2025

Proyecto 2

PEA DATA SCIENCE & IA

CARLOS ARREGUI Y JUAN PABLO DELZO

Tabla de contenido

Tabla de figuras

1. Introducción

El principal objetivo es que una vez ya desarrollado el primer proyecto de forma tradicional y gracias al feedback recibido, el siguiente paso es utilizar herramientas en IA y otras herramientas ya conocidas que no requieran programación.

El proyecto consta de 4 apartados:

- El primero está desarrollado con *WindSurf*, con el fin de crear codigo con el mínimo esfuerzo en Python (solo se tuvo que corregir o añadir puntualmente).
- El segundo apartado está desarrollado con *MySQL*, *DBeaver* y *Aiven*.
- El tercer apartado está desarrollado con *BigML*, *Vertex AI* e I*BM Wats*onx.
- El último apartado está desarrollado en PowerBI con Python y PowerQuery en el backend, y *DAX* en el frontend.

Preprocesamiento de datos:

- Análisis exploratorio de datos.
- Llenado o eliminación de datos nulos.
- Eliminación de atípicos.
- Escalado de datos.
- Transformación de variables.

Carga de datos:

- Creación de base de datos sin herramientas de programación

Procesamiento de datos y métodos supervisados:

- Herramientas ML en IA

Explotación de datos:

- Visualización interactiva de los datos.
- Visualización de modelos.

2. Exploración y limpieza de los datos

Mediante *WindSurf* se realizaron la siguiente serie de peticiones:

```
Con Python, Pandas y Matplotlib con fondo fivethirtyeight, con el
archivo adjunto en input:

1.Elimina la columna Unnamed: 0.

2.Analiza qué columnas tienen valores nulos.

3.Si la columna Neighborhood no tiene valores nulos, reemplaza
los valores nulos en función de la moda en su columna de cada
valor de Neighborhood.

4.En las columnas Rooms, Bathrom y Square_meters, reemplaza a
valor int redondeando los valores quedándote con la parte entera.

5.Chequea si Square_meters_price * Square_meters difieren con
price con una tolerancia de 1000 euros.

6.Elimina la columna Square_meters_price. Detecta si hay outliers
con el método de IQR con k igual a 3 en Price y apártalos del
dataframe.

7.Analiza si hay una relación lineal entre Price promedio y
Neighborhood. Si tal valor es menor de 0.1, con get_dummies de
Pandas, transforma la columna Neighborhood con drop_first True y
dtype a int.

8.Guarda dos archivos con extensión csv, uno con la tabla
transformada con get_dummies y el otro sin transformarlo.
```

El código, gráfico y tablas generados tardaron menos de dos minutos, fueron los siguientes:

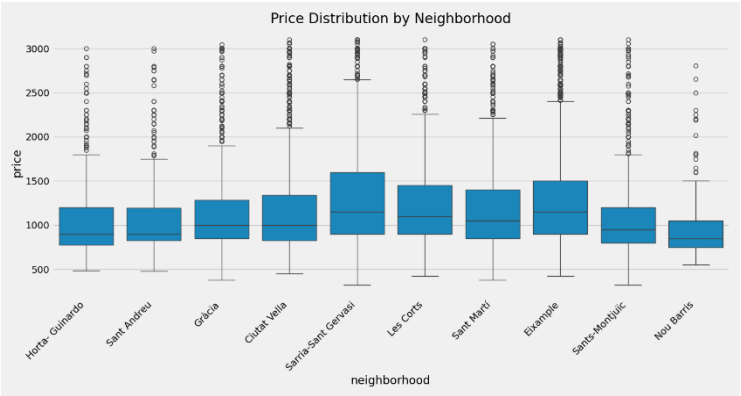


Figura 1. Boxplot de llos precios de alquiler por distrito

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy import stats

# Set the style for plots
plt.style.use('fivethirtyeight')

# Read the data
df = pd.read_csv('input/DATA_Barcelona_Fotocasa_HousingPrices_Augmented.csv')
print("\n")
print(df.dtypes)

# Remove Unnamed: 0 column
df = df.drop('Unnamed: 0', axis=1)

# Check null values
print("\nNull values in each column:")
print(df.isnull().sum())

# If neighborhood has no null values, fill nulls based on mode per neighborhood
if df['neighborhood'].isnull().sum() == 0:
    for column in df.columns:
        if column != 'neighborhood' and df[column].isnull().any():
            df[column] = df.groupby('neighborhood')[column].transform(lambda x: x.fillna(x.mode().iloc[0] if not x.mode().empty else x.mean()))

# Convert rooms, bathroom and square_meters to int
df['rooms'] = df['rooms'].round().astype(int)
df['bathroom'] = df['bathroom'].round().astype(int)
df['square_meters'] = df['square_meters'].round().astype(int)

# Check price consistency
price_diff = abs(df['square_meters_price'] * df['square_meters'] - df['price'])
inconsistent_prices = price_diff > 1000
print("\nNumber of inconsistent prices:", inconsistent_prices.sum())

# Remove square_meters_price column
df = df.drop('square_meters_price', axis=1)

# Detect outliers using IQR method
Q1 = df['price'].quantile(0.25)
Q3 = df['price'].quantile(0.75)
IQR = Q3 - Q1
k = 2.5
lower_bound = Q1 - k * IQR
upper_bound = Q3 + k * IQR

# Remove outliers
df_no_outliers = df[(df['price'] >= lower_bound) & (df['price'] <= upper_bound)]
print(f"\nRemoved {len(df) - len(df_no_outliers)} outliers")

# Create boxplot
plt.figure(figsize=(15, 8))
sns.boxplot(data=df_no_outliers, x='neighborhood', y='price')
plt.xticks(rotation=45, ha='right')
plt.title('Price Distribution by Neighborhood')
plt.tight_layout()
plt.savefig('output/price_neighborhood_boxplot.png')
plt.close()

# Calculate correlation using one-way ANOVA
neighborhoods = df_no_outliers['neighborhood'].unique()
price_groups = [df_no_outliers[df_no_outliers['neighborhood'] == n]['price'] for n in neighborhoods]
f_statistic, p_value = stats.f_oneway(*price_groups)
correlation = np.sqrt(f_statistic / (f_statistic + len(df_no_outliers) - len(neighborhoods)))
print(f"\nCorrelation strength (based on ANOVA): {correlation:.4f}")

# If correlation is less than 0.25, create dummy variables
if correlation < 0.25:
    df_dummy = pd.get_dummies(df_no_outliers, columns=['neighborhood'], drop_first=True, dtype=int)
    # Save both versions
    df_dummy.to_csv('output/housing_prices_dummy.csv', index=False)
    df_no_outliers.to_csv('output/housing_prices_processed.csv', index=False)
else:
    df_no_outliers.to_csv('output/housing_prices_processed.csv', index=False)
    print("\nCorrelation is >= 0.25, dummy variables were not created")
```

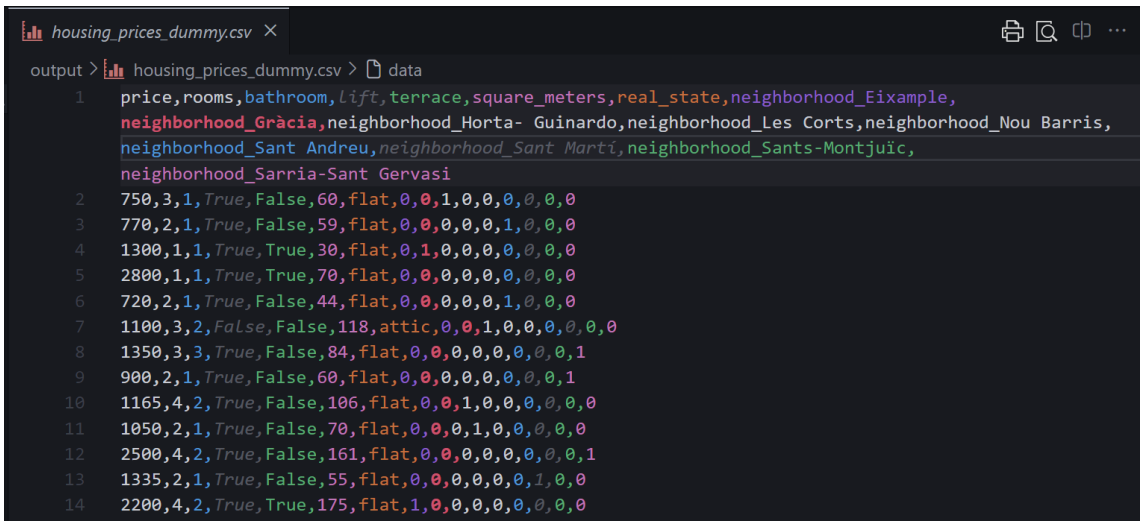
Figura 2. Código generado en Python para el ETL en el preprocesamiento de datos

housing_prices_processed.csv

output > housing_prices_processed.csv > data

```
price,rooms,bathroom,lift,terrace,square_meters,real_state,neighborhood
750,3,1,True,False,60,flat,Horta- Guinardo
770,2,1,True,False,59,flat,Sant Andreu
1300,1,1,True,True,30,flat,Gràcia
2800,1,1,True,True,70,flat,Ciutat Vella
720,2,1,True,False,44,flat,Sant Andreu
1100,3,2,False,False,118,attic,Horta- Guinardo
1350,3,3,True,False,84,flat,Sarria-Sant Gervasi
900,2,1,True,False,60,flat,Sarria-Sant Gervasi
1165,4,2,True,False,106,flat,Horta- Guinardo
1050,2,1,True,False,70,flat,Les Corts
2500,4,2,True,False,161,flat,Sarria-Sant Gervasi
1335,2,1,True,False,55,flat,Sant Martí
2200,4,2,True,True,175,flat,Eixample
2500,3,2,True,True,145,flat,Sarria-Sant Gervasi
980,2,2,True,False,82,flat,Eixample
```

Figura 3. Tabla en csv de los valores preprocesados sin la columna distrito transformada



housing_prices_dummy.csv

output > housing_prices_dummy.csv > data

```
price,rooms,bathroom,lift,terrace,square_meters,real_state,neighborhood_Eixample,
neighborhood_Gràcia,neighborhood_Horta- Guinardo,neighborhood_Les Corts,neighborhood_Nou Barris,
neighborhood_Sant Andreu,neighborhood_Sant Martí,neighborhood_Sants-Montjuïc,
neighborhood_Sarria-Sant Gervasi
750,3,1,True,False,60,flat,0,0,1,0,0,0,0,0,0
770,2,1,True,False,59,flat,0,0,0,0,0,1,0,0,0
1300,1,1,True,True,30,flat,0,1,0,0,0,0,0,0,0
2800,1,1,True,True,70,flat,0,0,0,0,0,0,0,0,0
720,2,1,True,False,44,flat,0,0,0,0,0,1,0,0,0
1100,3,2,False,False,118,attic,0,0,1,0,0,0,0,0,0
1350,3,3,True,False,84,flat,0,0,0,0,0,0,0,0,1
900,2,1,True,False,60,flat,0,0,0,0,0,0,0,0,1
1165,4,2,True,False,106,flat,0,0,1,0,0,0,0,0,0
1050,2,1,True,False,70,flat,0,0,0,1,0,0,0,0,0
2500,4,2,True,False,161,flat,0,0,0,0,0,0,0,0,1
1335,2,1,True,False,55,flat,0,0,0,0,0,0,1,0,0
2200,4,2,True,True,175,flat,1,0,0,0,0,0,0,0,0
```

Figura 4. Tabla en csv de los valores preprocesados con la columna de distritos transformados a variables binarias

3. Base de datos

3.1. Creación de la base de datos

Se crea una base de datos *MySQL* en la plataforma *Aiven*. Es una plataforma de gestión de bases de datos y servicios en la nube que permite a las empresas desplegar, operar y escalar fácilmente tecnologías de código abierto como *PostgreSQL*, *Kafka*, *OpenSearch*, *Redis*, *MySQL*, entre otras. A continuación, se detallan los pasos para la creación de la base de datos:

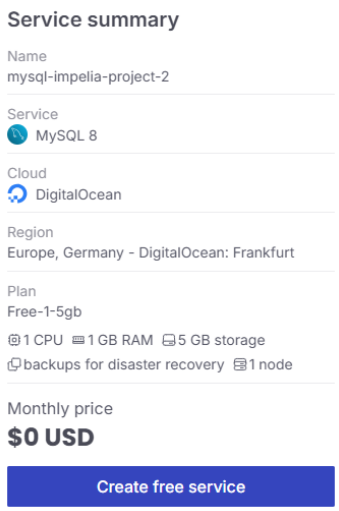


Figura 5. Creación de la cuenta en Aiven

Para la creación de la base de datos, se ha escogido un plan gratuito en el que se incluyen un servidor en Ámsterdam de 1 CPU, con 1 GB de memoria RAM y 5 GB de almacenamiento de datos. Los datos de conexión a la base de datos son los siguientes:

- Nombre defaultdb
- Host mysql-impelia-project-2-impelia-project-2.h.aivencloud.com
- Puerto 13961
- Usuario avnadmin
- Contraseña AVNS_VjEvOz2Yre8wHS8cDiw

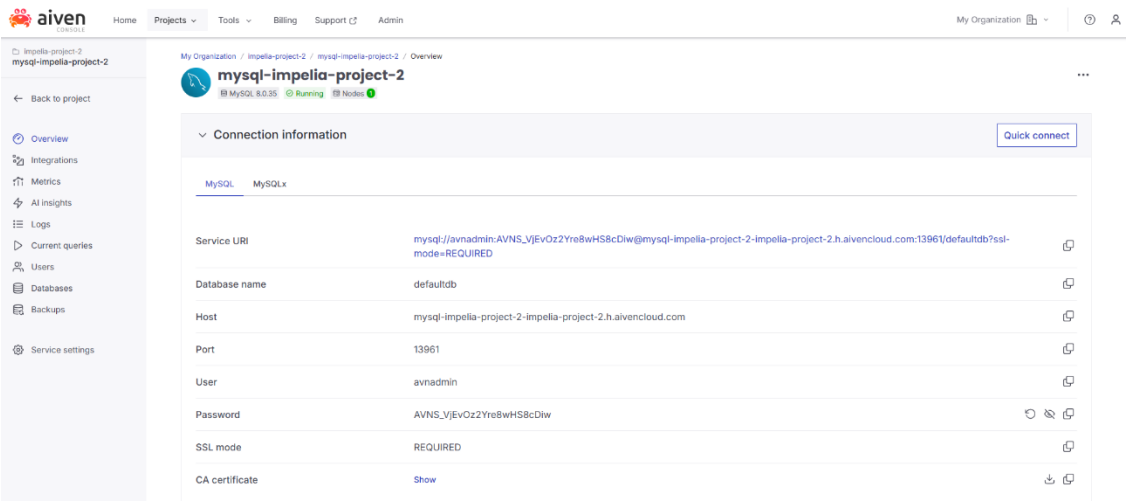


Figura 6. Credenciales asignadas para acceder a la base de datos en Aiven

3.2. Carga de datos a MySQL

Para cargar los datos en la base de datos *MySQL* de *Aiven*, se utilizó la plataforma *DBeaver*, un software gratuito especializado en la gestión de base de datos. Para poder subir los datos, primero nos conectamos a ella introduciendo la información mencionada en el apartado anterior. Además, también se autorizó usar el protocolo SSL y se cargó el certificado CA, descargado directamente desde *Aiven*.

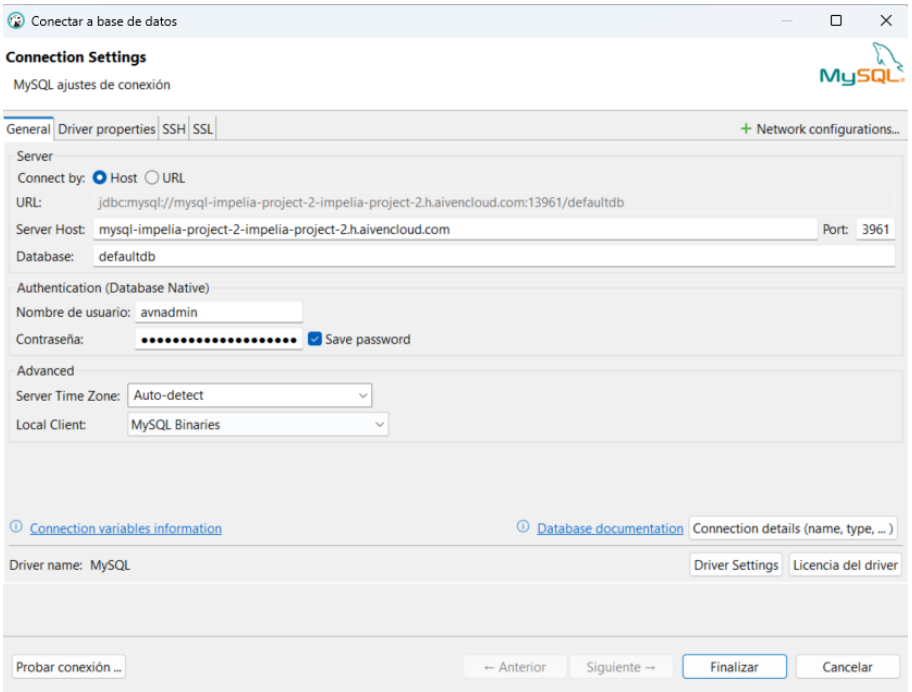


Figura 7. Ajustes de conexión a la base de datos de Aiven con DBeaver

Una vez terminada la conexión a la base de datos, se cargaron los datos del .csv posterior al ETL. Se adjuntan capturas de pantalla del proceso:

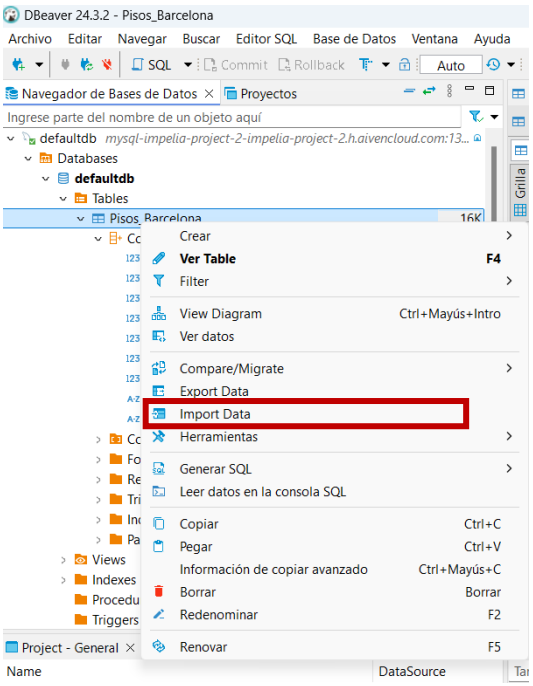


Figura 8. Carga de datos hacia la base de datos en Aiven

En primer lugar, se hace clica con el botón derecho sobre la tabla creada, y se va a *Import Data*. Acto seguido en la pantalla que se abre, se selecciona CSV como fuente de la importación.

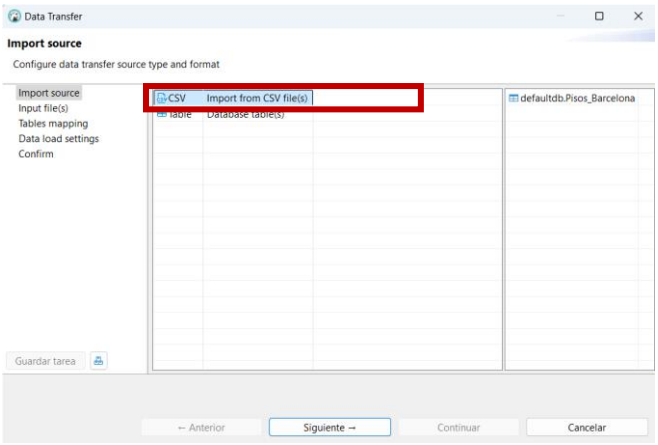


Figura 9. Selección de importación de datos mediante csv en DBeaver

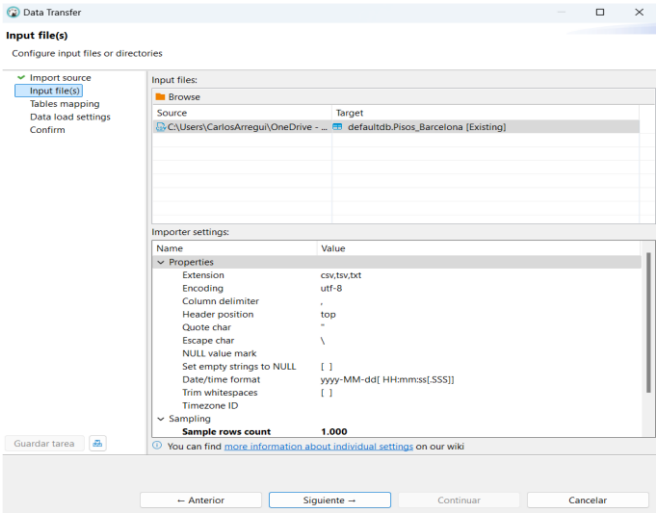


Figura 10. Configuración de parámetros para una lectura correcta de los datos

Se selecciona el archivo de la carpeta local para importar el CSV y se mapean los datos para comprobar que los está leyendo bien y que las columnas tienen el nombre que necesitamos.

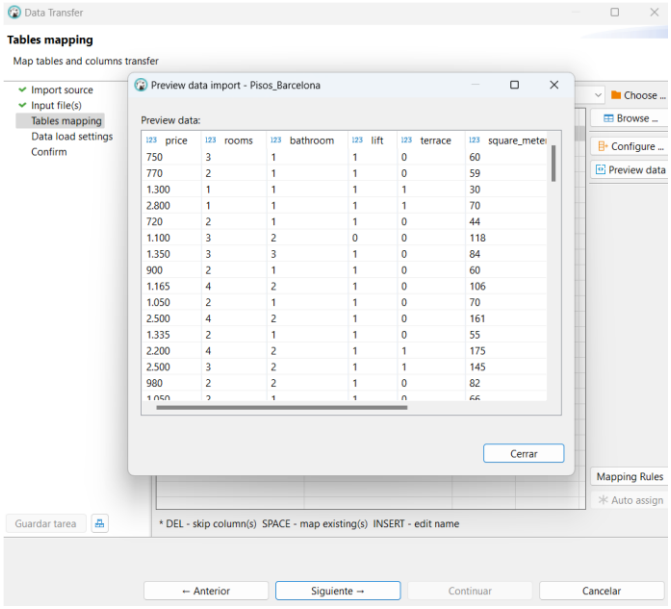


Figura 11. Comprobación de lectura de datos

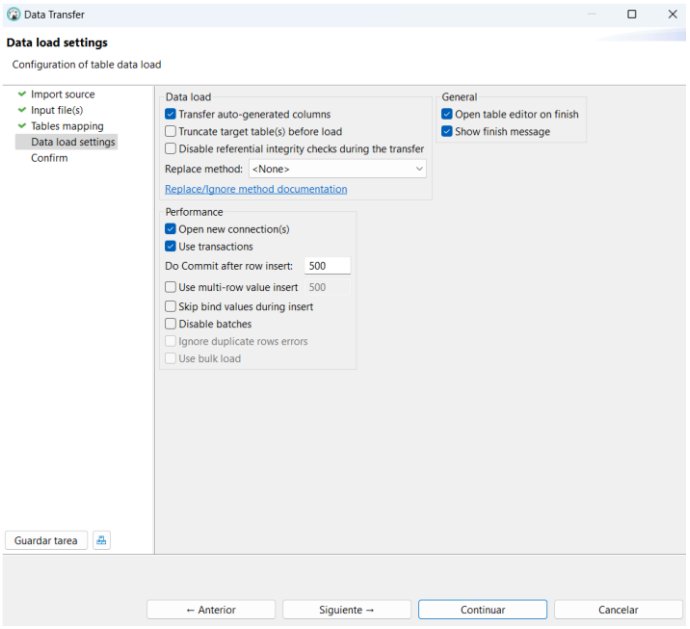


Figura 12. Configuración de parámetros para importación de datos

En la configuración de la carga de datos se deja la predeterminada, subiendo únicamente el *commit* cada 500 filas insertadas en vez de cada 100. Se clica sobre *Continuar* y se confirman todos los pasos anteriores para empezar la carga.

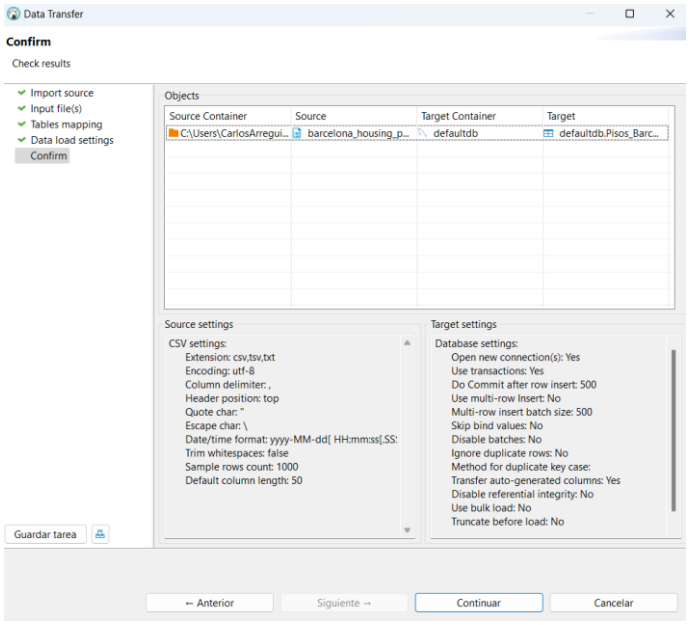


Figura 13. Paso de confirmación para la importación de los datos

Una vez terminada la carga de todas las filas del CSV, se puede comprobar que la carga ha sido realizada yendo a ver la tabla creada *Pisos_Barcelona*. Tal y como puede verse en la imagen siguiente, se ha creado automáticamente una columna que contiene un

identificador para cada piso y luego, todas las columnas que contiene el CSV separadas por una coma.

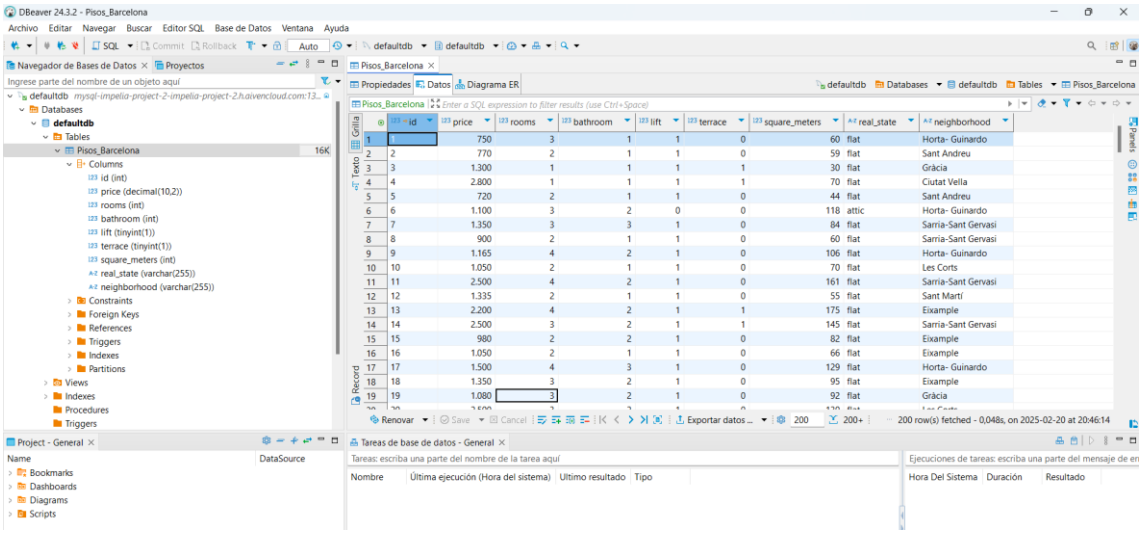


Figura 14. Imagen de demostración de la carga correcta de los datos del csv a la base de datos MySQL

4. Modelos de Machine Learning

En los últimos años, el auge del *Machine Learning* ha revolucionado múltiples sectores, incluyendo el mercado inmobiliario. La capacidad de analizar grandes volúmenes de datos y extraer patrones complejos permite mejorar la precisión en la estimación de precios de viviendas, optimizando tanto la toma de decisiones de compradores y vendedores como la estrategia de inversión de empresas del sector. En este contexto, el presente trabajo se centra en la predicción de precios de pisos en Barcelona mediante la implementación de modelos de *Machine Learning* en tres plataformas avanzadas: **BigML, Vertex AI y Watsonx**.

El objetivo principal de este estudio es construir modelos predictivos que permitan estimar el valor de un inmueble a partir de diversas características, como ubicación, superficie, número de habitaciones, antigüedad del edificio, entre otros factores relevantes. Para ello, se han utilizado diferentes técnicas de aprendizaje supervisado, explorando la efectividad de cada plataforma en términos de facilidad de uso, rendimiento del modelo y capacidad de escalabilidad.

BigML, con su enfoque centrado en la simplicidad y automatización, ha permitido una rápida implementación de modelos sin necesidad de una infraestructura compleja. Por otro lado, Vertex AI, la plataforma de Google ha proporcionado herramientas avanzadas para el entrenamiento y despliegue de modelos en la nube, facilitando la integración con grandes volúmenes de datos. Finalmente, Watsonx de IBM ha ofrecido capacidades de inteligencia artificial explicable, permitiendo analizar la interpretabilidad de los modelos y mejorar su transparencia en la toma de decisiones.

A lo largo del trabajo, se explicarán los procesos, paso a paso, que se han utilizado para crear estos modelos y se comparan los resultados obtenidos en cada plataforma, evaluando métricas de rendimiento como el error cuadrático medio (MSE), el coeficiente de determinación (R^2) y la eficiencia en el manejo de datos. Además, se analizan las ventajas y desafíos que presenta cada solución en el desarrollo de modelos para el sector inmobiliario.

Este estudio no solo busca determinar cuál de estas herramientas proporciona las predicciones más precisas, sino también aportar una visión práctica sobre su aplicabilidad en proyectos del mundo real, contribuyendo al desarrollo de modelos de estimación de precios más eficientes y accesibles para el mercado inmobiliario de Barcelona.

4.1. BigML

BigML es una plataforma de inteligencia artificial basada en la nube que se especializa en la creación de modelos de aprendizaje automático de manera accesible y simplificada. Su diseño intuitivo permite a los usuarios, independientemente de su nivel técnico, construir y desplegar modelos predictivos con facilidad.

El proceso general es sencillo y la aplicación es muy intuitiva. Primero nos conectamos a la base de datos de Aiven introduciendo los valores de Host, Puerto, Usuario, Contraseña y Nombre de la base de datos. Seguidamente se crea un dataset usando la misma fuente de datos, y finalmente se entrenan los tipos de modelos que se consideren necesarios.

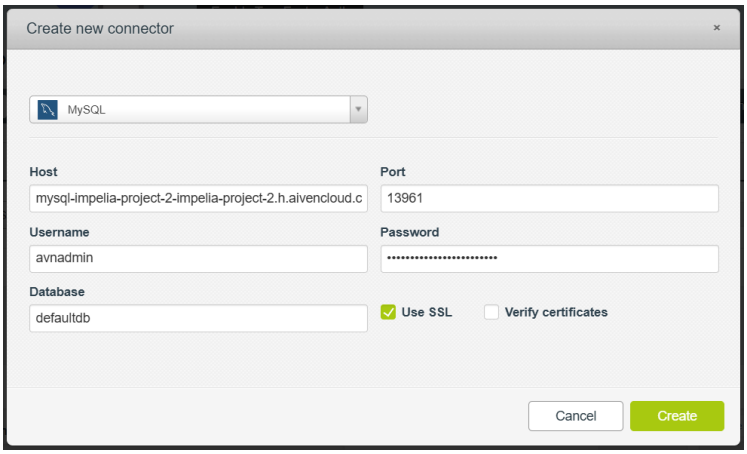


Figura 15. Creación de conexión de BigML a la base de datos MySQL de Aiven

Para crear el dataset, solo es necesario clicar a la derecha de la fuente de datos y seleccionar *1-click dataset* para crearlo.

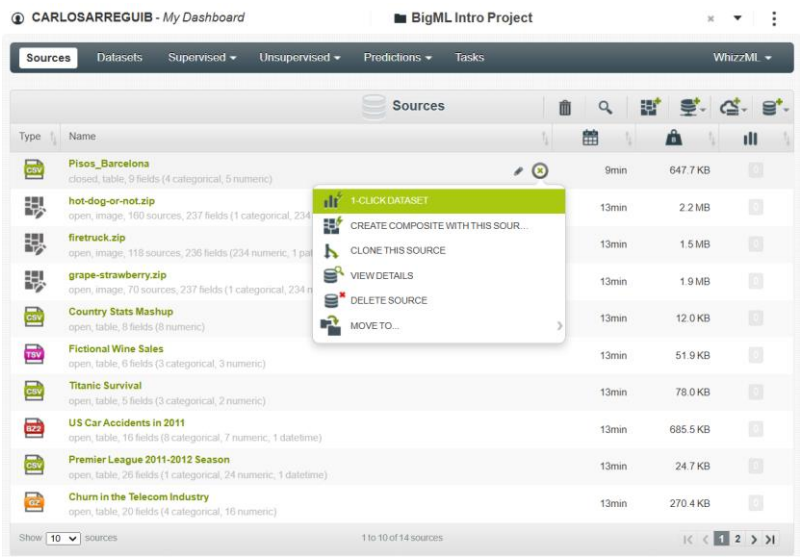


Figura 16. Creación del dataset en BigML

Una vez creado el dataset de los datos limpios de la base de datos de Aiven, se comprueba que se ha conectado correctamente y que se leen bien todos los datos. También se establece como columna objetivo el precio

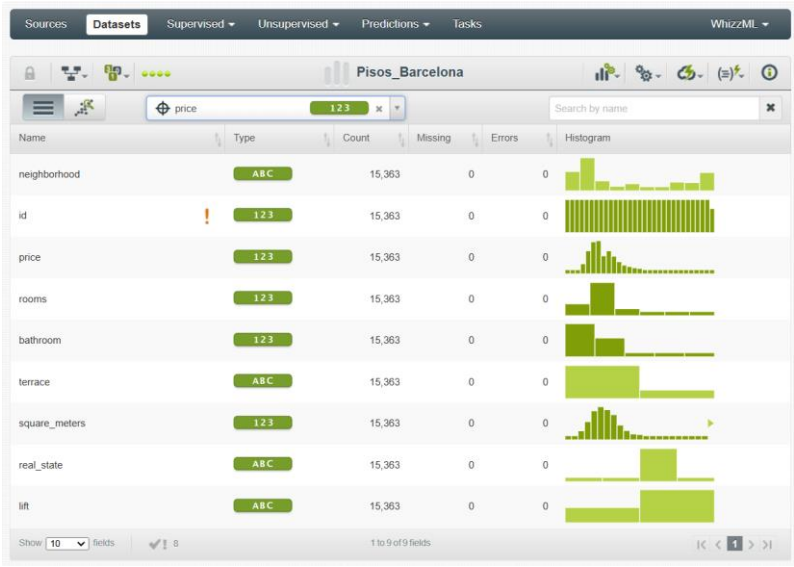


Figura 17. Comprobación del dataset y asignación de la columna objetivo en BigML

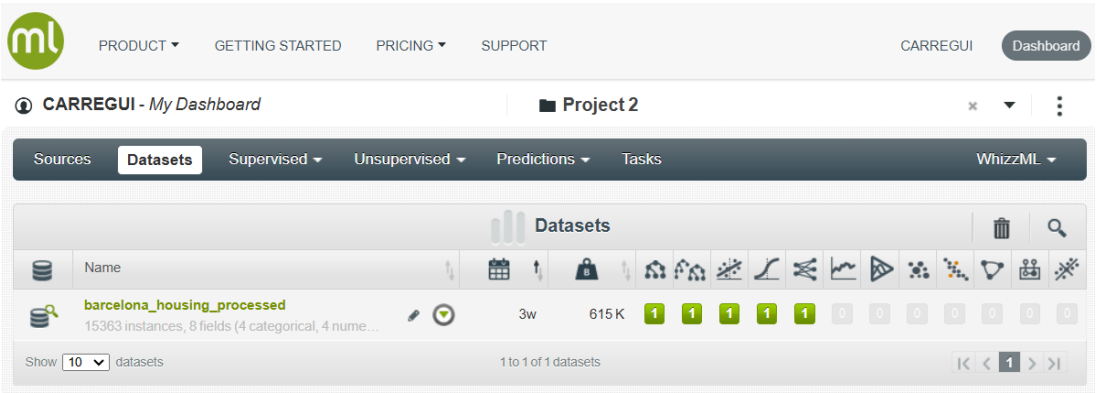


Figura 18. Dataset creado con los modelos de ML generados en verde a su derecha

Como se observa en la imagen anterior, una vez creado el dataset con el conjunto de datos, a la derecha se nos ofrecen todas las opciones de modelos disponibles (marcadas en verde). Para nuestro proyecto, se realiza la comparativa de varias opciones:

1. Random Forest
2. Ensemble
3. Linear Regression
4. DeepNet

4.2. Vertex.ai

Vertex AI es la plataforma de Google diseñada para simplificar el desarrollo, entrenamiento, implementación y monitoreo de modelos de inteligencia artificial (IA) y aprendizaje automático. Integra diversas herramientas y servicios en una sola interfaz, permitiendo a los desarrolladores y científicos de datos gestionar el ciclo de vida completo de los modelos de IA de manera eficiente.

El proceso para poder llegar a entrenar un modelo de ML a través de Vertex.ai empieza por cargar el conjunto de datos limpios. Para ello se crea un conjunto de datos y se adjunta el CSV. El nombre asignado para nuestro caso es *datos-impelia*.

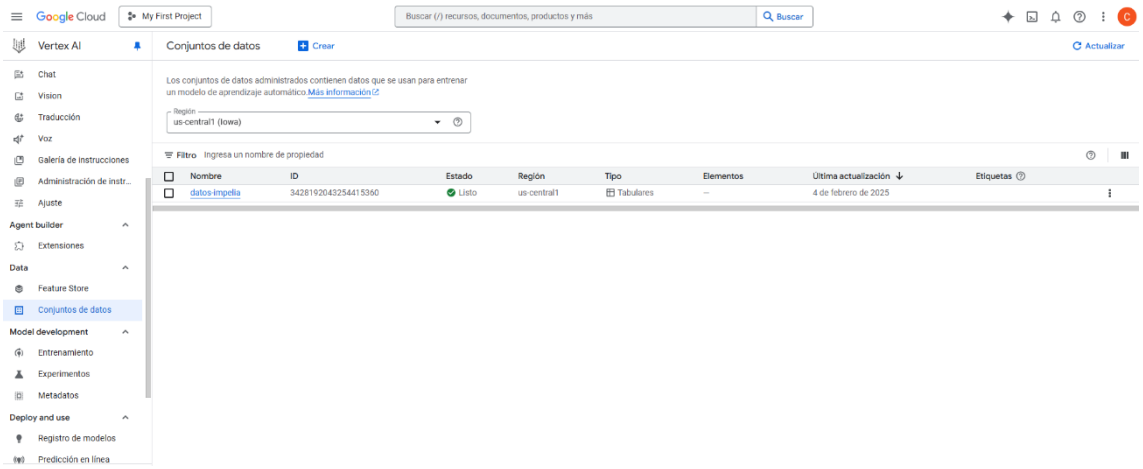


Figura 19. Carga de datos en Vertex.ai

Luego se clica en el apartado *Entrenamiento* para entrenar un modelo nuevo.

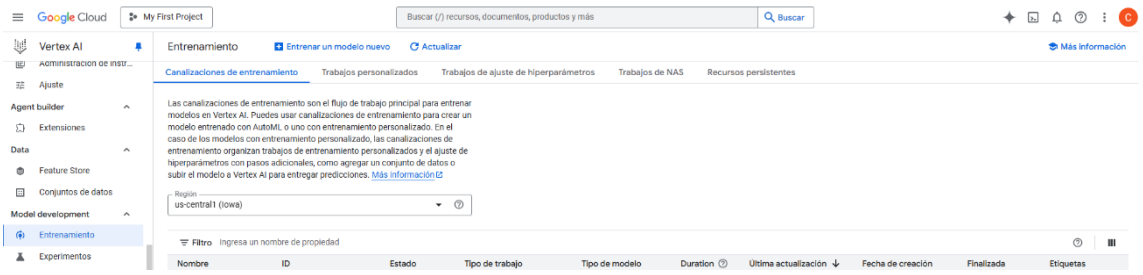


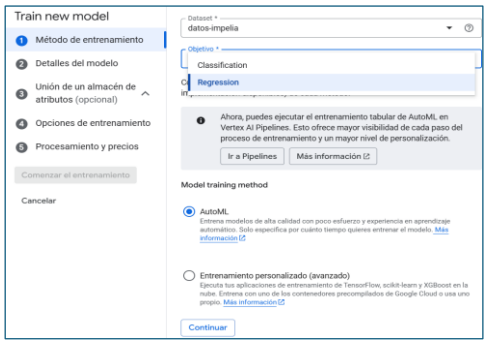
Figura 20. Ajuste para el entrenamiento de nuevos modelos en Vertex.ai

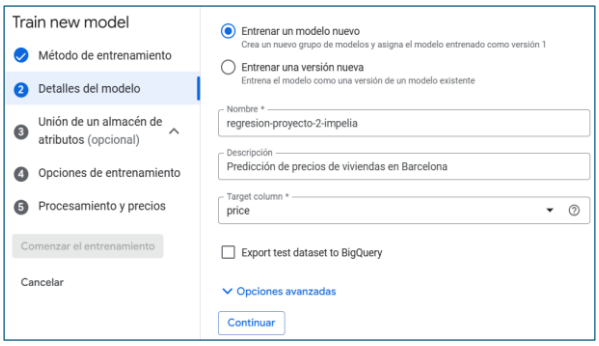
A continuación, se abre una pantalla que nos va guiando en todo el proceso de creación del modelo. En la primera pantalla se asigna el conjunto de datos que queremos utilizar el objetivo, en el cual marcaremos *Regresión*.

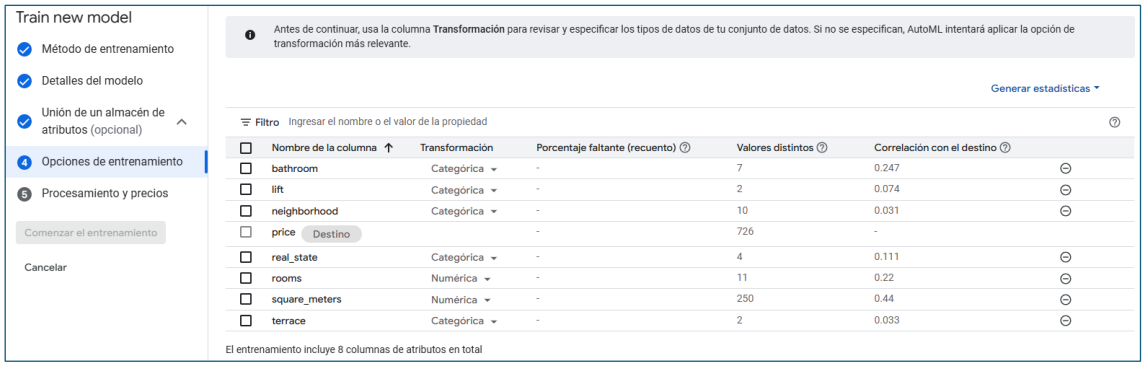
En la segunda pantalla se asigna el nombre del modelo (*regresion-proyecto-2-impelia*), una breve descripción del uso del modelo y la columna de los datos que se va a querer predecir, en nuestro caso *Price*. La tercera pantalla no la usaremos para este caso.

En la cuarta pantalla se realiza un análisis previo de los datos, pudiendo cambiar el tipo de datos de cada columna en caso de que no se haya especificado bien de forma automática, contar los valores distintos que hay en cada columna y la correlación con la columna de destino. Este paso sirve de comprobación de los datos en el caso de que no se haya hecho una ETL previa, o para comprobar que la ETL está bien hecha. También nos da la opción de eliminar alguna columna del entrenamiento del modelo si así lo consideramos. Para eliminarle peso a los pisos demasiado caros o que tengan alguna característica atípica, se selecciona el objetivo de optimización MAE, que es el error absoluto medio de modelo.

Para terminar la configuración, se ingresa un presupuesto de horas estimado, que es el tiempo estimado que tardará en finalizar. Se selecciona también la opción *Habilitar la interrupción anticipada*. Esta opción es importante porque si el modelo tarda menos en de lo esperado en entrenarse, se parará antes de consumir todas las horas establecidas como presupuesto, lo que puede ahorrar dinero y tiempo en la creación de un modelo de ML.







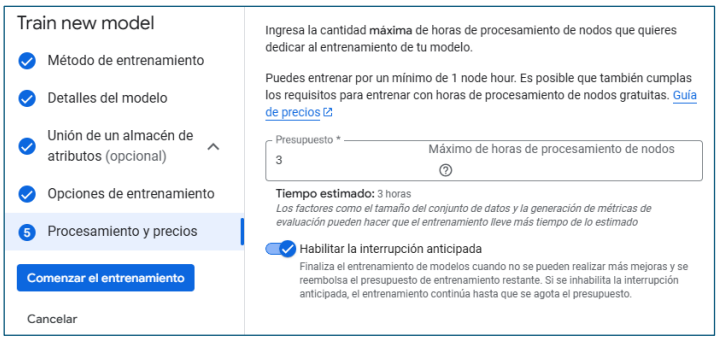


Figura 21. Secuencia de pasos para ajustar los parámetros de tiempo de ejecución y la métrica en la columna objetivo

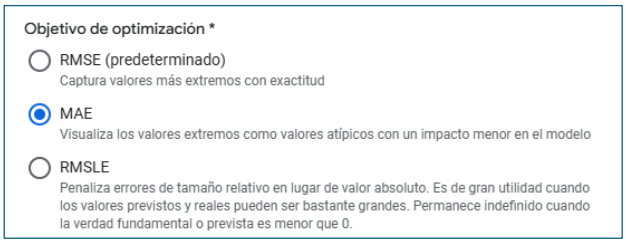


Figura 22. Selección del objetivo de optimización

Para comprobar la diferencia de modelos, se han creado dos modelos con objetivos de optimización distintos. El primer modelo se optimiza con la raíz cuadrada del promedio de los errores al cuadrado (RMSE), que va a permitirnos poder introducir y predecir pisos más caros o con características más atípicas de un modo más exacto. El segundo modelo se optimiza con el promedio de los valores absolutos de los errores (MAE), el cual se centra más en predecir mejor los pisos más comunes y deja los pisos extremos como valores atípicos.

4.3. Watson.x

Watsonx es la plataforma de inteligencia artificial empresarial de IBM, diseñada para facilitar la creación, entrenamiento, implementación y gestión de modelos de Machine Learning e IA generativa a escala.

El primer paso para crear el modelo de ML es crear una instancia. En nuestro caso, es una instancia gratuita en un servidor de Sídney con las características que se observan en la imagen:

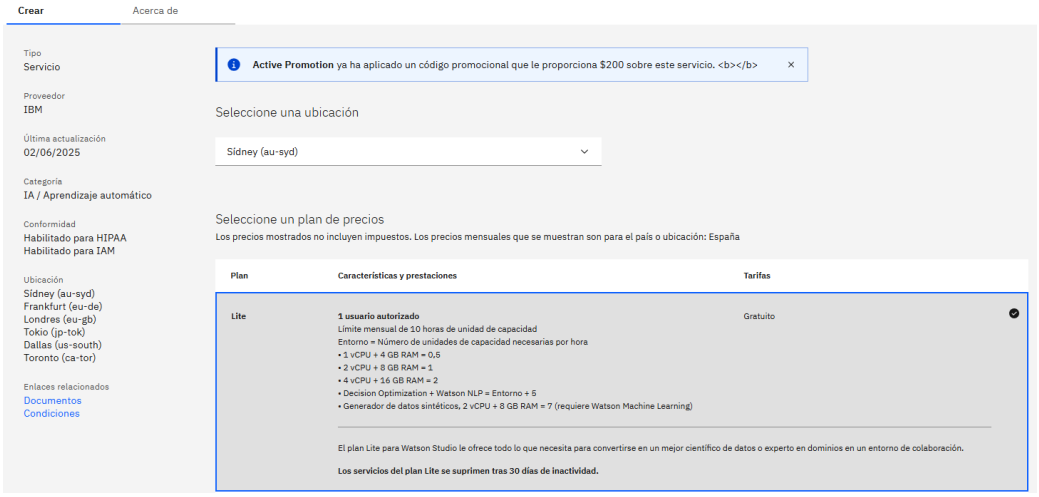


Figura 23. Creación de instancia en Watson.x

A continuación, se crea un proyecto. En este proyecto se adjuntarán los datos del CSV al que ya se le ha realizado el ETL previamente. Para cargarlos, es suficiente con arrastrar el archivo deseado a la pantalla *Importar archivos de datos*.

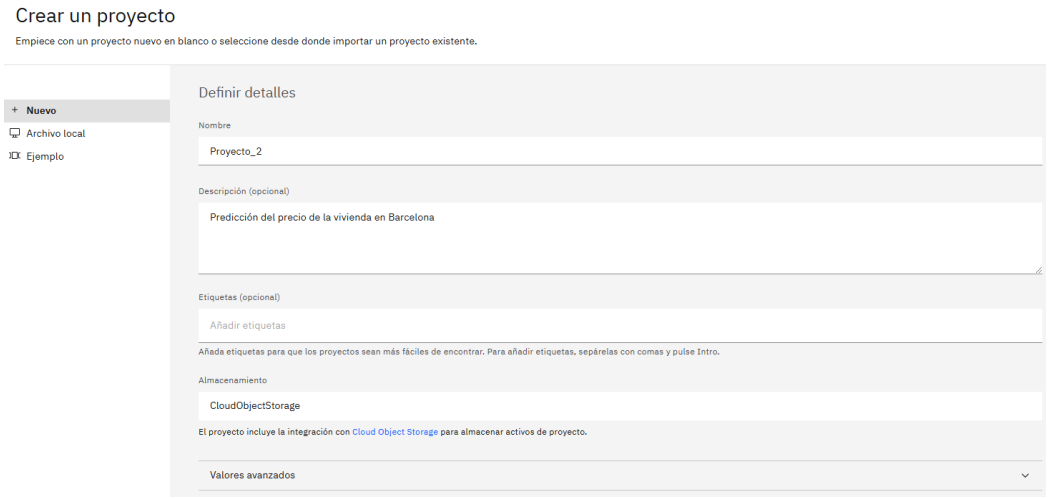


Figura 24. Creación de proyecto en Watson.x

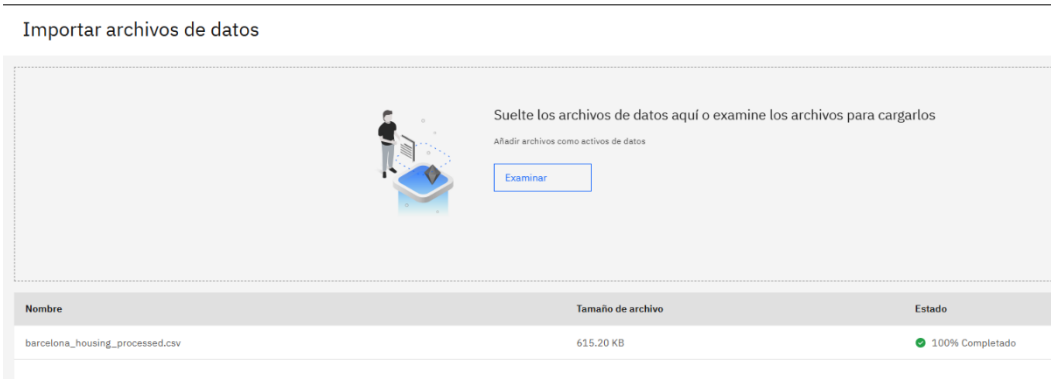


Figura 25. Importación de archivos en Watson.x

Con los datos cargados, debemos decidir qué queremos hacer con ellos. Para este proyecto, se pide crear un modelo de aprendizaje automático:

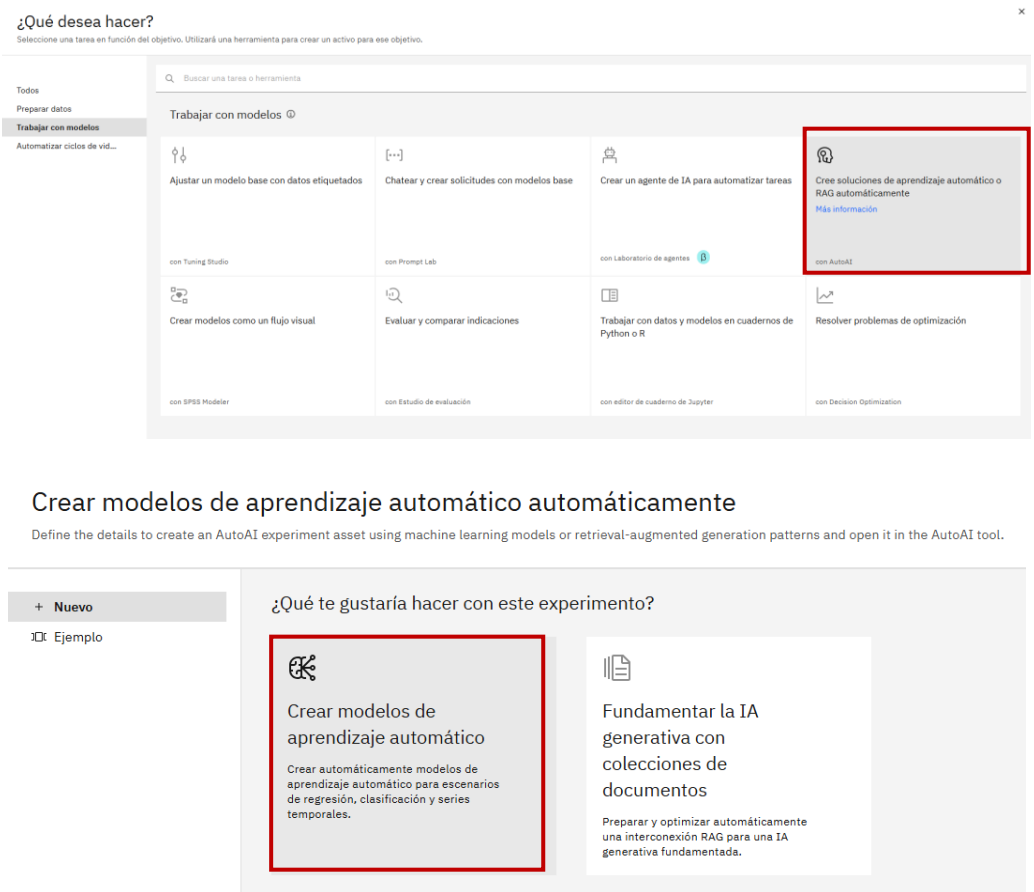


Figura 26. Elección de crear un modelo de aprendizaje automático con los datos cargados

Para poder crear el modelo, hay que asociar un servicio que sea capaz de realizar esa tarea. Se asocia un servicio predeterminado llamado *WatsonMachineLearning* que se encuentra en Sídney, la misma ubicación donde hemos creado la instancia.

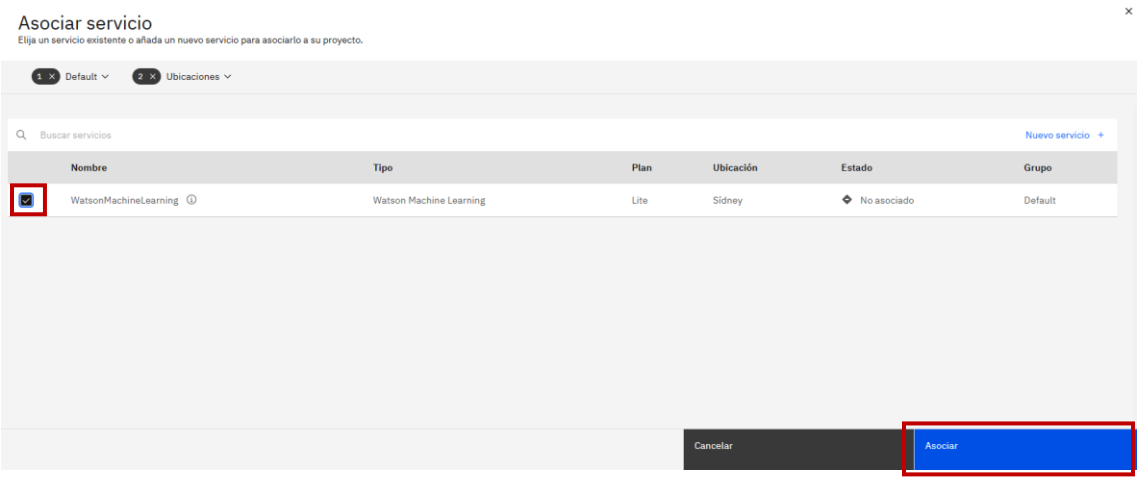


Figura 27. Asociación del servicio de WatsonMachineLearning en Watson.x

El siguiente paso es crear el modelo. Nuestro modelo lo llamaremos Proyecto_2_v1.0. Este nombre permitirá hacer un seguimiento de mejoras y cambios en el modelo gracias a poder cambiar la versión del nombre.

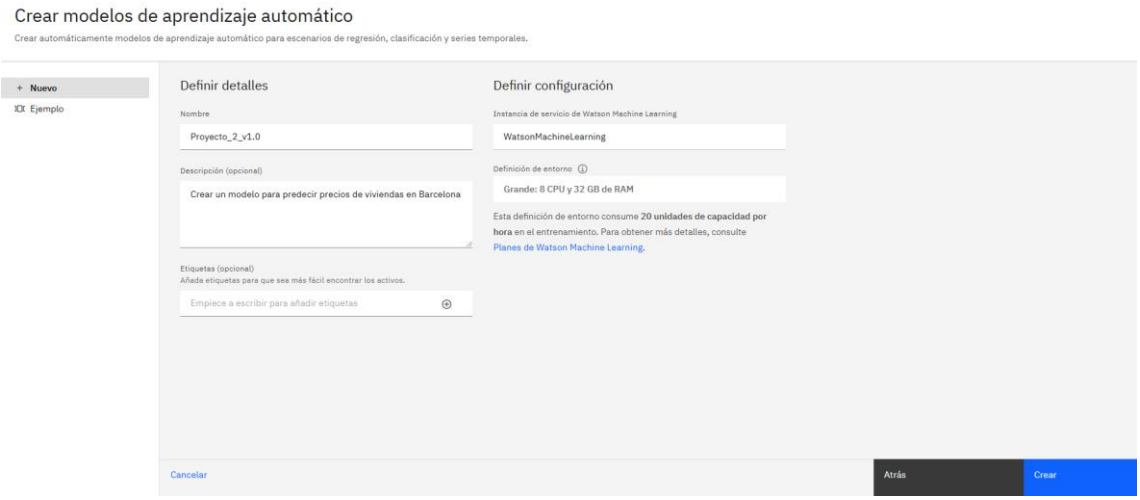


Figura 28. Creación del modelo en Watson.x

Se asigna el archivo CSV cargado previamente como origen de datos, se marca que no queremos un análisis de serie temporal y se indica que el objetivo de predicción debe ser la columna *Price*. Automáticamente el sistema asigna el tipo de predicción *Regresión* optimizado para *RMSE & tiempo de ejecución*.

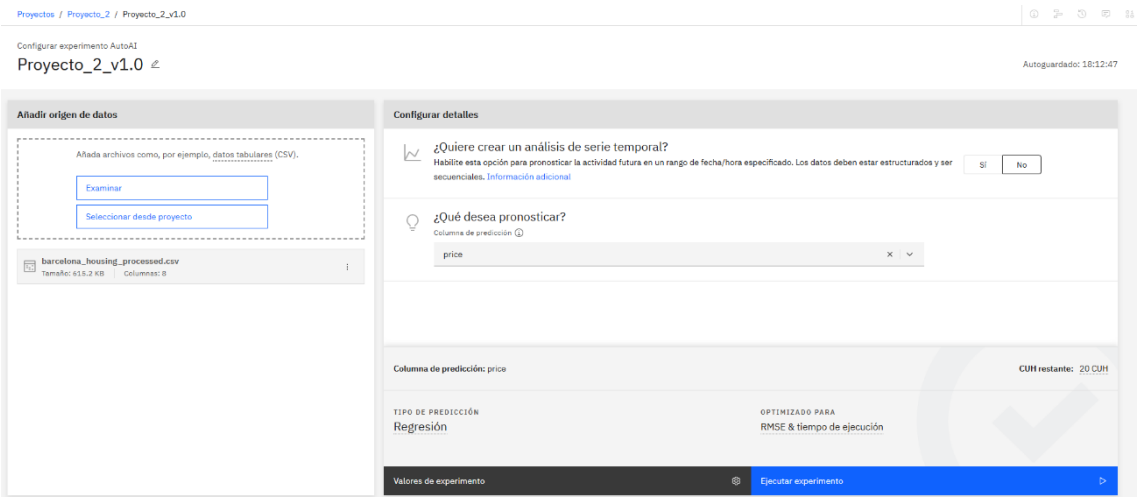


Figura 29. Asignación del archivo csv en Watson.x

5. Resultados de los modelos

5.1. BigML

Para comparar los resultados de los modelos escogidos en BigML, se escogerá un piso con unas características determinadas y se observarán las predicciones de todos ellos para todos los distintos barrios. Para nuestro caso, escogeremos un apartamento de 3 habitaciones con 2 baños, con ascensor y sin terraza. Los gráficos adjuntos muestran el precio de la vivienda en función de los metros cuadrados (eje X) y del barrio (eje Y). Se muestran los mapas de calor del precio para los distintos modelos:

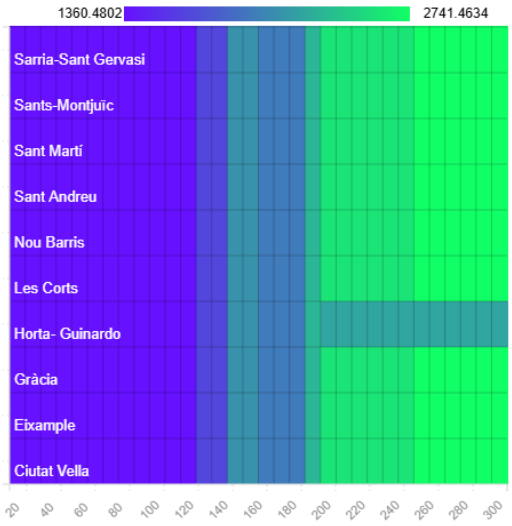


Figura 30. Resultados obtenidos por el modelo Random Forest

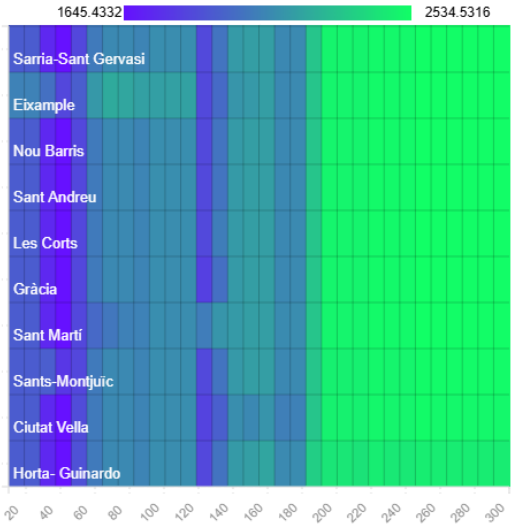


Figura 31. Resultados obtenidos por el modelo Ensemble

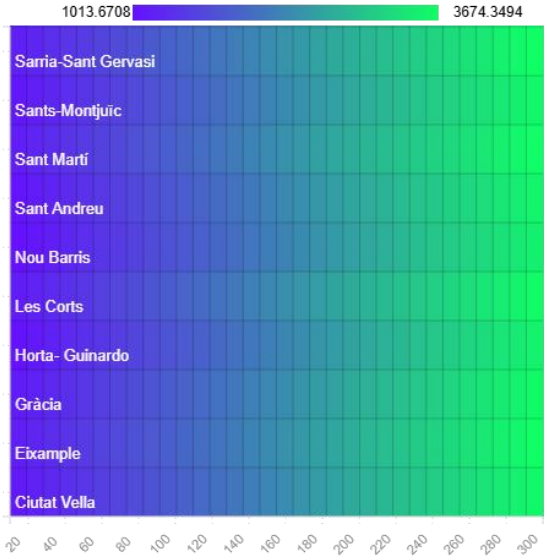


Figura 32. Resultados obtenidos por el modelo Linear Regression

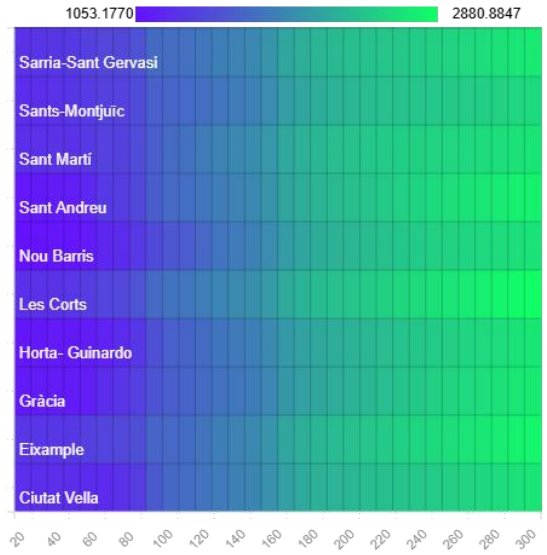


Figura 33. Resultados obtenidos por el modelo DeepNet

Tabla 1. Resumen de precios mínimos y máximos por cada modelo creado en BigML

Modelo	Precio mínimo	Precio máximo
Random Forest	1360 €	2741 €
Ensemble	1645 €	2535 €
Linear Regression	1014 €	3674 €
DeepNet	1053 €	2880 €

La tabla muestra una comparación de los precios mínimos y máximos estimados por diferentes modelos de predicción.

La diferencia en los rangos de precios estimados por los modelos puede deberse a varios factores relacionados con la forma en que cada modelo maneja los datos y los patrones en la información. Aquí hay algunas posibles explicaciones:

Dispersión y robustez del modelo

- Linear Regression muestra el rango más amplio (1014 € - 3674 €), lo que sugiere que el modelo puede estar capturando relaciones lineales pero no necesariamente las no lineales o interacciones complejas entre variables. Esto podría explicar la mayor dispersión en las predicciones.
- Ensemble, en cambio, tiene el rango más ajustado (1645 € - 2535 €), lo que indica que la combinación de múltiples modelos ayuda a estabilizar la predicción, reduciendo la variabilidad.

Capacidad para capturar no linealidades

- Random Forest y DeepNet pueden capturar relaciones no lineales mejor que Linear Regression, lo que podría hacer que sus predicciones sean más representativas de los datos reales. Sin embargo, DeepNet aún presenta una mayor dispersión (1053 € - 2880 €) en comparación con Random Forest (1360 € - 2741 €), posiblemente debido a la sensibilidad de redes neuronales a la cantidad y calidad de los datos disponibles.

Efecto de outliers y ruido en los datos

- Linear Regression es más sensible a valores atípicos, lo que podría explicar por qué su rango de predicción es tan amplio. Unos pocos valores extremos en los datos de entrenamiento pueden influir significativamente en la predicción del modelo.
- Modelos basados en árboles (como Random Forest) suelen ser más robustos a outliers, lo que ayuda a reducir la variabilidad en las predicciones.

Sobreajuste vs. Generalización

- DeepNet, a pesar de ser un modelo complejo, puede estar sobreajustando los datos de entrenamiento, lo que lleva a una mayor dispersión en las predicciones.
- Ensemble logra un mejor equilibrio al combinar modelos, reduciendo el sobreajuste y proporcionando predicciones más estables.

Conclusión

El modelo Linear Regression presenta la mayor variabilidad debido a su simplicidad y sensibilidad a valores extremos. Ensemble ofrece predicciones más consistentes gracias a la combinación de modelos, mientras que Random Forest y DeepNet capturan mejor las relaciones no lineales pero con diferentes niveles de estabilidad.

5.2. Vertex.ai

En Vertex.ai se han comparado dos modelos con criterios de optimización distintos: uno minimizando el RMSE y el otro minimizando el MAE. Se presentan recortes de la comparativa de los resultados de ambos modelos y de la diferencia de importancia de los atributos entre ellos:

Versión del modelo	Evaluación	MAE	MAPE	RMSE	RMSLE	R^2
regresion-proyecto-2-impelia > Versión 1	untitled_1214977826490190968	219.321	17.798	323.792	0.23	0.576
regresion-impelia-proyecto-2-MAE > Versión 1	untitled_7281355709692816704	213.667	16.364	333.746	0.232	0.564

Figura 34. Comparación de resultados obtenidos mediante optimización por RMSE y MAE

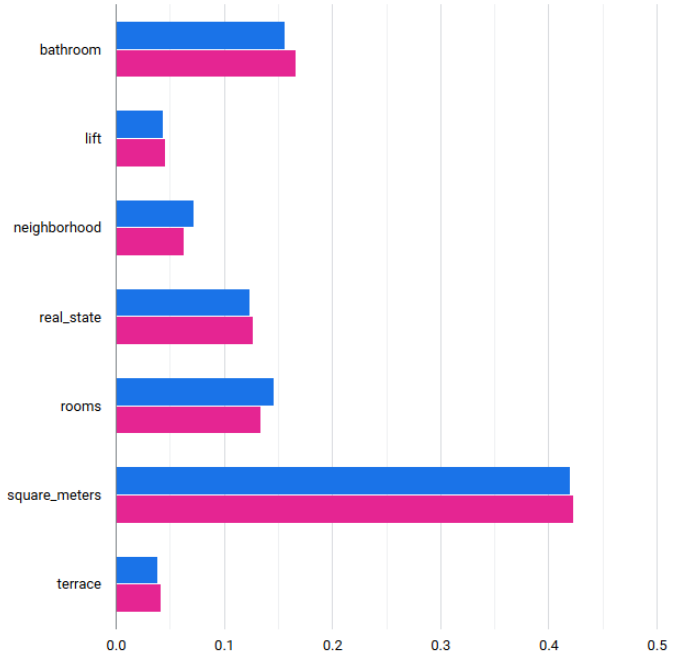


Figura 35. Comparación de la diferencia en importancias de cada atributo según el ajuste

Las pequeñas diferencias entre ambos modelos pueden deberse a distintos motivos. Algunos de los más importantes se listan a continuación:

Optimización y su impacto en las métricas

- El modelo optimizado con RMSE tiene un menor RMSE (323.792 vs. 333.746), lo que era esperado ya que este modelo fue ajustado para minimizar esta métrica.
- El modelo optimizado con MAE tiene un menor MAE (213.667 vs. 219.321), lo cual también es coherente con su criterio de optimización.

Diferencias en los errores

- MAE: Indica el error promedio absoluto. El modelo optimizado con MAE tiene un menor valor, lo que sugiere que, en promedio, sus predicciones están más cerca de los valores reales.
- RMSE: Penaliza más los errores grandes. El modelo optimizado con RMSE tiene un menor valor aquí, lo que indica que es mejor reduciendo grandes desviaciones en los datos.

R^2 y ajuste general

- El modelo optimizado con RMSE tiene un mejor R^2 (0.576 vs. 0.564), lo que sugiere que explica mejor la variabilidad en los datos en comparación con el modelo optimizado con MAE.
- Sin embargo, la diferencia en R^2 no es muy grande, por lo que ambos modelos tienen un desempeño similar en términos generales.

I**mpacto en la interpretación y aplicación**

- Si se busca estabilidad y predicciones más consistentes, el modelo optimizado con MAE puede ser preferible, ya que minimiza los errores absolutos en promedio.
- Si es crítico reducir grandes errores y mejorar el ajuste global, el modelo optimizado con RMSE es mejor, ya que penaliza más los errores altos y ofrece un R^2 ligeramente superior.

Conclusión:

- El modelo optimizado con MAE es mejor si se quiere minimizar el error medio absoluto, lo cual es útil en situaciones donde errores grandes y pequeños tienen la misma importancia.
- El modelo optimizado con RMSE es mejor si se quiere evitar grandes desviaciones, ya que castiga más los errores grandes y mejora la capacidad explicativa del modelo.
- La elección final dependerá del contexto de uso: Si hay valores atípicos que se quieren minimizar, RMSE es mejor. Si se busca precisión promedio sin importar valores extremos, MAE es la mejor opción.

5.3. Watson.x

Se muestran capturas de pantalla de los resultados obtenidos del modelo. Los resultados que se muestran son el mapa de relaciones, la tabla de clasificación de interconexiones y el gráfico de métricas.

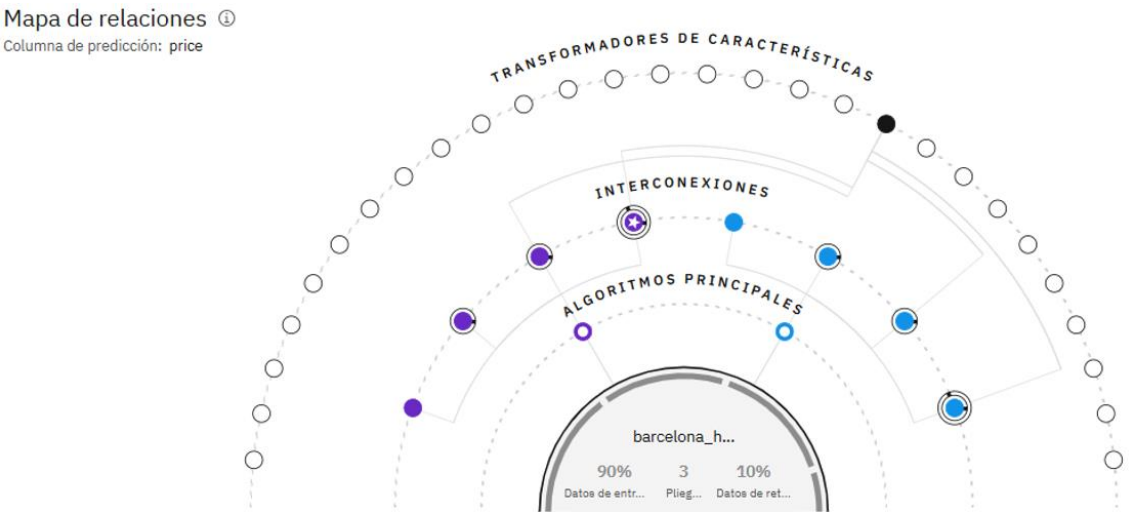


Figura 36. Mapa de relaciones obtenia por Watson.x

	Clasificación	Nombre	Algoritmo	RMSE (Optimizado) Validación Cruzada	Mejoras	Tiempo de creación
★	1	4 interconexiones	Regresor de bosque aleatorio de instantáneas	343.651	HPO-1 FE HPO-2	00:00:33
	2	3 interconexiones	Regresor de bosque aleatorio de instantáneas	343.651	HPO-1 FE	00:00:25
	3	2 interconexiones	Regresor de bosque aleatorio de instantáneas	344.709	HPO-1	00:00:06
	4	1 interconexiones	Regresor de bosque aleatorio de instantáneas	344.709	*Ninguna*	00:00:02
	5	8 interconexiones	Regresor de árbol de decisiones	358.274	HPO-1 FE HPO-2	00:00:29
	6	7 interconexiones	Regresor de árbol de decisiones	359.580	HPO-1 FE	00:00:25
	7	6 interconexiones	Regresor de árbol de decisiones	362.946	HPO-1	00:00:03
	8	5 interconexiones	Regresor de árbol de decisiones	376.624	*Ninguna*	00:00:01

Figura 37. Tabla de interconexiones del modelo de Watson.x

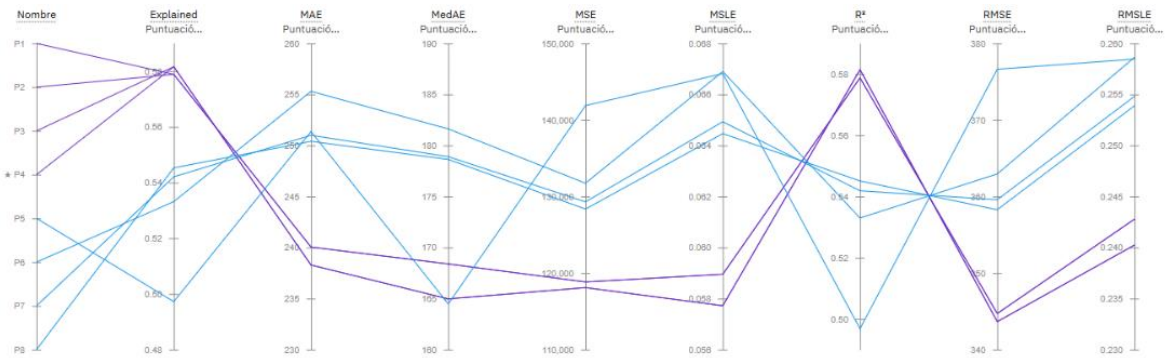


Figura 38. Gráfico de métricas del modelo de Watson.x

Analizando la tabla de métricas, se pueden observar varias líneas que representan diferentes modelos de regresión (P1 hasta P8) evaluados con diversas métricas de rendimiento:

- Varianza explicada: mide cuánta variabilidad de los datos es capturada por el modelo.
- **MAE** (Error Absoluto Medio): promedio de los errores absolutos.
- **MedAE** (Error Absoluto Mediano): la mediana de los errores absolutos.
- **MSE** (Error Cuadrático Medio): promedio de los errores al cuadrado.
- **MSLE** (Error Cuadrático Logarítmico Medio): similar al MSE, pero usando logaritmos.
- **R^2**: Coeficiente de determinación.
- **RMSE** (Raíz del Error Cuadrático Medio): raíz cuadrada del MSE.
- **RMSLE** (Raíz del Error Cuadrático Logarítmico Medio): Raíz cuadrada del MSLE.

Analizamos los resultados de los 8 modelos:

Modelos más destacados

P1 (línea púrpura) muestra el mejor rendimiento en varias métricas. Tiene la varianza explicada más alta (cercana a 1), un R^2 elevado, y valores bajos en métricas de error (MSE, MSLE). P3 y P4: También muestran un buen rendimiento en general.

Modelos con menor rendimiento

P8 (línea azul clara) presenta valores más altos en métricas de error y valores más bajos en varianza explicada y R^2. P7 también muestra rendimiento inferior en la mayoría de las métricas.

Comportamiento inconsistente

Se observa que algunos modelos tienen buen rendimiento en ciertas métricas, pero peor en otras, lo que podría indicar diferentes fortalezas según el tipo de error que se quiera minimizar.

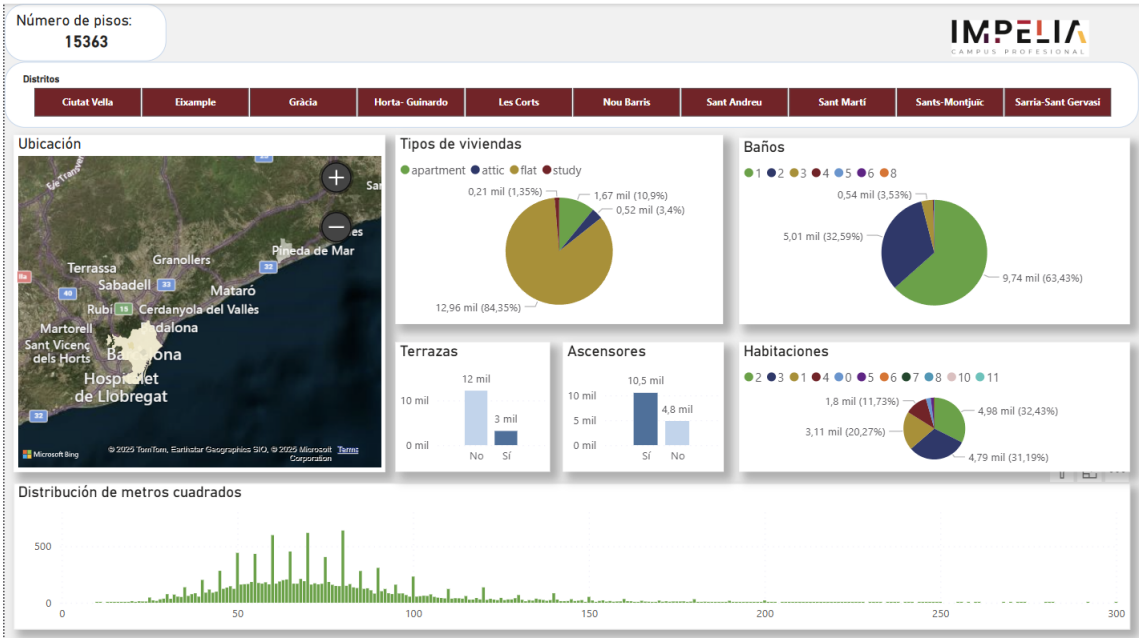
Conclusión

P1 parece ser el modelo con mejor rendimiento general, con la mejor combinación de alta varianza explicada, alto R^2 y bajos valores de error.

Hay compensaciones evidentes entre los diferentes modelos porque algunos tienen mejor MAE, pero peor MSE, lo que indica que responden de manera diferente a errores grandes versus pequeños.

6. PowerBI

A modo de introducción, se diseñaron dos dashboards en función del distrito. El primero viene a ser el estudio, columna por columna donde informa la distribución de variables. El segundo es la variación del precio de alquiler y las principales columnas que dependen de ello.



cdsv

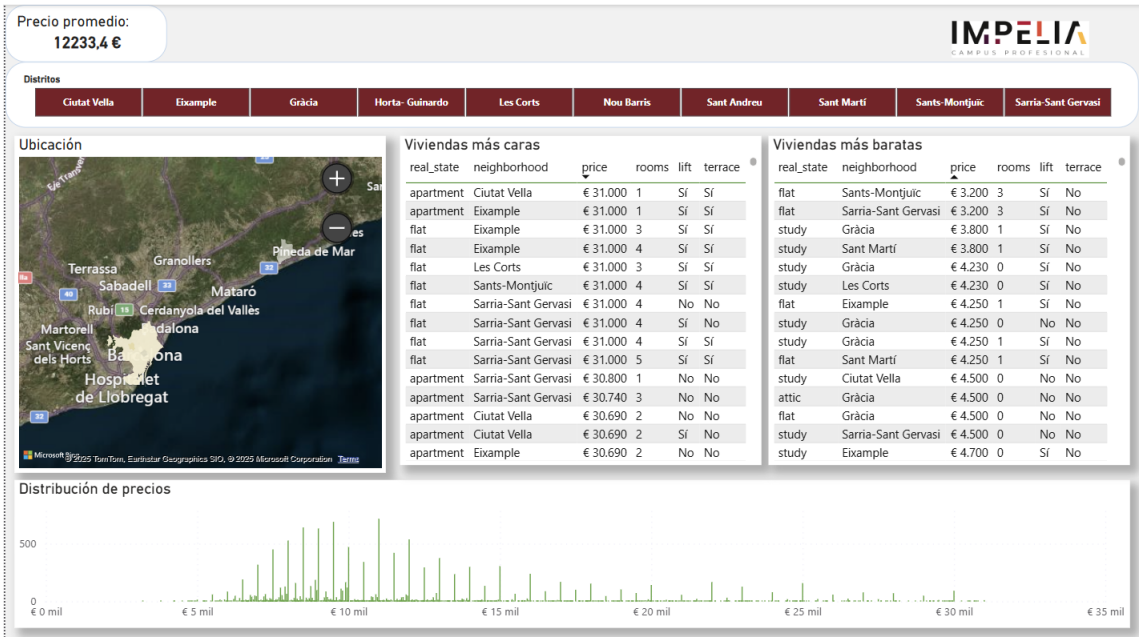


Figura 39. Minería de datos de la tabla procesada con WindSurf

7. Conclusiones

Después de realizar este proyecto, como equipo hemos llegado a las siguientes conclusiones:

1. Se comprobó que, mediante la conjunción de los métodos tradicionales de programación junto con las herramientas de IA como WindSurf, realizar el preprocesamiento de los datos ha sido una tarea mucho más rápida, sencilla y asistida.
2. Se han utilizado tres herramientas No-Code para la generación de modelos de aprendizaje automático: BigML, Vertex.ai (Google) y Watson.x (IBM). Todas ellas son de pago, y se ha sacado provecho de los periodos de prueba gratuitos que ofrecen para probar las aplicaciones.
3. Las R^2 obtenidas en los modelos de Vertex.ai y Watson.x no son valores demasiado altos (0.56 y 0.57, respectivamente), lo que indicaría que el modelo no representa con fidelidad los datos. Viendo que ambas plataformas han dado una R^2 similar, es probable que haya algún error en el criterio del preprocesamiento de datos que el equipo no haya sabido resolver.