

# Machine Learning

aplicado al

# Mercado Inmobiliario

*Detectando oportunidades de Inversión con tecnologías de bajo coste al alcance de las pymes*

Descubre, en este caso teórico-práctico desarrollado con python, por qué el conocimiento se ha convertido en el activo más valioso en el entorno empresarial actual y cómo las pymes están en una posición única para aprovecharlo y competir con gigantes corporativos en un mundo donde el tamaño de tu empresa ya no determina el éxito, sino la capacidad para aprovechar el poder del conocimiento. La revolución está aquí, y está al alcance de tu mano. ¿Estás listo para ser parte de ella?



Enrique Aranaz Tudela

FREELANCER | DATA DRIVEN | CONSULTOR BI | DATA SCIENTIST | MACHINE LEARNING PYTHON | POWER BI



El futuro pertenece a aquellos que entienden cómo utilizar sus datos de manera efectiva

**01 Introducción: El poder del conocimiento en la era digital**

**02 Caso práctico: InverSmart Realty**

**03 Que lecciones se aprenden de este caso**

# INDICE CONTENIDO

**04 Conclusiones: Una Era de Oportunidades para las PYMES**

**Anexo: detalle proyecto en python**



## 01 Introducción: El poder del conocimiento en la era digital

Imagina un mundo donde una pequeña panadería de barrio predice con precisión la demanda diaria de pan, reduciendo el desperdicio a casi cero. O una tienda local de ropa que personaliza sus ofertas para cada cliente con la misma eficacia que Amazon. Este no es un escenario de ciencia ficción; es el presente de las pequeñas y medianas empresas que han descubierto el verdadero poder del conocimiento en la era digital.

### La democratización del conocimiento

En un giro irónico del destino, las mismas tecnologías que una vez amenazaron con dejar atrás a las pequeñas empresas se han convertido en su salvación. En la era de la información, el conocimiento no es solo poder; es supervivencia. Y éste, lejos de ser del dominio exclusivo de los gigantes tecnológicos, se ha convertido en el campo de batalla donde las pymes, armadas tan sólo con su visión audaz y tecnologías gratuitas de código abierto como Python, chatbots de inteligencia artificial y plataformas de análisis de datos como Power BI están ganando terreno rápidamente.

En el 2025 los datos creados en el mundo se habrán multiplicado por 10 desde 2016. Pero los datos por sí solos no son suficientes; es la capacidad de extraer conocimiento de estos datos lo que marca la diferencia.



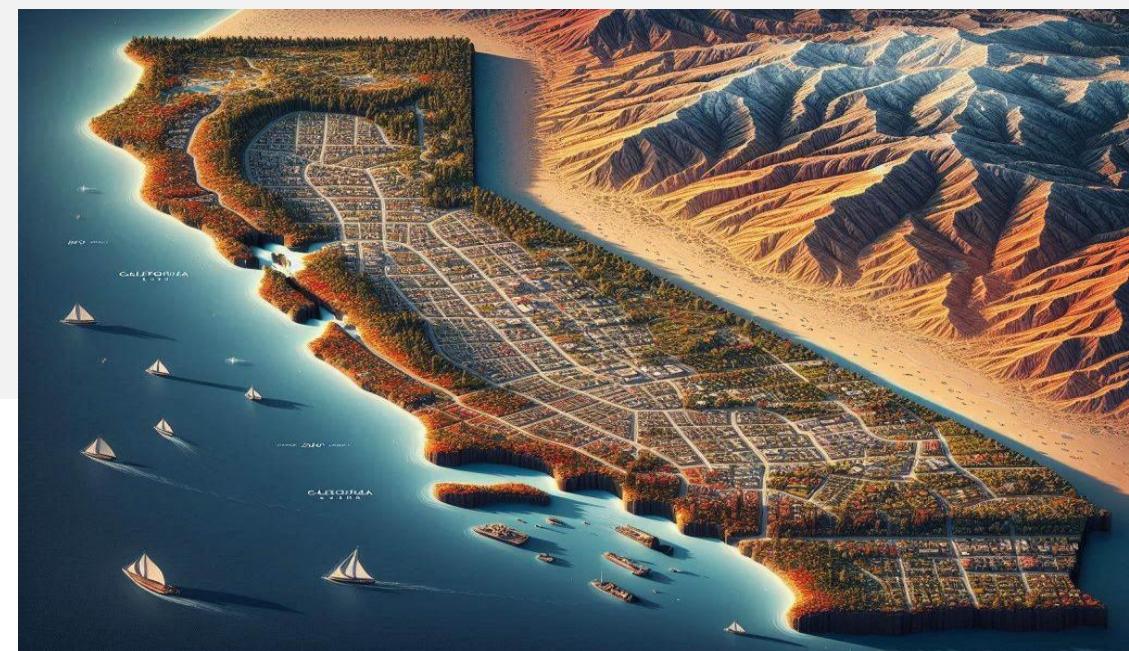
### Las pymes en la vanguardia de la revolución del conocimiento

Sorprendentemente, son las pymes las que están en una posición única para aprovechar esta revolución del conocimiento. Su agilidad y capacidad de adaptación les permiten implementar soluciones basadas en datos más rápidamente que las grandes corporaciones.

## 02 Caso práctico: InverSmart Realty

En este caso de estudio , muestro, a partir de datos reales publicados por la Oficina del Censo de los EE. UU. sobre el mercado inmobiliario de California a nivel Bloque\*, cómo cualquier pyme, utilizando una herramienta gratuita y de código abierto como python, puede aprovechar el acceso a multitud de fuentes de datos de su sector y el machine learning para transformar su negocio y obtener ventajas competitivas significativas.

Si bien los datos y el modelo de machine learning desarrollado en Python son completamente reales, a efectos prácticos creare una empresa ficticia a la que llamaré **InverSmart Realty**.



\* Un “bloque” en EEUU es similar a una “manzana” en España. Ambos términos se refieren a una unidad de medida urbana que comprende un grupo de edificios delimitados por calles.

## InverSmart Realty

InverSmart Realty es una pequeña empresa inmobiliaria con sede en California, especializada en identificar propiedades con potencial de inversión para sus clientes. Con un equipo de solo 15 personas, compiten contra grandes firmas inmobiliarias gracias a su enfoque data-driven.

### Desafío

En un mercado inmobiliario altamente competitivo, InverSmart Realty necesitaba una manera de identificar rápidamente propiedades infravaloradas o áreas con alto potencial de crecimiento antes que sus competidores.

### Solución

Utilizando el lenguaje de programación python (de código libre y gratuito) y herramientas de visualización como Power BI, han sido capaces de:

1. Extraer datos en tiempo real y de forma automática\* de una diversidad de fuentes de su sector (privadas, públicas, de su competencia, opiniones de sus clientes, redes sociales, etc.), que dan soporte en tiempo real a toda su estrategia data-driven.
2. Utilizando los datos publicados por la Oficina del Censo de los EE. UU. sobre el mercado inmobiliario de California a nivel grupo de bloques han conseguido segmentar el mercado en 8 zonas geográficas con perfiles similares de viviendas en cuanto a características inmobiliarias y precios.

A continuación, mostramos un resumen ejecutivo de la metodología empleada (para más detalle y código más relevante del mismo ver anexo).

\* mediante el uso de apps y técnicas de web scraping

#### Descripción del conjunto de datos:

El conjunto de datos tiene 20.640 registros correspondientes a el número total de bloques del censo de California, con 10 métricas por registro:

**Longitude y latitude:** Ubicación geográfica del bloque

**housing\_median\_age:** Edad media de las casas en el bloque

**total\_rooms:** Conteo del número total de habitaciones en todas las casas del bloque

**total\_bedrooms:** Conteo del número total de dormitorios en todas las casas del bloque

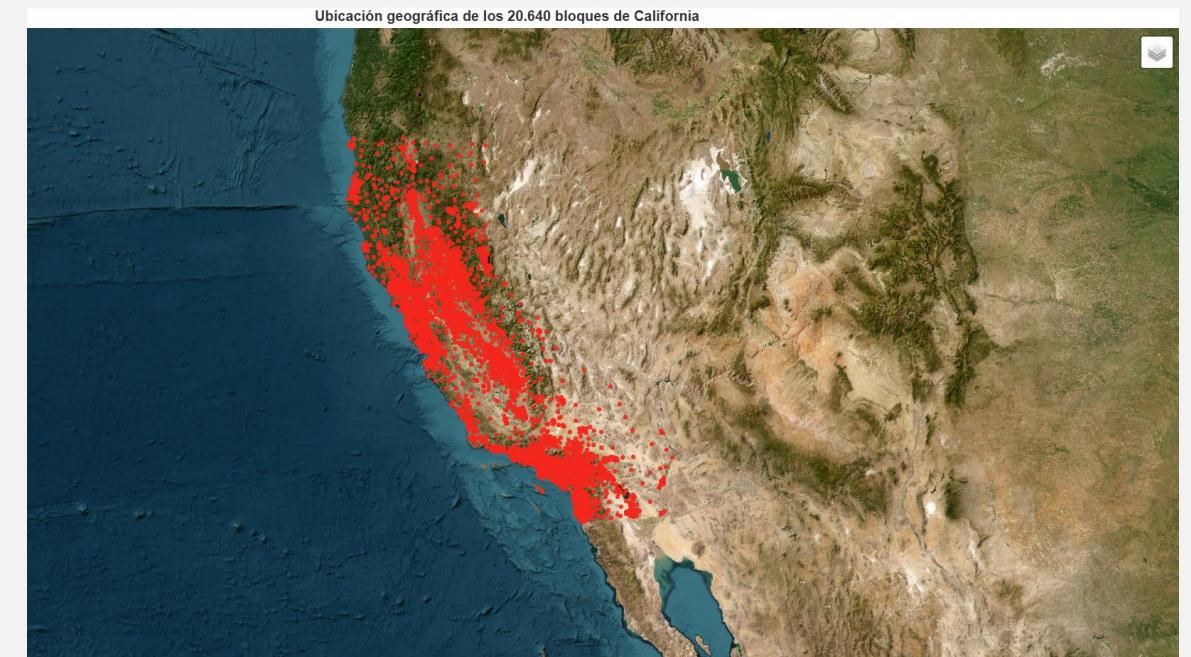
**Population:** Conteo del número total de población en el bloque

**Households:** Conteo del número total de hogares en el bloque

**median\_income:** Mediana del ingreso total de los hogares en todas las casas del bloque

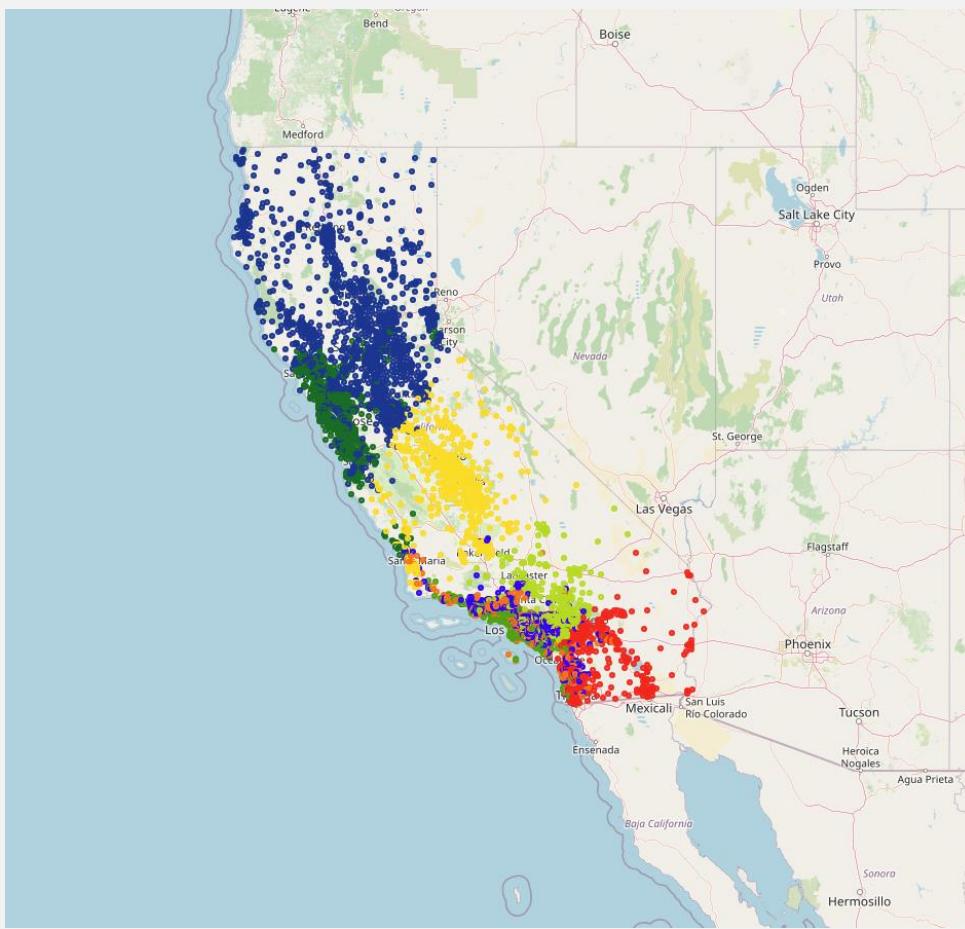
**ocean\_proximity:** Tipo de paisaje del bloque (isla, cerca bahía, cerca Océano, <1 hora, en el interior)

**median\_house\_value:** Mediana de los precios de las casas en el bloque



Mapa web interactivo de California creada con la librería Folium de python que muestra la situación geográfica de los 20.640 bloques que conforman la información censal.

Para segmentar el mercado inmobiliario se utilizó la técnica de K-means\*, con lo que se consiguió identificar 8 zonas geográficas con perfiles de precio y características distintos.

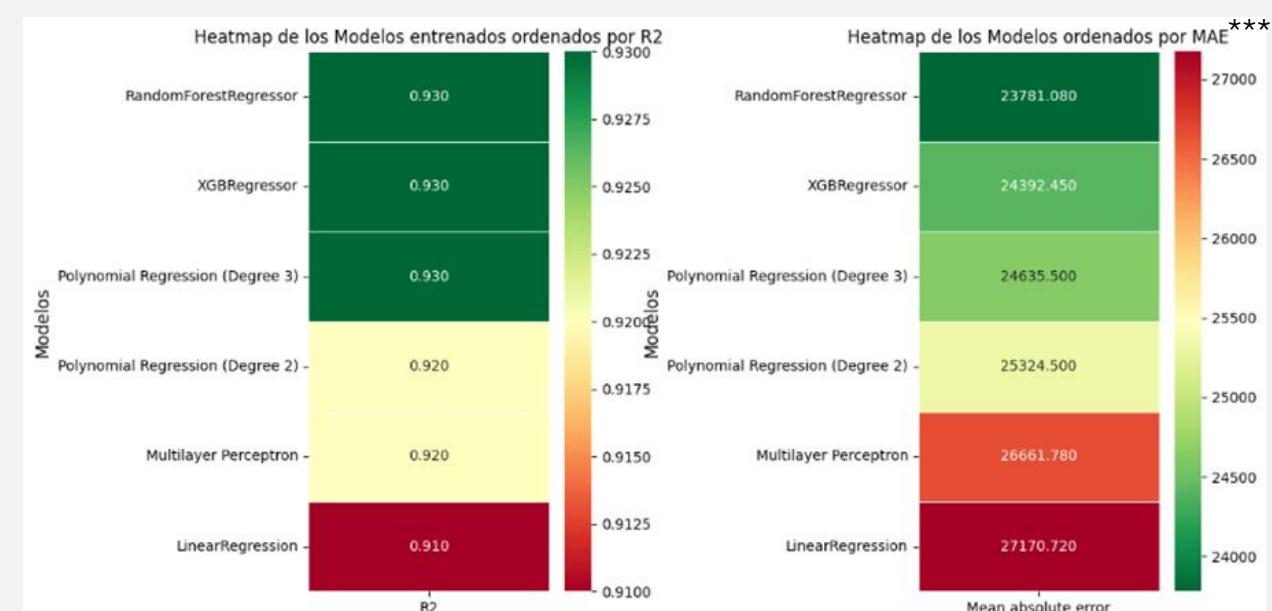


Mapa web interactivo de California con los 20.640 bloques segmentados.

\* la técnica de K-means es un algoritmo de aprendizaje no supervisado utilizado para agrupar datos en diferentes clusters o segmentos. Es especialmente útil en la segmentación de mercados, donde se busca identificar grupos de clientes o productos con comportamientos o características similares.

Utilizando esta segmentación del mercado inmobiliario en 8 zonas y la información disponible del censo a nivel bloque (Ingresos medios, población, número de hogares, habitaciones, etc.) elaboraron un modelo predictivo para predecir el precio medio de la vivienda en cada uno de los 20.640 bloques a partir de sus características:

- Para ello entrenaron varios modelos de Machine Learning (RandomForest, XGBRegressor, Linear Regresión, Polinomial Regression de grado 2 y 3) y Deep Learning (Multilayer Perceptron) que dieron un alto nivel de precisión, con  $R^2$  del orden de 0.93\*\*, que sugiere un alto poder predictivo.

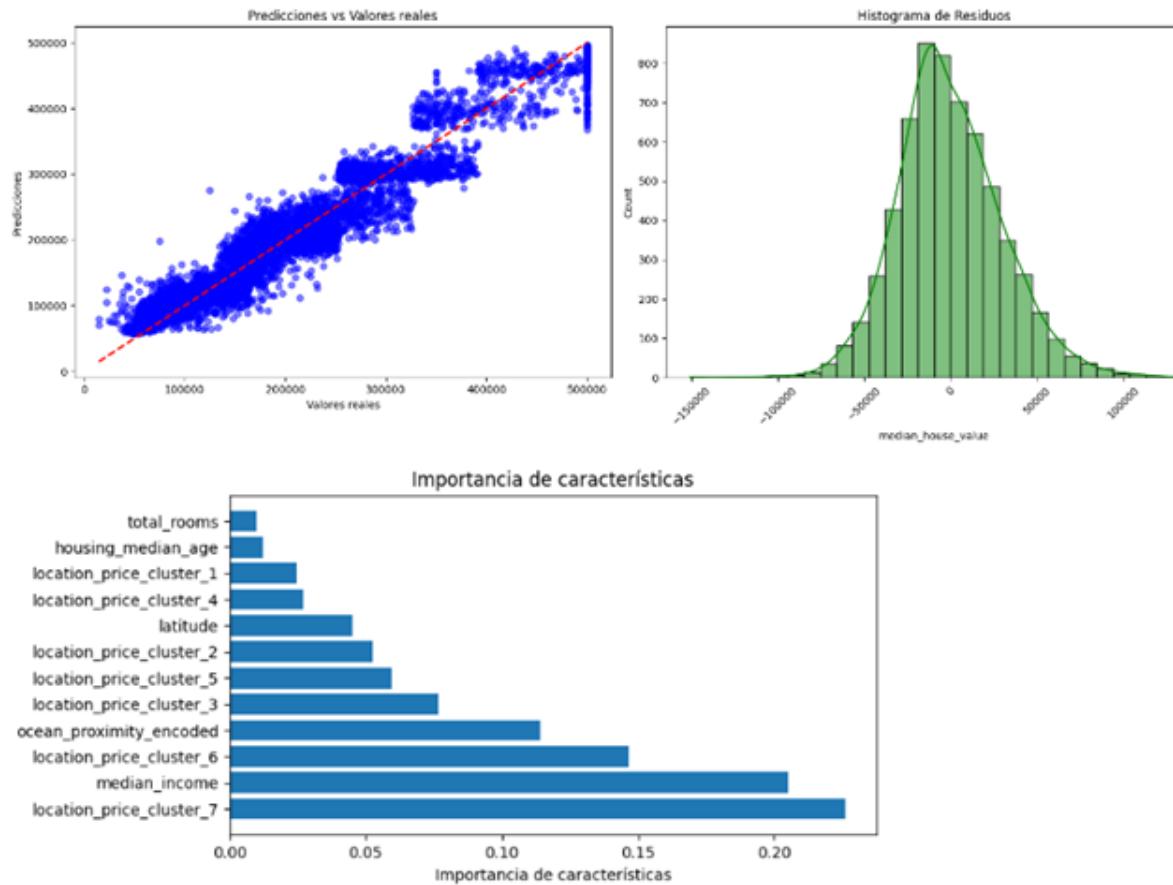


\*\* En términos simples, un  $R^2$  de 0.93 significa que el 93% de la variabilidad en los precios de las viviendas puede ser explicada por el modelo basándose en las variables que ha utilizado, lo que muestra un modelo muy preciso.

\*\*\* El **Error Absoluto Medio (MAE)** es una métrica utilizada para medir la precisión de un modelo de predicción. Un valor de 23.000 quiere decir que es capaz de predecir el precio medio de un bloque con una precisión en promedio de + - 23.000 \$ sobre su valor real.

## Resultados obtenidos por el modelo RandomForest\*:

Modelo: RandomForestRegressor  
Mean Squared Error: 922715905.82  
Mean Absolute Error: 23781.08  
R-Squared: 0.93



\* Es un modelo de aprendizaje automático basado en árboles de decisión. Crea múltiples árboles de decisión y promedia sus predicciones.

Apropiado para estimar precios de viviendas: Puede capturar relaciones complejas entre las características (como tamaño, ubicación, número de habitaciones,etc.) y el precio.

3. Además, con objeto de tener una visión más precisa del mercado inmobiliario de California, utilizaron la potencia y versatilidad de Python para conectarse - vía apps y aplicando técnicas de web scraping - a una gran variedad de fuentes de datos sobre su sector y extraer así información relevante en tiempo real.

### Fuentes de Datos del Mercado Inmobiliario de California

#### Portales Inmobiliarios

##### Zillow

URL: <https://www.zillow.com>, uno de los portales más grandes de EE.UU. con amplia cobertura en California. Ofrece API ([Zillow API](#)).

##### Realtor.com

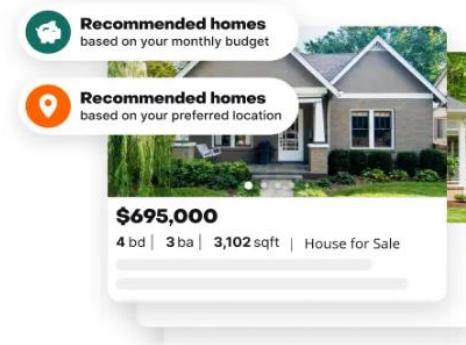
URL: <https://www.realtor.com>, portal oficial de la National Association of Realtors.

##### Redfin

URL: <https://www.redfin.com>, ofrece datos detallados y actualizados frecuentemente.

##### Homes.com

URL: <https://www.homes.com>, Buena cobertura de propiedades en California.



[Back to search](#)



[Save](#) [Share](#) [Hide](#) [...](#)



**\$739,000**  
5061 La Cuenta Dr, San Diego, CA 92124

4 beds 2 baths 1,450 sqft

[Request a tour](#)  
as early as today at 11:00 am

[Contact agent](#)

Est.: \$5,076/mo [Get pre-qualified](#)

Condominium

Built in 1974

-- sqft lot

\$752,400 Zestimate®

\$510/sqft

\$525/mo HOA

## What's special

SERENE BACKYARD EN-SUITE BATHROOM SPACIOUS PRIMARY SUITE LIGHT-FILLED HOME

BRAND NEW FLOORING ELEGANT HARDWOOD FLOORS STAINLESS STEEL APPLIANCES

Tucked away from the busy streets, this beautifully upgraded, light-filled home is located in the desirable Tierrasanta neighborhood of San Diego. With brand new flooring and paint throughout, this charming 3-bedroom, 2-bathroom residence also includes an optional loft bedroom, offering 1,450 sq ft of living space. Featuring elegant hardwood floors and vaulted ceilings, the home is situated in a quiet part of the complex.

[Show more](#)

6 days on Zillow | 1401 views | 80 saves | Likely to sell faster than [94% nearby](#)

**Ejemplo de base de datos de viviendas en venta tras ser extraídas de la web de Zillow utilizando Python y empleando conexiones vía APIs\* o a través de web scraping\*\***

[3]:

|   | Dirección                                   | Precio  | Precio_Lista | Habitaciones | Baños | Tamaño | Tipo_Vivienda | Año_Construcción | Características_Relevantes                         | Vistas | Saves | Días_Ofera |
|---|---|---------|--------------|--------------|-------|--------|---------------|------------------|--|--------|-------|------------|
| 0 | 1234 Sunset Blvd, Los Ángeles, CA 90001     | 1250000 | 1300000      | 4            | 3.0   | 220    | Unifamiliar   | 2015             | Diseño moderno - Entorno escénico - Electrodom...  | 1250   | 45    | 30         |
| 1 | 567 Ocean Drive, San Francisco, CA 94122    | 2100000 | 2200000      | 3            | 2.5   | 180    | Condominio    | 2010             | Vista privada al cañón - Electrodomésticos de ...  | 2100   | 78    | 15         |
| 2 | 890 Valley Road, San Diego, CA 92103        | 850000  | 875000       | 3            | 2.0   | 150    | Adosado       | 2005             | Zona verde con césped - Sistema de riego autom...  | 980    | 32    | 45         |
| 3 | 2468 Mountain View, Sacramento, CA 95814    | 680000  | 700000       | 2            | 2.0   | 120    | Apartamento   | 2018             | Diseño nuevo - Electrodomésticos modernos - Ai...  | 750    | 56    | 22         |
| 4 | 1357 Beachside Ave, Santa Barbara, CA 93101 | 3500000 | 3600000      | 5            | 4.5   | 350    | Unifamiliar   | 2020             | Entorno escénico - Electrodomésticos de cocina...  | 3200   | 102   | 7          |
| 5 | 246 Hillside Terrace, Berkeley, CA 94704    | 1800000 | 1850000      | 4            | 3.0   | 200    | Duplex        | 2000             | Vista privada - Lavadora y secadora - Aire aco...  | 1850   | 67    | 18         |
| 6 | 789 Redwood Lane, Napa, CA 94559            | 1450000 | 1500000      | 3            | 2.5   | 190    | Bungalow      | 2012             | Entorno escénico - Electrodomésticos nuevos - ...  | 1600   | 89    | 25         |
| 7 | 369 Tech Circle, San José, CA 95110         | 1600000 | 1650000      | 4            | 3.0   | 210    | Condominio    | 2017             | Diseño moderno - Electrodomésticos de cocina - ... | 1750   | 72    | 12         |
| 8 | 951 Desert Palm Dr, Palm Springs, CA 92626  | 750000  | 775000       | 3            | 2.0   | 160    | Unifamiliar   | 2008             | Diseño nuevo - Zona verde con césped - Aire ac...  | 890    | 41    | 38         |
| 9 | 753 Coastal Highway, Santa Cruz, CA 95060   | 2300000 | 2350000      | 4            | 3.5   | 240    | Unifamiliar   | 2019             | Vista privada al océano - Electrodomésticos de...  | 2450   | 95    | 10         |

\*Las APIs (Interfaces de Programación de Aplicaciones) son conjuntos de reglas y protocolos que permiten que diferentes aplicaciones se comuniquen entre sí. Ventajas:

- Datos estructurados y fáciles de procesar.
- Más eficiente y rápido que el web scraping.

\*\*Web scraping es una técnica para extraer información de sitios web analizando la estructura HTML de las páginas. Ventajas:

- Permite obtener datos de sitios que no ofrecen una API.
- Flexibilidad para extraer datos de cualquier sitio web público.
- Puede automatizar la recopilación de grandes cantidades de datos.

## Datos Públicos

**Census Bureau:** Datos demográficos y económicos que pueden ser útiles para análisis de mercado.

**Open Data Portals:** Portal de datos abiertos que incluye información sobre propiedades.

## Datos de Redes Sociales

**Twitter:** Análisis de sentimiento y tendencias relacionadas con el mercado inmobiliario.

**Facebook Marketplace:** Listados de propiedades en venta y alquiler.

4. Con esta información de viviendas en venta y alquiler, **InverSmart Realty** pudo integrar su modelo de segmentación y predicción de precios con datos más granulares, con lo que consiguió:

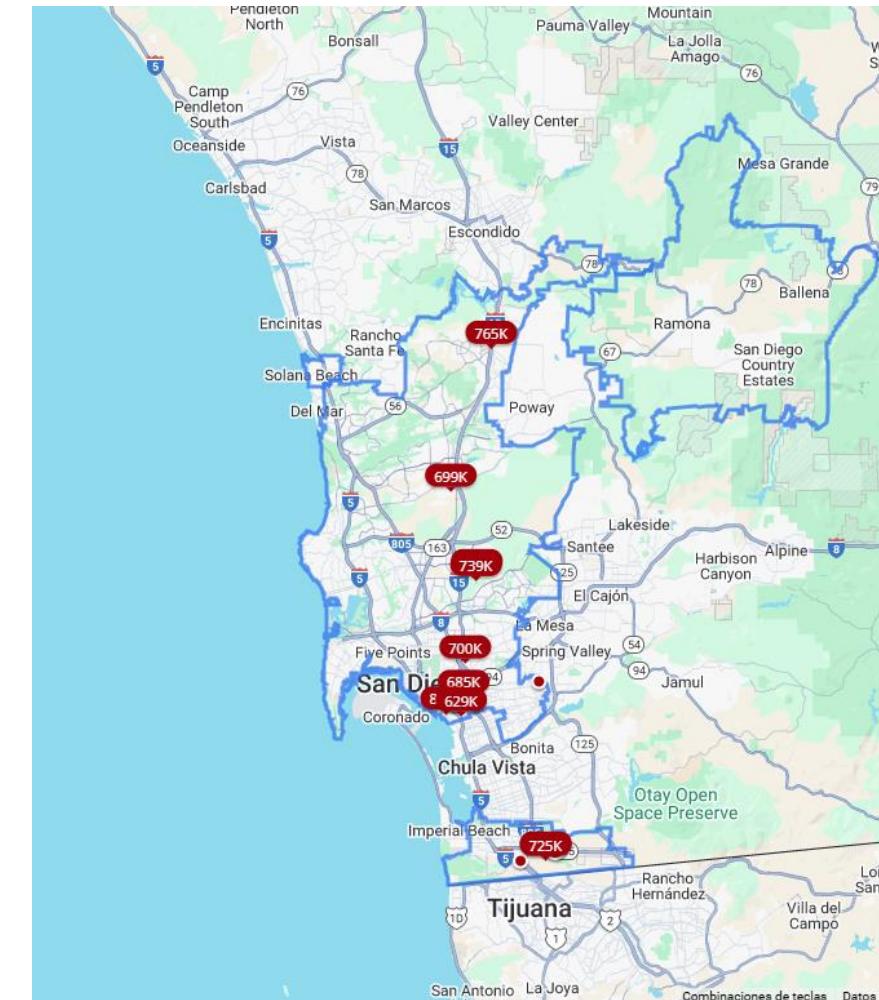
- Crear, aplicando modelos de Machine Learning, modelos muy precisos de valoración de viviendas, capaces de incorporar no solo las peculiaridades de cada zona, sino las características de cada vivienda particular.
- Construir, empleando una herramienta de visualización de datos como Power BI, Dashboards interactivos que muestran una visión 360° de la situación actual y tendencia futura del mercado inmobiliario en California, así como de las métricas y KPIs más relevantes de **InverSmart Realty**.



## Beneficios para InverSmart Realty de su enfoque data driven

Algunos de los muchos beneficios de esta estrategia basada en datos son:

- **Rápida identificación de propiedades que pueden ser oportunidades de inversión** y que son ofrecidas a sus clientes como oportunidad de compra.



- **Detección de áreas donde los precios están aumentando**, ideal para inversores a medio plazo que buscan alta rentabilidad.
- **Creación de mapas interactivos en tiempo real** (mapas de calor que muestran zonas de crecimiento, precios medios por vivienda, ratio de precio de venta vs precio de lista, áreas de alta demanda basado en búsquedas y visitas a propiedades, tiempo promedio en el mercado por área, etc.).
- **Creación y seguimiento de KPI's claves**:

**1) Análisis de Mercado:**

- **Índice de salud del mercado**: Indicador compuesto basado en tiempo en el mercado, diferencia entre precio de lista y venta, y volumen de transacciones
- **Análisis de sentimiento**: Resumen visual del sentimiento en reseñas de propiedades y áreas
- **Predicciones de precio**: precios actuales vs predicciones del modelo para los próximos 3-6 meses

**2) Rendimiento de Inversiones:**

- **ROI por tipo de propiedad y zona**
- **Simulador de inversiones**: Herramienta interactiva para proyectar retornos basados en diferentes escenarios

**3) Perfiles de compradores**: Segmentación de clientes basada en preferencias y comportamientos

**4) Eficiencia Operativa:**

- **Rendimiento de agentes**: Métricas clave por agente (ventas, tiempo promedio de cierre, satisfacción del cliente)

- **Embudo de ventas**: Visualización del proceso de venta desde el listado hasta el cierre



**• Sistemas de Alertas y notificaciones:**

- Propiedades infravaloradas y sobrevaloradas.
- Cambios significativos en el mercado
- Oportunidades de inversión basadas en el modelo predictivo

**• Personalización y Filtros**: Capacidad para personalizar vistas y crear dashboards específicos por rol (agente, gerente, inversor), filtros globales por fecha, tipo de propiedad, rango de precios, zona, etc.

- **Informes Automatizados**: Generación de informes semanales/mensuales con insights clave y predicciones

## **Resultados e Impacto en el negocio de la estrategia data driven de InverSmart Realty**

- Reducción de costos operativos al utilizar herramientas de código abierto y bajo coste.
- Aumentar la precisión y mejora de la tasa de éxito en recomendaciones de inversión en un 35% en comparación con sus métodos anteriores basados únicamente en la intuición y el conocimiento del mercado.
- Reducir el tiempo de búsqueda de propiedades adecuadas en un 60%, permitiéndoles ser más ágiles en un mercado competitivo.
- Aumento del 20% en el número de clientes en el primer año de implementación del sistema.
- Mejora del 25% en la satisfacción del cliente, medida a través de encuestas post-venta, gracias a servicios más personalizados y basados en datos.
- Capacidad para adaptarse rápidamente a cambios en el mercado.
- Expansión del negocio a nuevas áreas de California que previamente no consideraban rentables.
- Creación de una cultura de innovación y aprendizaje continuo dentro de la empresa, con personal más enfocado al negocio y no en tareas tediosas repetitivas.



## 03 Que lecciones se aprenden de este caso

- 1. El poder de la democratización de los datos:** El acceso a todo tipo de datos y la capacidad de extraer información de diversas fuentes en tiempo real han nivelado el campo de juego y son una oportunidad única para las pymes de competir en igualdad de condiciones con grandes corporaciones.
- 2. La importancia de la agilidad:** Las nuevas tecnologías permiten implementar rápidamente soluciones basadas en datos en tiempo real y hacen de la agilidad una ventaja competitiva significativa.
- 3. Tecnología accesible como catalizador:** Herramientas de código abierto como Python y plataformas de inteligencia de negocio asequibles como Power BI permiten implementar soluciones de análisis de datos y Machine Learning que antes solo estaban al alcance de grandes corporaciones.
- 4. La sinergia entre Machine Learning y el conocimiento del negocio:** El éxito se basa en la combinación de enfoque data driven con el conocimiento experto del negocio.
- 5. La importancia de la visualización de datos:** Los dashboards interactivos son cruciales para detectar y comunicar insights complejos de manera accesible tanto a clientes como a empleados.



- 6. Adaptabilidad e Innovación continua:** El enfoque data-driven permite adaptarse rápidamente a las cambiantes condiciones del mercado y fomenta una cultura de innovación continua.
- 7. Competitividad mejorada:** Adoptar el enfoque data-driven permite a una pyme no ya sólo competir sino destacar en un mercado altamente competitivo, demostrando que el tamaño de la empresa ya no es el factor determinante del éxito.
- 8. Ventaja competitiva sostenible:** Una estrategia basada en datos genera una ventaja competitiva sostenible que va más allá de los factores tradicionales como el tamaño o los recursos financieros.

## 04 Conclusiones: Una Era de Oportunidades para las PYMES

En la era digital, estamos presenciando una revolución sin precedentes. Las barreras que una vez separaban a las pequeñas empresas de los gigantes corporativos se están desmoronando, dando paso a un nuevo paisaje empresarial donde el ingenio y la agilidad triunfan sobre el tamaño y los recursos. Gracias a herramientas accesibles como Python y Power BI, las PYMES ahora tienen el poder de transformar datos en conocimiento accionable, compitiendo de igual a igual con empresas mucho más grandes.

Esta democratización del conocimiento no es solo una tendencia; es un cambio de paradigma que está redefiniendo las reglas del juego. Las PYMES que abrazan este enfoque data-driven no solo sobreviven, sino que prosperan, innovando a una velocidad que deja atrás a los gigantes corporativos. El futuro pertenece a los ágiles, a los audaces, a aquellos que ven en cada conjunto de datos una oportunidad para reinventar su industria.



La revolución está aquí, y está al alcance de cada emprendedor con visión y determinación. Es el momento de que las PYMES se levanten, se armen con datos y conocimientos, y reclamen su lugar en la vanguardia de la innovación. El mañana pertenece a aquellos que se atreven a desafiar el status quo, a los que ven en cada desafío una oportunidad para brillar. ¿Estás listo para ser parte de esta revolución?

**Enrique Aranaz Tudela**

| Data Driven | Consultor BI | Microsoft Certified Power BI Data Analyst | Data Scientist | Machine Learning Python



[www.linkedin.com/in/enrique-aranaz-tudela](https://www.linkedin.com/in/enrique-aranaz-tudela)



aranaz.enrique@gmail.com

aranaz.enrique@biparatupyme.com



# Anexo

## Anexo: Detalle desarrollo en python

Un modelo de machine learning sigue una metodología estructurada que, combinada con la versatilidad de Python, permite un desarrollo rápido y eficiente capaz de abordar todo tipo de problemas como segmentaciones, predicciones, etc.

Esta metodología típicamente incluye los siguientes pasos:

1. Análisis Exploratorio de Datos (EDA): Se examina y visualiza el conjunto de datos para entender sus características.
2. Preprocesamiento de Datos: Se limpian y transforman los datos según sea necesario.
3. Selección de Características: Se identifican las variables más relevantes para el problema.
4. Entrenamiento del Modelo: Se aplican algoritmos de machine learning a los datos preparados.
5. Evaluación del Modelo: Se mide el rendimiento del modelo utilizando métricas apropiadas.
6. Ajuste y Optimización: Se refinan los parámetros del modelo para mejorar su desempeño.

La flexibilidad de Python, junto con sus numerosas bibliotecas especializadas como Scikit-learn, Pandas, TensorFlow, etc. facilita la implementación eficiente de cada uno de estos pasos. Esto permite a los desarrolladores iterar rápidamente, probar diferentes enfoques y adaptar sus soluciones a una amplia gama de problemas de machine learning de manera ágil y efectiva.



# 0. Importación de librerías a utilizar y definición de funciones

## Indice

### 0. Importación de librerias a utilizar y definición de funciones

#### 1. Definición del problema

- 1.1 Introducción al problema que queremos abordar
- 1.2 Descripción del dataset y sus variables

#### 2. Preparación de los datos

- 2.1 Importación de librerias a utilizar
- 2.2 Carga del dataset y descripción de los datos
- 2.3 Data Cleaning

- 2.3.1 Identificación valores nulos, duplicados, valores no utiles para el modelo o con poca varianza
- 2.3.2 Análisis descriptivo univariante y bivariante
- 2.3.3 Identificación de valores atípicos y definición de estrategias con ellos
- 2.3.4 Data Transforms: Cambiando la escala o distribución de las variables

- 2.4 Feature engineering: Creación de nuevas variables

- 2.5 Feature Selection: Identificación de las variables más relevantes

- 2.5.1 Filter Methods: Correlación, Chi-cuadrado y Anova
- 2.5.2 Filter Methods empleando el método K-best
- 2.5.3 Embedded Methods: RandomForest

- 2.5.4 Conclusiones Feature Selection

- 2.6 Reducción de la dimensionalidad de los datos

#### 3. Entrenamiento y evaluación de modelos

- 3.1 Elección de algoritmos de machine learning adecuados para el problema.
- 3.2 Entrenamiento de los modelos y evaluación de su rendimiento.
- 3.3 Selección de los dos mejores modelos y optimización (ajuste hiperparámetros).

#### 4. Modelo final

```
[1]: # Librerias básicas:  
import numpy as np  
import pandas as pd  
%matplotlib inline  
import seaborn as sns  
import matplotlib.pyplot as plt  
import plotly.express as px  
import pylab  
import matplotlib.ticker as mtick  
import matplotlib.image as mpimg  
from scipy.stats import shapiro, normaltest, kstest, jarque_bera  
from scipy.stats import probplot  
from scipy.stats import stats  
from scipy.stats import mstats  
  
# Feature Selection, Preprocesado, Modelos, Métricas, GridSearch:  
from sklearn.pipeline import Pipeline  
from sklearn.preprocessing import LabelEncoder  
from sklearn.preprocessing import StandardScaler  
from sklearn.preprocessing import KBinsDiscretizer  
from sklearn.preprocessing import MinMaxScaler  
from imblearn.over_sampling import RandomOverSampler  
from imblearn.over_sampling import SMOTE  
from sklearn.model_selection import train_test_split  
from mlxtend.feature_selection import SequentialFeatureSelector as SFS  
from sklearn.feature_selection import SelectKBest, chi2, f_classif, mutual_info_classif  
from sklearn.linear_model import LogisticRegressionCV  
from sklearn.linear_model import LogisticRegression  
from sklearn.metrics import confusion_matrix  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.metrics import classification_report  
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score  
from sklearn.metrics import roc_curve, roc_auc_score  
from sklearn.ensemble import AdaBoostClassifier  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.ensemble import GradientBoostingClassifier  
from xgboost import XGBClassifier  
from sklearn.model_selection import GridSearchCV  
from sklearn.datasets import load_digits  
  
from keras.models import Sequential  
from keras.layers import Dense  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense  
from tensorflow.keras.metrics import Precision, Recall
```

# 1. Definición del problema

## 1.1 Introducción al problema que queremos abordar

```
[3]: mostrar_imagen('C:\MachineLearningInmobiliario\Introduccion.png')
```

### Introducción

La Oficina del Censo de los EE. UU. publica datos del censo de California que incluyen información a nivel distrito (grupos de bloques) de 10 tipos de métricas (precio medio bloque, ingreso medio, población, coordenadas de longitud y latitud, etc.).

Los distritos o grupos de bloques son las unidades geográficas más pequeñas para las cuales la Oficina del Censo de los EE. UU. publica datos de muestra (un grupo de bloques típicamente tiene una población de 600 a 3,000 personas). Hay 20,640 distritos en el conjunto de datos del proyecto.

**Objetivo del Problema:** El proyecto tiene como objetivo construir un modelo para predecir los valores medios de las casas en California utilizando el conjunto de datos proporcionado. Este modelo debe aprender de los datos y ser capaz de predecir el precio medio de la vivienda en cualquier distrito, dadas todas las demás métricas.

Para realizar el proyecto se emplearan técnicas de Machine Learning utilizando el lenguaje de ciencia de datos Python, lo que nos permitirá abordar el problema en todas sus fases, desde la preparación de los datos a la evaluación de distintos modelos y la selección final del más adecuado al objetivo buscado, creando así una metodología eficiente y automatizable que permita replicarse de forma ágil sobre nuestros datos en tiempo real, ajustándose el modelo a cualquier cambio en el mercado inmobiliario.

## 1.2 Descripción del dataset y sus variables

```
[7]: mostrar_imagen('C:\MachineLearningInmobiliario\Descripcion.png')
```

### Descripción del conjunto de datos:

El conjunto de datos tiene 20.640 registros correspondientes a el número total de registros de bloques del censo de California con 10 métricas por registro:

**Longitude y latitude:** Ubicación geográfica del bloque

**housing\_median\_age:** Edad media de las casas en el bloque

**total\_rooms:** Conteo del número total de habitaciones en todas las casas del bloque

**total\_bedrooms:** Conteo del número total de dormitorios en todas las casas del bloque

**Population:** Conteo del número total de población en el bloque

**Households:** Conteo del número total de hogares en el bloque

**median\_income:** Mediana del ingreso total de los hogares en todas las casas del bloque

**ocean\_proximity:** Tipo de paisaje del bloque ('Near Bay', '<1h Ocean', 'Inland', 'Near Ocean', 'Island')

**median\_house\_value:** Mediana de los precios de las casas en el bloque

# 2. Preparación de los datos

## 2.2 Carga del dataset y descripción de los datos

|   | longitude   | latitude  | housing_median_age | total_rooms | total_bedrooms | population | households | median_income | ocean_proximity | median_house_value |
|---|-------------|-----------|--------------------|-------------|----------------|------------|------------|---------------|-----------------|--------------------|
| 0 | -122.230000 | 37.880000 | 41                 | 880         | 129            | 322        | 126        | \$8,33        | NEAR BAY        | \$452.600          |
| 1 | -122.220000 | 37.860000 | 21                 | 7.099       | 1.106          | 2.401      | 1.138      | \$8,30        | NEAR BAY        | \$358.500          |
| 2 | -122.240000 | 37.850000 | 52                 | 1.467       | 190            | 496        | 177        | \$7,26        | NEAR BAY        | \$352.100          |
| 3 | -122.250000 | 37.850000 | 52                 | 1.274       | 235            | 558        | 219        | \$5,64        | NEAR BAY        | \$341.300          |
| 4 | -122.250000 | 37.850000 | 52                 | 1.627       | 280            | 565        | 259        | \$3,85        | NEAR BAY        | \$342.200          |
| 5 | -122.250000 | 37.850000 | 52                 | 919         | 213            | 413        | 193        | \$4,04        | NEAR BAY        | \$269.700          |
| 6 | -122.250000 | 37.840000 | 52                 | 2.535       | 489            | 1.094      | 514        | \$3,66        | NEAR BAY        | \$299.200          |
| 7 | -122.250000 | 37.840000 | 52                 | 3.104       | 687            | 1.157      | 647        | \$3,12        | NEAR BAY        | \$241.400          |
| 8 | -122.260000 | 37.840000 | 42                 | 2.555       | 665            | 1.206      | 595        | \$2,08        | NEAR BAY        | \$226.700          |

## 2.3 Data Cleaning

### 2.3.1 Identificación valores nulos, duplicados, valores no utiles para el modelo o con poca varianza

El tamaño del dataset es : (20640, 10)

---- Duplicados ----

Número de duplicados: 0

|                    | Valores únicos | Valores nulos | % Valores nulos | Type    |
|--------------------|----------------|---------------|-----------------|---------|
| longitude          | 844            | 0             | 0.000000        | float64 |
| latitude           | 862            | 0             | 0.000000        | float64 |
| housing_median_age | 52             | 0             | 0.000000        | int64   |
| total_rooms        | 5926           | 0             | 0.000000        | int64   |
| total_bedrooms     | 1923           | 207           | 1.002907        | float64 |
| population         | 3888           | 0             | 0.000000        | int64   |
| households         | 1815           | 0             | 0.000000        | int64   |
| median_income      | 12928          | 0             | 0.000000        | float64 |
| ocean_proximity    | 5              | 0             | 0.000000        | object  |
| median_house_value | 3842           | 0             | 0.000000        | int64   |

Utilizamos la librería Geopy para, a partir de la latitud y longitud obtener la ciudad y el Estado

```
import pandas as pd
from geopy.geocoders import Nominatim
from geopy.exc import GeocoderTimedOut
import time

def get_location_info(lat, lon):
    geolocator = Nominatim(user_agent="my_agent")
    tries = 0
    while tries < 3:
        try:
            location = geolocator.reverse(f"{lat}, {lon}")
            address = location.raw['address']
            city = address.get('city', address.get('town', address.get('village', '')))
            state = address.get('state', '')
            country = address.get('country', '')
            return city, state, country
        except GeocoderTimedOut:
            tries += 1
            time.sleep(1)
    return "", "", ""

def add_location_to_df(df, start, end):
    # Asegurarse de que start y end estén dentro de los límites del DataFrame
    start = max(0, start)
    end = min(len(df), end)

    # Crear un nuevo DataFrame con las filas seleccionadas
    df_slice = df.iloc[start:end].copy()

    # Aplicar la función get_location_info solo a las filas seleccionadas
    df_slice[['ciudad', 'estado', 'pais']] = df_slice.apply(
        lambda row: pd.Series(get_location_info(row['latitude'], row['longitude'])),
        axis=1
    )

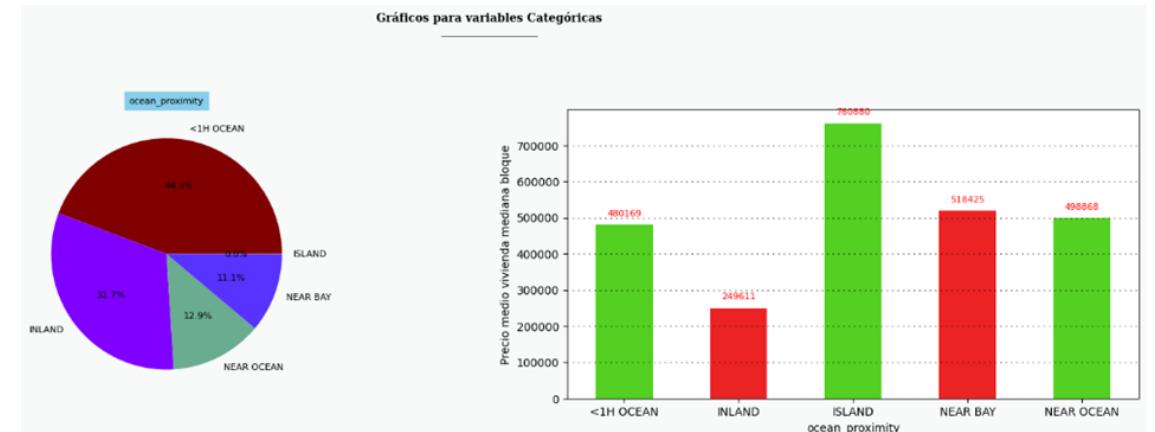
    # Actualizar las filas correspondientes en el DataFrame original
    df.iloc[start:end, df.columns.get_indexer(['ciudad', 'estado', 'pais'])] = df_slice[['ciudad', 'estado', 'pais']]

    return df

df = add_location_to_df(df, 20001, 20640)
```

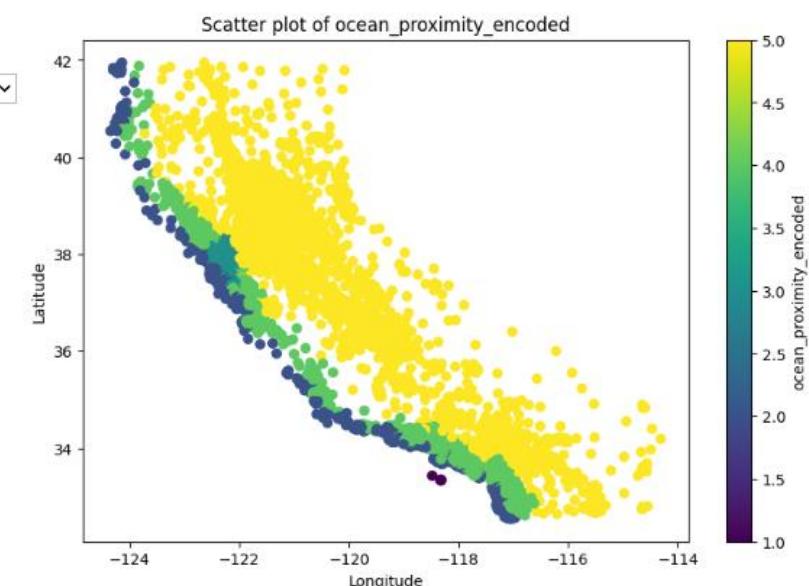
|       | longitude | latitude | ciudad      |
|-------|-----------|----------|-------------|
| 20410 | -121.53   | 39.06    | Olivehurst  |
| 20412 | -121.56   | 39.01    | Plumas Lake |
| 20415 | -121.44   | 39.00    | Wheatland   |
| 20418 | -121.52   | 39.12    | Olivehurst  |
| 20429 | -121.21   | 39.49    | Challenge   |

## 2.3.2 Análisis descriptivo univariante y bivariante



Seleccionar Variable

Variable: ocean\_proximity\_encoded  
Actualizar Gráfico



Utilizamos la librería Folium de Python para crear un mapa web interactivo con el posicionamiento de los 20.640 bloques de viviendas

```
# Crear un mapa centrado en California con la capa de satélite de Esri como predeterminada  
m = folium.Map(location=[36.7783, -119.4179], zoom_start=6, control_scale=True, tiles=None)
```

```
# Añadir la capa de satélite de Esri como capa base predeterminada
```

```
folium.TileLayer(  
    tiles='https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}',  
    attr='Esri',  
    name='Satellite',  
    overlay=False,  
    control=True  
) .add_to(m)
```

```
# Añadir OpenStreetMap como capa base alternativa
```

```
folium.TileLayer('openstreetmap', name='OpenStreetMap').add_to(m)
```

```
# Añadir puntos al mapa usando tu DataFrame df
```

```
for idx, row in df.iterrows():  
    folium.CircleMarker(  
        location=[row['latitude'], row['longitude']],  
        radius=2,  
        popup=f'Ubicación: {row["latitude"]}, {row["longitude"]}',  
        color='red',  
        fill=True,  
        fill_color='red',  
        fill_opacity=0.7,  
        weight=2  
) .add_to(m)
```

```
# Añadir título
```

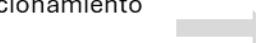
```
title_html = ''''  
    <h3 align="center" style="font-size:20px"><b>Ubicación geográfica de los 20.640 bloques de California</b></h3>  
'''
```

```
m.get_root().html.add_child(folium.Element(title_html))  
  
# Añadir control de capas
```

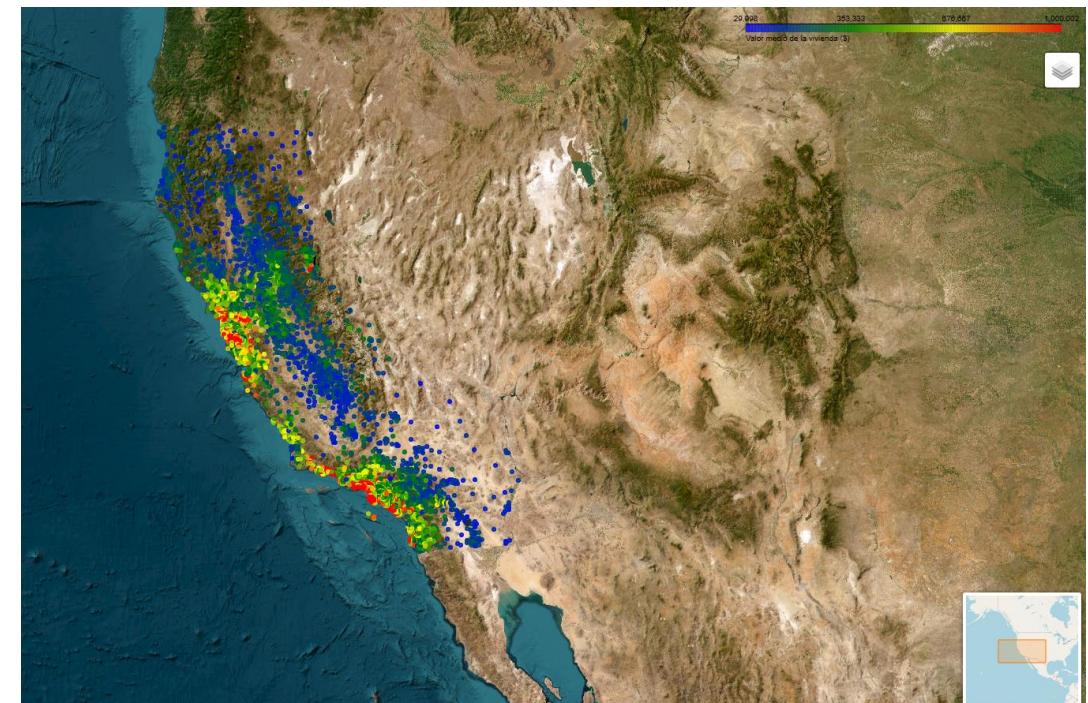
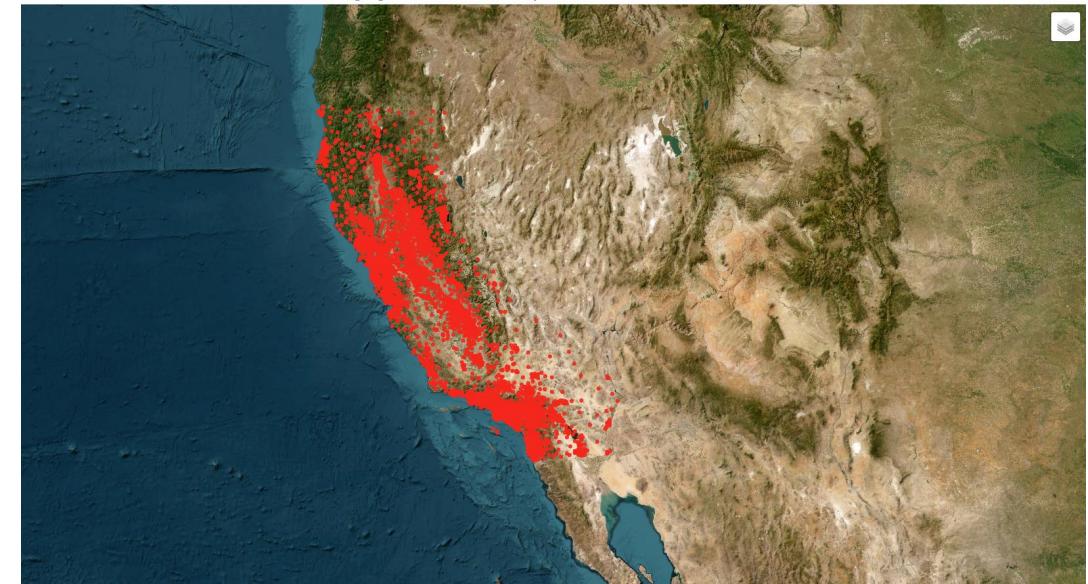
```
folium.LayerControl().add_to(m)
```

```
# Guardar el mapa en un archivo HTML
```

```
m.save('c:/MachineLearningInmobiliario/mapa_california.html')
```



Ubicación geográfica de los 20.640 bloques de California



Mapa web con el posicionamiento de los 20.640 bloques de viviendas tramificado en función valor medio vivienda

## Análisis variables numéricas

```
# Definir una paleta de colores personalizada
custom_palette = ["#800000", "#8000ff", "#6aac90", "#5833ff", "#da8829"]
bar_palette = {0: 'green', 1: 'red'}
colors = ["#53D021", "#EB2325"]

# Configurar la figura con constrained_layout
fig, axes = plt.subplots(len(numerical_cont_cols), 3, figsize=(36, len(numerical_cont_cols) * 6),
                        constrained_layout=True)
fig.patch.set_facecolor("#f8f9f9")

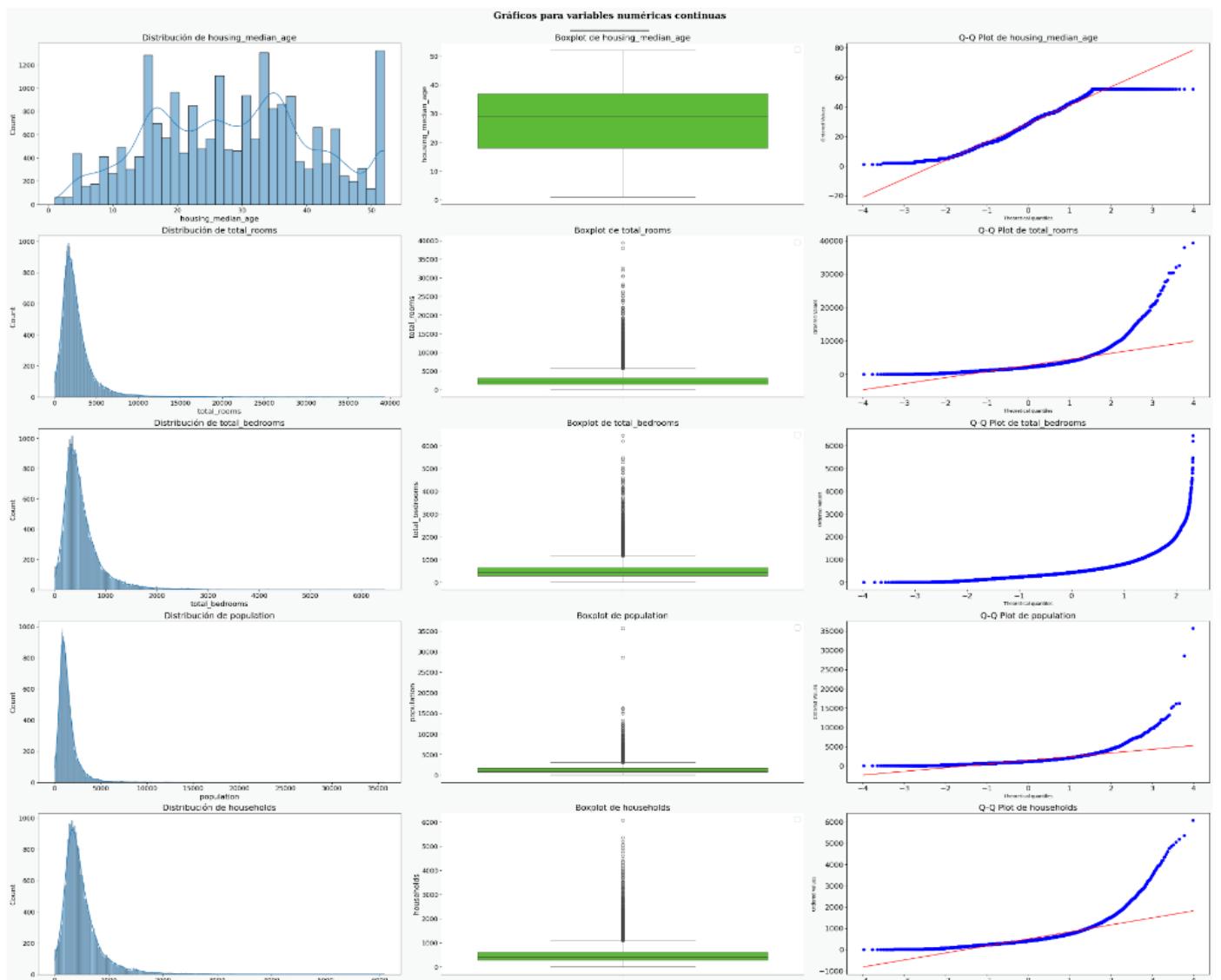
# Título del gráfico
fig.suptitle('Gráficos para variables numéricas continuas\n',
             fontsize=20, fontweight='bold', fontfamily='serif', color="#000000")

# Crear gráficos de histograma con KDE, de caja y Q-Q plot para cada columna numérica continua
for i, col in enumerate(numerical_cont_cols):
    # Histograma con KDE
    hist_ax = axes[i, 0]
    sns.histplot(ax=hist_ax, x=col, data=df, kde=True, palette=bar_palette)
    hist_ax.set_xlabel(col, fontsize=16)
    hist_ax.set_ylabel('Count', fontsize=16)
    hist_ax.set_title(f'Distribución de {col}', fontsize=18)
    hist_ax.tick_params(axis='both', which='major', labelsize=14)

    # Gráfico de caja
    box_ax = axes[i, 1]
    sns.boxplot(ax=box_ax, y=col, data=df, palette=colors)
    box_ax.set_title(f'Boxplot de {col}', fontsize=18)
    box_ax.set_ylabel(col, fontsize=16)
    box_ax.legend([], fontsize=16)
    box_ax.tick_params(axis='both', which='major', labelsize=14)
    for s in ["top", "right", "left"]:
        box_ax.spines[s].set_visible(False)

    # Gráfico Q-Q
    qq_ax = axes[i, 2]
    stats.probplot(df[col], dist="norm", plot=qq_ax)
    qq_ax.set_title(f'Q-Q Plot de {col}', fontsize=18)
    qq_ax.tick_params(axis='both', which='major', labelsize=16)

plt.show()
```



Dado que varias de nuestras variables numéricas no siguen una distribución normal, vamos a emplear la transformación de Box-cox para estabilizar la varianza y hacer que los datos se asemejen más a una distribución normal. Esto es especialmente útil cuando se trabaja con modelos de regresión, ya que muchos métodos estadísticos asumen que los datos siguen una distribución normal.

Estabilizar la varianza: Ayuda a reducir la heterocedasticidad, es decir, cuando la variabilidad de los datos no es constante a lo largo de los valores de las variables.

Normalizar los datos: Transforma los datos para que se asemejen más a una distribución normal, lo cual es un requisito común en muchos análisis estadísticos.

Mejorar la linealidad: Facilita la relación lineal entre las variables dependientes e independientes en un modelo de regresión.

Optimizar modelos predictivos: Al mejorar las propiedades estadísticas de los datos, se pueden obtener modelos más precisos y fiables.

¿Cómo se aplica?

La transformación Box-Cox se define mediante la fórmula:

$$y(\lambda) = \begin{cases} \lambda y - 1 & \text{si } \lambda \neq 0 \\ \ln(y) & \text{si } \lambda = 0 \end{cases}$$

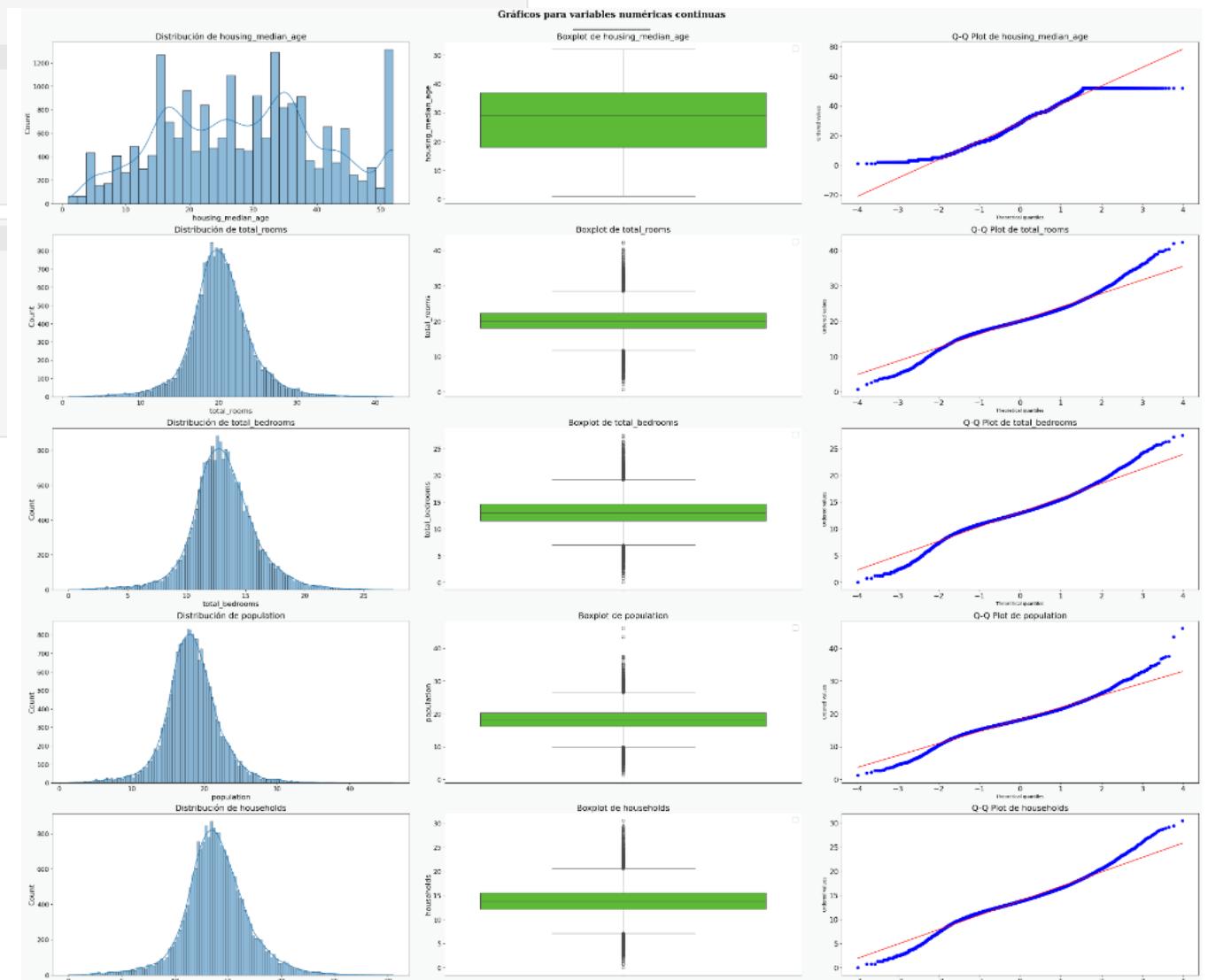
Donde:

( $y$ ) es el valor original de los datos.

( $\lambda$ ) es el parámetro de transformación que se estima a partir de los datos.

```
[1]: from scipy.stats import boxcox
```

```
# Aplicar la transformación Box-Cox
df['total_rooms'], best_lambda = boxcox(df['total_rooms'])
df['total_bedrooms'], best_lambda = boxcox(df['total_bedrooms'])
df['population'], best_lambda = boxcox(df['population'])
df['households'], best_lambda = boxcox(df['households'])
df['median_income'], best_lambda = boxcox(df['median_income'])
```



Vemos que las variables se han normalizado , al mismo tiempo que se han reducido el número de outliers

```

# Definir una paleta de colores personalizada
palette = {0: 'green', 1: 'red'}

# Configurar la figura
fig = plt.figure(figsize=(20, 20))
background_color = "#ffe6e6"
fig.patch.set_facecolor(background_color)

# Crear el pairplot
pairplot = sns.pairplot(df[numerical_cont_cols], palette=palette)

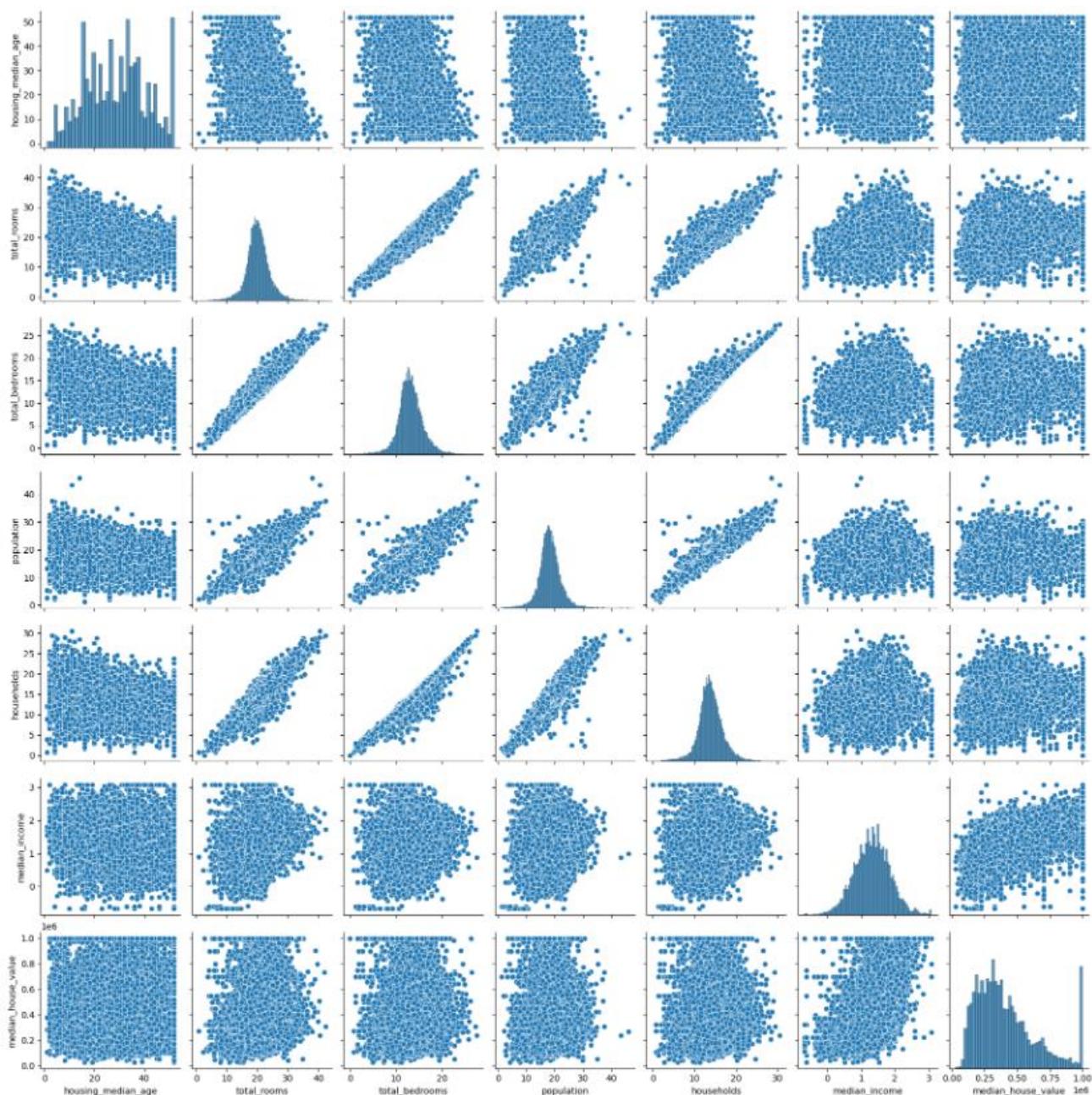
# Configurar el fondo de los ejes del pairplot
for ax in pairplot.axes.flatten():
    ax.set_facecolor('white') # Fondo blanco para cada gráfico

# Fondo general
fig.patch.set_facecolor(background_color)

# Mostrar el gráfico
plt.show()

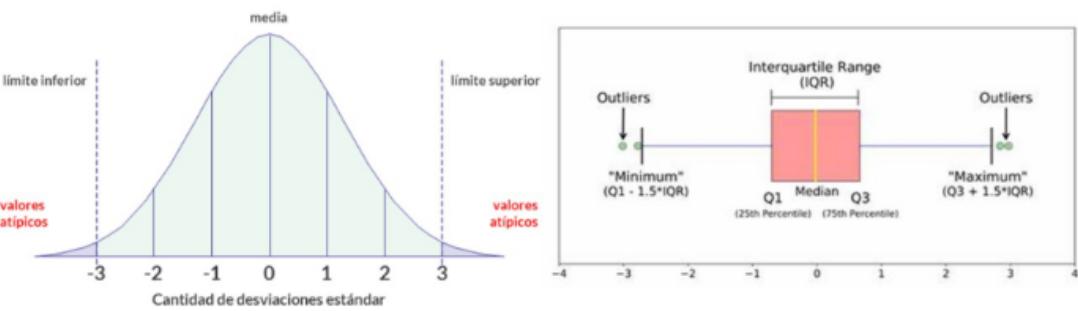
```

Pairplots para identificar relaciones entre las variables, ver histograma y detectar atípicos.



### 2.3.3 Identificación de valores atípicos y definición de estrategias con ellos

[54]: mostrar\_imagen('c:/Python/CorreccionAtipicos.jpg')



Vamos a analizar la presencia de atípicos en las variables numéricas continuas , para lo que emplearemos dos métodos: IQR y +/- 3 desv. típicas

[112]: # Análisis de outliers

```
print("---- Outliers en variables numéricas continuas ----")
print("")
print(numerical_cont_cols)
print("")

df_numerical_cont_cols = df[numerical_cont_cols]

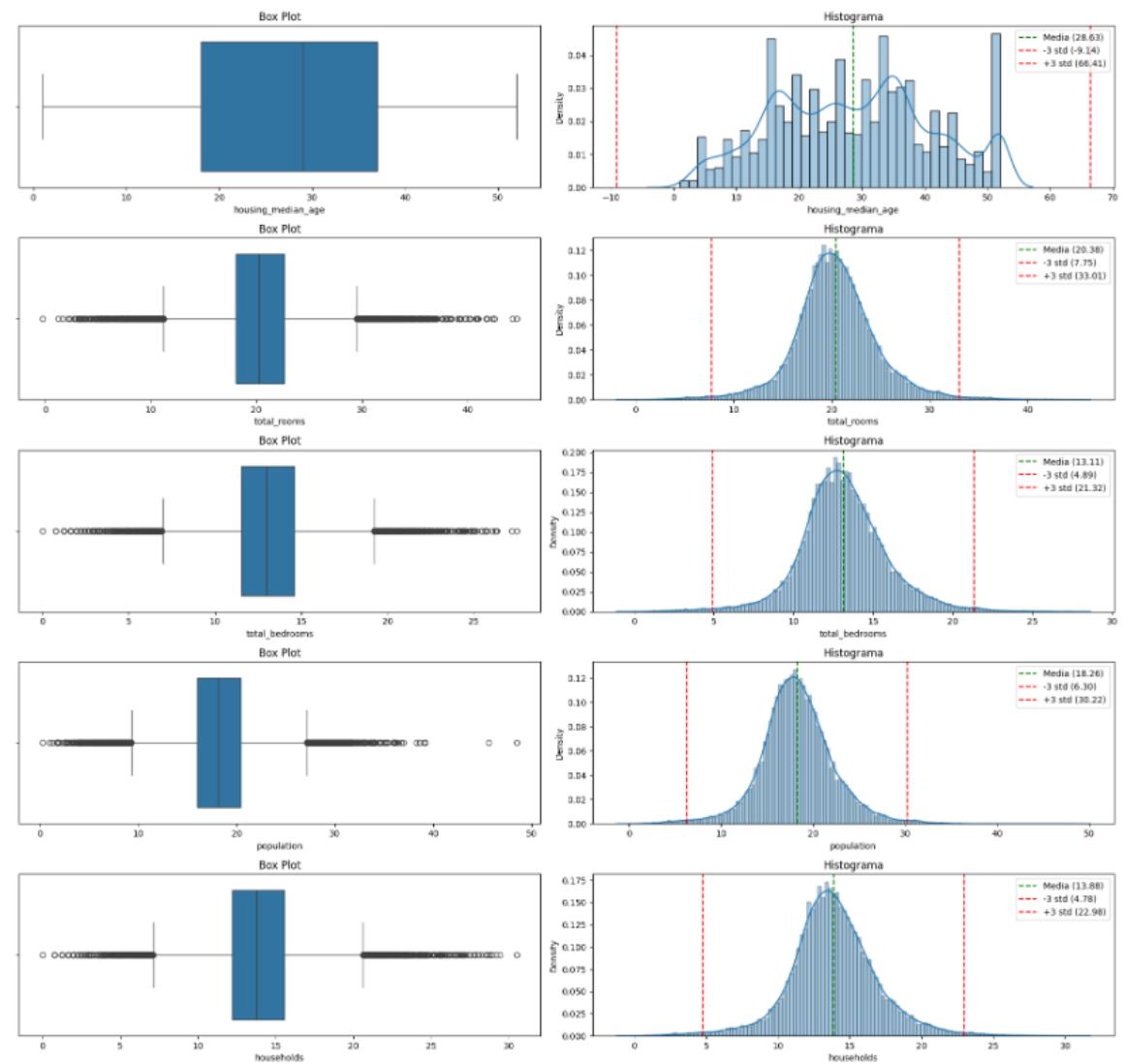
fig, axes = plt.subplots(len(numerical_cont_cols), 2, figsize=(18, 24))

for i, col in enumerate(numerical_cont_cols):
    sns.boxplot(ax=axes[i, 0], x=df_numerical_cont_cols[col])
    axes[i, 0].set_title(f'Box Plot', fontsize=12)

    sns.histplot(ax=axes[i, 1], x=df_numerical_cont_cols[col], kde=True, stat="density", kde_kws={'cut': 3}, alpha=.4)
    axes[i, 1].set_title(f'Histograma', fontsize=12)

    media = df_numerical_cont_cols[col].mean()
    std = df_numerical_cont_cols[col].std()
    axes[i, 1].axvline(x=media, color='green', linestyle='--', label=f'Media ({media:.2f})')
    axes[i, 1].axvline(x=media - 3 * std, color='red', linestyle='--', label=f'-3 std ({media-3*std:.2f})')
    axes[i, 1].axvline(x=media + 3 * std, color='red', linestyle='--', label=f'+3 std ({media+3*std:.2f})')
    axes[i, 1].legend()

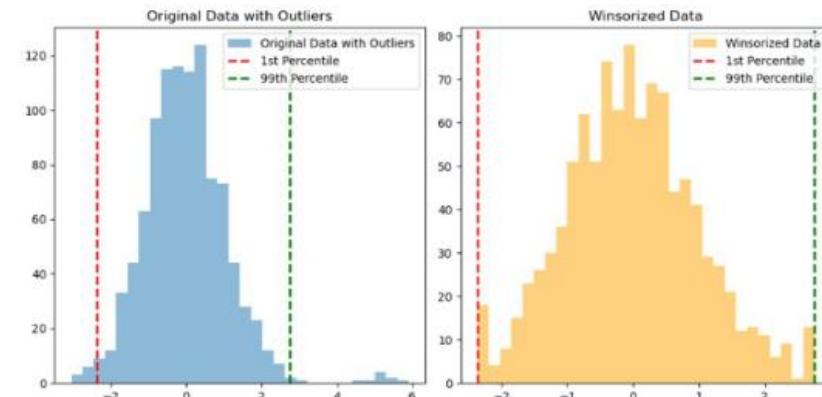
plt.tight_layout()
plt.show()
```



## Gestionando outliers con winsorización

La winsorización es una técnica utilizada para minimizar la influencia de valores atípicos en los datos y de esta forma mejorar la eficiencia de las técnicas de inferencia estadística y el rendimiento de los modelos de machine learning.

Para ello se especifica el % de datos que queremos winsorizar. En nuestro caso aplicaremos una winsorización de un 1% superior y 1% inferior (98% de winsorización) para posteriormente reemplazar los valores extremos superiores e inferiores por los valores máximos y mínimos en el umbral.



```
# Gestiónando outliers con winsorización

# Seleccionamos aquellas columnas con outliers
outliers_count_std_sel = outliers_count_std.loc[:, outliers_count_std.sum() > 0]
outliers_count_std_sel
outliers_corregir = outliers_count_std_sel.columns.tolist()

print("Columnas con outliers corregidos:", outliers_corregir)

# Aplicamos La winsorización para tratar Los outliers

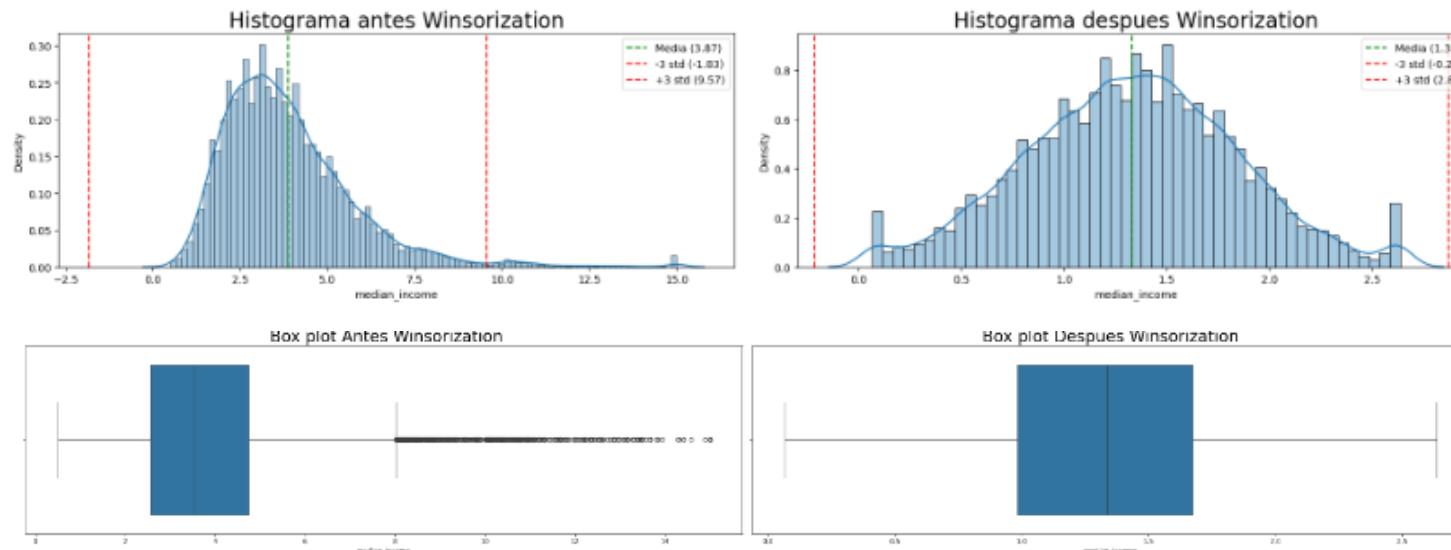
for col in outliers_corregir:
    df[col] = mstats.winsorize(df[col], limits=[0.01, 0.01])

if len(outliers_corregir) == 1:
    fig, axes = plt.subplots(1, 2, figsize=(20, 12))
    axes = [axes] # Convertir a lista para mantener La consistencia
else:
    fig, axes = plt.subplots(len(outliers_corregir), 2, figsize=(20, 12))

for i, col in enumerate(outliers_corregir):
    # Boxplot antes de La winsorización
    sns.boxplot(ax=axes[i][0], x=df_numerical_cont_cols[col])
    axes[i][0].set_title(f'Box plot Antes Winsorization', fontsize=12)

    # Boxplot después de La winsorización
    sns.boxplot(ax=axes[i][1], x=df[col])
    axes[i][1].set_title(f'Box plot Despues Winsorization', fontsize=12)

plt.tight_layout()
plt.show()
```



## 2.4 Feature engineering: Creación de nuevas variables

Vamos a realizar una segmentación intentando identificar zonas con comportamientos distintos respecto a las características de los bloques

[96]: # 2. Segmentación

```
# a) Clustering con precio
from sklearn.cluster import KMeans
# Primero, escalamos las características para que estén en escalas comparables
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df[['latitude', 'longitude', 'median_house_value']])

# Ahora realizamos el clustering
kmeans = KMeans(n_clusters=8, random_state=42)
df['location_price_cluster'] = kmeans.fit_predict(scaled_features)
```

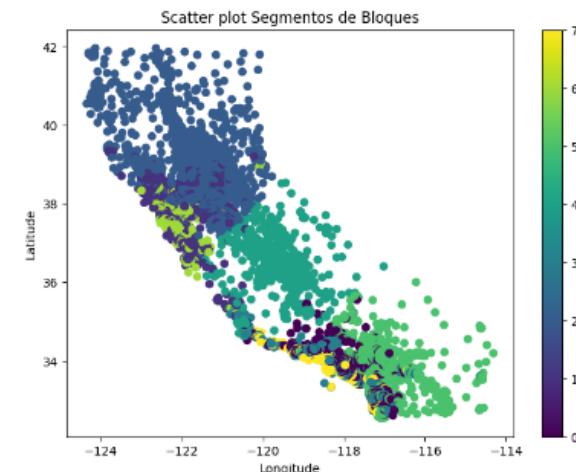
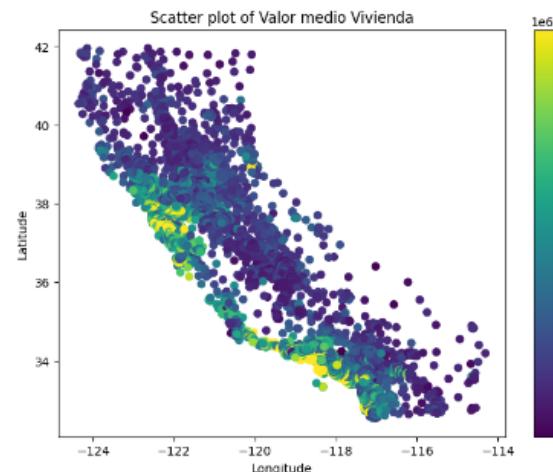
[114]:

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(18, 6))

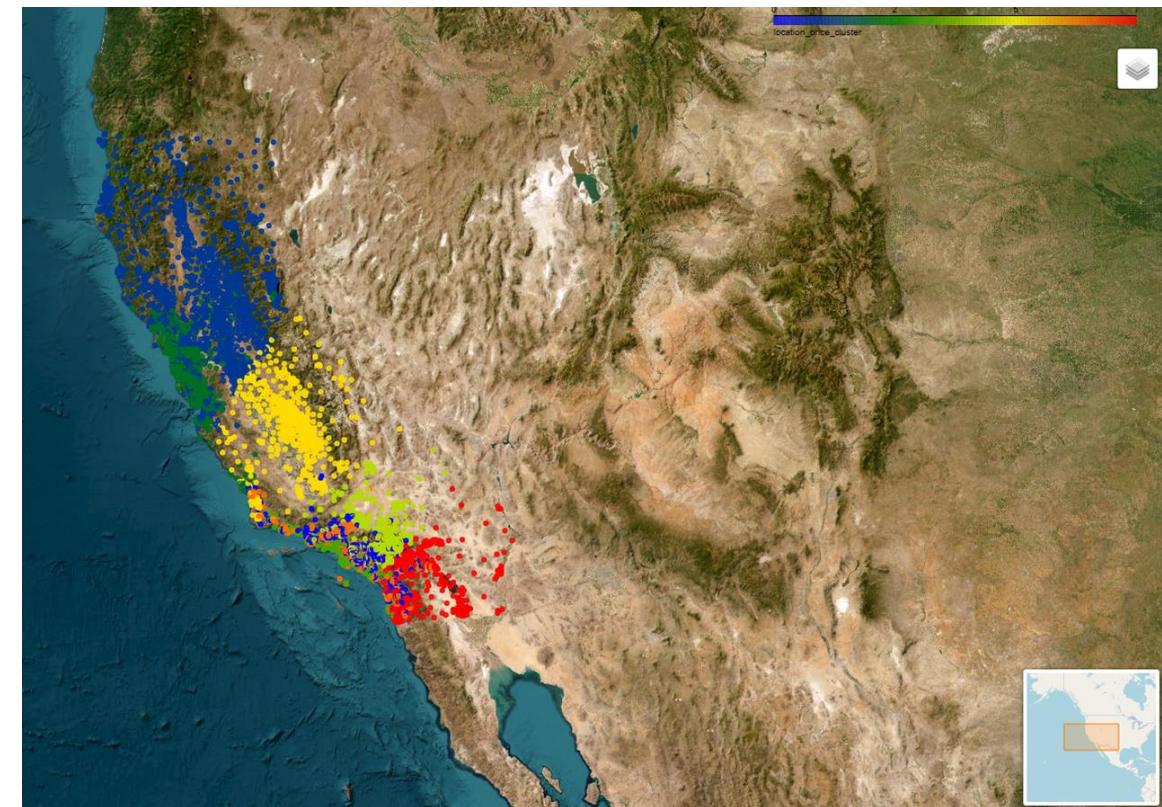
scatter1 = ax1.scatter(df.longitude, df.latitude, c=df['median_house_value'], cmap='viridis')
ax1.set_xlabel("Longitude")
ax1.set_ylabel("Latitude")
fig.colorbar(scatter1, ax=ax1, label="Valor medio vivienda")
ax1.set_title("Scatter plot de Valor medio Vivienda")

scatter2 = ax2.scatter(df.longitude, df.latitude, c=df['location_price_cluster'], cmap='viridis')
ax2.set_xlabel("Longitude")
ax2.set_ylabel("Latitude")
fig.colorbar(scatter2, ax=ax2, label="Cluster")
ax2.set_title("Scatter plot Segmentos de Bloques")

plt.show()
```



## Clústeres obtenidos tras aplicar la técnica de k-means\*



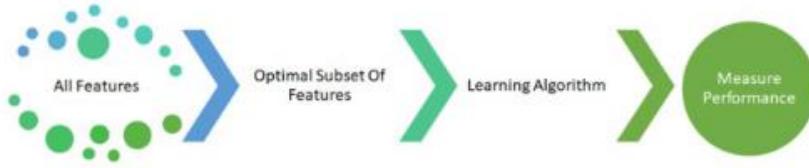
\* la técnica de K-means es un algoritmo de aprendizaje no supervisado utilizado para agrupar datos en diferentes clusters o segmentos. Es especialmente útil en la segmentación de mercados, donde se busca identificar grupos de clientes o productos con comportamientos o características similares.

## 2.4 Feature Selection: Identificación de las variables más relevantes

```
[67]: mostrar_imagen('c:/Phyton/FeatureSelection.jpg')
```

## Feature Selection

La selección de características (Feature Selection) es un proceso crucial en el desarrollo de modelos predictivos en machine learning. Su objetivo principal es reducir el número de variables de entrada, eliminando aquellas que son irrelevantes o redundantes, para mejorar la eficiencia y, en muchos casos, la precisión del modelo.

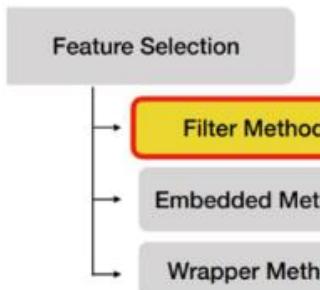


## **Beneficios de la Selección de Características**

- Reducción de la Dimensionalidad: Menos características significan menos datos que procesar, lo que puede reducir el tiempo de entrenamiento y la complejidad del modelo.
  - Mejora del Rendimiento: Al eliminar características irrelevantes o redundantes, el modelo puede generalizar mejor y evitar el sobreajuste.
  - Interpretabilidad: Modelos con menos características son más fáciles de interpretar y entender.

### 2.4.1 Filter Methods

```
[69]: mostrar_imagen('c:/Phyton/FilterMethods.jpg')
```



Los métodos de selección de características por filtro (filter methods) son una técnica utilizada en machine learning para seleccionar las características más relevantes de un conjunto de datos antes de entrenar un modelo. Estos métodos se basan en criterios estadísticos para evaluar la importancia de cada característica de forma independiente del modelo que se va a utilizar. En nuestro caso aplicaremos los siguientes métodos.

## 1. Correlación

Este método mide la relación lineal entre dos variables. Las características que tienen una alta correlación con la variable objetivo se consideran más importantes. Para datos continuos, se puede utilizar la correlación de Pearson, mientras que para datos categóricos se puede utilizar la correlación de Spearman.

## 2. Chi-cuadrado (Chi - squared)

Este método se utiliza principalmente para datos categóricos. Evalúa la independencia entre cada característica y la variable objetivo. Las características que tienen una mayor dependencia con la variable objetivo se consideran más importantes. Un valor alto de chi-cuadrado indica una fuerte asociación entre la característica y la variable objetivo.

### 3. ANOVA (Analysis of Variance)

ANOVA se utiliza para comparar las medias de diferentes grupos y determinar si hay diferencias significativas entre ellas. En el contexto de la selección de características, se utiliza para evaluar la relación entre una característica continua y una variable objetivo categórica.

```

# Highly correlated columns
corr = df.select_dtypes(include=['int', 'float']).corr()

# Filtrar las correlaciones que cumplen la condición
filtered_corr = corr[(corr > 0.7) | (corr < -0.7)]

# Construir el DataFrame con los pares de variables y sus correlaciones
pairs = []
for i in range(len(filtered_corr.columns)):
    for j in range(i):
        if abs(filtered_corr.iloc[i, j]) > 0.7:
            pairs.append((filtered_corr.index[i], filtered_corr.columns[j], filtered_corr.iloc[i, j]))

# Crear el DataFrame
df_pairs = pd.DataFrame(pairs, columns=['Var1', 'Var2', 'Correlacion'])

# Mostrar el DataFrame
print(df_pairs)

plt.figure(figsize = (17,8))
sns.heatmap(corr[(corr>0.7)|(corr<-0.7)], annot = True)


```

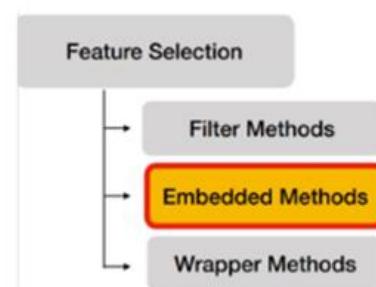
|   | Var1           | Var2           | Correlacion |
|---|----------------|----------------|-------------|
| 0 | latitude       | longitude      | -0.924616   |
| 1 | total_bedrooms | total_rooms    | 0.936674    |
| 2 | population     | total_rooms    | 0.853520    |
| 3 | population     | total_bedrooms | 0.891526    |
| 4 | households     | total_rooms    | 0.924902    |
| 5 | households     | total_bedrooms | 0.976697    |
| 6 | households     | population     | 0.923659    |

[158]: <Axes: >



## 2.4.3 Embedded Methods: RandomForest

```
[89]: mostrar_imagen('c:/Python/Embedded Methods.jpg')
```



Los Embedded methods (métodos embebidos) son técnicas de selección de características que integran el proceso de selección dentro de la construcción del modelo de aprendizaje automático. A diferencia de los métodos de filtro y envoltura, los embedded methods realizan la selección de características durante el entrenamiento del modelo, lo que las hace más eficientes y precisas.

### Ventajas de las Embedded Methods:

- Interacción de características: Consideran la interacción entre las características, similar a los métodos de envoltura.
- Eficiencia: Son más rápidas que los métodos de envoltura porque solo entrena un modelo.
- Precisión: Tienen a ser más precisas que los métodos de filtro.
- Menor riesgo de sobreajuste: Son menos propensas al sobreajuste en comparación con otros métodos.

### Proceso de las Embedded Methods:

1. Entrenamiento del modelo: Se entrena un modelo de aprendizaje automático.
2. Derivación de la importancia de las características: Se mide la importancia de cada característica en la predicción.
3. Selección de características: Se eliminan las características no importantes basándose en la importancia derivada.

### Ejemplos de Embedded Methods

- Regularización (Lasso): Añade una penalización a los parámetros de modelos de regresión lineal y logística para reducir su libertad y seleccionar características importantes.
- Métodos basados en árboles: Utilizan la ganancia de información para determinar la importancia de las características.

Estas técnicas son especialmente útiles cuando se trabaja con grandes conjuntos de datos y se desea construir modelos más interpretables y robustos.

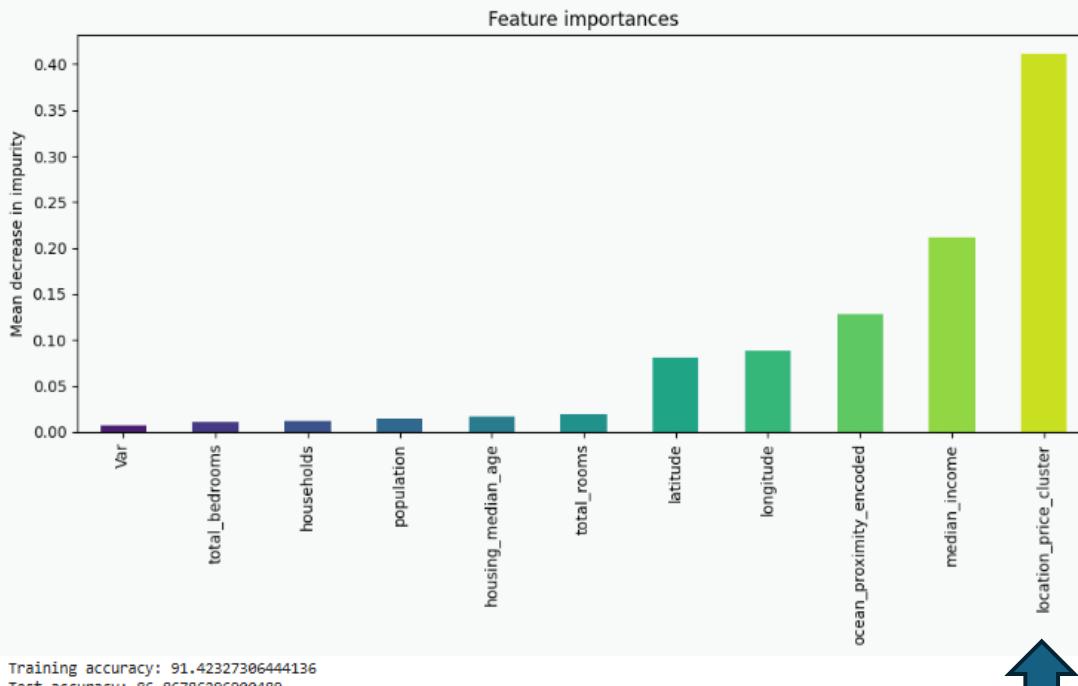
```
[ ]: En nuestro caso entrenaremos un modelo RandomForest para seleccionar las mejores características de nuestro df
```

```
[194]: #Añado una variable aleatoria que me va a servir para comparar con otras y ver un posible punto de corte para La significaciun  
np.random.seed(123)  
random_values = np.random.rand(df.shape[0])  
  
# Remodelar random_values para que coincida con la forma del DataFrame  
x1 = np.abs(np.random.randn(df.shape[0])).reshape(-1, 1)  
df['Var']=x1
```

```
# función para entrenar el modelo  
  
def entrenar_random_forest(X, umbral):  
  
    #Aplicación RandomOverSampler para corregir desequilibrio  
    # Dividir los datos en conjuntos de entrenamiento y prueba  
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)  
  
    # Aplicar RandomOverSampler solo al conjunto de entrenamiento  
    rand_over_samp = RandomOverSampler(random_state=42)  
    X_train, y_train = rand_over_samp.fit_resample(X_train, y_train)  
    X_train.shape,X_test.shape,y_train.shape,y_test.shape  
  
    # Entrenar el modelo de Random Forest  
    forest = RandomForestClassifier(max_depth=10, min_samples_split=20, n_estimators=500, random_state=1)  
    forest.fit(X_train, y_train)  
  
    # Obtener las importancias de las características  
    importances = forest.feature_importances_  
  
    # Crear una serie de pandas con las importancias  
    forest_importances = pd.Series(importances, index=X.columns)  
  
    # Seleccionar las características con importancia mayor al umbral  
    selected_features = forest_importances[forest_importances > umbral].index.tolist()  
  
    # Graficar las importancias de las características  
    fig, ax = plt.subplots(figsize=(10, 6))  
    background_color = "#f8f9f9"  
    fig.patch.set_facecolor(background_color)  
    ax.set_facecolor(background_color)  
    forest_importances.sort_values().plot.bar(ax=ax, color=sns.color_palette('viridis', len(forest_importances)))  
    ax.set_title("Feature importances")  
    ax.set_xlabel("Mean decrease in impurity")  
    fig.tight_layout()  
    plt.show()  
  
    # Mostrar las precisiones de entrenamiento y prueba  
  
    print('Training accuracy:', np.mean(forest.predict(X_train) == y_train) * 100)  
    print('Test accuracy:', np.mean(forest.predict(X_test) == y_test) * 100)  
    # Ordenar de mayor a menor  
    sorted_importances = sorted(forest_importances.items(), key=lambda item: item[1], reverse=True)  
  
    # Crear un DataFrame a partir de la lista de tuplas ordenadas  
    df_importances_forest = pd.DataFrame(sorted_importances, columns=['Feature', 'Importance'])  
    print('-----')  
    print(df_importances_forest)  
    print('-----')  
    print(f'Las variables que han sido seleccionadas con importancia mayor a {umbral} son:\n')  
    df_importances_umbral = df_importances_forest[df_importances_forest['Importance'] >= umbral]  
    print(df_importances_umbral)  
  
    return selected_features, forest
```

Entremo el modelo con todas las variables Establecer un umbral de corte sobre 0,05 que puede ser ajustado en función de la importancia de la variable aleatoria V

```
[160]: X = df.drop(['median_house_cat', 'ocean_proximity', 'median_house_value'], axis=1)
y = df['median_house_cat']
umbral=0.055
selected_features, forest = entrenar_random_forest(X, umbral)
```



```
-----  
          Feature  Importance  
0   location_price_cluster  0.411016  
1           median_income  0.211870  
2   ocean_proximity_encoded  0.127865  
3             longitude  0.087749  
4             latitude  0.080685  
5   total_rooms  0.018866  
6   housing_median_age  0.016512  
7           population  0.014965  
8           households  0.011832  
9   total_bedrooms  0.011175  
10          Var  0.007465
```

**El clúster obtenido a partir de la técnica de k-means se ha mostrado la característica más relevante a la hora de explicar el precio medio de un bloque.**

Las variables que han sido seleccionadas con importancia mayor a 0.055 son:

```
-----  
          Feature  Importance  
0   location_price_cluster  0.411016  
1           median_income  0.211870  
2   ocean_proximity_encoded  0.127865  
3             longitude  0.087749  
4             latitude  0.080685
```

#### 2.4.4 Conclusiones Feature Selection

##### Variables muy poco significativas

- En principio la mayoría de variables se muestran significativas en su relación con la variable objetivo Valor medio de la vivienda.

##### Variables más significativas

- La variable cluster que hemos creado 'location\_price\_cluster' es la que muestra mayor grado de significación, junto a la mediana del ingreso (median\_income) y la variable codificada 'ocean\_proximity\_encoded', que obtuvimos ordenando la variable original en función de la proximidad a la costa. Las coordenadas longitud y latitud también muestran significación

##### \*\*Variables con significación \*\*

- Las coordenadas longitud y latitud también muestran significación, si bien están ya recogidas en la variable 'location\_price\_cluster'
- Las variables total\_rooms, population, total\_bedrooms y households también muestran significación, si bien la alta correlación entre ellas nos lleva a considerar tan sólo una de ellas a la hora de entrenar nuestros modelos.

### 3. Entrenamiento y evaluación de modelos

```
# Hacemos dummy variable 'location_price_cluster'  
df = pd.get_dummies(df, columns=['location_price_cluster'], drop_first=True)  
  
# Seleccionamos las features que vamos a usar en nuestro entrenamiento, descartando las no significativas  
selected_features = ['latitude', 'housing_median_age', 'total_rooms', 'median_income', 'ocean_proximity_encoded', 'location_price_cluster_1',  
'location_price_cluster_2', 'location_price_cluster_3', 'location_price_cluster_4', 'location_price_cluster_5',  
'location_price_cluster_6', 'location_price_cluster_7']
```

Para intentar predecir el precio medio de la vivienda de cada bloque en función de las características de las que disponemos, usaremos los siguientes modelos de Machine Learning:

#### 1. RandomForestRegressor

**Descripción:** Es un modelo de aprendizaje automático basado en árboles de decisión. Crea múltiples árboles de decisión y promedia sus predicciones.

**Ventajas:** Es robusto frente a sobreajuste y maneja bien datos no lineales y características categóricas.

**Apropiado para estimar precios de viviendas:** Puede capturar relaciones complejas entre las características (como tamaño, ubicación, número de habitaciones) y el precio.

#### 2. XGBRegressor (XGBoost)

**Descripción:** Es una implementación optimizada de gradient boosting. Construye árboles de decisión secuencialmente, donde cada árbol corrige los errores del anterior.

**Ventajas:** Alta precisión, manejo eficiente de datos faltantes y regularización para evitar sobreajuste.

**Apropiado para estimar precios de viviendas:** Su capacidad para manejar datos complejos y su precisión lo hacen ideal para este tipo de tareas.

#### 3. LinearRegression

**Descripción:** Es un modelo de regresión lineal simple que asume una relación lineal entre las características y la variable objetivo.

**Ventajas:** Fácil de interpretar y rápido de entrenar.

**Apropiado para estimar precios de viviendas:** Funciona bien si la relación entre las características y el precio es aproximadamente lineal.

#### 4. PolynomialFeatures (degree=2 y 3)

**Descripción:** Transforma las características originales en características polinómicas de segundo y tercer grado.

**Ventajas:** Captura relaciones no lineales más complejas que la regresión lineal simple.

**Apropiado para estimar precios de viviendas:** Útil si hay relaciones cúbicas entre las características y el precio, aunque puede ser más propenso al sobreajuste.

#### 5. Multilayer Perceptron (Perceptrón Multicapa).

**Descripción:** Es un tipo de red neuronal artificial que se utiliza comúnmente en problemas de clasificación y regresión.

**Ventajas:**

**Capacidad de Capturar Relaciones Complejas:** Las redes neuronales son muy buenas para capturar relaciones no lineales complejas entre las características y la variable objetivo.

**Flexibilidad:** Puedes ajustar el número de capas y neuronas para mejorar el rendimiento del modelo.

**Generalización:** Con suficiente entrenamiento y datos, las redes neuronales pueden generalizar bien a nuevos datos.

Este modelo es adecuado para estimar el precio medio de una vivienda porque:

**Relaciones No Lineales:** Puede capturar relaciones complejas entre las características de la vivienda (como tamaño, ubicación, número de habitaciones) y el precio.

**Escalabilidad:** Puede manejar grandes conjuntos de datos y aprender patrones complejos.

**Personalización:** Puedes ajustar la arquitectura del modelo (número de capas y neuronas) según las necesidades específicas del problema.

```
def entrenar_modelos(X, y):  
    df_metricas = pd.DataFrame(columns=['Modelo', 'R2', 'Mean absolute error'])  
  
    # Dividir Los datos en conjuntos de entrenamiento y prueba  
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)  
  
    # Definiendo Los modelos que vamos a utilizar  
    modelos = [  
        RandomForestRegressor(max_depth=20, min_samples_split=10, n_estimators=300, min_samples_leaf=10,  
                             max_features='sqrt', random_state=42),  
        XGBRegressor(max_depth=4, n_estimators=100, alpha=10, learning_rate=0.1,  
                     objective='reg:squarederror', random_state=42),  
        LinearRegression(),  
        Pipeline([  
            ('poly', PolynomialFeatures(degree=2)),  
            ('linear', LinearRegression())  
        ]),  
        Pipeline([  
            ('poly', PolynomialFeatures(degree=3)),  
            ('linear', LinearRegression())  
        ])  
    ]  
  
    # Imprimir Los conjuntos de datos  
    print("X_train:\n", X_train.shape)  
    print("X_test:\n", X_test.shape)  
    print("y_train:\n", y_train.shape)  
    print("y_test:\n", y_test.shape)  
  
    num = 0  
    for modelo in modelos:  
        print("-" * 50)  
        print(f'Modelo: {type(modelo).__name__}')  
  
        pipe = Pipeline([  
            ('scaler', StandardScaler()),  
            ('modelo', modelo)  
        ])  
  
        # Entrenar el modelo  
        pipe.fit(X_train, y_train)  
  
        # Obtener Las predicciones para nuevos datos, usando Los datos de prueba  
        y_pred = pipe.predict(X_test)  
  
        # Evaluar el modelo  
        mse = round(mean_squared_error(y_test, y_pred), 2)  
        r2 = round(r2_score(y_test, y_pred), 2)  
        mae = round(mean_absolute_error(y_test, y_pred), 2)  
  
        print(f'Mean Squared Error: {mse}')  
        print(f'Mean Absolute Error: {mae}')  
        print(f'R-Squared: {r2}')
```

```

# Visualización de predicciones vs valores reales
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

# Gráfico de dispersión de predicciones vs valores reales
ax1.scatter(y_test, y_pred, color='blue', alpha=0.5)
ax1.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
ax1.set_xlabel('Valores reales')
ax1.set_ylabel('Predicciones')
ax1.set_title('Predicciones vs Valores reales')

# Histograma de residuos
rf_residuals = y_test - y_pred
sns.histplot(rf_residuals, bins=30, kde=True, color='green', ax=ax2)
ax2.set_title('Histograma de Residuos')
#ax2.set_xticks(ax2.get_xticks(), rotation=45)
plt.setp(ax2.get_xticklabels(), rotation=45) # Rotar las etiquetas de los ejes x

plt.tight_layout()
plt.show()

if isinstance(modelo, Pipeline):
    poly_features = modelo.named_steps.get('poly')
    if poly_features:
        if poly_features.degree == 2:
            nombre_modelo = 'Polynomial Regression (Degree 2)'
        elif poly_features.degree == 3:
            nombre_modelo = 'Polynomial Regression (Degree 3)'
    else:
        nombre_modelo = type(modelo).__name__

df_metricas.loc[num] = [nombre_modelo, r2, mae]

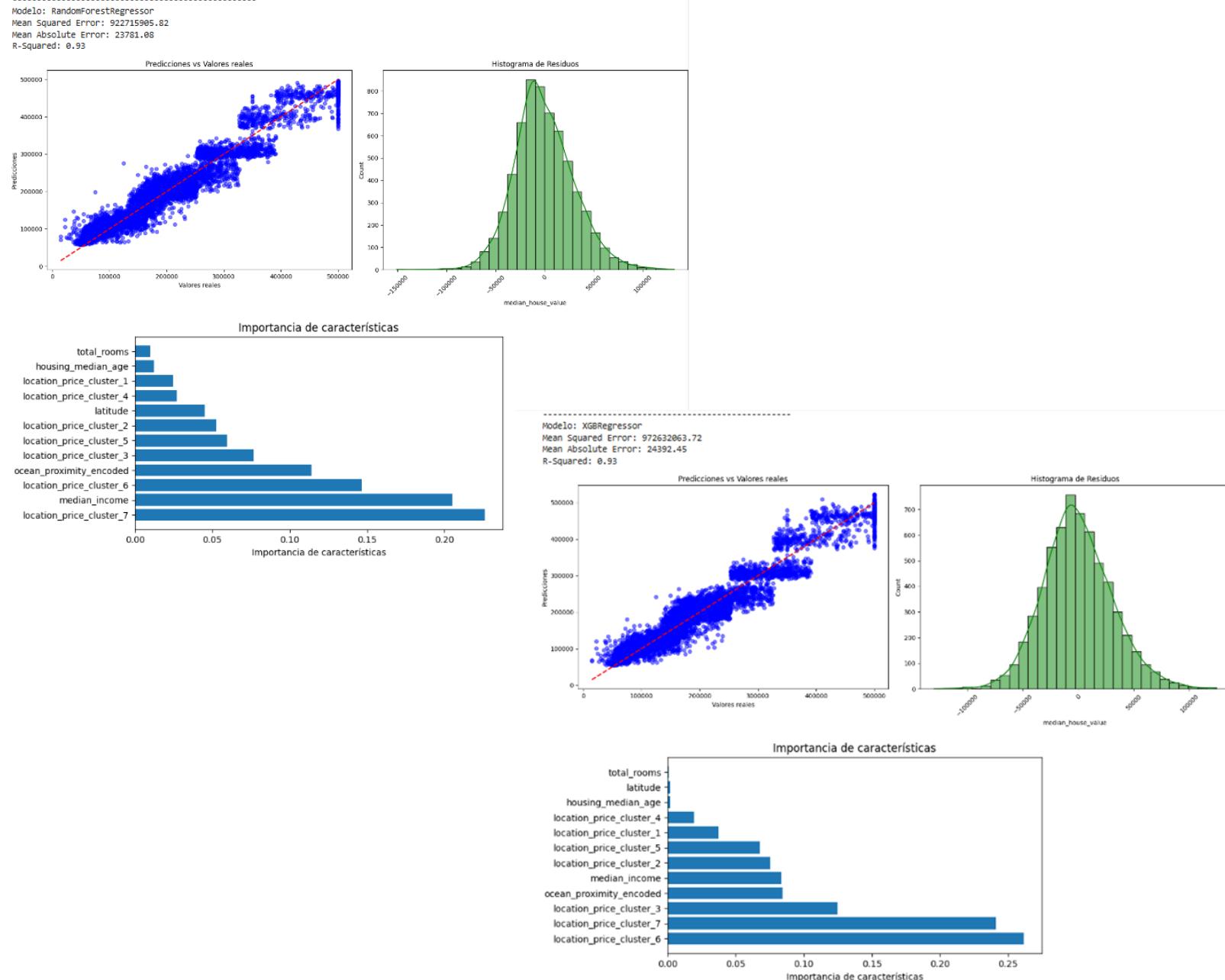
# Añadiendo importancia de características
if hasattr(modelo, 'feature_importances_'):
    # Modelos como RandomForest y XGBoost
    feature_importance = modelo.feature_importances_
    importance_df = pd.DataFrame({
        'feature': X.columns,
        'importance': feature_importance
    }).sort_values(by='importance', ascending=False)

    # Crear el gráfico de importancia de características
    plt.figure(figsize=(8, 4))
    plt.bar(importance_df['feature'], importance_df['importance'])
    plt.xlabel('Importancia de características')
    plt.title(f'Importancia de características')
    plt.tight_layout()
    plt.show()

    num += 1

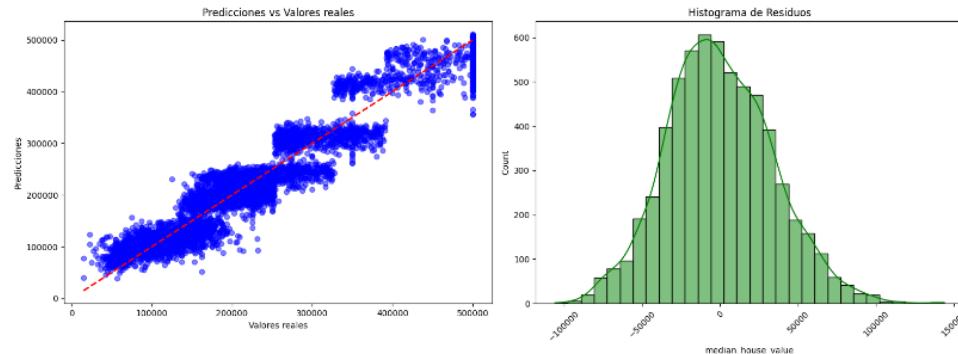
return df_metricas

```



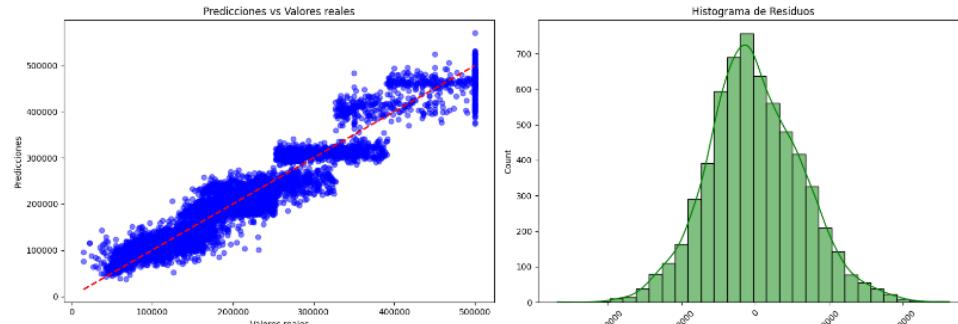
Modelo: LinearRegression  
Mean Squared Error: 1156943827.81  
Mean Absolute Error: 27170.72  
R-Squared: 0.91

### Linear Regression



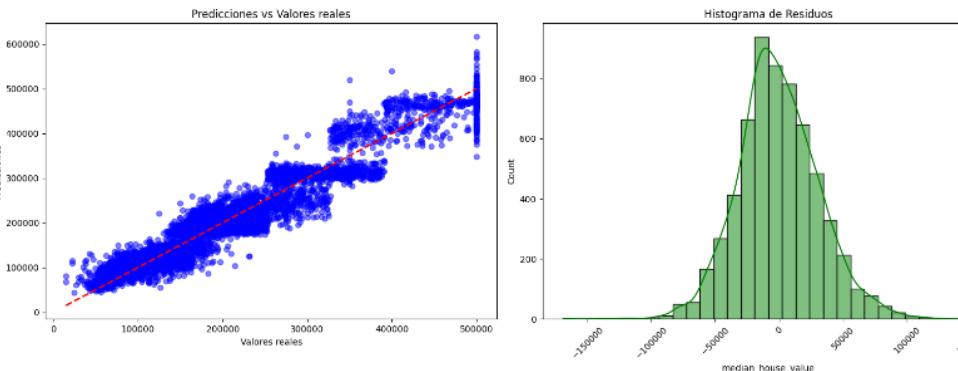
Modelo: Pipeline  
Mean Squared Error: 1031572864.33  
Mean Absolute Error: 25324.5  
R-Squared: 0.92

### Polynomial Regression (Degree 2)



Modelo: Pipeline  
Mean Squared Error: 990286895.74  
Mean Absolute Error: 24635.5  
R-Squared: 0.93

### Polynomial Regression (Degree 3)



### 5. Multilayer Perceptron (Perceptrón Multicapa).

Descripción: Es un tipo de red neuronal artificial que se utiliza comúnmente en problemas de clasificación y regresión.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Dividir Los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Escalar Los datos
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Definir el modelo de red neuronal
model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1))

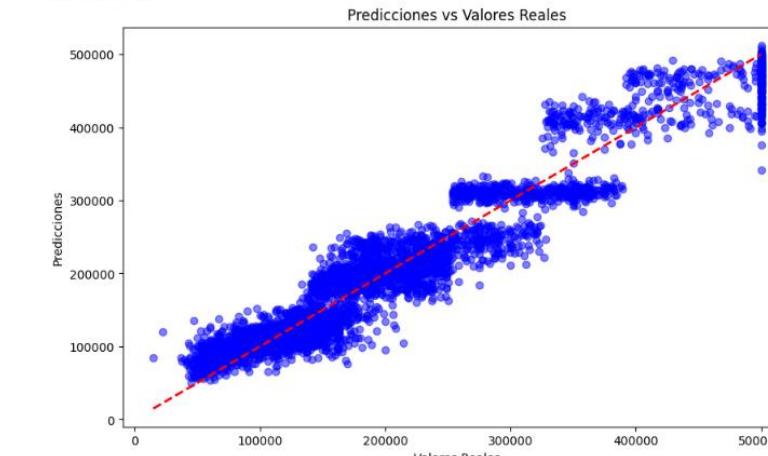
# Compilar el modelo
model.compile(optimizer='adam', loss='mean_squared_error')

# Entrenar el modelo
history = model.fit(X_train_scaled, y_train, epochs=100, validation_split=0.2, verbose=1)

# Evaluar el modelo
train_loss = model.evaluate(X_train_scaled, y_train, verbose=0)
test_loss = model.evaluate(X_test_scaled, y_test, verbose=0)
print(f'Training Loss: {train_loss}')
print(f'Testing Loss: {test_loss}')

# Hacer predicciones
y_pred = model.predict(X_test_scaled)
```

-----  
Modelo: Multilayer Perceptron  
Mean Squared Error: 1096978986.05  
Mean Absolute Error: 26661.78  
R-Squared: 0.92



## 4. Modelo final

df\_metricas

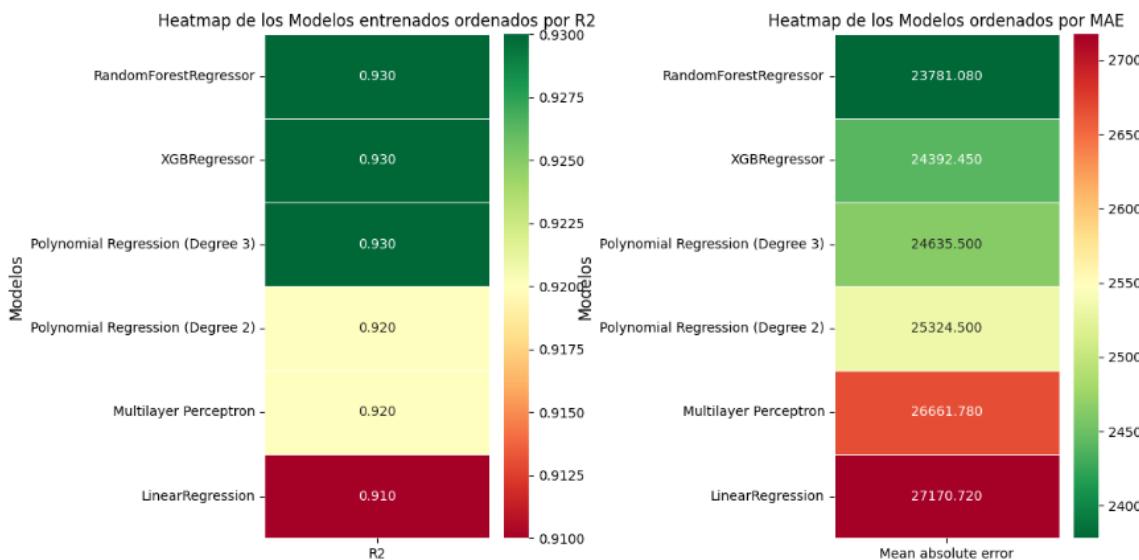
|   | Modelo                           | R2   | Mean absolute error |
|---|----------------------------------|------|---------------------|
| 0 | RandomForestRegressor            | 0.93 | 23781.08            |
| 1 | XGBRegressor                     | 0.93 | 24392.45            |
| 2 | LinearRegression                 | 0.91 | 27170.72            |
| 3 | Polynomial Regression (Degree 2) | 0.92 | 25324.50            |
| 4 | Polynomial Regression (Degree 3) | 0.93 | 24635.50            |
| 5 | Multilayer Perceptron            | 0.92 | 26661.78            |

```
# Crear una figura con dos subplots
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

# Primer heatmap: solo con la variable 'R2'
heatmap_data_r2 = df_metricas[['Modelo', 'R2']].set_index('Modelo').sort_values(by='R2', ascending=False)
sns.heatmap(heatmap_data_r2, annot=True, cmap='RdYlGn', linewidths=.5, fmt=".3f", ax=axes[0])
axes[0].set_title('Heatmap de los Modelos entrenados ordenados por R2', fontsize=12)
axes[0].set_ylabel('Modelos', fontsize=12)
#axes[0].set_xlabel('R2', fontsize=12)

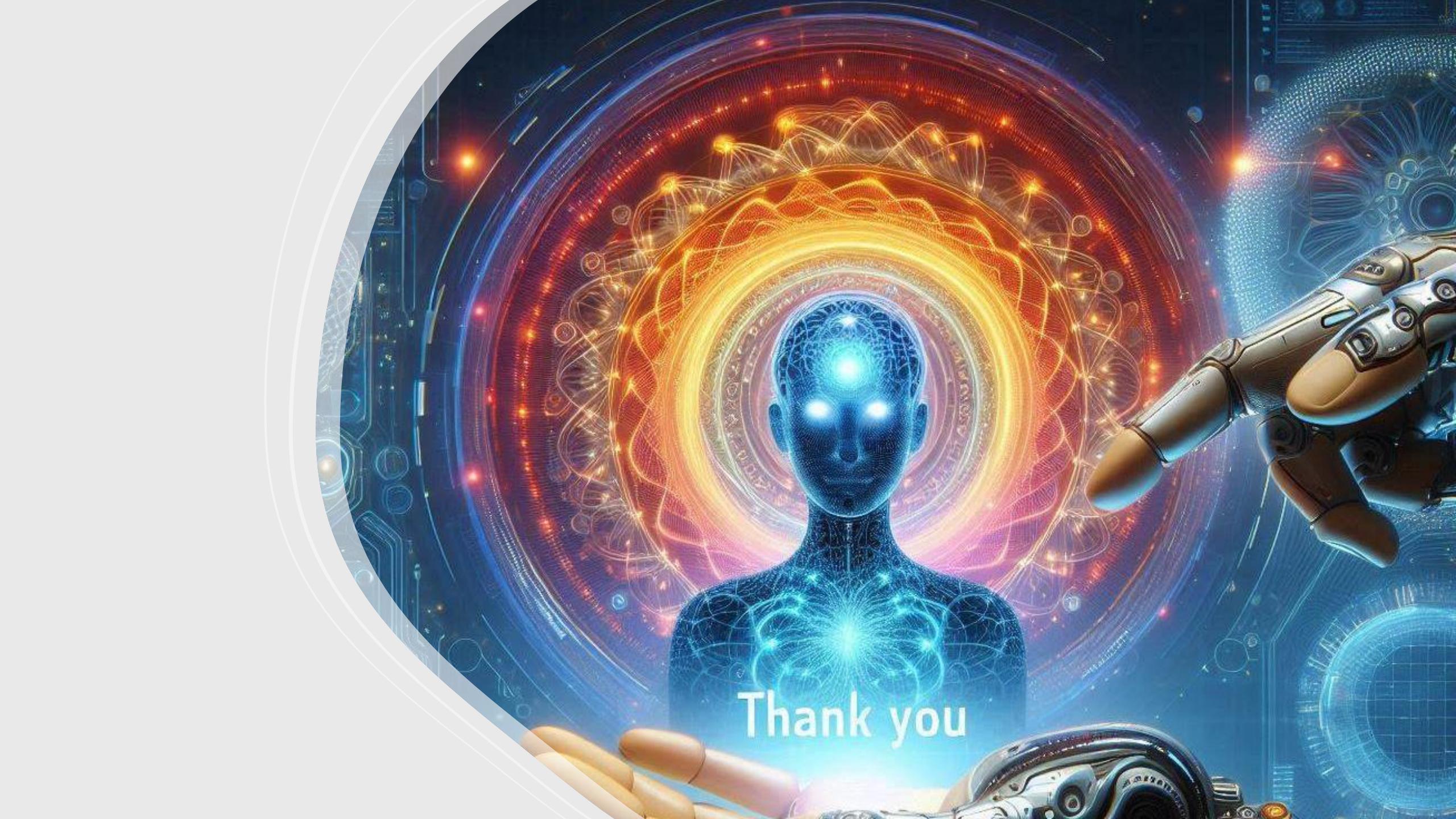
# Segundo heatmap: solo con la variable 'Mean absolute error'
heatmap_data_mae = df_metricas[['Modelo', 'Mean absolute error']].set_index('Modelo').sort_values(by='Mean absolute error', ascending=True)
sns.heatmap(heatmap_data_mae, annot=True, cmap='RdylGn_r', linewidths=.5, fmt=".3f", ax=axes[1]) # Usar 'RdylGn_r' para invertir los colores
axes[1].set_title('Heatmap de los Modelos ordenados por MAE', fontsize=12)
axes[1].set_ylabel('Modelos', fontsize=12)
#axes[1].set_xlabel('Mean absolute error', fontsize=12)

# Ajustar el layout para que no se solapen los títulos y etiquetas
plt.tight_layout()
plt.show()
```



De todos los modelos entrenados nos quedamos con el modelo **RandomForest** que ha dado un alto nivel de precisión, con un R<sup>2</sup> del orden de 0.93\*, que sugiere un alto poder predictivo y un error cuadrático medio (MAE) de 23.780 \$ (lo que indica, que, en promedio, las predicciones del modelo son del orden de ± 23.780 \$ de su valor real).

\*En términos simples, un R<sup>2</sup> de 0.93 significa que el 93% de la variabilidad en los precios de las viviendas puede ser explicada por el modelo basándose en las variables que ha utilizado, lo que muestra un modelo muy preciso.



Thank you