Airbnb Price Prediction

Pramiti Barua, Aayusha Shrestha, Divyansh Yadav, and Isaac Kobbi Anni

Bowling Green State University

December 1, 2022

Abstract

The purpose of the article is to utilize several machine learning strategies to predict the price for the Airbnb dataset.To comprehend the dependence of numerous attributes on the target variable, visualizations,and exploratory analysis are performed on the dataset. Different models such as Linear regression, Ridge regression, K-nearest regressor, Decision Tree regressors, Random Forest, Gradient Boosting, XGboost, SVR, and Neural Networks were used. The results indicate that Random Forest works well for this dataset when the train-test split approach is adopted while in case of the k-fold cross-validation approach, gradient boosting performed better.

Keywords: ML, MAE, MSE, SVR, KN Regressor, xGboost.

1 Introduction

Over 60 million guests have stayed at Airbnb properties since it launched, and the company is still expanding swiftly. For both hosts with vacant space and visitors looking for convenient and inexpensive housing choices, Airbnb offers a simple source of revenue. For both hosts and guests, it's crucial to try to observe and comprehend the underlying pricing dynamics of the Airbnb market. Homeowners may find it challenging to accurately market their property as users continue to increase on both the supply and demand sides. Due to its recognition of this, Airbnb has done extensive study on pricing recommendations from the supply side [1].

In order to better understand Airbnb prices around we examined several listings in the NYC area. Better price recommendation estimations can help Airbnb hosts determine an equilibrium pricing that balances affordability and profit. The goal of this project is to create a model that forecasts the best price for a property while taking listing characteristics and seasonality into account. The end goal of this project is to predict the price of Airbnb using several machine-learning methods.

2 Background

Finding a reasonable pricing for an Airbnb rental is a challenge that every user has encountered given the enormous number of listings. The issue is crucial to how both Airbnb customers and property owners conduct their business. Given that 53% of visitors utilize Airbnb to cut costs, a significant portion of rentals are exclusively made due to their cost. There is certainly opportunity for improvement as only 11% of the approximately 500,000 US listings are reserved on a normal night. Our initiative aims to address the question "Which model best fits to find the cheapest airbnb price". The solution to this query will make it easier for users and property owners to determine the market value for a listing, facilitating trade in a multibillion-dollar industry[2].

3 Methodology

3.1 Pre-processing and Data Cleaning

A few values for name and host name were missing and the last review and reviews per month columns both contain 10,052 missing values. With special characters like "#," "$," and "r," we filled in the blanks for name and hostname, respectively. Missing values are replaced with 0 for reviews per month. As our goal is to estimate the airbnb price of the NYC dataset, the column labeled "price" would be the desired variable.

3.2 Exploratory Data Analysis

We have used New York City's Airbnb open dataset which was available on Kaggle [7]. There

are a total of 40k data sets with each entry having 16 distinctive features like room, type, neighborhood groups etc. Before starting modeling, we investigated the data and discovered that it had 16 columns and roughly 49000 observations. Both categorical and numerical values are present in these columns. An initial review of the dataset was conducted before data cleaning.

3.2.1 Price distribution

The price variable was extremely skewed, as can be seen from the distribution plot below in figure 1. This could pose issues for future models. So, log transformation was applied. Figure 2 depicts that, after applying a log transformation to the target variable, the distribution appeared to be normal.

For the prediction task, we need to know which attributes are helpful to estimate prices. By examining the features, we discovered that the feature 'Id' had 48895 distinct values or nearly the same number of samples as there are in the dataset. Therefore, the "Id" field is not useful when performing a regression analysis and we removed the column.

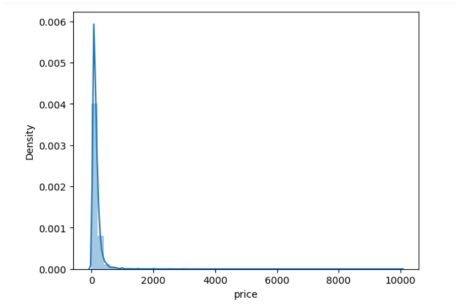

Figure 1: Highly Skewed Price

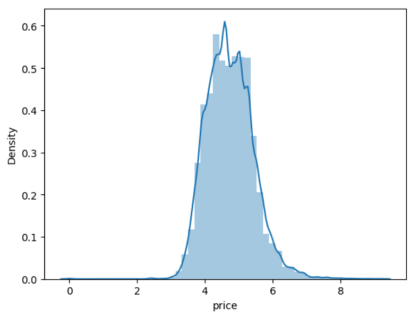

Figure 2: Normally distributed Price column

3.2.2 Visualization

The first visualization was carried using a heatmap between all of the attributes to examine the relationship between pricing and other columns. Apart from the number of reviews and reviews per month, we can observe that there is no clear association between the columns. From the negative correlation between price and longitude, it can be explained that individuals do not like locations that are far from the city. The

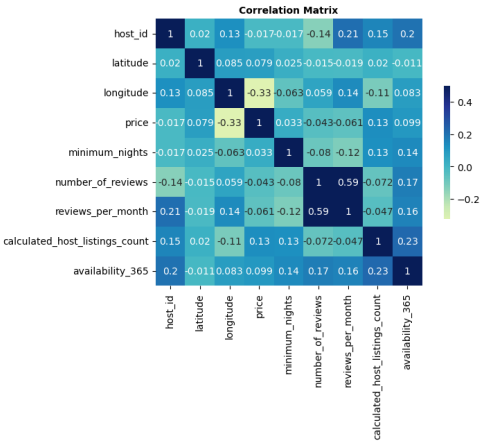

Figure 3: Correlation matrix for the response variables and the target

neighborhood is grouped in the second representation using a scatter plot with longitude and latitude. Figure 4 makes it evident that the latitude and longitude of the neighborhood group are significant columns to take into account since they correspond to the number of reservations in the neighborhood group. Additionally, it should be noted that Staten Island and the Bronx receive significantly fewer bookings than Brooklyn and Manhattan. It makes sense because Brooklyn and Manhattan are popular tourist attractions with sizable expenses.

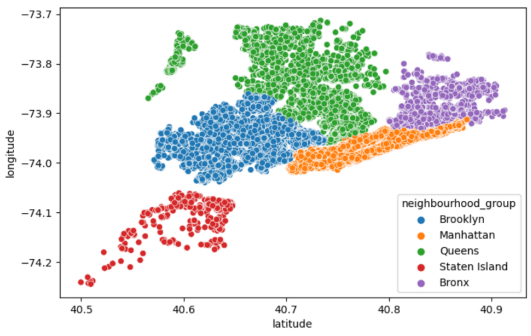

Figure 4: Booking distribution for neighborhood group

In figure 5, we attempted to investigate the pricing comparison by neighborhood. The pricing variations between the neighborhoods are clearly represented in the accompanying bar chart. The Bronx seemed to have the lowest pricing among the neighborhoods, whereas Manhattan has the

highest price. The data analysis reveals that various variables have a substantial impact on the price column.

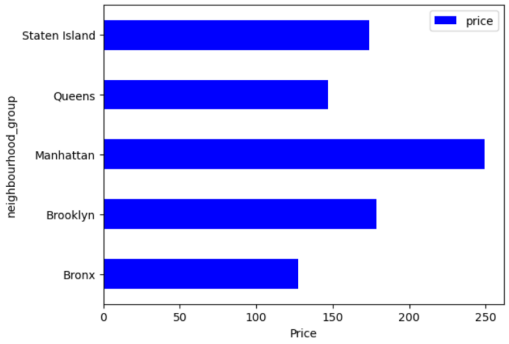

Figure 5: Neighborhood vs Price

3.2.3 Feature Selection

We employed the feature importance and chi2 techniques for feature selection. Results in fig 6 depict that the neighborhood group has low importance. Chi2 results also indicated low scores for the neighborhood group. But we think neighborhood group is important for price prediction. So, we wanted to go without a feature selection approach.

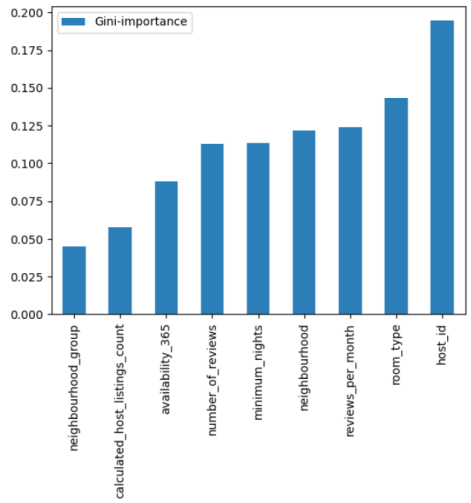

Figure 6: Feature Importance

4 Methods Used

After the dataset has been cleaned and preprocessed, modeling has been done using a variety of techniques, including Linear Regression, Ridge Regression, Support Vector Regression, Neural Networks, Gradient Boosting Trees, Extra Gradient Boosting, Decision tress and random forest. The dataset was split into training and testing using the train-test split method and the k-fold cross validation technique. When employing the train-test split method, the training set contained 80% of the data and the test set had 20%. We modelled the dataset using 10 folds in K-fold Cross-validation. We consider all of the features for X, excluding the "price" column. We have a column "price" as y. The general approach for the methodology is depicted in figure 7.

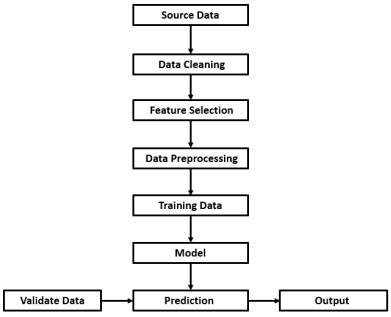

Figure 7: Methodology pipeline

4.1 Linear regression

The model can estimate unknown parameters from the data and use linear predictor functions to model the relationship. Linear regression examines if the independent variables are significant to predict the price. For example, the bigger the house, the higher the price. Linear relationships exist not only in two-dimensional space, but also multidimensional space. If two or more variables are in a linear relationship, a linear regression model can be used to predict the price based on the historical data.

4.2 Ridge Regression

Any data that indicates multicollinearity can be analyzed using the model-tuning technique known as ridge regression. This technique carries out L2 regularization. When the problem of multicollinearity arises, least-squares are unbiased, and variances are big, this leads to predicted values being distant from the actual values[5]. A penalty (L2) is applied to the squared error cost function when bias is added to the regression estimates. This causes the method to converge to linearly separable data and avoids over-fitting, which in turn lowers the standard errors. Regularization is a term used to describe this bias. By adjusting the lambda

parameter, model coefficients in ridge regression can be modified. Using Ridge() from Scikit library, we carried out a Ridge regression. We performed the modeling for the training dataset and the test dataset by varying different alpha values. When alpha was set to 0.1, the outcome was better. We used a range of alpha values, from (0,1).

4.3 K-Neighbors Regressor

K-Neighbours regression is a non-parametric method for approximating the connection between independent variables and continuous outcomes by intuitively averaging observations from the same neighborhood. To forecast the values of any new data points, the KN Regressor algorithm makes advantage of "feature similarity".[3] This suggests that a new point is valued according to how much it resembles the points in the training set. [4] The distance between a test observation and each observation in the training dataset is calculated, and the nearest K neighbors of the test observation are then found. Every test observation experiences this, which is how the resemblance is discovered.

4.4 Decision Tree Regressor

This model creates regression using a tree-like topology. A dataset is divided into more manageable chunks as a connected decision tree is incrementally built. The maximum number of leaf nodes that our model can have is 70. After splitting the data using the train test split method, a Decision Tree Regressor model is initialized and fits to the training data using DecisionTreeRegressor() module of scikit-learn.

4.5 Random Forest

With the help of several decision trees using a method known as Bootstrap Aggression, the ensemble approach Random Forest can carry out classification and regression tasks. Instead, relying solely on one decision tree to make a forecast, this method combines many decision trees. Here, we employ the RandomForestRegressor to both determine the most crucial variables, that is, the features to assist in price prediction. Importance assigns each feature a score based on how valuable or beneficial it was when building the model's enhanced decision trees. A variable's relative relevance increases the more decision trees are utilized to make important judgments. As a result, we can utilize feature importance to evaluate our data and identify the key characteristics that characterize our forecasts. The predictor variable in this instance, as seen in the bar chart above, is related with a taller bar, which denotes a stronger importance for the Random Tree Regression Model's price prediction.

4.6 Support Vector Regressor

In SVR, the errors are fitted using a specific margin of tolerance or threshold value. In order to conduct the linear separation support vector regression uses RBF kernel, which increases the dimension of the feature space from the data[6]. We used the Scikit library's support vector machines module to perform a support vector regression model. The dataset was split into training and testing data, and then the model was put into action. By switching the kernel to various techniques and tracking the values, we achieved improved outcomes. In the end, we decided to employ the RBF default algorithm for this model.

4.7 Neural Networks

The Keras library was utilized to generate this model. A sequential neural network model is used to this dataset. To train the neural network, we used three hidden layers, four hidden layers, and five hidden layers respectively to see how it effects the results. We changed the number of nodes in each layer in an effort to achieve better results. The "linear" activation function is employed for the output layer whereas the "Relu" activation function is used for the hidden layers. The output of the neural network is shown in figure 8. As the number of nodes in each layer or the number of hidden levels increases, we get good outcomes. If we increase the number of nodes in each hidden layer, the execution time increases but the results remain relatively constant. As a result, we took into account 4 hidden layers of 20, 30, 20 ,25 dimensions respectively and 100 epochs for the neural network modeling output. We selected the linear function because we are aiming for a regression issue. The loss function was "mean squared error" and the target variable is a continuous variable.

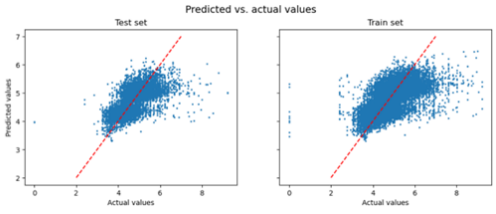

Figure 8: Predicted and Actual values for Test and Train Set using Neural

4.8 Gradient Boosting

Instead of changing the weights of the data points, the Gradient Boosting Model directly combines some weak learners to create an active learner to learn from the residual error. Each tree in a series is formed on the errors computed by the preceding tree and uses a regression tree as its base learner. Utilizing the Scikit learn library, we have adopted the gradient boosting regression from the ensemble module. First, we ran the model on the training and testing datasets with a learning rate of 0.1, a maximum depth of 3, and 100 estimators. The r-squared number was really low. The learning rate value was therefore changed from 0.01 to 1. Every learning rate value for testing and training data had a separate value. If the learning rate increases, the r-squared value for training data increases and decreases for testing data. To improve performance, we started modifying the settings for estimators and the maximum depth, which produced better outcomes.

4.9 XGboost

Gradient boost decision trees are implemented at high performance in the open-source package known as XGboost (like the decision trees that we have learnt). It is a machine learning model that can carry out prediction tasks whether they include classification or regression. Gradient Boosted Decision Trees' main principle is that they construct a succession of trees, each of which is taught to try to rectify the errors of the tree before it in the series. Like RandomForestRegressor, XGboost assign importance to the variables to aid predictions for the target variable, price. Some reasons why XGBoost outperforms other tree algorithms are as follows: regularised loss function, the weights of each new tree can be scaled down by a given constant, reducing the influence of a single tree on the final score and column-sampling, which functions similarly to random forests thereby factoring feature importance with high correlation to the target variable [7].

5 Results & Discussions

After training and testing of the models, a base performance was recorded, that is, performance based on random train split samples and analyses made on the results for each model. The metrics used to measure performance were R square, mean squared error, mean absolute error. The experiment was divided into two parts, first using the train test split and then cross validation methods.

5.1 Train Test Split Experiment Results

The table 1 shows the results for the 80% trained dataset and 20% testing sets for a total of 39k datasets. The results are presented in table 1. The results indicate that Random Forest is the best and Gradient Boost provides the second-best results with larger r-square scores and the lowest MAE and MSE errors. The Random Forest has an r-square value of 0.593 which outperforms all the other r-square scores the models we have. It has the lowest mean absolute error and lowest mean square error of 0.316 and 0.193 respectively. As expected, Linear regression, and Ridge regression performed least by having lower r-square values and larger MAE and MSE errors. Support vector regressors have some compatible results with the best performers. But it took a lot of time to train the models.

Table 1: Results with train-test split approach

Models	MAE	MSE	R Square
Linear Regression	0.391	0.278	0.413
Ridge Regression	0.393	0.283	0.413
K-Neighbors Regressor	0.356	0.215	0.545
Decision Tree Regressor	0.432	0.365	0.228
Random Forest	0.316	0.193	0.593
Gradient Boost	0.328	0.208	0.570
XGboost	0.341	0.218	0.54
SVR	0.336	0.220	0.537
Neural Network	0.373	0.262	0.453

5.2 K-fold Cross Validation Experiment Results

Cross-validation helps to have a better generalization of our model's training since at each iteration in the training and testing processes all the data get passed the model and the 10th part (since the number of splits was configured for 10 splits) used for evaluation on the defined metric(s). The table 2 below summarizes the results for the models but without the neural networks. Table 2 indicates that the gradient boost has the best results by having the highest r-square score of 0.554 and the lowest MAE and MSE errors of 0.330 and 0.213. The extra gradient boost performed better after the gradient boosting by having the r-square, MAE, and MSE scores of 0.531, 0.349, and 0.226 respectively. As noted in the case of train-test split performances, the linear regression and ridge regression performed the least again when the k-fold cross-validation approach was adopted. We

believe the linear regression and ridge regression did not perform well as we did not have a linear dataset. As opposed to linear regression and ridge regression the decision tree regressor performed better this time by having an r-square error of 0.501 and MAE and MSE score of 0.455 and 0.412 respectively. The best performer random forest moved the third place in the case of k-fold cross-validation. Although, the randon forest performed good in the train test split but not as expected in the cross validation, this is possible because of the different data partition in the two methods. However, overall, the cross validation serves the best generalization and stability of the models' performances, ensuring controlled overfitting and/or underfitting. The gradient boost is the algorithm that kept the result consistent by not fluctuating much in both cases.

Table 2: Results with K-fold cross validation approach

Models	MAE	MSE	R Square
Linear Regression	0.396	0.291	0.390
Ridge Regression	0.397	0.293	0.397
K-Neighbors Regressor	0.375	0.260	0.434
Decision Tree Regressor	0.455	0.412	0.501
Random Forest	0.343	0.230	0.514
Gradient Boost	0.330	0.213	0.554
XGboost	0.349	0.226	0.531
SVR	0.348	0.240	0.496

Conclusions

An initial review of the various features was conducted for data preparation and cleaning to clean the dataset and fill in the missing values. We applied log transformation to the price because the price distribution was skewed and appeared to contain outliers; this allowed us to apply linear regression and other techniques that rely on a normal distribution of the predictors. To see how one feature varies in relation to another, we visualized certain features. After preprocessing, we had 10 features. We employed the chi-square, correlation matrix, and feature importance methods for feature selection. Though the neighborhood group was deemed to be the least essential by the feature importance and chi-square technique, we believe the neighborhood group to be a significant feature. As a result, we conducted our analysis without feature selection. For our experiment, we used different machine-learning models to build Airbnb price-prediction models. To divide the data, we used the train-test split and k-fold cross-validation procedures. We found no difference between the two findings when so compared them, and k-fold cross-validation requires more time for the model prediction we contemplated using the train-test-split technique. To compare the performance of these different methods, we analyze the MAE, MSE, and r-squared values. In the case of the train-test split, Random Forest and gradient boosting were the best performers by having the highest r-square error and lowest MAE and MSE errors to fit the model best. In contrast, gradient boosting and extra gradient boosting performed the best for k-fold cross-validation. In general, the gradient boosting was consistent in having its position on the top list.

Our code can be found in the github repository https://github.com/PramitiBarua/Airbnb-price-prediction-.

References

[1] "Statistical Model Predicting Optimal Airbnb Listing Prices".link

[2] "Analysis of Supervised Learning Algorithms for Predicting the Price of Airbnb Listing".link

[3] A.Singh, "A Practical Introduction to K-Nearest Neighbours Algorithms for Regression".link

[4] "K Nearest Neighbour Regression".link

[5] Ferreira, Eurico J; Sirmans, G Stacy, Ridge Regression in Real Estate Analysis. The Appraisal Journal; Chicago Vol. 56, Iss. 3, Jul 1988.link

[6] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," ACM transactions on intelligent systems and technology (TIST), vol. 2, no. 3, p. 27, 2011.link

[7] Pan, B. (2018, February). Application of XGBoost algorithm in hourly PM2. 5 concentration prediction. In IOP conference series: earth and environmental science (Vol. 113, No. 1, p. 012127). IOP publishing.link

[8] New York City Open Airbnb Open Data link