CP322 Report

European Airbnb Prices

Anusaanth Sivakumaran

180886030

Abstract

The purpose of this report is to investigate the use of regression models to forecast Airbnb prices in European countries. The dataset was subjected to five distinct regression models: Linear, Ridge, Lasso, Decision Tree, and Random Forest. The Random Forest model gave the best accuracy, with an R2 score of 0.96 for training and 0.78 for testing. The model also identified the top ten most important features that influence price prediction. This study's findings show that regression models may be used to reliably estimate Airbnb prices, with the Random Forest model being the most effective in this context. Future research could look into using more powerful machine learning techniques or including other variables to increase forecast accuracy.

Introduction

Airbnb has transformed the way people travel and interact with individuals from all over the world. People may now locate affordable and distinctive housing options in several locations more easily than ever before thanks to its online marketplace platform. With so many elements influencing Airbnb rental pricing, both hosts and consumers may find it challenging to determine the most accurate and fair price for a listing. To address this issue, I investigated an Airbnb price dataset for numerous European countries. The dataset includes crucial factors such as room type, capacity, super host status, ratings, and location. I plan to employ a variety of machine learning algorithms to identify the characteristics that have the most influence on Airbnb rental rates and to build a prediction model that can accurately anticipate rental costs. The study's purpose is to provide information and guidance to Airbnb hosts on how to price

their listings appropriately based on their location and the amenities they offer. Furthermore, this study may help local clients and visitors understand the key characteristics to look for when renting an Airbnb home. We may get significant insights into the elements that drive Airbnb pricing by using the power of machine learning, which can ultimately lead to more informed and equitable pricing decisions.

I will begin my report by defining the dataset I utilized and give an overview of the many variables included in the dataset. Next, I'll go over the various machine learning techniques I used, such as linear regression, ridge regression, lasso regression, decision trees, and random forests. I will provide the outcomes of each model and discover which variables have the greatest influence on Airbnb rental prices. Finally, I will make recommendations to Airbnb hosts and customers based on the findings of this study.

Data Collection

I started this analysis by seeking for interesting datasets on several web sites, such as Kaggle. After looking through a few datasets, I came across one that peaked my interest: it had data on Airbnb prices in numerous European countries. Each cities dataset was divided into separate files for weekdays and weekends, such as Paris weekdays.csv and Paris weekend.csv. To create a thorough dataset for this study, I started by combining all the individual city files into one by using a function I created that combined the data from all the files into a single CSV file (Figure B). My dataset included the cities of Athens, Barcelona, Berlin, Lisbon, London, Paris, Rome, and Vienna (Figure A).

The combined dataset offered a variety of information that was useful in developing the predictive models. RealSum, room type, room shared, room private, person capacity, host is superhost, multi, business, cleanliness rating, guest satisfaction overall, bedrooms, dist, metro dist, lng, and lat were among the columns in the dataset. While the majority of these features are self-explanatory, a few do require further explanation. RealSum, for example, shows the total price of the Airbnb listing, whereas room type indicates the type of available room, such as private or shared. Also, multi denotes whether the listing is for many rooms, whilst biz denotes listings that are better suitable for business purposes. The distance from the city centre is represented by dist, whereas the distance from the nearest metro station is represented by metro dist. Finally, lng and lat represent the Airbnb listing's longitude and latitude.

Exploratory data analysis

In the initial stages of data pre-processing, I began by using the columns and head functions to obtain a better understanding of the columns and the data they contained. Throughout this process, I discovered an unnecessary column called "Unmade: 0" in the dataset. As a result, in order to streamline the data and make it easier to deal with, I removed this column from the dataset. To further optimize the dataset for machine learning techniques, I changed the categorical variables to numerical values. For example, in order for the algorithms to properly handle them, I altered the week time column such that the weekend variable is 0 and the weekday variable is 1. In addition, for the column room type I changed the variable "Entire home/apt" to 1 and all other room kinds, such as private and shared, to 0. This conversion aided in reducing performance concerns that often come when dealing with categorical data. Then, using the "isna" function, I found and addressed any null values in the

dataset (Figure C, For Output). Following that, I used methods like "describe" and "value counts" to get a full overview of the information, including the amount of data each city had (Figure D, For Output). These functions provided valuable data insights, assisting me in identifying potential outliers, and other data-related issues that needed to be addressed.

Finally, the pre-processed dataset was saved to a new file in the data folder. This phase was critical in ensuring that the dataset was clean, complete, and ready for analysis, laying the framework for the predictive models to be built.

Methodology

I obtained dummy variables for the column city to begin training the dataset. I chose the features to use using a correlation heatmap. A correlation heatmap is a graphical tool that may be used to determine the relationship between variables in a dataset. It assesses the strength and direction of a two-variable linear relationship. In this example, you could identify which factors were significantly connected and could be possible predictors by evaluating the heatmap. I opted not to utilize the variables room shared and room private because this information was already in the column room type. Also, I skipped the column bedrooms because the column multi indicates whether or not the listing has several bedrooms. The final variables I decided not to use were longitude and latitude, which I thought were unnecessary given that we already know which country the city is in and how far the listing is from the city centre. After deciding which features to utilize, I saved those columns as dependent variables in an x variable and created a y variable with the realSum (Price) column as the independent variable. I used the function "train test split" to train and split the data with the newly formed x

and y variables. Finally, in the train and test data, I scaled the numerical features (Figure E). Scaling numeric features is significant in machine learning since it ensures that all variables are on the same scale and do not contribute more to the model because of greater numeric ranges. Standardizing the variables also avoids outliers from having a disproportionate impact on the model's performance.

A linear regression model was utilized as the first machine learning model. This is a type of predictive analysis that is frequently used. A regression analysis can be performed to examine if a set of predictor variables correctly predicts an outcome variable. It also tells you which variables are significant predictors of the outcome variable. The variable to be forecasted is referred to as the dependent variable. The independent variable is the one that predicts the values of the other variables. This technique's main applications are determining predictor strength, anticipating an effect, and trend forecasting. I chose this method because I believe it offers various benefits. This model can also be quickly and readily trained. It also has a far lower time complexity than other models. Due to having a large amount of data, I reasoned that the lower time complexity would be useful. After fitting the linear regression model to the dataset, we obtained r2 scores of 0.45 and 0.51 for the training and testing sets, respectively (Figure F). The r2 score indicates how well the model matches the data, with higher values indicating a better fit. The explained variance of 0.51 indicates that the independent variables employed in the model can explain 51% of the variation in the price of Airbnb listings. Furthermore, the mean absolute error of 2.45 implies that the model's anticipated price is on average $2.45 higher than the actual price. The linear regression model's root mean squared error of 13.59 and mean squared error of 3.69 represent the average difference between the predicted and

actual values of the dependant variable. The lower the values of these metrics, the greater the predicted accuracy of the model. These findings imply that the linear regression model performs poorly.

Following that, the European countries' Airbnb pricing dataset was subjected to two regularization techniques: ridge and lasso regression. By inserting a penalty term, both of these models were used to prevent overfitting in linear regression. The shrinkage approach is used in the Lasso regression, which determines the coefficients and then shrinks them towards the central point as a mean. Ridge regression, on the other hand, is similar to Lasso, but instead of taking the magnitude of the coefficients, it takes the square. The ridge regression model produced a r2 of 0.45 for training and 0.51 for testing. The variance explained was 0.51 and the mean absolute error was 2.45. The root square error and root mean square error were 13.59 and 3.69, respectively. The parameter alpha, which controls the strength of regularization, was set to 3.05. The Lasso regression model, on the other hand, has a r2 score of 0.45 for training and 0.51 for testing. The explained variance was 0.51 and the mean absolute error was 2.44. The mean squared error and the root mean squared error were 13.63 and 3.69, respectively. The Lasso regression model had an alpha value of 0.01. Overall, the results of both models are quite similar to those of the linear regression model, indicating that regularization had little effect on the model's predictive power.

The decision tree regression model is an effective and efficient technique for creating a regression model with a tree structure. This approach breaks the data down into smaller groups and applies a simple model to each of them. One of the benefits of this technique is that it is highly interpretable, making it simple to understand which variables are important in predicting

the outcome. Another benefit of decision trees is that they can deal with both numerical and categorical data and are unaffected by outliers or missing values. Decision trees, in contrast to other techniques, involve less effort for data preparation and do not require data normalization. Since the model is very fast, it is excellent for swiftly investigating associations between variables, particularly when dealing with a large dataset. The decision tree regression model produced results with an R2 score of 1 for training and 0.71 for testing, showing a very excellent fit to the training data and good generalization to the testing data. The mean absolute error was 1.34, and the explained variance was 0.71. The root square error was 7.98 and the root mean square error was 2.82, respectively. These findings indicate that the decision tree regression model performed well in forecasting the prices of Airbnb listings in European countries based on the input variables.

The final regression model used for this study was a random forest regression. In dealing with complex data structures, this model has proven to be a reliable and robust machine learning technique. It combines ensemble learning methods with the decision tree framework to create several decision trees, each constructed from a random subset of the training data. These trees are then joined to give a final forecast, which frequently yields accurate results. The random forest regression model has several advantages over other methods. It is frequently highly accurate and can easily manage a huge number of variables. It is also less prone to overfitting than other models, which makes it a popular choice among data scientists. The random forest regression model provided the highest accuracy among all models used in this research when applied to the European Airbnb dataset. The training r2 score was 0.96, suggesting that the model explained 96% of the variance in the training data. The r2 score for

test data was 0.78, indicating the model's ability to generalize to previously unseen data. The mean absolute error was 1.48, while the explained variance was 0.78. The root mean square error was 2.46, and the root square error was 6.07. Due to the successful performance, some visualizations were made to further investigate the model. A scatter plot of predicted vs. actual Airbnb prices shows a linear relationship between the two, indicating a strong correlation (Figure G). A correlation heatmap of the variables was also developed to demonstrate the relationship between the variables (Figure H). Finally, a bar chart was developed that displayed the top ten important features in the random forest model, which comprised multi, city Barcelona, city Paris, city London, guest satisfaction overall, person capacity, room type, city Athens, metro dist, and dist (Figure I). These features may be used in the future to optimize pricing strategy.

Recommendation

I would advise hosts who desire higher-rent Airbnb properties to buy in places such as Barcelona, Paris, London, and Athens. They should also look for properties that are closer to the metro station and the city centre. Furthermore, they should ensure that their property can accommodate a reasonable number of people. Local clients and vacationers who want to save money on their trip should avoid the above-mentioned cities. They can also save money if they have other modes of transportation, such as a vehicle or a bike, by picking a location that is further from the metro station and city center. Also, getting a listing with a shared room or going with fewer people will help you save even more money.

Conclusion

Finally, the regression models utilized in this investigation analysis predicted Airbnb prices for European countries with promising results. For huge datasets, the decision tree and random forest models in particular displayed impressive accuracy and efficiency. The Lasso and Ridge models performed well as well, though not as well as the ensemble learning models. The findings reveal that dist, metro dist, city Athens, room type, and person capacity are critical factors in deciding Airbnb prices.

In terms of future work, it would be interesting to investigate more advanced regression approaches, such as neural networks or support vector regression, to enhance prediction accuracy even further. Furthermore, integrating new features into the algorithms, such as area demographics or Airbnb listing attributes, could result in more complex and accurate forecasts. Finally, additional research might be conducted to investigate the relationship between Airbnb prices and external factors such as economic indicators or regional events in order to better understand the market dynamics affecting the European Airbnb industry.

Appendix

Figure A

```python
# -*- coding: utf-8 -*-
import logging
from pathlib import Path
from dotenv import find_dotenv, load_dotenv
import pandas as pd


def main():
    """ Runs data processing scripts to turn raw data from (../raw) into
        cleaned data ready to be analyzed (saved in ../processed).
    """
    logger = logging.getLogger(__name__)
    logger.info('making final data set from raw data')

    # Load raw data
    output_filepath = '../data/raw/euro_data.csv'
    athens_weekdays = pd.read_csv('../data/raw/athens_weekdays.csv')
    athens_weekends = pd.read_csv('../data/raw/athens_weekends.csv')
    athens = comb(athens_weekdays, "weekdays", athens_weekends, "weekends", "athens")

    barcelona_weekdays = pd.read_csv('../data/raw/barcelona_weekdays.csv')
    barcelona_weekends = pd.read_csv('../data/raw/barcelona_weekends.csv')
    barcelona = comb(barcelona_weekdays, "weekdays", barcelona_weekends, "weekends", "barcelona")

    berlin_weekdays = pd.read_csv('../data/raw/berlin_weekdays.csv')
    berlin_weekends = pd.read_csv('../data/raw/berlin_weekends.csv')
    berlin = comb(berlin_weekdays, "weekdays", berlin_weekends, "weekends", "berlin")

    lisbon_weekdays = pd.read_csv('../data/raw/lisbon_weekdays.csv')
    lisbon_weekends = pd.read_csv('../data/raw/lisbon_weekends.csv')
    lisbon = comb(lisbon_weekdays, "weekdays", lisbon_weekends, "weekends", "lisbon")

    london_weekdays = pd.read_csv('../data/raw/london_weekdays.csv')
    london_weekends = pd.read_csv('../data/raw/london_weekends.csv')
    london = comb(london_weekdays, "weekdays", london_weekends, "weekends", "london")

    paris_weekdays = pd.read_csv('../data/raw/paris_weekdays.csv')
    paris_weekends = pd.read_csv('../data/raw/paris_weekends.csv')
    paris = comb(paris_weekdays, "weekdays", paris_weekends, "weekends", "paris")

    rome_weekdays = pd.read_csv('../data/raw/rome_weekdays.csv')
    rome_weekends = pd.read_csv('../data/raw/rome_weekends.csv')
    rome = comb(rome_weekdays, "weekdays", rome_weekends, "weekends", "rome")

    vienna_weekdays = pd.read_csv('../data/raw/vienna_weekdays.csv')
    vienna_weekends = pd.read_csv('../data/raw/vienna_weekends.csv')
    vienna = comb(vienna_weekdays, "weekdays", vienna_weekends, "weekends", "vienna")

    cities = [athens,barcelona,berlin,lisbon,london,paris,rome,vienna]

```

Figure B

```python
    euro_data = pd.concat(cities, ignore_index=True)

    euro_data.to_csv(output_filepath, index=False)


if __name__ == '__main__':
    log_fmt = '%(asctime)s - %(name)s - %(levelname)s - %(message)s'
    logging.basicConfig(level=logging.INFO, format=log_fmt)

    # not used in this stub but often useful for finding various files
    project_dir = Path(__file__).resolve().parents[2]

    # find .env automagically by walking up directories until it's found, then
    # load up the .env entries as environment variables
    load_dotenv(find_dotenv())

    main()

#Function to combine weekend and weekdays dataset for each city
def comb(csv, col, csv2, col2, city):
    csv["week_time"] = col
    csv2["week_time"] = col2

    merg = pd.concat([csv,csv2])
    merg["city"] = city

    return merg

```

Figure C

```
--------------------------------------------------------------------------------------------

Pre-processing:

Index(['Unnamed: 0', 'realSum', 'room_type', 'room_shared', 'room_private',
       'person_capacity', 'host_is_superhost', 'multi', 'biz',
       'cleanliness_rating', 'guest_satisfaction_overall', 'bedrooms', 'dist',
       'metro_dist', 'attr_index', 'attr_index_norm', 'rest_index',
       'rest_index_norm', 'lng', 'lat', 'week_time', 'city'],
      dtype='object')
   Unnamed: 0     realSum        room_type  room_shared  room_private  person_capacity  host_is_superhost  ...  attr_index_norm  rest_index  rest_index_norm       lng       lat  week_time    city
0           0  129.824479  Entire home/apt        False         False              4.0              False  ...         2.086871   78.778377         5.915160  23.76600  37.98300   weekdays  athens
1           1  138.963748  Entire home/apt        False         False              4.0               True  ...         9.060559  407.167696        30.572629  23.73168  37.97776   weekdays  athens
2           2  156.304923  Entire home/apt        False         False              3.0               True  ...         7.522257  395.967403        29.731642  23.72200  37.97900   weekdays  athens
3           3   91.627024  Entire home/apt        False         False              4.0               True  ...         1.500740   58.706578         4.408047  23.72712  38.01435   weekdays  athens
4           4   74.051508     Private room        False          True              2.0              False  ...         2.968577  113.325973         8.509204  23.73391  37.99529   weekdays  athens

[5 rows x 22 columns]
realSum                     0
room_type                   0
room_shared                 0
room_private                0
person_capacity             0
host_is_superhost           0
multi                       0
biz                         0
cleanliness_rating          0
guest_satisfaction_overall  0
bedrooms                    0
dist                        0
metro_dist                  0
attr_index                  0
attr_index_norm             0
rest_index                  0
rest_index_norm             0
lng                         0
lat                         0
week_time                   0
city                        0
dtype: int64
```

Figure D

```
            realSum     room_type  person_capacity         multi           biz  cleanliness_rating  ...  attr_index_norm    rest_index  rest_index_norm           lng           lat     week_time
count  45605.000000  45605.000000     45605.000000  45605.000000  45605.000000        45605.000000  ...     45605.000000  45605.000000     45605.000000  45605.000000  45605.000000  45605.000000
mean     275.621575      0.612499         3.145554      0.290670      0.361495            9.379542  ...        13.452285    658.027403        21.599516      6.515594     45.204614      0.489486
std      327.241489      0.487185         1.305563      0.454076      0.480439            0.969511  ...        10.009584    515.320827        17.286099      9.788916      5.355912      0.499895
min       42.884259      0.000000         2.000000      0.000000      0.000000            2.000000  ...         1.040228     19.576924         0.592757     -9.226340     37.953000      0.000000
25%      150.328478      0.000000         2.000000      0.000000      0.000000            9.000000  ...         6.168156    266.790387         8.171002     -0.106860     41.377000      0.000000
50%      212.954128      1.000000         3.000000      0.000000      0.000000           10.000000  ...        11.423793    552.776514        16.225274      2.356000     41.935000      0.000000
75%      315.267033      1.000000         4.000000      1.000000      1.000000           10.000000  ...        17.555840    882.844937        31.433403     12.562000     51.459150      1.000000
max    18545.450285      1.000000         6.000000      1.000000      1.000000           10.000000  ...       100.000000   6696.156772       100.000000     23.786020     52.641410      1.000000

[8 rows x 17 columns]
london       9993
rome         9027
paris        6688
lisbon       5763
athens       5280
vienna       3537
barcelona    2833
berlin       2484
Name: city, dtype: int64
```

Figure E

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

def train():

    #Input file and read.csv
    input_file = '../data/processed/euro_data_proc.csv'
    euro_data = pd.read_csv(input_file)

    #Create dummy variable
    euro_data = pd.get_dummies(data=euro_data, columns=['city'])

    #Features
    feature_col =['room_type', 'person_capacity', 'host_is_superhost', 'multi', 'guest_satisfaction_overall', 'dist', 'metro_dist', 'week_time',
                  'city_athens', 'city_barcelona', 'city_berlin' , 'city_lisbon', 'city_london', 'city_paris', 'city_rome', 'city_vienna']

    #x and y fir train test split
    x =  euro_data[feature_col]
    y = euro_data["realSum"]

    #Train and split data
    X_train, X_test, y_train, y_test = train_test_split(x, np.sqrt(y),  test_size=0.15,random_state= 42)

    #Scaling numeric features using sklearn StandardScalar
    numeric=['person_capacity', 'guest_satisfaction_overall', 'dist','metro_dist']
    sc=StandardScaler()
    X_train[numeric]=sc.fit_transform(X_train[numeric])
    X_test[numeric]=sc.transform(X_test[numeric])

    return X_train, X_test, y_train, y_test
```

Figure F

```
Visualization:

##################################################

Linear Regression
-10352122319762.324

Coeffients:
room_type = 3.2262044135544237
person_capacity = 1.6515339188297378
host_is_superhost = 0.16922156283056916
multi = -0.00067643328144989
guest_satisfaction_overall = 0.24893129511508372
dist = -1.1627317568741664
metro_dist = -0.09338148106975583
week_time = -0.12617262499438023
city_athens = 10352122319769.666
city_barcelona = 10352122319777.992
city_berlin = 10352122319777.457
city_lisbon = 10352122319774.383
city_london = 10352122319780.098
city_paris = 10352122319778.969
city_rome = 10352122319773.895
city_vienna = 10352122319774.389

(('room_type', 3.2262044135544237), ('person_capacity', 1.6515339188297378), ('host_is_superhost', 0.16922156283056916), ('multi', -0.00067643328144989), ('guest_satisfaction_overall', 0.24893129511508372
('dist', -1.1627317568741664), ('metro_dist', -0.09338148106975583), ('week_time', -0.12617262499438023), ('city_athens', 10352122319769.666), ('city_barcelona', 10352122319777.992), ('city_berlin', 10352
19777.457), ('city_lisbon', 10352122319774.383), ('city_london', 10352122319780.098), ('city_paris', 10352122319778.969), ('city_rome', 10352122319773.895), ('city_vienna', 10352122319774.389))

Regression: R^2 score on training set 0.44875797032408027
Regression: R^2 score on test set 0.5068980981655297
Explained_variance 0.5069526290015816
Mean absolute Error:  2.44854761618175
Root Square Error:  13.589674635602583
Root Mean Square Error:  3.6864175883372985
##################################################

Ridge Regression

(('room_type', 3.2223365163837334), ('person_capacity', 1.6504340811292l8), ('host_is_superhost', 0.16926821193754554), ('multi', -0.00328510878453489), ('guest_satisfaction_overall', 0.2475949882074875
dist', -1.1646786300932601), ('metro_dist', -0.09227070996349859), ('week_time', -0.12317294246518325), ('city_athens', -6.182588047142649), ('city_barcelona', 2.1207514132049576), ('city_berlin', 1.62046
8971017), ('city_lisbon', -1.4634851980965755), ('city_london', 4.24754429670633l), ('city_paris', 3.1009837379119105), ('city_rome', -1.9696967185205103), ('city_vienna', -1.47396974693999))
```

Figure G

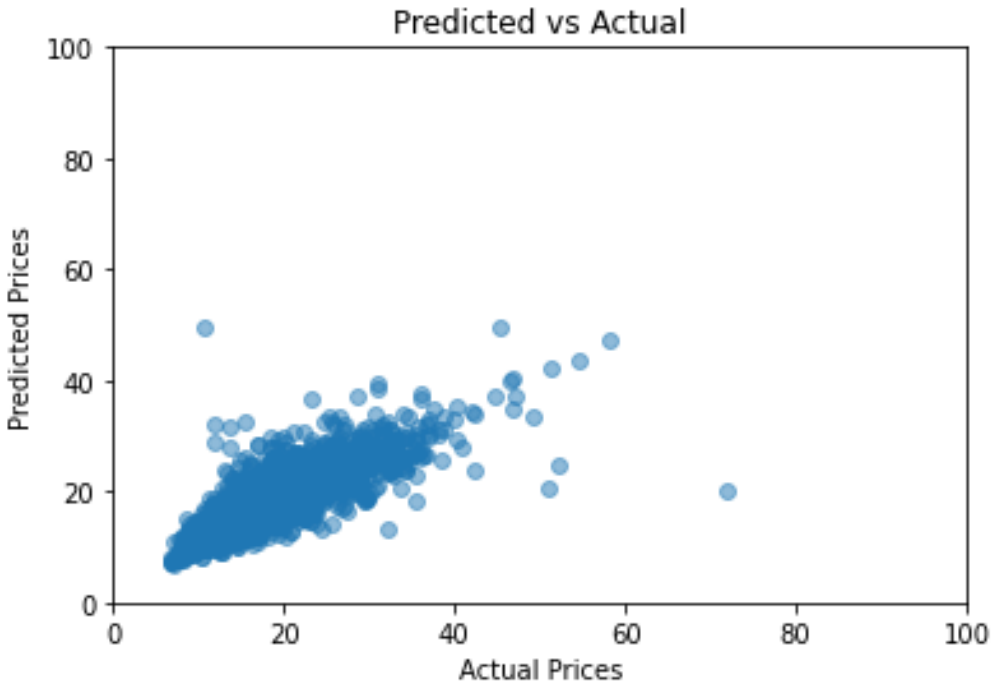


Figure H

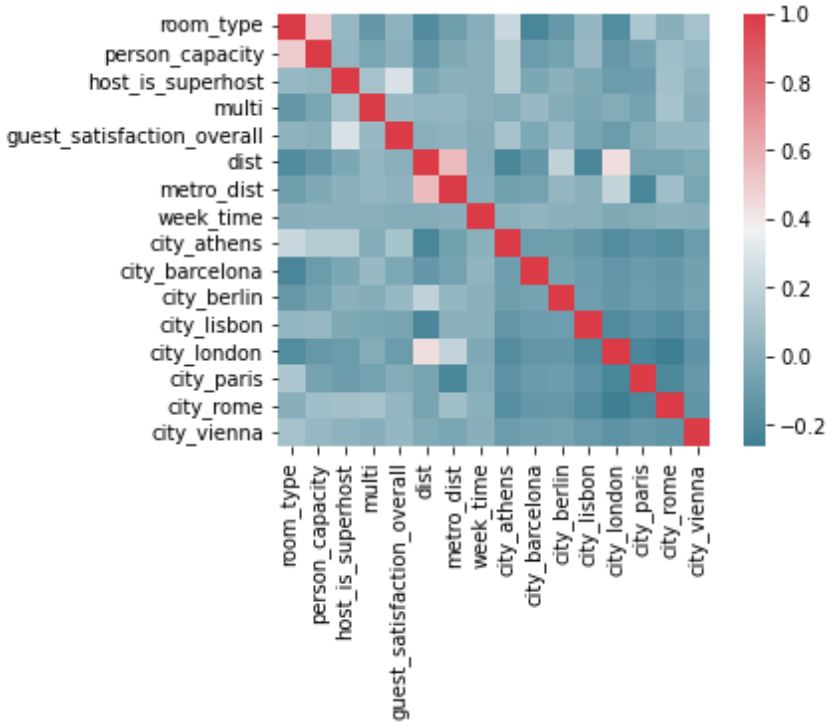
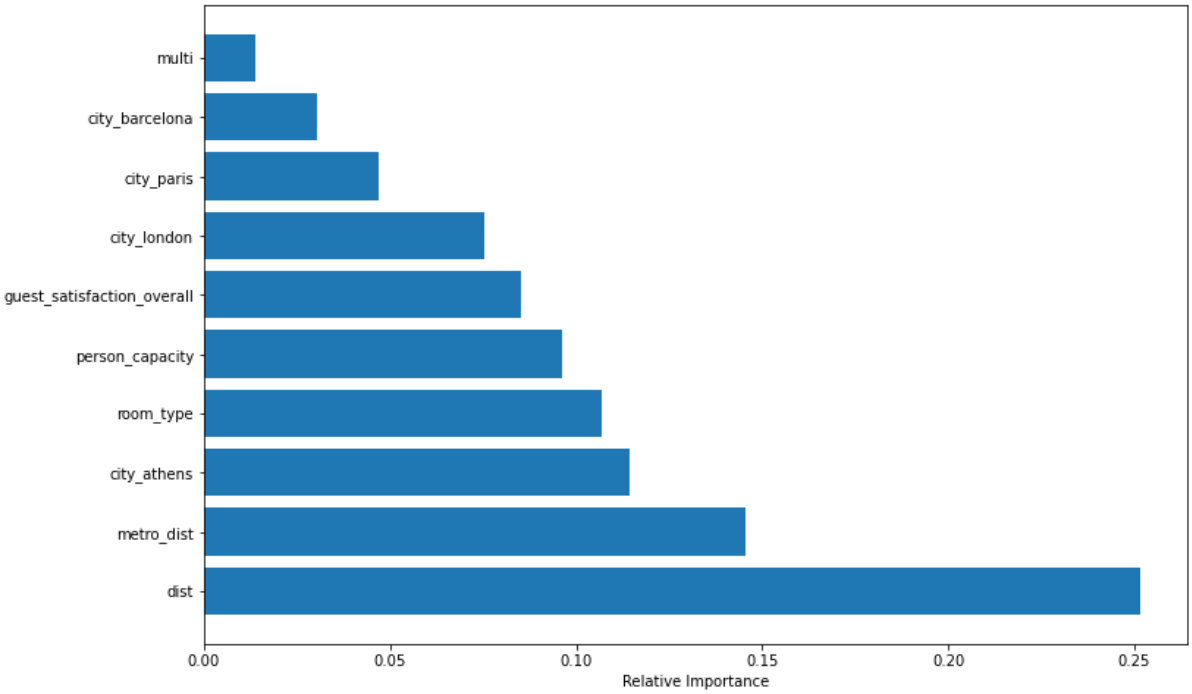


Figure I



References

Airbnb prices in European cities. Kaggle. (n.d.). Retrieved April 6, 2023, from https://www.kaggle.com/datasets/thedevastator/airbnb-prices-in-european-cities

Presentation Video -https://youtu.be/jED-b7bDPv8