Predicting Airbnb Fares Using Regression

Archana Subba
ITCS-5156 Spring 2022

February 28, 2022

Previous Research Paper/Journal
"Analysis of Airbnb Prices using Machine Learning Techniques"

Previous Research Paper/Journal Authors
Jasleen Dhillon, Nandana Priyanka Eluri, Damanpreet Kaur, Aafreen Chhipa, Ashwin Gadupudi, Rajeswari Cherupulli Eravi, Matin Pirouz,

Abstract

This report is presented as a survey of a previous work. Any assertions made within are subjective and do not represent those of the original author.

The sharing economy was hard to envision few years ago but due to its rapid growth, its importance has risen. Airbnb is a prime example and understanding it's pricing model will provide insights to this new business model.
Regression analysis is a supervised machine learning technique which is used for estimating the relationship between a dependent variable and one or more independent variable. However, it's not an easy task to predict the accuracy of continuous variable. In this paper, initially data cleaning and data pre-processing is performed. Then exploratory analysis is performed to get better understanding about the nature of the data. The analysis helps in understanding the important attribute which are required for prediction of the price of our Airbnb listings. For predicting the price, Linear, Random Forest and XGBoost model were implemented. After implementing all the models mentioned above, XGBoost was tuned with RandomizedSearchCV and Hyperopt hyperparameter strategies and finally the best model was chosen based on the RMSE value of the model. In this paper, the best model was for the XGBoost model tuned with Hyperopt.

1. Introduction

Until sharing business was introduced, hotels dominated the hospitality industry. Airbnb which was founded in 2008, introduced a new business model. This model of sharing economy had revolutionized the hospitality industry. It is a home-sharing platform that allows home-owners and renters to put their properties online, so that guests can pay to stay in them. The company does not own any property, but they serve as a third-party by receiving commissions on each booking made through them. Hosts are expected to set their own prices for their listings. Although Airbnb and other sites provide Paid third-party pricing software is available, but generally hosts are required to put in their expected average nightly price.

1.1 Problem

The hospitality industry is a competitive industry and Airbnb pricing is important to get right, particularly in large cities like New York 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 because the host also needs to make sure that the prices are not too high to attract enough visitors and not too low to miss out on a lot of potential income.

1.2 Motivation

Pricing becomes more complicated in the context of the sharing economy. However, it is also critical to understand the pricing structure of Airbnb because it drives consumers' decision making as well as stakeholders' profitability [7]. The motivation of the project is to further improve the price prediction model.

1.3 Approach

This project aims to solve the price prediction problem, by using Machine learning, regression analysis and hyperparameters to predict the base price for properties in New York. I've explored the preparation and cleaning of Airbnb data and conducted some data analysis and visualization below. All the non-essential features were dropped and categorical data were handled using One-Hot Encoding. I have used the Root Mean Square Error as a metric to determine the most accurate model.

2. Related Works

Based on the large number of public datasets provided by Airbnb, Airbnb price prediction with machine learning technique becomes a popular study. A class of studies, which is more pertinent to this work, where Rong Li, Ang Zhu, and Zehao Xie[3] developed an Airbnb price prediction model using Random Forest, XGBoost, and Neural Network. They eliminated several features, such as host_name and last_review to reduce noise and keep features like neighbourhood group and room type to build the model. After performing extensive feature engineering and extraction on the New York dataset, XGBoost and Random Forest perform better than other models. R-Squared and generalized cross validation (GCV) are used to examine the result. The R-Squared error for XGBoost is 0.618, and 0.612 for Random Forest.

	LM	GAM	DNN	RF	XGBoost
R^2	55.8%	59.1%	60.4%	61.2%	61.8%
GCV	0.195	0.182	0.179	0.175	0.17

Table 1: Summary statistics of each model

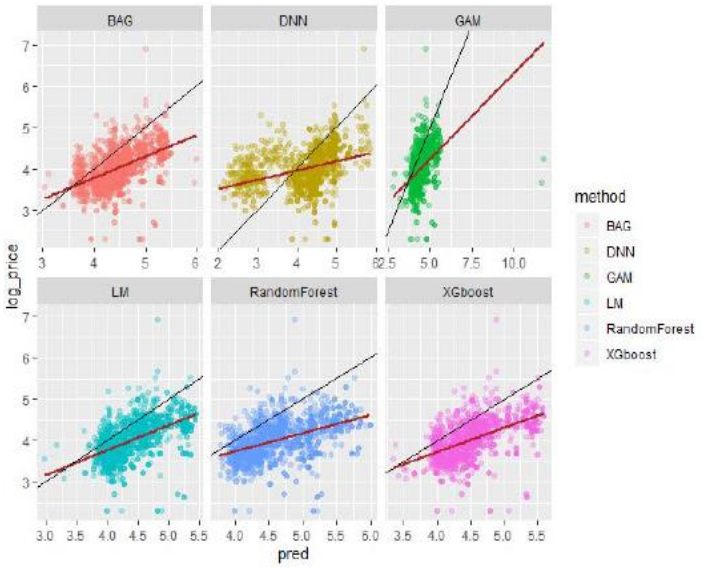


Figure 1: Model prediction indicating overestimation of actual price

Another notable implementation of Airbnb price prediction used features like country_code and number of bedrooms to build the model. Yuanhang Luo, Xuanyu, Zhou, and Yulian Zhou[6] developed an Airbnb price prediction model using Random Forest, XGBoost, and Neural Network. They eliminated several features, such as host_id and customer_name to reduce noise Textual data, such as house description and neighbourhood_review, is also considered as textual features. After performing extensive feature engineering and extraction on the New York and Paris dataset, XGBoost and Neural Network perform

better than other models. R-Squared and Median Squared Error (MSE) are used to examine the result. The R-Squared error for XGBoost is 0.722, and 0.769 for Neural Networks.

All the studies mentioned above used extra features such as textual data, and location, other than using only numerical data from the dataset. The availability of the room throughout the year, neighborhood of the listings are important factors that is helpful to build the price predictor. Chapter three and four introduces Machine Learning, Regression analysis that are used in this project and other price prediction studies.

Although such data serves different purposes, for regression analysis such extensive feature addition is not required. Presence of such features only increases the process of data cleaning, pre-processing process. In my project I have implemented Regression Analysis and it works with numerical data. Presence of categorical data also adds to the complexity of data analysis. However, similar to the approaches of both the studies, including data exploration and, training and evaluation of the Machine Learning models, I have approached my work.

2.1. Price Prediction Models

Machine Learning has been used in various fields such as financial market analysis, image recognition, medical analysis, etc. It builds models to learn from data without human intervention. This paper uses Linear Regression, Random Forest and Extreme Gradient Boost with and without hyper parameters. Each model has different structure, works differently and produces different predictions. The models have been introduced in chapter two and have been compared in chapter five.

2.1.1 Linear Regression

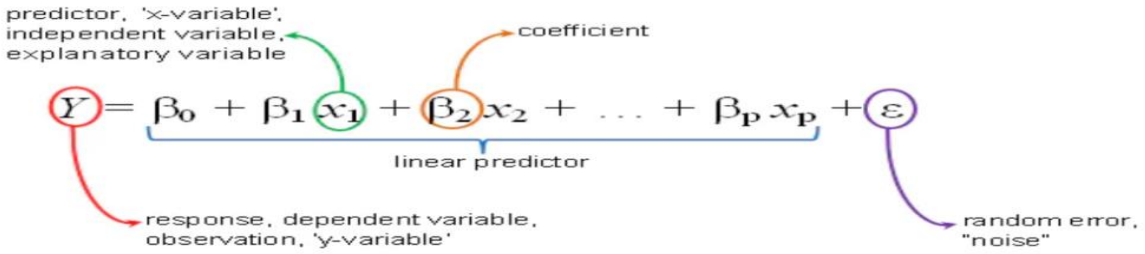
The most straightforward approach of this analysis is to use a linear regression model. It is a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).
The mathematical equation for simple linear regression is as follows:

$$y = B0 + B1*x, \text{ where}$$

y = dependent variable (target variable) or the output
y = Independent variables (predictor variables)
a and b are the linear coefficients

In higher dimensions when we have more than one input (X), the line is called a plane or a hyper-plane. The representation therefore is the form of the equation and the specific values used for the coefficients

The General equation for a Multiple linear regression with p - independent variables looks like this:



The goal of linear regression is to estimate the values for the model coefficients m_1, m_2, m_3, …, m_n and try to fit the training model with least root mean squared error and predict the values for the target variable

2.1.2 Random Forest

Random forest is a type of supervised learning algorithm that uses ensemble methods (bagging) to solve both regression and classification problems. The algorithm operates by constructing a multitude of decision trees at training time and outputting the mean/mode of prediction of the individual trees.
The fundamental concept behind random forest is the wisdom of crowds wherein a large number of uncorrelated models operating as a committee will outperform any of the individual constituent models.

It combines many different individual trees into an ensemble, and introduces random variation while building each decision tree. The term "random" in random forest has a dual meaning: a subset of the data is randomly chosen to establish each tree, and features are randomly selected in each split test as well. The resulting ensemble of trees is averaged to produce an overall prediction, which reduces overfitting while allowing for complex individual learners.

Random Forests have several good characteristics, including high accuracy among current algorithms, efficiency on large data sets, the ability to handle high dimensional input variables and missing data. Thus, Random Forest Regression is widely used and often yields very good results on a variety of problems.

The Random Forest Regression algorithm is as follows:

1. Pick n_{tree} bootstrap samples from the dataset;
2. Build an unpruned regression tree based on each of bootstrap samples. When picking the best split for a node, a random subset of features is selected to be searched over, rather than finding the best split across all possible features;
3. Predict new data using the mean of n_{tree} trees' predictions.

2.1.3 Extreme Gradient Boosting

Extreme Gradient Boost or XgBoost is a scalable end-to-end tree boosting system which introduce several new applications for approximate tree learning and weighted quantile sketch for efficient proposal calculation. It uses K additive trees to create the ensemble model,

$$\hat{y} = \hat{f}(x) = \sum_{i=0}^{K} \hat{f}_i(x), \hat{f}_i(x) \in F$$

$$\text{where } F = f(x) = w_{q(x)}(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$$

q represents the structure of the tree that maps an input to the corresponding leaf index at which it ends up being. *T* is the number of leaves in the tree. Each regression tree contains a continuous score on each of its leaf. w_i represents the score on i-th leaf.

2.1.4 Hyperparameter

Hyperparameters are points of choice or configuration that allow a machine learning model to be customized for a specific task or dataset. It is defined manually before the training of the model with the historical dataset. Its value cannot be evaluated from the datasets.
The following two hyperparameter techniques have been used in the project to improve the accuracy of the extreme gradient boost regressor model

A. RandomizedSearchCV

RandomizedSearchCV defines a search space as a bounded domain of hyperparameter values and randomly sample points in that domain. Its randomness yields high variance during computing**.** In this project max_depth has been used. The max_depth parameter specifies the maximum depth of each tree. The default value for max_depth is None, which means that each tree will expand until every leaf is pure. A pure leaf is one where all of the data on the leaf comes from the same class.

B. Hyperopt

Hyperopt is a Python library for serial and parallel optimization over awkward search spaces, which may include real-valued, discrete, and conditional dimensions. It uses a form of Bayesian optimization for parameter tuning.
Formulating an optimization problem in Hyperopt requires four parts:

1. **Objective Function**: takes in an input and returns a loss to minimize
2. **Domain space**: the range of input values to evaluate
3. **Optimization Algorithm**: the method used to construct the surrogate function and choose the next values to evaluate
4. **Results**: score, value pairs that the algorithm uses to build the model

2.2. Performance metric

Performance of machine learning models needs to be evaluated. To do that, the output generated by the models is compared with the actual values of the groundwater levels, this is done by using RMSE or Root Mean Square Error.

$$RMSE = \frac{\sqrt{\sum_{i=1}^{m} (e_i - o_i)}}{m}$$

i = variable i
m = number of non-missing data points
e_i = actual observations time series
o_i = estimated time series

The best model is the one with the lowest RMSE.

3. Dataset

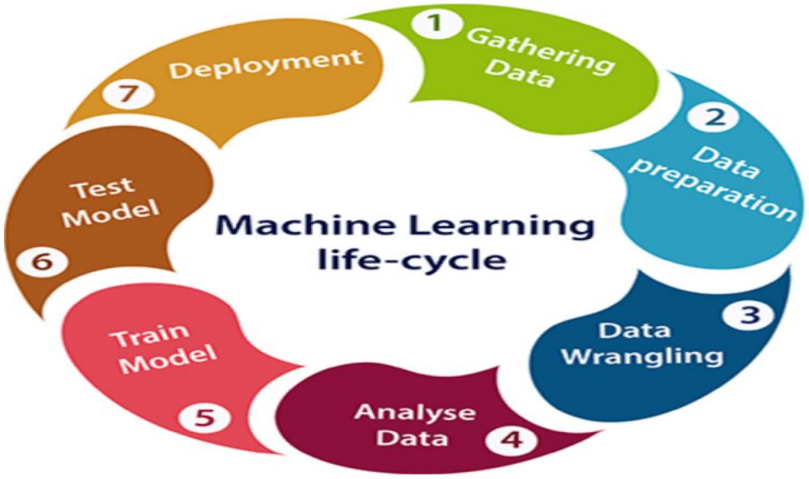
In this paper, dataset collected by Denis Gomonov and posted on Kaggle [4] is used. This dataset contains details of listings in New York and their attributes information from Airbnb website. It contains data for the year 2019.There are 16 variables and 48895 observations in the dataset. Following are the variables in the dataset:

1. id: Identification number of the Airbnb property
2. name: Name of the Airbnb property
3. host_id: Host identification number of the host
4. host_name: host name of the property
5. neighbourhood_group: neighborhood city of the listing in New York City
6. neighbourhood: neighborhood area of the listing
7. latitude: latitude coordinates
8. longitude: longitude coordinates
9. room_type: listing space type
10. price: price in dollars
11. minimum_nights: minimum number of nights
12. number_of_reviews: number of reviews of the listing
13. reviews_per_month: number of reviews per month of the listing
14. calculated_host_listings_count: number of listings per host
15. availability_365: number of days per year that listing is available

Some of the information such as id, names, host_id are eliminated in the training phase to reduce the noise. Neighbourhood_group, availability 365 are used to predict the price. There is total 48895 rows and 16 features.

4. Implementation and Experiment

The implementation of work coincides with the lifecycle of Machine Learning. I have included every step excluding the deployment phase



4.1. Environmental Setup

This project is executed and analyzed in a Windows 10 operating system with a core i5 processor and 8GB processor. The project has been implemented using Python which is a suitable programming language to build Machine Learning models. Its simplicity allows developers to focus on the model implementation instead of the language. Additionally, Python provides many Machine Learning frameworks to reduce the implementation time [5]. In this project, NumPy, Scikit-learn, Pandas, Matplotlib and Seaborn has been used for data processing and model implementation.

4.2.1 Data Wrangling

As the data in the original dataset was not in the proper format for preparing the model. There is requirement of transforming the data into proper format before applying the machine learning technique. Following few methods has been used to transform the data into proper format:

1. Replacing Not a Number (NaN) values for column such as reviews_per_month with zeros as default.
2. Removing columns such as name, host_name, last_review that are not related to my work.
3. Handling Outliers.

In the first step, I used Panda's isna() function to check for any NaN values. As shown in Figure 2. four fe atures have NaN. The NaN from reviews_per_month were imputed with zeros. Secondly, the columns tha t were not related to my work. The columns available in the dataset are - name', 'host_id', 'host_name', 'nei ghbourhood_group', 'neighbourhood', 'latitude', 'longitude', 'room_type', 'price', 'minimum_nights', 'numbe r_of_reviews', 'last_review', 'reviews_per_month', 'calculated_host_listings_count', 'availability_365'. Outliers in a dataset are those data points that deviate so much from the normal or average data points. These abnormal patterns in the dataset need to be examined carefully and to be removed from the dataset in order to increase the accuracy of the prediction model. They have been removed using standard values for each feature such as price less than 1 have been dropped.

```
name                                16
host_id                              0
host_name                           21
neighbourhood_group                  0
neighbourhood                        0
latitude                             0
longitude                            0
room_type                            0
price                                0
minimum_nights                       0
number_of_reviews                    0
last_review                      10052
reviews_per_month                10052
calculated_host_listings_count       0
availability_365                     0
dtype: int64
```

Figure 2: Representation of NAN

4.2.2 Data Visualization

Data visualization gives us a clear idea of what the information means by giving it visual context through maps or graphs.

1. A heatmap is a plot of rectangular data as a color-encoded matrix. As parameter it takes a 2D dataset. That dataset can be coerced into a ndarray. Using Figure 3, I was able to understand the relationship between the features available.

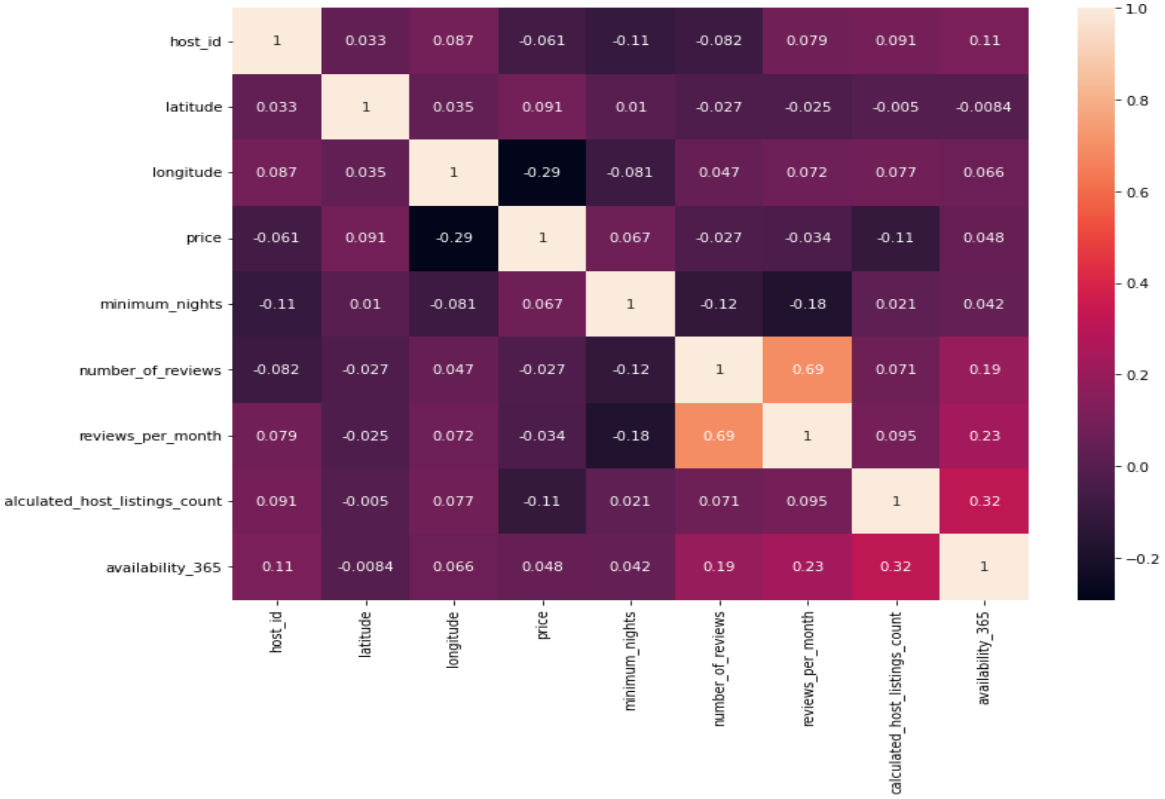


Figure 3: Heatmap representing the relationship between the features

2. Through Figure 4, we can infer the following
    - In **neighborhood group(location)**, Manhattan and Brooklyn stand in the top 2 positions respectively owing to the size and number of people who booked a room and it is followed by Queens, Bronx and Staten Island
    - The top 10 of **neighborhood(area)** is displayed second and Williamsburg stands top with most no. of bookings of nearly 3000

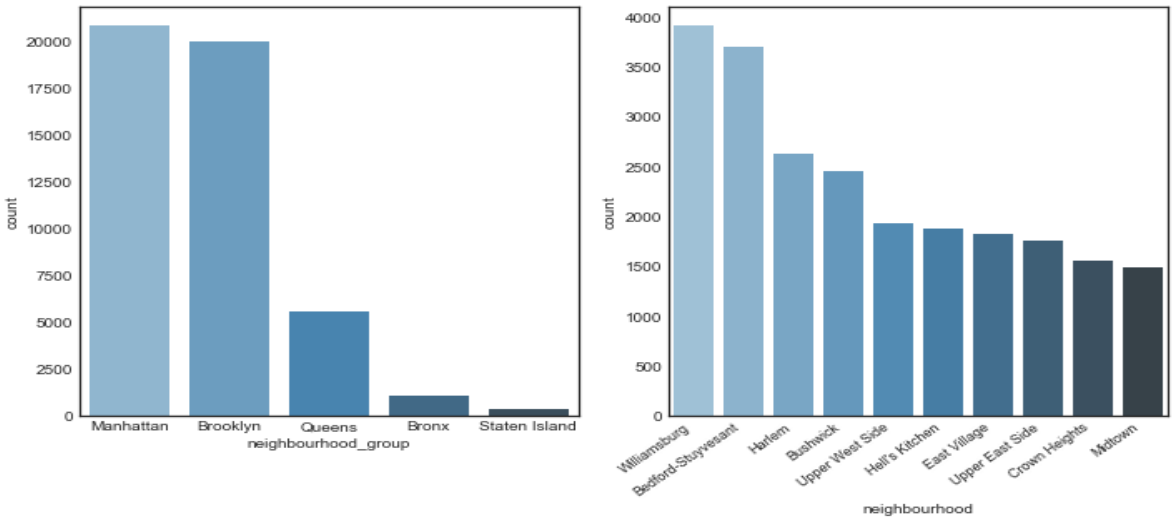


Figure 4: Study of Location features.

3. Type of rooms was one of the features selected. Using the plot in Figure 5, we can infer that the entire home or apartments are more preferred followed by private and shared room.

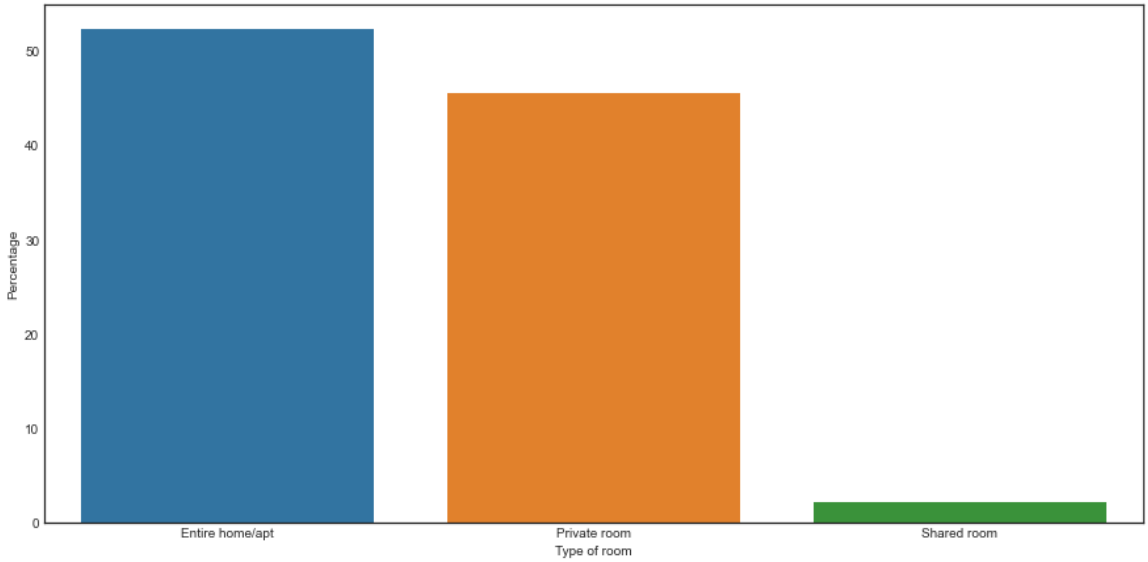


Figure 5: Type of rooms booked

4. One of the factors that is consider by guest for deciding the which to reserve is the availability. So, Figure 6 shows the percentage of number of days availability. From figure 6 it is observed that 75% listings are available for 0 days and only 5% are available for 365 days.

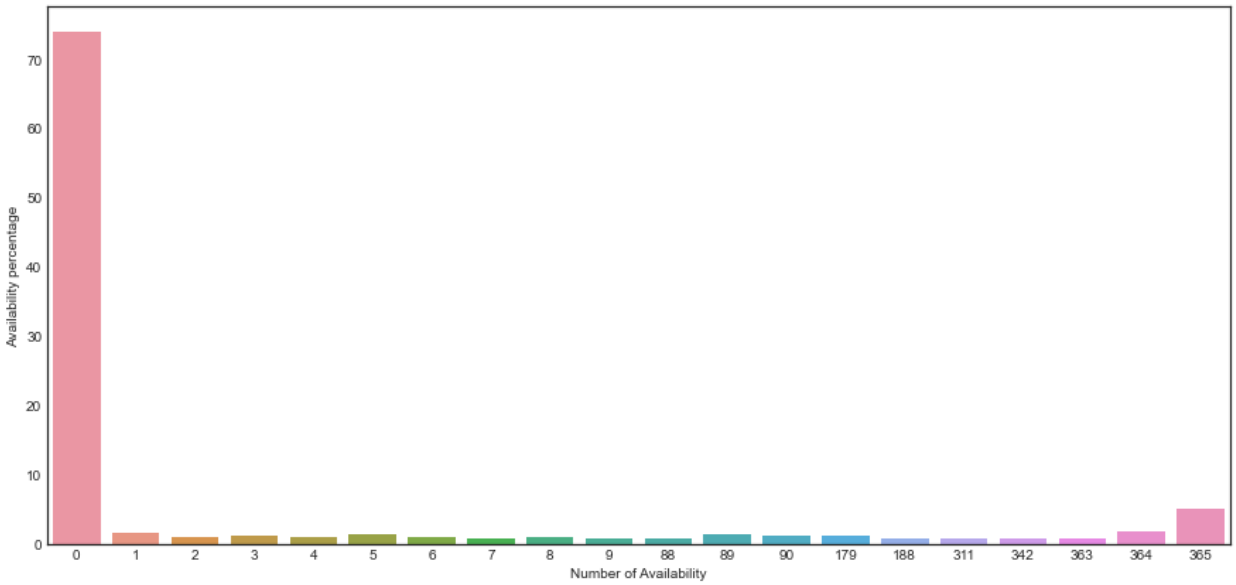


Figure 6: Study of number of availabilities

4.2.3 Data Preprocessing

A. Feature selection

Features such as neighbourhood group, room type, minimum nights, availability_365, number of reviews, reviews_per_month and calculated_host_listings_count were dropped after visualizing and checking their importance

B. Label Encoder

By using sklearn library label encoding can performed. A very efficient tool for encoding the levels of categorical features into numeric values is provided by sklearn library. Label encoder encodes label with a value between 0 and n-1 where n is the number of distinct labels. Same value is assigned which was assigned earlier if any label gets repeated. This was applied to the two categorical columns neighbourhood_group and room_type.

C. Data Split

In our project, we divide the training and testing data as 80% and 20% respectively using the train_test_split function from sklearn library.

4.3 My experiments and contributions

The goal of this project is to select the machine learning model that makes the most accurate prediction on Airbnb house prices and check for other methods to improve the accuracy. The proposed prediction method is making price prediction using regression analysis and using RMSE as its accuracy metric. I have used the New York dataset [4] to compare its performance with the two prediction methods from previous researches, in which models are not tuned using any hyperparameters. The untuned models yielded similar results to [1] and [2].
The hyperparameter were not part of the previous works that I referred to. I found its implementation in Kaggle [5] and slightly adapted the structure of Hyperopt in my work. My implementation is an extension of an existing project that uses standard regression models.

Three machine learning model from chapter 2 are trained and analyzed for each prediction method. The performance of models is analyzed using Root Mean Square Error. Finally, the predicted and true values of the model with the least value of RMSE value is compared.

4.3.1 Linear Regression

The Linear Regression model is chosen as the baseline model. The LinearRegression module from scikit-learn allows us to use multiple parameters, however, I have used a simple module with no parameters. It was easy to implement.

The implementation is provided below

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error,accuracy_score
from sklearn.metrics import classification_report

linear_model = LinearRegression()
linear_model.fit(X_train,t_train)

linear_pred_test = linear_model.predict(X_test)
linear_pred_train = linear_model.predict(X_train)
```

Figure: 7 demonstrates how the predicted values differ from the actual values for this model. A table has also been provided with the first 25 rows of actual and predicted data

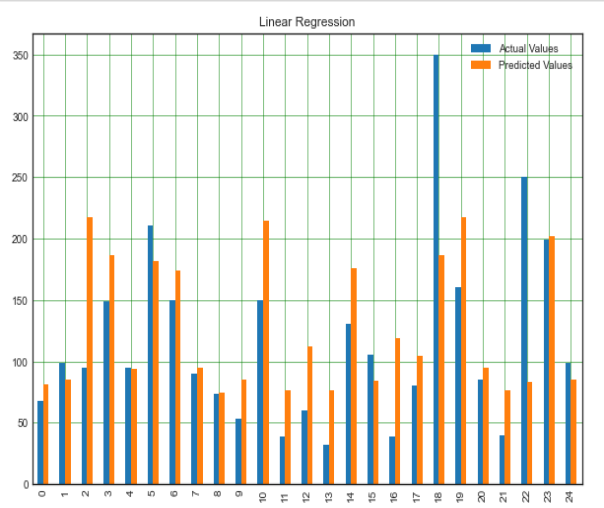


Figure 7 : Actual VS Predicted values

	Actual Values	Predicted Values
0	68	81.442983
1	99	84.737120
2	95	217.400393
3	149	185.972650
4	95	94.167845
5	211	181.766361
6	150	173.582128
7	90	94.839800
8	73	74.685646
9	53	85.459953
10	150	214.834165

Table 2 : Actual and Predicted Values in a Linear Regression

4.3.2. Random Forests

We use the RandomForestRegressor from the sklearn package to build the regression random forest model. Similar to the LinearRegression, no parameters were selected.

The implementation is provided below

```
#RandomForest
from sklearn.ensemble  import RandomForestRegressor


RFTree=RandomForestRegressor()
RFTree.fit(X_train,t_train)
rftree_pred=RFTree.predict(X_test)
```

Figure: 8 demonstrates how the predicted values differ from the actual values for this model.

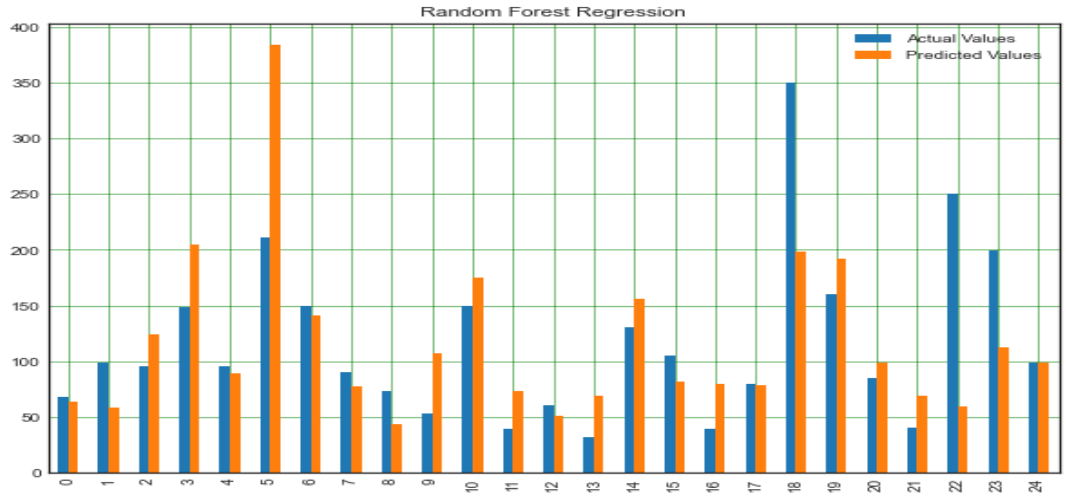


Figure 8 : Actual VS Predicted values using RandomForest Regressor model

4.3.3. XGB Regressor

XGBoost Regressor is also ran without its parameter and the implementation also did not take much time. The implementation is provided below

```
#XGBRegressor
import xgboost as xgb

xgbr = xgb.XGBRegressor()
xgbr.fit(X_train, t_train)

xgbr = xgbr.predict(X_test)
```

Figure: 9 demonstrates how the predicted values differ from the actual values for this model

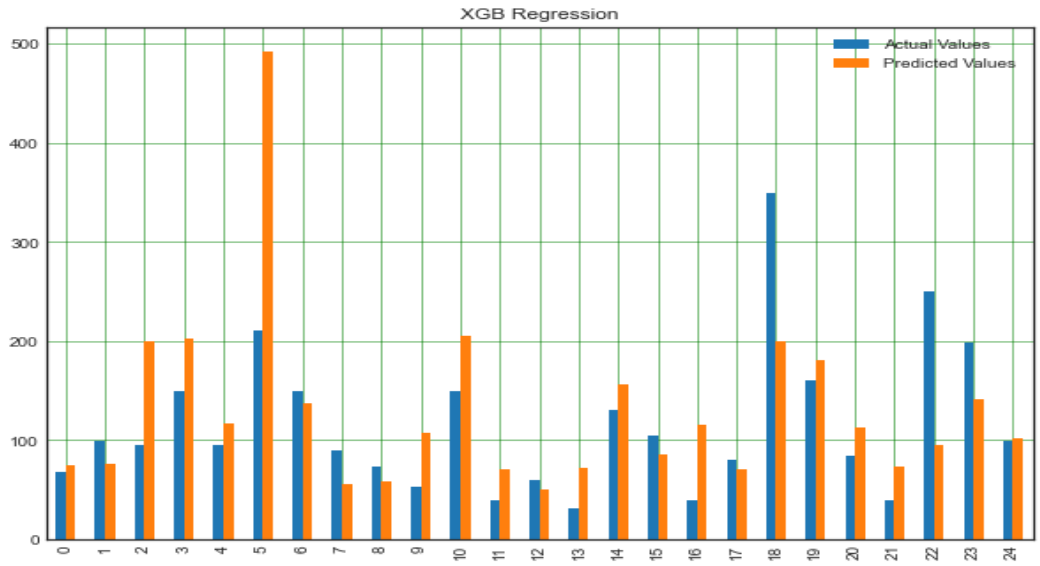


Figure 9 : Actual VS Predicted values using Extreme Gradient Boost Regressor model

4.3.4 Hyperparameters

A. RandomizedSearchCV
RandomizedSearchCV is ran using dictionary parameter with max_depth values 4,16,1. All the parameters' values were chosen randomly. The implementation took comparatively more time than the prior standard implementations.

After running the RandomizedSearchCV, the best parameter, max_depth = 4, was selected and then the XGB Regressor was ran again with the tuned parameter. This yielded less RMSE than simple standard implementation of XGB Regressor.
The implementation is provided below

```
# Running the model tuned model

booster = xgb.XGBRegressor(max_depth = 4)

# train
booster.fit(X_train, t_train)

# predict
booster_pred_RSCV = booster.predict(X_test)

print('RMSE: %f' % np.sqrt(mean_squared_error(t_test,booster_pred_RSCV)))


RSCV_score = np.sqrt(mean_squared_error(t_test,booster_pred_RSCV))
```

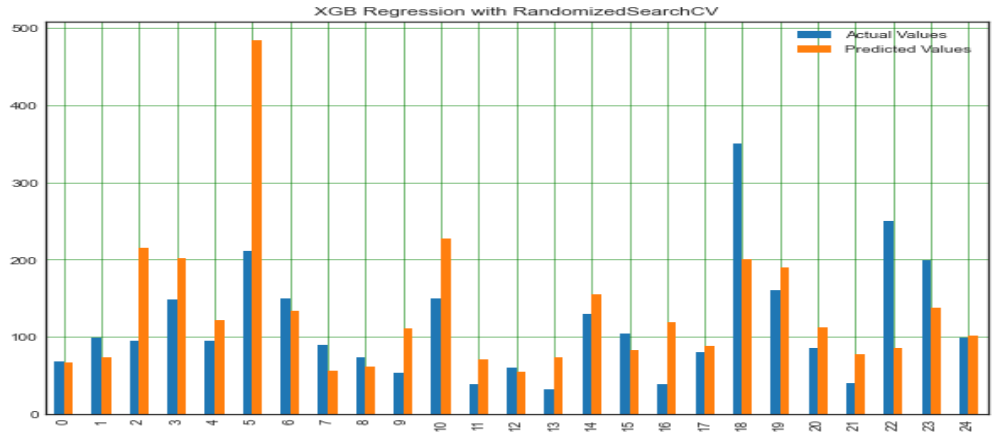


Figure 10 : Actual VS Predicted values using Extreme Gradient Boost Regressor model with tuned parameters using RandomizedSerachCV

B. Hyperopt

Hyperopt is also ran using the same max_depth values as RandomizedSearchCV, however, the implementation took much longer than RandomizedSearchCV.

Similar to the previous hyperparameter strategy, the best parameters were run with the XGB Regressor again(tuned) and a lower RMSE compared to before was observed.

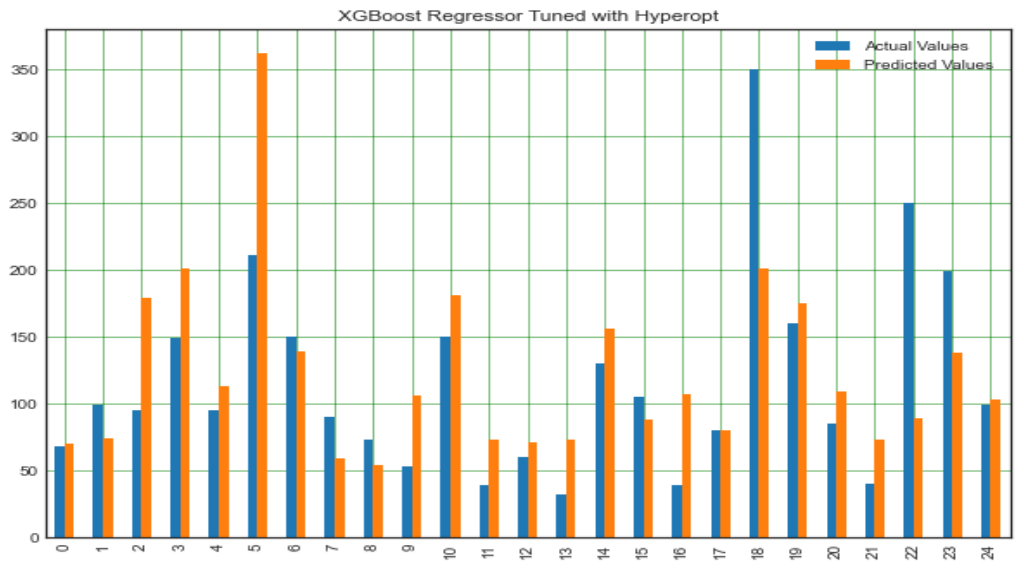


Figure 11 : Actual VS Predicted values using Extreme Gradient Boost Regressor model with tuned parameters using Hyperopt

The implementation of Hyperopt is provided in Figure 12 and 13

```
#Define the hyperopt objective.
def objective(space):
    print(space)
    clf = xgb.XGBRegressor(max_depth = int(space['max_depth']),)

    eval_set  = [( X_train, t_train), ( X_test, t_test)]

    clf.fit(X_train, t_train,
            eval_set=eval_set, eval_metric="rmse",
            early_stopping_rounds=10,verbose=False)

    pred = clf.predict(X_test)
    mse_scr = mean_squared_error(t_test, pred)
    print("SCORE:", np.sqrt(mse_scr))
    #change the metric if you like
    return {'loss':mse_scr, 'status': STATUS_OK , 'model': clf}


space ={'max_depth': hp.quniform("x_max_depth", 4, 16, 1)
    }


trials = Trials()
best = fmin(fn=objective,
            space=space,
            algo=tpe.suggest,
            max_evals=100,
            trials=trials)

print(best)
```

Figure 12: Defining the Hyperopt objective

```
best_model = trials.results[np.argmin([r['loss'] for r in trials.results])]['model']

print(best_model)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

Figure 13: Best parameter from Hyperopt

4.4 Results

To evaluate the performance of each model, the RMSE between all the implemented models is compared. Table 3 shows the results of the models that were implemented with and without any hyperparameter. Simple Linear Regression as usual performed the worst among all the implemented models and had a RMSE of 115.4. Similar to [1], Random Forest Regressor has comparatively worked better than the simple Regression model in terms of RMSE.

My work also included the introduction and addition of the hyperparameters. The RMSE of the tuned XGB Regressor models are lesser than the standard model. The least RMSE is 104.57 for the XGB Regressor model that was tuned using Hyperopt. Similar discrepancies can be observed in the Table 2 and Table 4 values of Linear Regression and XGBoost Regressor model using Hyperopt.

	RMSE
Linear Regression	115.417224
Random Forest Regressor	113.126609
XGBoost Regressor	105.964552
RandomizedSearchCV(XGB Regressor)	105.095723
Hyperopt(XGB Regressor)	104.573182

Table 3: RMSE across all implemented models.

	Actual Values	Predicted Values
0	68	70.308197
1	99	74.204086
2	95	179.519928
3	149	201.097519
4	95	113.080307
5	211	362.575989
6	150	138.659042
7	90	59.508053
8	73	54.381794
9	53	106.355232
10	150	180.647491

Table 4 : Actual and Predicted Values in a XGBoost Regressor using Hyperopt

5. Conclusion and Afterthoughts

In this paper, Regression Analysis with hyperparameter were introduced. This paper aims to build the best-performing Airbnb market prediction model on the basis of a number of features, including property specifications, input from hosts and consumers on the listings. To compare the performance of three methods and the hyperparameter, the root mean square error was analyzed.

I believe this project has provided me opportunity to learn more. I was able to able to implement almost all the knowledge that I was able to acquire through the class labs in this project. For me, implementation of the models was easier than data analysis. Through this project, I was able to play with raw data more.

One of the most challenging parts of the project was implementation of the hyperparameters. Understanding of Hyperopt took a lot of time but gradually I got used to it and was able to understand it working much better after running many trials and errors with it.
In the future, I am looking forward to working with price prediction using Natural Language Processing

References –

1. Jasleen Dhillon, Nandana Priyanka Eluri, Damanpreet Kaur, Aafreen Chhipa, Ashwin Gadupudi, Rajeswari Cherupulli Eravi, Matin Pirouz, “Analysis of Airbnb Prices using Machine Learning Techniques”
2. Brahmaih, Kala. (2020). Predicting Airbnb Listing Price Across NewYork. 10.13140/RG.2.2.28089.70246.
3. Ang Zhu, Rong Li, Zehao Xie, “Machine Learning Prediction of New York Airbnb Prices
4. https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data
5. https://www.kaggle.com/dreeux/hyperparameter-tuning-using-hyperopt
6. Luo, Y. et al. “Predicting Airbnb Listing Price Across Different Cities.” (2019).
7. C. Gibbs, D. Guttentag, U. Gretzel, J. Morton, and A. Goodwill, ”Pricing in the sharing economy: a hedonic pricing model applied to Airbnb listings,” J. Travel Tour. Mark., pp. 1–11, Apr. 2010.