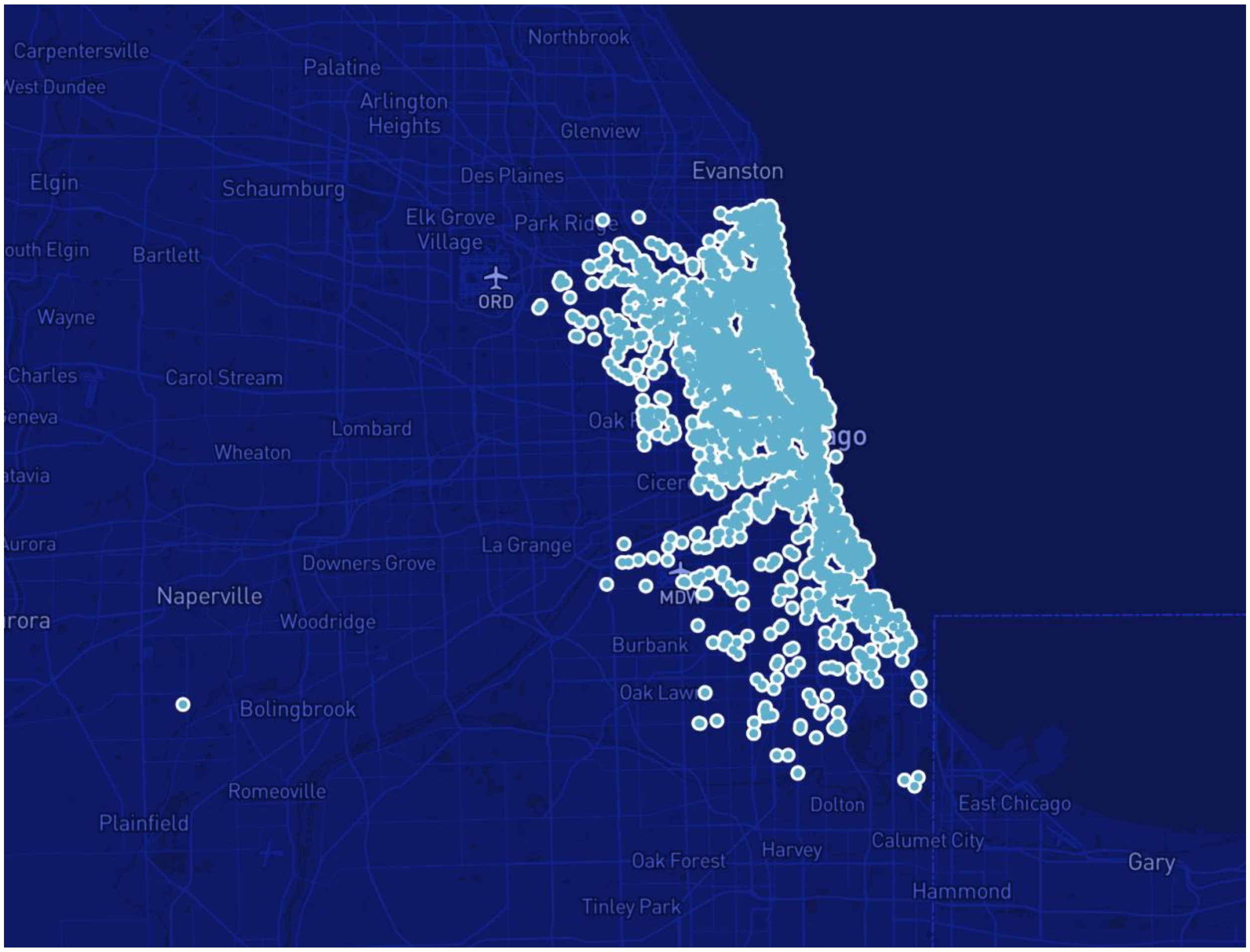
Machine Learning Analytics Group Project

Airbnb Chicago Price Prediction



`

Group 18

Youngjoo (Jennie) Ryu

Yaohui Wu

Xiaofan Zhu

Sahithi Muddana

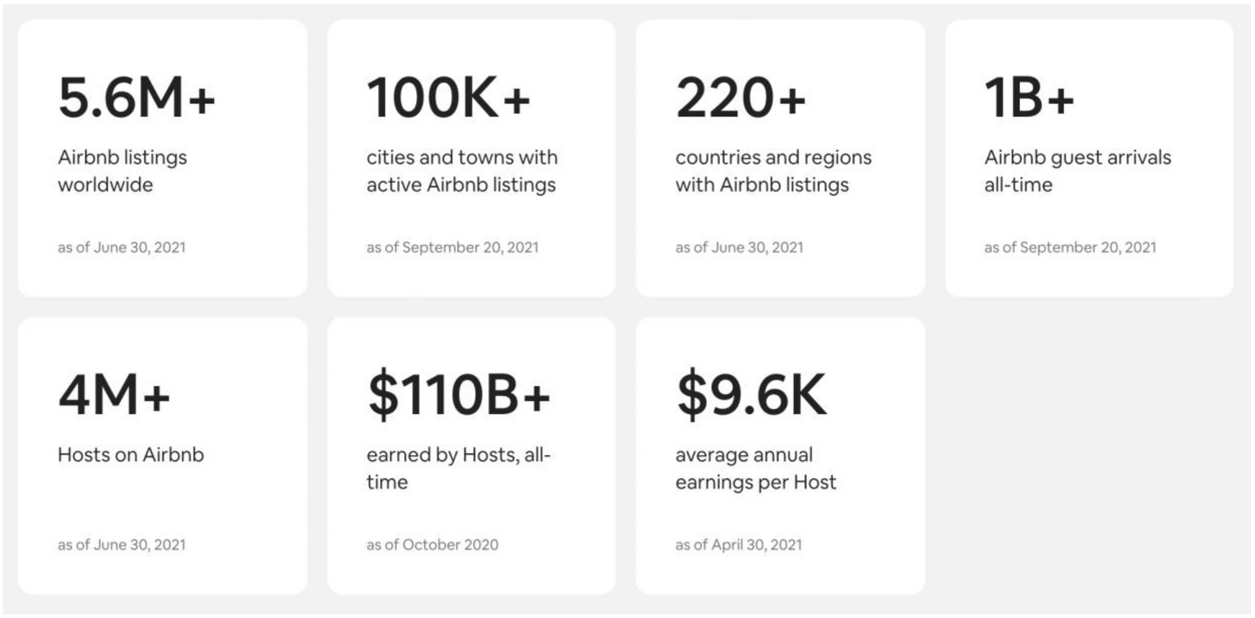
Jai Agrawal

Table of Contents

1. Introduction
2. Market Analysis
3. Data
    - a. Data overview
    - b. Description
    - c. Visualization
4. Data Analysis
    - 4.1 Before Preprocessing (Benchmarking)
        - a. Logistic Regression
        - b. Decision Tree
        - c. Random Forest
        - d. K-nearest Neighbors
    - 4.2 Preprocessing
    - 4.3 With Preprocessing
        - a. Logistic Regression
        - b. Decision Tree
        - c. Random Forest
        - d. K-nearest Neighbors
5. Key Takeaways
6. Conclusion

1. Introduction

Airbnb is an online marketplace that lets property owners rent out their spaces to travelers looking for a place to stay. Every day, Hosts offer unique stays and one-of-a-kind activities that make it possible for guests to experience the world in a more authentic, connected way.


5.6M+
Airbnb listings worldwide
as of June 30, 2021

100K+
cities and towns with active Airbnb listings
as of September 20, 2021

220+
countries and regions with Airbnb listings
as of June 30, 2021

1B+
Airbnb guest arrivals all-time
as of September 20, 2021

4M+
Hosts on Airbnb
as of June 30, 2021

$110B+
earned by Hosts, all-time
as of October 2020

$9.6K
average annual earnings per Host
as of April 30, 2021

Since its establishment in 2008, Airbnb has developed rapidly. Undoubtedly, its development has brought a great impact on the traditional hotel industry. However, as more competitive enterprises enter this industry, Airbnb must meet more challenges and the loss of market share. In addition, with the advent of the epidemic, Airbnb's global short rent business has been greatly impacted. According to data, the pandemic directly led to a huge loss for Airbnb. How will Airbnb develop in the future? To retain the highest market share, we propose that Airbnb should concentrate on building more user-friendly applications.

2. Market Analysis

Airbnb is currently the leader in the home-sharing market and since its efforts to globalize began in 2011, the company has entered the accommodation market in most countries around the world. As of 2019, North America drew in most of Airbnb's revenue worldwide with a sum of two billion U.S. dollars. Like the rest of the travel industry, Airbnb has been challenged by the pandemic, laying off about a quarter of its employees last May, but the home-sharing platform has fared better than most of its competitors thanks to the flexibility of its model. As travel demand shifted to nearby destinations outside of cities, Airbnb hosts were there to meet those travelers, giving the company a clear advantage over hotels. Now guests aren't just traveling on Airbnb, they are living on Airbnb. In the first quarter of 2021, 24% of nights booked were for stays of 28 nights or longer especially when numbers of people remote working increased. Remote work has become popular during the pandemic, and a hybrid model seems likely to persist even when offices reopen. Due to these post-covid phenomena, more users are seeking to use Airbnb in their daily life. As such demand increases, Airbnb users' priority consideration moved from considering whether it's a "tourist" place to whether they can stay for a long time in a certain city and whether the price is reasonable. For these users, Airbnb needs to classify accommodations in specific cities by price.

From the large amount of data Airbnb has accumulated for the past few years, we want to build a model that can recommend the best accommodation to Airbnb users looking to stay in Chicago. According to the data, Airbnb currently recommends accommodations that fit the user's desired location, number of days of stay, and various other amenities. However, unless the user adds a specific filter for the price range, only price filters users can apply to their options are low to high or the opposite. Using the data obtained from Airbnb, we would like to suggest a method of grouping and presenting the appropriate price into two segments (economy and luxury) based on the number of dates of stay, price range, and room type etc. desired by the user. With this method, users can understand the overall price range for the type of accommodation. Users looking for a long-term housing for a remote worker could easily browse through economy listings and users looking for a short-term housing for a fancy vacation could easily browse through luxury listings. In addition, hosts can also see if their property should be priced as a luxury property, or an economy property based off the property features and customer reviews.

3. Data

a. Data Overview

Our dataset is directly pulled from the Airbnb website. We have two datasets that we are using, “listings.csv” and “review.csv”. The dataset ‘listings’ contains 6,529 samples and ‘reviews’ contain 302,750 samples. There are 36 features that we are using from both datasets. The ‘listings.csv’ dataset contains information about the host accommodations (such as neighborhood, room type, number of bathrooms, etc.) and the ‘review.csv’ contains customer review information. Our classification variable is “Price” which we have divided based on the price point of 126 (which is the average value when outliers removed). Values above 126 we have labeled luxury, and values below 126 we have labeled Economy. We will look at the distribution of the classification variable later in the report.

b. Description

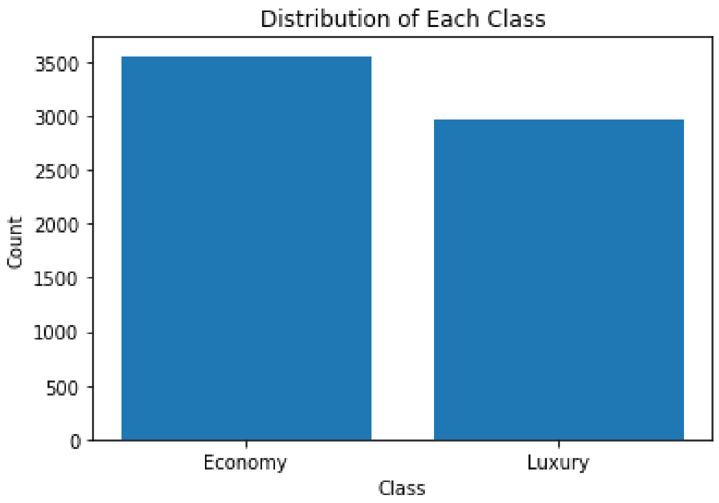
Below is the detailed description of the features from the datasets:

Host_acceptance_rate	Booking requests acceptance by the host of the listing.
Host_response_rate	The rate at which the host responds to customers
neighbourhood_cleansed	The neighborhood was geocoded using the latitude and longitude of the listing against neighborhoods as defined by open or public digital shapefiles
Latitude	Determines the latitude of the listing
Longitude	Determines the longitude of the listing
host_is_superhost	Checks if host is an experienced host i.e. superhost(0/1)
host_has_profile_pic	Checks if host has a profile picture(0/1)
host_identity_verified	Checks if host identity is verified(0/1)
Room_type	[Entire home/apt|Private room|Shared room|Hotel] All listings are grouped into the following three room types: - Entire place: This is best if someone is looking to have the whole space to themselves. - Private room: This provides customers with their own private room for sleeping and some shared space with others.

	- Shared room: The entire space is shared with others and will be sleeping in a shared space. This is budget friendly. - Hotel: These are unique boutique hotels
Hotel_room	Created dummy variable called Hotel room if room_type=hotel and checks the value as 0/1
Private_room	Created dummy variable called Private room if room_type=private and checks the value as 0/1
Shared_room	Created dummy variable named Shared room if room_type=shared and checks the value as 0/1
bathrooms_text	Description about the bathrooms of the listing
Private	Created dummy variable called Private based on bathroom_text
Shared	Created dummy variable called Shared based on bathroom_text
Num_bathrooms	The total number of bathrooms extracted from bathrooms_text
beds	Number of beds in the listing
bedrooms	Number of bedrooms in the listing
accommodates	The maximum capacity of the listing
minimum_nights	Minimum number of night stay
maximum_nights	Maximum number of night stay
number_of_reviews	Total number of reviews
review_scores_rating	Total rating of the listing
review_scores_accuracy	Rating based on accuracy of listing
review_scores_cleanliness	Rating based on cleanliness of listing
review_scores_checkin	Rating based on checking of listing
review_scores_ communication	Rating based on communication of listing
review_scores_location	Rating based on location of listing
review_scores_value	Rating based on value of listing
reviews_per_month	Total number of reviews in a month
listing_id	Numeric id of the listing in reviews data which is same as “id” in listings dataset.
comments	Review on specific listing
pos	Score based on positive sentiment words
neg	Score based on negative sentiment words
neu	Score based on neutral sentiment words
compound	Normalised sum of scores of pos,neg,neu

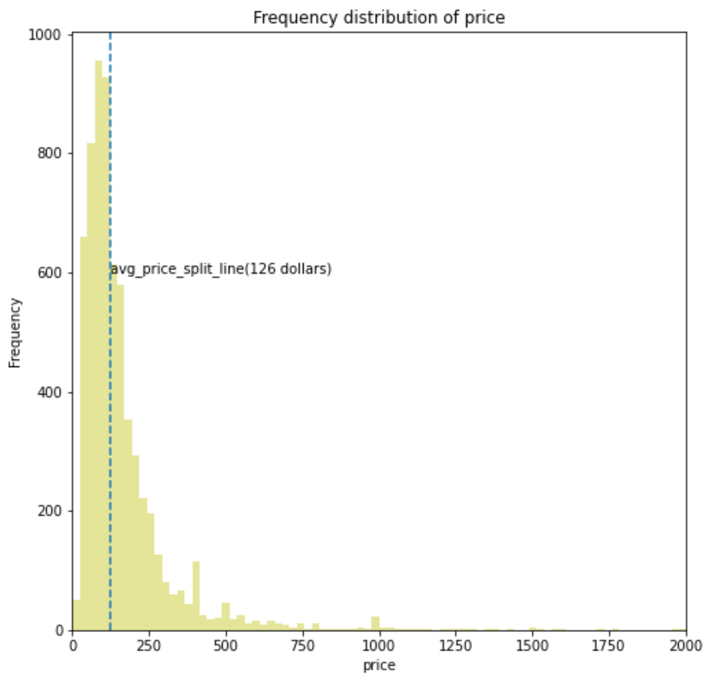
c. Visualization

As our task is predicting the price class, it’s necessary to visualize the distribution of each class. Below is a bar chart showing the distribution of each class:



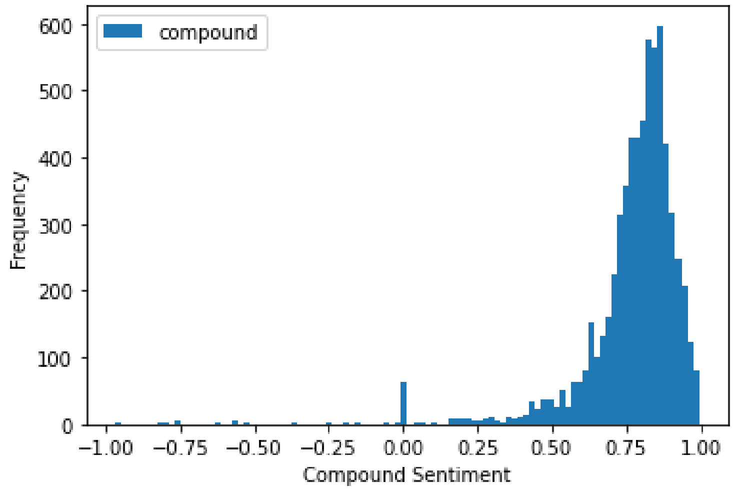
From the above graph we can learn that the proportion of these two classes is close to 1: 1. Therefore, it’s not necessary to use machine learning tools like SMOTE to balance the data.

We also wanted to take a look at how price is distributed in each class, to make sure that the range of each of the classes is not too different. Below is a frequency distribution of price with the dotted line representing the divide between economy and luxury:

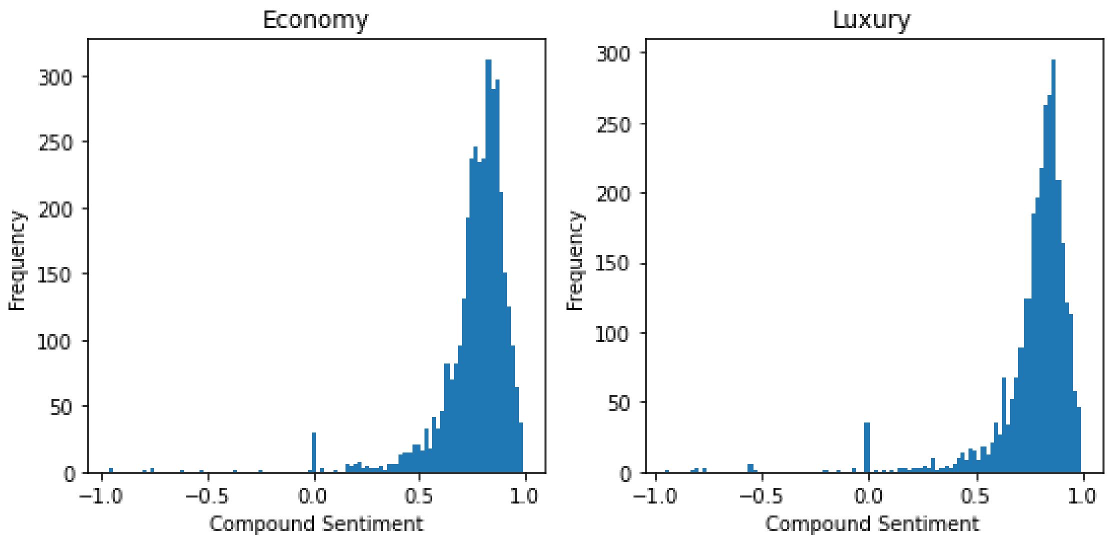


We can conclude that the majority of price in Economy is close to 100 while most of the price in Luxury is between 126 and 250 dollars.

Since we also add sentiment scores into our model as additional features, it's also essential that we visualization the distribution of the sentiment score of the entire dataset and each group separately:

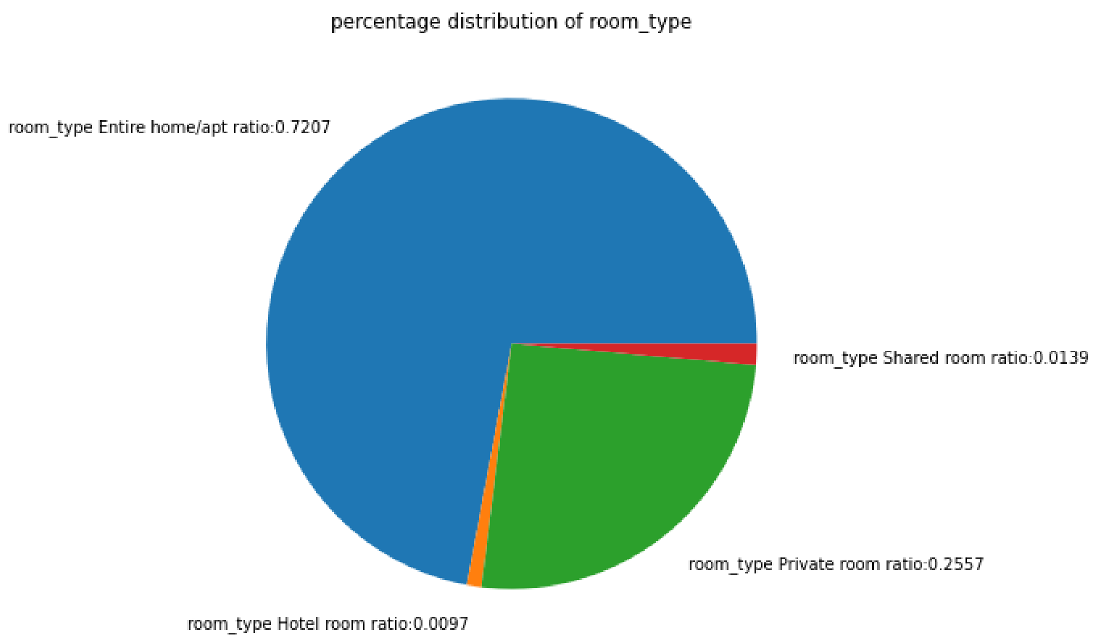


Here, we only visualize the compound score as this sentiment score could reflect sentimental emotion of the comments (The more positive compound is the more positive the text is and the more negative compound is the more negative the text is). We can see that most of our compound scores are distributed above 0.5. Below is distribution of the compound scores of each class:



From the compound score graph of each class we can conclude that Luxury is slightly better than Economy.

In our data, we converted a few features from categorical variables into dummy variables. Below is an example of a categorical variable, room_type, which we split into dummy variables that our model could process:

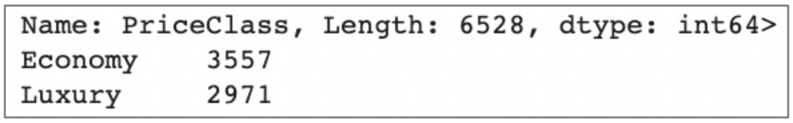


4. Analysis

Since our task was binary classification (we turned price into our classification variable: Economy and Luxury), we decided to use the following models for classification:

a. Logistic Regression:
    - Logit (statsmodels module)
    - Logistic Regression (sklearn module)
b. Decision Tree
c. Random Forest
d. K-nearest Neighbors

Before we did any real pre-processing, we decided to bench mark the above models to see the accuracy before and after we made adjustments. We split the model 30% test (validation) and 70% training (for both before pre-processing and after to keep our methods consistent). For validation we used accuracy score, confusion matrix, and AUC. We felt that all methods were appropriate because our classification data was almost evenly distributed with 3557 elements classified as Economy and 2971 elements classified as Luxury (after test train split we had 1,106 elements in Economy and 853 elements in Luxury). To keep our accuracy scores consistent we used the Sklearn package (the exception to this is logit which we did not use in our accuracy considerations).



```
Name: PriceClass, Length: 6528, dtype: int64>
Economy    3557
Luxury     2971
```

4.1 Before Preprocessing (Benchmarking)

Although this initial benchmarking is for our accuracy scores before pre-processing, we still needed to do some pre-processing in order for data to be accepted in the various different models. This mainly consisted of replacing null values (which we replaced with the column averages in this benchmark) and creating dummy variables for categorical variables. An example of this is the feature 'bathrooms_text'. For this feature we first had to separate the numeric value from the text, fill any null values with the averages of the numeric values, and then convert the text (which were categorical data) into dummy variables.

To keep the integrity of our benchmark we tried to do as little pre-processing as possible in this stage to see the efficiency and accuracy of our models (this means that some of the features in the initial benchmarking stage were not used as they would have required extensive pre-processing to be included in the model). The following segment of the report will go over a brief explanation of the model and our initial benchmark without preprocessing.

a. Logistic Regression

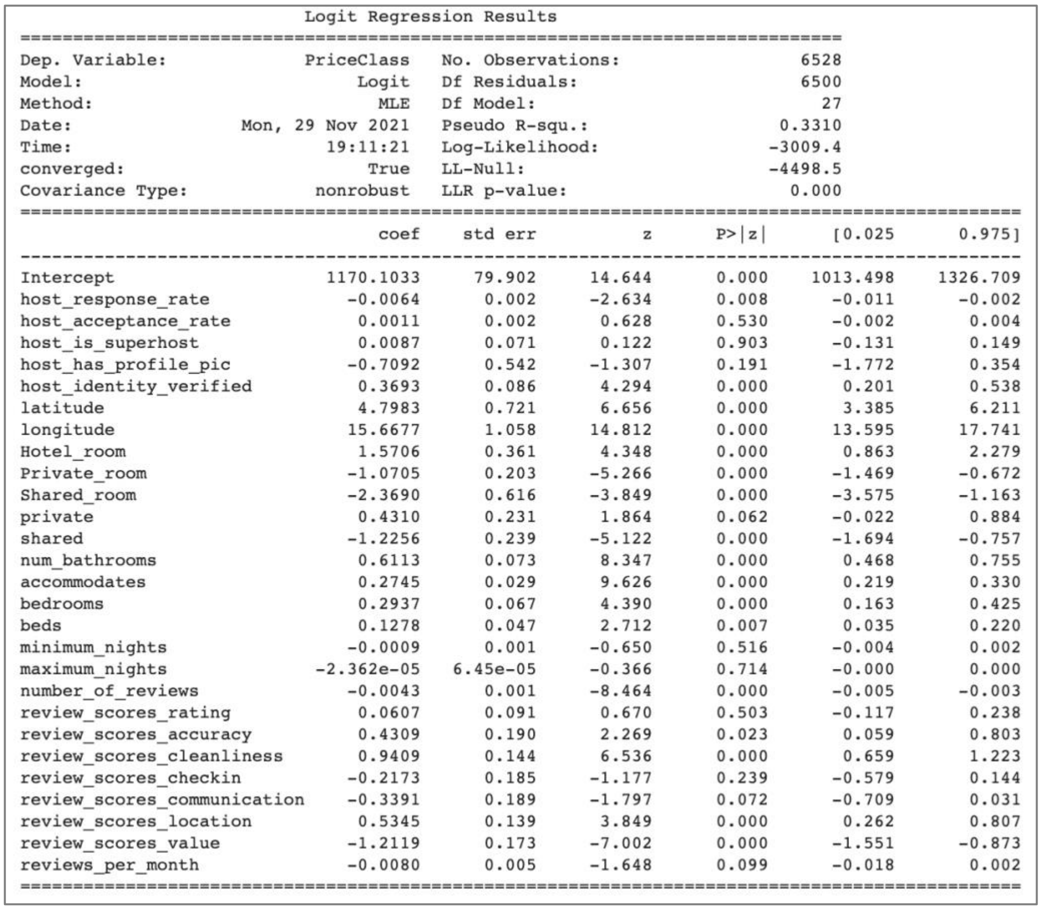
The first approach we considered was logistic regression. Logistic regression is a great classifier to begin with because it helps us easily identify which predictors have significant impact on whether a property is Economy or Luxury. The method is based on the logistic function which is as follows:

$$g(z) = \frac{1}{1 + e^{-z}}$$

The logistic function takes any/all real numbers and distinguishes them as a value between 0 and 1 (but never exactly 0 or 1). Logistic regression builds on this function to predict/model the probabilities of the response class (when the classification is binary interpreting the class values as 0 and 1). The way that this is done is simply by inputting feature values into the logistic regression equation and retrieving the result value as a probability. Then to classify, based of the threshold 0.5, we can say if the value is less than 0.5 it is in class 0 and if its greater than 0.5 it is in class 1. For our purposes, we used two packages to perform the classification. The first, Statsmodel's logit, was to identify the key predictor values and get an overall sense of our model's strength. The second. Sklearn's logistic regression, was to record the accuracy the model and to keep the package's across our model the same.

- **Logit(Statsmodel)**

  After running the logit function using our non-processed data we get the following result:



```
                           Logit Regression Results
==============================================================================
Dep. Variable:             PriceClass   No. Observations:                 6528
Model:                          Logit   Df Residuals:                     6500
Method:                           MLE   Df Model:                           27
Date:                Mon, 29 Nov 2021   Pseudo R-squ.:                  0.3310
Time:                        19:11:21   Log-Likelihood:                -3009.4
converged:                       True   LL-Null:                       -4498.5
Covariance Type:            nonrobust   LLR p-value:                     0.000
===============================================================================================
                                  coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------------------
Intercept                    1170.1033     79.902     14.644      0.000    1013.498    1326.709
host_response_rate             -0.0064      0.002     -2.634      0.008      -0.011      -0.002
host_acceptance_rate            0.0011      0.002      0.628      0.530      -0.002       0.004
host_is_superhost               0.0087      0.071      0.122      0.903      -0.131       0.149
host_has_profile_pic           -0.7092      0.542     -1.307      0.191      -1.772       0.354
host_identity_verified          0.3693      0.086      4.294      0.000       0.201       0.538
latitude                        4.7983      0.721      6.656      0.000       3.385       6.211
longitude                      15.6677      1.058     14.812      0.000      13.595      17.741
Hotel_room                      1.5706      0.361      4.348      0.000       0.863       2.279
Private_room                   -1.0705      0.203     -5.266      0.000      -1.469      -0.672
Shared_room                    -2.3690      0.616     -3.849      0.000      -3.575      -1.163
private                         0.4310      0.231      1.864      0.062      -0.022       0.884
shared                         -1.2256      0.239     -5.122      0.000      -1.694      -0.757
num_bathrooms                   0.6113      0.073      8.347      0.000       0.468       0.755
accommodates                    0.2745      0.029      9.626      0.000       0.219       0.330
bedrooms                        0.2937      0.067      4.390      0.000       0.163       0.425
beds                            0.1278      0.047      2.712      0.007       0.035       0.220
minimum_nights                 -0.0009      0.001     -0.650      0.516      -0.004       0.002
maximum_nights              -2.362e-05   6.45e-05     -0.366      0.714      -0.000       0.000
number_of_reviews              -0.0043      0.001     -8.464      0.000      -0.005      -0.003
review_scores_rating            0.0607      0.091      0.670      0.503      -0.117       0.238
review_scores_accuracy          0.4309      0.190      2.269      0.023       0.059       0.803
review_scores_cleanliness       0.9409      0.144      6.536      0.000       0.659       1.223
review_scores_checkin          -0.2173      0.185     -1.177      0.239      -0.579       0.144
review_scores_communication    -0.3391      0.189     -1.797      0.072      -0.709       0.031
review_scores_location          0.5345      0.139      3.849      0.000       0.262       0.807
review_scores_value            -1.2119      0.173     -7.002      0.000      -1.551      -0.873
reviews_per_month              -0.0080      0.005     -1.648      0.099      -0.018       0.002
===============================================================================================
```

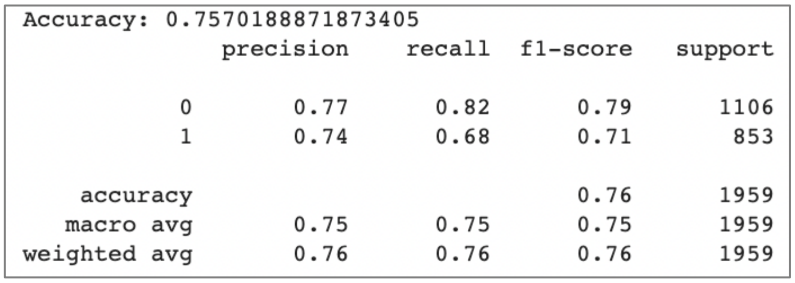
From the results, we can see that we have several strong predictors:

- host_response_rate
- host_identity_verified
- latitude/longitude
- Hotel_room/Private_room/Shared_room
- num_bathrooms
- accommodates
- bedrooms/beds
- number_of_reviews
- review_scores_cleanliness
- review_scores_location
- review_scores_value

One thing to be wary of is our R-squared value (0.33) which is relatively low. Although this is not necessarily an indicator of a strong or poor model, it is important to keep in mind when assessing the strength of it. When we remove the insignificant values, our R-squared drops (as there is information loss) but not significantly indicating that the predictors listed above are our strongest. We are not looking at any accuracy score's with this model as we want to use the same package when comparing accuracy. We will take a look at the accuracy in the following model.

- Logistic Regression(Sklearn)

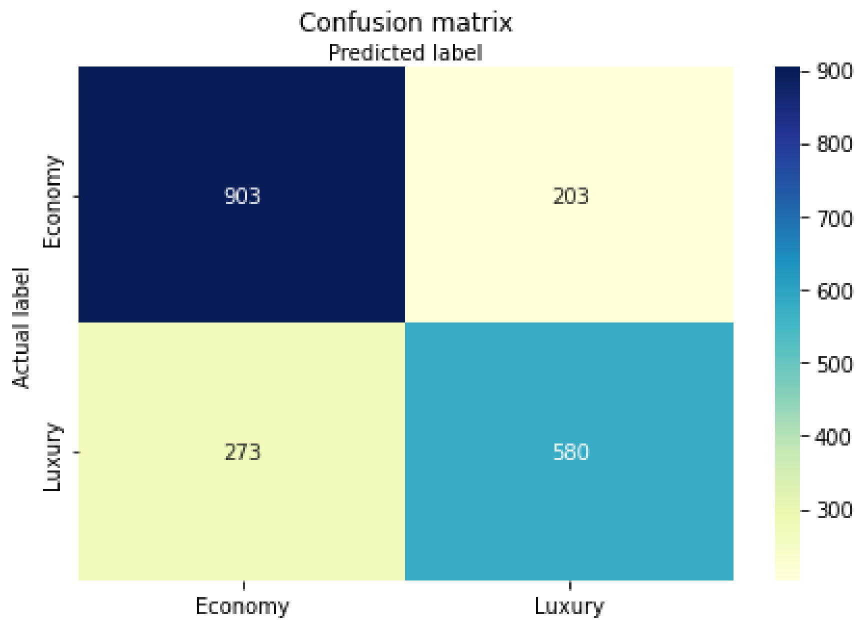
After running the Logistic Regression function using our non-processed data we get the following result:

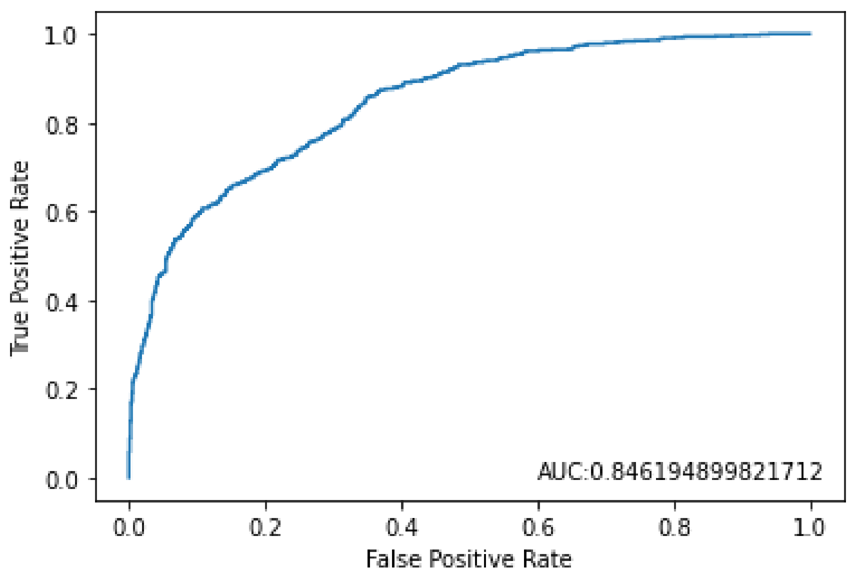


```
Accuracy: 0.7570188871873405
              precision    recall  f1-score   support

           0       0.77      0.82      0.79      1106
           1       0.74      0.68      0.71       853

    accuracy                           0.76      1959
   macro avg       0.75      0.75      0.75      1959
weighted avg       0.76      0.76      0.76      1959
```



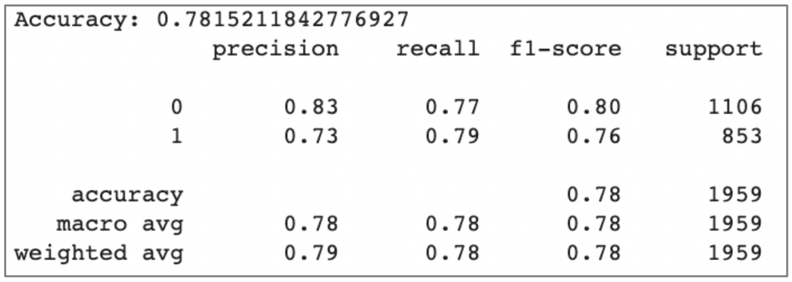


We can see from the results that when no preprocessing is done (or very minimal pre-processing), that the accuracy is approximately 0.76. Our classifier correctly identified 903 elements as Economy out of 1106 possible elements (203 incorrectly classified) and classified 580 elements Luxury out of 853 possible elements (273 incorrectly classified). The AUC is also 0.84, which is in between 0.5 and 1, and indicates that our model can distinguish the two class models pretty well. We can also see that our precision for both 0 and 1 is above .7 and likewise with our f1-score. So without pre-processing our model is already looking pretty good. In our “with pre-processing” section we will try to see if we can improve this accuracy.

b. Decision Tree

The next method we will explore is decision tree. Decision tree utilizes several algorithms to separate data into groups based on certain rules created utilizing predictors. The way the decision tree algorithm works is that it evaluates attributes based on information gain (high information gain is good for a feature) or entropy (low entropy is good for a feature) and based on this evaluation a tree is generated, where the top node is the node (with either the highest information gain or the lowest entropy) and nodes following either have a split or are terminal nodes. Nodes that split mean there is usually additional decisions needing to be made and nodes that are terminal can tell us whether or not the element is in our classification set or not.

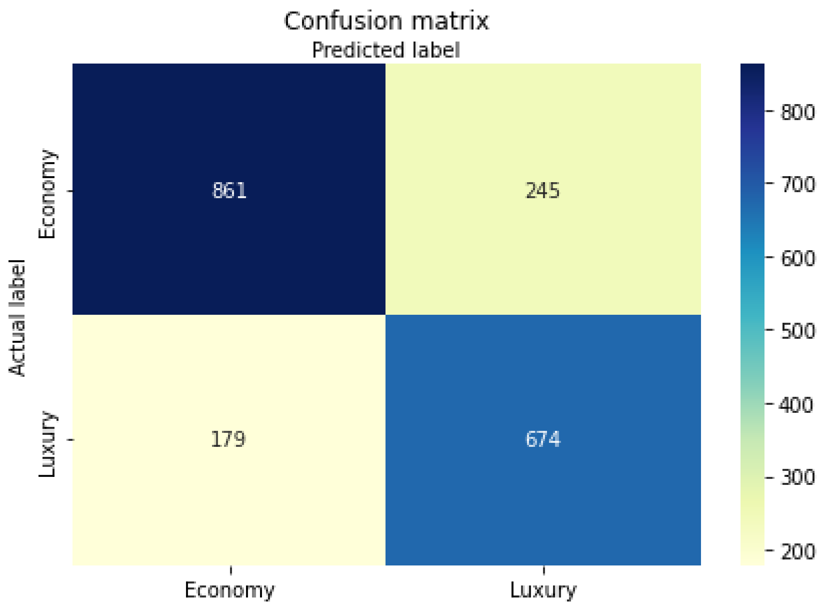
With that in mind, we can take a look at our results from our initial benchmarking:

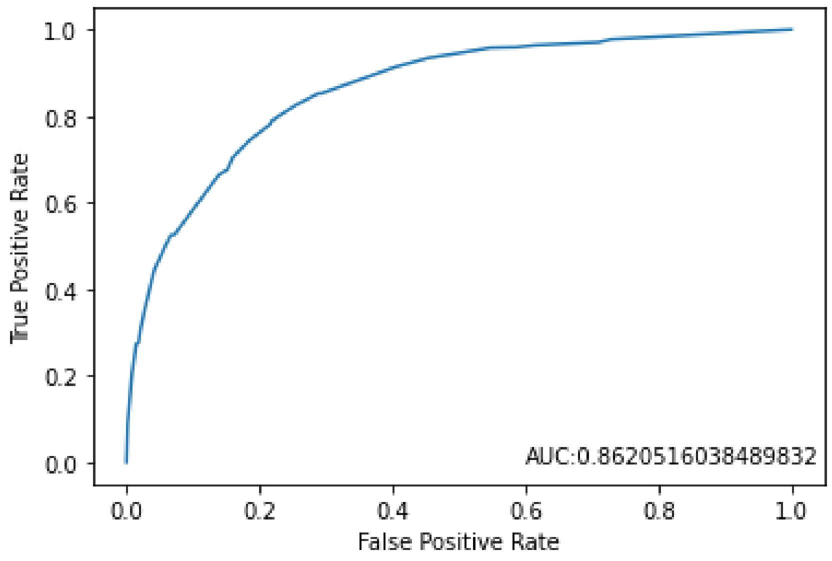


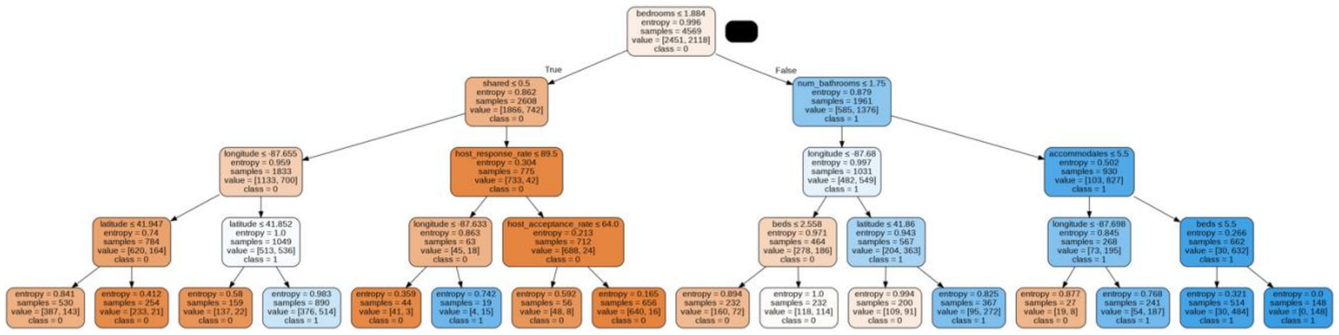
```
Accuracy: 0.7815211842776927
              precision    recall  f1-score   support

           0       0.83      0.77      0.80      1106
           1       0.73      0.79      0.76       853

    accuracy                           0.78      1959
   macro avg       0.78      0.78      0.78      1959
weighted avg       0.79      0.78      0.78      1959
```





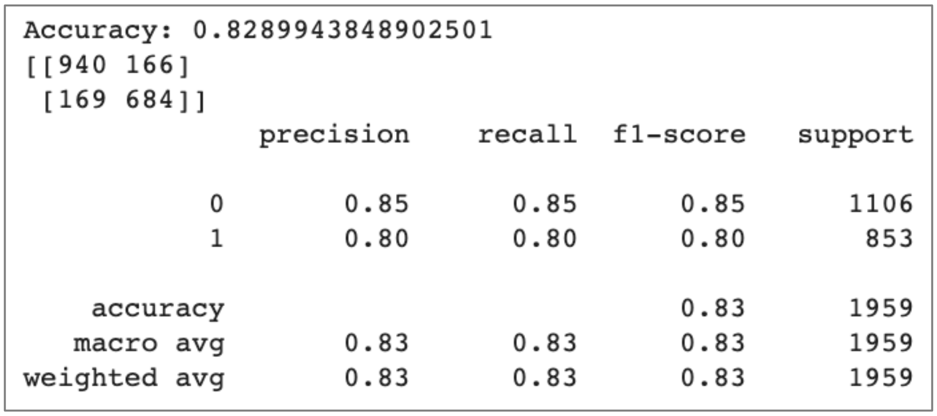

bedrooms ≤ 1.884 entropy = 0.996 samples = 4569 value = [2451, 2118] class = 0
True
False
shared ≤ 0.5 entropy = 0.862 samples = 2608 value = [1866, 742] class = 0
num_bathrooms ≤ 1.75 entropy = 0.879 samples = 1961 value = [585, 1376] class = 1
longitude ≤ -87.655 entropy = 0.959 samples = 1833 value = [1133, 700] class = 0
host_response_rate ≤ 89.5 entropy = 0.304 samples = 775 value = [733, 42] class = 0
longitude ≤ -87.68 entropy = 0.997 samples = 1031 value = [482, 549] class = 1
accommodates ≤ 5.5 entropy = 0.502 samples = 930 value = [103, 827] class = 1
latitude ≤ 41.947 entropy = 0.74 samples = 784 value = [620, 164] class = 0
latitude ≤ 41.852 entropy = 1.0 samples = 1049 value = [513, 536] class = 1
longitude ≤ -87.633 entropy = 0.863 samples = 63 value = [45, 18] class = 0
host_acceptance_rate ≤ 64.0 entropy = 0.213 samples = 712 value = [688, 24] class = 0
beds ≤ 2.558 entropy = 0.971 samples = 464 value = [278, 186] class = 0
latitude ≤ 41.86 entropy = 0.943 samples = 567 value = [204, 363] class = 1
longitude ≤ -87.698 entropy = 0.845 samples = 268 value = [73, 195] class = 1
beds ≤ 5.5 entropy = 0.266 samples = 662 value = [30, 632] class = 1
entropy = 0.841 samples = 530 value = [387, 143] class = 0
entropy = 0.412 samples = 254 value = [233, 21] class = 0
entropy = 0.58 samples = 159 value = [137, 22] class = 0
entropy = 0.983 samples = 890 value = [376, 514] class = 1
entropy = 0.359 samples = 44 value = [41, 3] class = 0
entropy = 0.742 samples = 19 value = [4, 15] class = 1
entropy = 0.592 samples = 56 value = [48, 8] class = 0
entropy = 0.165 samples = 656 value = [640, 16] class = 0
entropy = 0.894 samples = 232 value = [160, 72] class = 0
entropy = 1.0 samples = 232 value = [118, 114] class = 0
entropy = 0.994 samples = 200 value = [109, 91] class = 0
entropy = 0.825 samples = 367 value = [95, 272] class = 1
entropy = 0.877 samples = 27 value = [19, 8] class = 0
entropy = 0.768 samples = 241 value = [54, 187] class = 1
entropy = 0.321 samples = 514 value = [30, 484] class = 1
entropy = 0.0 samples = 148 value = [0, 148] class = 1

From our results we can see that our model is fairly strong already. The accuracy is approximately 0.78 (which is slightly better than our logistic regression classifier) and our precision and f1-score for both classes are pretty high. We can see from the confusion matrix that for the Economy class 861 observations were classified correctly (with 245 misses) and 674 Luxury class observations were classified correctly (with 179 misses). Our AUC was higher than 0.8 suggesting that the model is capable of distinguishing the classes as well. Overall, for an initial benchmark the model is fairly decent.

c. Random Forest

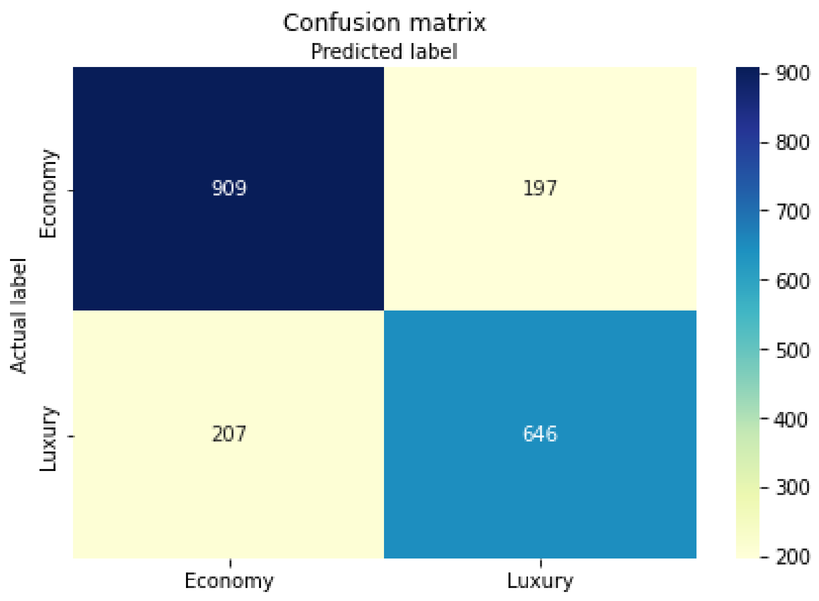
The next model we considered was Random Forest. Random Forest, put very basically, uses a combination of bagging and feature randomness to create a series of decision trees that are then merged together to get the most stable and accurate predictions. So, in most cases, random forest will perform better than decision tree and we expect to see that in our results.

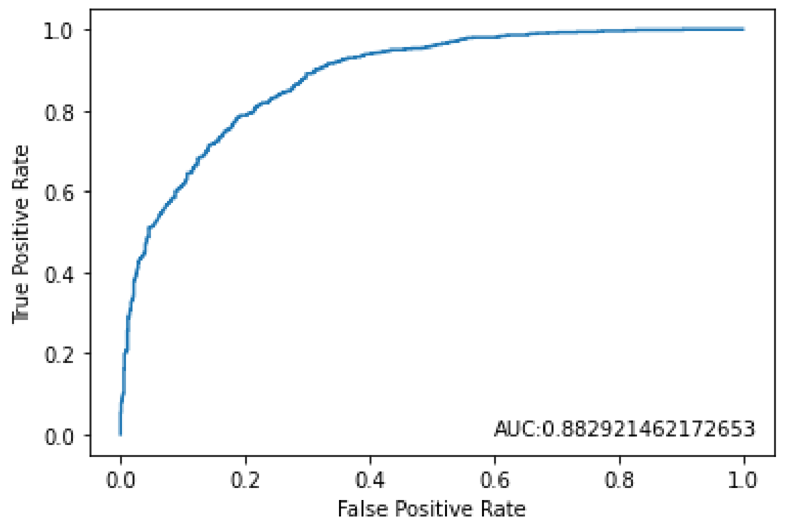
After performing the random forest classifier on our initial data we received these results:


```
Accuracy:  0.8289943848902501
[[940 166]
 [169 684]]
              precision    recall  f1-score   support

           0       0.85      0.85      0.85      1106
           1       0.80      0.80      0.80       853

    accuracy                           0.83      1959
   macro avg       0.83      0.83      0.83      1959
weighted avg       0.83      0.83      0.83      1959
```



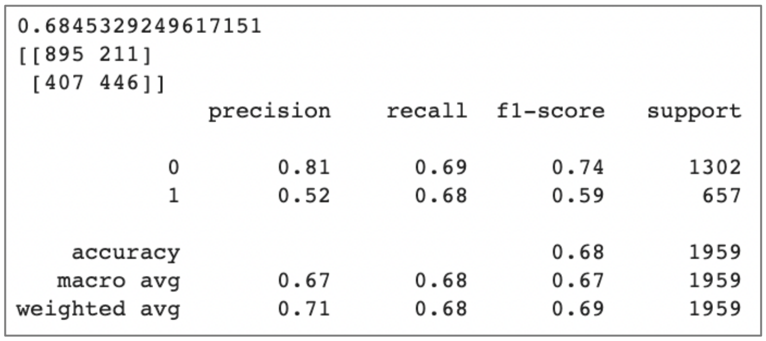


As expected, our random forest accuracy exceeds that of all the other models. The accuracy we received from initial benchmark was approximately 0.79. Looking at the classification report we can also see that there is high precision and f1-score for both classes. The model had 909 hits for the Economy class (197 misses) and 646 hits for the Luxury class (207 misses). The AUC score, was also the highest so far at approximately 0.88. This is by far our strongest model before pre-processing.

d. K-nearest Neighbors

The final model we considered was KNN. Put simply, the way KNN works is by grouping elements based on their similarity and classifying them. To elaborate further, KNN takes distances between an input and all the example data points in the data and then classifies it as the most frequent label present in these data points. An important point of consideration when running this model is the value of K (the number of the nearest neighbors that we would use to classify data points).

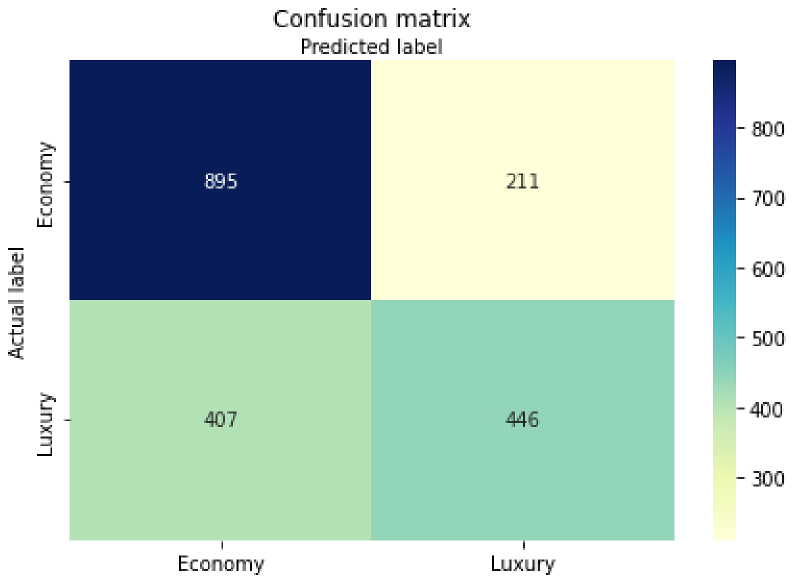
After running KNN on our data we received these results:

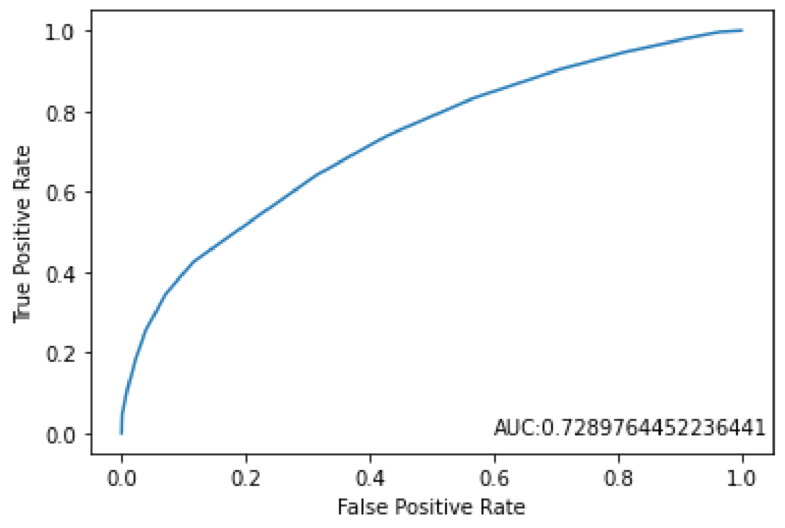


```
0.6845329249617151
[[895 211]
 [407 446]]
              precision    recall  f1-score   support

           0       0.81      0.69      0.74      1302
           1       0.52      0.68      0.59       657

    accuracy                           0.68      1959
   macro avg       0.67      0.68      0.67      1959
weighted avg       0.71      0.68      0.69      1959
```





Our KNN model, with no pre-processing, performed the worst out of all the models. The accuracy of our model is 0.68 (lower than the 0.7 mark). When we take a look at the confusion matrix we can see that the model had 887 hits in the Economy class (219 misses) and 448 hits in the Luxury class (405 misses). We can see from this (also from the precision and f1-score) that KNN is much weaker at classifying the Luxury class than our other models. The AUC is still above 0.7 meaning that, the model is capable of distinguishing the two classes from each other. We will try to improve this score, our "with pre-processing" section.

4.2 Preprocessing

Since our dataset had a relatively equal distribution of the class variable we decided that oversampling and under sampling (techniques like SMOTE) were not required. Before we went into pre-processing for specific models, we did some overall pre-processing across our dataset. There are two major pre-processing steps we took that impact all the models equally: text processing and changing the way we filled null values.

1. Text processing:

   In our dataset there were several columns that were text, as well as a companion dataset that had review text. So we decided that it would be interesting to consider sentiment analysis as potential feature to include in our models. In order to incorporate the text as sentiment score we would have to run it through separate model. To start with however we would have to clean the text. So first, using the text data from the review dataset (“listing_id”, “id”, “date”, “review_id”, “comments” etc.), we used a python function to retain only English (there were other languages present in the dataset). Then we used an apply method to transfer all letters in lowercase. Finally, we split the sentences into separate words and used the nltk package to remove “stopwords” (which are a set of words that have no meaning). With the text cleaned, we used a sentiment intensity analyzer to calculate the sentiment score of each review. This came as set of four scores: “pos”, “neg”, “neu” and “compound”. Compound is a combination of the former three scores (and ranges from -1 to 1). The more positive compound is the more positive the text is and the more negative compound is the more negative the text is. Some of the listings had multiple reviews so we took an average of this score to represent the overall sentiment of the reviews. After all these steps were completed we attached them to our main dataset.

2. Filling null/na values:

   There are several ways to fill null values. There are methods like filling null values with the mean, the last non-null value, the mode, etc. We decided that we would fill null values with the average value of the column grouped by the “neighborhood_cleansed” feature instead of just broadly filling null values with averages (which could vary greatly from the actual). This allowed us to create a lot more accurate averages for some the column values for each listing. Neighborhoods that just contained null values (and therefore provided no averages) we removed from the dataset as they were a very small portion of it.

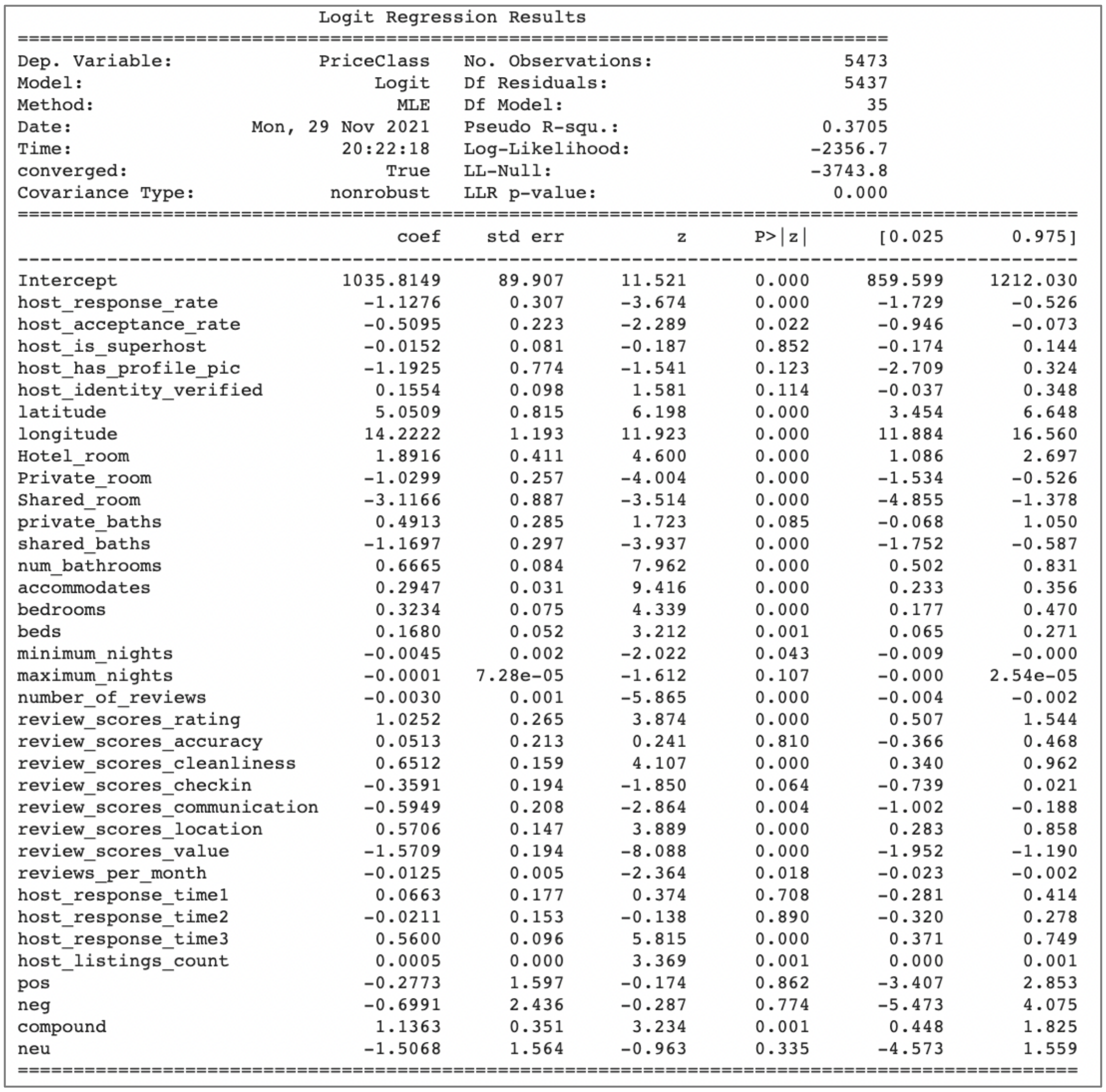
4.3 With Preprocessing

In this section we will go over all the results from the models after the previous pre-processing steps were completed. We will also go over the various optimization steps we took (such as adjusting feature selection or changing hyperparameters) for each model:

a. Logistic Regression

- **Logit(Statsmodel)**

For the logit stats model we did not do any specific pre-processing since we aren't using the accuracy as a reference. Instead we are checking to see if any of the predictors became more or less significant after our overall preprocessing: Below you fill find the results of the logit function:



```
                           Logit Regression Results
==============================================================================
Dep. Variable:             PriceClass   No. Observations:                 5473
Model:                          Logit   Df Residuals:                     5437
Method:                           MLE   Df Model:                           35
Date:                Mon, 29 Nov 2021   Pseudo R-squ.:                  0.3705
Time:                        20:22:18   Log-Likelihood:                -2356.7
converged:                       True   LL-Null:                       -3743.8
Covariance Type:            nonrobust   LLR p-value:                     0.000
===============================================================================================
                                  coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------------------
Intercept                    1035.8149     89.907     11.521      0.000     859.599    1212.030
host_response_rate             -1.1276      0.307     -3.674      0.000      -1.729      -0.526
host_acceptance_rate           -0.5095      0.223     -2.289      0.022      -0.946      -0.073
host_is_superhost              -0.0152      0.081     -0.187      0.852      -0.174       0.144
host_has_profile_pic           -1.1925      0.774     -1.541      0.123      -2.709       0.324
host_identity_verified          0.1554      0.098      1.581      0.114      -0.037       0.348
latitude                        5.0509      0.815      6.198      0.000       3.454       6.648
longitude                      14.2222      1.193     11.923      0.000      11.884      16.560
Hotel_room                      1.8916      0.411      4.600      0.000       1.086       2.697
Private_room                   -1.0299      0.257     -4.004      0.000      -1.534      -0.526
Shared_room                    -3.1166      0.887     -3.514      0.000      -4.855      -1.378
private_baths                   0.4913      0.285      1.723      0.085      -0.068       1.050
shared_baths                   -1.1697      0.297     -3.937      0.000      -1.752      -0.587
num_bathrooms                   0.6665      0.084      7.962      0.000       0.502       0.831
accommodates                    0.2947      0.031      9.416      0.000       0.233       0.356
bedrooms                        0.3234      0.075      4.339      0.000       0.177       0.470
beds                            0.1680      0.052      3.212      0.001       0.065       0.271
minimum_nights                 -0.0045      0.002     -2.022      0.043      -0.009      -0.000
maximum_nights                 -0.0001   7.28e-05     -1.612      0.107      -0.000    2.54e-05
number_of_reviews              -0.0030      0.001     -5.865      0.000      -0.004      -0.002
review_scores_rating            1.0252      0.265      3.874      0.000       0.507       1.544
review_scores_accuracy          0.0513      0.213      0.241      0.810      -0.366       0.468
review_scores_cleanliness       0.6512      0.159      4.107      0.000       0.340       0.962
review_scores_checkin          -0.3591      0.194     -1.850      0.064      -0.739       0.021
review_scores_communication    -0.5949      0.208     -2.864      0.004      -1.002      -0.188
review_scores_location          0.5706      0.147      3.889      0.000       0.283       0.858
review_scores_value            -1.5709      0.194     -8.088      0.000      -1.952      -1.190
reviews_per_month              -0.0125      0.005     -2.364      0.018      -0.023      -0.002
host_response_time1             0.0663      0.177      0.374      0.708      -0.281       0.414
host_response_time2            -0.0211      0.153     -0.138      0.890      -0.320       0.278
host_response_time3             0.5600      0.096      5.815      0.000       0.371       0.749
host_listings_count             0.0005      0.000      3.369      0.001       0.000       0.001
pos                            -0.2773      1.597     -0.174      0.862      -3.407       2.853
neg                            -0.6991      2.436     -0.287      0.774      -5.473       4.075
compound                        1.1363      0.351      3.234      0.001       0.448       1.825
neu                            -1.5068      1.564     -0.963      0.335      -4.573       1.559
===============================================================================================
```

From the results, we can see that the pre-processing didn't really change which predictors were more significant. However, with the pre-processing described above, we gained an additional feature that was significant: compound. The most significant predictors were:

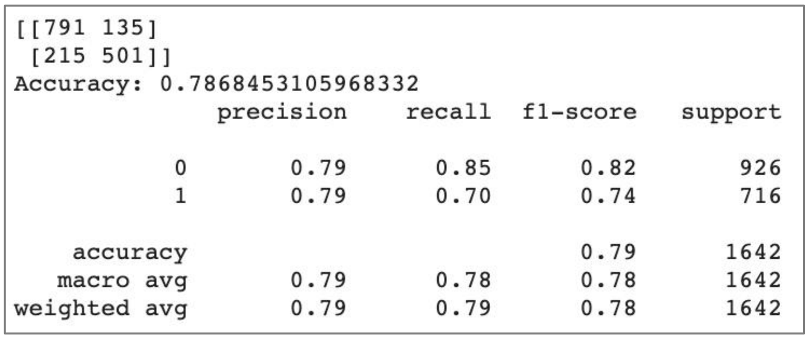
- host_response_rate
- host_identity_verified
- latitude/longitude
- Hotel_room/Private_room/Shared_room
- num_bathrooms
- accommodates
- bedrooms/beds
- number_of_reviews
- review_scores_cleanliness
- review_scores_location
- review_scores_value
- compound.

Our R-squared value (0.37) remained relatively low which is something to keep in mind in our accuracy evaluations later. Once again, we are not looking at any accuracy score here, we use the Sklearn package to check accuracy.

- Logistic Regression(Sklearn)

  For our Logistic Regression classifier we made some specific changes when pre-processing just for this model. First, we tried removing the least significant features completely to see if the accuracy changed. With that method we saw a small decrease in accuracy so, we then tried forward variable selection to get the best combination of features.

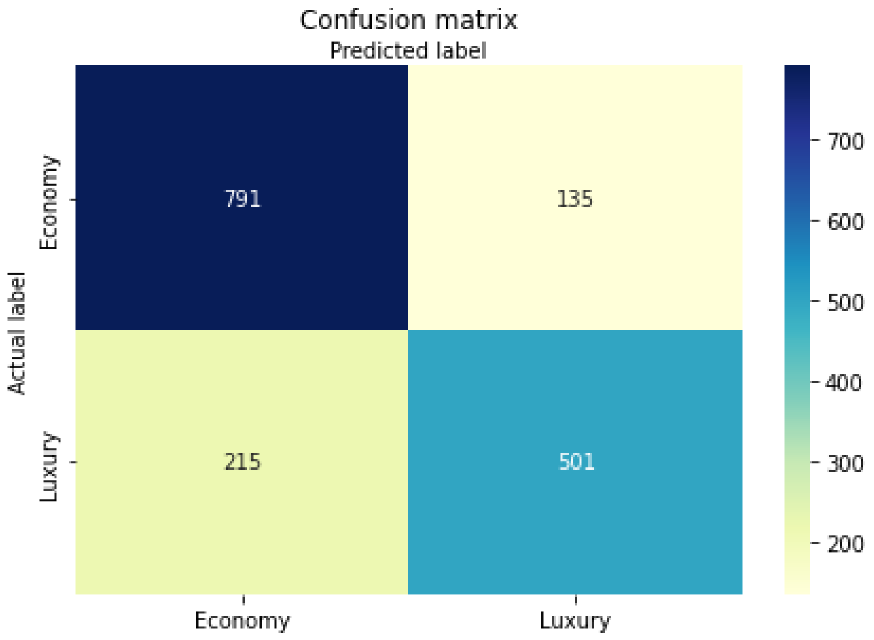
  Below is the results of the best run:

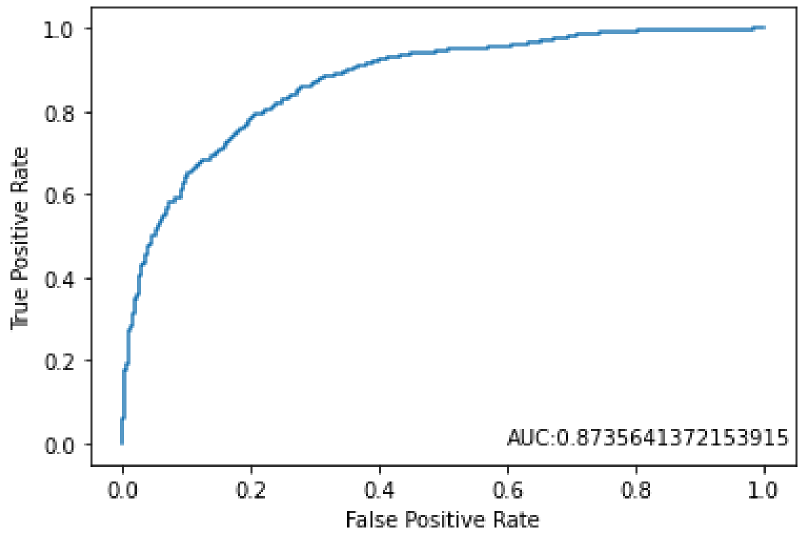


```
[[791 135]
 [215 501]]
Accuracy: 0.7868453105968332
              precision    recall  f1-score   support

           0       0.79      0.85      0.82       926
           1       0.79      0.70      0.74       716

    accuracy                           0.79      1642
   macro avg       0.79      0.78      0.78      1642
weighted avg       0.79      0.79      0.78      1642
```



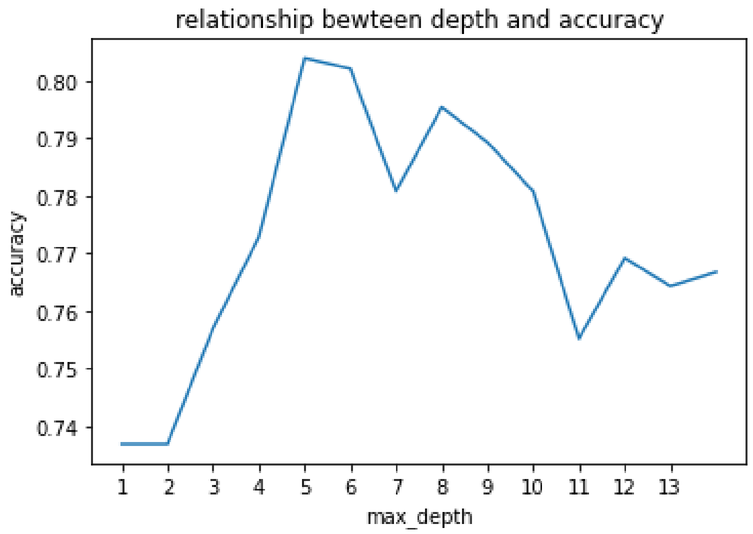


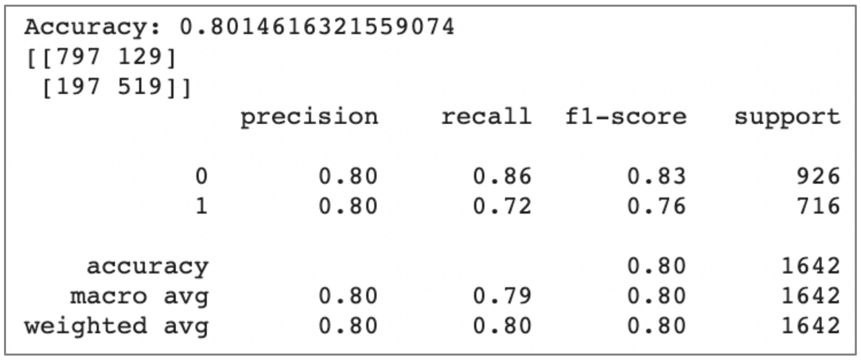
We can see from the results that when preprocessing is done, that the accuracy is approximately 0.79. Our classifier correctly identified 791 elements as Economy out of 926 possible elements (135 incorrectly classified) and classified 501 elements Luxury out of 716 possible elements (215 incorrectly classified). The AUC is also 0.87 and indicates that our model is able to distinguish the two class models pretty well. We can also see that our precision for both 0 and 1 is above .7 and likewise with our f1-score. So when comparing the model with our benchmark the accuracy of our new model increased by 3% (and our AUC score increased by 0.03). So with pre-processing we can see that our model has slightly better performance than without pre-processing.

b. Decision Tree

With decision tree, the main changes we focused on when trying to improve the model's accuracy were the hyper parameters. We decided to employ pre-pruning to prevent overfitting and see at which depth we would receive the best accuracy. We used a combination of k-fold analysis and python packages to find the best depth at which the tree could perform. After running that analysis we saw that the best depth for our model was 5. Another hyper-parameter we changed was switching from the gini-index to entropy.

Let's take a look at our results by feeding the model with pre-processed data:

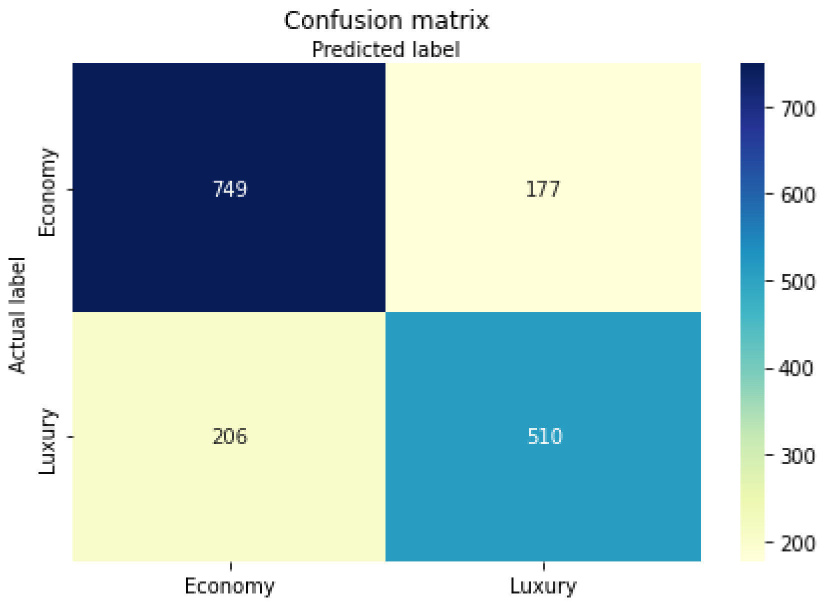


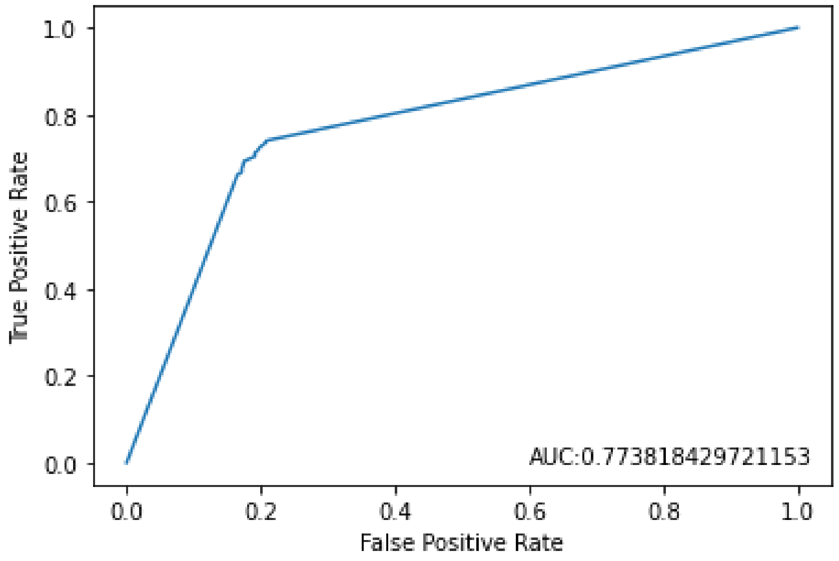


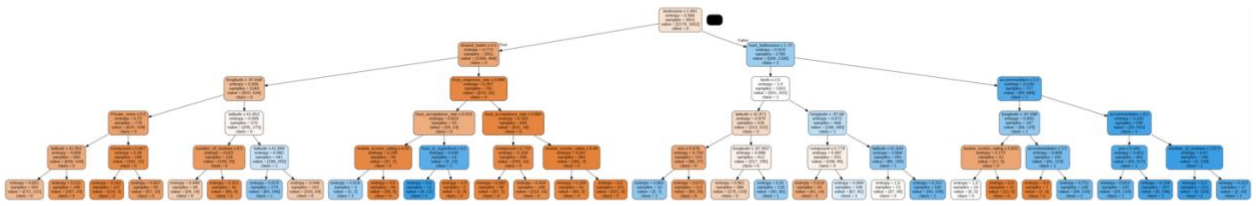
```
Accuracy: 0.8014616321559074
[[797 129]
 [197 519]]
              precision    recall  f1-score   support

           0       0.80      0.86      0.83       926
           1       0.80      0.72      0.76       716

    accuracy                           0.80      1642
   macro avg       0.80      0.79      0.80      1642
weighted avg       0.80      0.80      0.80      1642
```





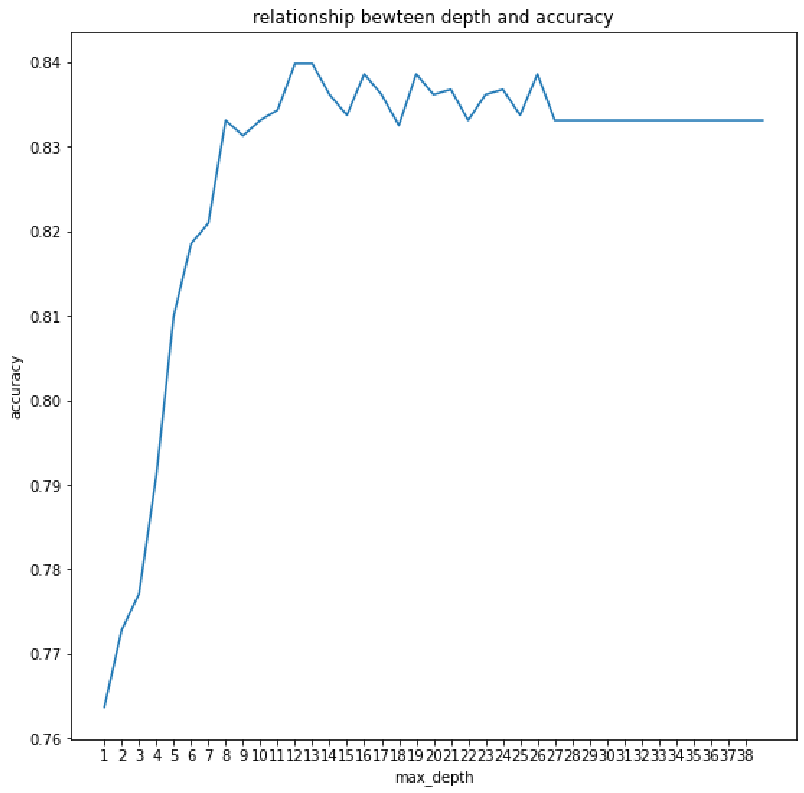


From our results we can see that our model is strong. The accuracy is approximately 0.80 (which is slightly better than our logistic regression classifier and increased 0.2 compared with benchmark) and our precision and f1-score for both classes are pretty high. We can see from the confusion matrix that for the Economy class 803 observations were classified correctly (with 123 misses) and 506 Luxury class observations were classified correctly (with 210 misses). Our AUC was 0.87 suggesting that the model is capable of distinguishing the classes as well. Overall, our decision tree model is slightly better than the benchmark.

c. Random Forest

With Random Forest, we conducted very similar steps as we did with decision tree. We conducted steps to find the max depth (pre-pruning and various python packages). Also we graphed the accuracy based off of depth to see which depth would be the best for our model. After running everything we determined that the best depth for our random forest classifier was 12.

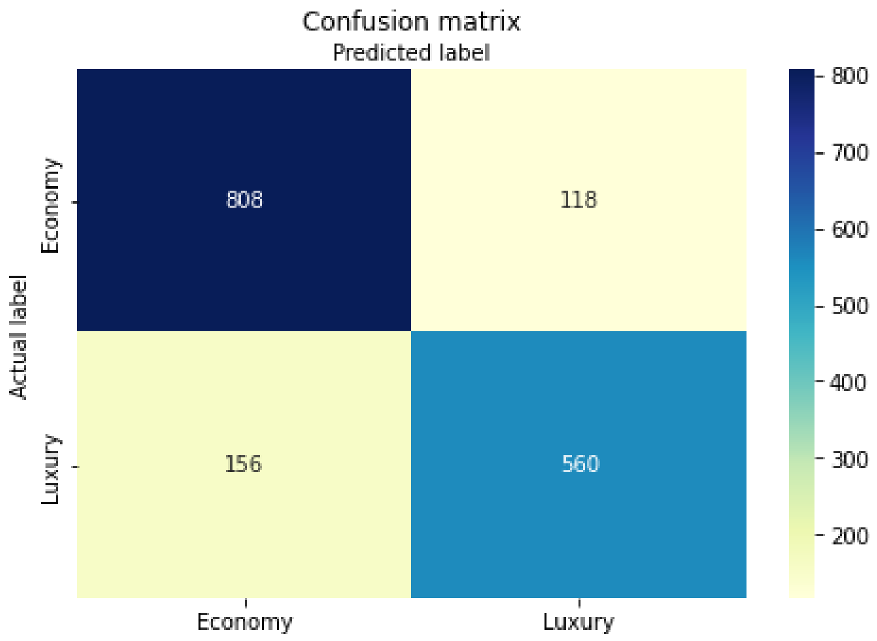
After performing the random forest classifier on our preprocessed data we received these results:



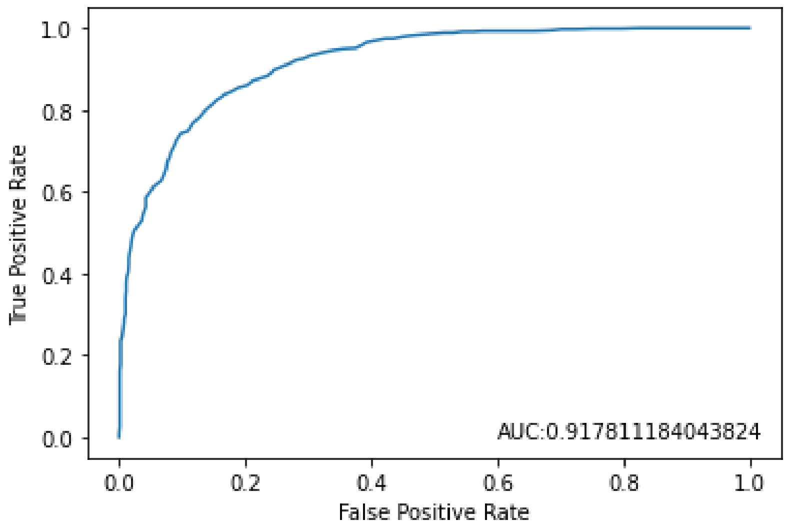
```
Accuracy: 0.8398294762484775
[[817 109]
 [154 562]]
              precision    recall  f1-score   support

           0       0.84      0.88      0.86       926
           1       0.84      0.78      0.81       716

    accuracy                           0.84      1642
   macro avg       0.84      0.83      0.84      1642
weighted avg       0.84      0.84      0.84      1642
```



\

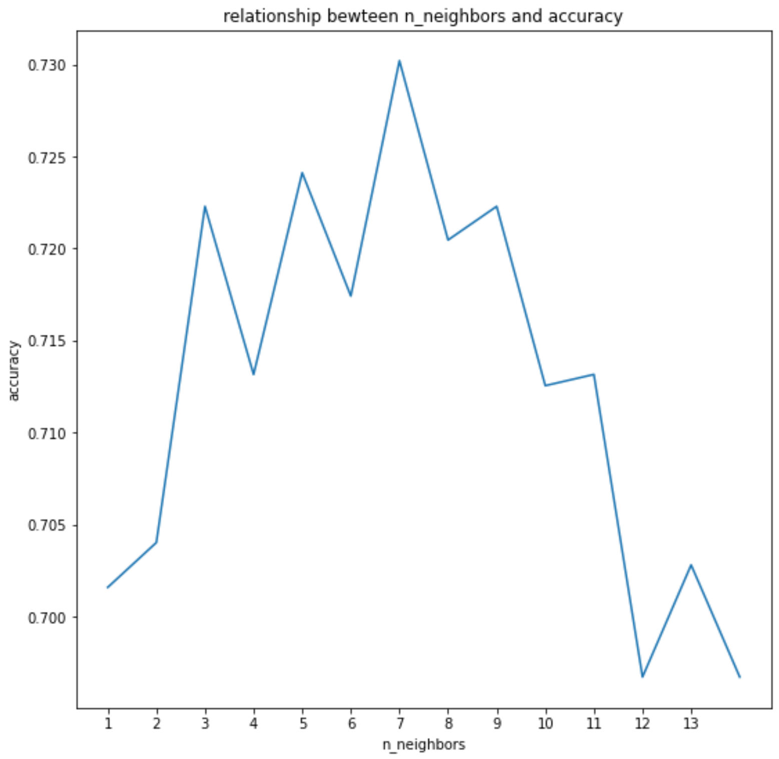


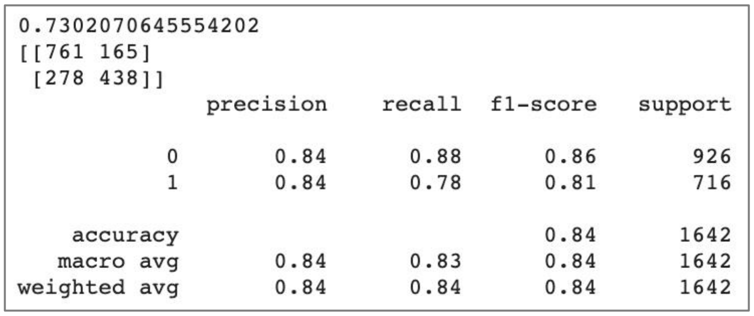
As expected, our random forest accuracy exceeds that of all the other models. The accuracy we received from this model was approximately 0.82 which increased by 3%. Looking at the classification report we can also see that there is high precision and f1-score for both classes. The model had 812 hits for the Economy class (114 misses) and 532 hits for the Luxury class (184 misses). The AUC score, was also the highest so far at approximately 0.90. This is by far our strongest model.

d. K-nearset Neighbors

For KNN we also focused on changing the hyperparameters in order to improve accuracy. Since the model's success heavily depends on the value of K we focused on this parameter the most. We modeled the change in accuracy vs the change in K to determine finally which K we should use. After running this, we saw that the accuracy of the model didn't really fluctuate too much when we changed K but the K with the best accuracy was 7.

After running KNN on our data we received these results:

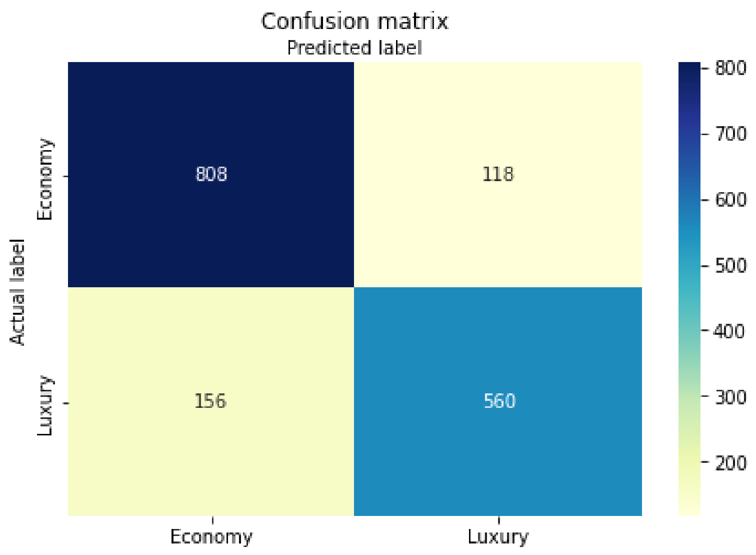


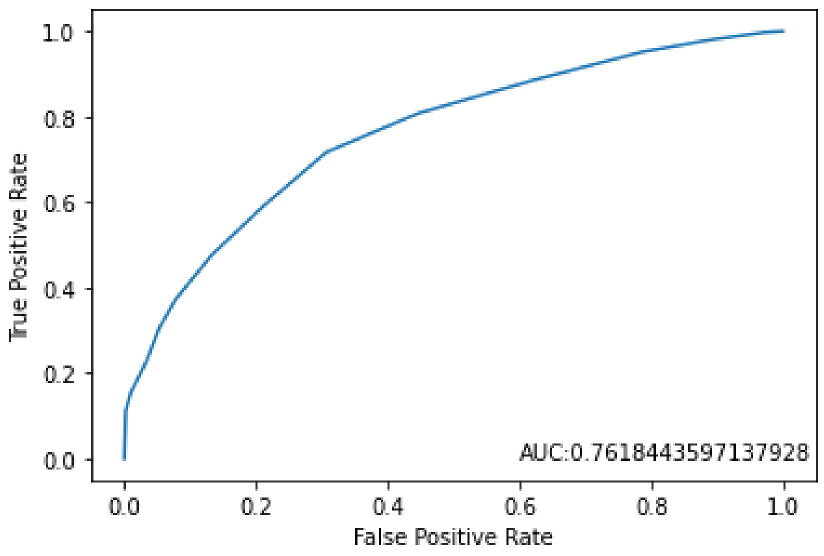


```
0.7302070645554202
[[761 165]
 [278 438]]
              precision    recall  f1-score   support

           0       0.84      0.88      0.86       926
           1       0.84      0.78      0.81       716

    accuracy                           0.84      1642
   macro avg       0.84      0.83      0.84      1642
weighted avg       0.84      0.84      0.84      1642
```





Our KNN model, with pre-processing, remain the worst out of all the models. The accuracy of our model is 0.72. When we take a look at the confusion matrix we can see that the model had 811 hits in the Economy class (115 misses) and 372 hits in the Luxury class (344 misses). We can see from this (also from the precision and f1-score) that KNN is still much weaker at classifying the Luxury class than our other models. The AUC is 0.78 meaning that, the model is capable of distinguishing the two classes from each other. Despite our KNN model has the worst performance among all of the model in this section, the accuracy of the KNN model increased by 4% compare with without pre-processing. So this is a strong indicator that our data pre-processing has positive impact on our model.

5. Key Takeaways for each model

1. For logistic regression, our pre-processed model yielded better results than our benchmark model. We saw a 3% increase in accuracy, but even with this increase it was still not our best performing model.
2. With decision tree, our model's accuracy increased by 2% when using the pre-processed data. It performed slightly better than the logistic regression.
3. For random forest, this machine learning model out performed all of other models both in benchmarking and with pre-processing. This is because random forest has a strong noise immunity capacity and uses ensemble methods to get the best results. The only down side to this model is the time and space complexity.
4. With KNN, we had the least success. The model performed the worst in most metrics. Although the accuracy increased by feeding the model with pre-processed data, it still was much weaker than the other models, especially when classifying the Luxury class.

Method	Accuracy	Precision	Recall	AUC	F1-score
LogisticRegression(benchmark)	0.76	0.76	0.76	0.85	0.76
LogisticRegression(pre-processing)	0.79	0.79	0.79	0.87	0.78
DecisionTree(benchmark)	0.78	0.79	0.78	0.86	0.78
DecisionTree(pre-processing)	0.79	0.80	0.80	0.87	0.80

RandomForest(benchmark)	0.79	0.79	0.79	0.88	0.79
RandomForest(pre-processing)	0.82	0.82	0.82	0.90	0.82
KNN(benchmark)	0.68	0.71	0.68	0.73	0.69
KNN(pre-processing)	0.72	0.82	0.82	0.78	0.82

6. Conclusion

Our goal with this project was to use the data we received from Airbnb to suggest a method of grouping and presenting the appropriate price into two segments (economy and luxury). By implementing the machine learning methods we learned from the class, we were able to successfully accomplish this. The random forest method (the most accurate model we created) helps users to make appropriate decisions regarding pricing with 84% accuracy. Areas of improvement that we would employ upon further expansion of this project would be: multi-class classification (not just economy and luxury, but also categories in between) and further accuracy improvement. Overall, we learned that machine learning is very effective and that Airbnb should employ this technique (if they haven’t don’t so already) to improve their service quality.