
Análisis Predictivo de Precios y Segmentación de Usuarios en Airbnb

Un enfoque desde la Ciencia de Datos

ÍNDICE


- 1 ¿De qué trata la investigación?
- 2 Preguntas de investigación
- 3 Metodología y técnicas empleadas
- 4 Información relevante
- 5 Resultados



¿De qué trata la investigación?



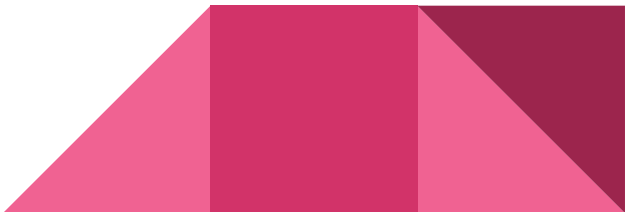
¿De qué trata la investigación?

- Predecir precios Airbnb aplicando ciencia de datos y machine learning en ciudades españolas.
 - Determinar factores clave en precios: ubicación, características y servicios ofrecidos.
 - Extraer insights valiosos mediante técnicas estadísticas para procesar y mejorar datos.
 - Clasificar experiencias de usuarios analizando reseñas con procesamiento del lenguaje natural.
 - Análisis temporal del sentimiento, extracción de temas y palabras clave en cada cluster de opiniones.
- 

Preguntas de investigación



Preguntas de investigación

- ¿Qué variables explicativas presentan un mayor impacto en la determinación del precio de los alojamientos en Airbnb?
 - ¿Existen patrones latentes en las reseñas de los usuarios que permitan una segmentación significativa basada en su comportamiento y opiniones?
 - ¿Es factible construir modelos predictivos que permitan estimar con precisión el precio de un alojamiento a partir de sus características estructurales y descriptivas?
 - ¿Cómo difiere la percepción del servicio entre distintos grupos de usuarios identificados mediante técnicas de clustering?
- 

Metodología y técnicas empleadas



Metodología CRISP-DM (Análisis y Predicción de precios)

Comprensión del Negocio: Definición de objetivos para el modelo predictivo.

Comprensión de los Datos: Recolección y exploración preliminar aplicando técnicas estadísticas.

Preparación de los Datos: Limpieza, selección de variables y transformación para el modelado.

Modelado: Construcción y optimización de modelos predictivos de precios.

Evaluación: Valoración de resultados respecto a objetivos iniciales.

Despliegue: Implementación del modelo en dashboard interactivo.



Técnicas usadas I (Análisis y Predicción de precios)

Test de Little: Evalúa aleatoriedad de datos faltantes.

Algoritmo MICE: Imputación de valores faltantes mediante ecuaciones encadenadas.

Rango Intercuartílico: Identifica valores atípicos fuera de $1.5 \times \text{IQR}$.

Shapiro-Wilk: Prueba de normalidad en distribuciones.

Kruskal-Wallis: Compara diferencias entre tres o más grupos independientes.

Correlación de Pearson/Spearman: Mide relaciones entre variables numéricas.

Mann-Whitney U: Compara dos grupos independientes sin normalidad.

Intervalo de confianza: Estima rango probable de la media poblacional.

DBSCAN: Clustering espacial basado en densidad.



Técnicas usadas II (Análisis y Predicción de precios)

Índice de Moran: Mide autocorrelación espacial.

Chi-cuadrado: Analiza relaciones entre variables categóricas.

Análisis de Correspondencias: Visualiza relaciones entre categorías.

Test de Levene: Evalúa homogeneidad de varianzas.

PCA: Reduce dimensionalidad preservando información relevante.

Fisher: Prueba asociación en muestras pequeñas.

Cohen's d: Cuantifica tamaño del efecto entre grupos.

Friedman: Compara mediciones repetidas.

LightGBM/Random Forest/XGBoost: Modelos predictivos.



Metodología PLN (Clasificación de usuarios)

Perfiles de usuarios: Se generan características por usuario: número de reseñas, promedio de sentimiento, días activos y longitud promedio de reseñas.

Selección de características: Se usan número de reseñas, sentimiento promedio, días activos y longitud de reseñas para segmentar.

Normalización: Características se normalizan con StandardScaler para igualar escalas.

Reducción dimensional: PCA reduce datos a 2D para visualizar segmentos.

Segmentación con DBSCAN: DBSCAN agrupa usuarios similares



Técnicas usadas (Clasificación de usuarios)

Agregación de datos: Generación de características por usuario (número de reseñas, sentimiento promedio, días activos, longitud de reseñas) mediante agrupación.

Normalización: Aplicación de StandardScaler para estandarizar las características y evitar sesgos por diferencias de escala.

Reducción dimensional: Uso de PCA para proyectar datos en un espacio 2D para visualización.

Clustering con DBSCAN: Empleo de DBSCAN para identificar segmentos de usuarios y outliers.

Visualización: Creación de gráficos de dispersión para mostrar segmentos y outliers en el espacio PCA.

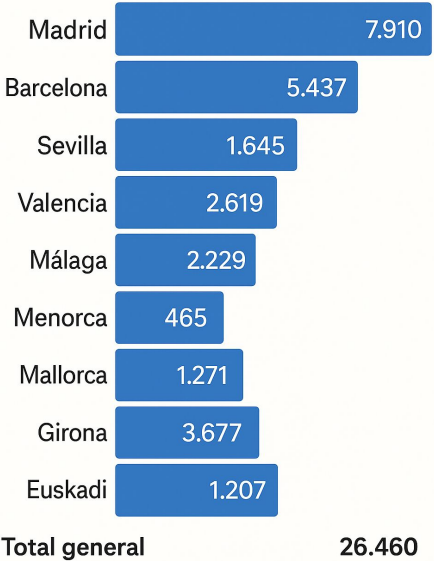


Información relevante



Conjunto inmuebles (Datos relevantes)

Registros de Airbnb por ciudad en 2024



Conjunto de datos entrenamiento

(85%): 22491 registros

Conjunto de datos prueba

(15%): 3969 registros

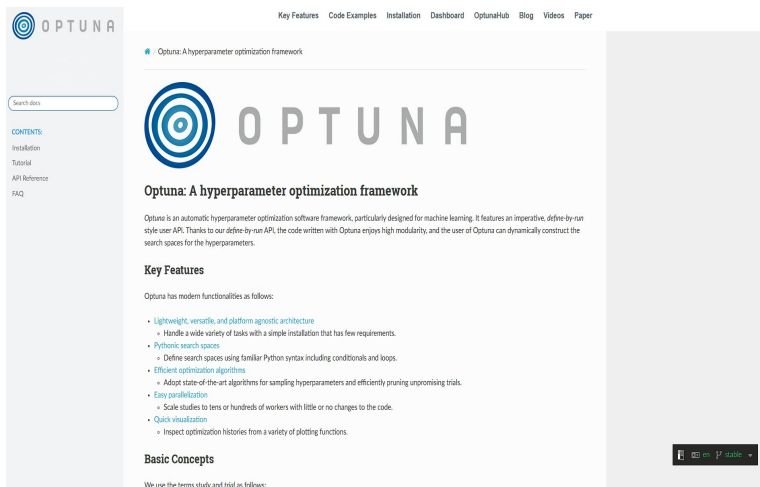
Variables relevantes: tipo de propiedad, tipo de habitación, capacidad de huéspedes, distancia al centro, total de comodidades, noche mínima de estancia

Estadísticas descriptivas variable objetivo en conjunto de prueba

- Media: 66.11
- Mínimo: 10
- Máximo: 100
- Desviación Estándar: 23.02

Conjunto inmuebles (Optimización de hiperparámetros)

Optuna



The screenshot shows the Optuna website homepage. At the top, there's a navigation bar with links: Key Features, Code Examples, Installation, Dashboard, OptunaHub, Blog, Videos, and Paper. Below this is a search bar and a 'CONTENTS' sidebar with links to Installation, Tutorial, API Reference, and FAQ. The main content area features the Optuna logo (a blue target icon) and the text 'OPTUNA'. Below the logo, it says 'Optuna: A hyperparameter optimization framework'. A paragraph describes Optuna as an automatic hyperparameter optimization software framework for machine learning. Under 'Key Features', it lists: Lightweight, versatile, and platform agnostic architecture; Pythonic search spaces; Efficient optimization algorithms; Easy parallelization; and Quick visualization. The 'Basic Concepts' section starts with 'We use the terms study and trial as follows:'.

Key Features

- Lightweight, versatile, and platform agnostic architecture
 - Handle a wide variety of tasks with a simple installation that has few requirements.
- Pythonic search spaces
 - Define search spaces using familiar Python syntax including conditionals and loops.
- Efficient optimization algorithms
 - Adopt state-of-the-art algorithms for sampling hyperparameters and efficiently pruning unpromising trials.
- Easy parallelization
 - Scale studies to tens or hundreds of workers with little or no changes to the code.
- Quick visualization
 - Inspect optimization histories from a variety of plotting functions.

Basic Concepts

We use the terms study and trial as follows:

MÉTRICAS DE RENDIMIENTO EVALUADAS

ERROR ABSOLUTO MEDIO


RAÍZ DEL ERROR CUADRÁTICO MEDIO

COEFICIENTE DE DETERMINACIÓN

ERROR PORCENTUAL MEDIO ABSOLUTO

Conjunto inmuebles (Modelos ML entrenados)

- Máquina de Potenciación del Gradiente Ligero
- Bosque Aleatorio
- Máquina de Potenciación del Gradiente Extremo



• XGBoost Documentation

View page source

Search docs

Installation Guide

Building From Source

Get Started with XGBoost

XGBoost Tutorials

Frequently Asked Questions

GPU Support

XGBoost Parameters

Prediction

Tree Methods

Python Package

R Package

JVM Package

Ruby Package

Swift Package

Julia Package

C Package

C++ Interface

CLI Interface

Contribute to XGBoost


Release Notes

XGBoost Documentation

XGBoost is an optimized distributed gradient boosting library designed to be highly **efficient**, **flexible** and **portable**. It implements machine learning algorithms under the **Gradient Boosting** framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

Contents

- Installation Guide
- Building From Source
- Get Started with XGBoost
- XGBoost Tutorials
 - Introduction to Boosted Trees
 - Introduction to Model IO
 - Learning to Rank
 - DART booster
 - Monotonic Constraints
 - Feature Interaction Constraints
 - Survival Analysis with Accelerated Failure Time
 - Categorical Data
 - Multiple Outputs
 - Random Forests(TM) in XGBoost
 - Distributed XGBoost on Kubernetes
 - Distributed XGBoost with XGBoost4J-Spark
 - Distributed XGBoost with XGBoost4J-Spark-GPU
 - Distributed XGBoost with Dask
 - Distributed XGBoost with PySpark
 - Distributed XGBoost with Ray
 - Using XGBoost External Memory Version
 - C API Tutorial
 - Text Input Format of DMATRIX
 - Notes on Parameter Tuning



• Welcome to LightGBM's documentation!

Contents

Installation Guide

Quick Start

Python Quick Start

Features

Experiments

Parameters

Parameters Tuning

C API

Python API

R API

Distributed Learning Guide

GPU Tutorial


Advanced Topics

FAQ

Development Guide

• Welcome to LightGBM's documentation!

View page source



Welcome to LightGBM's documentation!


LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel, distributed, and GPU learning.
- Capable of handling large-scale data.

For more details, please refer to [Features](#).

Contents:

- Installation Guide
- Quick Start
- Python Quick Start
- Features
- Experiments
- Parameters
- Parameters Tuning
- C API
- Python API
- R API
- Distributed Learning Guide
- GPU Tutorial
- Advanced Topics



Install

User Guide

API

Examples

Community

More

ExtraTreesClassifier

ExtraTreesRegressor

GradientBoostingClassifier

GradientBoostingRegressor

HistGradientBoostingClassifier

HistGradientBoostingRegressor

IsolationForest

RandomForestClassifier

RandomForestRegressor

RandomTreesEmbedding

StackingClassifier

StackingRegressor

VotingClassifier

VotingRegressor

sklearn.exceptions

sklearn.experimental

sklearn.feature_extraction

sklearn.feature_selection

sklearn.feature_extraction

sklearn.gaussian_process

sklearn.impute

sklearn.inspection

sklearn.isotonic

sklearn.kernel_approximation

sklearn.kernel_ridge

sklearn.linear_model

sklearn.manifold

sklearn.metrics

• API Reference

sklearn.ensemble

RandomForestRegressor

RandomForestRegressor

```
class sklearn.ensemble.RandomForestRegressor(n_estimators=100, *,
      criterion='squared_error', max_depth=None, min_samples_split=2, min_samples_leaf=1,
      min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None,
      min_impurity_decrease=0, bootstrap=True, oob_score=False, n_jobs=None,
      random_state=None, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None,
      monotonic_cst=None) [source]
```

A random forest regressor.

A random forest is a meta estimator that fits a number of decision tree regressors on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Trees in the forest use the best split strategy (i.e. equivalent to passing `splitter="best"`) to the underlying [DecisionTreeRegressor](#). The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

For a comparison between tree-based ensemble models see the example [Comparing Random Forests and Histogram Gradient Boosting models](#).

Read more in the [User Guide](#).

Parameters:

n_estimators : *int, default=100*
The number of trees in the forest.

Changed in version 0.22: The default value of `n_estimators` changed from 10 to 100 in 0.22.

criterion : {"squared_error", "absolute_error", "friedman_mse", "poisson"},
default="squared_error"
The function to measure the quality of a split. Supported criteria are "squared_error" for the mean

On this page

RandomForestRegressor

apply

decision_path

estimators_samples

feature_importances

fit

get_metadata_routing

get_params

predict

score

set_fit_request

set_params

set_score_request

Gallery examples

This Page

- [Show Source](#)

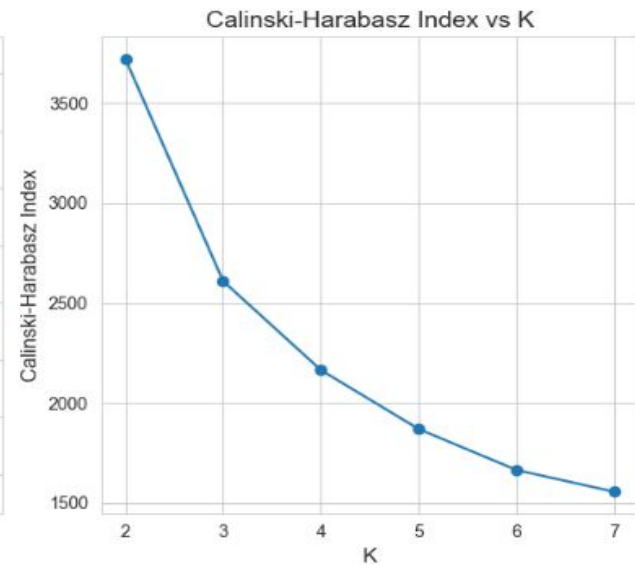
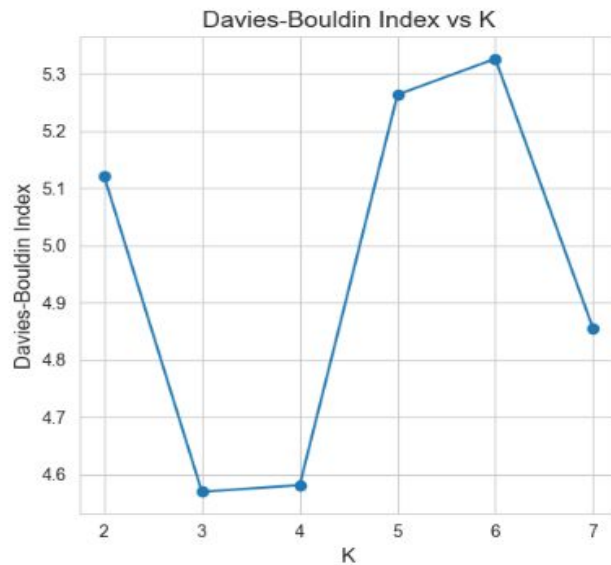
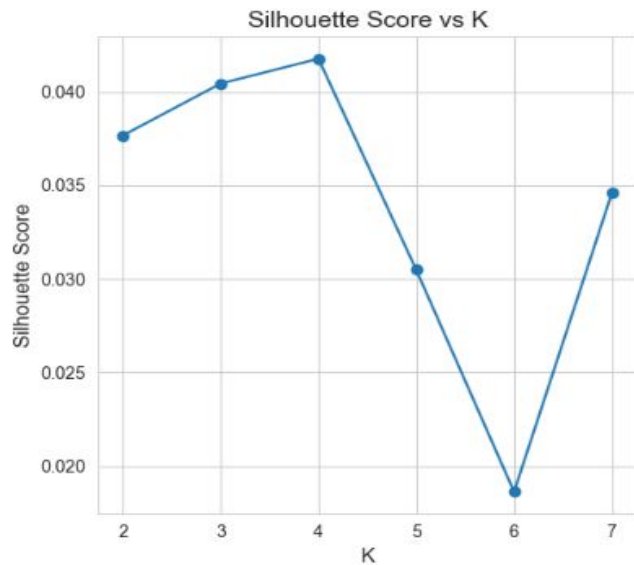
Conjunto reseñas (Datos relevantes)

Conjunto de datos : 50000 reseñas

Rango de fechas: 2011 - 2024

Localización reseñas: España (Madrid, Barcelona, Sevilla, Valencia, Menorca, Mallorca, Euskadi)

Número óptimo de clusters: 3

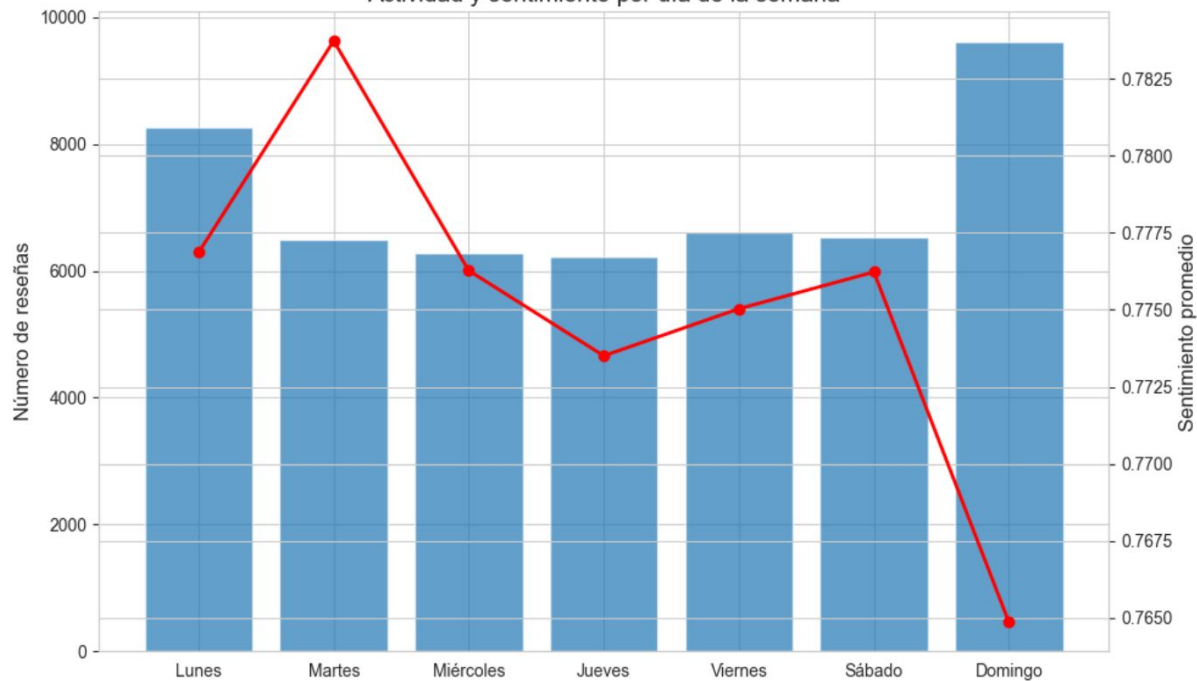


Resultados

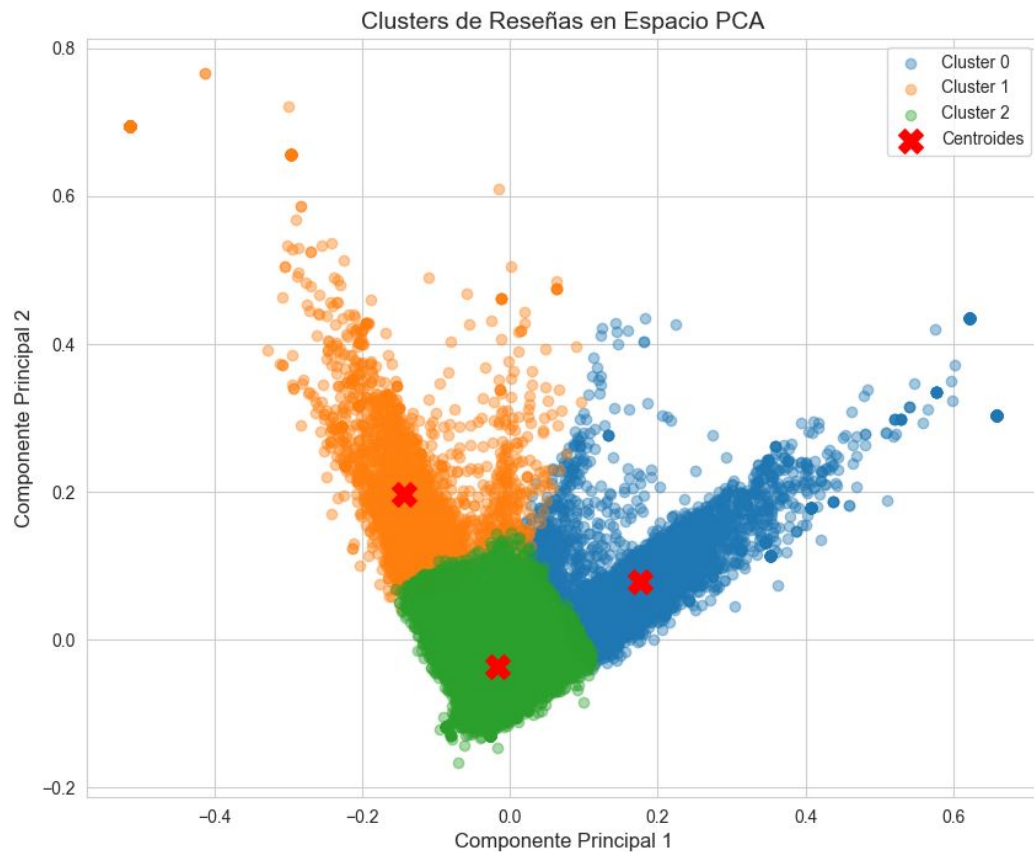
Clasificación de usuarios



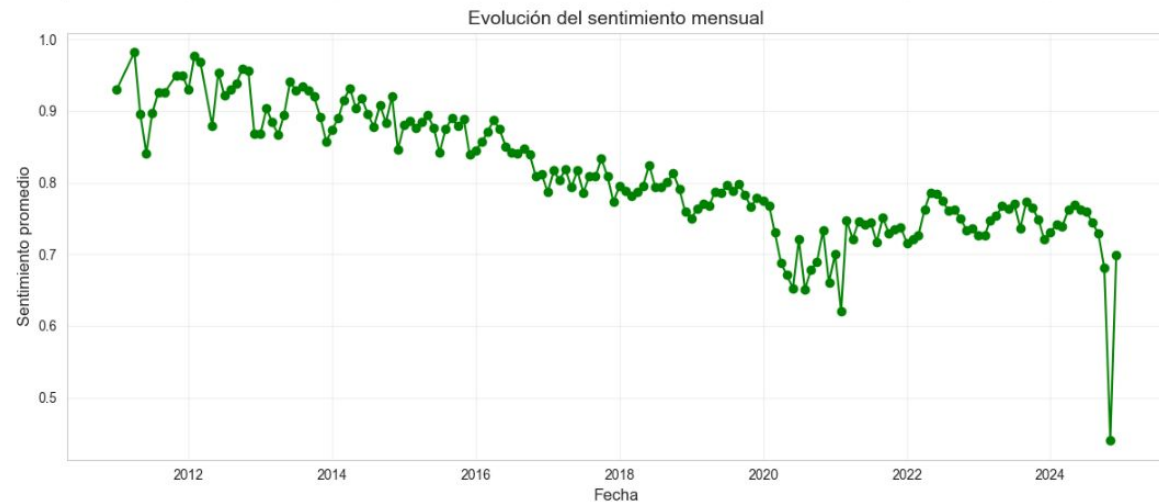
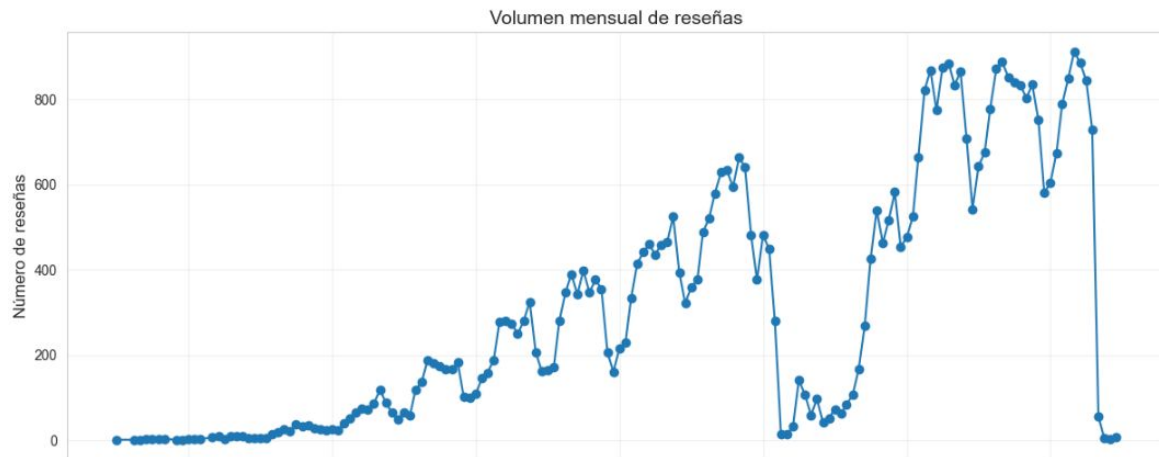
Actividad y sentimiento por día de la semana



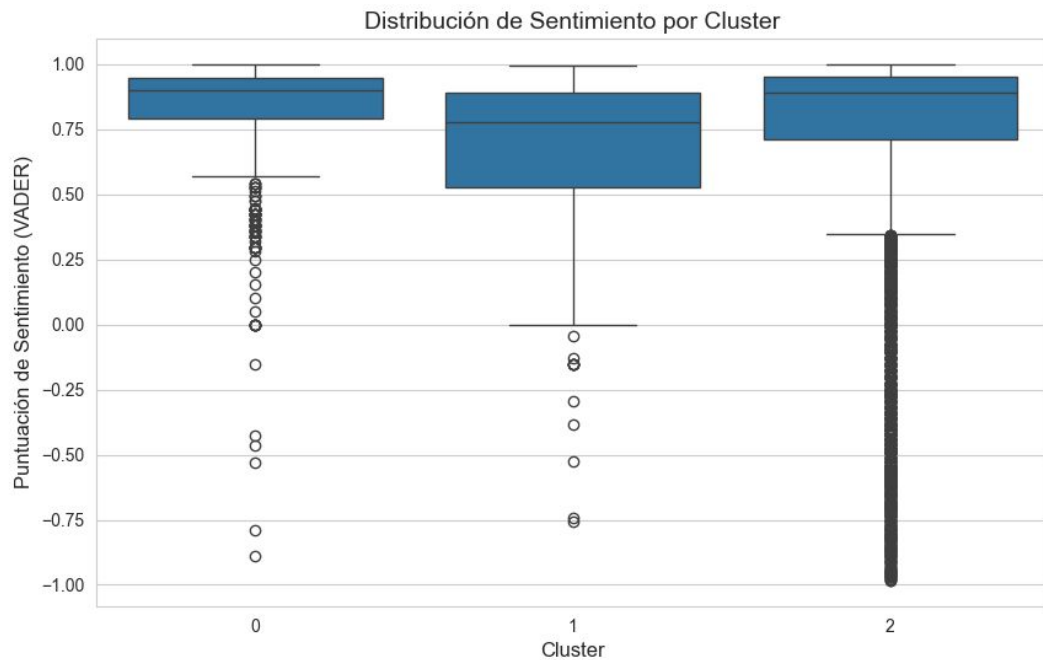
Actividad semanal



Clusters de
reseñas en
espacio PCA



Tendencias temporales



Sentimiento por
cluster

[illegible]

Palabras clave

Cluster 1

Palabras Clave del Cluster 1



Palabras clave Cluster 2



[illegible]

Palabras clave

Cluster 3



Temas principales detectados

```
},
"temas principales": {
  "Tema_1": "0.028*\"check\" + 0.021*\"help\" + 0.020*\"give\" + 0.017*\"time\" + 0.015*\"arrive\" + 0.012*\"leave\" + 0.011*\"arrival\" + 0.011*\"not\" + 0.011*\"question\" + 0.010*\"early\"",
  "Tema_2": "0.030*\"room\" + 0.018*\"not\" + 0.016*\"bed\" + 0.016*\"bathroom\" + 0.016*\"small\" + 0.013*\"kitchen\" + 0.013*\"night\" + 0.011*\"work\" + 0.010*\"shower\" + 0.010*\"bedroom\"",
  "Tema_3": "0.074*\"accommodation\" + 0.051*\"pleasant\" + 0.028*\"welcome\" + 0.025*\"locate\" + 0.016*\"description\" + 0.014*\"foot\" + 0.012*\"functional\" + 0.011*\"photo\" + 0.011*\"available\" + 0.011*\"practi",
  "Tema_4": "0.040*\"apartment\" + 0.038*\"great\" + 0.037*\"stay\" + 0.036*\"location\" + 0.027*\"place\" + 0.027*\"good\" + 0.025*\"clean\" + 0.020*\"recommend\" + 0.019*\"nice\" + 0.017*\"host\"",
  "Tema_5": "0.064*\"house\" + 0.034*\"attentive\" + 0.033*\"hostel\" + 0.020*\"attention\" + 0.018*\"department\" + 0.014*\"floor\" + 0.012*\"position\" + 0.011*\"doubt\" + 0.011*\"meter\" + 0.010*\"wide\"",
},
```

Resultados

Predicción y Análisis de precios



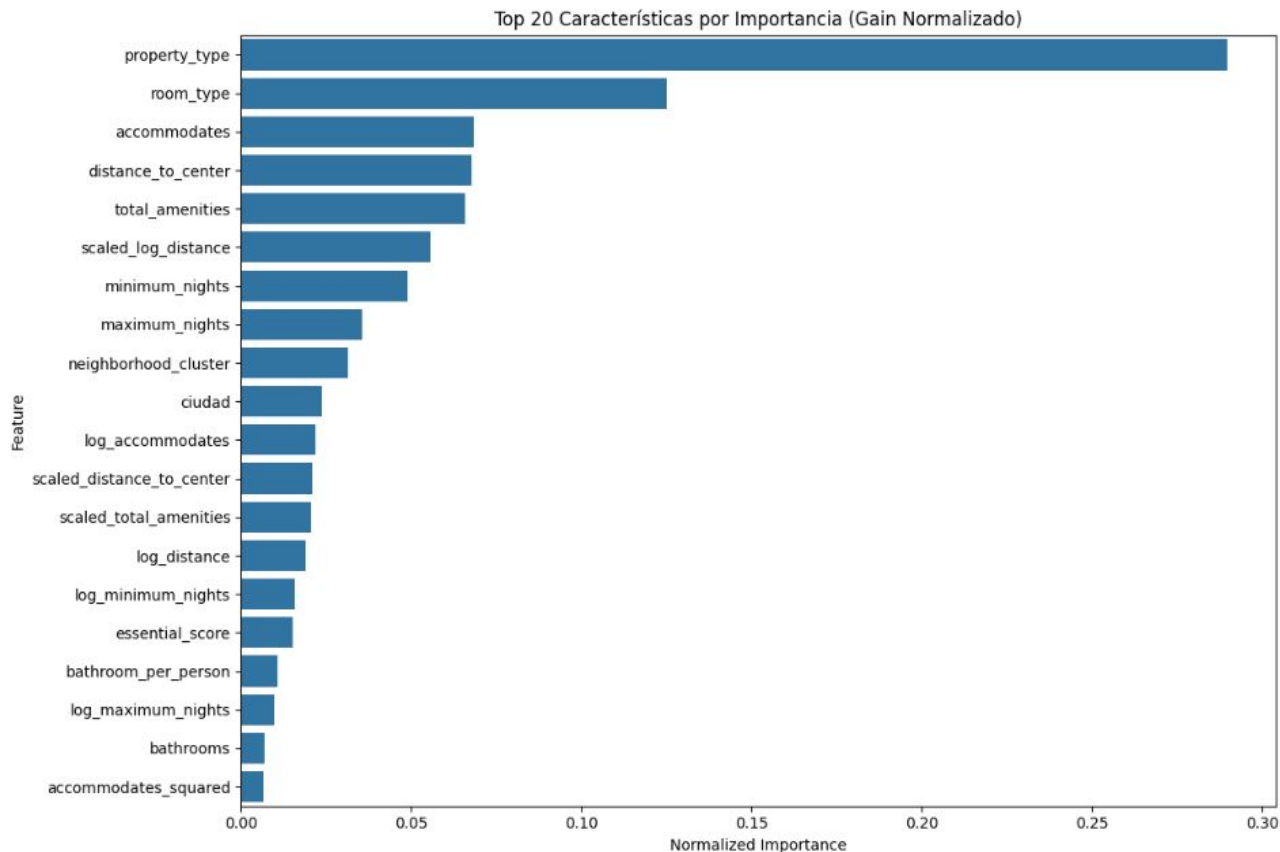
Métricas de rendimiento en la prueba

Tras completar la optimización de los hiperparámetros observamos que el modelo LGBM presenta el mejor rendimiento en la prueba

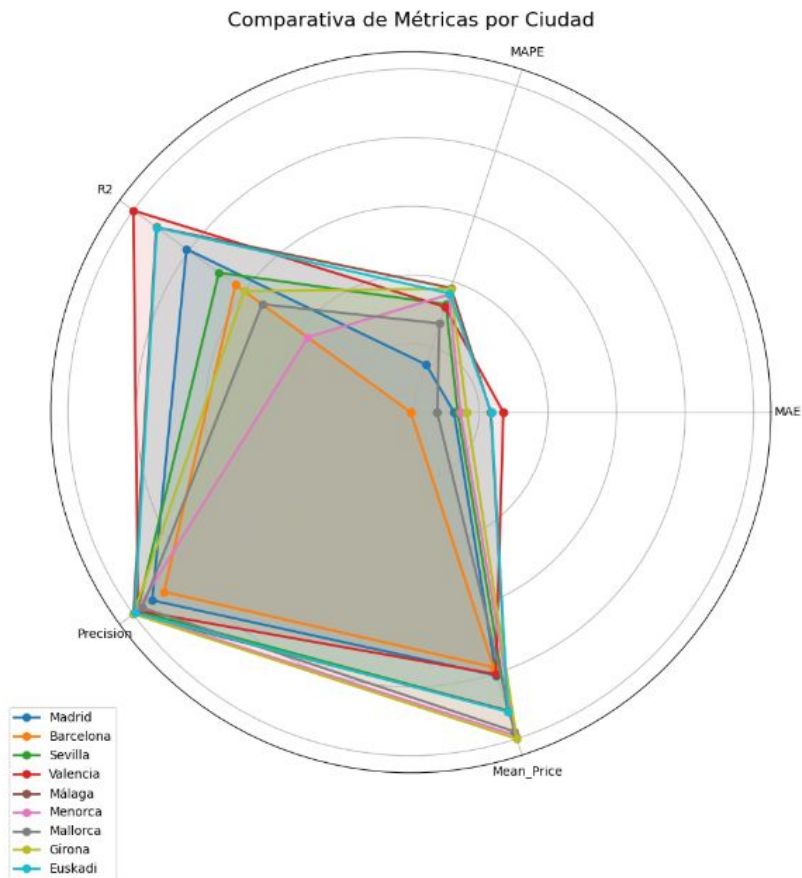
	LGBM	XGBM	RF
RMSE	14.69	14.90	16.39
R2	0.5938	0.582	13.09
MAE	11.14	11.47	0.4939
MAPE	19.22%	19.91%	23.10%



Características más influyentes en el mejor modelo



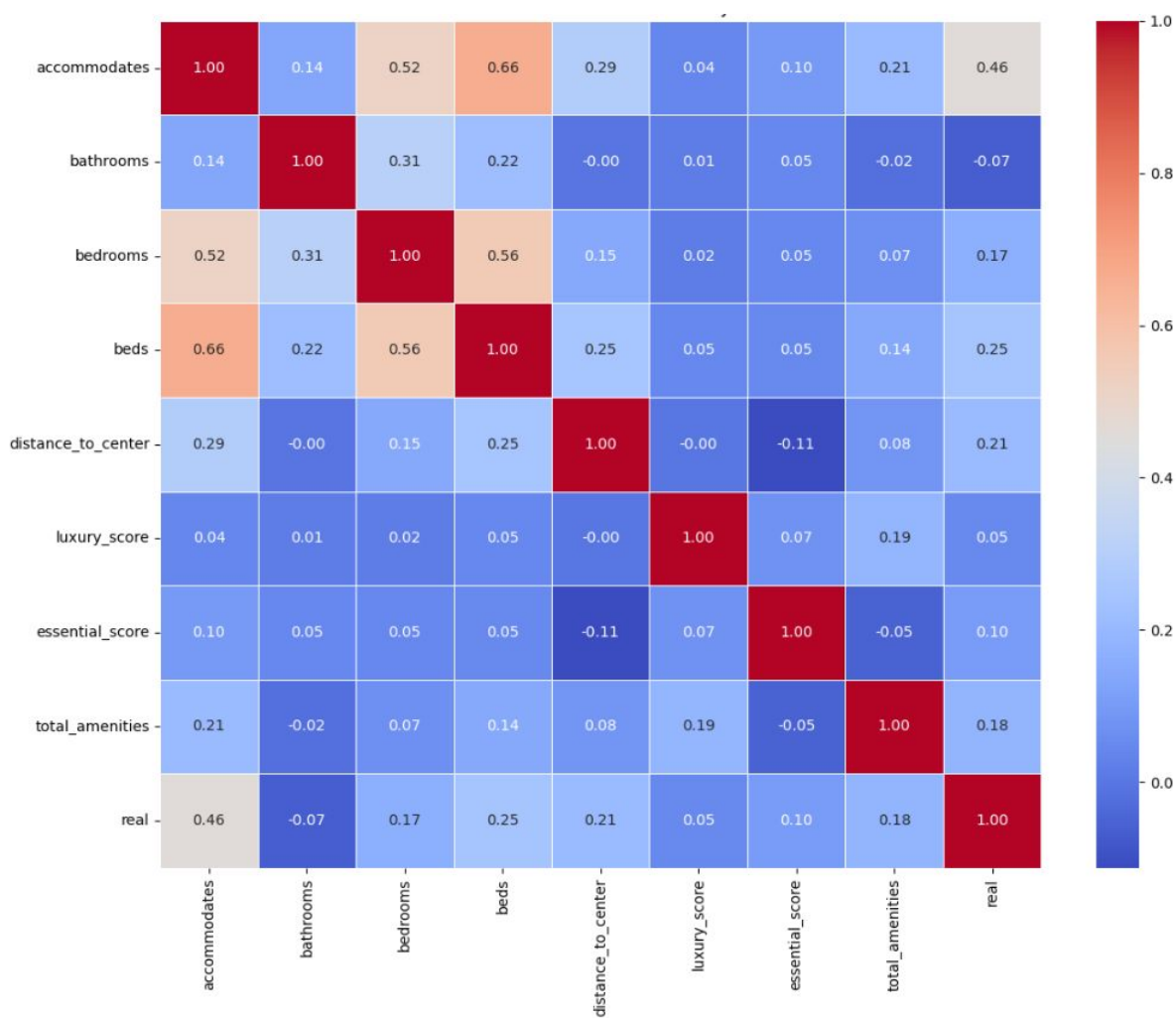
Métricas de rendimiento por ciudad (gráfica)



Métricas de rendimiento por ciudad (output textual)

```
1  ANÁLISIS DE ERRORES POR CIUDAD
2  =====
3
4  Métricas de error por ciudad:
5
6  Ciudad          MAE          MAPE (%)    R²
7  -----
8  Madrid          11.22          20.96      0.5917
9  Barcelona       12.83          24.54      0.4625
10 Sevilla         11.08          16.42      0.5064
11 Valencia        9.37           16.62      0.7320
12 Málaga          9.84           15.20      0.6715
13 Menorca         10.98          15.68      0.2723
14 Mallorca        11.85          17.87      0.3922
15 Girona          10.74          15.25      0.4393
16 Euskadi         9.82           15.61      0.6700
17
```

Ciudad	MAE	MAPE (%)	R²
Madrid	11.22	20.96	0.5917
Barcelona	12.83	24.54	0.4625
Sevilla	11.08	16.42	0.5064
Valencia	9.37	16.62	0.7320
Málaga	9.84	15.20	0.6715
Menorca	10.98	15.68	0.2723
Mallorca	11.85	17.87	0.3922
Girona	10.74	15.25	0.4393
Euskadi	9.82	15.61	0.6700



Correlación entre
características
clave y precio real

Ejemplos de predicciones

```
Muestra 1: Real: 94.00, Predicción: 74.25, Error: 19.75
Muestra 2: Real: 81.00, Predicción: 77.87, Error: 3.13
Muestra 3: Real: 24.00, Predicción: 45.37, Error: 21.37
Muestra 4: Real: 57.00, Predicción: 73.29, Error: 16.29
Muestra 5: Real: 90.00, Predicción: 80.50, Error: 9.50
Muestra 2: Real: 81.00, Predicción: 77.87, Error: 3.13
Muestra 3: Real: 24.00, Predicción: 45.37, Error: 21.37
0
Muestra 6: Real: 70.00, Predicción: 76.14, Error: 6.14
Muestra 7: Real: 68.00, Predicción: 68.86, Error: 0.86
Muestra 8: Real: 45.00, Predicción: 58.03, Error: 13.03
Muestra 9: Real: 94.00, Predicción: 82.90, Error: 11.10
Muestra 10: Real: 21.00, Predicción: 39.75, Error: 18.75
```

Conclusiones segmentación de usuarios



Conclusiones predicción de precios



Rendimiento general y evaluación estadística


LGBM demostró el mejor desempeño con un R^2 de 0.5938 y MAPE de 19.22%, superando consistentemente a XGBoost y Random Forest en todas las métricas evaluadas.

Las estadísticas descriptivas de la variable objetivo (rango 10-100, media 66.11, desviación típica 23.02), el RMSE de 14.69 representa el 16.3% del rango total y 63.8% de la desviación estándar.

MAE de 11.14 equivale al 29.3% del rango intercuartílico (48-86), un valor razonable frente a la variabilidad natural de los datos.

El modelo explica casi el 60% de la variabilidad en un mercado con alta influencia de factores subjetivos no cuantificables, lo que representa un rendimiento satisfactorio.

MAE es significativamente menor que la variabilidad natural (desviación estándar), indicando que el modelo aporta valor predictivo sustancial.



Factores determinantes del precio

Tipo de propiedad: factor más influyente (28.96%)

Tipo de habitación: segundo factor más relevante (12.52%)

Factores principales: Capacidad de alojamiento (6.83%), Distancia al centro (6.76%), Total de comodidades (6.59%)

