Data Analytics

Real Estate Market Analysis in Madrid

Halyna Abelchakova

July, 2024

Table of Contents

Introduction

Business Use Case and Project Overview

As a former real estate agent with over five years of experience in Ukraine, I have a deep-rooted passion for the property market. This background, combined with my current pursuit of a data analytics course, has led me to an exciting project focused on the Madrid real estate market. The inspiration for this analysis comes from a personal connection - a friend planning to relocate to Madrid - and aims to provide data-driven insights to support her decision-making process.

Goal and Objectives

The primary goal of this project is to conduct a comprehensive analysis of the Madrid real estate market. By leveraging data analytics techniques, we aim to uncover valuable insights that could also assist potential buyers in making informed decisions. This analysis will cover various aspects of the market, including price trends, neighborhood comparisons, and property feature evaluations.

Key objectives include:

1. **Data Collection and Preparation:** Analyze and clean the dataset collected from real estate advertisement websites in Madrid.
2. **Exploratory Data Analysis (EDA):** Conduct thorough statistical analysis and create visualizations to understand market trends, price distributions, and key factors influencing property values.
3. **Database Design and Management:** Develop a robust SQL database structure to efficiently store and manage the collected data, facilitating easy access and analysis.
4. **API Development:** Create an API to allow programmatic access to the database, enabling future applications and integrations.
5. **Predictive Modeling:** Develop and train machine learning models to predict property prices based on various features, providing a valuable tool for estimating

market values.

6. **Actionable Insights:** Generate meaningful insights and recommendations that can guide potential buyers, like my friend, in their property search and investment decisions.

Potential Insights:

1. **Price Distribution by Neighborhood:** Analyzing the distribution of property prices across different neighborhoods to reveal high-demand and high-value areas in Madrid. This will help potential buyers like my friend identify areas that fit their budget and preferences.
2. **Impact of Features on Property Prices:** Investigating how various features (such as number of rooms, bathrooms, floor area, and amenities like parking or storage rooms) influence property prices. This analysis will help buyers understand which features offer the best value for money and prioritize their requirements.
3. **Correlation Between Property Size and Price:** Studying the relationship between the size of the property (in square meters) and its price to determine if larger properties command higher prices consistently. This insight will help buyers assess the price efficiency (price per square meter) across different areas or property types.
4. **Neighborhood Price Range Analysis:** Categorizing neighborhoods based on their price ranges to help potential buyers quickly identify areas that match their budget constraints.

Data and Data Sources

Data Collection

The primary dataset for this project was sourced from Kaggle. The dataset, titled "Madrid Real Estate Market," contained extensive information extracted from various online property advertisements. It consisted of 58 columns and 21,742 rows, capturing comprehensive details about properties available for sale in Madrid.

```
1  # Check shape
2  housing_madrid.shape
```

✓ 0.0s

```
(21742, 58)
```

Webscraping for Augmentation

To enrich the dataset and provide additional context, ongoing webscraping efforts targeted Wikipedia for gathering supplementary data, like a table exposing population and square of each district in Madrid.

District	Population (1 Jan 2020)[127]	Area (ha)
Centro	140,991	522.82
Arganzuela	156,176	646.22
Retiro	120,873	546.62
Salamanca	148,405	539.24

EDA, Data Cleaning and Visualizations

Initial Dataset Exploration

Upon initial inspection, the dataset was scrutinized for its integrity and relevance. The dataset contained 58 columns and 21,742 rows, providing a wide range of information, including property-specific details such as price, size, number of rooms, and amenities.

```
1  # Check for missing values
2  housing_madrid.isna().sum()
```

✓ 0.0s

```
1  # Check for duplicates
2  housing_madrid.duplicated().sum()
```

✓ 0.0s

```
1  print(housing_madrid['floor'].unique())
```

✓ 0.0s

```
['3' '4' '1' 'Bajo' '2' '7' '6' 'Semi-sótano' '5' 'Entreplanta exterior'
 '8' '9' 'Entreplanta interior' 'Entreplanta' 'Semi-sótano exterior'
 'Sótano interior' 'Semi-sótano interior' 'Sótano' 'Sótano exterior']
```

```
1  print(housing_madrid['house_type_id'].unique())
```

✓ 0.0s

```
['HouseType 1: Pisos' 'HouseType 4: Dúplex' 'Unknown'
 'HouseType 5: Áticos' 'HouseType 2: Casa o chalet']
```

Data Cleaning and Preprocessing

To ensure data quality and suitability for analysis, a meticulous cleaning and preprocessing phase was undertaken (Handling Missing Values, Standardization, Normalization):

```python
# Drop irrelevant columns
columns_to_drop = ['Unnamed: 0', 'title', 'sq_mt_useful', 'latitude', 'longitude',
                   'is_exact_address_hidden', 'street_name', 'street_number',
                   'raw_address', 'portal', 'operation', 'is_floor_under', 'door',
                   'rent_price_by_area', 'is_rent_price_known', 'is_buy_price_known',
                   'buy_price_by_area', 'is_furnished', 'is_kitchen_equipped',
                   'is_new_development', 'are_pets_allowed', 'is_exterior', 'is_accessible', 'rent_price',
                   'has_central_heating', 'has_individual_heating',
                   'energy_certificate', 'has_private_parking','has_public_parking']
housing_madrid.drop(columns=columns_to_drop, inplace=True)
housing_madrid.isna().sum()
```

✓ 0.0s

```python
# Handling missing values

# 1. Drop rows with missing values in columns 'sq_mt_built' and 'n_bathrooms'
housing_madrid.dropna(subset=['sq_mt_built', 'n_bathrooms'], inplace=True)

# 2. Replace missing values in columns 'n_floors', 'parking_price' and 'sq_mt_allotment' with 0
housing_madrid['n_floors'].fillna(0, inplace=True)
housing_madrid['parking_price'].fillna(0, inplace=True)
housing_madrid['sq_mt_allotment'].fillna(0, inplace=True)

# 3. Replace missing values in column 'house_type_id' with 'Unknown'
housing_madrid['house_type_id'].fillna('Unknown', inplace=True)

# 4. Replace missing values in column 'built_year' with average
average_built_year = housing_madrid['built_year'].mean()
housing_madrid['built_year'].fillna(average_built_year, inplace=True)

# 5. Replace missing values in column 'floor' with mode
floor_mode = housing_madrid['floor'].mode()[0]
housing_madrid['floor'].fillna(floor_mode, inplace=True)

# 6. Update a specific value in the 'built_year' column
# 6.1. Identify the outlier value
outlier_value = 8170
new_value = 1970

# 6.2. Replace the outlier value with the new value
housing_madrid.loc[housing_madrid['built_year'] == outlier_value, 'built_year'] = new_value

# Verify the changes
print(housing_madrid.info())
```

✓ 0.0s

```python
# Define a mapping for 'floor' column
floor_mapping = {
    'Bajo': 0,
    'Entreplanta': 1,
    'Entreplanta interior': 1,
    'Entreplanta exterior': 1,
    'Semi-sótano': -1,
    'Semi-sótano interior': -1,
    'Semi-sótano exterior': -1,
    'Sótano': -2,
    'Sótano interior': -2,
    'Sótano exterior': -2,
    '1': 1,
    '2': 2,
    '3': 3,
    '4': 4,
    '5': 5,
    '6': 6,
    '7': 7,
    '8': 8,
    '9': 9,
}

# Apply the mapping to 'floor' column
housing_madrid['floor'] = housing_madrid['floor'].map(floor_mapping)

print(housing_madrid['floor'].unique())
```

✓ 0.0s

```
[ 3  4  1  0  2  7  6 -1  5  8  9 -2]
```

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) involved a comprehensive examination of the dataset to uncover patterns, spot anomalies, and test hypotheses. This phase was critical for understanding the underlying structure of the data and informing subsequent modeling efforts.

- ! Statistical Summaries:

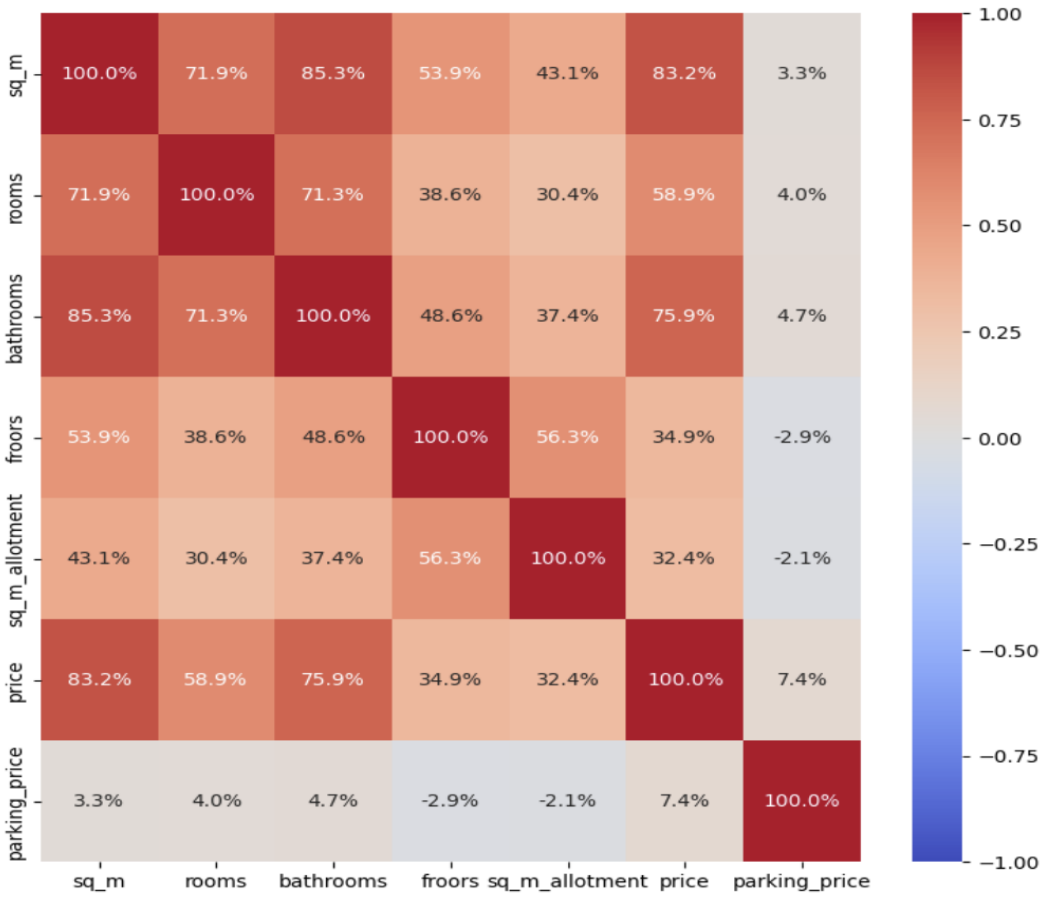
```python
housing_madrid.describe()
```

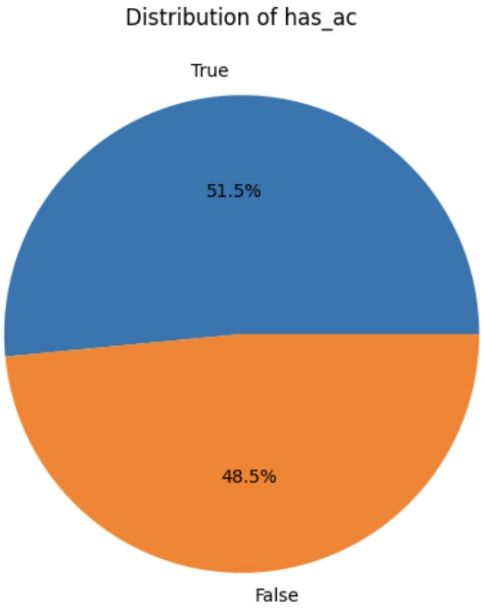
✓ 0.0s Python

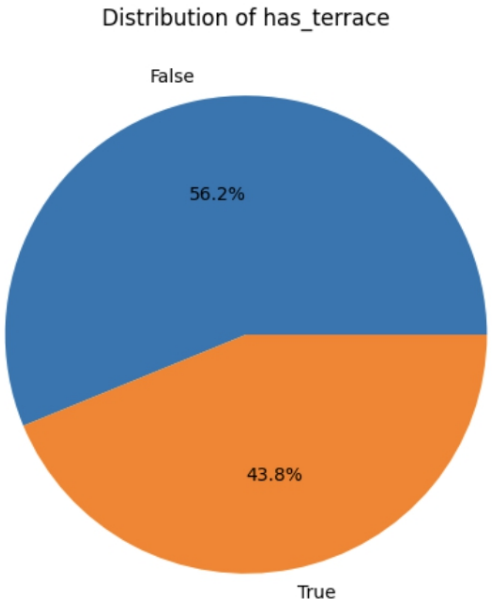
	Unnamed: 0	id	sq_mt_built	sq_mt_useful	n_rooms	n_bathrooms	n_floors	sq_mt_allotment	latitude	longitude	portal	door	
count	21742.000000	21742.000000	21616.000000	8228.000000	21742.000000	21726.000000	1437.000000	1432.000000	0.0	0.0	0.0	0.0	2
mean	10870.500000	10871.500000	146.920892	103.458192	3.005749	2.091687	3.128740	241.692737	NaN	NaN	NaN	NaN	-5
std	6276.519112	6276.519112	134.181865	88.259192	1.510497	1.406992	0.907713	247.484853	NaN	NaN	NaN	NaN	5
min	0.000000	1.000000	13.000000	1.000000	0.000000	1.000000	1.000000	1.000000	NaN	NaN	NaN	NaN	-3
25%	5435.250000	5436.250000	70.000000	59.000000	2.000000	1.000000	2.000000	2.000000	NaN	NaN	NaN	NaN	7
50%	10870.500000	10871.500000	100.000000	79.000000	3.000000	2.000000	3.000000	232.000000	NaN	NaN	NaN	NaN	1
75%	16305.750000	16306.750000	162.000000	113.000000	4.000000	2.000000	4.000000	354.000000	NaN	NaN	NaN	NaN	1
max	21741.000000	21742.000000	999.000000	998.000000	24.000000	16.000000	7.000000	997.000000	NaN	NaN	NaN	NaN	2

Visualizations Using Matplotlib and Seaborn

To effectively communicate the insights gained from EDA, a variety of visualizations were created using the Matplotlib and Seaborn libraries:




Distribution of has_ac
True
51.5%
48.5%
False


Distribution of has_terrace
False
56.2%
43.8%
True

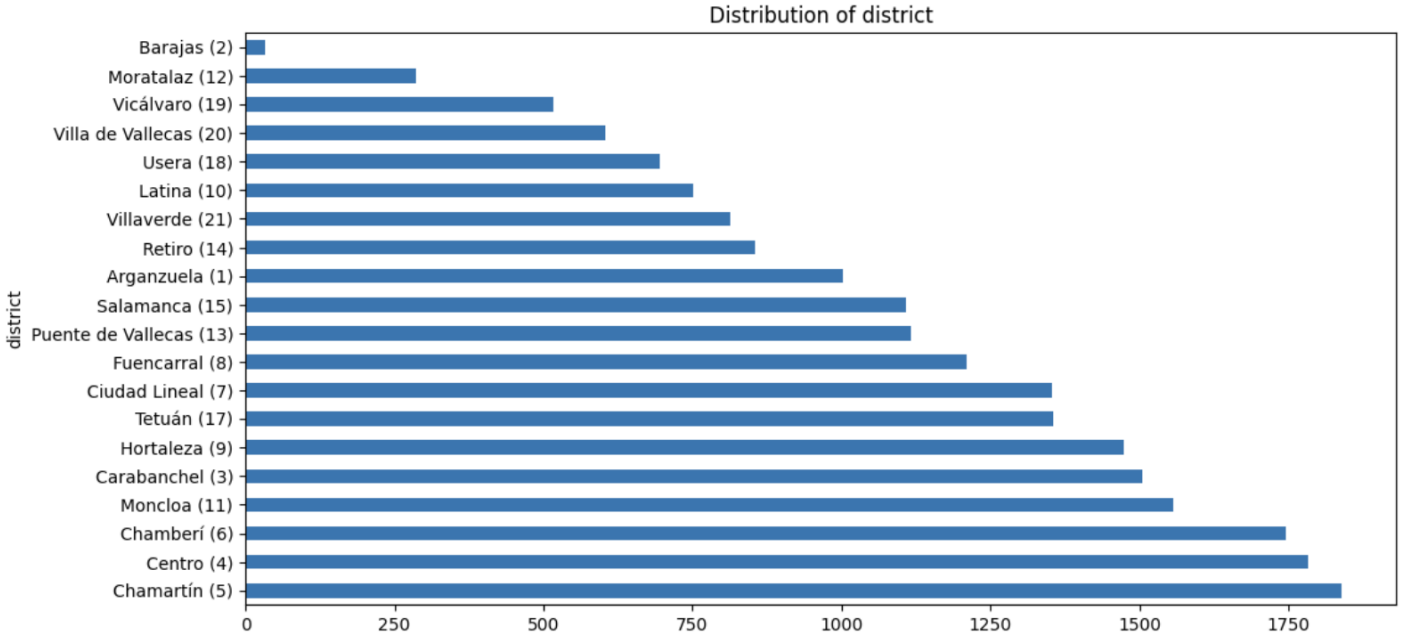
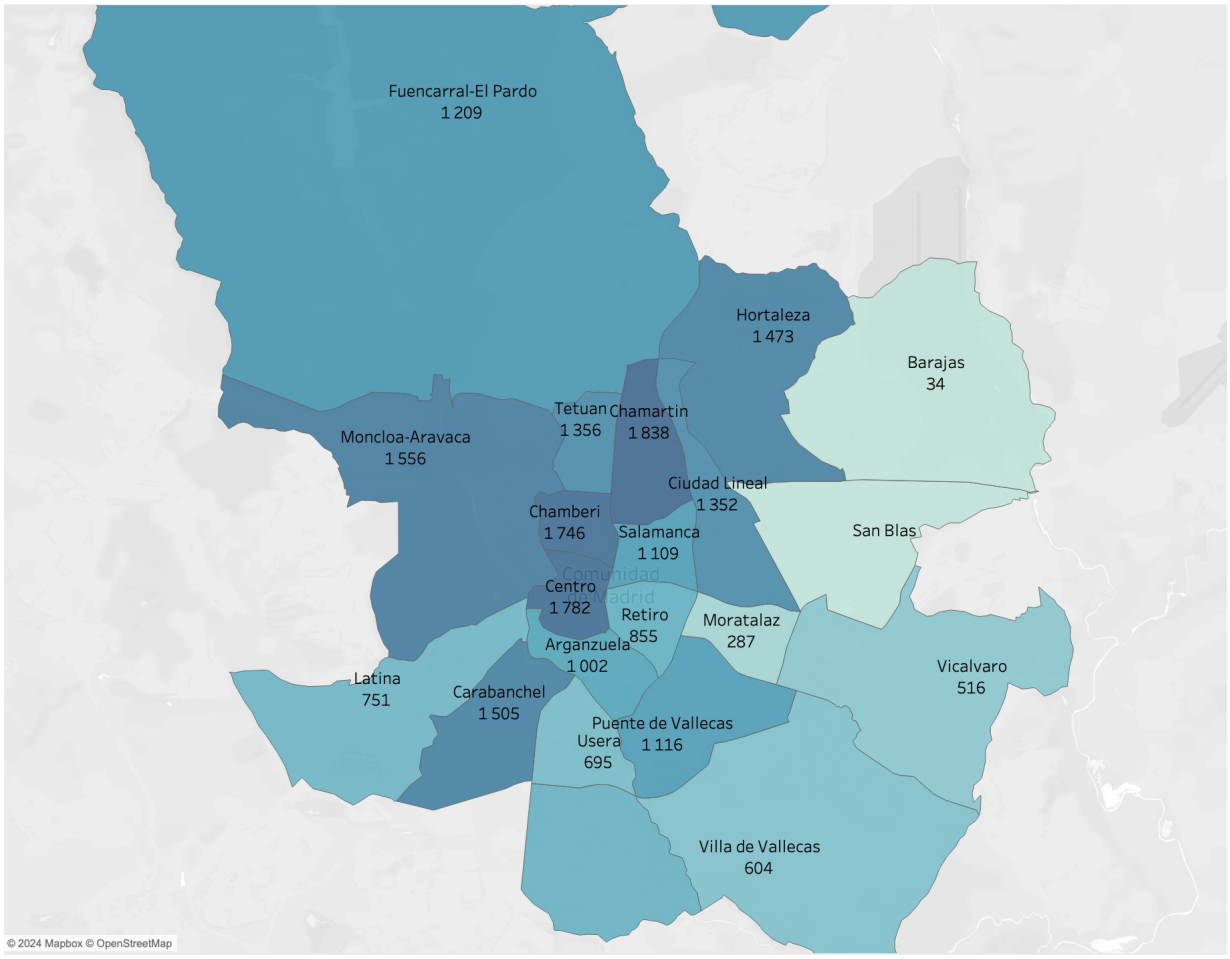

Distribution of district
district
Barajas (2)
Moratalaz (12)
Vicálvaro (19)
Villa de Vallecas (20)
Usera (18)
Latina (10)
Villaverde (21)
Retiro (14)
Arganzuela (1)
Salamanca (15)
Puente de Vallecas (13)
Fuencarral (8)
Ciudad Lineal (7)
Tetuán (17)
Hortaleza (9)
Carabanchel (3)
Moncloa (11)
Chamberí (6)
Centro (4)
Chamartín (5)
0
250
500
750
1000
1250
1500
1750

Tableau Visualizations

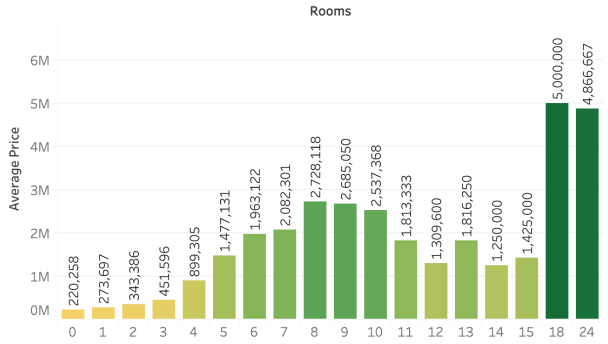
Several visualizations were created using Tableau to provide interactive insights into the Madrid real estate market. These visualizations include:

- Geographic distribution of properties across Madrid.
- Impact on Price and Price per Square Meter of:
  - Number of Rooms;
  - Number of Bathrooms;
  - Size;
  - House type;
- Distribution of House Types

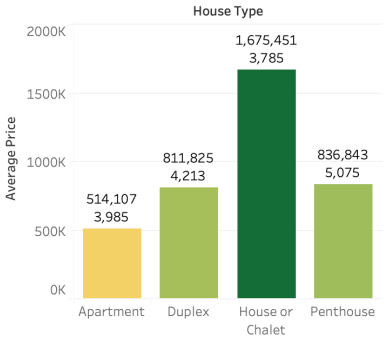
Properties for sale/ District



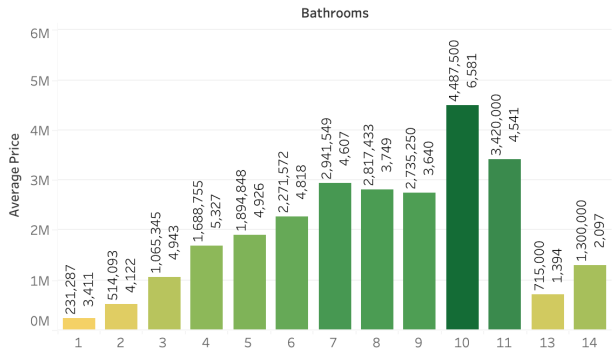
Average Price/ Number of Rooms



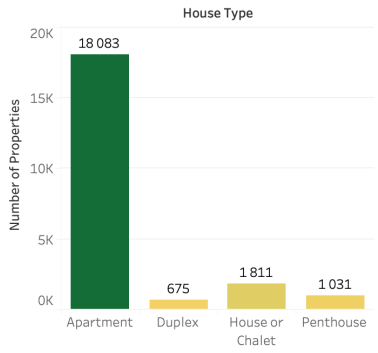
Average Price/ House Type



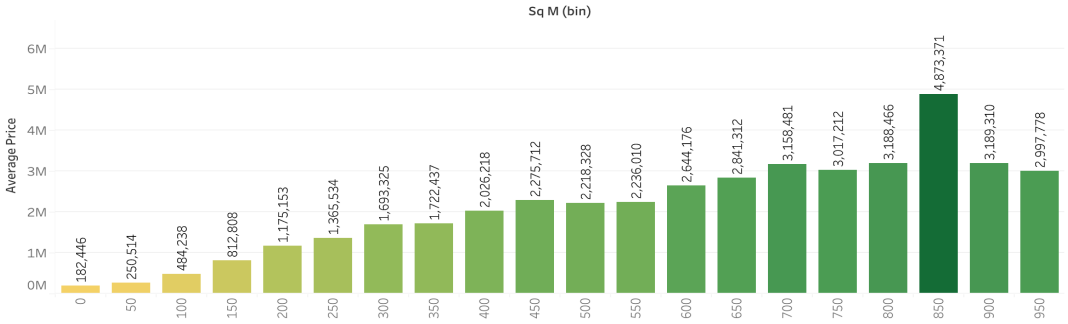
Average Price/ Number of Bathrooms



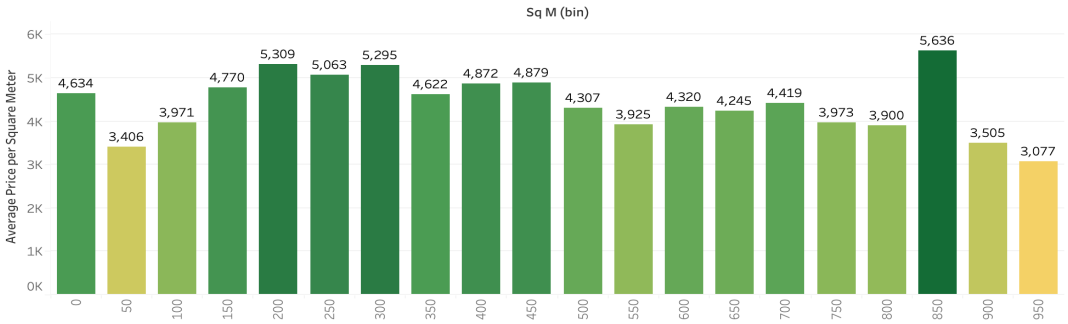
Distribution of House Types



Avg Price/ Size



Avg Sq_m Price/ Size



SQL: Database Creation, ERD & SQL Queries

Database Creation and Schema Design

The process began with the creation of a relational database schema named real_estate.

```sql
-- Create Schema
CREATE SCHEMA IF NOT EXISTS real_estate;
```

Table Creation:

- **Districts Table:** Contains information about the different districts in Madrid.
- **Neighborhoods Table:** Details on neighborhoods within each district.
- **House Types Table:** Categories of properties (e.g., apartment, villa, studio).
- **New Housing Madrid Table:** Comprehensive information on each property, including its price, size, number of rooms, and amenities.

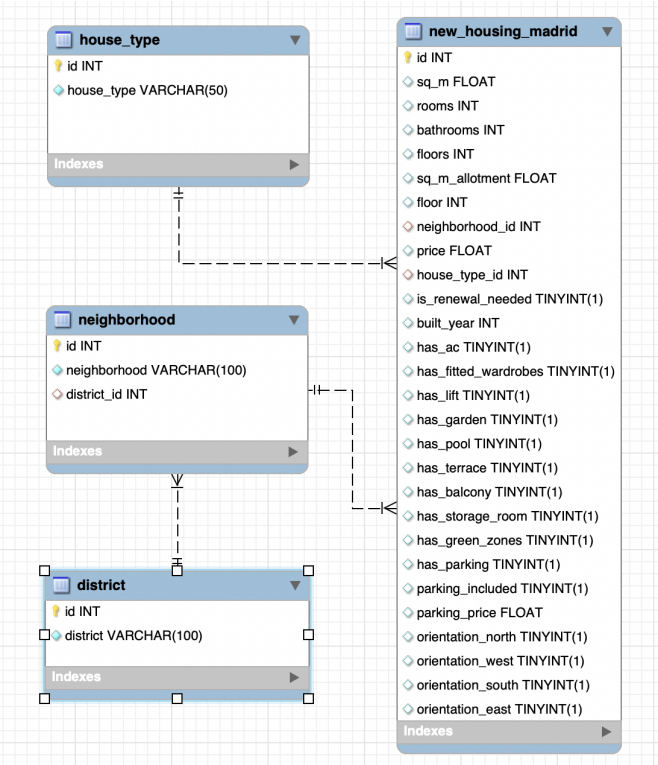
```sql
-- Create district table
CREATE TABLE district (
    id INT PRIMARY KEY AUTO_INCREMENT,
    district VARCHAR(100) UNIQUE NOT NULL
);

-- Create neighborhood table
CREATE TABLE neighborhood (
    id INT PRIMARY KEY AUTO_INCREMENT,
    neighborhood VARCHAR(100) UNIQUE NOT NULL,
    district_id INT,
    FOREIGN KEY (district_id) REFERENCES district(id)
);

-- Create house_type table
CREATE TABLE house_type (
    id INT PRIMARY KEY AUTO_INCREMENT,
    house_type VARCHAR(50) UNIQUE NOT NULL
);
```

Entity-Relationship Diagram (ERD)

An Entity-Relationship Diagram (ERD) was designed to illustrate the relationships between the tables within the database. This diagram served as a blueprint for the database structure, showing how entities like districts, neighborhoods, house types, and property listings are interconnected.



Data Population

With the schema and relationships defined, the next step involved populating the database tables with data. SQL queries were meticulously crafted to insert the collected data into the respective tables. This involved several key steps:

```sql
-- Populate district table
INSERT IGNORE INTO district (district)
SELECT DISTINCT district FROM housing_madrid;

-- Populate neighborhood table
INSERT IGNORE INTO neighborhood (neighborhood, district_id)
SELECT DISTINCT hm.neighborhood, d.id
FROM housing_madrid hm
JOIN district d ON hm.district = d.district;
```

Data Extraction for Analysis

Once the database was populated, SQL queries were employed to extract relevant information for analysis. These queries were designed to gather insights into the real estate market in Madrid, such as:

Average Price per Square Meter by District

```sql
-- Calculate the Average Price per Square Meter by District
SELECT d.district, ROUND(AVG(nhm.price / nhm.sq_m), 2) AS avg_price_per_sqm
FROM new_housing_madrid nhm
JOIN neighborhood n ON nhm.neighborhood_id = n.id
JOIN district d ON n.district_id = d.id
GROUP BY d.district
ORDER BY avg_price_per_sqm DESC;
```

district	avg_price_per_sqm
Salamanca	6550.42
Chamberí	5777.22
Chamartín	5529.03
Centro	5488.99
Retiro	5166.87
Moncloa	4194.55
Arganzuela	4112.38
Tetuán	3964.46
Hortaleza	3929.33
Fuencarral	3669.33

Distribution of Properties in a Specific District

```sql
-- Retrieve All Properties in a Specific District
SELECT nhm.id, nhm.sq_m, nhm.rooms, nhm.bathrooms, nhm.price, n.neighborhood, d.district
FROM new_housing_madrid nhm
JOIN neighborhood n ON nhm.neighborhood_id = n.id
JOIN district d ON n.district_id = d.id
WHERE d.district = 'Villaverde';
```

id	sq_m	rooms	bathrooms	price	neighborhood	district
1	64	2	1	85000	San Cristóbal	Villaverde
15	64	3	1	72000	San Cristóbal	Villaverde
89	65	3	1	94000	San Cristóbal	Villaverde
159	66	3	1	104900	San Cristóbal	Villaverde
161	74	3	1	100000	San Cristóbal	Villaverde
167	69	3	1	107000	San Cristóbal	Villaverde
193	95	2	2	184000	San Cristóbal	Villaverde
207	60	2	1	73000	San Cristóbal	Villaverde
213	65	2	1	110000	San Cristóbal	Villaverde

Number of Properties with Specific Features

```sql
-- Count the Number of Properties with Specific Features
SELECT
    COUNT(*) AS total_properties,
    SUM(has_garden) AS properties_with_garden,
    SUM(has_pool) AS properties_with_pool,
    SUM(has_terrace) AS properties_with_terrace
FROM new_housing_madrid;
```

total_properties	properties_with_garden	properties_with_pool	properties_with_terrace
21600	1445	5063	9455

Connecting Python to MySQL

```
1  import pandas as pd
2  import numpy as np
3  from sqlalchemy import create_engine
4  import os
```

✓ 0.0s

```
1  pw = os.getenv('MYSQL')
2  connection_string = 'mysql+pymysql://root:' + pw + '@localhost:3306/'
3  engine = create_engine(connection_string)
```

✓ 0.0s

```
1  housing_madrid.to_sql(name='housing_madrid', con=engine, schema='real_estate', index=False, if_exists='replace')
```

✓ 1.4s

```
21600
```

Exposing Data via API

API Development Overview

To facilitate programmatic access to the real estate data, an API was developed using Flask. The API serves as an interface for users and applications to interact with the real estate database, allowing for flexible and efficient data retrieval based on user-specified criteria.

1. **Setting Up Flask Application:**

   ! The Flask framework was initialized to create the API application.

```
from flask import Flask, abort, request, jsonify
import pymysql
import json
import math

app = Flask(__name__)
```

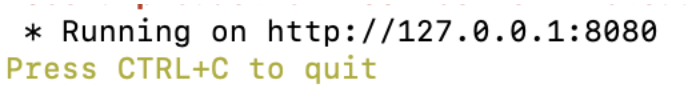
   ! Configuration settings were applied to ensure smooth operation, including setting up the connection to the MySQL database.

```
db_conn = pymysql.connect(
    host="localhost",
    user="root",
    password="16041988",
    database="real_estate",
    cursorclass=pymysql.cursors.DictCursor
)
```

2. **Endpoints for Data Retrieval:** Several endpoints were developed to enable users to retrieve filtered property listings based on various criteria. Key endpoints included:

```
@app.route("/new_housing_madrid")
def listing():
    page = int(request.args.get('page', 0))
    page_size = int(request.args.get('page_size', MAX_PAGE_SIZE))
    page_size = min(page_size, MAX_PAGE_SIZE)
    rooms = request.args.get('rooms')
    district = request.args.get('district')
    house_type = request.args.get('house_type')
```

3. **Deploying the API:** The Flask application was run on a local development server to test the API functionalities and ensure that all endpoints operated as expected.



API Functionalities and Usage

The API enables users to interact with the real estate data seamlessly, offering some key functionalities to retrieve properties listings: users can fetch comprehensive lists of properties, filtered by criteria such as district, neighborhood, number of rooms, and house type. This allows for targeted searches, making it easier to find properties that match specific preferences.


```
{
  "last_page": "/new_housing_madrid?page=215&page_size=100&rooms=None&district=None&house_type=None",
  "new_housing_madrid": [
    {
      "district": "Arganzuela",
      "h_id": 20007,
      "house_type": "Apartment",
      "neighborhood": "Imperial",
      "rooms": 2,
      "sq_m": 78.0
    },
    {
      "district": "Arganzuela",
      "h_id": 20012,
      "house_type": "Apartment",
      "neighborhood": "Imperial",
      "rooms": 3,
      "sq_m": 70.0
    },
    {
      "district": "Arganzuela",
      "h_id": 20038,
      "house_type": "Apartment",
      "neighborhood": "Imperial",
      "rooms": 3,
      "sq_m": 125.0
    },
```


Filtering search by number of rooms:


```
  {
    "district": "Villa de Vallecas",
    "h_id": 1837,
    "house_type": "Apartment",
    "neighborhood": "Ensanche de Vallecas - La Gavia",
    "rooms": 1,
    "sq_m": 61.0
  },
  {
    "district": "Usera",
    "h_id": 1862,
    "house_type": "Apartment",
    "neighborhood": "Pradolongo",
    "rooms": 1,
    "sq_m": 31.0
  },
```

Machine Learning

The machine learning component focused on predicting property prices using various features. This involved feature engineering, model training, evaluation, and hyperparameter tuning.

1. Feature Engineering

Feature engineering prepared the dataset for modeling:

1. **Encoding Categorical Variables:**
    - Converted categorical features into numerical format using **one-hot encoding.**

```
# Get dummies out of house_type_id and district columns
categorical = pd.get_dummies(housing_madrid[['house_type_id','district']])
categorical
```

2. **Feature Scaling**:
    - Normalized or standardized numerical features.

```
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

2. Model Training and Evaluation

Various regression models were trained and evaluated:

```
# Initialize models
models = {
    'KNN': KNeighborsRegressor(),
    'Bagging': BaggingRegressor(estimator=DecisionTreeRegressor(), random_state=1),
    'Random Forest': RandomForestRegressor(random_state=1),
    'Gradient Boosting': GradientBoostingRegressor(random_state=1),
    'AdaBoost': AdaBoostRegressor(estimator=DecisionTreeRegressor(), random_state=1),
    'XGBoost': XGBRegressor(random_state=1)
}
```

3. Hyperparameter Tuning

Hyperparameter tuning optimized model performance:

```
# Define function for hyperparameter tuning with Grid Search and Randomized Search
def best_model(model, param_grid, X_train_scaled, y_train):
    grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='neg_mean_squared_error', n_jobs=-1)
    random_search = RandomizedSearchCV(estimator=model, param_distributions=param_grid, n_iter=10, cv=5, scoring='neg_mean_squared_error', n_jobs=-1)

    grid_search.fit(X_train_scaled, y_train)
    random_search.fit(X_train_scaled, y_train)

    best_grid = grid_search.best_estimator_
    best_random = random_search.best_estimator_
```

```
# Perform hyperparameter tuning and evaluate models
results = {}
for model_name, model in models.items():
    best_model_selected = best_model(model, param_grids[model_name], X_train_scaled, y_train)
    results[model_name] = evaluate_model_regression(best_model_selected, X_train_scaled, y_train, X_test_scaled, y_test)
```

	MAE	RMSE	R2 Score
KNN	212645.142778	446221.786758	0.649880
Bagging	113202.226693	255086.862254	0.885583
Random Forest	113285.087235	254884.169352	0.885764
Gradient Boosting	113267.822344	262184.606438	0.879127
AdaBoost	109234.366710	253857.046533	0.886683
XGBoost	117279.152071	271279.143374	0.870596

The AdaBoost model was the best predictive model, accurately predicting property prices based on various features in the Madrid real estate market.

Conclusions

This project analyzed the Madrid real estate market using a data-driven approach. We acquired data from Kaggle, cleaned and preprocessed it rigorously to ensure accuracy. Visualizations with Matplotlib, Seaborn, and Tableau revealed insights into property prices, correlations, and market trends.

Using MySQL, we structured the data into a relational database for efficient storage and retrieval, supported by SQL queries for data transformation. An API built with Flask provided secure access to filtered property listings.

Machine learning techniques, including feature engineering and models like K-Nearest Neighbors, Random Forest, Bagging, AdaBoost and XGBoost, predicted property prices with optimized performance through hyperparameter tuning.

In summary, this project demonstrates effective data science methods for understanding the Madrid real estate market, aiding stakeholders in informed decision-making. I hope this analysis will help my friend in finding the perfect apartment in Madrid.