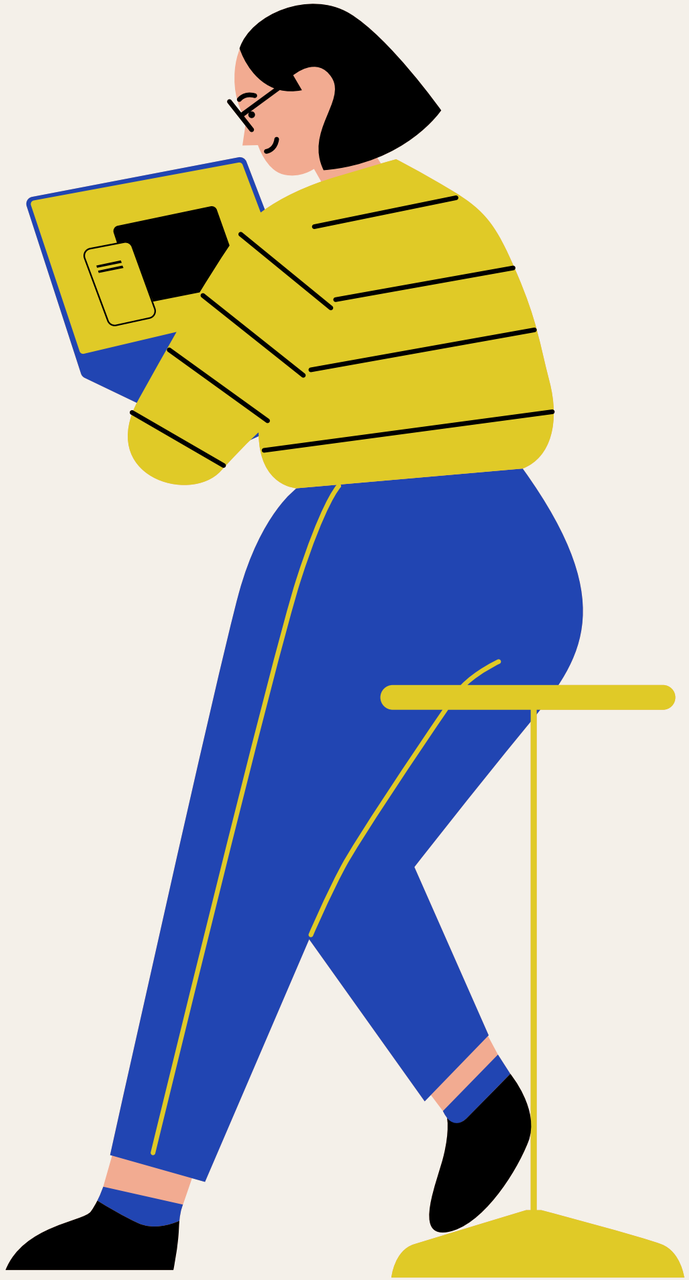
AIRBNB DATA ANALYSİS AND PRICE PREDICTION

MAT 388E - Project Presentation

Berfin Duman- 040190108
Bike Sönmez- 090190326

Contents



1 - Data Set Introduction

- Airbnb Open Data is an open-source platform that offers an overview of the lodging options available at Airbnb locations.
- It is yearly data and information gathering gives travelers a perspective to comprehend, choose, and manage the function of rental housing. Our goal is to examine and predict their data across various Airbnb sites.
- Data from 2019 New York Airbnb listings make up our dataset.

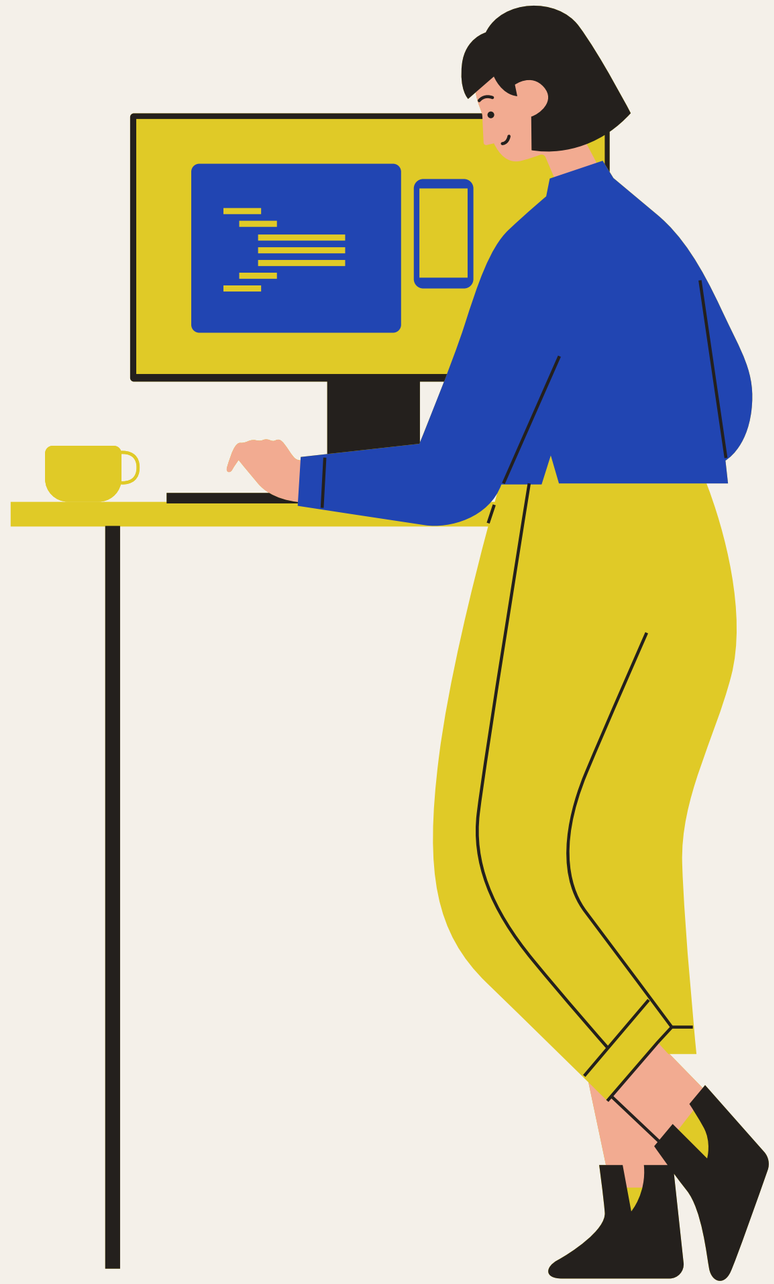
Dataset Features

Variables:

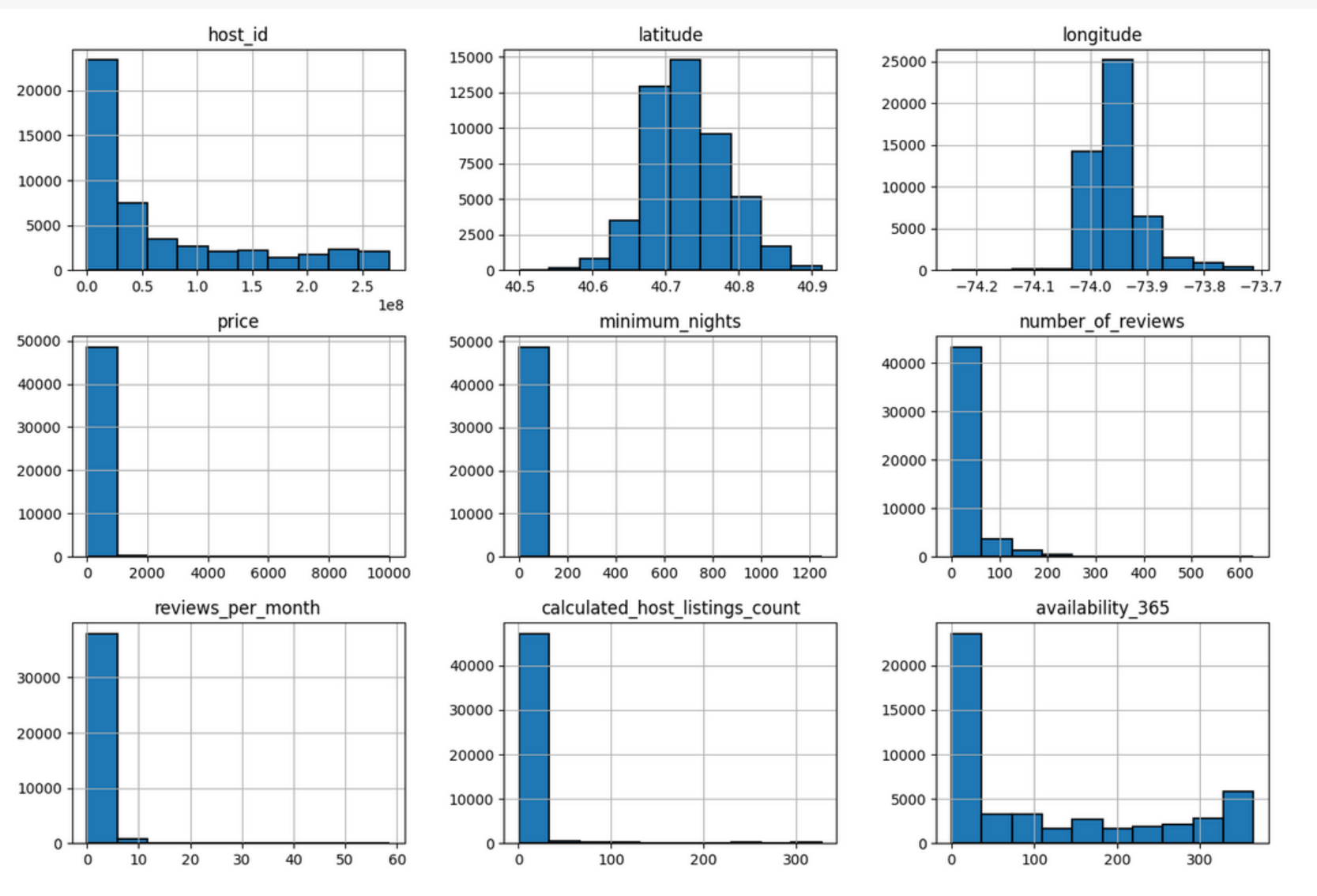
- id: A unique identification number for each Airbnb accommodation.
- name: The name of the accommodation unit.
- host_id: The unique identification number of the host.
- host_name: The name of the host.
- neighbourhood_group: The borough where the accommodation unit is located.
- neighbourhood: The neighborhood where the accommodation unit is located.
- latitude: The latitude coordinate of the accommodation unit.
- longitude: The longitude coordinate of the accommodation unit.
- room_type: The type of the accommodation unit (private room, shared room, entire home/apartment).
- price: The daily price for the accommodation unit.
- minimum_nights: The minimum number of nights that can be booked.
- number_of_reviews: The number of reviews that have been written for the accommodation unit.
- last_review: The date of the last review.
- reviews_per_month: The average number of reviews per month.
- calculated_host_listings_count: The total number of accommodation units that the host has.
- availability_365: The number of days in a year that the accommodation unit is available.
- number_of_reviews_ltm: The number of reviews the listing has (in the last 12 months)
- license: The licence/permit/registration number

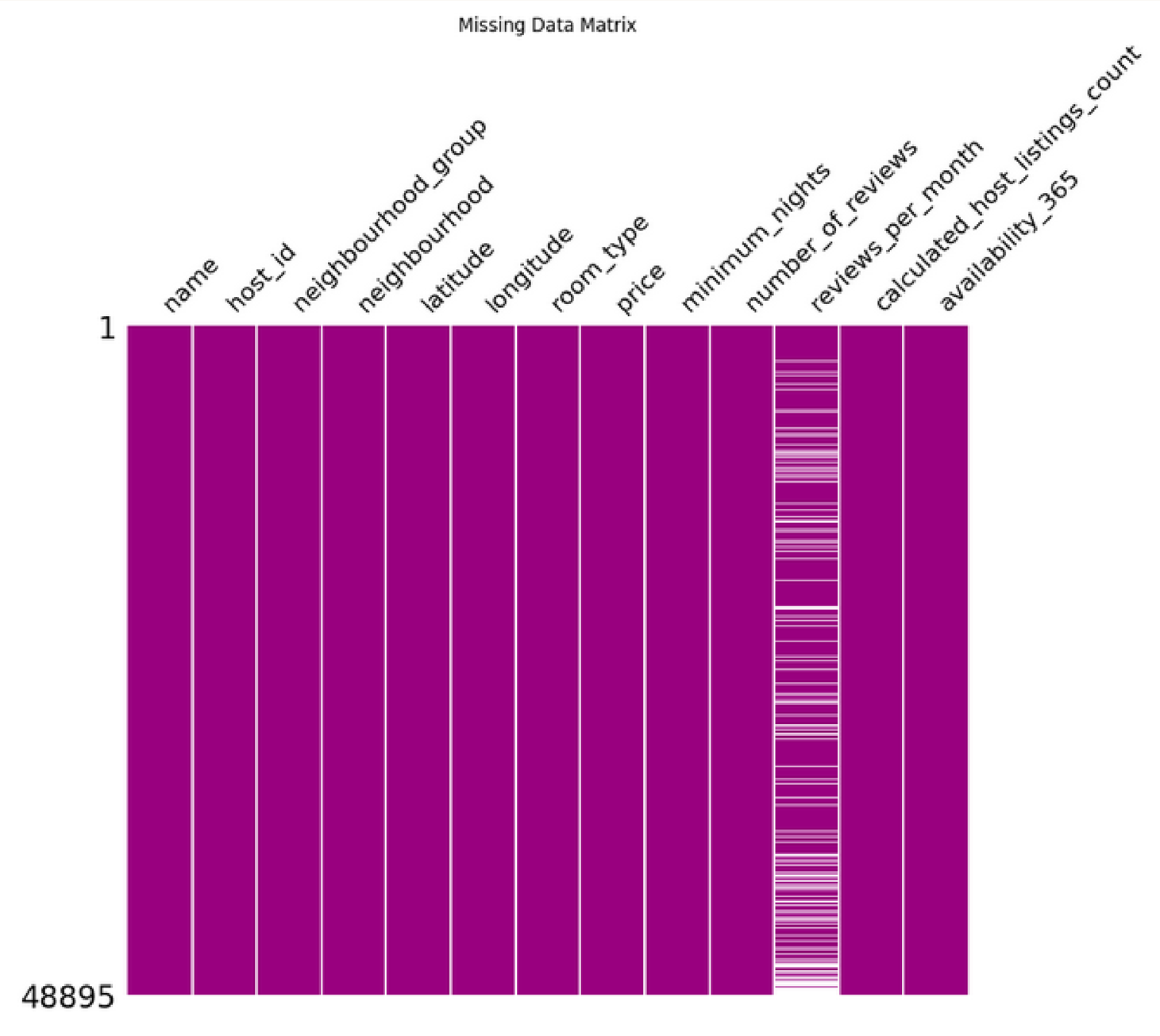
2 - Data Analysis and Visualization

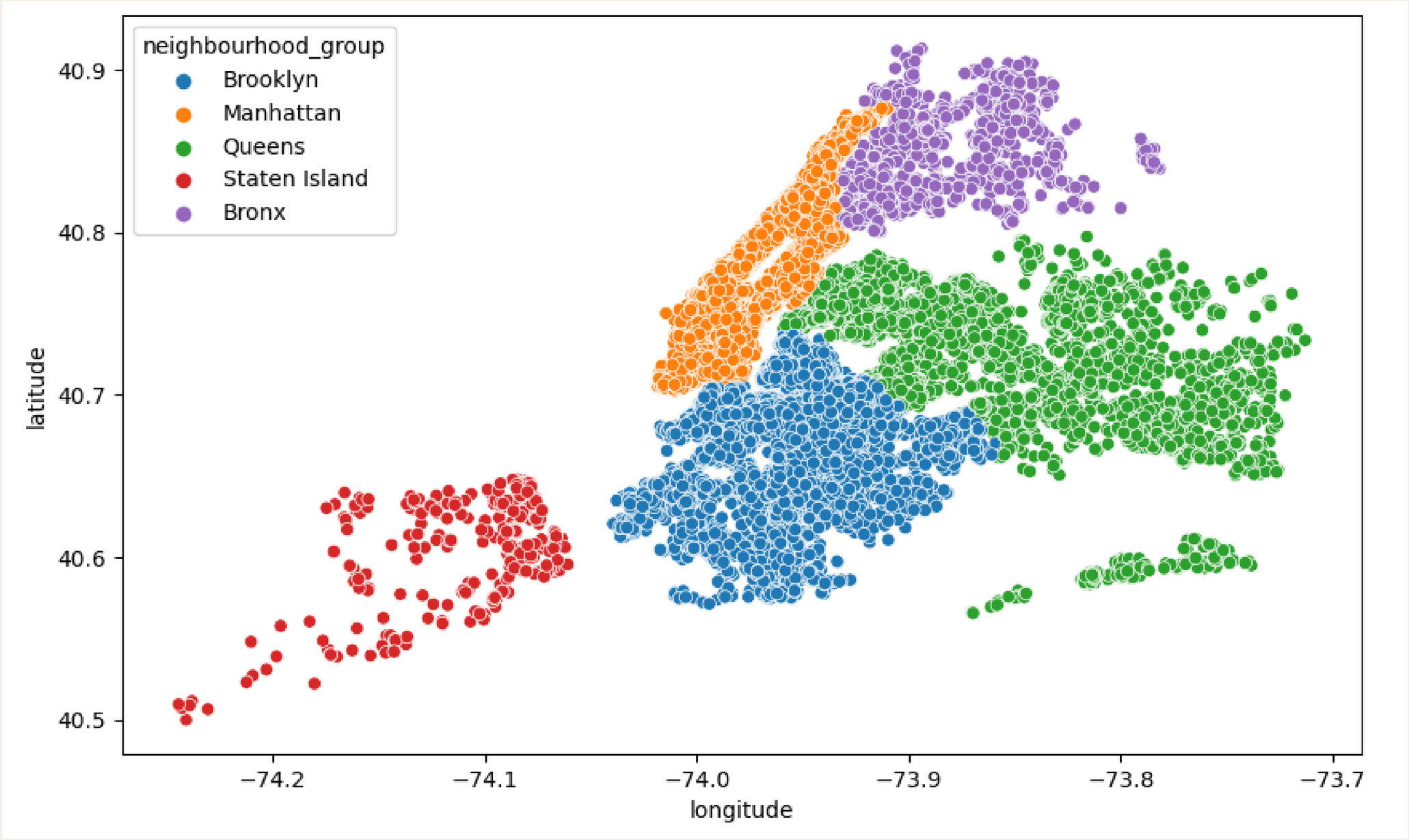
- Necessary libraries and dataset imported
- Performed EDA operations to understand the data
- Faetures were visualised and their relationships were observed
- Data were separated into train and test and correlation analyses were performed only for numerical values



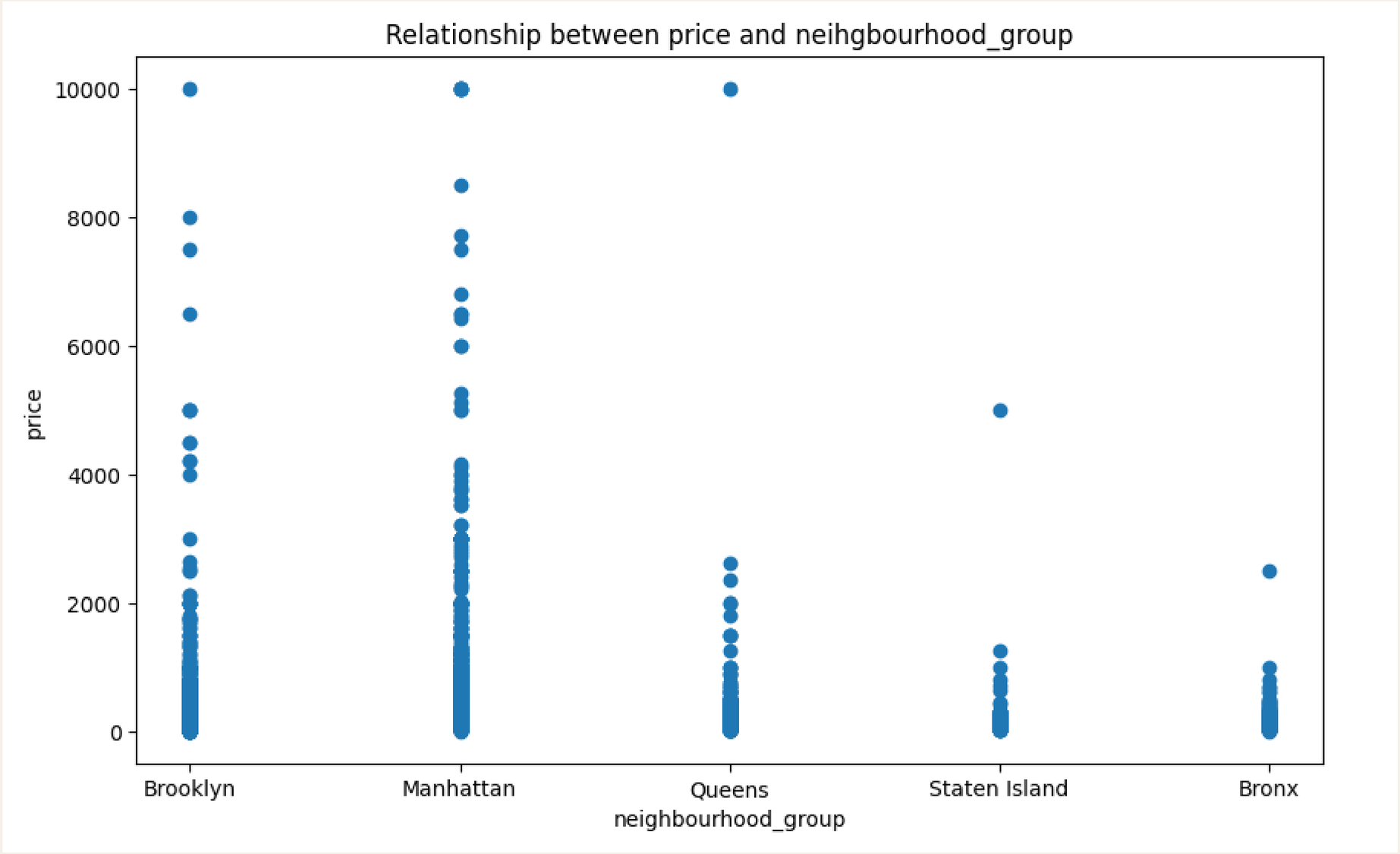
Some Data Distributions

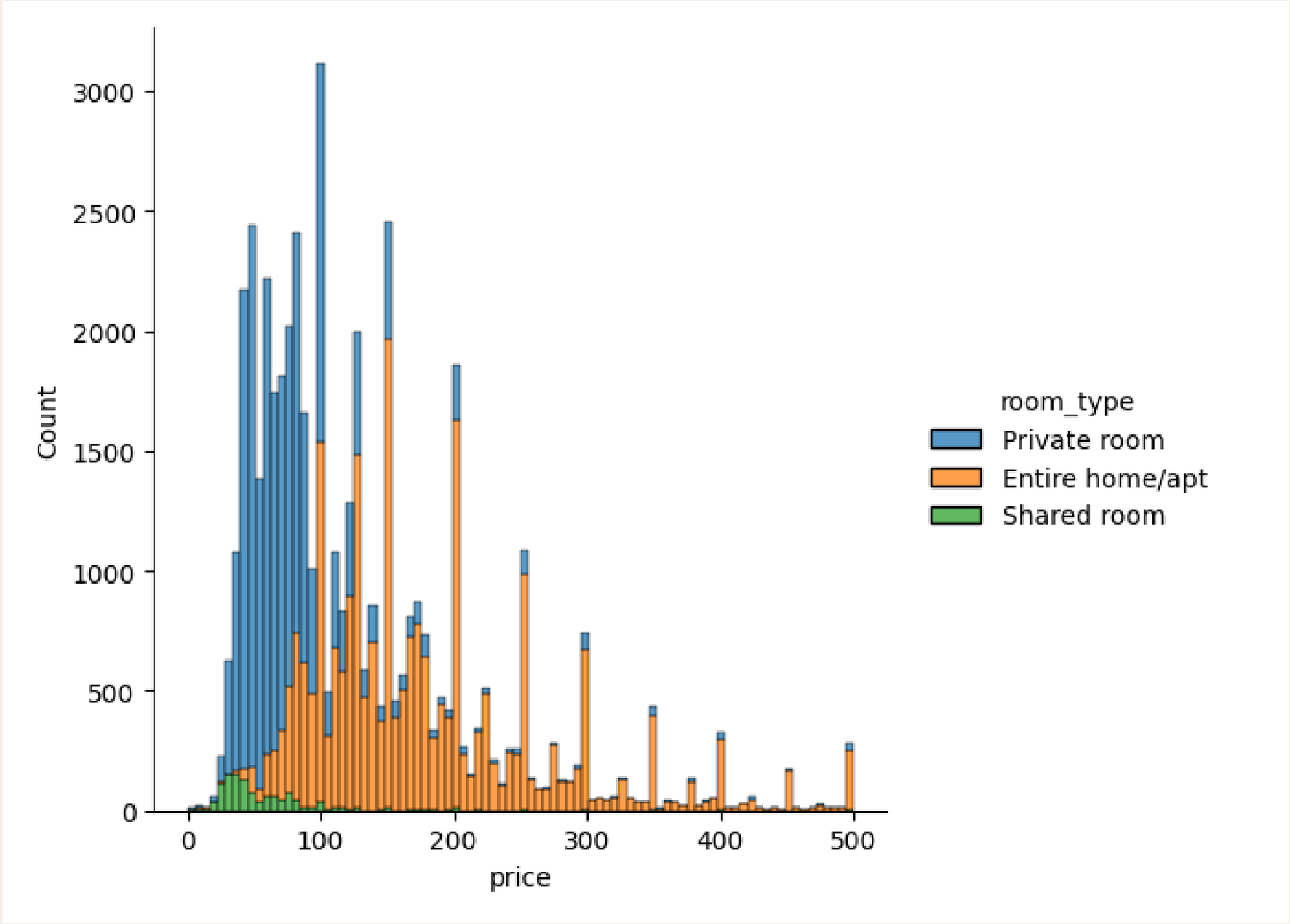



Missing Data Matrix
name
host_id
neighbourhood_group
neighbourhood
latitude
longitude
room_type
price
minimum_nights
number_of_reviews
reviews_per_month
calculated_host_listings_count
availability_365
1
48895

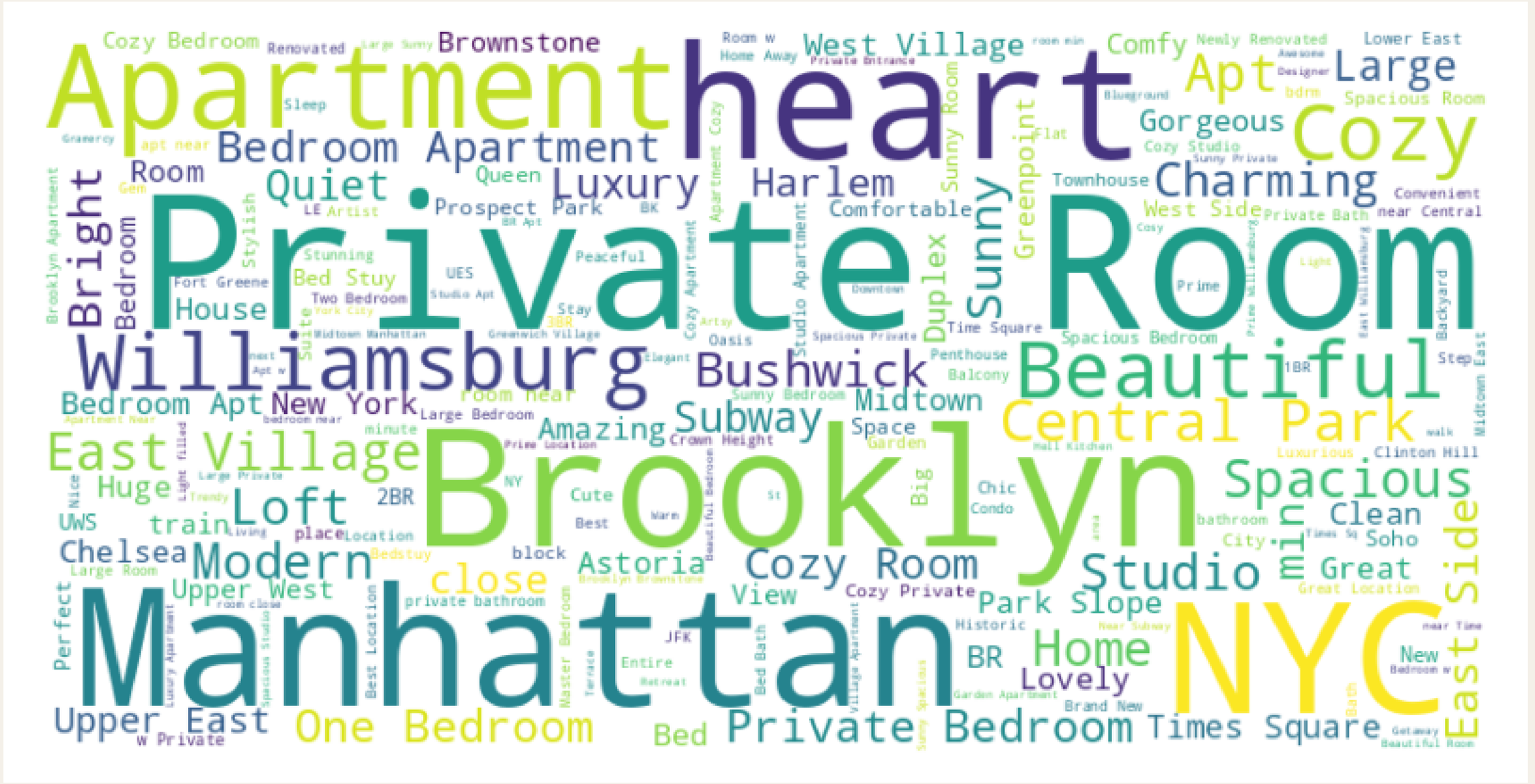

neighbourhood_group
Brooklyn
Manhattan
Queens
Staten Island
Bronx
40.9
40.8
40.7
40.6
40.5
latitude
−74.2
−74.1
−74.0
−73.9
−73.8
−73.7
longitude

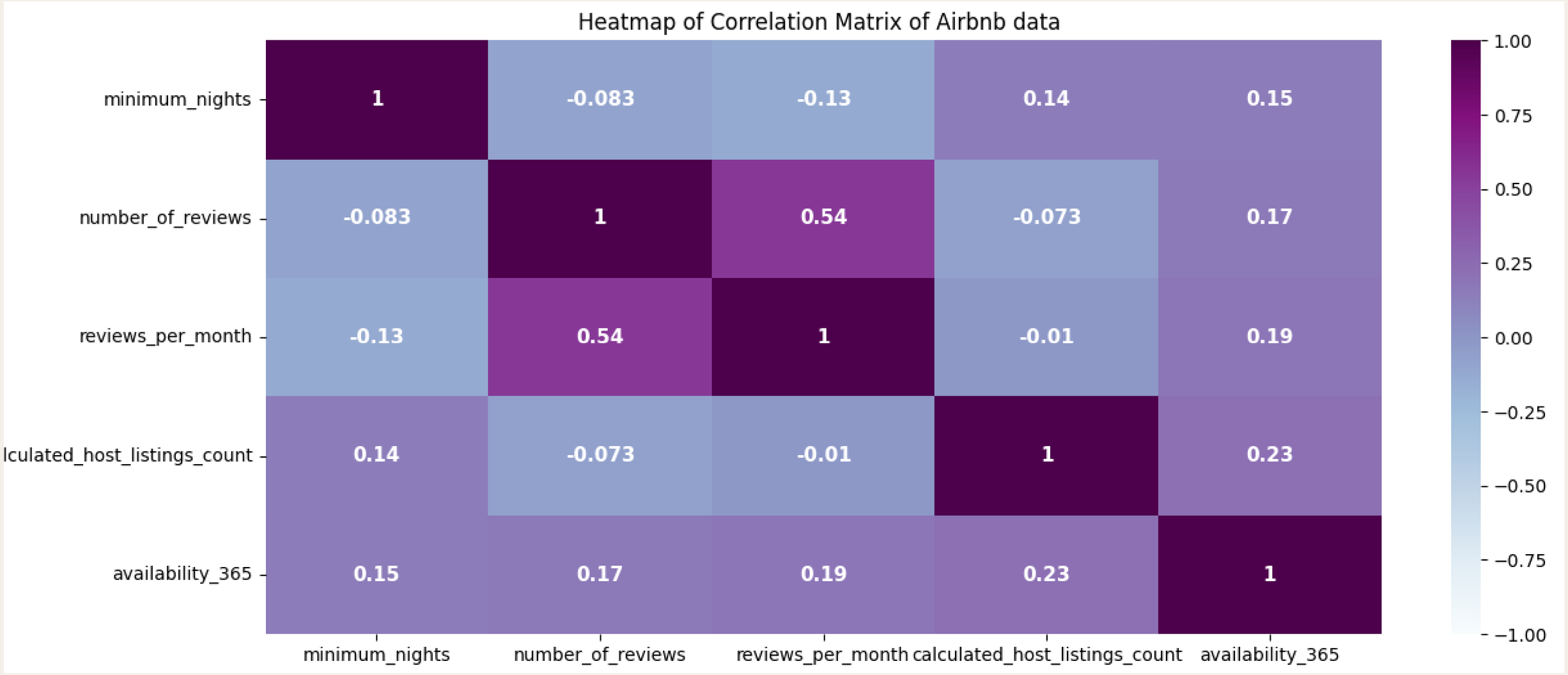
Some Relations between features




3000
2500
2000
1500
1000
500
0
Count
0
100
200
300
400
500
price
room_type
Private room
Entire home/apt
Shared room

Most common words in airbnb apartment descriptions in the name column




Heatmap of Correlation Matrix of Airbnb data
minimum_nights
number_of_reviews
reviews_per_month
lculated_host_listings_count
availability_365
1
-0.083
-0.13
0.14
0.15
-0.083
1
0.54
-0.073
0.17
-0.13
0.54
1
-0.01
0.19
0.14
-0.073
-0.01
1
0.23
0.15
0.17
0.19
0.23
1
minimum_nights
number_of_reviews
reviews_per_month
calculated_host_listings_count
availability_365
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

2 - Preprocessing-1

Operations on the dataset before including it in the price prediction model

- Some features,which any information when for making price predictions, dropped this dataset (id, last_review, host_id, name,host_name)
- The arguments in the dataset are separated as X, and the price property as the target value y
- We divided the x and y values into train and test at a ratio of 0.2.
- X values were divided into numeric and categorical values. Numeric values included long and lat values, which are spatial features.
- Numeric values should be filled empty values and scaled numerica vals.
- Categorical values must be encoded with OneHotEncoding. These operations were handled in one go with the ColumnTransform function by creating a pipeline.

2 - Preprocessing-2

Operations on the dataset before including it in the price prediction model

- During encoding, the neighborhood categorical property created very sparse matrix when encoded, which negatively affected the models (causing overfitting), so we removed this property.
- As a result of preprocess processes, our train data:

```
X_train_comb.head()
```

	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365	neighbourhood_group_Brooklyn	neighbourhood_group_Manhattan	neighbourhood_group_Queens	neighbourhood_group_Staten Island	room_type_Private room	room_type_Shared room
5714	-0.157660	-0.365699	-0.837354	-0.156983	-0.856544	0.0	1.0	0.0	0.0	1.0	0.0
14157	-0.316517	-0.522672	0.000000	-0.187282	-0.856544	1.0	0.0	0.0	0.0	1.0	0.0
3173	-0.263565	-0.388124	-0.837354	-0.187282	-0.856544	1.0	0.0	0.0	0.0	1.0	0.0
14196	-0.210612	-0.500247	-0.870563	-0.187282	1.506329	0.0	0.0	0.0	0.0	0.0	0.0
6094	-0.316517	-0.432973	-0.870563	-0.096385	0.397070	1.0	0.0	0.0	0.0	0.0	0.0

3- Price Prediction -Regression

Models were created to predict the price values in the test data by performing regression analysis over the Price target value by using the train dataset.
A prediction model was created using these models:

- Linear Regression,
- Lasso, Ridge,
- DecisionTreeRegressor,
- RandomForestRegresssor
- GradientBoostingRegressor

Also made hyperparametre tuning with GridSearch and RandomSearchCv to find the most optimal parameters of all models except Linear Regression.

3- Price Prediction -Regression Models

1. Linear Regression: Models linear relationships between variables.
2. Lasso Regression: Performs feature selection by shrinking coefficients towards zero.
3. Ridge Regression: Controls overfitting by shrinking coefficients using L2 regularization.
4. DecisionTreeRegressor: Decision tree regression is a regression algorithm that uses a decision tree to predict the value of the dependent variable. A decision tree divides the data based on features and makes predictions at the decision nodes. This method provides flexibility to model complex relationships.
5. RandomForestRegressor: Random forest regression is a regression model created by combining multiple decision trees. Each tree is trained on different subsets of the data, and the results are combined to make predictions. This method can calculate variable importance levels and reduce overfitting.
6. GradientBoostingRegressor: Gradient boosting regression is a method that combines multiple weak predictors (usually decision trees) to create a strong regression model. Gradient boosting adds trees sequentially, correcting the errors of the previous trees to improve predictions. This method is used to achieve high prediction performance.

3- Price Prediction -Regression

In linear regression models, the normal distribution of the target variable or closer to linear relationships can help the model perform better. Log transformation is a transformation method used to correct or clarify linear relationships on the target variable. The log transformation was also applied to the target values.

```
[22] y_train_log = np.log(y_train+1)
     y_test_log= np.log(y_test+1)
```

3- Price Prediction -Regression Conclusion -1

Models	Train Score	Test Score
Linear Regression	0.4675	0.4743
Ridge	0.4674	0.4743
Lasso	0.4674	0.4744
DecisionTreeRegressor	0.4710	0.4818
RandomSearchRegressor	0.4754	0.4862
GradientBoostingRegressor	0.5267	0.5556

3- Price Prediction -Regression Conclusion -2

```
# Print the best scores
print("Best R^2 Score of Train Data: ", grid_search.best_score_)
print("Best R^2 Score of Test Data: ", best_reg.score(X_test_comb, y_test_log))
```

```
Best R^2 Score of Train Data:  0.5266725505845173
Best R^2 Score of Test Data:  0.5556042788142574
```

Gradient Boosting creates a strong prediction model by sequentially adding weak predictors (usually decision trees). Each stage adds a new estimator to correct the errors of the previous stages. As this process continues by adding more stages, the model's ability to correct previous errors increases and prediction performance improves.

GradientBoostingRegressor has advantages such as the ability to capture complex relationships, good generalization, low error rate, and high prediction accuracy. However, the Gradient Boosting models took longer to train, especially the hyperparameter tuning part.

3- Price Prediction - Classification Preprocess

The target value in the dataset is a continuous value, but in order to experience the project-specific classification model, we gathered these values into 4 categories and used them as a classifier dataset.

While making the classification, the first attention was paid to the equal distribution of the classes, for this statistically it used the describe() function. The percentiles of the Train Dataset were divided by 4 and 4 intervals were created (fits), and the train and test target continuous values were converted into 4 classes according to these intervals.

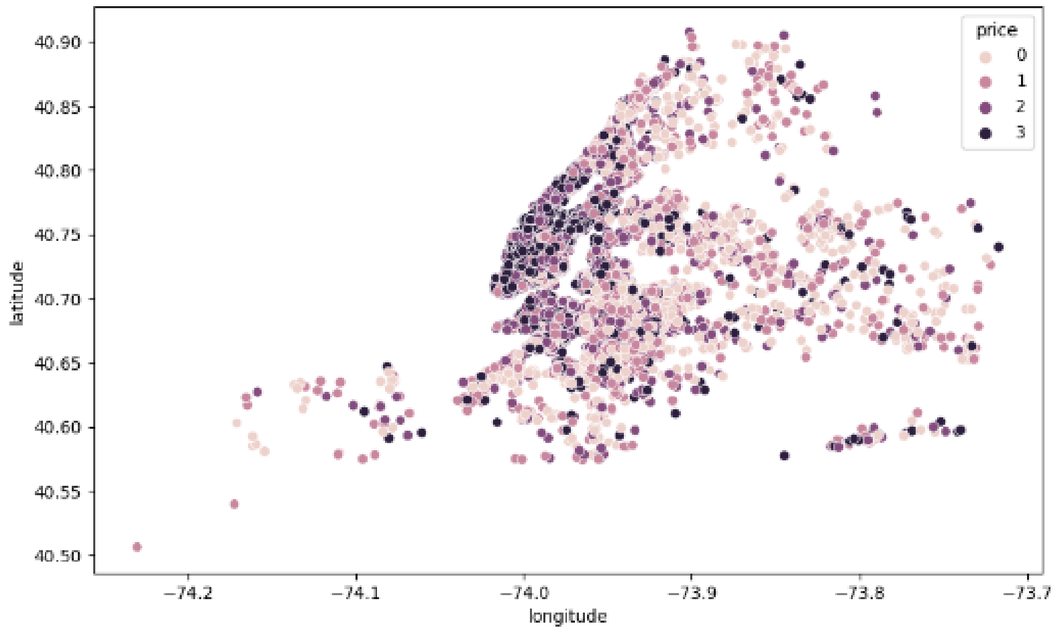
```
category_1 = (y_train_clsf.describe().loc["min"], y_train_clsf.describe().loc["25%"])
category_2 = (y_train_clsf.describe().loc["25%"], y_train_clsf.describe().loc["50%"])
category_3 = (y_train_clsf.describe().loc["50%"], y_train_clsf.describe().loc["75%"])
category_4 = (y_train_clsf.describe().loc["75%"], y_train_clsf.describe().loc["max"])

y_train_clsf.loc[y_train_clsf.between(category_1[0], category_1[1])] = 0
y_train_clsf.loc[y_train_clsf.between(category_2[0], category_2[1])] = 1
y_train_clsf.loc[y_train_clsf.between(category_3[0], category_3[1])] = 2
y_train_clsf.loc[y_train_clsf.between(category_4[0], category_4[1])]= 3
```

```
y_test_clsf.loc[y_test_clsf.between(category_1[0], category_1[1])] = 0
y_test_clsf.loc[y_test_clsf.between(category_2[0], category_2[1])] = 1
y_test_clsf.loc[y_test_clsf.between(  category_3[0], category_3[1])] = 2
y_test_clsf.loc[y_test_clsf.between(category_4[0], category_4[1])]= 3
```

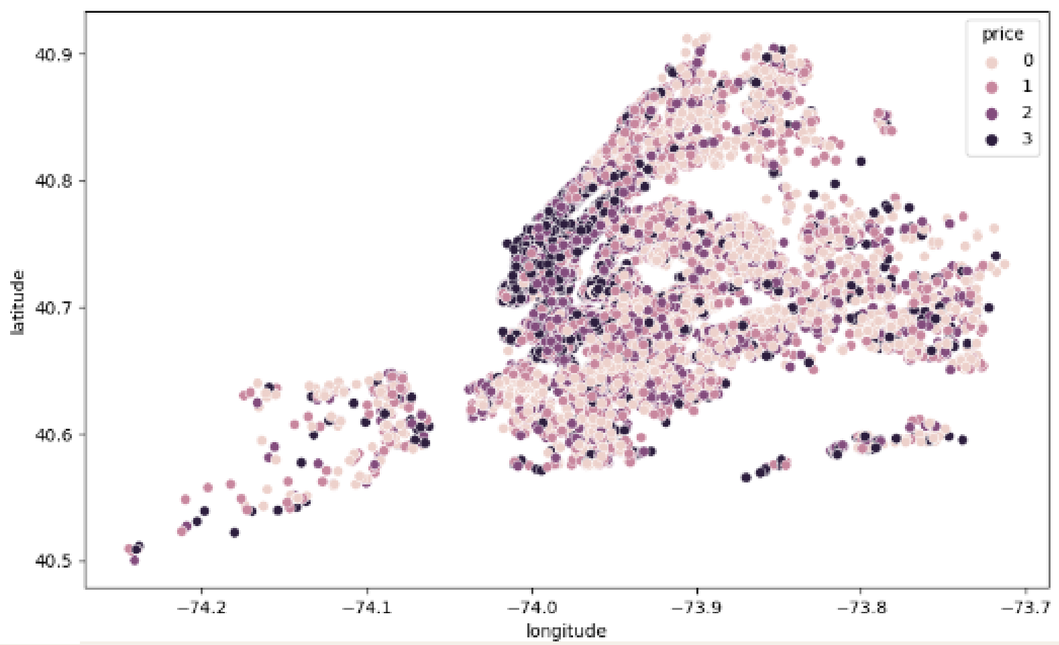
3- Price Prediction - Classification Preprocess

After re-created target value for classification, map view of classes:

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_test.longitude, y=X_test.latitude, hue=y_test_clsf)
plt.ioff()
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_train.longitude, y=X_train.latitude, hue=y_train_clsf)
plt.ioff()
plt.show()
```



3- Price Prediction - Classification

After re-created target value for classification, various models were tried and the most accurate model was obtained by tuning these models.

- Logistic Regression,
- Gaussian Naive Bayes,
- Support Vector Machine (SVM),
- K-Nearest NeighborsClassifier
- Decision Tree Classifier,
- RandomForest Cllassifier, Also made hyperparametre tuning with GridSearch and
- RandomSearchCv to find the most optimal parameters of all models except Logistic Regression.

3- Price Prediction - Classification Models - 1

- Logistic Regression: Logistic regression is a statistical model used for classification problems. It calculates the probability of a data point belonging to a certain class using a logistic function. It is commonly used for binary classification or multi-class classification tasks.
- Gaussian Naive Bayes: Gaussian Naive Bayes is a classification algorithm based on Bayes' theorem. It assumes that features are independent and follows a Gaussian (normal) distribution. It calculates the probability of a data point belonging to each class and uses these probabilities to make predictions.
- Support Vector Machine (SVM): Support Vector Machine is a classification algorithm that finds an optimal hyperplane to separate data points of different classes. It aims to maximize the margin between the classes. SVM can handle linear and non-linear classification problems and performs well in high-dimensional datasets.

3- Price Prediction - Classification Models - 2

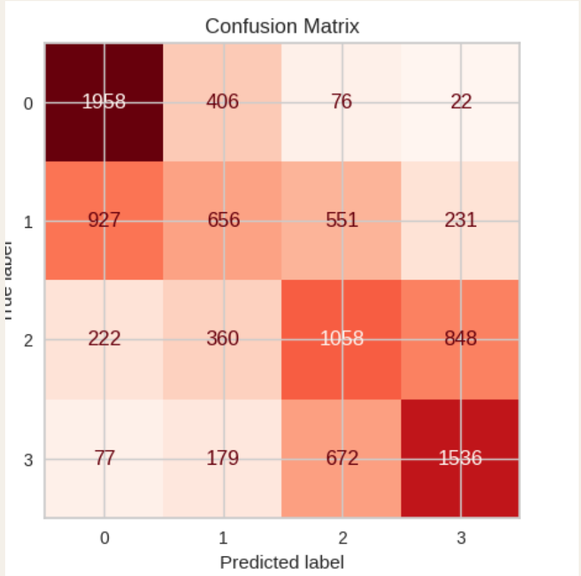
- K-Nearest Neighbors Classifier: K-Nearest Neighbors Classifier is a simple and effective algorithm for classification. It classifies a new data point based on the majority vote of its K nearest neighbors in the feature space. The value of K is a user-defined parameter.
- Decision Tree Classifier: Decision Tree Classifier represents data in a tree-like structure for classification tasks. Each internal node represents a feature and a decision point, while the leaf nodes hold the class labels. The decision tree branches based on the decision points in the data to classify new instances.
- RandomForest Classifier: RandomForest Classifier is an ensemble learning algorithm that combines multiple decision trees. It creates a set of decision trees using different random samples and feature subsets. Each tree independently classifies data points, and the final prediction is made by aggregating the predictions of all the trees. RandomForest provides high performance, robustness, and generalization capabilities in classification problems.

3- Price Prediction -Classification Conclusion -1

Models	Train Score	Test Score
Logistic Regression	0.5226	0.5296
Gaussian Naive Bayes	0.4943	0.5015
Support Vector Machine	0.5361	0.5326
K-Nearest NeighborsClassifier	0.6283	0.4923
Decision Tree Classifier	0.5243	0.5301
RandomForest Cllassifier	0.5267	0.5556

3- Price Prediction -Classification Conclusion -2

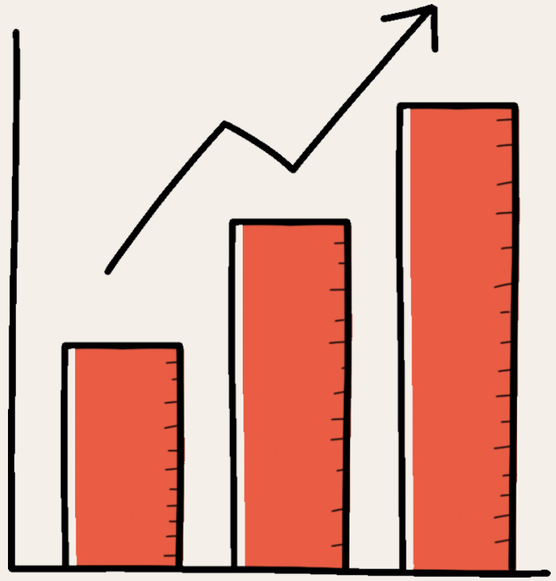
```
Train Accuracy: 0.5360721955210144
Test Accuracy: 0.532569724123121
```



Although the accuracies as a result of the classification models were quite close to each other, SVM performed better than other classification algorithms. SVM uses support vectors to clearly separate the classes and therefore may be less sensitive to noise or discarded data in the data. SVM performs well on high-dimensional datasets. can show. Because SVM only classifies with support vectors and these support vectors focus on informative features to separate classes. This can provide better performance than other algorithms as the number of features increases. Although computational complexity and parameter selection are sometimes disadvantages, this was the model that gave the best results on the dataset.

4- Conclusion

Although our dataset did not give the accuracy values we expected, we tried to increase the success rate by trying different models and hyperparameter tunings.



Thanks For Listening