calle Decano Joaquín Wuester, 3	Churriana-El Pizarrillo-La Noria-Guadalsol, Málaga	36.6659	-4.4731	215000	
Chalet adosado en venta en Churriana-El Pizarrillo-La Noria-Guadalsol	Churriana, Málaga	36.6634	-4.4986	205000	
Casa o chalet en venta en Churriana-El Pizarrillo-La Noria-Guadalsol	Churriana, Málaga	36.6633	-4.5029	264500	
Chalet adosado en venta en Churriana-El Pizarrillo-La Noria-Guadalsol	Churriana, Málaga	36.6634	-4.4989	299990	
Casa o chalet independiente en venta en carretera de Coin, 108	Churriana-El Pizarrillo-La Noria-Guadalsol, Málaga	36.6618	-4.5131	422500	





Made with Streamlit

Esta primera parte muestra el conjunto de datos final, e indica el número total de viviendas con rentabilidad positiva.
En la columna izquierda se puede ir filtrando y los datos se irán modificando automáticamente en función de las características seleccionadas.

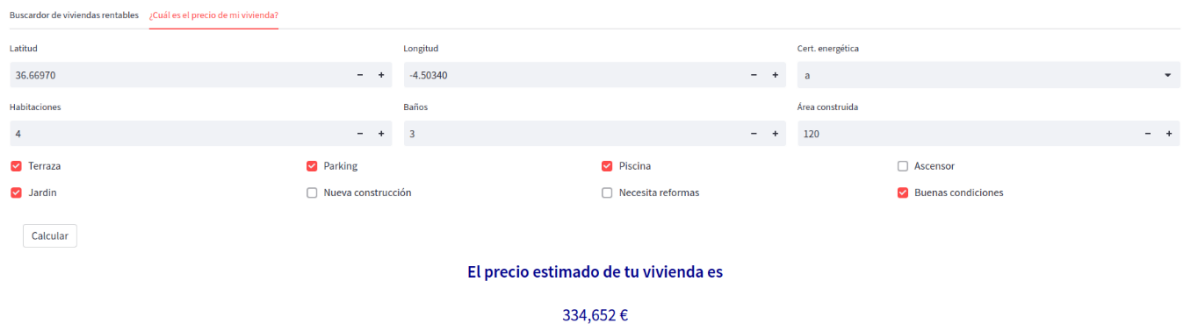
Aparecerá una etiqueta con el número de viviendas disponibles con esas características, el precio medio y los metros construidos de media.

Se podrá ver las viviendas que cumplen esos requisitos en una tabla que contiene toda la información ordenada por las viviendas más rentables en orden descendente y además se acompaña con 3 gráficos, dos gráficos de barras para ver el número de habitaciones y los baños y un gráfico final de la distribución de la rentabilidad de las viviendas resultantes tras los filtros aplicados.

Segunda parte, ¿cuál es el precio de mi vivienda?

En esta segunda parte de la app podemos ver el calculador de precios, en función de las características de la vivienda insertada la app te dará un precio estimado de la vivienda.

Un ejemplo del funcionamiento de dicha sección con los siguientes datos dados de una vivienda inventada (caso de uso como si fuera mi vivienda y quisiera ponerla a la venta):



Para los siguientes datos de entrada nos indica que el precio de nuestra vivienda estimado sería de 334.652€. Esto nos da una idea de por dónde andan los precios de la zona y nos sirve de punto de partida para poder establecer un precio justo y aceptable.

CONCLUSIONES Y MEJORAS

El proyecto a pesar de haberse desarrollado con todos sus pasos no es un modelo que se pueda usar en la vida real. El modelo ha presentado malos ajustes por overfiting, había muy pocos datos para poder llevar con éxito un modelo de ML.

El Scrapping desde primera hora me ha dado problemas, tras dedicar demasiado tiempo a esa parte en un principio me conformé con los datos que tenía con idea de mejorar esa parte en un futuro cuando tuviera más desarrollado las otras partes del proyecto. Finalmente no he logrado sacar el tiempo suficiente para poder buscar una solución. Tras analizar otras opciones quizás utilizando Selenium podríamos evitar el captcha de Idealista. O incluso probar con otros portales del sector.

A pesar de que los resultados no son buenos, el trabajo realizado y el tiempo empleado en pelearme con cada paso dado me ha servido para refrescar muchos conceptos y aprender mucho sobre nuevas cosas. Es la primera vez que me enfrento a un caso real desde inicio a fin y aunque me queda mucho por aprender este es el inicio de un largo camino.

Mejoras:

- Los datos usados son de noviembre de 2022, muchas viviendas ya no están disponibles a la venta y hay nuevas. El modelo se podría mejorar si pudiéramos extraer de forma automatizada diariamente los datos de las viviendas disponibles.
- Para hacer un análisis con datos representativos es necesario scrapear mayor volumen de datos de viviendas así como de mayor cantidad de características por vivienda. Entre ellas

la variación de precio de una vivienda ya que a mayor tiempo si no se vende suelen ir bajando el precio, sería interesante capturar estos datos, pero para ello se debería tener una carga automática diaria.

- Sería interesante incluir datos externos de la zona para nutrir el estudio y la predicción.
- En el proyecto solo se analiza la zona de Churriana, Málaga, se podría escalar el mismo y ampliar las zonas analizadas para una mayor visión y poder hacer análisis comparativos entre unas zonas y otras.
- Disponer de mayor cantidad de datos permitiría realizar un modelo más realista para su uso en la vida real.
- En ciertas partes del proyecto como en la limpieza de datos, se podría mejorar aplicando funciones que nos permitieran que si posteriormente se quisieran incorporar nuevos datos de entrada de otras zonas se pudiera aplicar automáticamente llamando a la función y no teniendo que cambiar todo de forma manual.
- Para mejorar la visualización se podrían incluir los datos de contacto del propietario así como las imágenes de la vivienda para que sin salir de la misma puedan consultar toda la información necesaria así como contactar en caso deseado.
- Se podría incluir un buscador dentro del frontend para que el usuario pudiera buscar la zona que desea y que en tiempo real le diera los resultados del mismo.

BIBLIOGRAFÍA

He explorado cientos de páginas pero no guardé los enlaces de la mayoría. Pongo algunos de los mismos que he conseguido recabar:

- Scrapeo con BeautifulSoup [https://realpython.com/beautiful-soup-web-scraper-python/]
- Aprendizaje supervisado modelos y aplicaciones [https://scikit-learn.org/stable/supervised_learning.html#supervised-learning]
- Instalación de miniconda [https://docs.conda.io/en/latest/miniconda.html]
- Almacenar variables en jupyter [https://www.analyticslane.com/2023/01/30/almacenar-variables-en-jupyter/]
- Documentación API librería Streamlit [https://docs.streamlit.io/]
- Librería Pickle [https://docs.python.org/es/3/library/pickle.html]
- API de librería Matplotlib [https://matplotlib.org/stable/api]
- Librería Pandas [https://pandas.pydata.org/docs/user_guide/index.html#user-guide]
- Scikit-Learn [https://scikit-learn.org/stable/modules/classes.html]
- Compartir variables entre diferentes notebooks [https://thecodingbot.com/how-to-share-variables-between-two-different-jupyter-notebooks/]
- Librería Seaborn [https://seaborn.pydata.org/index.html]
- Guía de uso GIT [https://rogerdudler.github.io/git-guide/]