Data Mining for Business (BUDT758T)

# Competitive Airbnb Host


## Team Members:

## Di Sun

## Sanjing Xu

## Wenxi Wu

## Yiqing Pan

## Zezhong Zhang

ORIGINAL WORK STATEMENT

We the undersigned certify that the actual composition of this proposal was done by us and is original work.

| Work | Typed Name | Signature |
|---|---|---|
| Model Optimization Report | **Di Sun** | DS |
| Model Optimization Report | **Sanjing Xu** | SX |
| Data Processing Model Selection | **Wenxi Wu** | WX |
| Model Optimization Report | **Yiqing Pan** | YP |
| Data Processing Report | **Zezhong Zhang** | ZZ |

## Executive Summary

Our report was commissioned by Airbnb, the peer-to-peer house sharing platform, to accurately predict which listings would have a high review score rate and identify which factors influence listing score rates.

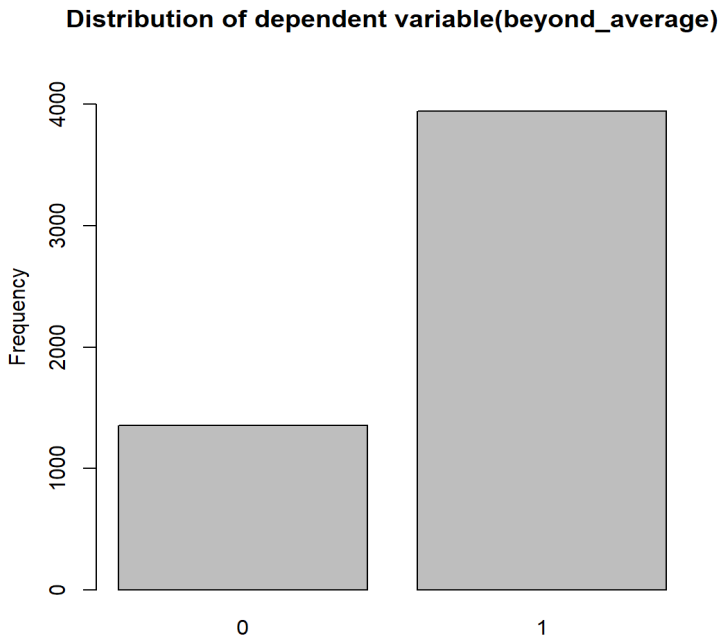
Though Airbnb gives hosts a variety of metrics to help them derive insights, it currently does not have a way to accurately predict if listings are expected to be highly scored or not. Our goal was to be able to predict whether or not listings were beyond average ratings with as high of an accuracy as possible so that we can identify listings that can be modified to become beyond average ratings. Thus, our most important performance measure in our prediction models was accuracy. The end goal is to have a classification prediction model that can be used on future listings to predict whether or not we expect them to be highly scored or not with a high accuracy.

During this prediction process, we need to choose and transform the variables from 74 columns of original data, then select features and choose the best model and parameters to get highest accuracy on the test data set. After trying 6 models, we finally choose the Random Forest model because of its high accuracy, then we can analyze the significant variables to give listing owner recommendations. In the model selection process, we have four steps to be done:

First, analyze every column in the training data set, then figure out how we should clean each column (deal with null value and missing value). Second, decide how to process text columns, for example, the amenities, we do text mining analysis, to extract some meaningful amenities and then transform them to dummy-like variables. Third, try different models, adjust parameters and then compare the best accuracy of each model to choose the best one. Last, based on the parameters which give us best accuracy, we analyze the features that Random Forest is significant for our prediction, then give the owner recommendation to help them get a high score rate.

# Data Description

- **Distribution of dependent variable**




- **Data Source(s)**
    - Washington D.C airbnb housing dataset from inside Airbnb
        - http://insideairbnb.com/get-the-data.html
- **Variables(dependent and independent variables should be identified if relevant)**
  We use domain knowledge and personal experiences and preferences to narrow down the variables we believed were important and should be included in our cleaned dataset:
    - Sample size: 5295
    - 11 independent variables and 1 dependent variables
    - dependent variable is review_scores_rating(using a cutoff and change it into **'beyond_average'**)
- **What the data are**
    - Deleting repeated geographical columns: We delete host_neighbourhood, neighborhood, neighborhood_cleansed, neighborhood_group_cleansed, latitude, longitude. Because we only focused on Washington.DC area, too much

information about geography is not necessary for us to keep it, so we only keep host_location;

- Removing all columns with individual host info: For id, scrape_id, last_scraped, name, host_id, host_name, host_about, host_response_time, host_response_rate, host_acceptance_rate, host_has_profire_pic, and we only keep host_is_superhost, host_identity_verified;
- Removing all columns with listing room info: For property_type, beds, minimum_nights, maximum_nights, minimum_minimum_nights, number_of_reviews, ect, so we only keep room_type, accommodates, bedrooms, amenities, prices, instant_bookable;
- Create dummy variables for specific models: after dealing with missing values, we create dummy variables for all categorical variables;
- Numerical variables processing: Standardize format : remove $ in price;
- Date variables processing: convert ‘host_since’ to ‘host_years’: For host_since, we subtract with system time and calculate how many years the host has been. Eg, we convert 03/01/2008 into 14 years, so we keep ‘host_years’;’
- Categorical list: Convert ‘host_location’ to ‘host_inside_dc’’, when the host is in Washington DC, we assign them to 1, otherwise 0. Convert ‘room_type’ to ‘room_type_binary’: when the room type is the ‘shared room’’, we assign them to 0; when the type is ‘hotel room’, we assign them to 1; when the type is ‘private room’, we assign them to 2; and assign ‘Entire room/apt’ is 3. Convert ‘review_scores_rating’ to ’beyond_average’: when the review_score_rating is over the average of mean for all rates in the column, we assign them to 1, otherwise 0.

- **Why the data are of interest**
  - It is competitive for hosts to get a high review score rating on the Airbnb platform. So it makes sense to quantify important elements highly related to each house in order to help hosts improve their houses’ score rating.

## Research Questions

- Which amenity influences the score rating the most?

- What independent variables will influence the score rating most?

- When a host has a new house to rent, how can he/she know the score rating of their house and how to improve it？

## Methodology

We decided to first try out **Logistic Regression** because not only is it a tried-and-true classification method, but it also outputs an easily interpretable output that explains the relationship between the predictor variables and the outcome variable. In this case, our problem is a traditional classification problem(beyond rating is either 1 or 0). However, it should be noted that with either selected features or all features, our team discovered an accuracy of around 75.26% on our validation/test data.

We then tried **KNN** and **Naive Bayes** to see if these methods would improve our accuracy. KNN  is one of the simplest and widely used algorithms which depends on its k value(Neighbors). After all, we got accuracy of KNN with approximately 76.58%. The final output shows that we built a Naive Bayes classifier that can predict if a review is beyond the average rates or not, with an accuracy of approximately 72.9%. We can increase model accuracy in the train test while adding more observations.

**Classification tree** is a type of supervised learning algorithm that is mostly used in classification problems. We thought a classification tree would be helpful since it has an easily interpretable output, and would automatically split the tree on the most important

variables to predict if one case is beyond_average_rate(1) or not(0). Again, the accuracy of the classification tree model is 70.25%.

Among all algorithms, as an implementation of gradient boosted decision trees **XGBoost** is good at speed. We used this algorithm focusing on improving the model's performance which is measured by prediction accuracy. Without feature selection, full features were put into this model to get accuracy. One essential way to improve the model's performance was to adjust parameters of XGBoost manually. Finally, we adjusted parameters like objective, max_depth, eta, gamma, colsample_bytree, min_child_weight, nrounds and got accuracy 76.58%.

We also tried to use an ensemble method like **Random Forests** because they are known for their ability to model complex relationships and predict with high accuracy since combining several trees together usually improves accuracy. Random forest was our BEST model type with 78.56% accuracy. It was selected to be tested further, tuned, re-tested repeatedly to improve accuracy measures. We believe that this process optimizes time and effort to produce the most accurate predictions.

## Results and Finding

| **Model** | **Accuracy** |
|---|---|
| Classification tree | 0.7025496 |
| Naive bayes | 0.7289896 |
| Logistic regression | 0.7525968 |
| KNN | 0.7658168 |
| xgboost | 0.7658168 |
| RandomForest | 0.7856468 |

*Figure. Performance of each model*

From the results we got by running models above, we can tell the random forest model has the best accuracy, which is 0.7856.

We can also identify from the roc curve plot for each model,  the red line means randomforest, blue line means xgboost, black line means logistic regression, orange line means KNN, purple line means Naive Bayes, and green line means tree model. Random forest has the highest AUC among the other models, which is 0.781. The model that has better performance on prediction will have higher AUC.

Diving into the random forest model, we found out the top five variables are Host_is_superhost, host_years, host_total_listing_count, price, and accommodates. The chart below shows the MeanDecreaseGini of each variable, higher indicating significance of the variables.

Washer, coffee, parking, shampoo, workspace are the top five most important factors in amenities for a host to consider. We concluded from MeanDecreaseGini of the amenities plot. When a host decides to list a new property on Airbnb, he should consider whether these amenities are fittable for the guests.

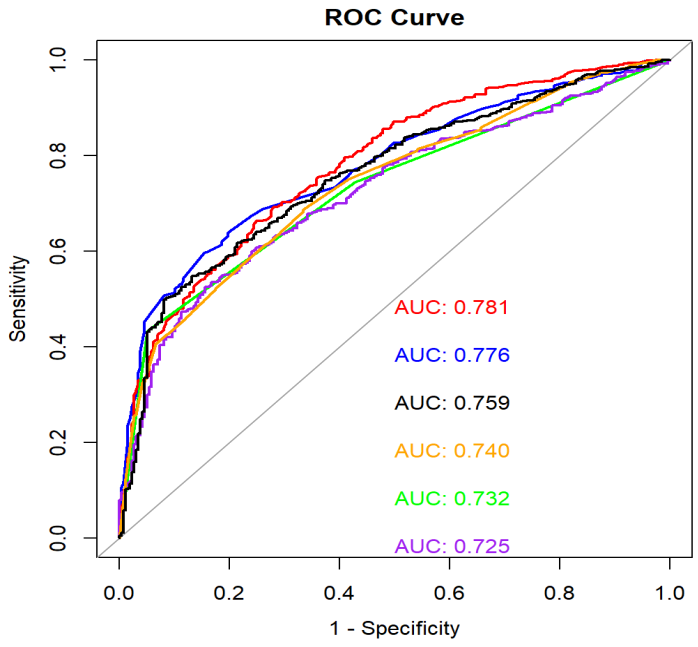



*Figure. ROC Curve of models*

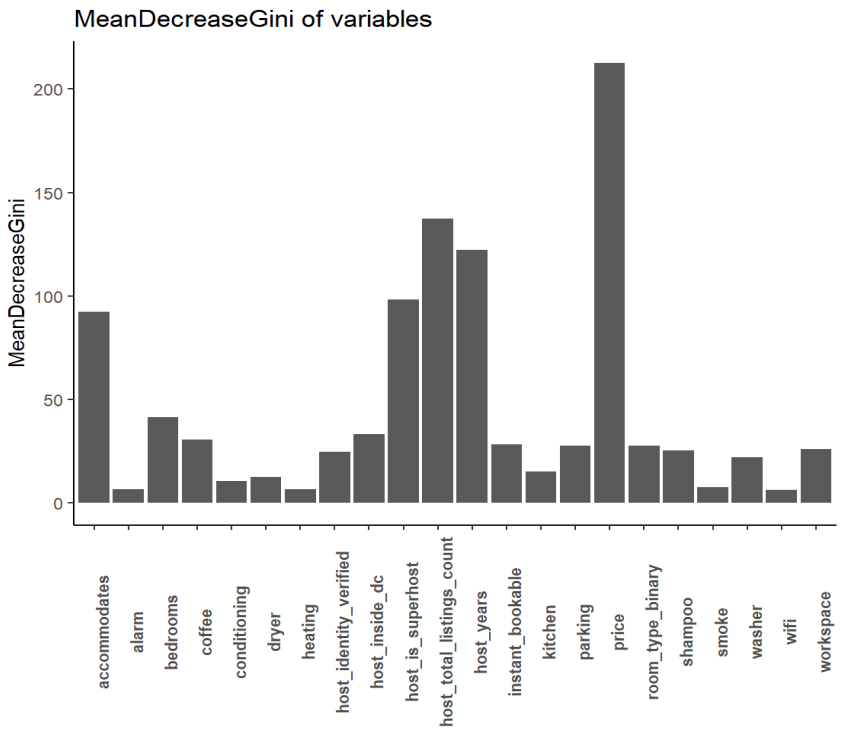



*Figure. MeanDecreaseGini of dependent variables*

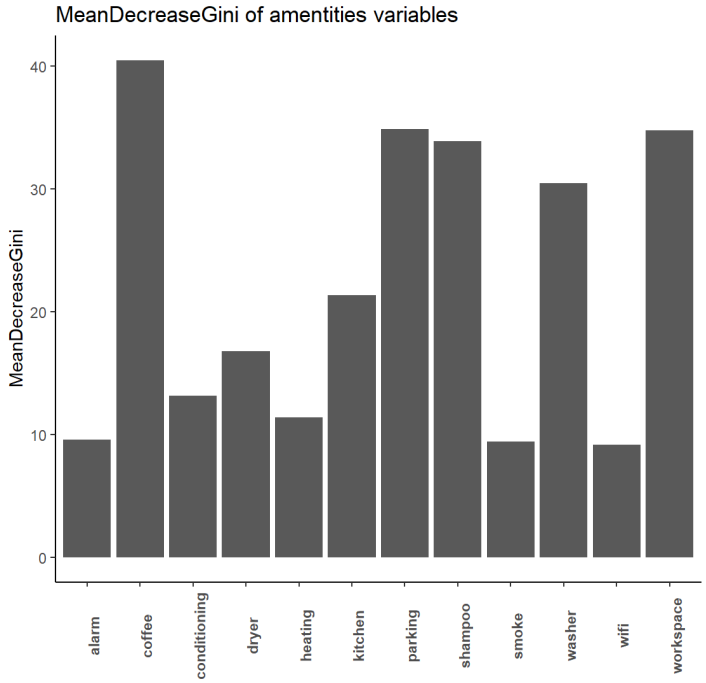



*Figure. MeanDecreaseGini of amenities*

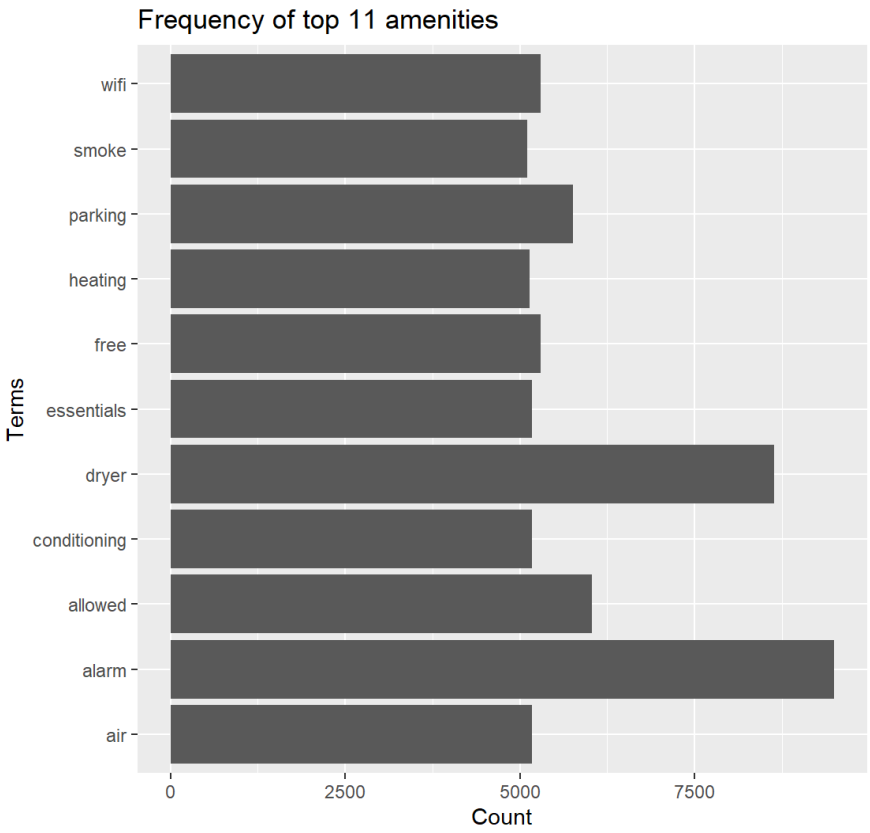



*Figure. Frequency of amenities*

## Conclusion

Through preprocessing the data and running analysis with different predictive models. We can conclude that based on the parameters which give us best accuracy, we analyze the features that Random Forest is significant for our prediction. Therefore, We want to emphasize some independent variables that have significant influence on the dependent variable(beyond_average).

First are Amenities: Washer, coffee, parking, shampoo, workspace are the top five most important factors for a host to consider. When a host decides to list a property on Airbnb, he should consider whether these amenities are fittable for the guests. Second is price, if the host decides to list a new property on Airbnb, the price is still the key consideration for him/her to consider. Setting a reasonable price is also important for high score rating. Third variable is accommodations. From which spaces in your

place are available to guests to the number of guests who can stay, you can tailor your booking settings based on your needs, availability, and preferences. If you want to gain a high score rating, try to make your accommodations as high as possible. Last but not least is superhost: A Superhost is someone who goes above and beyond in their hosting duties and is a shining example of how a Host should be. You can easily identify one from the badge that appears on their listing and profile. Getting superhost verification would be more likely to gain a higher score rating.

# Appendix (Any additional information to be submitted):

## Model Evaluation

- **Data Partition**

From the original dataset we split data into 80% training data and 20% testing data for the models that we run on our non-imputed training dataset.

- **Modeling/Methodology**

### 1) logistic regression

Logistic Regression is a classification method that models the probability of an observation belonging to one of two classes. In this case, our problem is a traditional classification problem(beyond rating is either 1 or 0), so the first model that came into our mind is logistic regression, which is also the first classification model we learned in this semester. First, we put our partitioned training data with all features into our feature selection model, the first several lines of our logistic coefficients are as below:

```
Call:
glm(formula = beyond_average ~ ., family = "binomial", data = df_train[,
    -7])

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6190  -0.9541   0.3800   0.8623   1.8920

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)             -1.027e+00  4.880e-01  -2.105   0.0353 *
host_is_superhostt       1.987e+00  1.307e-01  15.194  < 2e-16 ***
host_total_listings_count  1.747e-04  1.711e-04   1.021   0.3071
host_identity_verifiedt    2.018e-01  1.056e-01   1.912   0.0559 .
accommodates            -5.509e-02  3.115e-02  -1.769   0.0770 .
bedrooms                 2.401e-02  7.625e-02   0.315   0.7528
price                    1.908e-05  1.771e-04   0.108   0.9142
instant_bookablet       -4.020e-01  8.645e-02  -4.650 3.31e-06 ***
workspace               -2.267e-01  1.021e-01  -2.220   0.0264 *
wifi                     2.397e-01  2.885e-01   0.831   0.4059
washer                   1.653e-01  1.298e-01   1.274   0.2026
smoke                    2.641e-01  5.756e-01   0.459   0.6464
shampoo                  1.115e-01  1.094e-01   1.019   0.3082
parking                 -9.797e-04  9.864e-02  -0.010   0.9921
kitchen                 -9.686e-02  1.568e-01  -0.618   0.5368
heating                 -1.697e-01  2.358e-01  -0.719   0.4719
dryer                    1.286e-01  1.900e-01   0.677   0.4985
conditioning             3.374e-01  2.391e-01   1.411   0.1582
coffee                   1.830e-01  9.398e-02   1.948   0.0515 .
alarm                   -6.714e-02  6.040e-01  -0.111   0.9115
host_years              -2.287e-02  1.706e-02  -1.341   0.1800
host_inside_dc           6.021e-01  9.247e-02   6.511 7.46e-11 ***
room_type_binary1       -9.500e-01  6.509e-01  -1.460   0.1444
room_type_binary2        4.591e-01  2.836e-01   1.619   0.1054
room_type_binary3        9.145e-01  2.802e-01   3.263   0.0011 **
```

As the output above shows, some features such as accommodates, instant_bookable have a negative relationship with beyond_rate, meaning that the smaller the accommodates is, the less possible the room is going to get a higher review rate. Conversely, room_type_binary3 has positive relationships with beyond_average_rate, indicating the private room type is the more possible the room is going to get a higher review rate. Later we used some feature selection methods, and these features with very low p-value in the logistic models were almost all selected in our final selected features.

However, it should be noted that with either selected features or all features, our team discovered an accuracy of around 75.25% on our validation/test data.

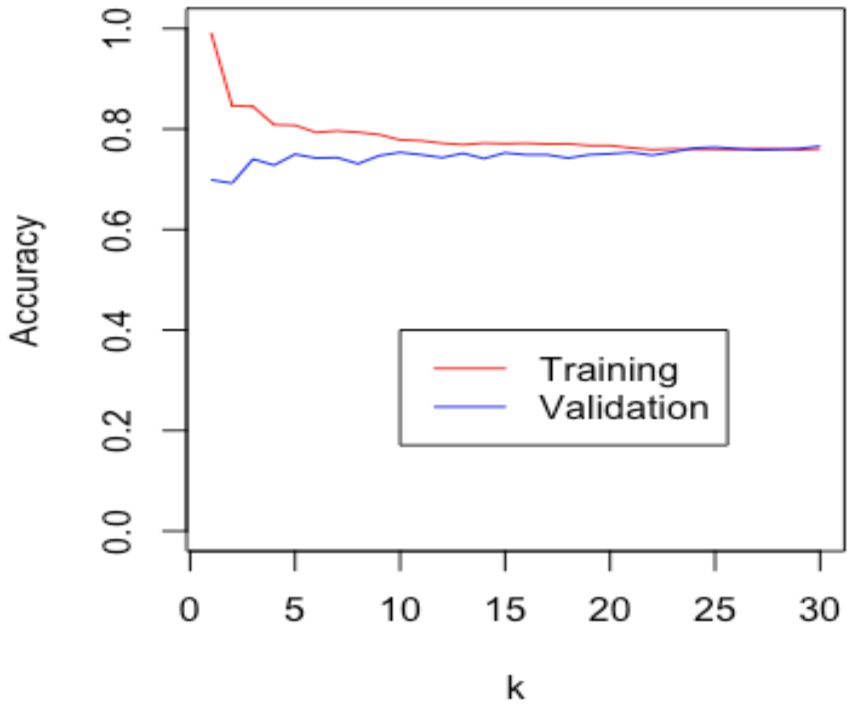
```
   pred_logistic_class
      0   1
  0 111 146
  1 116 686
> acc_logistic
[1] 0.7525968
```

2) KNN

K-Nearest Neighbor or K-NN is a Supervised Nonlinear classification algorithm. K-NN is a Non-parametric algorithm i.e it doesn't make any assumption about underlying data or its distribution. It is one of the simplest and widely used algorithms which depends on its k value(Neighbors).

In this model, we make 'host_is_superhost','host_identity_verified','instant_bookable','room_type_binary' to be dummies. Then, variables in KNN need to be scaled and converted into numerical datatype.

First, we try to use the train function in caret because it can find the best K automatically. When K=30, we get an accuracy of 76.58%. Compared to logistic regression, KNN havs minus improvement, so let's wait and see the performances from other models.




**3) Naive Bayes**

Naive Bayes is a classification technique based on Bayes' Theorem with an assumption of independence among predictors.

According to our project, we convert the 'beyond_average' to factor at the beginning. The Conditional probability for each feature or variable is created by model separately. The apriori probabilities are also calculated which indicates the distribution of our data.

```
> head(cbind(pred_NB, df_test[,-c(7,24)]))
   pred_NB host_is_superhost host_total_listings_count host_identity_verified accommodates bedrooms price instant_bookable
7        1                f                         4                      t            1        1    42                f
11       1                t                         1                      t            4        1   125                f
12       1                t                         1                      t            3        1    90                f
17       1                f                         2                      t            8        3   379                t
27       0                t                         5                      t            1        1    99                f
31       1                f                         1                      t            2        1   135                f
   workspace wifi washer smoke shampoo parking kitchen heating dryer conditioning coffee alarm host_years host_inside_dc
7          1    1      1     1       1       1       1       1     1            1      1     1         13             1
11         1    1      1     1       1       1       1       1     1            1      1     1         12             1
12         1    1      0     1       1       1       1       1     1            1      1     1         12             1
17         0    1      1     1       0       1       1       1     1            1      0     1         12             1
27         0    1      1     0       0       1       1       1     1            1      0     0         13             1
31         1    1      1     1       1       1       1       1     1            1      0     1         11             1
   room_type_binary
7                 2
11                3
12                2
17                3
27                2
31                3
```

To check the efficiency of the model, we are now going to run the testing data set on the model, after which we will evaluate the accuracy of the model by using a Confusion matrix. The final output shows that we built a Naive Bayes classifier that can predict if a review is beyond the average rates or not, with an accuracy of approximately 72.9%. You can increase model accuracy in the train test while adding more observations.

```
   pred_NB
      0   1
  0  57 200
  1  87 715
> acc_NB
[1] 0.7289896
```

**4) Classification Tree**

Classification tree is a type of supervised learning algorithm that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this project, we used a classification tree to do prediction for binary prediction, whether one case is beyond_average_rate(1) or not(0). For our classification tree model, full features were used, the value (class) obtained by the terminal node in the training data is the mode of observations falling in that region.

```
> summary(model_full_tree)

Classification tree:
tree(formula = beyond_average ~ ., data = df_train[, -7])
Variables actually used in tree construction:
[1] "host_is_superhost"         "host_total_listings_count"
Number of terminal nodes:  3
Residual mean deviance:  0.9925 = 4201 / 4233
Misclassification error rate: 0.259 = 1097 / 4236
```

Because there is not a big difference between cv.tree and original tree, we do not prune the tree. Again, the accuracy of the classification tree model is 70.25%.

**5) Random forest**

We know that random forest is also an “ensemble” function based on TREEs. Each tree makes their classification decisions based on different variables.

To keep the result really random, Random Forests do this in two ways:

• The first trick is to use bagging, for bootstrap aggregating. Bagging takes a randomized sample of the rows in your training set.

• The second source of randomness gets past this limitation though. Instead of looking at the entire pool of available variables, Random Forests take only a subset of them, typically the square root of the number available.

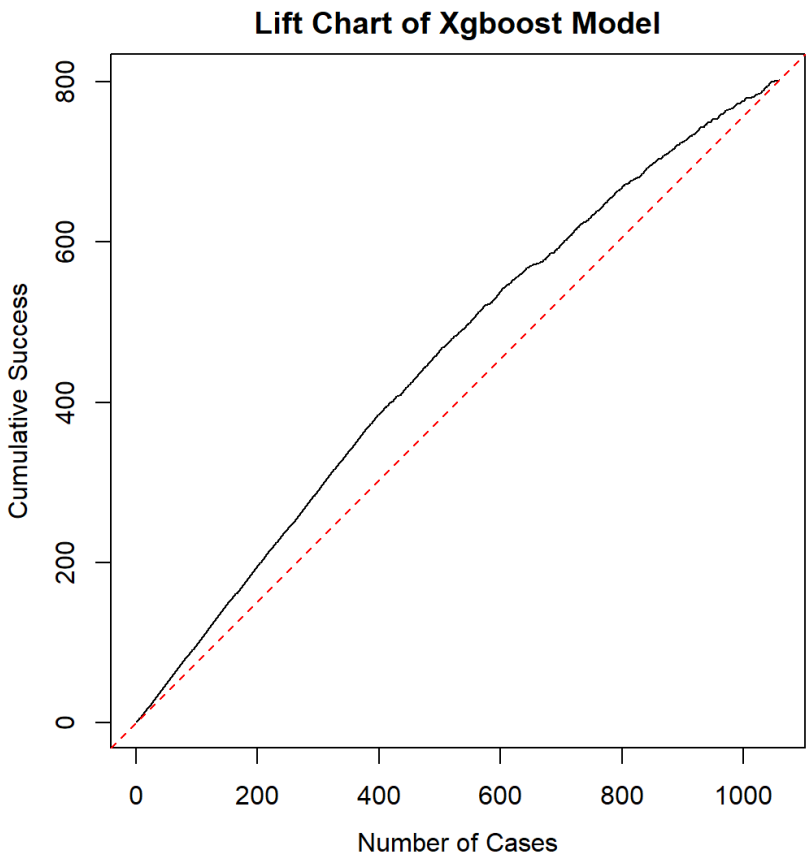
```
> model_randomForest

Call:
 randomForest(formula = beyond_average ~ ., data = df_train[,      -7], mtry = 5, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 5

        OOB estimate of  error rate: 22.78%
Confusion matrix:
    0    1 class.error
0 375  722  0.65815861
1 243 2896  0.07741319
```

Hence, we did not use cross validation in random forest. When we are training our model, we want to keep our random forest as random as possible. Finally, we found out that with ntree=500 will provide us a 78.56% accuracy on our test data. This model was selected to be tested further, tuned, re-tested repeatedly to improve accuracy measures since they performed significantly better than the other models.

```
> acc_randomForest<-(CM_randomForest[1,1]+CM_randomForest[2,2])/sum(CM_randomForest)
> acc_randomForest
[1] 0.7856468
```

**6) Xgboost**

Among all algorithms, as an implementation of gradient boosted decision trees XGBoost is good at speed. We used this algorithm focusing on improving the model's performance which is measured by prediction accuracy. Without feature selection, full features were put into this model to get accuracy. One essential way to improve the model's performance was to adjust parameters of XGBoost manually. Finally, we adjusted parameters like objective, max_depth, eta, gamma, colsample_bytree, min_child_weight, nrounds and got accuracy 76.58%.


Lift Chart of Xgboost Model
Cumulative Success
0
200
400
600
800
0
200
400
600
800
1000
Number of Cases