GREAT LAKES
INSTITUTE OF MANAGEMENT, CHENNAI

greatlearning
Learning for Life

Post Graduate Program – Data Science and Engineering

JUNE 2020

PRICE PREDICTION OF AIRBNB LISTINGS – U.S.A

Project report submitted by:

Jency Priscilla R
Aishwarya M
Harrish Pakker A
Srinivas C S
Mahalakshmi R
Tharaknath G

Mentored by :

Mr. Ankush Bansal

ACKNOWLEDGEMENT

Our mentor and guide ANKUSH BANSAL was excellent in training us and teaching us all the techniques, his support and guidance were average at best. We would like to thank all our faculty members for their encouragement and guidance without which we could have not completed this project. A special thanks to Dr. Senthil TK who did more than prepare us academically, his confidence and dedication in teaching not only made learning fun and easy for us, but also had raised our bars in exploring further.

We would like to thank all our peers, for their constant support and with whom the learning grew exponentially.

ABSTRACT

Airbnb has become increasingly popular among travellers for accommodation across the world. Accordingly, there are large datasets being collected from the Airbnb listings with rich features. In this project, we aim to predict Airbnb listing price in the United States of America with various machine learning approaches. With one of our best approaches XGBoost algorithm, we have achieved r-squared values of greater than 0.7 in train and 0.69 in tests on the combined dataset.

TABLE OF CONTENTS

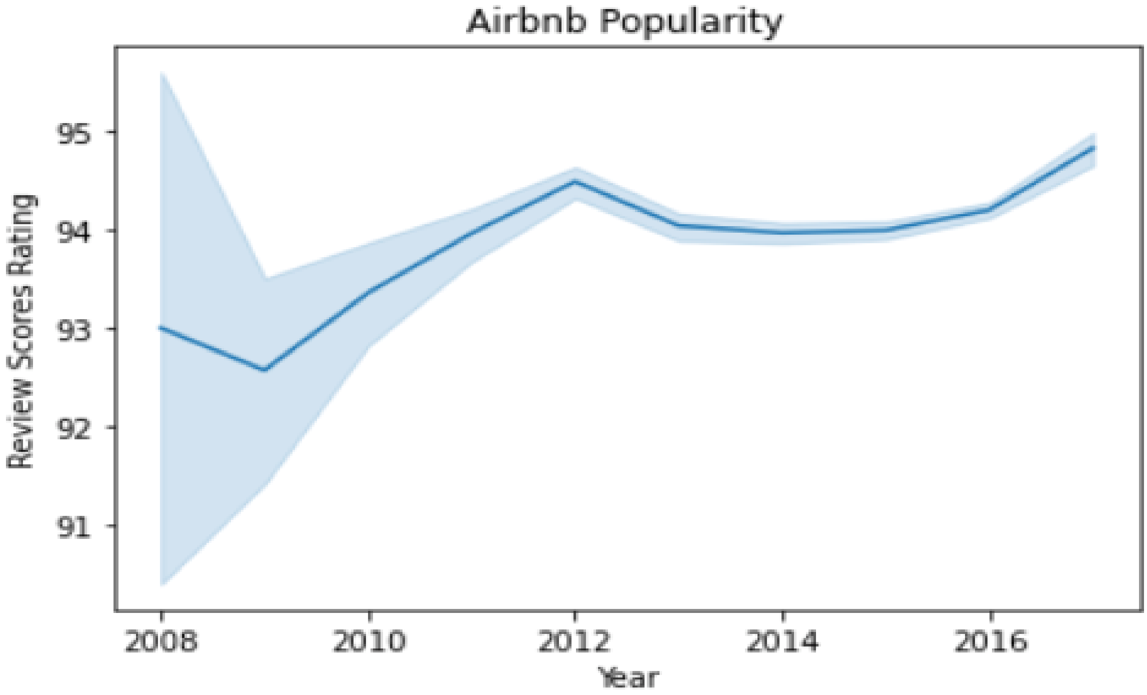
CHAPTER 1
INTRODUCTION

1.1 GENERAL

Airbnb provides a platform for hosts to accommodate guests with short-term lodging and tourism-related activities. Guests can search for lodging using filters such as lodging type, dates, location, and price, and can search for specific types of homes, such as bed and breakfasts, unique homes, and vacation homes. Before booking, users must provide personal and payment information. Some hosts also require a scan of government-issued identification before accepting a reservation. Guests can chat with hosts through a secure messaging system. Hosts provide prices and other details for their rental or event listings, such as the allowed number of guests, home type, rules, and amenities Hosts and guests have the ability to leave reviews about the experience.

Airbnb Plus designates hosts who provide a verified level of conditions, including a clean refrigerator, full cooking equipment, stocked toiletries, fast Wi-Fi, and strong water pressure. Airbnb Plus listings are marked with a badge to differentiate from standard listings. Airbnb Collections includes Airbnb for Families, Airbnb for Work, and home venues for weddings and other gatherings.

In addition to lodging, Airbnb includes listings for specific services on its platform, as Experiences; members may book both virtual and live activities with guides, including cooking classes, tours, and meetups.



1.2 NEED OF STUDY

Hotels have been an age-old phenomenon where a person stays and pays a particular price determined by the Hotelier. Airbnbs are like the online retailer for these ‘hotels’. However, the hotels don’t usually list themselves in Airbnbs. There are ‘hosts’ who are property owners looking to rent out their rental properties. Airbnb gives them an option to either rent out their entire property as a whole or part of it. However, Airbnb does not suggest any sort of pricing based on the property that a host has, and the host can name the price. Here, our host runs the risk of either overpricing it and losing popularity or under-pricing it and incurring losses. Our model suggests the appropriate price based on various input factors.

1.3 OBJECTIVES

Predicting the appropriate price for a property based on property specific factors.

Identify the level of significance that each variable contributes to the Price of a property. Interpreting the variables to find out the possible reasons that results adhering for a major population in data.

To help hosts identify unique patterns within the data which in turn might help them revise certain practices in Airbnbs.

The goal of the analysis is to allow hosts to make a decision on the price for their property without going under the average of the area and also not overpricing it.

1.4 SCOPE OF STUDY

The aim of the study is to make sure that hosts price their properties competitively, while also not scaring customers away by overpricing. Our model aids in predicting an appropriate price based on host property features, and its upto the host to decide whether or not, they would take the price up or add a markup to it

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Airbnb is a home-sharing platform that allows home-owners and renters (‘hosts’) to put their properties (‘listings’) online, so that guests can pay to stay in them. Hosts are expected to set their own prices for their listings. Although Airbnb and other sites provide some general guidance, there are currently no free and accurate services which help hosts price their properties using a wide range of data points.Paid third party pricing software is available, but generally you are required to put in your own expected average nightly price (‘base price’), and the algorithm will vary the daily price around that base price on each day depending on day of the week, seasonality, how far away the date is, and other factors.

Airbnb pricing is important to get right, particularly in big cities like London where there is lots of competition and even small differences in prices can make a big difference. It is also a difficult thing to do correctly — price too high and no one will book. Price too low and you’ll be missing out on a lot of potential income.

2.2 PROPOSED SYSTEM

This project aims to solve this problem, by using machine learning and deep learning to predict the base price for properties in London. I’ve explored the preparation and cleaning of Airbnb data and conducted some exploratory data analysis in previous posts. This post is all about the creation of models to predict Airbnb prices.

CHAPTER 3
DATASET DESCRIPTION

3.1 OVERVIEW

The dataset used for this project comes from Insideairbnb.com, an anti-Airbnb lobby group that scrapes Airbnb listings, reviews and calendar data from multiple cities around the world. The dataset was scraped on 9 April 2019 and contains information on all London Airbnb listings that were live on the site on that date (about 80,000).

The data is quite messy, and has some limitations. The major one is that it only includes the advertised price (sometimes called the ‘sticker’ price). The sticker price is the overall nightly price that is advertised to potential guests, rather than the actual average amount paid per night by previous guests. The advertised prices can be set to any amount by the host.

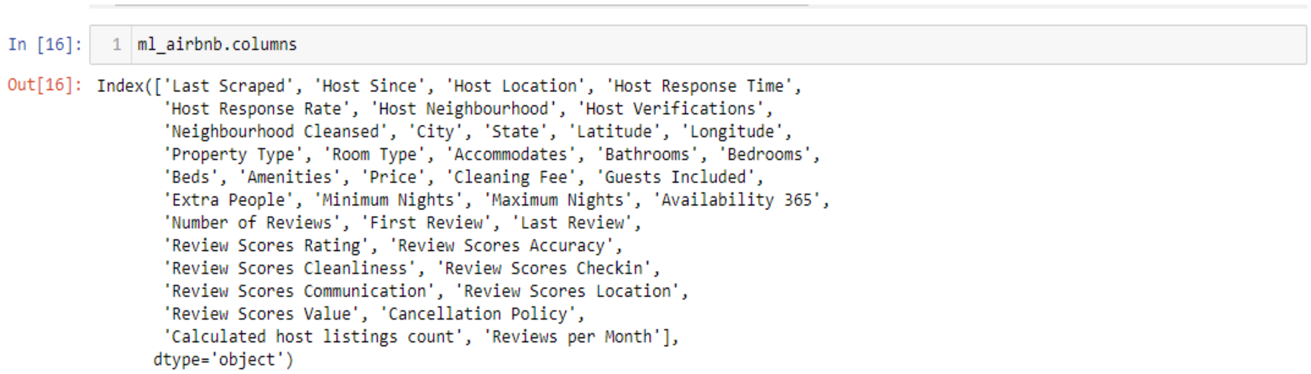
Nevertheless, this dataset can still be used as a proof of concept. A more accurate version could be built using data on the actual average nightly rates paid by guests, e.g. from sites like AirDNA that scrape and sell higher quality Airbnb data.

Here we only list a few of them that are both representative and important for the task, such as room size {accommodates, bathrooms, bedrooms, beds, ...}, extra fees {security deposit, cleaning fee, extra people, ...}, reviews scores {review scores rating, review scores accuracy, review scores cleanliness, ...}, location {neighbourhood, state, latitude, longitude, ...}, facilities {transit, amenities, property type, ...}, and booking related {availability, cancellation policy, host verification, ...}.

3.2 PRE – PROCESSING

3.2.1 COLUMNS CONSIDERED

Below we can see the screenshot where we have the list of our columns taken into consideration. The below list has been compiled after meticulously going through all 89 columns of our dataset and considering all the columns that have the highest chance of making our model useful.



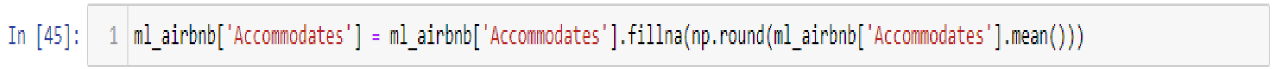
```
In [16]: ml_airbnb.columns
```

```
Out[16]: Index(['Last Scraped', 'Host Since', 'Host Location', 'Host Response Time',
                'Host Response Rate', 'Host Neighbourhood', 'Host Verifications',
                'Neighbourhood Cleansed', 'City', 'State', 'Latitude', 'Longitude',
                'Property Type', 'Room Type', 'Accommodates', 'Bathrooms', 'Bedrooms',
                'Beds', 'Amenities', 'Price', 'Cleaning Fee', 'Guests Included',
                'Extra People', 'Minimum Nights', 'Maximum Nights', 'Availability 365',
                'Number of Reviews', 'First Review', 'Last Review',
                'Review Scores Rating', 'Review Scores Accuracy',
                'Review Scores Cleanliness', 'Review Scores Checkin',
                'Review Scores Communication', 'Review Scores Location',
                'Review Scores Value', 'Cancellation Policy',
                'Calculated host listings count', 'Reviews per Month'],
               dtype='object')
```

With the above considered columns, we look at the null values and try to see how they can be treated without affecting the spread of our data.

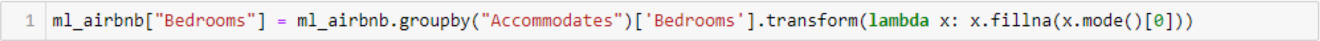
3.2.2. NULL VALUE TREATMENT

We have used a few different methods for imputation on the finalized columns dataset, ml_airbnb. We have taken values from other columns into consideration for the imputation, which can be seen below.

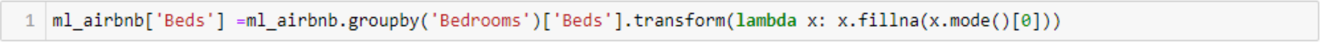


```
In [45]: ml_airbnb['Accommodates'] = ml_airbnb['Accommodates'].fillna(np.round(ml_airbnb['Accommodates'].mean()))
```

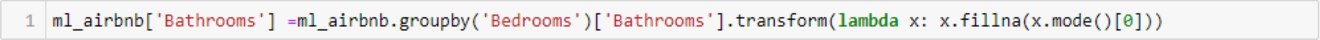
We see in the above graph that the Accommodates column has been mean imputed with the mean of the accommodates column. Below we can see that three columns have been imputed with their highly related counterparts and each uses the next to impute the columns.



```
ml_airbnb["Bedrooms"] = ml_airbnb.groupby("Accommodates")['Bedrooms'].transform(lambda x: x.fillna(x.mode()[0]))
```

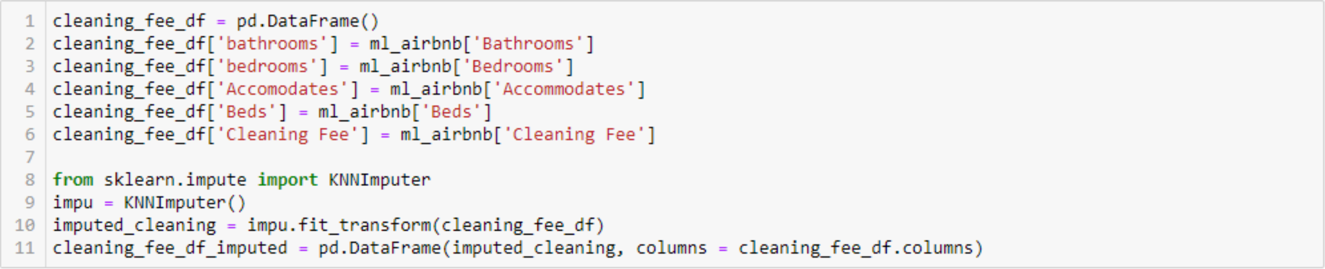


```
ml_airbnb['Beds'] =ml_airbnb.groupby('Bedrooms')['Beds'].transform(lambda x: x.fillna(x.mode()[0]))
```



```
ml_airbnb['Bathrooms'] =ml_airbnb.groupby('Bedrooms')['Bathrooms'].transform(lambda x: x.fillna(x.mode()[0]))
```

Below we can see that cleaning fee column is highly correlated with the target column. Also, we see that we have used the KNN imputer for the cleaning fee.

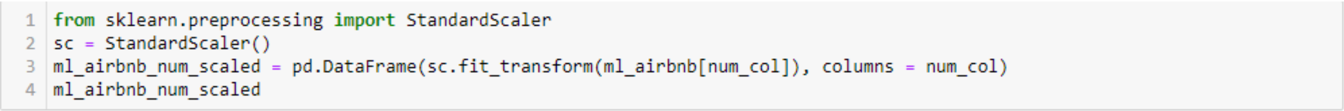


```
cleaning_fee_df = pd.DataFrame()
cleaning_fee_df['bathrooms'] = ml_airbnb['Bathrooms']
cleaning_fee_df['bedrooms'] = ml_airbnb['Bedrooms']
cleaning_fee_df['Accomodates'] = ml_airbnb['Accommodates']
cleaning_fee_df['Beds'] = ml_airbnb['Beds']
cleaning_fee_df['Cleaning Fee'] = ml_airbnb['Cleaning Fee']

from sklearn.impute import KNNImputer
impu = KNNImputer()
imputed_cleaning = impu.fit_transform(cleaning_fee_df)
cleaning_fee_df_imputed = pd.DataFrame(imputed_cleaning, columns = cleaning_fee_df.columns)
```

3.2.3 SCALING VALUES

Below we can see that cleaning fee column is highly correlated with the target column. Also, we see that we have used the KNN imputer for the cleaning fee.



```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
ml_airbnb_num_scaled = pd.DataFrame(sc.fit_transform(ml_airbnb[num_col]), columns = num_col)
ml_airbnb_num_scaled
```

3.2.4 FEATURE ENGINEERING

We have engineered a total of six features, three of which are explained in the upcoming slide.

We have taken amenities, which is a list of different amenities that the property provides. However, each property provides multiple amenities, which are given in a single cell.

We also have a similar column Host verifications, which contains the type of verification required to check in to the property.

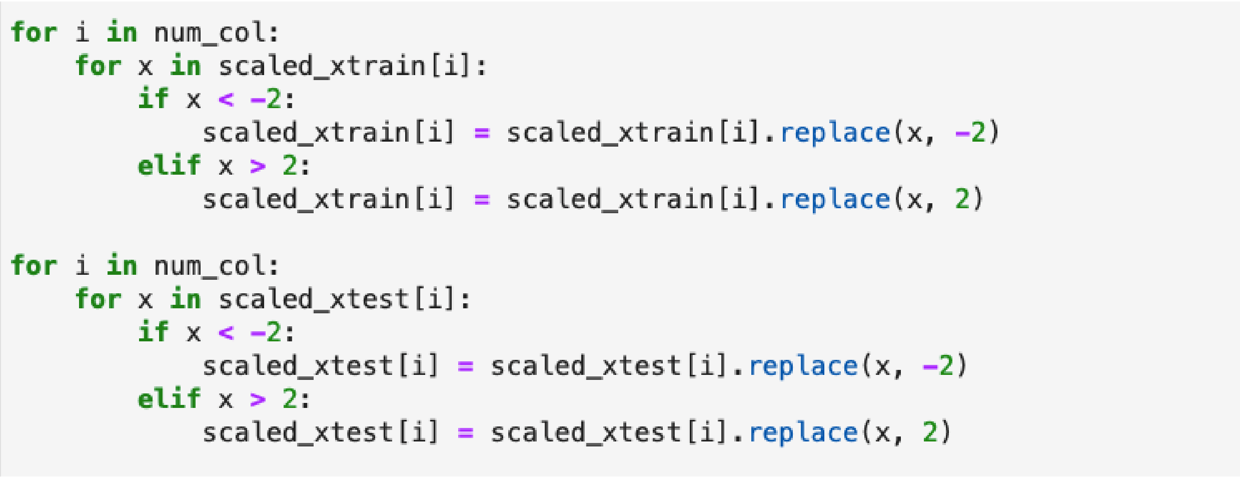
We have taken both and imported csv classifying them into Basic and Luxury amenities and Online and offline Host verifications.

Also, Since there were an excessive number of states to take into consideration, we sorted them into East, West and Central states.

We have also engineered property age as a new column which explains the age of the property, Inactivity years, which tells us how long a particular host has been inactive and the Host on location column, which tells us if the Host and the property are on the same address.

3.2.5 OUTLIER TREATMENT

Below are the codes that have been used for outlier treatment. We have initially scaled the data and then applied the outlier treatment. Here we have just capped the scaled value between -2 and 2 values so that our outliers are replaced with the max whisker value.



```
for i in num_col:
    for x in scaled_xtrain[i]:
        if x < -2:
            scaled_xtrain[i] = scaled_xtrain[i].replace(x, -2)
        elif x > 2:
            scaled_xtrain[i] = scaled_xtrain[i].replace(x, 2)

for i in num_col:
    for x in scaled_xtest[i]:
        if x < -2:
            scaled_xtest[i] = scaled_xtest[i].replace(x, -2)
        elif x > 2:
            scaled_xtest[i] = scaled_xtest[i].replace(x, 2)
```

3.2.6 FEATURE SELECTION

We have performed two different types of feature selection on our dataset. RFECV and VIF. We have taken inputs from both and it has been explained below.

3.2.6.1 VARIANCE INFLATION FACTOR (VIF)

The Variance Inflation Factor(VIF) detects multicollinearity in classification analysis. Multicollinearity is when there’s correlation between predictors (i.e. independent variables) in a model; its presence can adversely affect classification results. The VIF estimates how much the variance of a classification coefficient is inflated due to multicollinearity in the model.

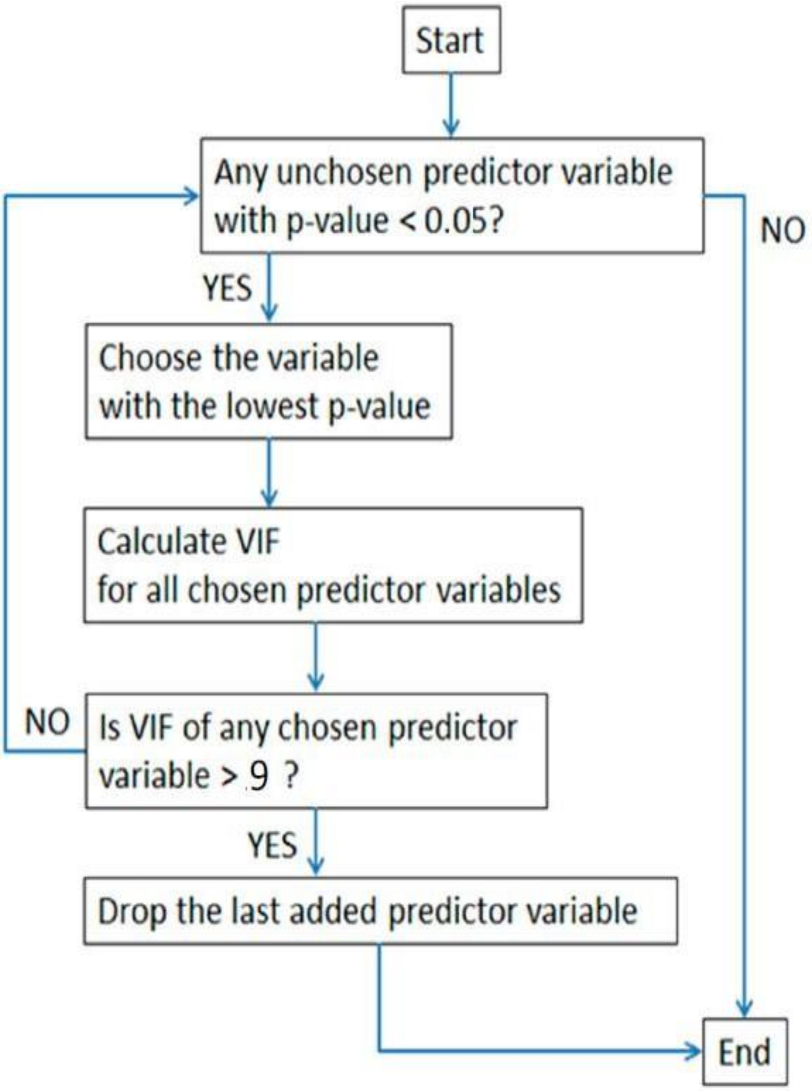
$$\text{VIF} = \frac{1}{1 - R_i^2}$$

- ➢ R - R squared Statistic
- ➢ i - the particular feature
- ➢ 1 = not correlated.
- ➢ Between 1 and 5 = moderately correlated.
- ➢ Greater than 5 = highly correlated

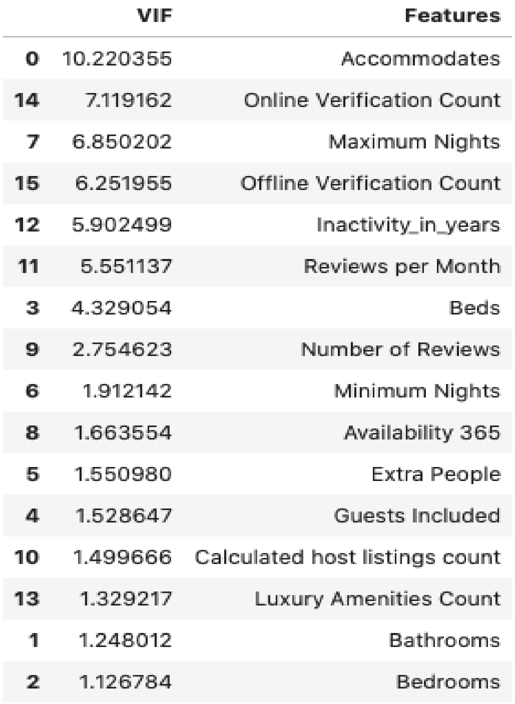
A VIF can be computed for each predictor in a predictive model. A value of 1 means that the predictor is not correlated with other variables. The higher the value, the greater the correlation of the variable with other variables.

Values of more than 4 or 5 are sometimes regarded as being moderate to high, with values of 10 or more being regarded as very high. These numbers are just rules of thumb; in some contexts a VIF of 2 could be a great problem. This value varies from case to case.

VIF WORKFLOW :


Start
Any unchosen predictor variable with p-value < 0.05?
NO
YES
Choose the variable with the lowest p-value
Calculate VIF for all chosen predictor variables
NO
Is VIF of any chosen predictor variable > 9 ?
YES
Drop the last added predictor variable
End

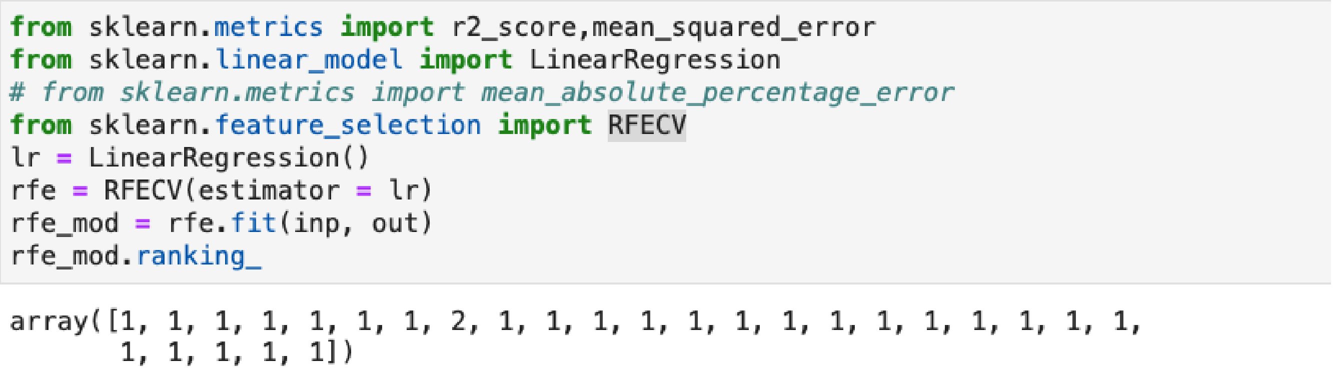
In this case the VIF cutoff is chosen as 11 and the features having VIF more than 11 are to be removed and the others are kept. The columns that are removed are shown below.



	VIF	Features
0	10.220355	Accommodates
14	7.119162	Online Verification Count
7	6.850202	Maximum Nights
15	6.251955	Offline Verification Count
12	5.902499	Inactivity_in_years
11	5.551137	Reviews per Month
3	4.329054	Beds
9	2.754623	Number of Reviews
6	1.912142	Minimum Nights
8	1.663554	Availability 365
5	1.550980	Extra People
4	1.528647	Guests Included
10	1.499666	Calculated host listings count
13	1.329217	Luxury Amenities Count
1	1.248012	Bathrooms
2	1.126784	Bedrooms

3.2.6.2. RFECV

We have considered RFECV to find out the best parameters for our model and we have managed to get particularly good results on the features that have been considered. Below is the screenshot of the same.



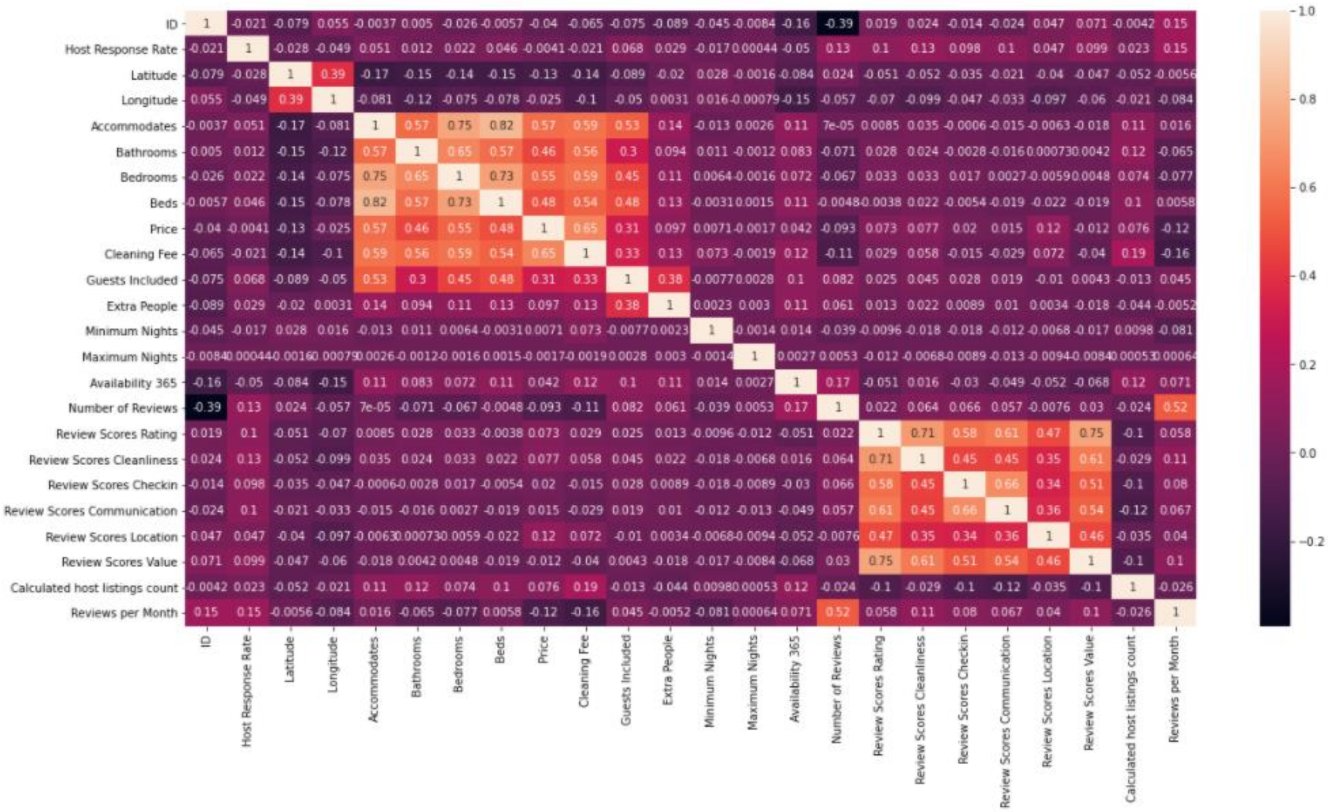
```
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.linear_model import LinearRegression
# from sklearn.metrics import mean_absolute_percentage_error
from sklearn.feature_selection import RFECV
lr = LinearRegression()
rfe = RFECV(estimator = lr)
rfe_mod = rfe.fit(inp, out)
rfe_mod.ranking_
```

```
array([1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1])
```

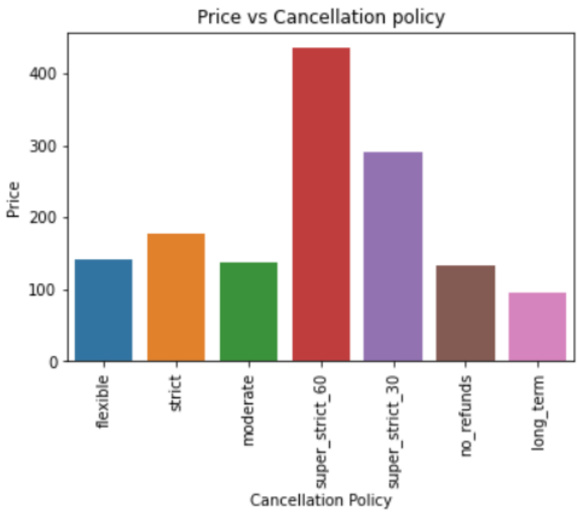
3.2.7 VISUALIZATIONS

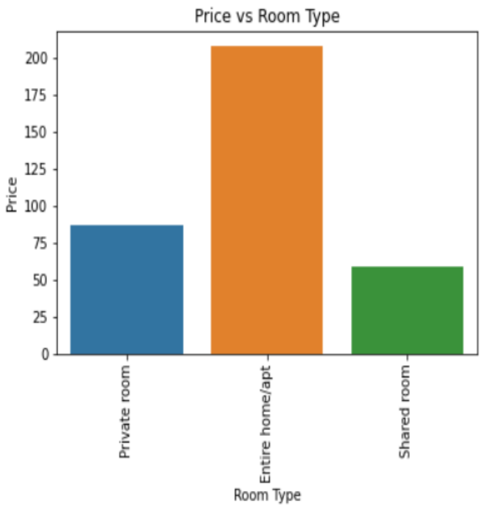
In this section the insights on all the features with bivariate and multivariate analysis are presented visually. This plot shows the distribution of no of patients in each Race in the data, there are five different categories.

This plot shows the distribution and correlation of various numerical features in our dataset.

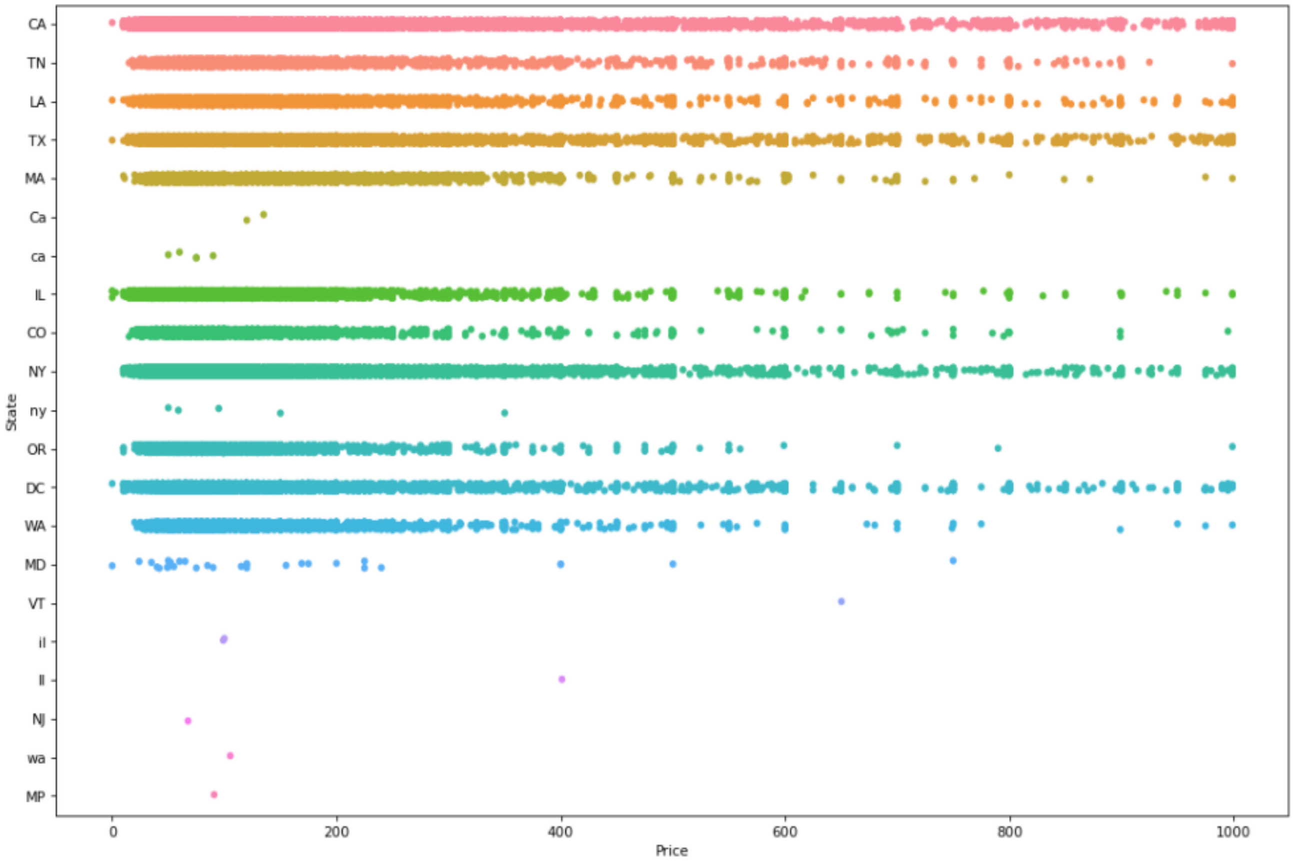


This plot below shows how the cancellation policy affects the price

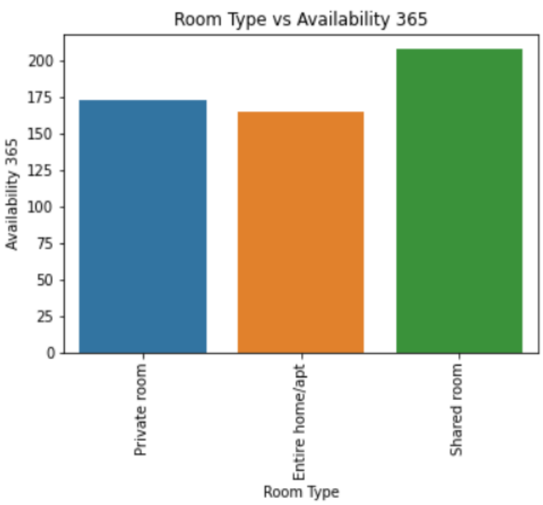


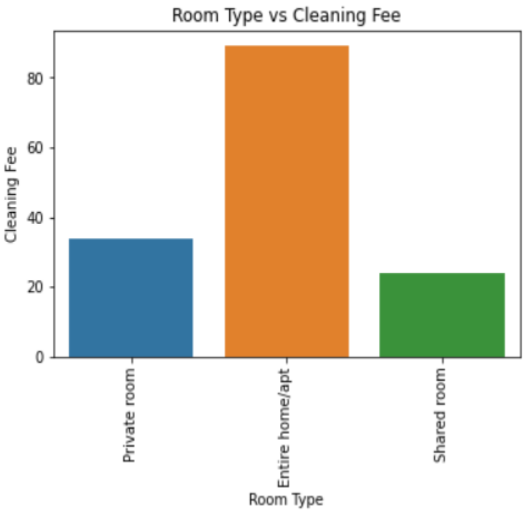


This plot shows the distribution of properties across states.

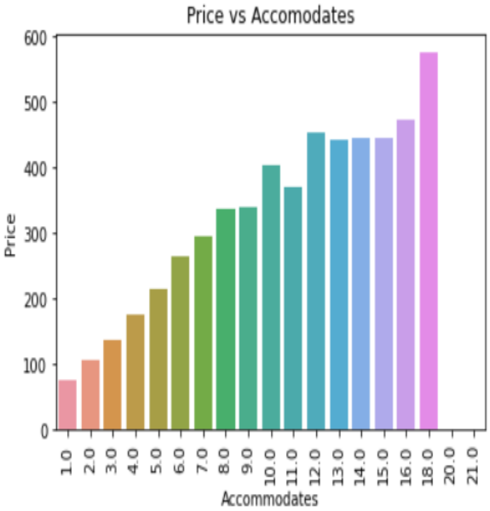


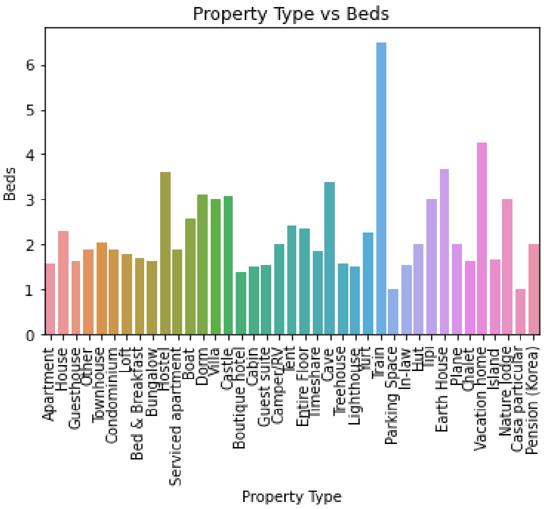
This plot shows the distribution of availability and cleaning fee across the different room types.





This plot shows the distribution of price in axis with the number of people a property accommodates.





CHAPTER 4
MODEL IMPLEMENTATION

4.1 INTRODUTION

Our initial price prediction problem is based on a regression problem.

4.2 REGRESSION

Regression is a **supervised learning** technique which helps in finding the correlation between variables and enables us to predict the continuous output variable based on the one or more predictor variables. It is mainly used for **prediction, forecasting, time series modeling, and determining the causal-effect relationship between variables**.

Various regression algorithms used are:

- OLS model
- Linear Regression
- Random Forest regressor
- Xgboost Regressor

4.2.1 OLS MODEL

Ordinary least squares (OLS) regression is a statistical method of analysis that estimates the relationship between one or more independent variables and a dependent variable; the method estimates the relationship by minimizing the sum of the squares in the difference between the observed and predicted values of the dependent variable configured as a straight line.

4.2.2 LINEAR REGRESSION

- Linear regression is a statistical regression method which is used for predictive analysis.
  - Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), hence called linear regression.
- The mathematical equation for Linear regression:

  Y= aX+b where , Y = dependent variables (target variables),

  X= Independent variables (predictor variables),

  a and b are the linear coefficients

4.2.2.1 VARIOUS ASSUMPTIONS FOR LINEAR REGRESSION

Below are some important assumptions of Linear Regression. These are some formal checks while building a Linear Regression model, which ensures to get the best possible result from the given dataset.

- **Linear relationship between the features and target:**

  Linear regression assumes the linear relationship between the dependent and independent variables.

- **Small or no multicollinearity between the features:**

  Multicollinearity means high-correlation between the independent variables. Due to multicollinearity, it may difficult to find the true relationship between the predictors and target variables. Or we can say, it is difficult to determine which predictor variable is affecting the target variable and which is not. So, the model assumes either little or no multicollinearity between the features or independent variables.

- **Homoscedasticity Assumption:**

  Homoscedasticity is a situation when the error term is the same for all the values of independent variables. With homoscedasticity, there should be no clear pattern distribution of data in the scatter plot.

- **Normal distribution of error terms:**

  Linear regression assumes that the error term should follow the normal distribution pattern. If error terms are not normally distributed, then confidence intervals will become either too wide or too narrow, which may cause difficulties in finding coefficients.

  It can be checked using the **q-q plot**. If the plot shows a straight line without any deviation, which means the error is normally distributed.

- **No autocorrelations:**

  The linear regression model assumes no autocorrelation in error terms. If there will be any correlation in the error term, then it will drastically reduce the accuracy of the model. Autocorrelation usually occurs if there is a dependency between residual errors.

the most important and common evaluation metric for classification problems. It gives us the number of correct predictions made as ratio of all predictions made.

4.2.3 XG BOOST

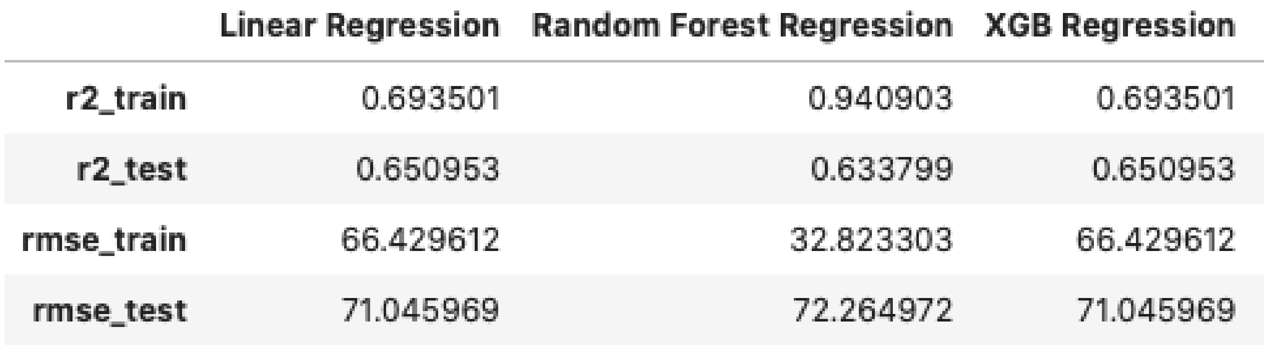
Ensemble learning involves training and combining individual models (known as base learners) to get a single prediction, and XGBoost is one of the ensemble learning methods. XGBoost expects to have the base learners which are uniformly bad at the remainder so that when all the predictions are combined, bad predictions cancels out and better one sums up to form final good predictions.

4.2.4 RANDOM FOREST REGRESSOR

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

4.3 COMPARISON OF MODEL PERFORMANCES

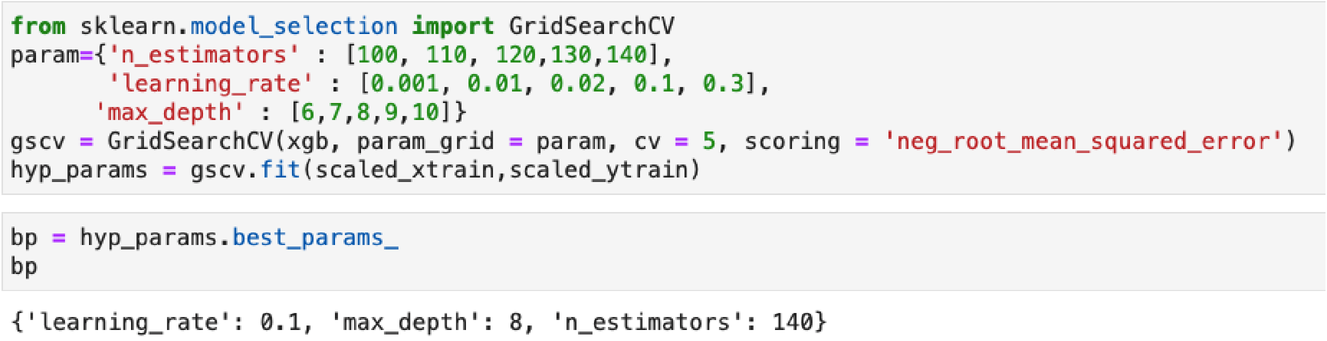
We have built models for Linear Regression, Random Forest Regression, XGB Regression. And since our tuned XGB regression model performs a little bit worse than the vanilla model, in r2 score, we are taking RMSE score into consideration.



	Linear Regression	Random Forest Regression	XGB Regression
r2_train	0.693501	0.940903	0.693501
r2_test	0.650953	0.633799	0.650953
rmse_train	66.429612	32.823303	66.429612
rmse_test	71.045969	72.264972	71.045969

4.4 HYPER PARAMETER TUNING

Our model performances show that the XGboost model has had the best performance, so we have tuned the XGboost model and the tuning has been shown below.



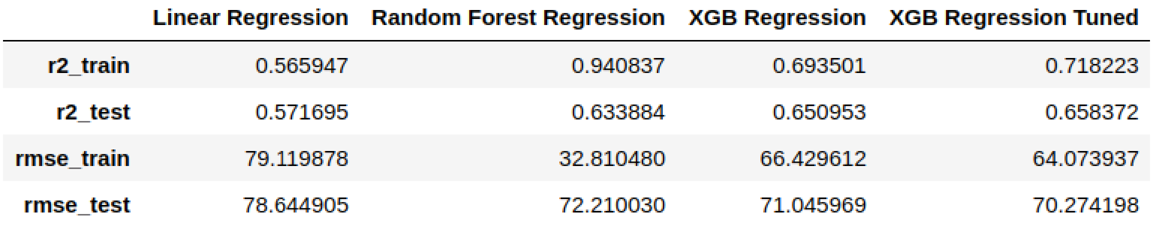
```
from sklearn.model_selection import GridSearchCV
param={'n_estimators' : [100, 110, 120,130,140],
       'learning_rate' : [0.001, 0.01, 0.02, 0.1, 0.3],
      'max_depth' : [6,7,8,9,10]}
gscv = GridSearchCV(xgb, param_grid = param, cv = 5, scoring = 'neg_root_mean_squared_error')
hyp_params = gscv.fit(scaled_xtrain,scaled_ytrain)
```

```
bp = hyp_params.best_params_
bp
```

```
{'learning_rate': 0.1, 'max_depth': 8, 'n_estimators': 140}
```

4.5 COMPARISON OF MODELS AFTER TUNING

Below we can see that our tuned XGB regression model performs a little bit worse than the vanilla model in r2 score, so we are taking RMSE score into consideration.



	Linear Regression	Random Forest Regression	XGB Regression	XGB Regression Tuned
r2_train	0.565947	0.940837	0.693501	0.718223
r2_test	0.571695	0.633884	0.650953	0.658372
rmse_train	79.119878	32.810480	66.429612	64.073937
rmse_test	78.644905	72.210030	71.045969	70.274198

CHAPTER 5

RESULTS AND FUTURE SCOPE

Overall, we have performed extensive feature extraction and engineering, and experimented with various machine learning approaches in predicting Airbnb listing price. We showed that XGBoost out-perform other approaches, and achieve r-squared value greater than 0.7. We can expect to have better results as the number of samples goes up.

In the future, we can build neural network models for the same dataset and have better results as a few have already undertaken the neural network have shown better results on this particular dataset.

REFERENCES

1. Prior works on rental price prediction based on Airbnb data are deficient in terms of evaluation metrics and performance. Tang and Sangani [2015] work on the task of price prediction for San Francisco Airbnb listings. They turn the regression problem into a binary classification problem by spiliting the price according to the median, which effectively reduces the difficulty of the task.

2. In a more recent work, Kalehbasti et al. [2019] works on price regression for Airbnb listings in New York. They use a range of methods including tree-based models, SVR, KMC, NN, etc and integrate sentiment analysis into their model. While they claim to achieve an highest R^2 of 0.7246, they evaluate their metrics (R^2 and MSE) on the logarithmic scale of price instead of the original scale.

3. Our project works on the original price regression problem, without transforming to a classification problem or evaluating on logarithmic scale. Besides traditional machine learning methods, we would integrate text data (from descriptions and reviews) into our model. To our best knowledge, there is no existing literature that uses text data as input variables for Airbnb price prediction.