Alícia Chimeno Sarabia
Marc Fortó Cornella

Barcelona Housing Price Prediction

Machine Learning Project Report

This project is organized within the specified Google Drive Repository [1], where all scripts pertinent to the sections outlined below are available. Additionally, the applications have been included in the provided ZIP file.

Contents

Alícia Chimeno Sarabia
Marc Fortó Cornella

Abstract

This project explains the creation of a Machine Learning model to forecast housing prices in Barcelona using real data from Idealista, complemented by social and economic data sourced from OpenData Barcelona.

The first part of the report describes the preprocessing steps, including handling missing values and imbalanced variables, analyzing outliers, and encoding categorical variables. It then details the feature selection techniques used, from where we extracted relevant information of housing prices.

Finally, the modeling section is addressed, implementing multiple models such as Random Forest Classifier (RFC), Support Vector Machines (SVM), Logistic Regression (LR), and K-Nearest Neighbors (KNN) to predict housing price categories. Each model is evaluated based on precision, recall, and F1-scores, with RFC obtaining the best results.

Alícia Chimeno Sarabia
Marc Fortó Cornella

1 Introduction

This project aims to forecast housing prices in Barcelona, using real data from Idealista, on apartments and houses from the different neighborhoods of the city, supplemented by social and economic data sourced from OpenData Barcelona.

id **Idealista Source:** Data extracted from the idealista Search API [2]
(script: `idealista_data_extractor.ipynb`)

- Due to limitations of 100 requests per month, each containing data for 50 houses, we were able to extract a total of 9068 rows, excluding duplicates. The description of the variables is on the table 7 in the appendix.

OpenData Barcelona Source: Data extracted from the OpenData API
(script: `opendata_bcn_extractor.ipynb`)
Each dataset is divided by district and neighborhood.

- Population by continent of birth, sex and five-year age groups [3].
- P80/P20 tax income distribution[1] in the city of Barcelona [4].
- Population aged 16 years old and over by by level of education and sex [5].

2 Data exploration

2.1 Preprocessing

2.1.1 Idealista Dataset

Exploring the Dataset:
Based on these initial observations, we proceeded to drop certain variables:

- Non-informative features: `externalReference`, `thumbnail`, `url`, `distance`, `suggestedTexts`, `hasVideo`, `showAddress`, `numPhotos`, `hasStaging`, `has3DTour`,.

- Features with high percentage of missing values: `newDevelopmentFinished`, `labels`.

- Highly unbalanced features: `topPlus`, `newDevelopment`, `municipality`, `highlight`.

- High variability categorical variable: `address`.

- Variables that are subsumed into others: `priceByArea`.

- Zero-variability features: `operation`, `province`, `country`, `topNewDevelopment`.

---

[1]Indicator for measuring inequality in the distribution of personal or family income. This ratio fluctuates between 0% (maximum equality) and 100% (maximum inequality).

Then, since the information retrieved from the Idealista API was provided in JSON format, certain variables remained in struct format (`priceInfo`, `parkingSpace`, `detailedType`). Consequently, we had to flatten those variables, modifying their format and extracting the relevant information they contained. Upon examination, only the `parkingSpace` variable yielded useful information. We derived the `hasParkingSpace` variable from it and included the price of the parking space in the total price of the property.

Handling missing values and Imbalanced variables:

This part of the project was one of the longest, so we will not explain each step in detail but will highlight the main decisions.

First, we analyzed the `neighborhood` and `district` variables. We observed that some levels of both variables had a very small number of rows. These levels corresponded to houses from neighboring territories such as L'Hospitalet de Llobregat or Sant Adrià de Besòs. We decided to remove some of them and keep the ones that contained more tuples, changing their neighborhood and district values to the more appropriate neighboring ones in Barcelona. Following this, we observed that the `neighborhood` variable was very unbalanced. Therefore, we decided to only use it for joining the distinct datasets and remove it at the end and preserve the district, longitude, and latitude information for geographical location.

Then, we handled missing values in the `exterior` and `floor` variables. First, we detected that all properties of type *'chalet'* had missing values for these two variables. To address this, we created a new `floor` level, *'chalet'*, and set all `exterior` missing values to TRUE, as a chalet usually has a window facing the road.

The rest of the missing values were imputed using the `description` variable. We searched the text within the description for clues about the floor of the apartment (e.g., 'primera planta', 'entrada directa desde calle', 'ático', etc.), and if found, we assigned a value to the floor based on that information. Afterward, there were still some remaining missing values, and since we didn't want to lose any information by removing them, we used a Decision Tree Classifier to predict their values based on the `neighborhood`, `price`, and `propertyType` variables.

The `floor` variable was also highly imbalanced, which prompted us to group some levels together.

The remaining missing values for the `exterior` and `hasLift` variables were imputed using the same procedure. We searched for clue words within the `description` variable to infer these missing values.

Outliers:

We first performed univariate outlier analysis on each variable, removing some of the severe outliers that corresponded to data errors or to excessively large or small values that could influence the model.

Then we performed a multivariate outlier analysis using the EllipticEnvelope class from scikit-learn. We selected an appropriate value for the contamination parameter and, after analyzing the returned outliers, we removed them.

Categorize target variable:

Finally, for simplification reasons, we decided to discretize the `price` variable into 4 levels (Low, Medium, High, Very High). The discretization was performed using the KBinsDiscretizer class from scikit-learn with the KMeans parameter. This method utilizes K-means clustering to divide the dataset into `n_bins` (the number of clusters), which was set to 4 in this case. We decided to use the K-means approach as it captured the distinct ranges of prices very well. However, this approach resulted in an unequal number of rows per level, as you can see in the following plots.
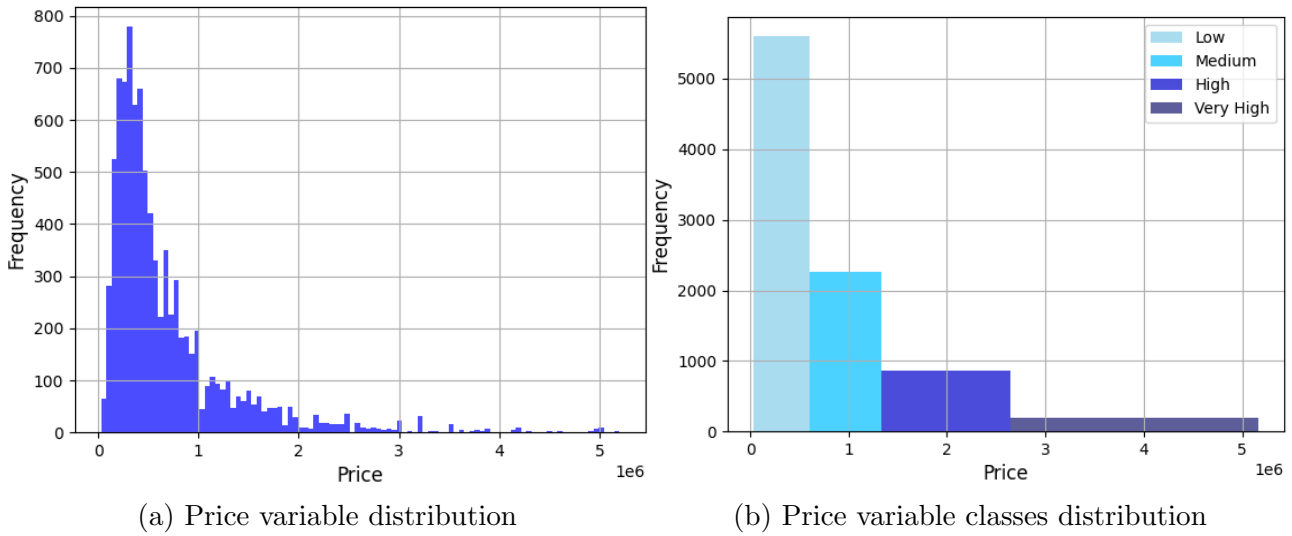

(a) Price variable distribution   (b) Price variable classes distribution

Figure 1: Price variable discretization

2.1.2   OpenData Datasets

These dataframes required minimal preprocessing as there were no missing values. The features that we extracted from them, for each neighborhood, can be seen in the appendices Table [6].

2.1.3   Splitting the Data

Finally, we split the data into 80% training data and 20% test data. We decided to do this because using a larger proportion of the data for training helps in building a more generalized model. We decided not to use a validation set as some of our categorical variables, such as `district`, have a reduced number of rows, and splitting them could increase the risk of overfitting. Therefore, we opted for a larger training set which allows us to implement cross-validation techniques to internally validate the model during the training process, and a reliable test set that could be representative of the overall data distribution.

The above preprocessing steps have been performed in the `data_cleaning.ipynb` script.

2.2   Feature Selection

Once we have preprocessed and joined our datasets, we started our feature selection analysis. The below sections have been performed in the `modeling.ipynb` script.

2.2.1   Scale and Encode Data

Before starting to perform feature analysis methods, we first **scaled all of our numerical variables** using the RobustScaler class from scikit-learn, as it is more robust to outliers than other scaling techniques. This enhances model performance by preventing features with larger ranges from disproportionately influencing the model, which can lead to biased results.

Then we also **encoded the categorical variables**. We used two types of encoding, Label Encoding and One-Hot Encoding. The decision on which encoder to use for each variable is based on the nature of the categorical data, specifically whether the categories have an inherent order or not.

- For `floor` and `Price_Category`, as they both have a natural order (e.g. 'sótano', 'bajo', '1', '2'... or 'Low', 'Medium', 'High', 'Very High') we encoded them with a Label Encoder for maintaining the order.

- For `propertyType`, `district` and `status` variables as they all have nominal categories, meaning they do not have any specific order, one-hot encoding was used to convert each category into a separate binary feature, allowing the model to handle these nominal categories without assuming any order.

One-hot encoding can sometimes lead to a combinatorial explosion of variables, especially if there are many categories within each feature. Nevertheless, in our case, `propertyType` has 6 categories, `status` has 3 categories, and `district` has 10 categories. The latter would be on the verge of acceptability, and perhaps we could have considered encoding it differently. However, we did not want to lose any additional information.

2.2.2   Correlation matrix

We first plotted the **correlation matrix** [9] between numerical features and removed highly correlated features to prevent collinearity problems. We decided to remove the following variables:

- All the age range variables (`Population_20_44`, `Population_45_69`, `Population_less_than_19`, `Population_greater_than_70`), as they are highly correlated with each other and with the `European_population` variable.

- The `African_population` variable, as it is highly correlated with the `Asian_population` and the `Primary` education variable.

- Finally, two education variables (`Less_than_Primary`, `Lower_Secondary`), as they are highly correlated with the other three education variables.

From this matrix, we can also identify numerical features that are highly correlated positively or negatively with the target variable. These are the main ones:

- `size` (0.84), `bathrooms` (0.75), `Tertiary` (0.51), `Distribucio_P80_20` (0.50), `rooms` (0.46)

- `Primary` (-0.42), `longitude` (-0.24)

2.2.3 Principal Component Analysis (PCA)

Then we performed a PCA on the numerical variables with the aim of reducing dimensionality while preserving most of the variance in the data.

First, we performed PCA with the target variable for visualizing relations between variables and then without it for creating a new dataset of Principal Components that we will use for fitting the model.

This alternative training subset that we generated includes the first 7 principal components, as they explain more than 95% of the cumulative variance (see Figure 8), along with the remaining categorical variables.

In the right figure, we plotted the Factorial Map of PC1 vs PC2 for visualizing the relationships between variables in the dataset.
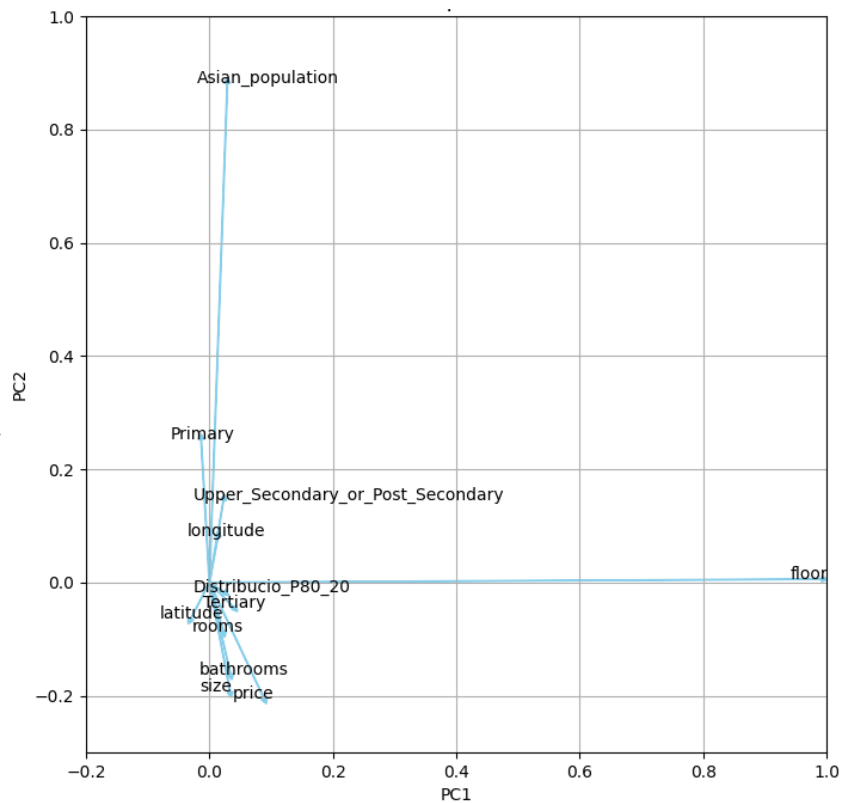

Figure 2: Factorial Map PC1 vs PC2

2.2.4 Clustering

This sections have been performed in the `clustering.ipynb` script.

We employed the unsupervised K-means algorithm to identify clusters based on housing and social characteristics. We used only the numerical variables as K-means uses Euclidean distance, which isn't always meaningful for one-hot encoded variables. By analyzing the elbow plot, we determined that 3 clusters is optimal. The "elbow" point on the plot, where the inertia begins to decrease more gradually, indicates this optimal number. Specifically, the plot shows

a noticeable change in the slope at three clusters, suggesting that adding more clusters beyond this point yields diminishing returns in terms of reducing inertia.

First, we obtained a list of the most important variables to distinguish between clusters. We now want to create a brief profile of each cluster and describe the values that predominate over the rest of the clusters. This profiling is based on the mean scaled values of the features for each cluster.

- **Cluster one:** characterized by larger and more expensive properties with a higher number of rooms and even higher number of bathrooms compared to the others.

- **Cluster two:** has a significant amount of Asian population, moderate property sizes, and a high number of individuals with primary and secondary education levels.

- **Cluster three:** shows average-sized properties with a more balanced demographic and low income inequality.

This information was extracted by performing some statistical analyses, as well as creating box plots and bar plots. The main plot can be found in Figure [11].

2.2.5 Extracted Information

Housing price depends principally on the size of the house and the number of bathrooms and rooms it has. Moreover, neighborhoods with a higher number of qualified residents tend to have more expensive houses and surprisingly, neighborhoods with more inequality among inhabitants are also related to higher house prices.

Demographically, we can visualize a slightly lower number of Asian and African inhabitants in neighborhoods with more expensive houses.

Geographically, we can observe that longitude is inversely correlated with the price variable. This means that as you move towards the east, house prices tend to decrease. The eastern part of Barcelona includes districts such as Sant Martí, Sant Andreu, or Nou Barris, which gather some of the poorer neighborhoods of the city such as Ciutat Meridiana, El Besòs i el Maresme, La Trinitat Vella, or Vallbona, as cited in the journal news [6] or as we can see on the plotted map in Figure 3.
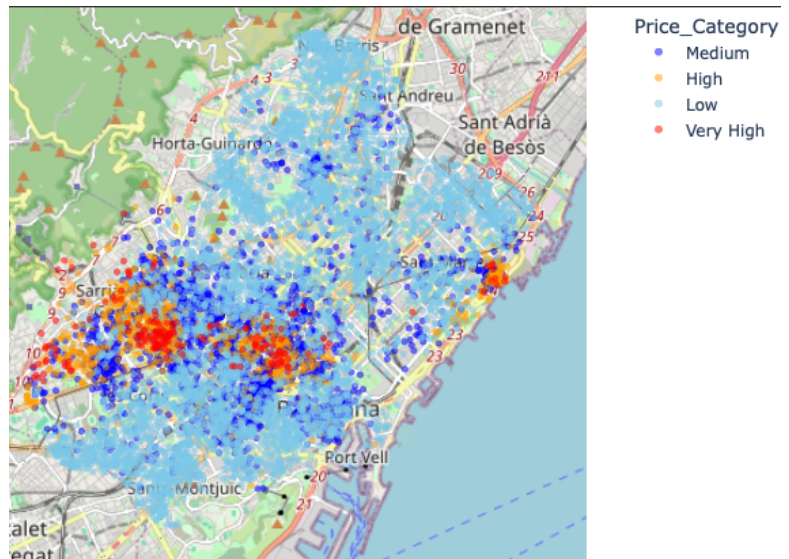

Figure 3: Housing Price distribution over Barcelona Map

2.2.6 Best Features Choice

At this point, after filtering some variables using the previous methods, our Main dataset is composed of 33 variables, while the Principal Components dataset contains 23 variables. We are now considering applying a final step of Recursive Feature Elimination (RFE) to extract the best features from each dataset.

This process starts by fitting a Random Forest Classifier using all our variables and extracting from it a list of the features sorted by importance. Subsequently, it iteratively trains the model using subsets of features, starting from the most important ones and gradually including less important ones. At each step, it evaluates the model's performance using the OOB (Out-of-Bag) error. Finally, it selects the subset of features that yields the best performance.

As our classes are imbalanced, we used the **weighted F-score metric for model selection**. We also decided to consider a score better only if it is at least 0.005 higher than the previous best F1-score. By setting a minimum improvement criterion, we avoid selecting subsets of features that may not offer a substantial enhancement in model performance, thus helping to ensure that the final model is robust and not overly sensitive to small fluctuations in performance. The following diagram provides a more illustrative explanation of the process.
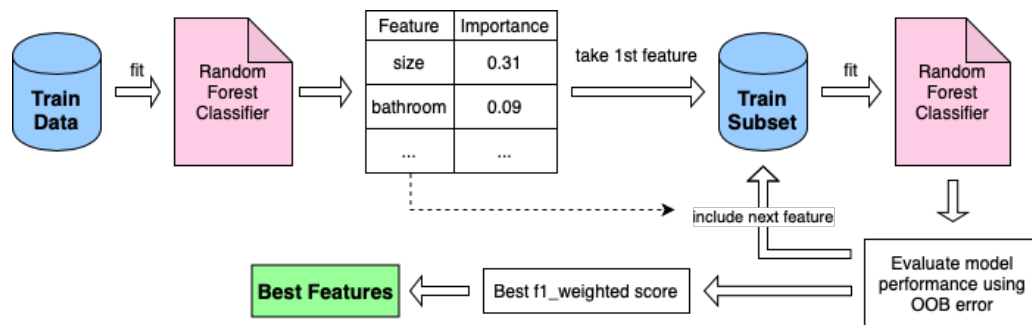

Figure 4: Feature Selection Diagram

Pros:

- By using Random Forests, we are able to capture non-linear relationships between features and the target variable.

- Evaluating the model's performance using OOB error at each step of the feature selection process helps ensure that the selected features generalize well to unseen data.

Cons:

- This process is computationally expensive, especially for a dataset of 33 variables. While we considered using Decision Trees instead of Random Forests due to their computational efficiency, they are more prone to overfitting. As time was not a significant constraint for us, we prioritized obtaining a more accurate model over computational efficiency.

- This process assumes that features are independent of each other, which may not always hold true.

Results:

For the dataset with no PCs (Main Dataset) we obtained the following subset of 8 features: `size`, `bathrooms`, `latitude`, `longitude`, `Tertiary`, `Primary`, `floor`, `rooms`.
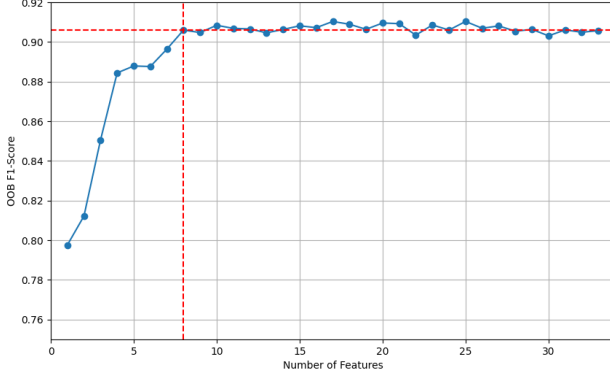
For the dataset with PCs (PC Dataset) we obtained the following subset of 8 features: `PC3`, `PC2`, `PC4`, `PC5`, `PC6`, `PC7`, `PC1`, `hasParkingSpace`

Figure 5: OOB F1-Score vs Number of Features (Main Dataset)
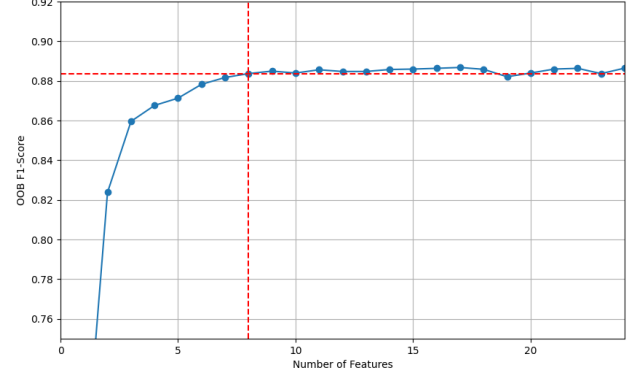

Figure 6: OOB F1-Score vs Number of Features (PC Dataset)

Results are much lower for the PC Dataset, therefore we will only retain the Main Dataset for addressing the Modeling section, as they both have the same number of features.

3 Modeling and Results

3.1 Logistic Regression (LR)

The first modeling method that we considered was the Multinomial Logistic Regression. This model didn't perform as well as the others because Logistic Regression generally has better performance with relatively small to medium-sized datasets, which is not the case here.

After **tuning its hyperparameters** for regularization strength, with a 5-fold cross-validation, we obtained the following results.

Class	Precision	Recall	F1-Score	Support
High	0.67	0.57	0.62	176
Low	0.92	0.95	0.94	1114
Medium	0.73	0.74	0.73	461
Very High	0.87	0.41	0.55	32
Accuracy			0.85	1783
Macro Avg	0.80	0.67	0.71	1783
Weighted Avg	0.84	0.85	0.84	1783

Table 1: LR Results

3.2   Random Forest Classifier (RFC)

The next model we tried was the already mentioned Random Forest Classifier.

We started the modeling by **tuning its hyperparameters** with the OOB error again, using the weighted F-score metric for robustness against the imbalanced categories.

The hyperparameter ranges to search were as follows:

- `n_estimators_list = [100, 150, 180]`

- `max_samples_list = [0.5, 0.7, 0.9]`

- `max_features_list = ['sqrt', 'log2']`

The best parameters found were: `n_estimators: 100`, `max_samples: 0.7`, `max_features: 'sqrt'`, `best_score: 0.9103383526154075`.

Then, we proceeded to estimate the generalization error of the model by making predictions on the test dataset and evaluating the results. The obtained results can be seen in Table 2.

Class	Precision	Recall	F1-Score	Support
High	0.84	0.82	0.83	176
Low	0.96	0.96	0.96	1114
Medium	0.84	0.86	0.85	461
Very High	0.91	0.66	0.76	32
Accuracy			0.92	1783
Macro Avg	0.89	0.82	0.85	1783
Weighted Avg	0.92	0.92	0.92	1783

Table 2: RFC Results

3.3   Support Vector Machines (SVM)

SVM is an effective model for high-dimensional spaces. For this reason, we observed that it obtained better results with a higher number of features and consequently, we added more features to the training dataset. We decided to keep the top 25 features as they corresponded to the best overall F1-score for feature selection (see Figure 5).

We started the modeling by **tuning its hyperparameters**, using grid search with 5-fold cross-validation. GridSearchCV from scikit-learn performs an exhaustive search over specified parameter values for an estimator.

The hyperparameter ranges to search were as follows:

- `C = [1, 100, 150]`

- `kernel = ['rbf']`

- gamma = ['scale', 'auto']

The best parameters found were: `C: 150, gamma: 'auto', kernel: 'rbf'`.

Then, we proceeded to estimate the generalization error of the model by making predictions on the test dataset and evaluating the results. The obtained results can be seen in Table 3.

Class	Precision	Recall	F1-Score	Support
High	0.82	0.79	0.80	176
Low	0.95	0.96	0.96	1114
Medium	0.85	0.84	0.84	461
Very High	0.74	0.72	0.73	32
Accuracy			0.91	1783
Macro Avg	0.84	0.83	0.83	1783
Weighted Avg	0.91	0.91	0.91	1783

Table 3: SVM Results

3.4 K-Nearest Neighbors (KNN)

Finally, we implemented K-Nearest Neighbors. The choice of k, which represents the number of nearest neighbors used to predict the class of an unseen point, is crucial for determining the decision boundaries. To fine-tune this hyperparameter, we employed 5-fold cross-validation. This process identified the optimal value of k as 1. You can see the resulting plot in the appendices, Figure [10].

Then, we proceeded to estimate the generalization error of the model by making predictions on the test dataset and evaluating the results. The obtained results can be seen in Table [4].

Class	Precision	Recall	F1-Score	Support
High	0.78	0.73	0.76	176
Low	0.95	0.94	0.95	1114
Medium	0.79	0.82	0.80	461
Very High	0.74	0.81	0.78	32
Accuracy			0.89	1783
Macro Avg	0.82	0.83	0.82	1783
Weighted Avg	0.89	0.89	0.89	1783

Table 4: KNN Results

4   Final Model

	RFC	SVM	LR	KNN
Num. of features	8	25	8	8
F1-Score macro avg.	0.85	0.83	0.71	0.82
F1-Score weighted avg.	0.92	0.91	0.84	0.89

Table 5: Results Comparison

Observing the results in Table 5, we can see that the model with the best performance is the **Random Forest Classifier** with the following hyperparameters: `n_estimators: 100`, `max_samples: 0.7`, `max_features: 'sqrt'`.

With only 8 features, the model manages to obtain an F1-Score weighted average of 0.92 and an F1-Score macro average of 0.85 on the test dataset. Moreover, the chosen hyperparameters help in reducing computational cost and improving generalization. For example:

- `max_features: 'sqrt'` forces the model to consider at each split the square root of the total number of features, which promotes diversity among the trees and reduces overfitting.

- `max_samples: 0.7` indicates that each tree in the forest will be trained on a random sample consisting of 70% of the original training dataset. This ensures that each tree is slightly different, contributing to the diversity of the forest and reducing the computational cost of the model.

Looking at the bellow confusion matrix, we can extract the following conclusions:
Strengths:

- High accuracy for the "Low" class with 1070 true positives.

- Reasonable accuracy for the "Very High" class despite the smaller number of instances.

Weaknesses:

- The "Medium" class has the highest number of misclassifications, indicating that the model has difficulty distinguishing it from the "High" and "Low" classes.

- Some confusions between the "High" and "Medium" classes, as well as the "Very High" and other classes, suggesting that the model might struggle with boundary distinctions between these classes.

Alícia Chimeno Sarabia
Marc Fortó Cornella

(a) Confusion Matrix RFC on test dataset
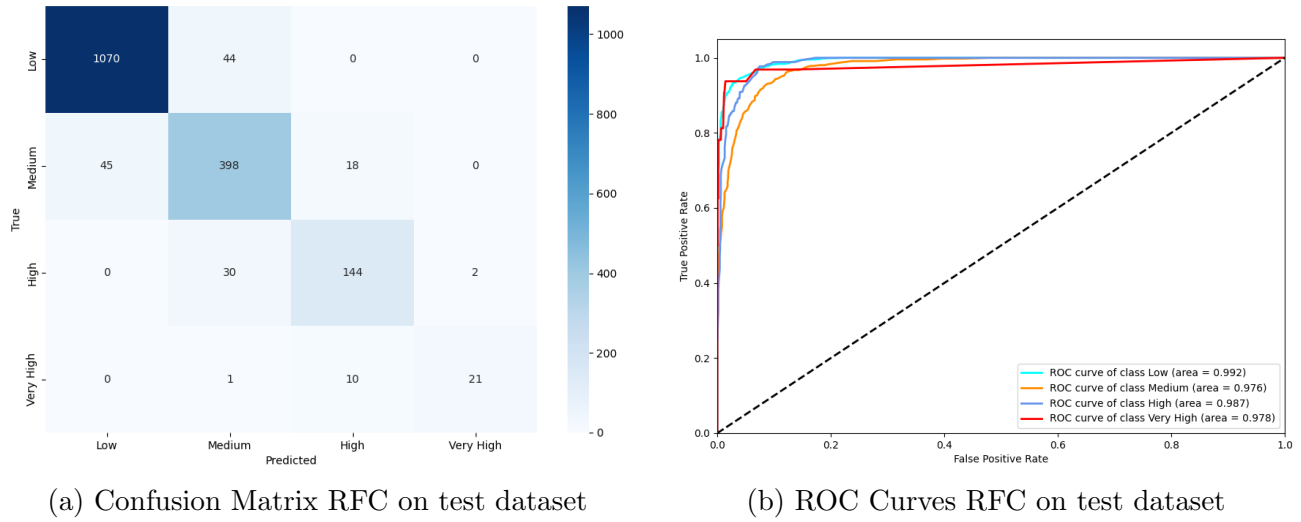

(b) ROC Curves RFC on test dataset

Figure 7: Comparison of Confusion Matrix and ROC Curves

The ROC curves demonstrate very good performance across the 4 classes. All AUC (Area Under the Curve) values are above 0.97, which indicates high performance in classifying instances for all four classes. However, some improvements could still be made to obtain a higher F1-score. In section 6 we highlighted the main ones.

5   Conclusions

This project successfully developed a machine learning model to predict housing prices in Barcelona using data from Idealista and OpenData Barcelona. From the feature extraction section, we observed that housing prices are mainly influenced by the size of the property and the number of bathrooms. Additionally, more expensive houses are found in neighborhoods with more qualified residents and greater inequality, and a soft relation of house prices decreasing when moving eastward was also depicted. In the modeling phase, the Random Forest Classifier outperformed other models, achieving a high accuracy rate, particularly in predicting the "Low" and "Very High" price categories. However, the model struggled with distinguishing "Medium" and "High" categories, indicating areas for improvement.

We personally enjoyed working on this project as we not only learned interesting facts about Barcelona but also gained valuable insights into machine learning. Moreover, we think that the insights gained from this project have significant practical implications for various stakeholders as could be:

- **Real Estate Agents:** The model provides a robust tool for predicting housing prices, which can be used by real estate agents and developers to assess property values accurately.

- **Home Buyers and Sellers:** Potential home buyers and sellers can leverage the model to make informed decisions.

- **Investors:** Real estate investors can use the predictive model to identify investment opportunities by analyzing price trends and future value projections.

6   Future work

Possible Improvements:

- **Class Imbalance Handling**: Check for class imbalance and consider techniques such as resampling.

- **External Data Sources**: Incorporate additional external data sources, such as economic indicators or neighborhood statistics, to enrich the model.

- **Feature Engineering**: Improve feature selection or creation to better differentiate between classes, especially "Medium".

- **Error Analysis**: Conduct a detailed error analysis to understand why misclassifications occur, which could help in refining the model further.

Possible Extensions:

- **Numerical Target Variable**: Develop a more robust model to predict the exact price of the property.

Appendices

Name	Type	Description (population)
Less_than_Primary	numerical	Less than primary education
Lower_Secondary	numerical	Lower secondary education
Primary	numerical	Primary education
Tertiary	numerical	Tertiary education
Upper_Secondary_or_Post_Secondary	numerical	Upper secondary education
Distribucio_P80_20	numerical	Income distribution (80th/20th ratio)
African_population	numerical	African population
American_population	numerical	American population
Asian_population	numerical	Asian population
European_population	numerical	European population
Oceanian_population	numerical	Oceanian population
Population_20_44	numerical	Population aged 20-44
Population_45_69	numerical	Population aged 45-69
Population_less_than_19	numerical	Population aged less than 19
Population_greater_than_70	numerical	Population aged greater than 70

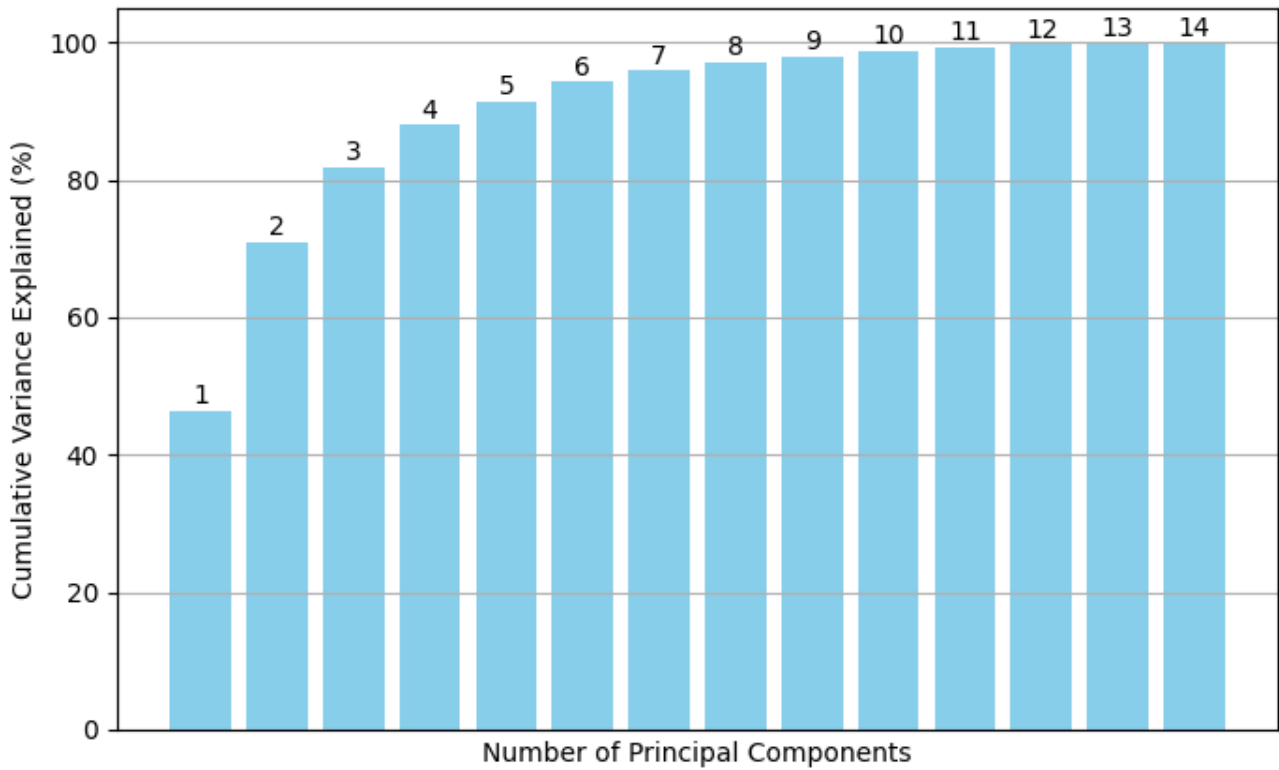Table 6: OpenData Variables Description

Figure 8: Cumulative Variance Explained by Principal Components

Name	Type	Levels	Description
propertyCode	categorical	9068	Unique property codes
thumbnail	categorical	9335	Image thumbnails URL
externalReference	categorical	7557	External references
numPhotos	numerical	98	Number of photos
floor	numerical	31	Floor of the property
price	numerical	1237	Price of the property
priceInfo	categorical	1948	Price information
propertyType	categorical	6	Type of property
operation	categorical	1	Type of operation
size	numerical	445	Size of the property
exterior	binary	2	Whether the property faces the road
rooms	numerical	16	Number of rooms
bathrooms	numerical	10	Number of bathrooms
address	categorical	2369	Address of the property
province	categorical	1	Province of the property
municipality	categorical	3	Municipality of the property
district	categorical	17	District of the property
country	categorical	1	Country of the property
neighborhood	categorical	64	Neighborhood of the property
latitude	numerical	8835	Latitude of the property
longitude	numerical	8856	Longitude of the property
showAddress	binary	2	Whether to show the address
url	categorical	9068	URL of the property
distance	numerical	2770	Distance to query center, in meters
description	categorical	8716	Description of the property
hasVideo	binary	2	Whether the property has a video
status	categorical	3	Status of the property
newDevelopment	binary	2	Whether the property is a new development
hasLift	binary	2	Whether the property has a lift
parkingSpace	categorical	52	Parking space information
priceByArea	numerical	3979	Price per area
detailedType	categorical	9	Detailed type of property
suggestedTexts	categorical	3223	Suggested texts
hasPlan	binary	2	Whether the property shows the plan
has3DTour	binary	2	Whether the property has a 3D tour
has360	binary	2	Whether the property has a 360 view
hasStaging	binary	2	Whether the property has a staging
highlight	categorical	3	Highlight status
topNewDevelopment	binary	1	Top new development status
topPlus	binary	2	Top Plus status
labels	categorical	33	Labels of the property
newDevelopmentFinished	binary	2	New development finished status

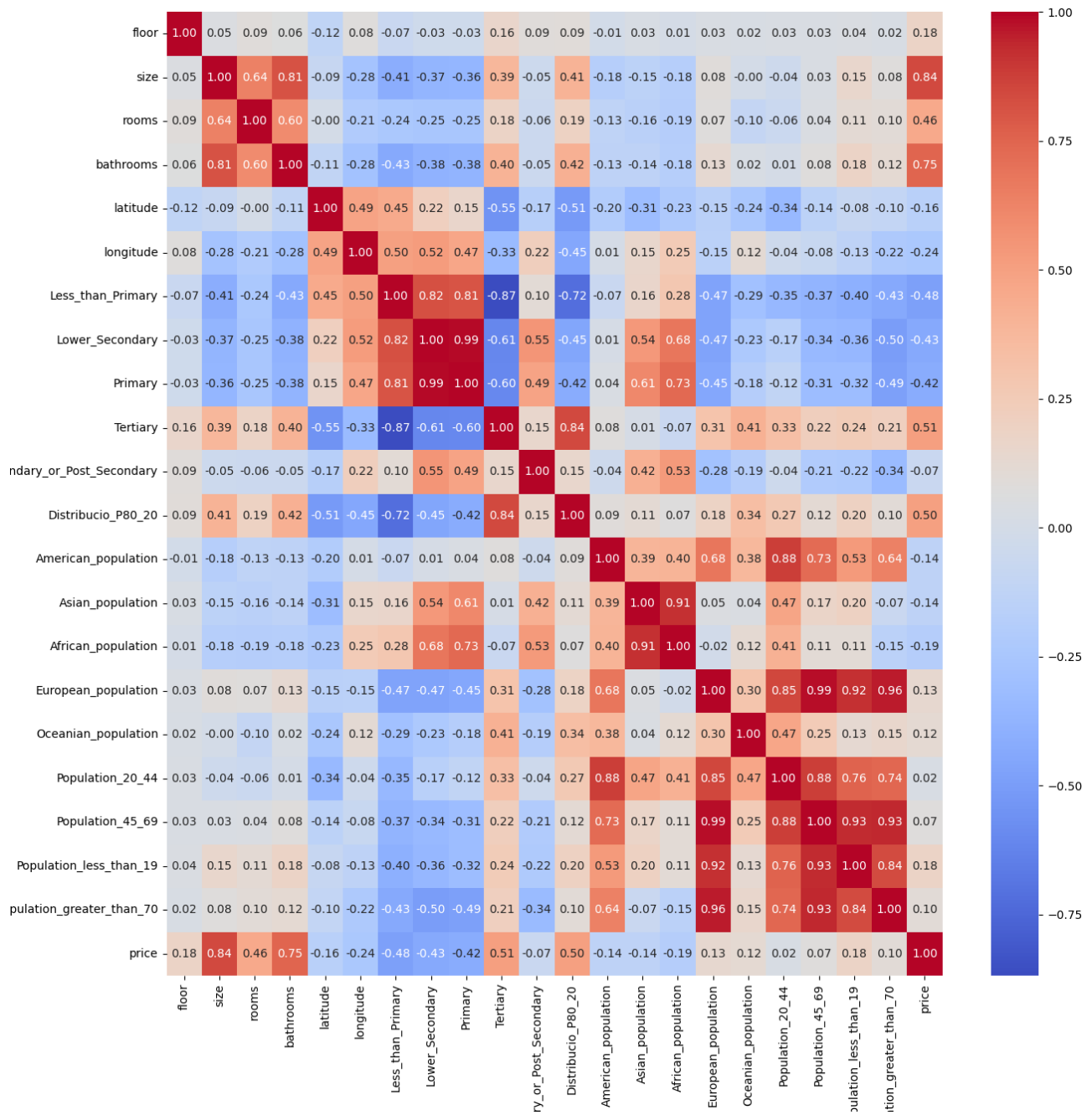Table 7: Idealista Raw Variables Description
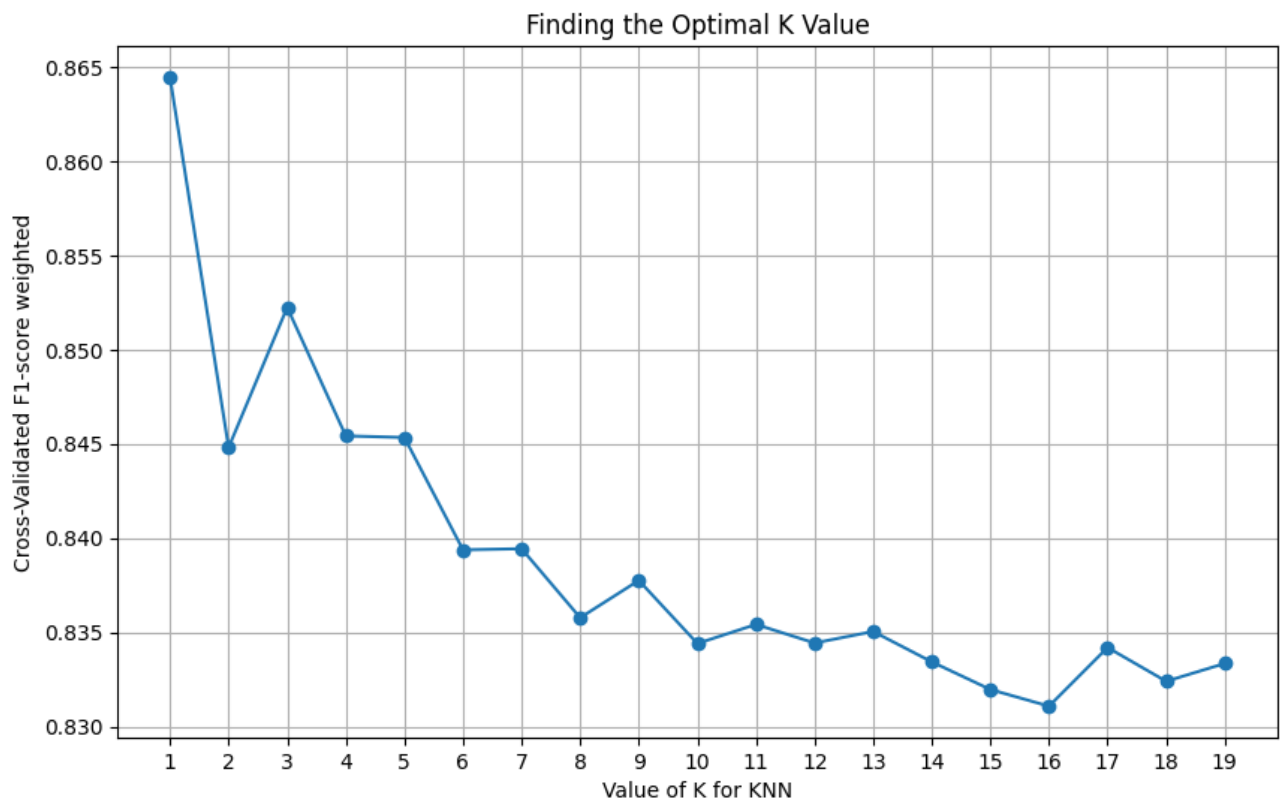
Figure 9: Correlation Matrix
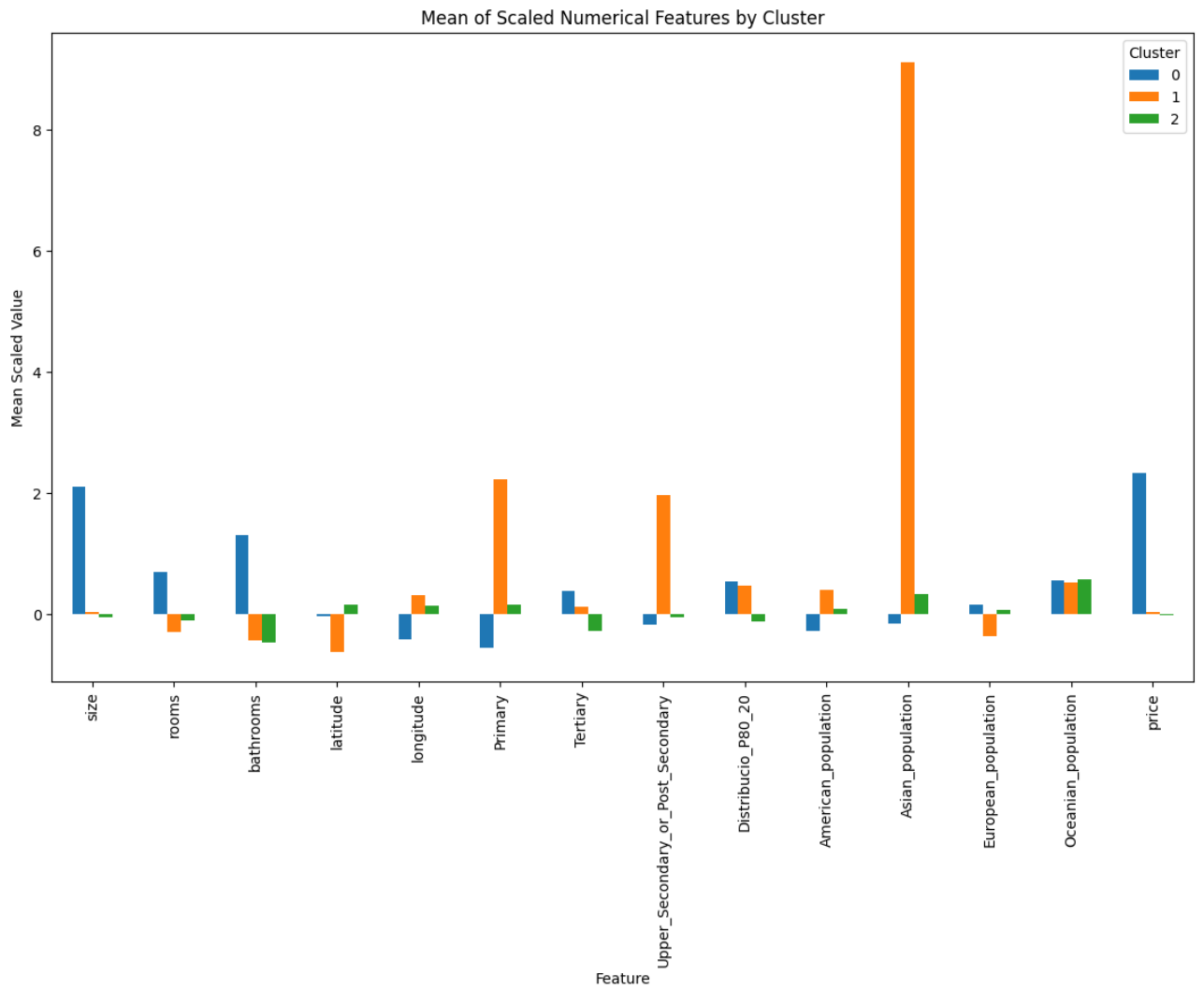
Figure 10: K-NN, K value tuning result

Figure 11: Bar plots of the features for each cluster

References

[1] Marc Fortó Alicia Chimeno. Google Drive Repository. `https://drive.google.com/drive/folders/1yGLkDDBx_boXrBte-uRe1vPvg-_QXto9?usp=sharing`, 2024.

[2] Idealista API. `https://developers.idealista.com/access-request`.

[3] OpenData Barcelona. Population by continent of birth, sex and five-year age groups. `https://opendata-ajuntament.barcelona.cat/data/en/dataset/pad_mdb_lloc-naix-continent_edat-q_sexe/resource/04a4ddca-4789-4312-ad10-a935aded5f65`.

[4] OpenData Barcelona. P80/P20 tax income distribution in the city of Barcelona. `https://opendata-ajuntament.barcelona.cat/data/en/dataset/atles-renda-p80-p20-distribucio`.

[5] OpenData Barcelona. Population aged 16 years old and over by by level of education and sex. `https://opendata-ajuntament.barcelona.cat/data/en/dataset/pad_mdbas_niv-educa-esta_sexe`.

[6] El Periódico. Aquests són els cinc barris més pobres i els cinc més rics de Barcelona. `https://www.elperiodico.cat/ca/barcelona/20230720/barris-mes-pobres-mes-rics-barcelona-idescat-2020-90126034`.