# Airbnb Price Prediction



*Team3:* Kevin Cheung (kevin.cheung@uci.edu), Kathy Han (kathy.han@uci.edu), Changyeol Lee (changyl8@uci.edu), Peggy Lee (peggyl3@uci.edu), Yuqing Wang (yuqinw32@uci.edu)

**Table of Contents 1**

[**I. Introduction**](#_zifxx8dbusuh) **2**

[**II. Business Implications**](#_o35d471cf0p4) **2**

[**III. Dataset summary and description**](#_xj2swo3r32s5) **2**

[**IV. Visualization**](#_e1iaornfg5fk) **3**

[**V. Analysis**](#_6lv1e9ri46mn) **6**

[**a. Logistic Regression**](#_3hze042suxn7) **7**

[**b. Naive Bayes**](#_s7d2rzb3ftx6) **15**

[**c. Random Forest**](#_44nrisaaoupu) **18**

[**d. K-nearest neighbors algorithm**](#_n3h4scjm921p) **21**

[**VI. Conclusion**](#_4w7vywltimuh) **24**

# I. Introduction

Airbnb is a vacation online rental marketplace, that allows travelers to book a listing from Airbnb hosts who offer up their own property for rental. As traveling around the world and exploring new cities is increasingly popular, people want to immerse themselves in the culture by living in homes instead of hotels. In this project, we want to focus on predicting Airbnb listing prices in Los Angeles and what attributes affect these pricing decisions.

# II. Business Implications

Ever since Airbnb launched in 2008, its users, listings, bookings, and revenue has increased exponentially throughout the years. In 2015, Airbnb had a total of 25 million bookings and this has increased to a total of 187 million in 2019. The number of listings has increased from 1.2 million in 2015 to 7 million in 2019. The number of Airbnb users has increased from 50 million in 2015 to 150 million in 2019 and their revenue skyrocketed from $900 million in 2015 to $4.7 billion in 2019. Airbnb hosts have earned a total of $65 billion throughout the years.

From the statistics above, we can see that as the number of listings continues to increase, hosts will need a better understanding of how to price their listings. Therefore, utilizing predictive models that analyze listing attributes and predict a listing’s price range will be beneficial for hosts. By finding the optimum price for a listing, hosts can utilize this predictive model over doing painstaking research - comparing their listing to other listings.

In addition, by being able to predict housing prices in specific regions, Airbnb can leverage this information by seeing which areas have the attributes that play a big role in determining if a listing is considered as a “high” priced listing. By finding regions with “high” price listings, Airbnb can market towards these hosts to consider their property as an Airbnb listing. Therefore, increase their revenue by gaining bigger commission from higher price listings.

# III. Dataset summary and description

For the data we found on Kaggle, there are 29 attributes and about 74.1K instances in the dataset. Here are all 29 attributes within the dataset, including a short description:

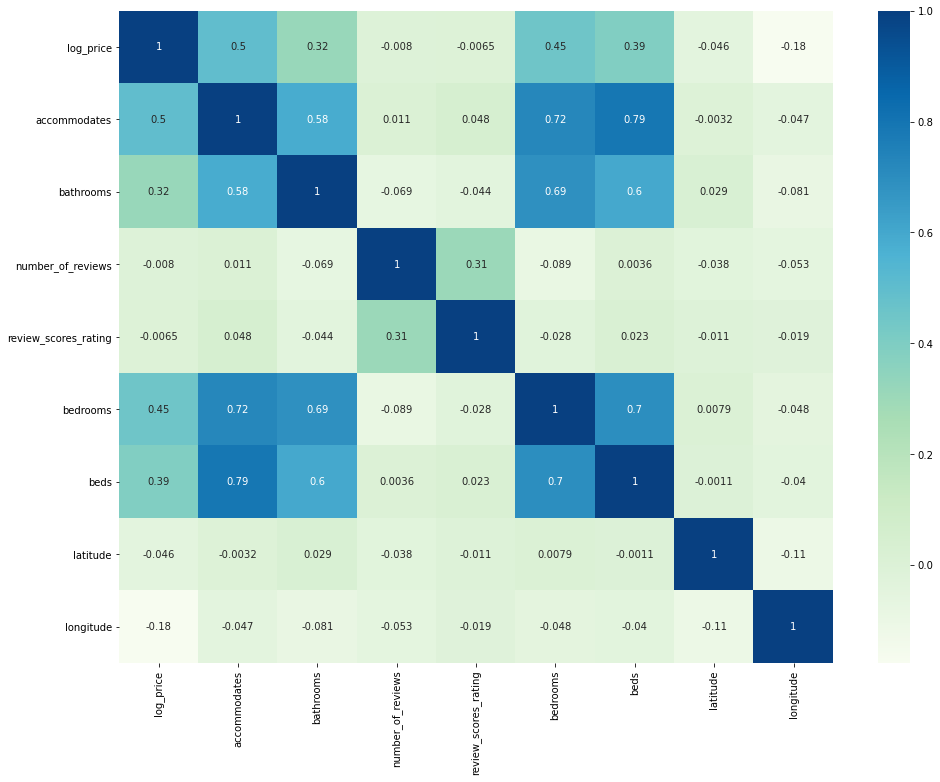
* 'id': listing ID
* 'log\_price': price of listing
* 'property\_type': property type, for example, apartment, house, townhouse, etc.
* 'room\_type': type of place, for example, entire place (have a place to yourself), private room, or shared room
* 'amenities': what is included in the listing, for example, wifi, pool, kitchen, tv, etc.
* 'accommodates': how many people the listing can accommodate
* 'bathrooms': how many bathrooms are in the listing
* 'bed\_type': type of bed, for example, real bed, futon, pull-out sofa, etc.
* 'cancellation\_policy': how strict is the host on canceling bookings, for example, flexible, super\_strict\_30 (need to cancel 30 days before booking), etc.
* 'cleaning\_fee': does customer have to pay for cleaning fee after staying at the listing
* 'city': which city the booking is listed in
* 'description’: hosts personal description of listing
* 'first\_review': date the listing was first reviewed
* 'host\_has\_profile\_pic': does the host include a profile picture
* 'host\_identity\_verified': is the host verified
* 'host\_response\_rate': does the host answer all questions from customer
* 'host\_since': the date the host became an Airbnb host
* 'instant\_bookable': can customer book a listing right away
* 'last\_review': the last date the listing was reviewed by a customer
* 'latitude': latitude of listing
* 'longitude': longitude of listing
* 'name': name of the listing
* 'neighbourhood': name of neighbourhood the listing was located
* 'number\_of\_reviews': number of reviews the listing had
* 'review\_scores\_rating': rating of listing given by customers
* 'thumbnail\_url': url of picture of the listing
* 'zipcode': zip code the listing was located
* 'bedrooms': number of bedrooms in the listing
* 'beds': number of beds in the listing

Within this dataset, our class variable of interest is named “log\_price”, which is the price of the listing. We discretized the log\_price into two 2 categories. If the log\_price is greater than or equal to the average log\_price we assign the variable as “high” corresponding to a value of 1, if the log\_price is lower than the average, we assign the variable to “low”, corresponding to a value of 0.

For the purposes of this assignment, we chose to focus on Airbnb listings located in Los Angeles, California. By doing so, this reduced the number of instances from 74.1k to 22.4k. The class distribution of log\_price is approximately 55% 0 (“low”) and 45% (“high”). Furthermore, we disregarded the following variables: id, amenities, city, description, property type, first review, host since, last review, name, neighborhood, thumbnail url, and zipcode. These variables either included free text responses, dates, or included too many objects (neighborhood property type, and zipcode), which would not be beneficial in our analysis.

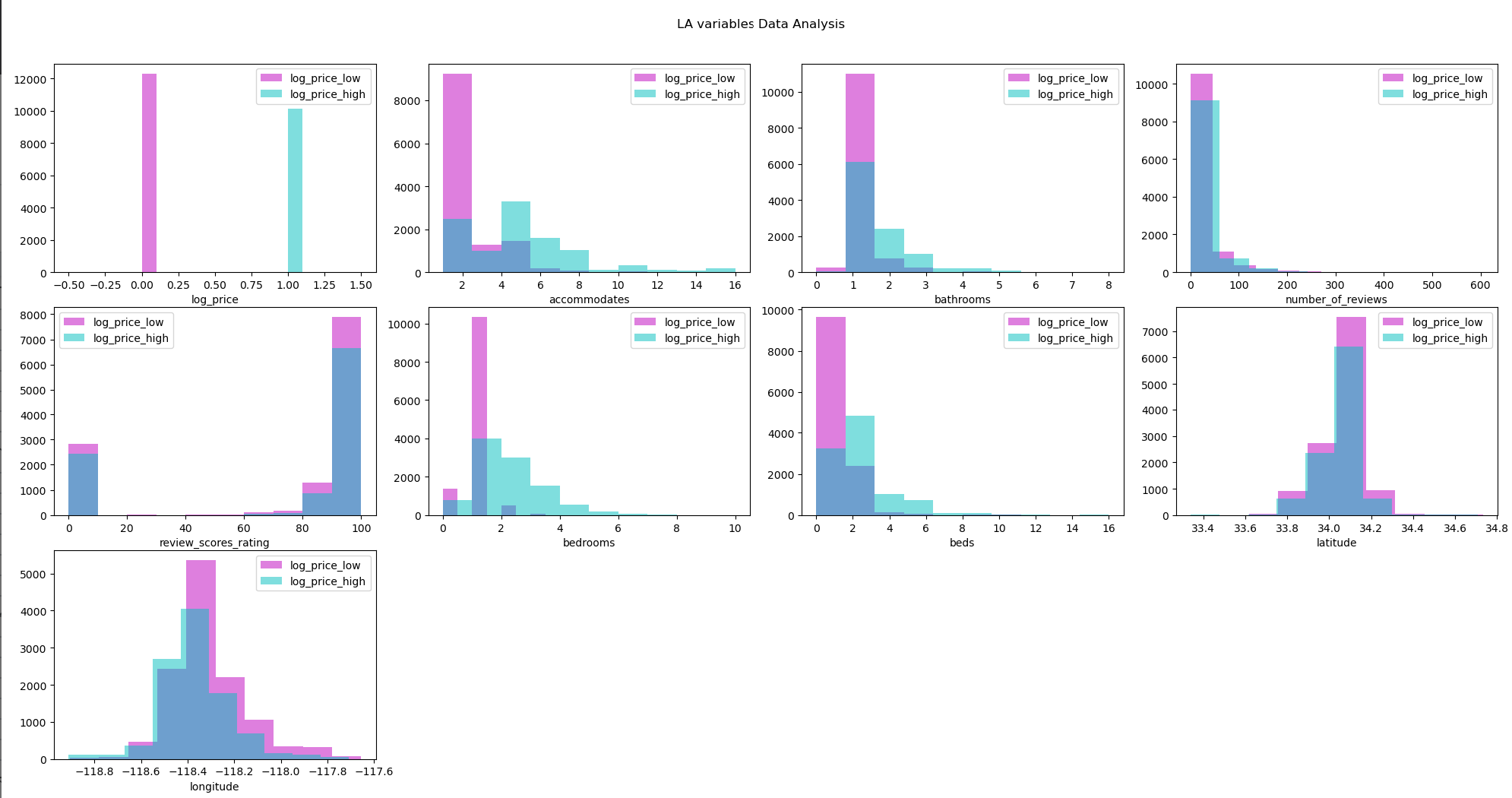
# IV. Visualization

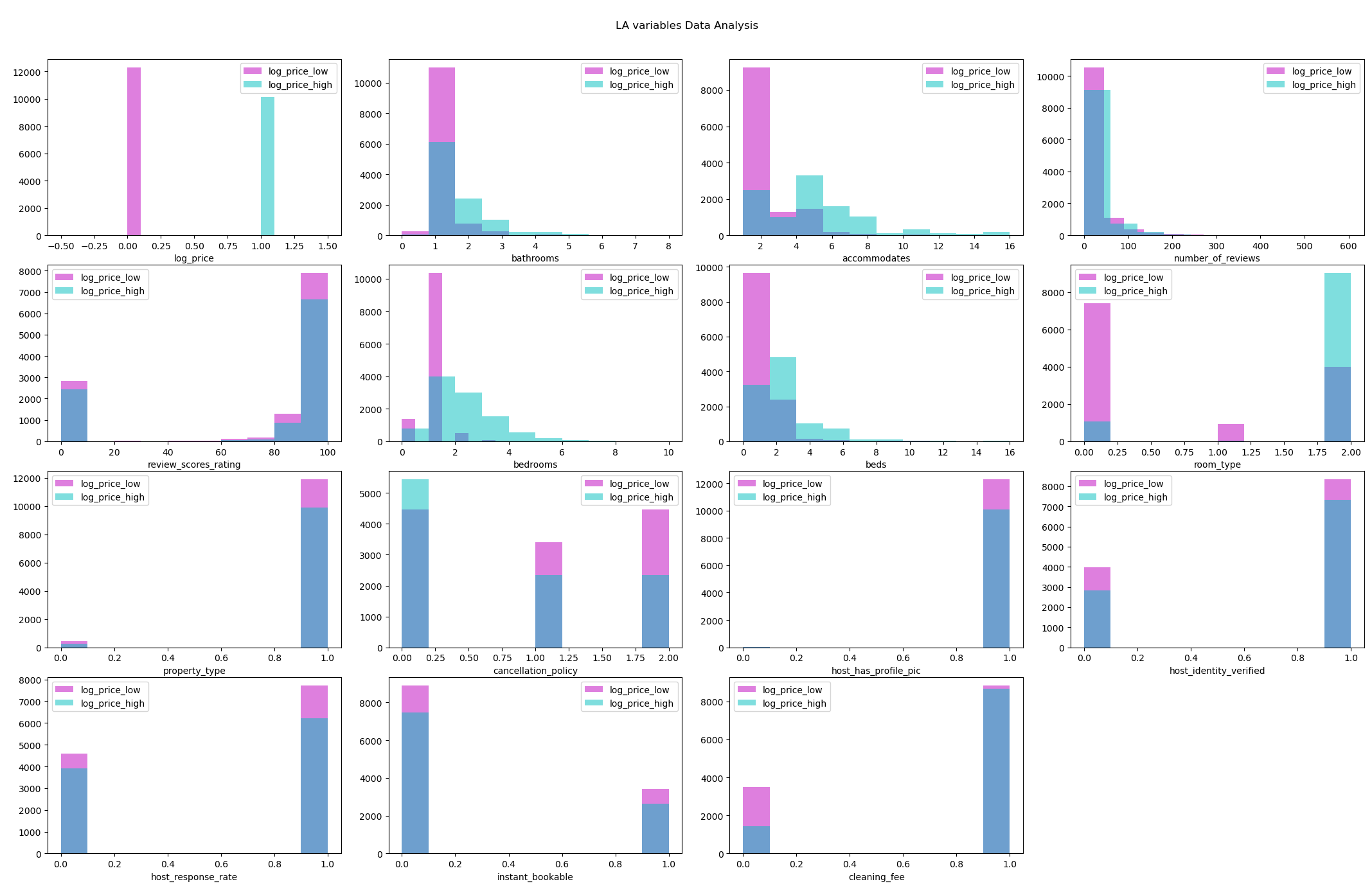
In the first correlation heatmap shown below, these variables will be used for our benchmark, as these variables are all the numeric variables that did not need to be pre-processed. In the second heatmap shown below, we extracted variables that can be processed using different techniques. We can see that multiple independent variables are highly correlated with each other. In the first and second heatmap, bedrooms, beds, and accommodates all have high correlations with one another, all exceeding 70%. Looking at the second heatmap, when viewing the class variable, we can see that room type, bedrooms, and accommodates have an average correlation of about 50% with log price. In contrast, some variables have extremely low correlation with log price such as, the host has profile pic, review scores rating, and the number of reviews.





In the histograms shown below, the first figure shows the distribution of variables that will be used in the benchmark, and the second figure shows the distribution of variables that will be used for processing with different techniques. From the figures below, we can see the skewness and separateness of each independent variable and how it affects our class variable, log price. For example, we can see that accommodations, bathrooms, bedrooms, and beds are skewed to the left, however, log price low and log price high are pretty separated, thus we can say these attributes may affect log price. On the other hand, log price low and high are not separated at all in latitude and longitude, thus these variables may not have much effect on log price. Furthermore, in the second figure below, since these variables will be used for processing, we had to categorize some of the variables, such as cancellation policy, host identity verified, host response rate, instant bookable, and cleaning fee.





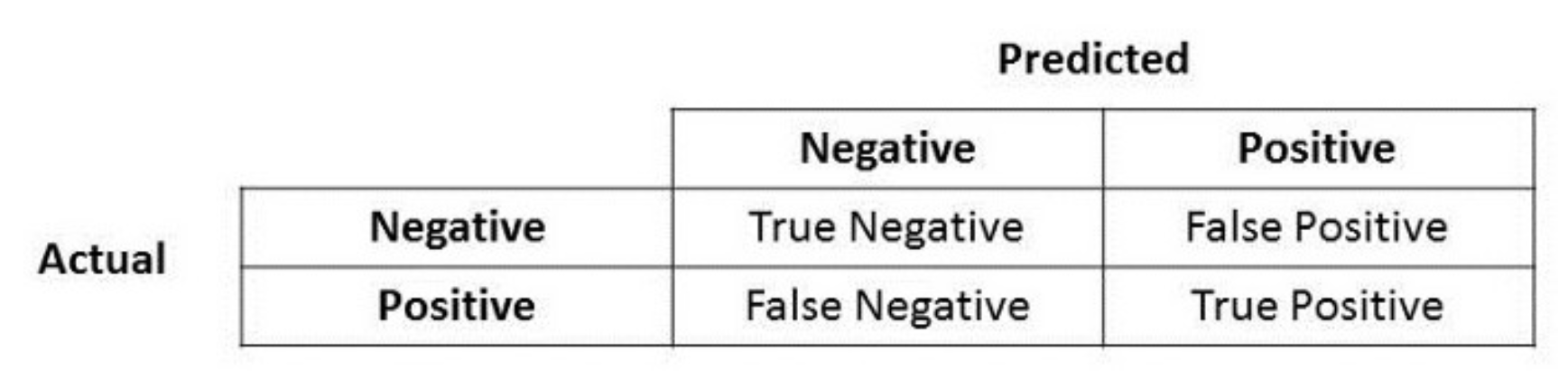
# V. Analysis

Since this is a binary classification problem, we are predicting whether an Airbnb listing is considered as “low” or “high” price, we used the following models:

* Logistic regression
* Bernoulli Naive Bayes
* Random Forest
* KNN

Within these models, we used a training size of 70% and the remaining 30% was used for testing. We also used 10-fold cross validation on Logistic regression, Naive Bayes, and KNN to prevent overfitting. We did not run 10-fold cross validation on Random forest because random forest already includes bagging method. To compare across these models, we will first run a benchmark model which has little to no preprocessing. Then, we will do the following pre-processing techniques: filter method with pearson correlation and wrapper method with sequential backward elimination. Pearson correlation finds the linear correlation between 2 variables. Using this filter method, we first created a correlation heatmap to plot the correlation between variables. From there, we set a threshold of 75%, meaning for any variables that have a correlation of 75% or higher with each other, we will choose to include the variable with the higher correlation with the class variable, and drop the other variable. In sequential backward elimination, we start with all the independent variables and each one is removed one at a time if they exceed our p-value of 0.05. Upon running each model with each pre-processing technique, we will compare the confusion matrices, accuracy scores, AUC, ROC curve, and other accuracy measures such as precision, recall, and f1-score.

When looking at the confusion matrix, we will compare the outputs under each category (as shown in diagram below).



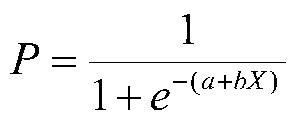
Furthermore, in recall and precision, recall is the metric to use when selecting our best model when there is a high cost associated with false negatives. Alternatively, precision is a good measure to use when there is a high cost associated with false positives. Precision talks about how precise our model is out of the total predicted positives. Recall talks about how many of the actual positives are captured in the model. For the purpose of this project, there is a higher cost associated with false negatives, or in this case, when a listing is priced as low when it is actually high, because both Airbnb and the hosts are losing on profits. Therefore, recall will be an important metric to note when comparing models. The f1-score is useful when there is a large number of true negatives and it seeks to find a middle ground between precision and accuracy. A good f1-score means that the model has low false positives and low false negatives. Thus, we will look at f1-score as well, because we have more low price listings than high price listings.

The Area Under the Curve (AUC) tells us how much the model is able to differentiate between classes - the higher the AUC, the better the model - and is used as a summary of the ROC curve.

The ROC curve stands for “receiver operating characteristic” which explains the performance of a classification model at all classification thresholds. The random classifier (red dotted line) produces a ROC point along the diagonal line. By analogy, the higher the AUC, the better the model is at distinguishing between high and low log\_price.

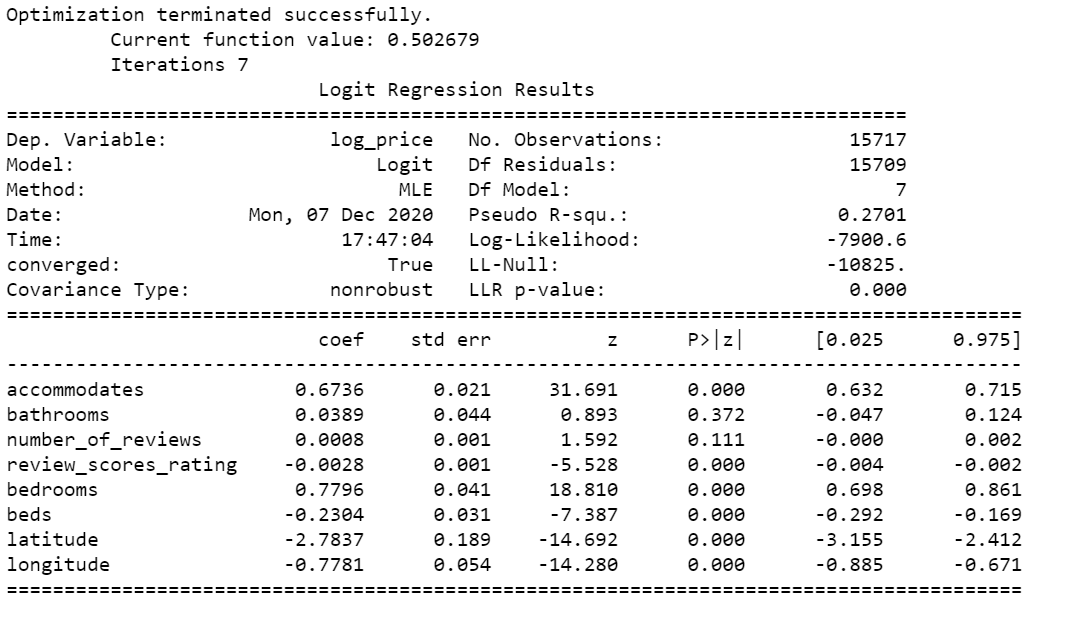
### a. Logistic Regression

Logistic regression is a statistical technique used to predict the probability of occurrence of an event using linear combinations of independent variables. The formula for the logistic regression is as follows:

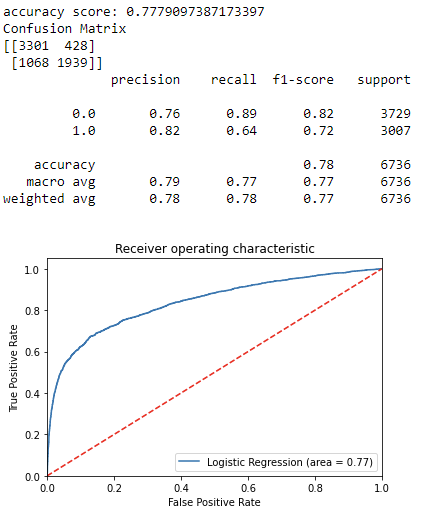


Denoting the independent variable as x, the dependent variable as P ranges from 0 to 1. Logistic regression has the following advantages. First, the assumption of a normal distribution does not apply to an independent variable because the relationship between the dependent variable and the independent variable is identified as a nonlinear relationship. Also, various types of data can be used for independent variables, including nominal, continuous and ordinal variables.Within the logistic regression model, we would like to explore what attributes have a larger effect on the probability of a listing being priced as high or low.

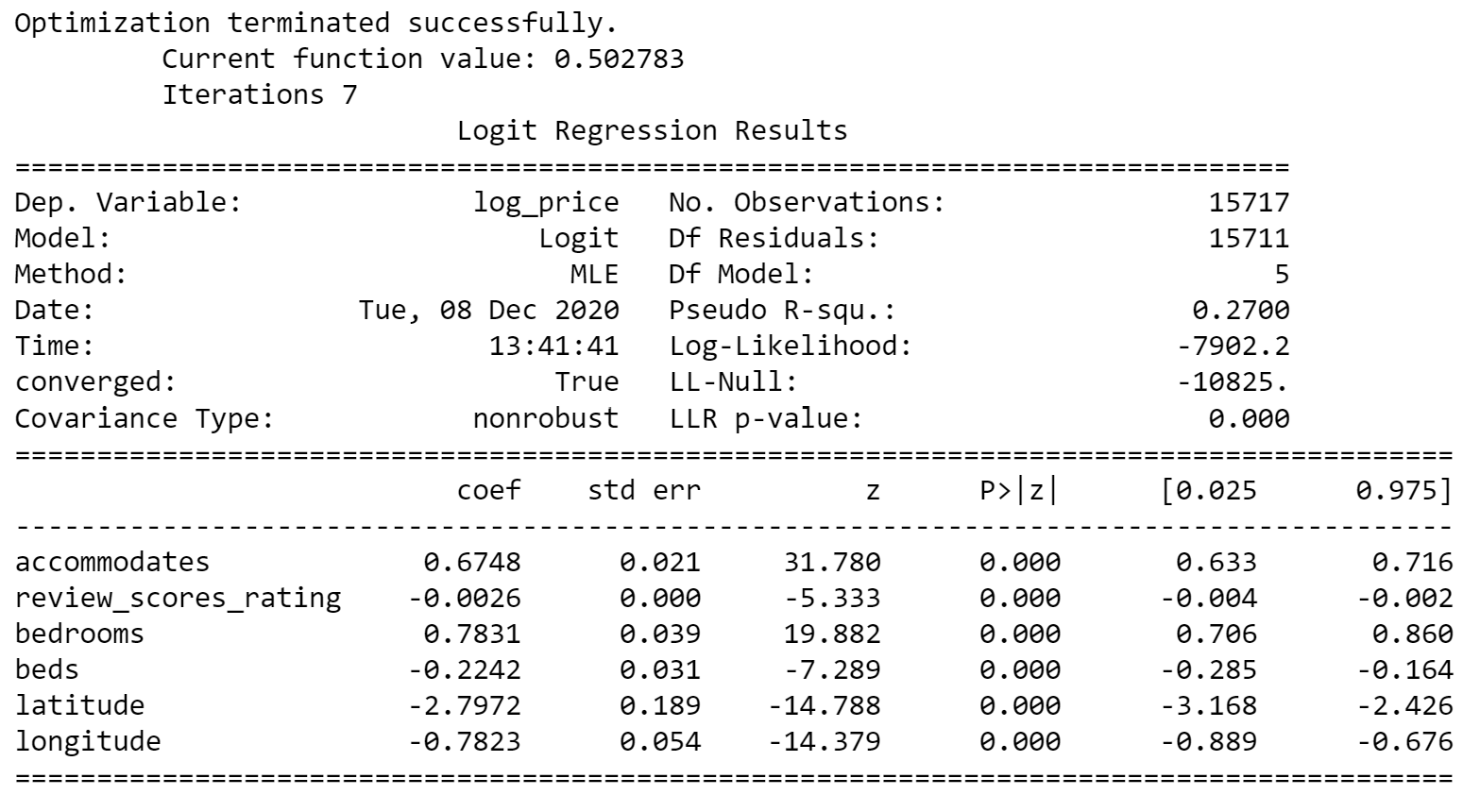
Benchmark:

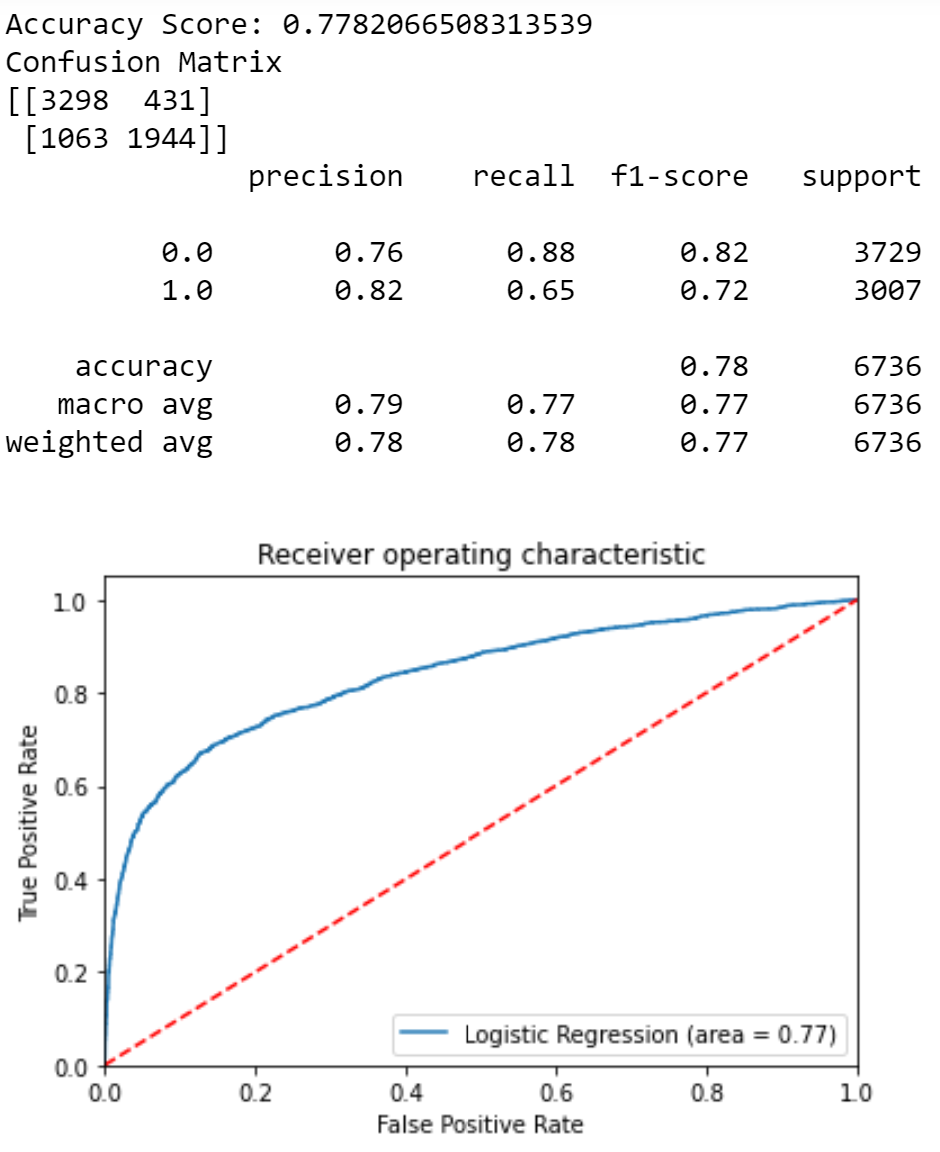


The logistic regression table above shows that accommodates, review scores rating, bedrooms, beds, latitude, and longitude attributes have a significant influence (p-values < 0.05) on log\_price. Whereas, bathrooms and number of reviews have no significant influence (p-values > 0.05) on log\_price. Furthermore, we can see that accommodates and bedrooms have the largest beneficial effect on log price as they have the greatest coefficients.



In terms of the confusion matrix, the total number of test cases is 6,736. The number of correction classifications is 5,240, while the number of incorrect classifications is 1,496. Thus, the overall accuracy is 0.778. Under recall for “0” is 0.89, whereas, the recall for “1” is 0.64, Therefore, this model will need to be improved. The scores for other outputs, such as precision and f1-score, were generally high. The AUC score is reported to be 0.77, which is acceptable (a score between 0.7 - 0.8).

We can see that bathrooms and number of reviews are not significant at alpha level 0.05, so we rerun the model without these two variables. Here is the result:

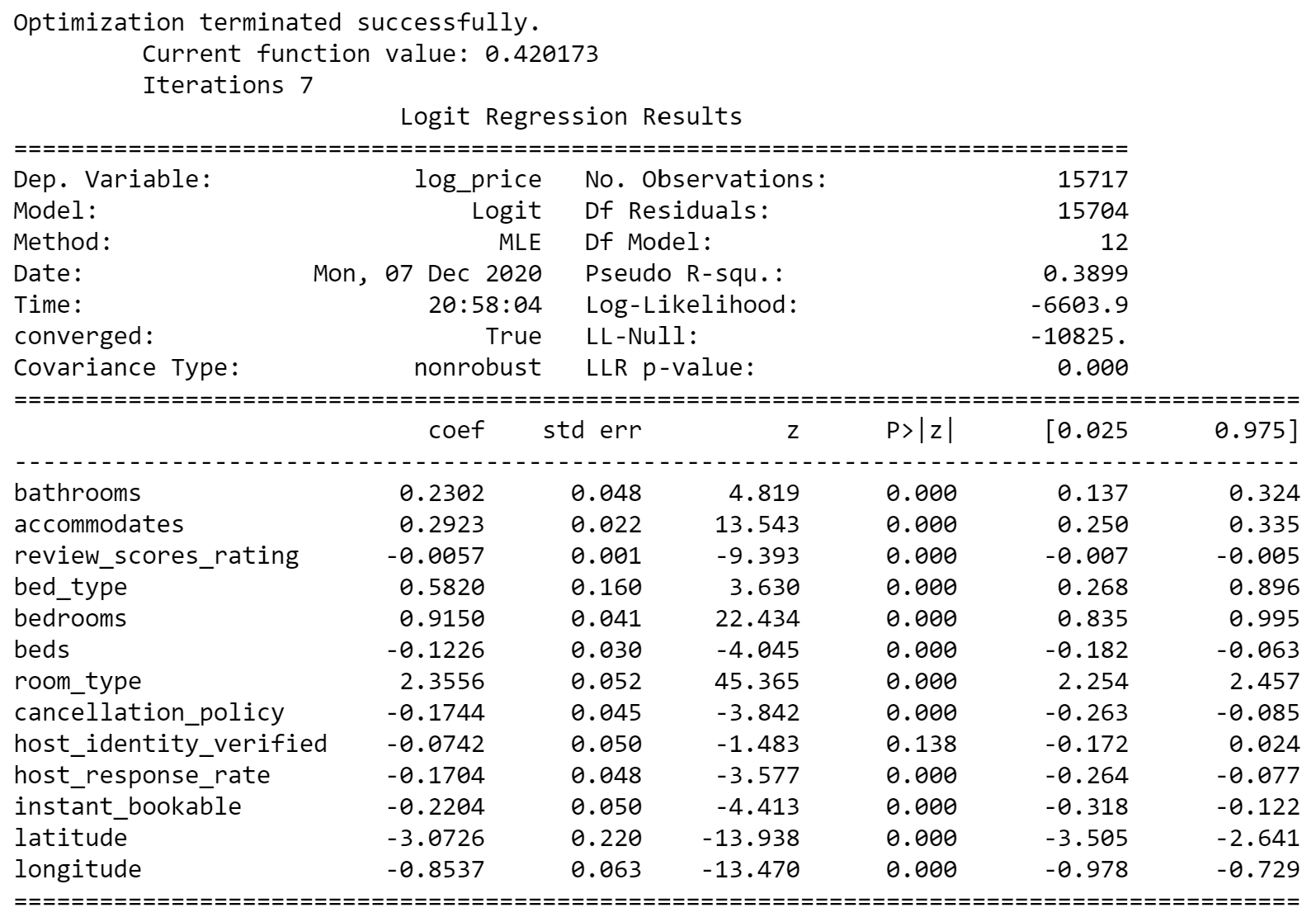
~~~~

The number of correction classifications is 5,242, while the number of incorrect classifications is 1,494. Thus, the overall accuracy is 0.778. After removing bathrooms and number of reviews from the logistic regression model, our accuracy score increased slightly. The recall score for “1” is 0.65, while the recall score for “0” is 0.88. Thus, our recall score for “1” increased by 0.01.

The AUC score is reported to be 0.77, which is acceptable. There was no significant difference when removing bathrooms and number of reviews when running the logistic regression model again.

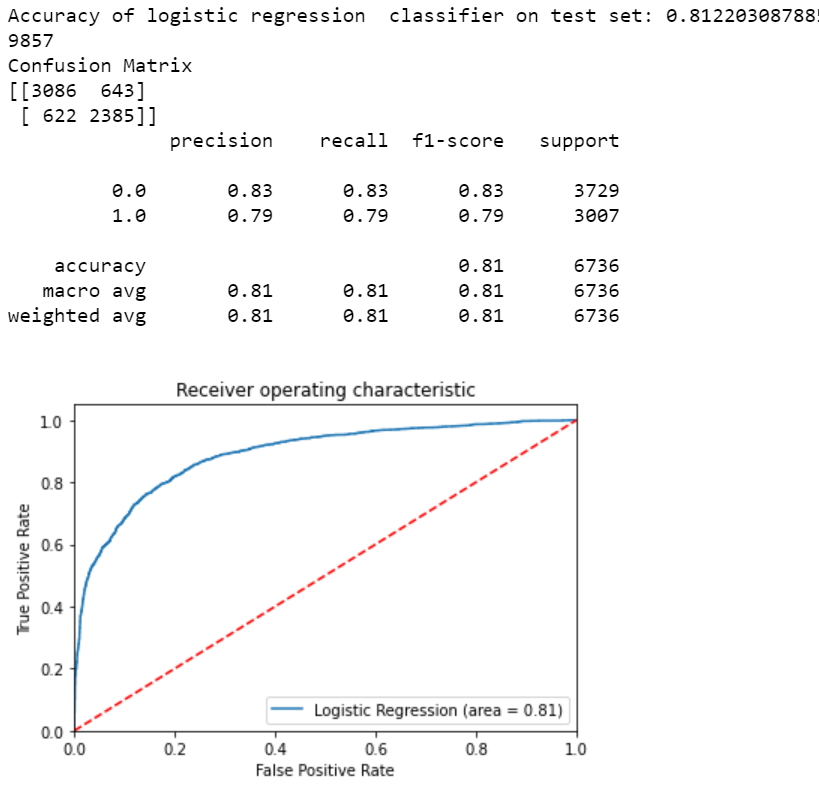
The logistic regression model didn’t improve much, however, this may change by including other categorical variables. Therefore, we proceeded to do some data cleaning and try other methods to select features, such as sequential backward elimination and pearson correlation as shown below. We first dropped all instances with NA. Then, we discretized the categorical variables into binary variables.

Sequential Backward Elimination:



Here in sequential backward elimination, we were able to include more variables. We first extracted the following variables from the dataset: 'bathrooms', 'accommodates', 'number\_of\_reviews', 'review\_scores\_rating', 'bed\_type', 'bedrooms', 'beds', 'city', 'room\_type', 'cancellation\_policy', 'host\_has\_profile\_pic', 'host\_identity\_verified', 'host\_response\_rate', 'instant\_bookable','cleaning\_fee', 'latitude', 'longitude'. After running backwards elimination on these features, we have the following features: 'bathrooms', 'accommodates', 'review\_scores\_rating', 'bed\_type', 'bedrooms', 'beds', 'room\_type', 'cancellation\_policy', 'host\_identity\_verified', 'host\_response\_rate', 'instant\_bookable', 'latitude', 'longitude'. Number\_of\_reviews, city, host\_has\_profile\_pic, and cleaning\_fee were removed after running backward elimination.

The table above showed that only the host\_identity\_verified label has no significant influence (p-values > 0.05) on log\_price. In this logistic regression we can conclude that host\_identity\_verified does not influence our dependent variable log\_price.

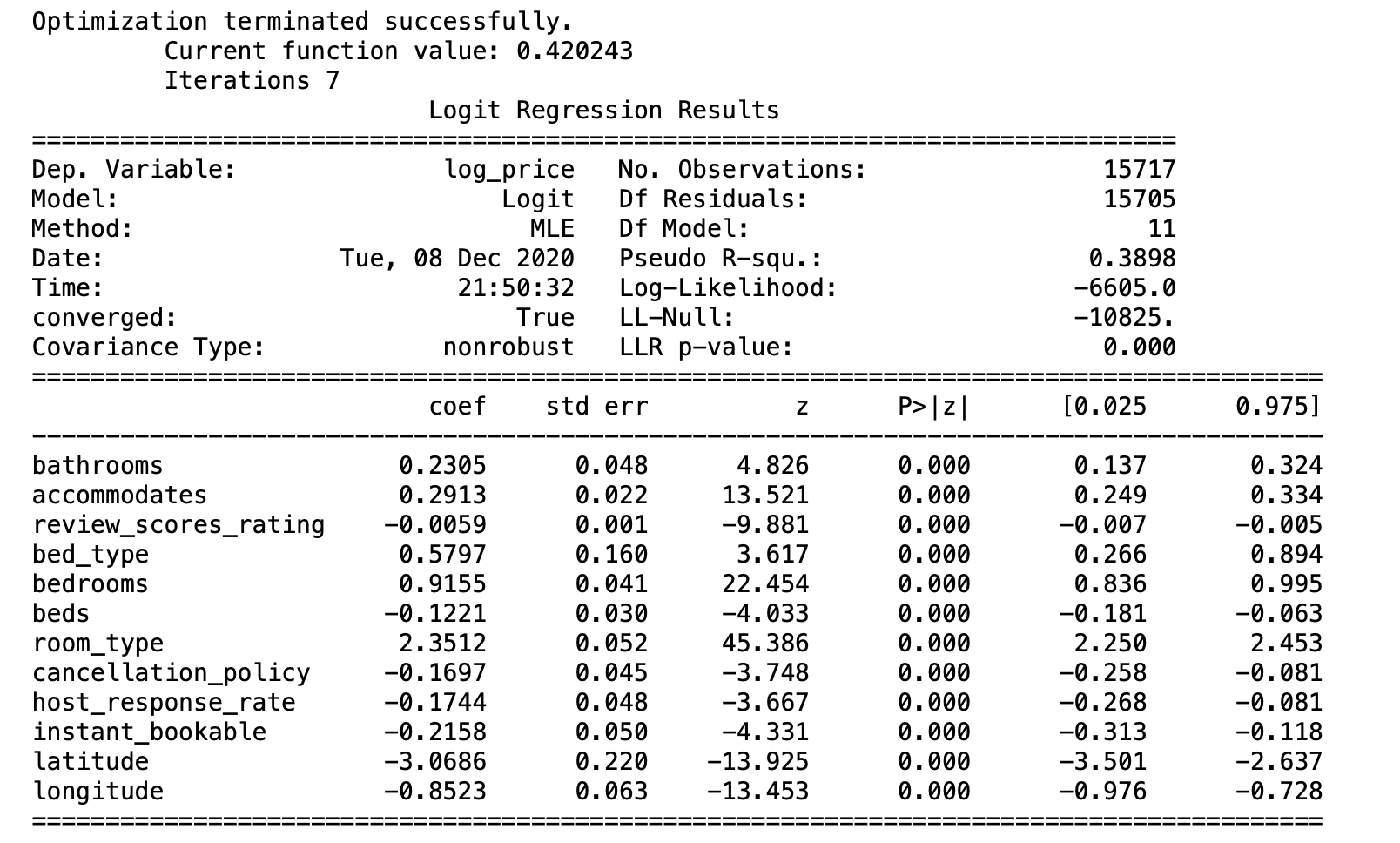


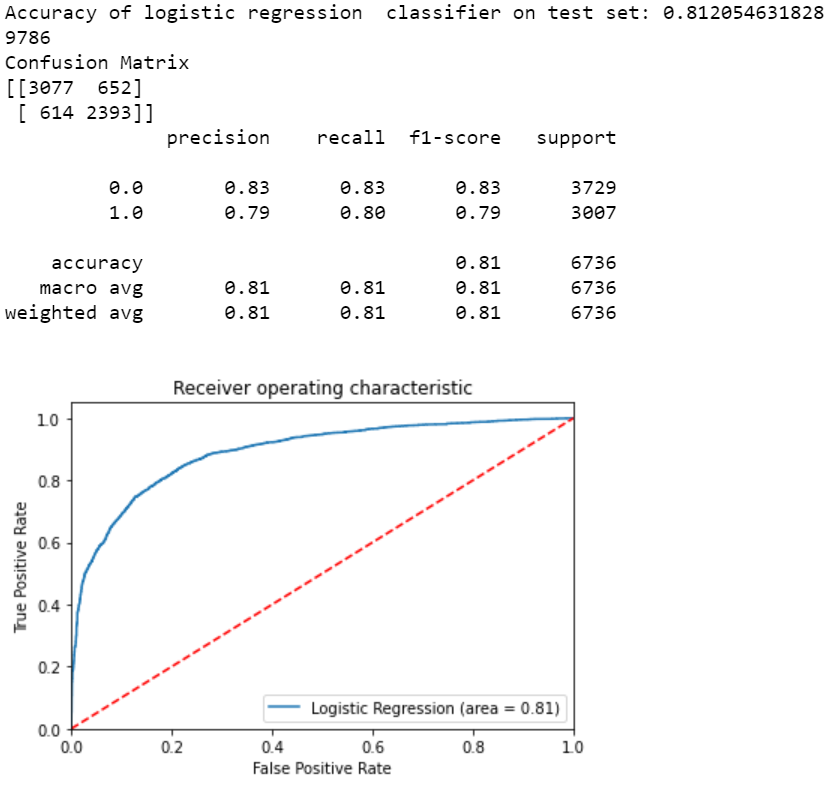
The accuracy of logistic regression with sequential backward elimination is 0.812.

In terms of the confusion matrix, the total number of test cases is 6,736. The number of correction classifications is 5,471, which is greater than the raw data of 5,242, which means the backward elimination has improved the accuracy from 0.78 to 0.81.

Regarding 0 (log\_price\_low), the precision and f1-score increased, while recall decreased(0.88 to 0.83). However, in terms of 1 (log\_price\_high), the precision decreased from 0.82 to 0.79, meanwhile the recall and f1-score both increased. Compared to our benchmark, the backward elimination model improved as our AUC score increased from 0.77 to 0.81, which is excellent (within the range of 0.80-0.90).

We can see that host\_identity\_verified is not significant at alpha level 0.05, so we rerun the model without the variables. Here is the result:

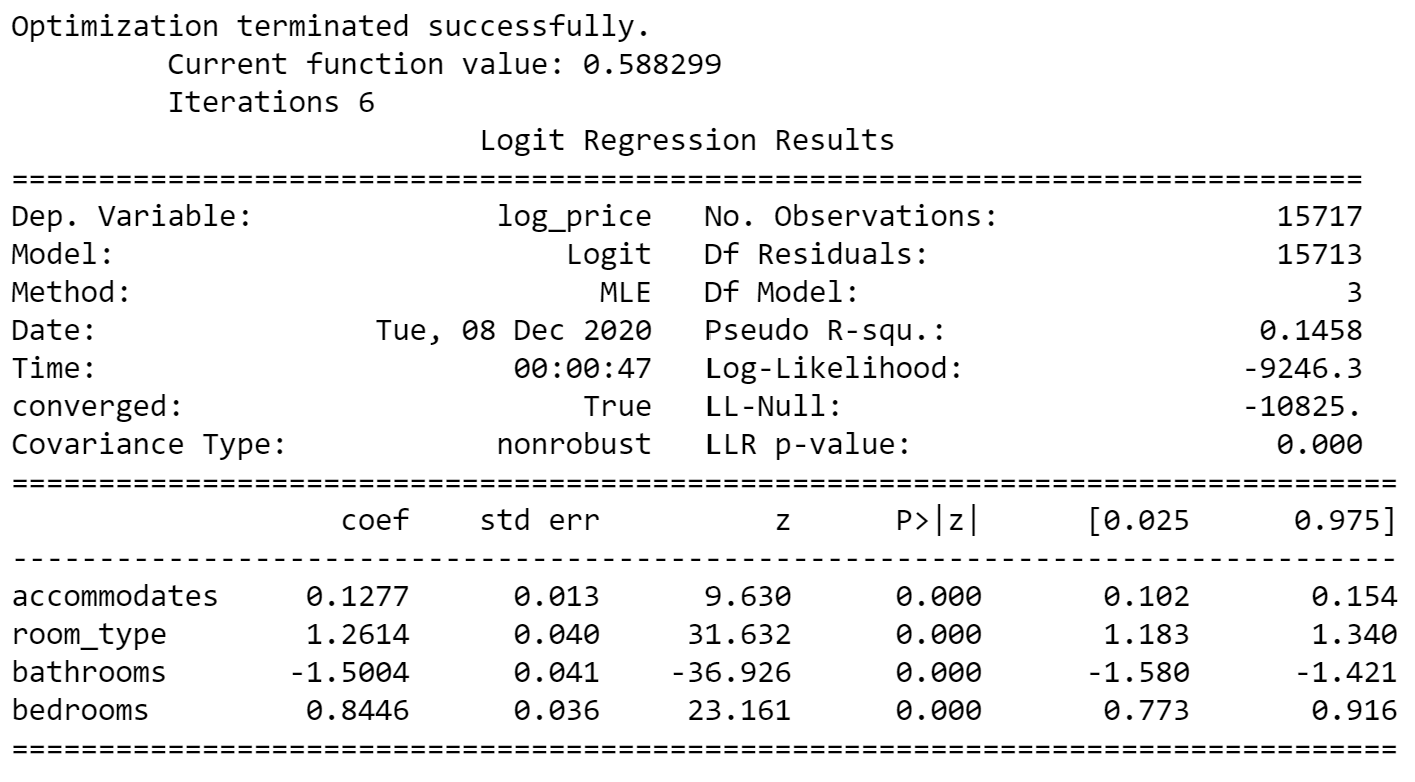
~~~~

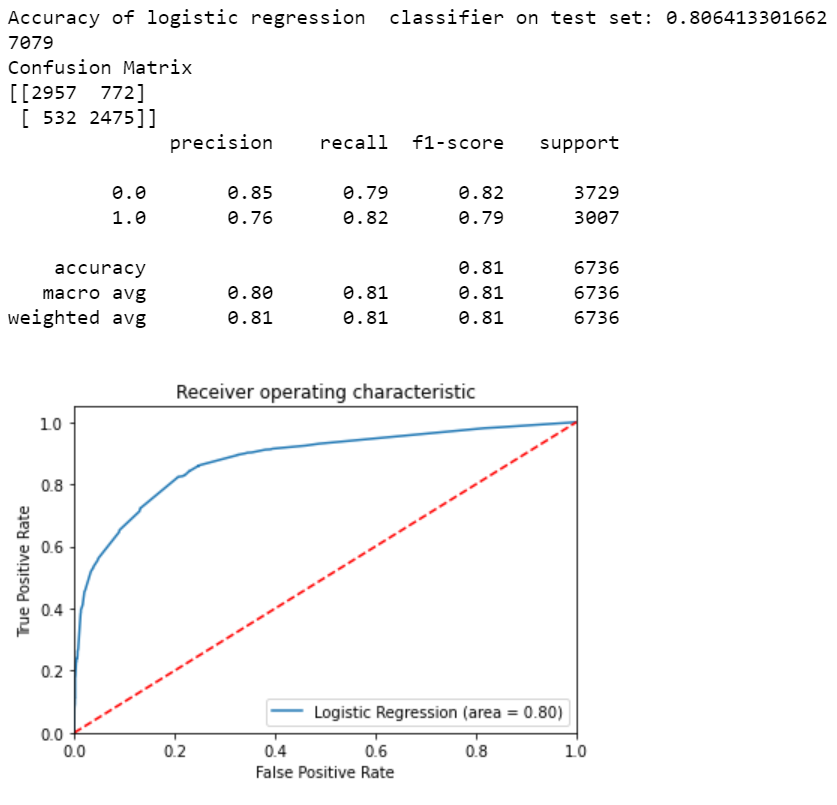
~~~~

The accuracy of logistic regression with backward elimination after removing host\_identity\_verified stayed about the same with an accuracy of 0.812. The precision, recall, and f-1 score values were also relatively the same as before.

When we compare to the benchmark, in terms of the confusion matrix, the total number of test cases is 6,736. The number of correction classifications is 5,470, which is greater than the benchmark model, which had 5,242. This means the backward elimination has improved the modeling prediction. Also, the number of incorrect classifications is 1,266, which is less than the benchmark of 1,494. The overall accuracy has increased from 0.78 to 0.81. Precision for ‘1’ dropped from 0.82 to 0.79 and precision for ‘0’ increased to 0.83 from 0.76. The recall for ‘1’ backward elimination increased significantly from the benchmark, from 0.65 to 0.80.

Pearson Correlation:





Here in pearson correlation, we set our correlation target at 30% to include attributes that had a correlation of 30% or higher with log price. Then, we set a threshold of 75% to prevent multicollinearity. In doing so, beds and accommodates had a correlation of 79%, therefore, we decided to remove the beds attribute as accommodates had a higher correlation with log price. Thus, our logistic regression model took in accommodates, room\_type, bathrooms, and bedrooms as independent variables.

Compared with the benchmark data, the accuracy of logistic regression with pearson correlation improved to 0.81 and AUC score also increased to 0.80. However, when we compared pearson correlation with backward elimination, the performance results were relatively similar. Regarding 0 (log\_price\_low), the precision increased, while recall decreased, f1-score remained the same. However, in terms of 1 (log\_price\_high), the precision increased, meanwhile the recall and f1-score both decreased.

Both pearson correlation and backward elimination were able to improve from the benchmark model. In the tables below, we compiled the model’s performance:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0(Low price) | Precision | Recall | f1-score | AUC |
| Logistic Regression (Raw data) | 0.76 | 0.89 | 0.82 | 0.77 |
| Logistic Regression (Raw data without bathrooms and number of reviews) | 0.76 | 0.88 | 0.82 | 0.77 |
| Logistic Regression (Backward Elimination) | 0.83 | 0.83 | 0.83 | 0.81 |
| Logistic Regression (Backward Elimination with out host\_identity\_verified) | 0.83 | 0.83 | 0.83 | 0.81 |
| Logistic Regression (Pearson Correlation) | 0.85 | 0.79 | 0.82 | 0.80 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1(High price) | Precision | Recall | f1-score | AUC |
| Logistic Regression (Raw data) | 0.82 | 0.64 | 0.72 | 0.77 |
| Logistic Regression (Raw data without bathrooms and number of reviews) | 0.82 | 0.65 | 0.72 | 0.77 |
| Logistic Regression (Backward Elimination) | 0.79 | 0.79 | 0.79 | 0.81 |
| Logistic Regression (Backward Elimination with out host\_identity\_verified) | 0.79 | 0.80 | 0.79 | 0.81 |
| Logistic Regression (Pearson Correlation) | 0.76 | 0.82 | 0.79 | 0.80 |

​

|  |  |
| --- | --- |
| **Logistic Regression** | Accuracy |
| Train/Test Split Accuracy (Benchmark) | 0.778 |
| 10-fold Cross validation Accuracy (Benchmark) | 0.773 |
| Train/Test Split Accuracy (Backward Elimination) | 0.812 |
| 10-fold Cross validation Accuracy (Backward Elimination) | 0.805 |
| Train/Test Split Accuracy (Pearson Correlation) | 0.806 |
| 10-fold Cross validation Accuracy (Pearson Correlation) | 0.799 |
| **Remove bathrooms and number of reviews** |  |
| Train/Test Split Accuracy (Benchmark) | 0.778 |
| 10-fold Cross validation Accuracy (Benchmark) | 0.773 |
| **Remove host\_identity\_verified** |  |
| Train/Test Split Accuracy (Backward Elimination) | 0.812 |
| 10-fold Cross validation Accuracy (Backward Elimination) | 0.807 |

In the table above, we compared the accuracy between train-test split and 10-fold cross validation across the benchmark, sequential backward elimination, and pearson correlation methods. We can see that there was no significant difference between the train-test split and 10-fold cross validation methods. Therefore, since cross validation prevents overfitting, there was no overfitting present in the train-test split method across the models. In logistic regression, we observed that the train-test split method gave slightly higher accuracies.

### b. Naive Bayes

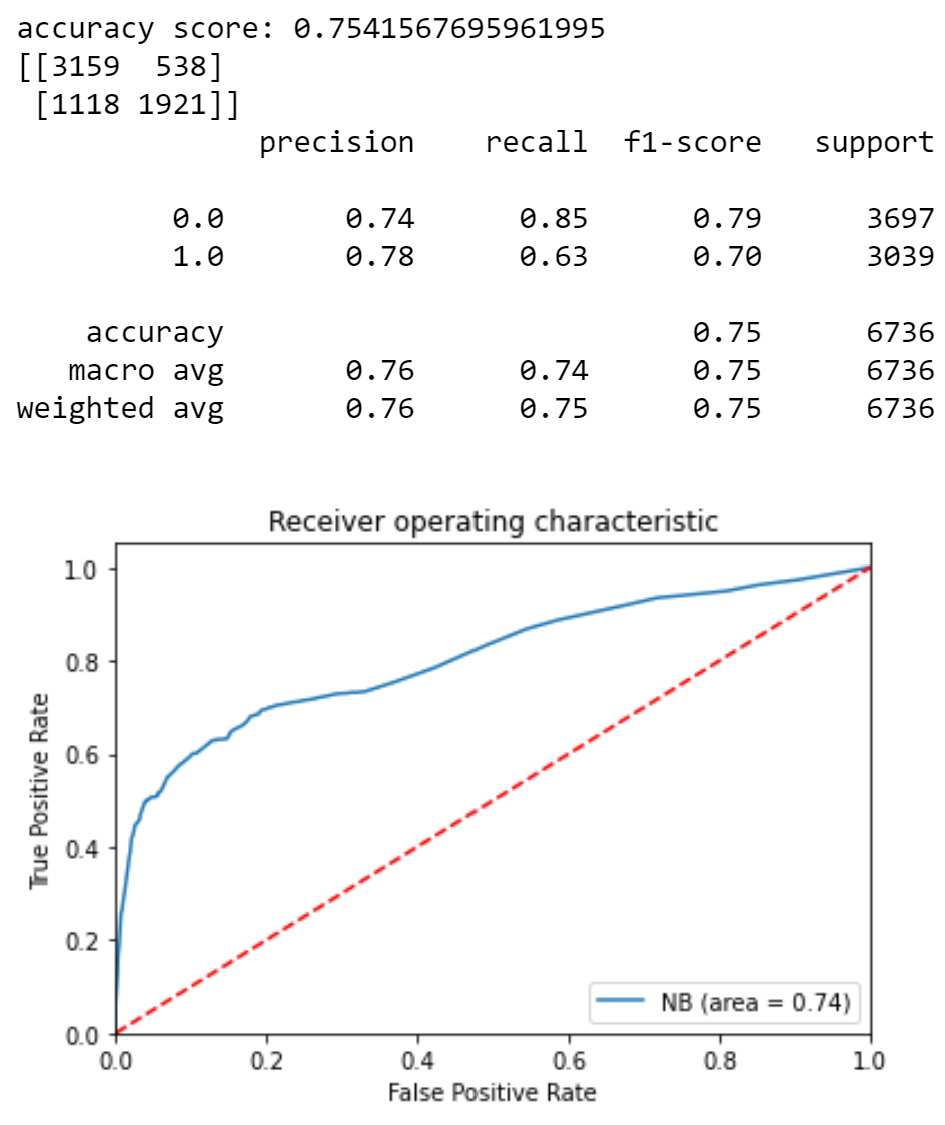
Naive Bayes is a simple machine learning technique for creating classifiers. The algorithm is trained using multiple algorithms based on general principles. All Naive Bayes commonly assume that all characteristic variables are independent. The formula for Naive Bayes is as follows:



The advantage of Naive Bayes is that Naive Bayes can be trained efficiently in a supervised learning environment. Also, the amount of training data to estimate the parameters needed for classification is very small. Naive Bayes works precisely in many complex real situations.

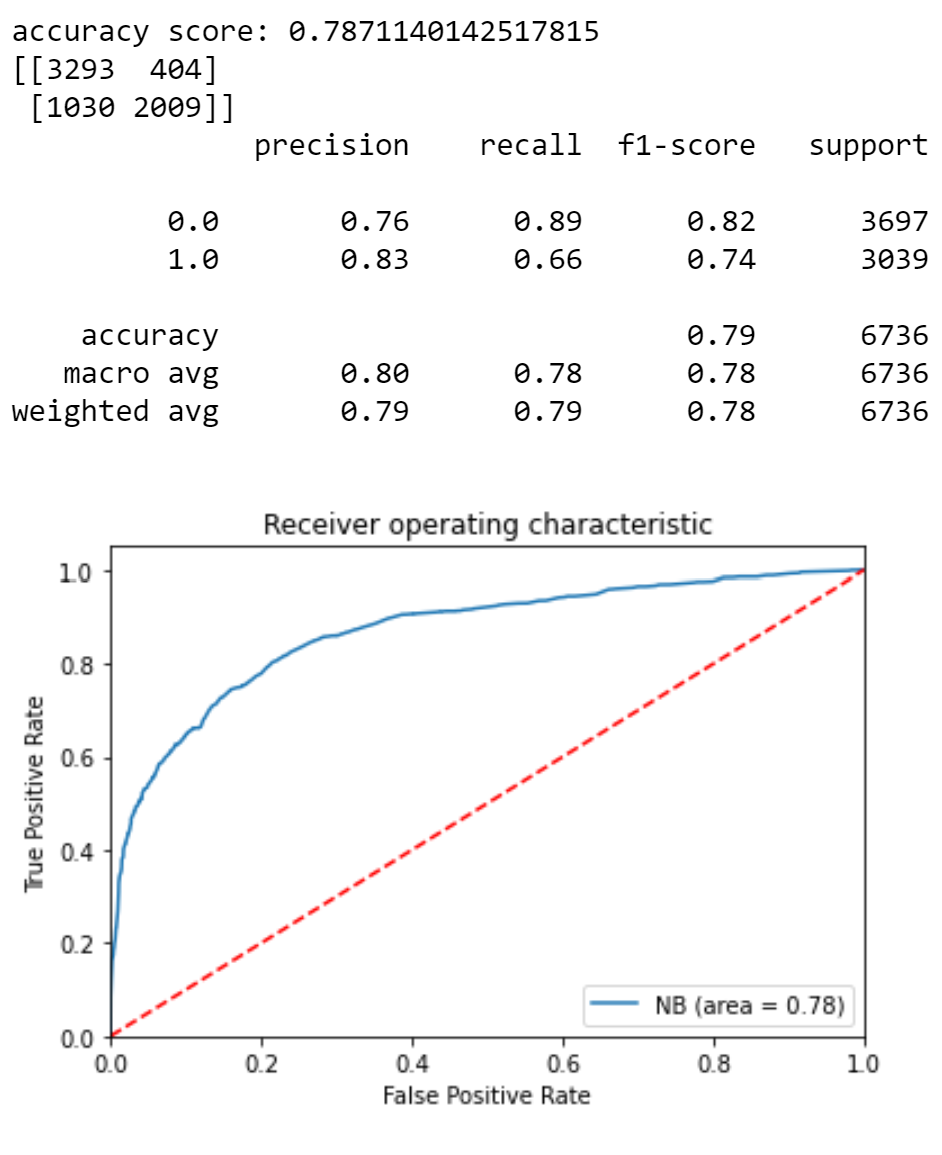
To apply Naive Bayes for our data, we needed to discretize all numerical variables of interest that we used and change them to binary. We did this by finding the average and if the value was less than the average, then we categorized as 0 and if the value was greater or equal to the average, then we categorized as 1.

Benchmark:

****

After we applied the Bernoulli Naive Bayes model on the benchmark, we observed that this model had an accuracy score of 0.754. The number of correct classifications for this model is 5,080 and the ROC was 0.75. To be specific, precision, recall, and f1-score values were high. However, the recall value of ‘1’ was 0.63, which means the stratified accuracy of '1’ is low.

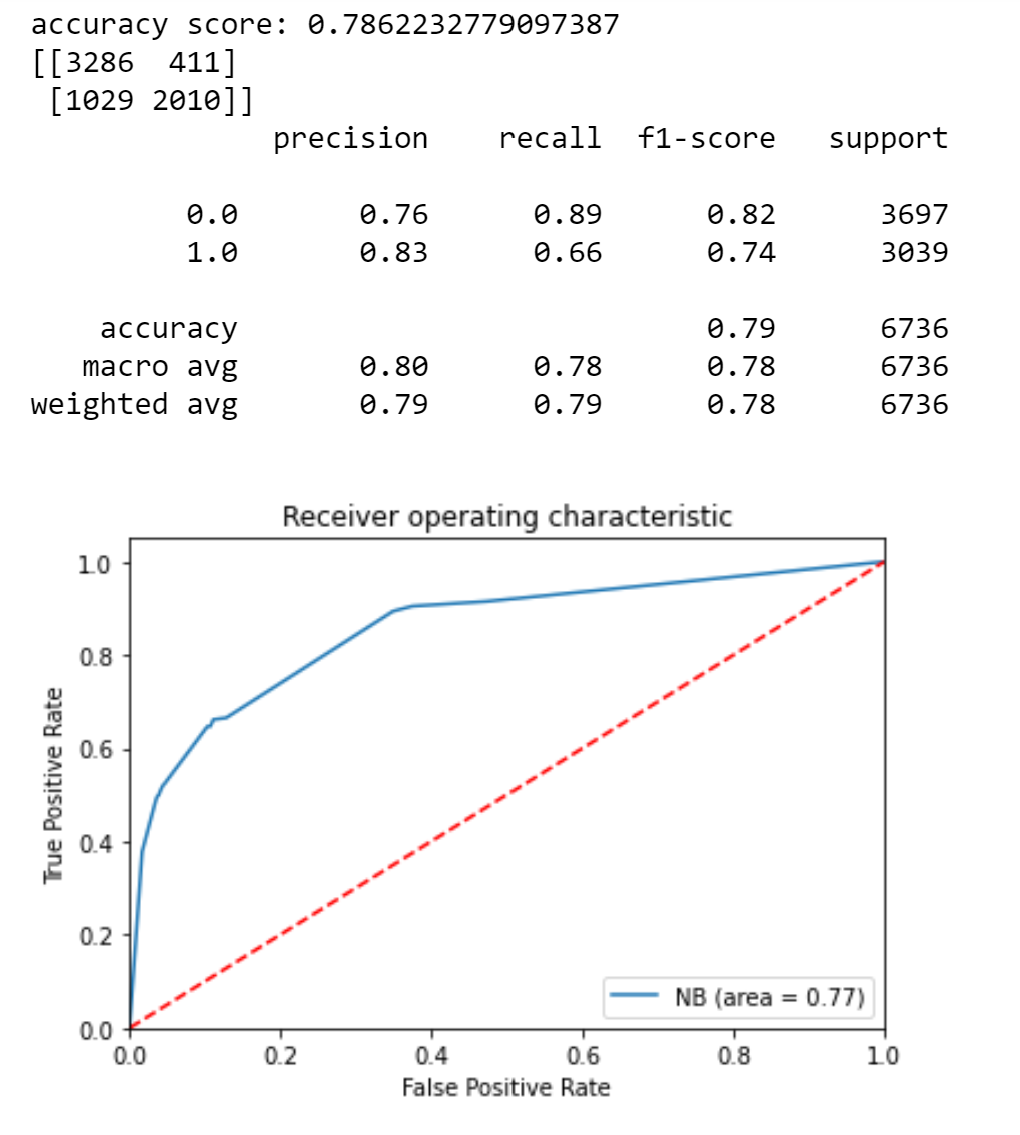
Sequential Backward Elimination:



Upon running the Bernoulli Naive Bayes model after processing with sequential backward elimination, the accuracy score improved to 0.787. The number of correct classifications was 5,302 and the AUC was 0.78. In detail, the values for precision, recall and f-score were generally high, although the value for recall was 0.66.

Comparing the model with our benchmark, the overall accuracy of the model has improved. Furthermore, every value for precision, recall and f1-score for ‘0’ and ‘1’ has increased. The AUC was higher, as well. Thus, we could conclude that the backward elimination has improved our prediction accuracy. The f1-score for both high and low price improved compared to our benchmark.

Pearson Correlation:



Upon running the Bernoulli Naive Bayes model after processing pearson correlation, our accuracy was 0.786. From the result, we can see that precision, recall, and f1-score increased compared to the benchmark model. The AUC increased 0.77, which means using Naive bayes with pearson correlation was better at predicting the class variable than the benchmark.

However, when we compared pearson correlation with the backward elimination model, pearson correlation did slightly worse than the backward elimination model.

Both pearson correlation and backward elimination were able to improve from the benchmark model. In the tables below, we compiled the model’s performance:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0(Low price) | Precision | Recall | f1-score | AUC |
| Naive Bayes (Raw data) | 0.74 | 0.85 | 0.79 | 0.74 |
| Naive Bayes (Backward Elimination) | 0.76 | 0.89 | 0.82 | 0.78 |
| Naive Bayes (Pearson Correlation) | 0.76 | 0.89 | 0.82 | 0.77 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1(High price) | Precision | Recall | f1-score | AUC |
| Naive Bayes (Raw data) | 0.78 | 0.63 | 0.70 | 0.74 |
| Naive Bayes (Backward Elimination) | 0.83 | 0.66 | 0.74 | 0.78 |
| Naive Bayes (Pearson Correlation) | 0.83 | 0.66 | 0.74 | 0.77 |

​

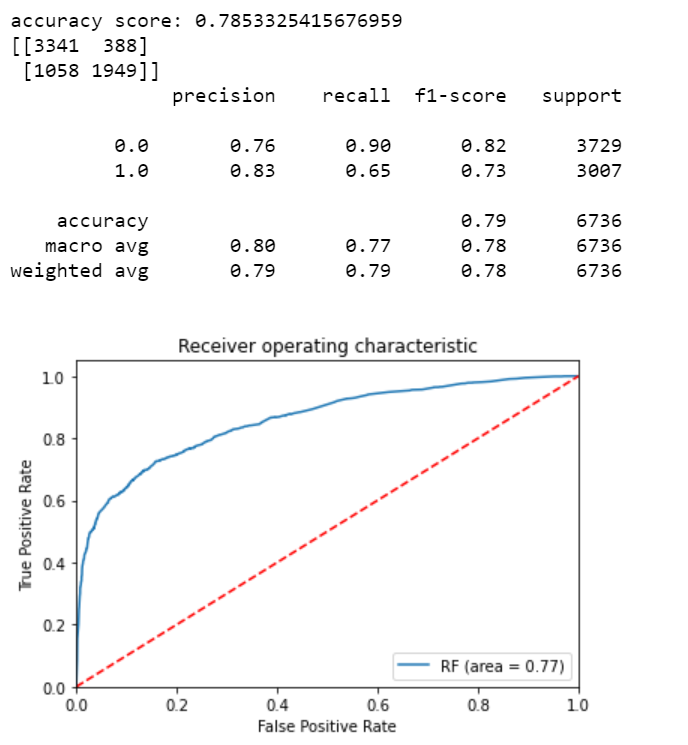
|  |  |
| --- | --- |
| **Naïve Bayes** | Accuracy |
| Train/Test Split Accuracy (Benchmark) | 0.754 |
| 10-fold Cross validation Accuracy (Benchmark) | 0.757 |
| Train/Test Split Accuracy (Backward Elimination) | 0.787 |
| 10-fold Cross validation Accuracy (Backward Elimination) | 0.791 |
| Train/Test Split Accuracy (Pearson Correlation) | 0.786 |
| 10-fold Cross validation Accuracy (Pearson Correlation) | 0.792 |

In the table above, we compared the accuracy between train-test split and 10-fold cross validation across the benchmark, sequential backward elimination, and pearson correlation methods. We can see that there was no significant difference between the train-test split and 10-fold cross validation methods. Therefore, since cross validation prevents overfitting, there was no overfitting present in the train-test split method across the models. In naive bayes, we observed that the 10-fold cross validation method gave slightly higher accuracies.

### c. Random Forest

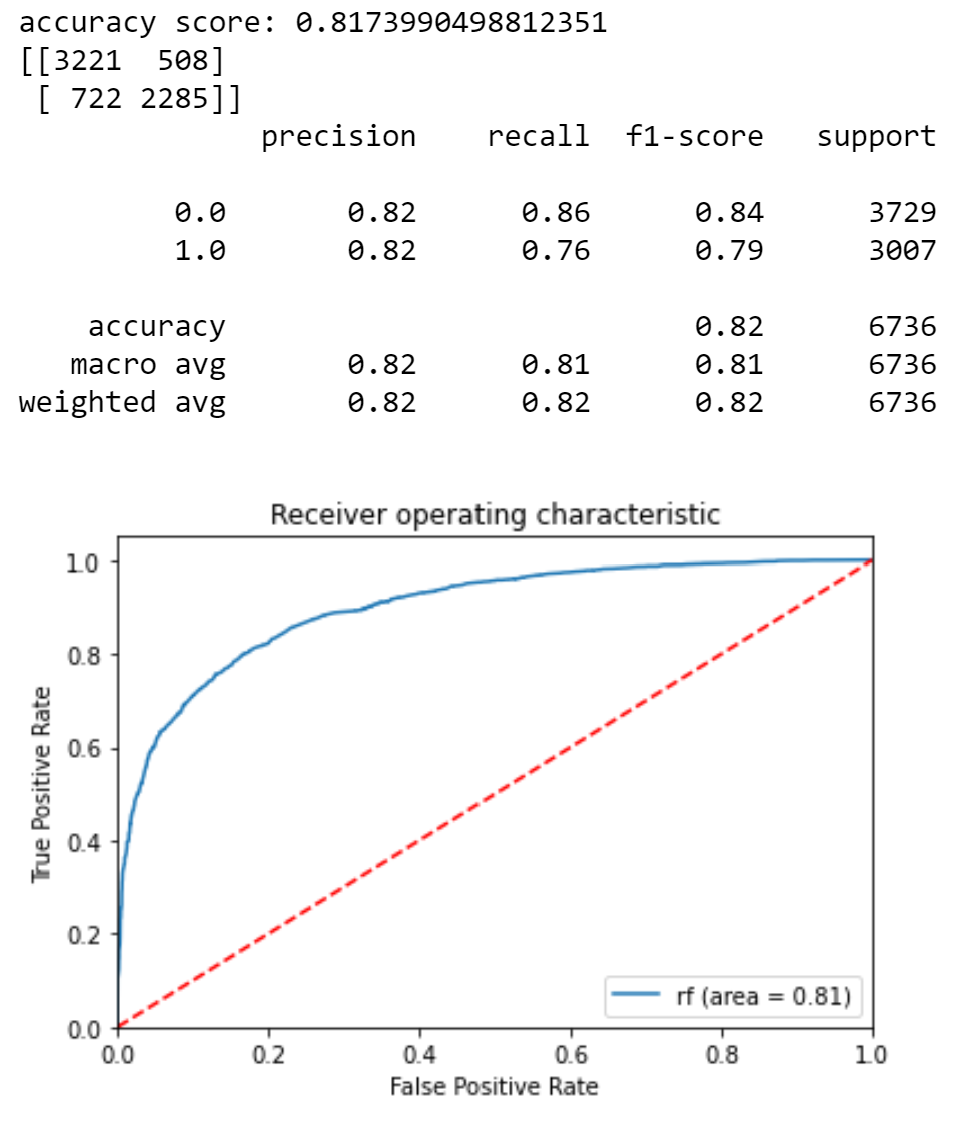
The Random Forest method is an ensemble method, which combines results from multiple trees. The biggest feature of the random forest is that each tree has slightly different characteristics from each other based on the randomness. This characteristic makes a prediction of each tree to be decorrelated, and consequently improves the generalization performance. Also, randomization makes the forest precise for data with noise, which is conducted in training each tree. Bagging, which is an ensemble learning method using random learning data extraction methods, was used in our random forest model.

Benchmark:



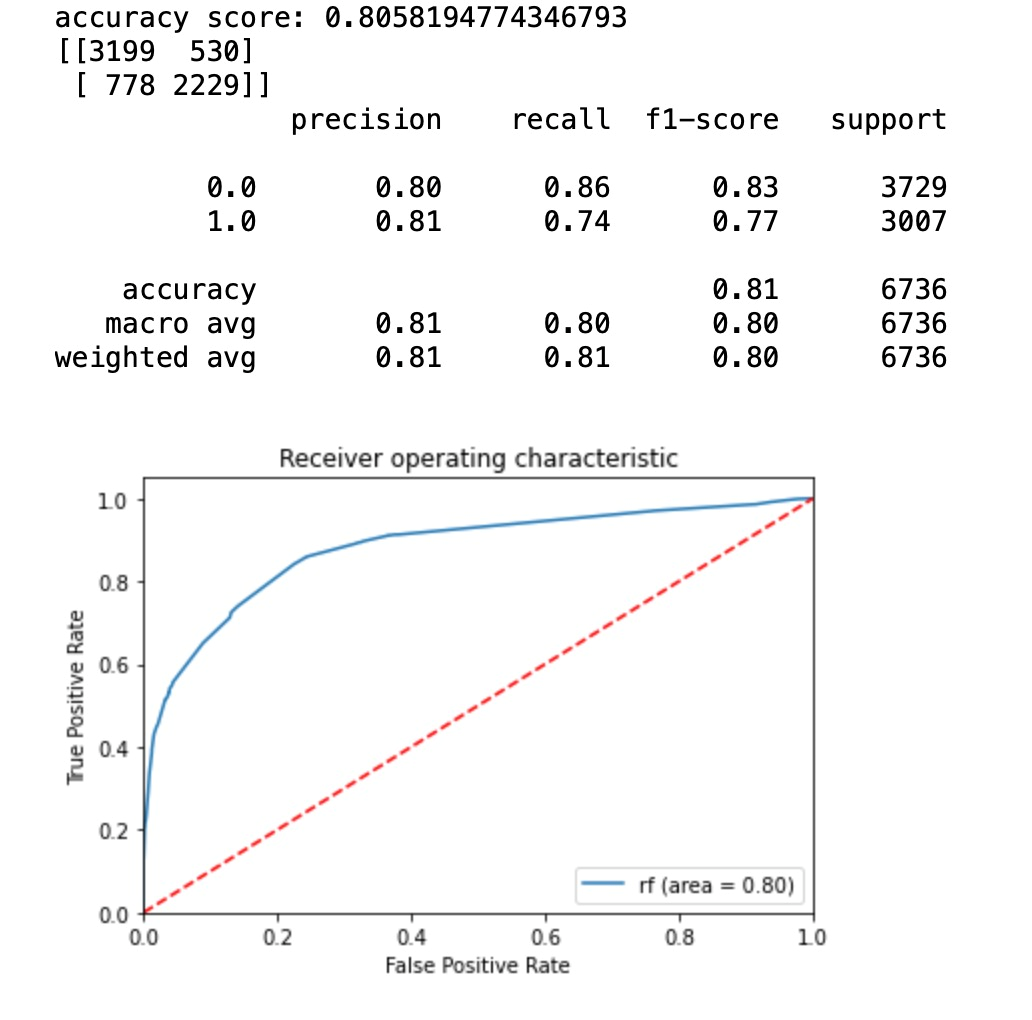
Testing our benchmark data with random forest, we observed that the overall accuracy score of this model was 0.785, with a correct classification of 5,290 and an AUC of 0.77. To be specific, although the recall value of ‘1’ was low, the rest of the values were high with more than 0.70.

Sequential Backwards Elimination:



Upon running the random forest model after processing with sequential backward elimination, an accuracy score of this model was 0.817, with an AUC score of 0.81. The precision, recall, and f1-score all have a value of over 0.70. Recall of the “0” was lower but precision and f1-score increased. Precision for “1” decreased by 1%, but recall and f1-score both increased. Compared with our benchmark, backward elimination has improved the overall accuracy by approximately 3%.

Pearson Correlation:



Upon running the Random Forest model after processing with pearson correlation, our model had an accuracy of 0.806 and an AUC of 0.80. The precision, recall, and f1-score all have a value of over 0.70. When comparing pearson correlation with the benchmark, the overall accuracy improved by about 2%. However, backward elimination had a better accuracy score by a little more than 1%.

Both pearson correlation and backward elimination were able to improve from the benchmark model. In the tables below, we compiled the model’s performance:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0(Low price) | Precision | Recall | f1-score | AUC |
| Random Forest (Raw data) | 0.76 | 0.90 | 0.82 | 0.77 |
| Random Forest (Backward Elimination) | 0.82 | 0.86 | 0.84 | 0.81 |
| Random Forest (Pearson Correlation) | 0.80 | 0.86 | 0.83 | 0.80 |

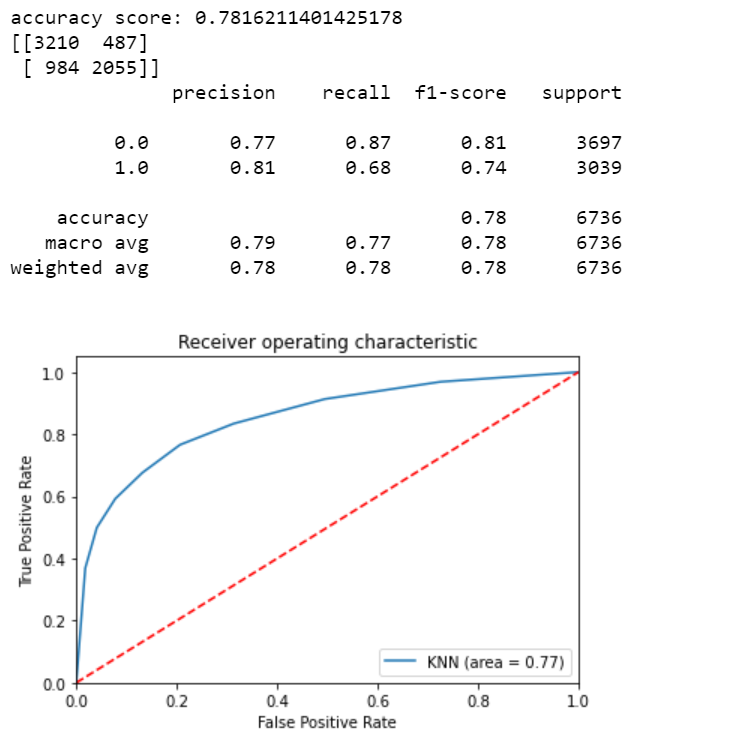
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1(High price) | Precision | Recall | f1-score | AUC |
| Random Forest (Raw data) | 0.83 | 0.65 | 0.73 | 0.77 |
| Random Forest (Backward Elimination) | 0.82 | 0.76 | 0.79 | 0.81 |
| Random Forest (Pearson Correlation) | 0.81 | 0.74 | 0.77 | 0.80 |

### d. K-nearest neighbors algorithm

K-nearest neighbors algorithm (KNN) is an algorithm that classifies data into K labels that are close to data. Euclidean distance calculation is mainly used, when measuring the distance between labels and data. The main features of KNN are simple and efficient. Also, the algorithm is precise when classifying numerical-based data.

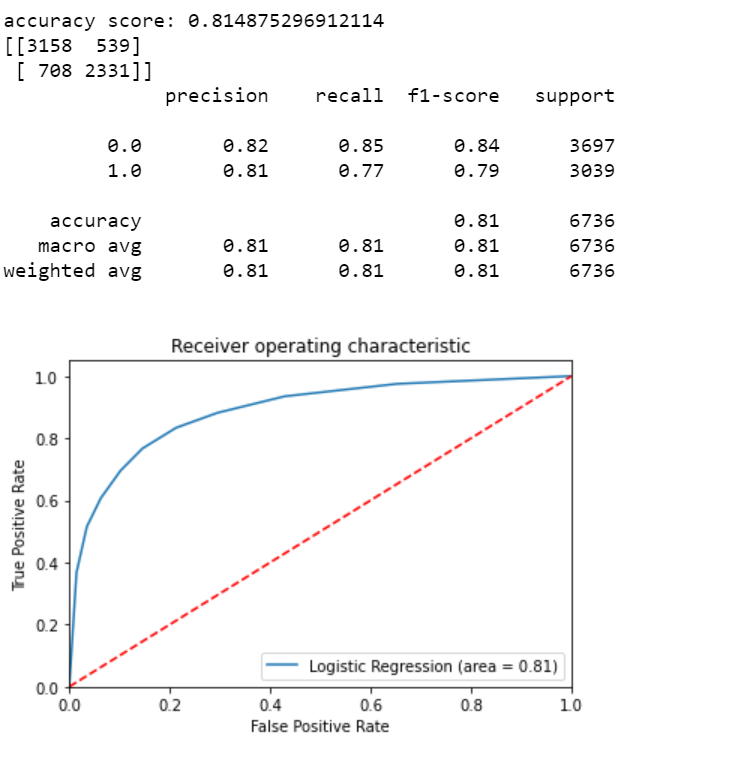
In order to apply KNN to our data, the most important part is determining the appropriate number of K and scaling the data. We used StandardScaler and KNeighborsClassifier methods and we determined 8 as our K.

Benchmark:



Testing our benchmark with KNN, we observed that the overall accuracy score was 0.7816with the correct classification of 5,265 and we had an AUC of 0.77. The scores for precision, recall, and f1-score were higher than 0.70. However, the recall score for ‘1’ was lower than 0.70.

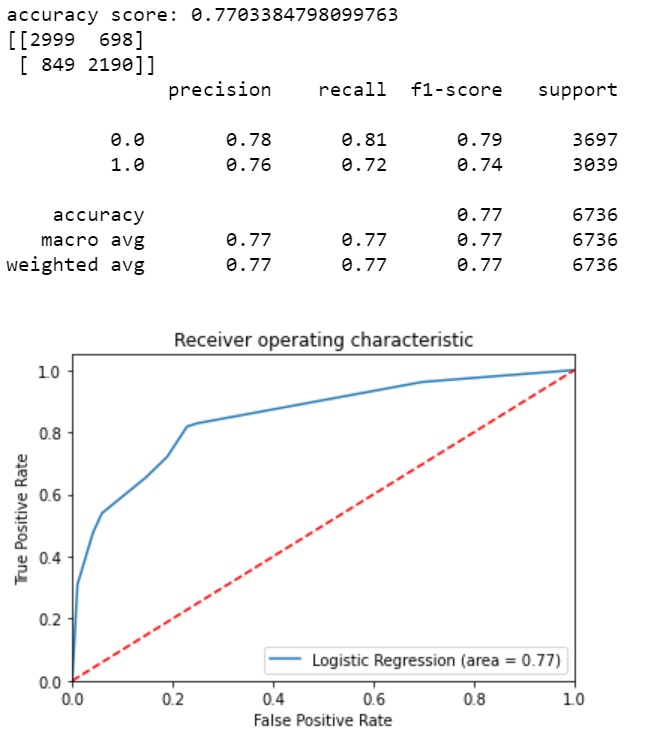
Sequential Backward Elimination:



Upon running KNN with sequential backward elimination, the accuracy score of this model was 0.815, with correct classification of 5,489 and an AUC score of 0.81. The precision, recall, and f1-score all have a score of over 0.70.

When comparing backward elimination with the benchmark, the accuracy improved by about 3%. All values for precision, recall, and f1-score increased as well, except for precision under ‘1’, which stayed the same.

Pearson Correlation:



Upon running KNN model with pearson correlation, our model had an accuracy of 0.770 and an AUC of 0.77. The precision, recall, and f1-score all have a score of over 0.70.

When comparing Pearson correlation with the benchmark, the accuracy decreased by a little less than 1%. However, backward elimination had a better accuracy score by about 4%.

Both pearson correlation and backward elimination were able to improve from the benchmark model. In the tables below, we compiled the model’s performance:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0(Low price) | Precision | Recall | f1-score | AUC |
| KNN (Raw data) | 0.77 | 0.87 | 0.81 | 0.77 |
| KNN (Backward Elimination) | 0.82 | 0.85 | 0.84 | 0.81 |
| KNN (Pearson Correlation) | 0.78 | 0.81 | 0.79 | 0.77 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1(High price) | Precision | Recall | f1-score | AUC |
| KNN (Raw data) | 0.81 | 0.68 | 0.74 | 0.77 |
| KNN (Backward Elimination) | 0.81 | 0.77 | 0.79 | 0.81 |
| KNN (Pearson Correlation) | 0.76 | 0.72 | 0.74 | 0.77 |

|  |  |
| --- | --- |
| **​KNN** | Accuracy |
| Train/Test Split Accuracy (Benchmark) | 0.782 |
| 10-fold Cross validation Accuracy (Benchmark) | 0.793 |
| Train/Test Split Accuracy (Backward Elimination) | 0.815 |
| 10-fold Cross validation Accuracy (Backward Elimination) | 0.820 |
| Train/Test Split Accuracy (Pearson Correlation) | 0.770 |
| 10-fold Cross validation Accuracy (Pearson Correlation) | 0.786 |

In the table above, we compared the accuracy between train-test split and 10-fold cross validation across the benchmark, sequential backward elimination, and pearson correlation methods. We can see that there was no significant difference between the train-test split and 10-fold cross validation methods. Therefore, since cross validation prevents overfitting, there was no overfitting present in the train-test split method across the models. In KNN, we observed that the 10-fold cross validation method gave slightly higher accuracies.

# VI. Conclusion

Across all the models, when we utilized sequential backward elimination and pearson correlation, we were able to beat the benchmark performance, except for KNN with pearson correlation, which had a less than 1% decrease in accuracy. Furthermore, by using both train-test split and 10-fold cross validation, we were able to determine there was no overfitting involved.

When comparing across the models, we found that the random forest model and logistic regression model showed better performance scores than naive bayes and knn. All of these models show that we have about a 78%-80% accuracy to predict whether a listing is priced as high or low in Los Angeles. The common attributes found in both backward elimination and pearson correlation are accommodates, room\_type, bathrooms, and bedrooms. These four attributes have a significant influence in determining if a listing price will be high or low. By utilizing these models and different feature selection techniques, Airbnb can discern which areas have these specific attributes that affect price, and market to properties in these regions to become hosts, therefore gaining a bigger profit from high priced listings. If we want to predict the price with higher accuracy, we need to conduct in-depth research on more data to see the relationship between them and generate a better prediction.

A limitation of this analysis is that we primarily focused on Los Angeles. However, if we were to analyze datasets from other regions around the world, we can have a better understanding if other variables have an effect on pricing.