AIRBNB – REPORT - OUTLIERS

SONALIKA BHANDARI| SRAGDHARA PATTANAIK| KIRUTHIKA SANKARAN| ROHINI SHARMA| RASHI TANDON

Team: Outliers

Contents

1. EXECUTIVE SUMMARY

This report has been compiled by 'Team Outliers' and enlists the approach, methods and results upon analyzing the given Airbnb data to predict how popular an Airbnb listing is based on the features of the listing. The appendix delineates the individual contribution towards the completion of this project.

Airbnb collects data about its listings in terms of location, house features, amenities, pricing and likewise. These features however, not only highly influence travelers' decisions to book the listing or not but also provide effective insights and feedback to the owner as to which features are more likely to skew the travelers' decision. To assess the 'high booking rate' of the listing as a binary classification problem, an iterative process of including different features by using multiple models was followed.

The first step entailed data pre-processing which included identifying the features that would aid the decision making by using domain knowledge and then cleansing the data for any missing or misplaced values for the identified features. To bolster these findings lasso regression and recursive feature elimination were run so as to identify the variables that consistently contribute to the 'high booking rate' decision. Furthermore, correlations between variables was ascertained to account for any interactions that would further enhance the prediction model.

Features like amenities, host verifications which are seemingly important were converted from text for multi-class categories which contributed reasonably to the prediction models. For other multi-class attributes like cancellation policy, missing values were replaced with majority class or based on domain knowledge. In essence, each variable was assessed based on the context of the problem and handled accordingly.

<u>Our observations and recommendations:</u>

Out of the different models tried; Logistic Regression, ADA Boosting, Gradient Boosting and Random forests, based on their corresponding performance and accuracy, Random Forest gave the maximum accuracy at 83.34%. Therefore, random forests, in this case does a better job at weighing the more significant variables.

Also, features like 'accommodates', 'bedrooms', 'bathrooms' etc. are important and could be derived from all models, but upon including more amenities iteratively boosted the accuracy of the model indicating that 'amenities' differentiate the listings from one another.

Further, inclusion of features like 'host verifications', and 'first review date', 'transit' and 'availability_365' are a consequence of their positive impact on the prediction model. Therefore, these features can attribute to the popularity of a listing and hence can provide meaningful insights to the listings' owner.

Future scope:

The findings can be further improved by using more data from Airbnb; for example count of reviews, reviews (if accounted through sentiment analysis), images of the listing etc. for each listing.

Secondly, this data focuses on North American listings and therefore, the results would be in general more applicable to future listings in North America. In order to have a more generalized estimate of popularity of listings, data from multiple geographies may be included.

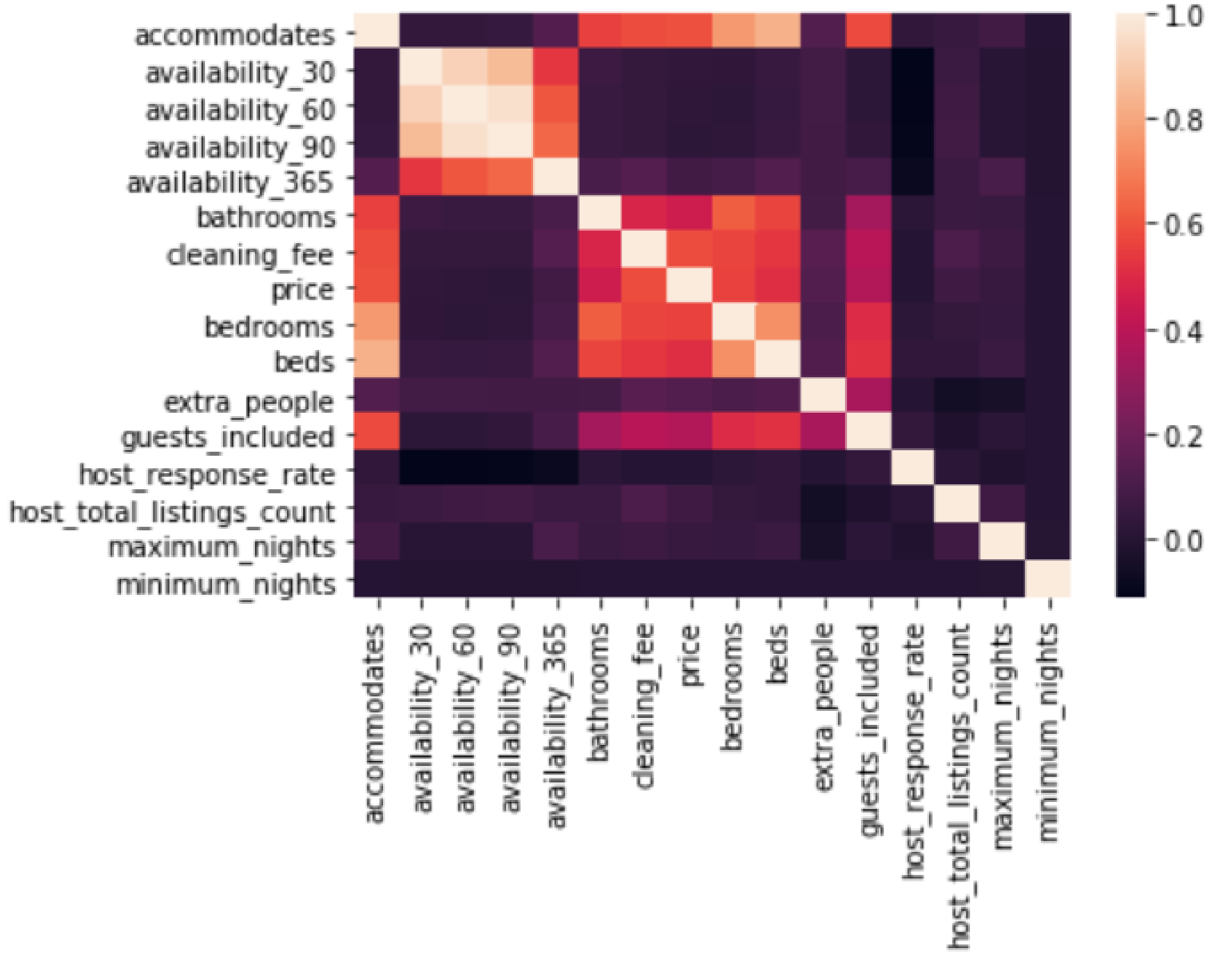
2. EXPLORATORY DATA ANALYSIS/FEATURE ENGINEERING

a. Data Preprocessing:

Type of Variable	**Method of Preprocessing**	**Variable names**
Numerical Variables	Removed the text entries and replaced the missing values with the mode of the variable	accommodates, availability_30, availability_90, availability_60, bathrooms, cleaning_fee, price, Bedrooms, host_response_rate, host_total_listing_count, maximum_nights, minimum_nights, guests_included
Categorical – 2 levels – True/False	Direct conversion of 0/1 – filled the missing values with majority class	security_deposit, host_has_profile_pic, host_identity_verified, host_is_superhost,instant_bookable, is_business_travel_ready, is_location_exact, monthly_price, require_guest_phone_verification, house_rules, weekly_price, requires_guest_profile_pic
Categorical – multi levels	filled the missing values with majority class.	bed_type, property_type, cancellation_policy, host_response_time, room_type
Text mining and conversion to multi-level categories	Did basic string match and converted to multilevel categories	host_verifications, transit
Date to categories	extracted the year from the date and converted the year to multilevel categories	first_review, host_since
Conversion to separate columns	Each of the amenities have been converted to a separate column and filled with 0/1 based on their availability.	amenities
Manual Preprocessing	Few variables have their entries from other columns. So preprocessed manually.	City, state, market, country, property type

b. correlation:

The correlation between the predictors have been found by the heatmap using matplotlib in python. Although addition of interaction variable improved the accuracy from 77.7% to 78.74% in our initial model, the same did not have any effect on our final random forest model. So we removed one of the two variables if interaction between two variables are found.
Variables removed – availability_60, beds



c. Feature Selection:

To find the significant predictors for the target, 3 methods were used.

LASSO Regression

Out of the 64 predictors we preprocessed and created lasso predicted 13 variables to be having significant impact on high_booking_rate. Adding these 13 variables alone in the model gave 78% of accuracy.

```
> coef(glmnet_lasso.cv, s="lambda.min")
65 x 1 sparse Matrix of class "dgCMatrix"
                                                1
(Intercept)                         -1.087368e+01
accommodates                         .
availability_30                      .
availability_60                      .
availability_90                      .
availability_365                     5.096133e-06
bathrooms                            .
bed_type                             .
bedrooms                             .
beds                                 .
cancellation_policy                  .
cleaning_fee                        -4.788734e-04
city                                 .
extra_people                         .
year_first_review                    .
guests_included                      .
host_has_profile_pic                 .
host_identity_verified               .
host_is_superhost                    1.267994e-01
host_response_rate                   .
host_response_time                  -3.612715e-02
host_total_listings_count            .
house_rules                          .
instant_bookable                     1.212924e-01
is_business_travel_ready             .
is_location_exact                    .
maximum_nights                       .
minimum_nights                       .
monthly_price                        .
price                               -5.321407e-05
require_guest_phone_verification     .
require_guest_profile_picture        .
requires_license                     .
room_type                            .
security_deposit                     .
state                                .
weekly_price                         .
wifi                                 .
parking                              .
Kitchen                              .
Breakfast                            .
ac                                   .
Heating                              .
year_first_review.1                  .
year_host                            5.456040e-03
Reviews                              .
Email                                .
Phone                                .
Facebook                             .
transit_available                   -9.683296e-03
hair_dryer                           4.915259e-02
hangers                              1.163394e-02
iron                                 5.720110e-03
laptop                               .
shampoo                              1.278625e-02
essentials                           .
washer                               .
dryer                                .
tv                                   .
pool                                 .
gym                                  .
cable                                .
wireless                             .
internet                             .
elevator                             .
```

Recursive Feature Elimination:

Recursive feature elimination gives out the important predictors for the target variable by eliminating the unimportant features. This gave additional 20 variables along with the 13 variables Lasso generated. This addition of 20 variables improved the accuracy to 82%

```
In [23]: model = LogisticRegression()
         # create the RFE model and select best attributes
         rfe = RFE(model)
         rfe = rfe.fit(X,y)
         # summarize the selection of the attributes
         print(rfe.support_)
         print(rfe.ranking_)
```

```
[ True False False False False  True False False False False  True  True
 False False  True False False  True False False False  True False False
  True False  True False  True False False  True  True  True  True  True
 False  True False False False  True False  True  True  True False  True
  True  True  True  True  True  True  True  True  True  True False False
 False False False False  True  True]
[ 1 15 31 16 32  1 18 19 24  5  1  1  9 11  1 22 29  1 12 14 28  1 30 13  1
  7  1 20  1 25 34  1  1  1  1  1 33  1  2 26 17  1 10  1  1  1  4  1  1  1
  1  1  1  1  1  1  1  1  8  3 27 21 23  6  1  1]
```

```
In [24]: print('Selected features: %s' % list(X.columns[rfe.support_]))
```

```
Selected features: ['accommodates', 'bathrooms', 'Kitchen', '24-hour-check', 'Heating', 'bedrooms', 'cancellation_policy', 'hos
t_has_profile_pic', 'host_is_superhost', 'host_response_time', 'minimum_nights', 'instant_bookable', 'is_business_travel_read
y', 'is_location_exact', 'room_type', 'monthly_price', 'require_guest_phone_verification', 'Phone', 'Email', 'verification', 'w
eekly_price', 'transit_available', 'hair_dryer', 'hangers', 'iron', 'laptop', 'shampoo', 'essentials', 'washer', 'dryer', 'tv',
'family/kid friendly', 'elevator']
```

Classifier Trees: Trees predicted the important features that could add some significance to the model.

```
In [16]: model = ExtraTreesClassifier()
         model.fit(X,y)

Out[16]: ExtraTreesClassifier(bootstrap=False, class_weight=None, criterion='gini',
                    max_depth=None, max_features='auto', max_leaf_nodes=None,
                    min_impurity_split=1e-07, min_samples_leaf=1,
                    min_samples_split=2, min_weight_fraction_leaf=0.0,
                    n_estimators=10, n_jobs=1, oob_score=False, random_state=None,
                    verbose=0, warm_start=False)
```

```
In [17]: print(model.feature_importances_)

[ 0.02000355  0.05071319  0.03957564  0.04525842  0.03962778  0.0134661
  0.00312877  0.02831715  0.0096405   0.01440858  0.00855124  0.01197594
  0.01030592  0.01189817  0.00436622  0.01862433  0.02799133  0.01738724
  0.01653844  0.01497037  0.02036705  0.0187598   0.02078204  0.01822734
  0.00038388  0.01291294  0.03027049  0.01218173  0.02801858  0.02428115
  0.01946768  0.02795454  0.03237341  0.00646246  0.01158474  0.01277001
  0.01943386  0.0013884   0.02387427  0.0215813   0.00304413  0.00384115
  0.01384007  0.00100926  0.00299171  0.00299428  0.01309937  0.00839498
  0.01272822  0.01975055  0.00944666  0.00978517  0.01163662  0.01060407
  0.00554917  0.01219493  0.00324428  0.01136833  0.00612408  0.00622289
  0.00631961  0.01327311  0.0097875   0.0084881   0.01461944  0.00981779]
```

3. MODEL EVALUATION

Target Variable Preprocessing:
Another important step before modelling is preprocessing the target variable. The target variable had 19 NA entries. We converted them to 0 as 0 is the majority class. It is also interesting to note that 70% of the target variable is classified as the majority class.

Baseline Model:
We did a majority class model as the baseline model, by setting the target variable to be all 0's. this gave the prediction of 75.5%. we used this model to evaluate our other models, to check the accuracy.

Evaluation:
In order to evaluate the model, we partitioned the data as 70% training data and 30% validation data. This data was chosen randomly every time by setting different seed.

Model Comparison:
Best model was chosen on the basis of accuracy. The model which gave higher accuracy is chosen as the best model. The confusion matrix was checked every time to make note of the precision and recall. Also, one more factor was we checked if the model generated the same 70-30 ratio of 0's and 1's on the test data.

4. MODELLING:

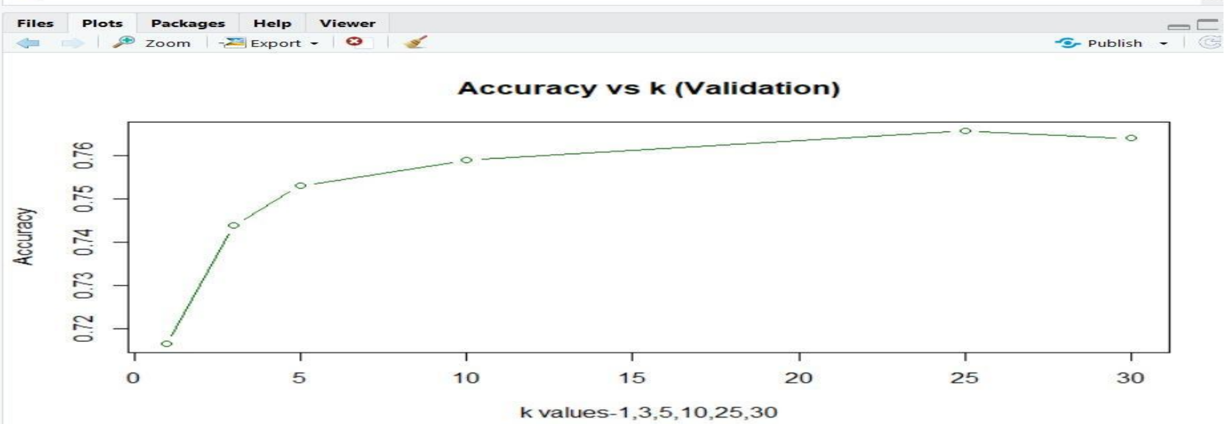
Logistic Regression: 39 predictors including 2 interactions variable between availability_30 and availability_60, availability_60 and availability_90. Accuracy: 78.74%

```
In [16]: X = data[['accommodates', 'availability_30', 'availability_60', 'availability_90', 'availability_365','bathrooms','bed_type',
                   'cleaning_fee', 'Wifi', 'parking', 'Kitchen', '24-hour-check', 'Breakfast', 'ac', 'Heating', 'property_type', 'price
                   'bedrooms', 'beds','security_deposit','state', 'cancellation_policy', 'extra_people', 'guests_included',
                 'host_has_profile_pic', 'host_identity_verified', 'host_is_superhost', 'host_response_rate', 'host_response_time',
                'host_total_listings_count', 'maximum_nights', 'minimum_nights', 'instant_bookable', 'is_business_travel_ready',
                 'is_location_exact', 'room_type', 'city','avai_30_60','avai_60_90']]
         y = train_y['high_booking_rate']
```

```
In [20]: predictions = logmodel.predict(X_test)
         accuracy_score(y_test, predictions)

Out[20]: 0.7874333333333333
```

K-NN Classifier: The accuracy was checked in KNN classifier by changing the K value every time. It is found that till k=25, the accuracy was constantly increasing and after which the accuracy dropped.

```
Console   Terminal
D:/RASHI TANDON/R_Script/Session 1/DMP/
> #view the Accuracy and k values in the data frame
> df_knn
   Accuracy  k
1 0.7166333  1
2 0.7438000  3
3 0.7530000  5
4 0.7589333 10
5 0.7657000 25
6 0.7639667 30
> #plot the graph of "Accuracy vs k" for Validation data
> plot(df_knn$k, df_knn$Accuracy, type = "b", main = "Accuracy vs k (Validation)", xlab =
 "k values-1,3,5,10,25,30", ylab = "Accuracy", col = "dark green")
>
```



ADA Boosting: 42 Variables were chosen based on an iterative process of including different variables and identifying the ones which contribute to the model's accuracy and also leveraging on lasso regression results:

Interaction:

- Availability_30*availability_60
- Availability_90*availability_60
- Accommodates*bedrooms
- Accommodates*beds

Model Tuning: Input parameters; number of trees and Learning factor was varied in order to select the best performing model; details of the trials as attached in **Appendix B**

```
In [44]: X = data[['accommodates','availability_30','availability_60','availability_90', 'availability_365', 'bathrooms',
                   'bed_type',
                      'cleaning_fee', 'Wifi', 'parking', 'Kitchen', '24-hour-check', 'Breakfast', 'ac', 'Heating',
                   'property_type', 'price', 'bedrooms', 'beds','security_deposit', 'cancellation_policy', 'state',
                   'extra_people', 'guests_included','host_has_profile_pic', 'Reviews', 'year_first_review',
                   'host_is_superhost', 'host_response_rate',  'host_response_time', 'host_total_listings_count',
                   'maximum_nights', 'minimum_nights', 'instant_bookable', 'is_business_travel_ready', 'is_location_exact',
                   'room_type', 'city', 'availability_interaction_1' ,'availability_interaction_3',
                   'interaction_accommodates_1', 'interaction_accommodates_2']]
         y = train_y['high_booking_rate']

In [45]:    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

In [46]: dt = DecisionTreeClassifier()
         clf = AdaBoostClassifier(n_estimators=200, base_estimator=dt,learning_rate=1)
         #Above I have used decision tree as a base estimator, you can use any ML learner as base estimator if it ac# cepts sampl
         clf.fit(X_train,y_train)

Out[46]: AdaBoostClassifier(algorithm='SAMME.R',
                   base_estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_split=1e-07, min_samples_leaf=1,
                     min_samples_split=2, min_weight_fraction_leaf=0.0,
                     presort=False, random_state=None, splitter='best'),
                   learning_rate=1, n_estimators=200, random_state=None)

In [47]: predictions = clf.predict(X_test)

In [48]: from sklearn.metrics import accuracy_score
         accuracy_score(y_test, predictions)

Out[48]: 0.8187333333333331
```

GRADIENT Boosting: Applied gradient boosting on selected 38 variables chosen through an iterative process of finding the ones which contribute to the model significantly and then varying the number of trees.

Interaction:

- Availability_30*availability_60
- Availability_30*availability_90
- Availability_90*availability_60
- Accommodates*bedrooms
- Accommodates*beds

Input parameters; number of trees was varied in order to select the best performing model; details of the trials as attached in **Appendix B**

```
In [165]: X = data[['availability_60', 'availability_365', 'bathrooms','bed_type',
                       'cleaning_fee', 'Wifi', 'parking', 'Kitchen', '24-hour-check', 'Breakfast', 'ac', 'Heating',
                    'property_type', 'price', 'bedrooms', 'beds','security_deposit', 'cancellation_policy', 'state',
                    'extra_people', 'guests_included','host_has_profile_pic', 'Reviews', 'year_first_review',
                    'host_is_superhost', 'host_response_rate',  'host_response_time', 'host_total_listings_count',
                    'maximum_nights', 'minimum_nights', 'instant_bookable', 'is_business_travel_ready', 'is_location_exact',
                    'room_type', 'city', 'availability_interaction_1', 'availability_interaction_2', 'availability_interaction_3',
                    'interaction_accommodates_1',  'interaction_accommodates_2']]
          y = train_y['high_booking_rate']

In [166]: from sklearn.ensemble import RandomForestClassifier
          rfc = RandomForestClassifier(n_estimators=600)
          from sklearn.model_selection import train_test_split

In [167]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

In [168]: import pandas
          from sklearn import model_selection
          from sklearn.ensemble import GradientBoostingClassifier

In [169]: ##gradient boosting
          X = X_train
          Y = y_train
          seed = 7
          num_trees = 200
          kfold = model_selection.KFold(n_splits=10, random_state=seed)
          model = GradientBoostingClassifier(n_estimators=num_trees, random_state=seed)
          results = model_selection.cross_val_score(model, X, Y, cv=kfold)
          print(results.mean())

          0.827028571429
```

BEST MODEL:

Random Forests:

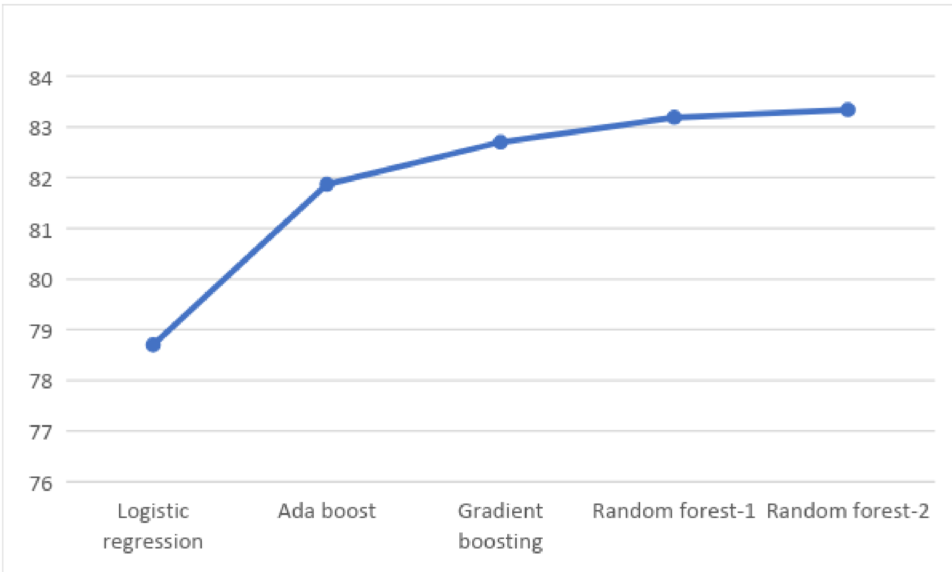
Removed the interaction variables to give best accuracy. 600 trees were generated and model was fitted using default cut off of 0.5. The accuracy obtained was 83.34%

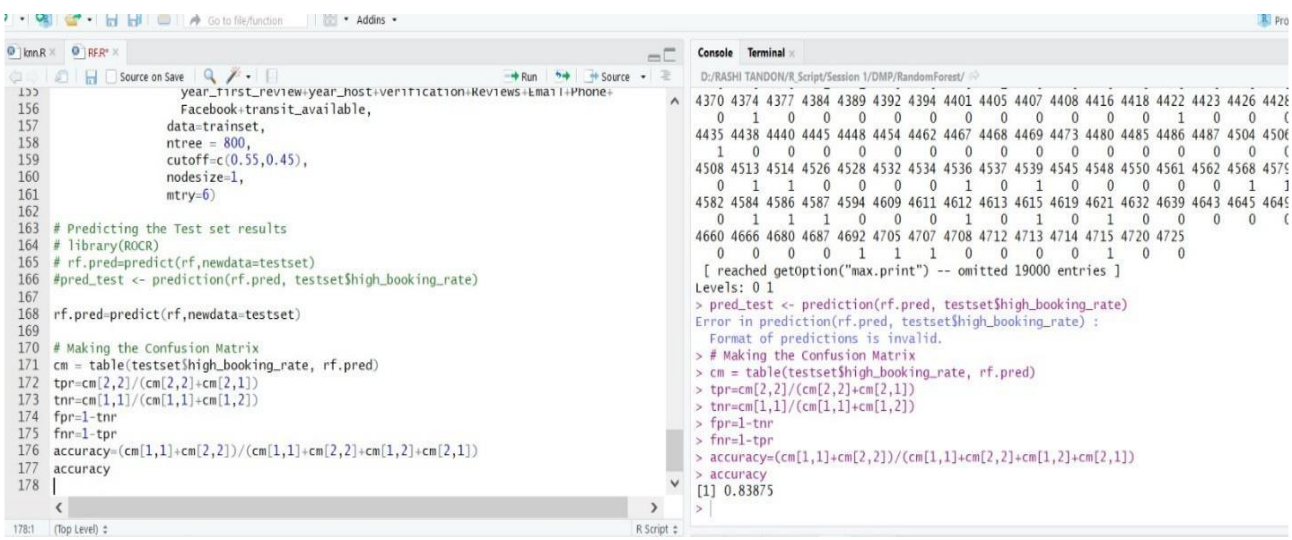
```
In [182]: X = data[['accommodates', 'availability_30', 'availability_90', 'availability_365','bathrooms','bed_type',
                      'cleaning_fee', 'Wifi', 'parking', 'Kitchen', '24-hour-check', 'Breakfast', 'ac', 'Heating', 'property_type', 'price
                      'bedrooms','security_deposit','state', 'cancellation_policy', 'extra_people', 'guests_included',
                  'host_has_profile_pic', 'host_identity_verified', 'host_is_superhost', 'host_response_rate', 'host_response_time',
                 'host_total_listings_count', 'maximum_nights', 'minimum_nights', 'instant_bookable', 'is_business_travel_ready',
                  'is_location_exact', 'room_type', 'city', 'monthly_price','year_host','year_first_review',
                   'require_guest_phone_verification', 'Facebook', 'Phone', 'Email', 'verification',
                   'house_rules', 'weekly_price','transit_available', 'hair_dryer','hangers','iron','laptop','shampoo','essentials',
                  'washer','dryer','tv','pool','gym','hot tub','wireless','internet','family/kid friendly','elevator']]
          y = train_y['high_booking_rate']

In [183]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

In [184]: from sklearn.ensemble import RandomForestClassifier
          rfc = RandomForestClassifier(n_estimators=600)
          from sklearn.model_selection import train_test_split

In [185]: rfc.fit(X_train,y_train)

Out[185]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      min_impurity_split=1e-07, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=600, n_jobs=1, oob_score=False, random_state=None,
                      verbose=0, warm_start=False)

In [186]: predictions = rfc.predict(X_test)

In [187]: from sklearn.metrics import accuracy_score
          accuracy_score(y_test, predictions)

Out[187]: 0.83340000000000003
```

5. PERFORMANCE:

Best model was chosen based on the accuracy and its performance on the holdout data (validation data).

By changing the cut off accuracy changed, this was tested once in R, by setting the cut off 0.55 for 1 and 0.45 for 0. This relatively improved the accuracy to 83.875%.





APPENDIX A – ROLES AND RESPONSIBILITIES:

Feature engineering – data preprocessing:

(possible overlap of 2 or more people working on same variable)

Kiruthika – 18 variables, Rohini – 14 variables, Rashi – 10 variables, Sonalika – 7 variables

Sragdhara – 5 variables

Feature Selection:

Kiruthika – Lasso regression, Recursive feature elimination, Trees, Identification of interaction.

Rohini – handled interaction variables while modelling

Rashi – Binning of amenities to different categories

Model:

Trial 1:

Kiruthika, Rashi – Logistic regression with 78% accuracy

Rohini, Rashi, Kiruthika, Sonalika – Consolidation of final cleaned data file.

Trial 2:

Kiruthika – Random forest with 82.08% accuracy

Rohini– Ada boosting 81.5% accuracy

Rashi – KNN classifier

Trial 3:

Kiruthika, Rohini – Random forest with 82.19% accuracy

Trial 4:

Kiruthika – Random forest with 83.34% accuracy

Rohini – Gradient boosting with 82.7% accuracy, ada boosting with 81.87%

Trial 5: (done for testing purpose to check if there is boosting in performance after report submission)

Rohini - Tuned input parameters and got enhanced accuracy using boosting algorithms

-Number of trees and learning factor for Ada Boosting- Best accuracy 82.19%

-Number of trees for gradient boosting- Best accuracy 83.66%

Rashi, Kiruthika – Random forest with 83.87% by changing cutoff in R

APPENDIX B: MODEL TUNING
ADA Boosting- Tuning Parameters

Model	n_estimators (no. of trees)	Learning factor	Accuracy
Model 1	300	1	80.27
Model 2	400	1	80.17
Model 3	600	1	81.25
Model 4	300	0.5	81.14
Model 5	400	0.5	81.35
Model 6	600	0.5	81.50
Model 7	300	0.3	81.86
Model 8	400	0.3	81.87
Model 9	600	0.3	82.03
Model 10	300	0.2	81.64
Model 11	**400**	**0.2**	**82.19**
Model 12	600	0.2	80.76

Gradient Boosting:

Model Tuning: By varying the number of trees for the given parameters, following change in accuracies was observed:

Model	n_estimators (no. of trees)	Accuracy
Model 1	300	83.13
Model 2	400	83.31
Model 3	500	83.50
Model 4	600	83.62
Model 5	**700**	**83.66**

DATA PREPROCESSING

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
% matplotlib inline
```

In [2]:
```
train_x =pd.read_csv("airbnb_train_x.csv")
```

```
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2
717: DtypeWarning: Columns (2,4,6,7,8,10,11,28,35,43,45,69) have mixed types.
Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [3]:
```
train_x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 70 columns):
Unnamed: 0                     100000 non-null int64
access                         65564 non-null object
accommodates                   100000 non-null object
amenities                      100000 non-null object
availability_30                100000 non-null object
availability_365               100000 non-null int64
availability_60                100000 non-null object
availability_90                100000 non-null object
bathrooms                      99754 non-null object
bed_type                       99999 non-null object
bedrooms                       99907 non-null object
beds                           99917 non-null object
cancellation_policy            100000 non-null object
city                           99951 non-null object
city_name                      99999 non-null object
cleaning_fee                   81676 non-null object
country                        99999 non-null object
country_code                   99993 non-null object
description                    99973 non-null object
experiences_offered            99997 non-null object
extra_people                   100000 non-null object
first_review                   99992 non-null object
guests_included                100000 non-null float64
host_about                     69188 non-null object
host_acceptance_rate           8242 non-null object
host_has_profile_pic           99858 non-null object
host_identity_verified         99852 non-null object
host_is_superhost              99858 non-null object
host_listings_count            99855 non-null object
host_location                  99589 non-null object
host_name                      99853 non-null object
host_neighbourhood             82767 non-null object
host_response_rate             84206 non-null object
host_response_time             84206 non-null object
host_since                     99858 non-null object
host_total_listings_count      99858 non-null object
host_verifications             99990 non-null object
house_rules                    69607 non-null object
instant_bookable               99988 non-null object
interaction                    64120 non-null object
is_business_travel_ready       55449 non-null object
is_location_exact              99981 non-null object
jurisdiction_names             47310 non-null object
latitude                       99990 non-null object
license                        14279 non-null object
longitude                      99989 non-null object
market                         99567 non-null object
maximum_nights                 99990 non-null float64
minimum_nights                 99990 non-null float64
monthly_price                  20564 non-null object
name                           99967 non-null object
neighborhood_overview          69257 non-null object
neighbourhood                  86447 non-null object
notes                          51467 non-null object
```

```
price                              99981 non-null object
property_type                      99980 non-null object
require_guest_phone_verification   99981 non-null object
require_guest_profile_picture      99981 non-null object
requires_license                   99981 non-null object
room_type                          99981 non-null object
security_deposit                   59352 non-null object
smart_location                     99980 non-null object
space                              78955 non-null object
square_feet                        1581 non-null float64
state                              99978 non-null object
street                             99981 non-null object
summary                            97036 non-null object
transit                            70833 non-null object
weekly_price                       24105 non-null object
zipcode                            98951 non-null object
dtypes: float64(4), int64(2), object(64)
memory usage: 53.4+ MB
```

In [4]:

```
#variable-2-not considered
```

In [5]:

```
#variable-3-accommodates
train_x.accommodates.isnull().value_counts()
train_x.accommodates = pd.to_numeric(train_x['accommodates'], errors='coerce')
train_x.accommodates.isnull().value_counts()
train_x.accommodates.fillna(train_x.accommodates.mode()[0], inplace = True)
train_x.accommodates.isnull().value_counts()
```

Out[5]:

```
False    100000
Name: accommodates, dtype: int64
```

In [6]:

```
train_x['amenities'] = train_x['amenities'].str.lower()
```

In [7]:

```
#variable-4-amenities
train_x['Wifi'] = np.where(train_x['amenities'].str.contains('wifi'), 'yes',
'no')
train_x['parking'] = np.where(train_x['amenities'].str.contains('parking'), 'y
es', 'no')
train_x['Kitchen'] = np.where(train_x['amenities'].str.contains('kitchen'), 'y
es', 'no')
train_x['24-hour-check'] = np.where(train_x['amenities'].str.contains('24-hour
 check-in'), 'yes', 'no')
train_x['Breakfast'] = np.where(train_x['amenities'].str.contains('breakfast'
), 'yes', 'no')
train_x['ac'] = np.where(train_x['amenities'].str.contains('air conditioning'
), 'yes', 'no')
train_x['Heating'] = np.where(train_x['amenities'].str.contains('heating'), 'y
es', 'no')
```

In [8]:
```
train_x['hair_dryer'] = np.where(train_x['amenities'].str.contains('hair drye
r'), 'yes', 'no')
train_x['hangers'] = np.where(train_x['amenities'].str.contains('hangers'), 'y
es', 'no')
train_x['iron'] = np.where(train_x['amenities'].str.contains('iron'), 'yes',
'no')
train_x['laptop'] = np.where(train_x['amenities'].str.contains('laptop friendl
y workspace'), 'yes', 'no')
train_x['shampoo'] = np.where(train_x['amenities'].str.contains('shampoo'), 'y
es', 'no')
train_x['essentials'] = np.where(train_x['amenities'].str.contains('essential
s'), 'yes', 'no')
train_x['washer'] = np.where(train_x['amenities'].str.contains('washer'), 'ye
s', 'no')
train_x['dryer'] = np.where(train_x['amenities'].str.contains('dryer'), 'yes',
 'no')
train_x['tv'] = np.where(train_x['amenities'].str.contains('tv'), 'yes', 'no')
train_x['pool'] = np.where(train_x['amenities'].str.contains('pool'), 'yes',
'no')
train_x['gym'] = np.where(train_x['amenities'].str.contains('gym'), 'yes', 'n
o')
train_x['hot tub'] = np.where(train_x['amenities'].str.contains('hot tub'), 'y
es', 'no')
train_x['cable'] = np.where(train_x['amenities'].str.contains('cable'), 'yes',
 'no')
train_x['wireless'] = np.where(train_x['amenities'].str.contains('wireless'),
'yes', 'no')
train_x['internet'] = np.where(train_x['amenities'].str.contains('internet'),
'yes', 'no')
train_x['family/kid friendly'] = np.where(train_x['amenities'].str.contains('f
amily/kid friendly'), 'yes', 'no')
train_x['elevator'] = np.where(train_x['amenities'].str.contains('elevator'),
'yes', 'no')
```

In [9]:
```
#variable-5-availability_30
train_x.availability_30 = pd.to_numeric(train_x['availability_30'], errors='co
erce')
train_x.availability_30.fillna(train_x.availability_30.mode()[0], inplace = Tr
ue)
train_x.availability_30.isnull().value_counts()
```

Out[9]:
```
False    100000
Name: availability_30, dtype: int64
```

In [10]:
```
#variable-6-availability-365-completely clean
```

In [11]:
```
#variable-7-availability_60
train_x.availability_60 = pd.to_numeric(train_x['availability_60'], errors='co
erce')
train_x.availability_60.fillna(train_x.availability_60.mode()[0], inplace = Tr
ue)
train_x.availability_60.isnull().value_counts()
```

Out[11]:
```
False    100000
Name: availability_60, dtype: int64
```

In [12]:
```
#variable-8-availability_90
train_x.availability_90 = pd.to_numeric(train_x['availability_90'], errors='co
erce')
train_x.availability_90.fillna(train_x.availability_90.mode()[0], inplace = Tr
ue)
train_x.availability_90.isnull().value_counts()
```

Out[12]:
```
False    100000
Name: availability_90, dtype: int64
```

In [13]:
```
#variable-9-bathroom
train_x.bathrooms = pd.to_numeric(train_x['bathrooms'], errors='coerce')
train_x.bathrooms.fillna(1, inplace = True)
train_x.bathrooms.isnull().value_counts()
```

Out[13]:
```
False    100000
Name: bathrooms, dtype: int64
```

In [14]:
```
#variable-10-bed_type
train_x.bed_type = train_x.bed_type.str.replace('(\d+)%','Real Bed')
train_x.bed_type.fillna('Real Bed', inplace = True)
train_x.bed_type.isnull().value_counts()
```

Out[14]:
```
False    100000
Name: bed_type, dtype: int64
```

In [15]:
```
#variable-11-bedrooms
train_x.bedrooms = pd.to_numeric(train_x['bedrooms'], errors='coerce')
train_x.bedrooms.fillna(1, inplace = True)
train_x.bedrooms.isnull().value_counts()
```

Out[15]:
```
False    100000
Name: bedrooms, dtype: int64
```

In [16]:
```
#variable-12-beds
train_x.beds = pd.to_numeric(train_x['beds'], errors='coerce')
train_x.beds.fillna(1, inplace = True)
train_x.beds.isnull().value_counts()
```

Out[16]:
```
False    100000
Name: beds, dtype: int64
```

In [17]:
```
#variable-13-cancellation-policy
train_x['cancellation_policy'] = train_x['cancellation_policy'].str.replace
('super_strict_30', 'very_strict')
train_x['cancellation_policy'] = train_x['cancellation_policy'].str.replace
('super_strict_60', 'very_strict')
train_x['cancellation_policy'] = train_x['cancellation_policy'].str.replace
('5', 'strict')
train_x['cancellation_policy'] = train_x['cancellation_policy'].str.replace
('1', 'strict')
train_x['cancellation_policy'] = train_x['cancellation_policy'].str.replace
('2', 'strict')
train_x['cancellation_policy'] = train_x['cancellation_policy'].str.replace
('no_refunds', 'strict')
train_x['cancellation_policy'] = train_x['cancellation_policy'].str.replace
('strict.0', 'strict')
train_x.cancellation_policy.value_counts()
```

Out[17]:
```
strict         47385
moderate       30411
flexible       21837
very_strict      367
Name: cancellation_policy, dtype: int64
```

In [18]:
```
#variable-14-cleaning_fee
train_x.cleaning_fee = train_x.cleaning_fee.str.replace('$','')
train_x.cleaning_fee = pd.to_numeric(train_x['cleaning_fee'], errors='coerce')
train_x.cleaning_fee.fillna(train_x.cleaning_fee.mode()[0], inplace = True)
train_x.cleaning_fee.isnull().value_counts()
```

Out[18]:
```
False    100000
Name: cleaning_fee, dtype: int64
```

In [19]:
```
#variable-15-21- city,city_name,county,country,country_code,latitude,longitude

#variable-22-23 - description, experiences_offered
```

In [20]:
```
extra = pd.read_csv("city-room-street.csv")
```

In [21]:
```
#variable- 16-city_name
extra['city'] = extra['city'].str.replace('Washington, D.C.', 'Washington DC')
train_x.city = extra.city
```

In [22]:
```
#variable-24- extra_people
extra.extra_people = extra.extra_people.str.replace('$','')
extra.extra_people.value_counts()
train_x.extra_people = extra.extra_people
```

In [23]:
```
#variable-25-first_review
train_x['date_first_review'] = pd.to_datetime(train_x['first_review'],errors=
'coerce')
train_x['year_first_review'] = pd.DatetimeIndex(train_x['date_first_review']).
year
train_x.year_first_review = pd.to_numeric(train_x['year_first_review'], errors
='coerce')
train_x.year_first_review.fillna(train_x.year_first_review.mode()[0], inplace
= True)
train_x.year_first_review.value_counts()
```

Out[23]:
```
2016.0    31378
2017.0    26571
2015.0    21178
2014.0     9424
2013.0     4560
2018.0     2894
2012.0     2457
2011.0     1092
2010.0      362
2009.0       80
2008.0        4
Name: year_first_review, dtype: int64
```

In [24]:
```
#variable-26-guests_included
train_x['guests_included'] = np.where(train_x['guests_included'] <0, train_x.h
ost_has_profile_pic,train_x['guests_included'])
train_x.guests_included = pd.to_numeric(train_x['guests_included'], errors='co
erce')
train_x.guests_included.value_counts()
```

Out[24]:
```
1.0     61790
2.0     21905
4.0      7427
3.0      2898
6.0      2573
5.0      1222
8.0       899
10.0      365
0.0       356
7.0       276
12.0      103
9.0        67
16.0       30
14.0       29
15.0       23
11.0       20
13.0       14
22.0        1
18.0        1
20.0        1
Name: guests_included, dtype: int64
```

In [25]:
```
#variable-29-host_has_profile_pic
train_x.host_has_profile_pic.fillna('t', inplace = True)
train_x['host_has_profile_pic'] = np.where(~train_x['host_has_profile_pic'].as
type(str).str.contains('f'), 't', 'f')
train_x.host_has_profile_pic.value_counts()
```

Out[25]:
```
t    99828
f      172
Name: host_has_profile_pic, dtype: int64
```

In [26]:
```
#variable-30-host_identity_verified
train_x.host_identity_verified.value_counts()
train_x.host_identity_verified.fillna('t', inplace = True)
train_x['host_identity_verified'] = np.where(~train_x['host_identity_verified'
].astype(str).str.contains('f'), 't', 'f')
train_x.host_identity_verified.value_counts()
```

Out[26]:
```
t    70041
f    29959
Name: host_identity_verified, dtype: int64
```

In [27]:
```
#variable-31-host_is_superhost
train_x.host_is_superhost.fillna(0, inplace = True)
train_x.host_is_superhost = train_x.host_is_superhost.str.replace('f','0')
train_x.host_is_superhost = train_x.host_is_superhost.str.replace('t','1')
train_x.host_is_superhost = pd.to_numeric(train_x['host_is_superhost'], errors
='coerce')
train_x.host_is_superhost.fillna(0, inplace = True)
train_x.host_is_superhost.value_counts()
```

Out[27]:
```
0.0    73870
1.0    26130
Name: host_is_superhost, dtype: int64
```

In [28]:
```
#variable-33-host_location
train_x.host_location.value_counts()
```

```
Out[28]: New York, New York, United States
                                                              25669
         Los Angeles, California, United States
                                                              13237
         US
                                                               7988
         Austin, Texas, United States
                                                               4572
         Washington, District of Columbia, United States
                                                               3978
         San Francisco, California, United States
                                                               3803
         Chicago, Illinois, United States
                                                               3535
         San Diego, California, United States
                                                               3419
         Nashville, Tennessee, United States
                                                               3320
         Portland, Oregon, United States
                                                               3314
         New Orleans, Louisiana, United States
                                                               3033
         Seattle, Washington, United States
                                                               2587
         Denver, Colorado, United States
                                                               2520
         Boston, Massachusetts, United States
                                                               2502
         Brooklyn, New York, United States
                                                               1245
         Oakland, California, United States
                                                                926
         Santa Monica, California, United States
                                                                628
         Asheville, North Carolina, United States
                                                                573
         Long Beach, California, United States
                                                                495
         United States
                                                                403
         Beverly Hills, California, United States
                                                                390
         West Hollywood, California, United States
                                                                366
         Pasadena, California, United States
                                                                347
         Queens, New York, United States
                                                                338
         Santa Cruz, California, United States
                                                                329
         Spokane, Washington, United States
                                                                288
         California, United States
                                                                238
         Malibu, California, United States
                                                                207
         Montreal, Quebec, Canada
```

```
                                                                    188
Marina del Rey, California, United States
                                                                    153

                                                                  ...
20 year native to Harlem NYC
                                                                      1
U.S.
                                                                      1
Seattle WA and Portland OR
                                                                      1
Antioquia, Colombia
                                                                      1
Oregon City, Oregon, United States
                                                                      1
Newtown, Connecticut, United States
                                                                      1
West Hempstead, New York, United States
                                                                      1
usa
                                                                      1
Brazil
                                                                      1
Del Rey
                                                                      1
Los Altos Hills, California, United States
                                                                      1
California, US
                                                                      1
Maroubra, New South Wales, Australia
                                                                      1
Fredericksburg, Virginia, United States
                                                                      1
Orléans, Centre-Val de Loire, France
                                                                      1
Greenville, South Carolina, United States
                                                                      1
My partner and I stay either at the Airbnb house - or at his place in Nort
h Austin.   When the house is rented, we stay at his place.           1
Great Falls, Virginia, United States
                                                                      1
Eureka, California, United States
                                                                      1
Greenport, New York, United States
                                                                      1
Alabama
                                                                      1
Durango, Durango, Mexico
                                                                      1
Jericho, Vermont, United States
                                                                      1
El Cerrito, California, United States
                                                                      1
Flower Mound, Texas, United States
                                                                      1
Norristown, Pennsylvania, United States
                                                                      1
```

```
Wellington, Wellington, New Zealand
                                                                 1
Lima Region, Peru
                                                                 1
London
                                                                 1
Bridgewater, New Jersey, United States
                                                                 1
Name: host_location, Length: 2154, dtype: int64
```

In [29]:
```
#variable-36-host_response_rate
train_x.host_response_rate.value_counts()
train_x.host_response_rate = train_x.host_response_rate.str.replace('%','')
train_x.host_response_rate = pd.to_numeric(train_x['host_response_rate'], erro
rs='coerce')
train_x.host_response_rate.fillna(100, inplace = True)
train_x.host_response_rate.value_counts()
```

```
Out[29]: 100.0    82348
         90.0      2877
         80.0      1333
         99.0      1090
         98.0       844
         50.0       765
         97.0       707
         0.0        681
         96.0       659
         94.0       622
         95.0       616
         92.0       555
         67.0       538
         70.0       533
         88.0       524
         93.0       519
         89.0       445
         83.0       422
         75.0       420
         86.0       398
         60.0       395
         91.0       344
         33.0       177
         78.0       177
         87.0       153
         71.0       153
         40.0       149
         85.0       131
         81.0       124
         82.0       109
                  ...
         44.0        22
         17.0        21
         58.0        20
         53.0        17
         46.0        15
         10.0        14
         38.0        12
         55.0        12
         48.0        11
         59.0        10
         29.0        10
         62.0         9
         52.0         7
         14.0         7
         61.0         7
         27.0         6
         13.0         6
         36.0         6
         22.0         5
         42.0         5
         11.0         4
         41.0         3
         47.0         3
         31.0         3
         4.0          2
         18.0         2
```

```
51.0         1
21.0         1
35.0         1
66.0         1
Name: host_response_rate, Length: 79, dtype: int64
```

In [30]:
```
#variable-37-host_response_time
train_x.host_response_time.fillna('within an hour', inplace = True)
train_x.host_response_time = train_x.host_response_time.str.replace('within an
 hour','0')
train_x.host_response_time = train_x.host_response_time.str.replace('within a
 few hours','1')
train_x.host_response_time = train_x.host_response_time.str.replace('within a
 day','2')
train_x.host_response_time = train_x.host_response_time.str.replace('a few day
s or more','3')
train_x.host_response_time = pd.to_numeric(train_x['host_response_time'], erro
rs='coerce')
train_x.host_response_time.fillna(0, inplace = True)
train_x.host_response_time.value_counts()
```

Out[30]:
```
0.0    70676
1.0    17262
2.0    10657
3.0     1405
Name: host_response_time, dtype: int64
```

In [31]:
```
#variable-38-host_since
train_x['host_since'] = pd.to_datetime(train_x['host_since'],errors='coerce')
train_x['year_host'] = pd.DatetimeIndex(train_x['host_since']).year
train_x.year_host = pd.to_numeric(train_x['year_host'], errors='coerce')
train_x.year_host.fillna(train_x.year_host.mode()[0], inplace = True)
train_x.year_host.value_counts()
```

Out[31]:
```
2015.0    20666
2014.0    18985
2016.0    16144
2013.0    15143
2012.0    11557
2017.0     6984
2011.0     6719
2010.0     2391
2009.0      959
2018.0      314
2008.0      138
Name: year_host, dtype: int64
```

In [32]:
```
#variable-39-host_total_listings_count
train_x.host_total_listings_count = pd.to_numeric(train_x['host_total_listings
_count'], errors='coerce')
train_x.host_total_listings_count.fillna(1, inplace = True)
```

In [33]:
```
#variable-40-host_verifications
train_x['verification'] = np.where(train_x['host_verifications'].str.contains(
    'email' or 'phone' or 'google' or 'reviews' or 'jumio' or 'kba' or 'work_e
mail' or
    'facebook' or 'linkedin' or 'selfie' or 'identity_manual' or 'government_i
d' or
    'amex' or 'offline_government_id'
    or 'sent_id' or 'photographer' or 'sesame' or 'sesame_offline' or 'weibo'
), 'yes', 'no')
```

In [34]:
```
train_x['host_verifications'] = train_x['host_verifications'].str.lower()
```

In [35]:
```
train_x['host_verifications'].value_counts()
```

```
Out[35]: ['email', 'phone', 'reviews', 'kba']
                                       17422
         ['email', 'phone', 'reviews']
                                       15070
         ['email', 'phone', 'reviews', 'jumio']
                                       12492
         ['email', 'phone', 'facebook', 'reviews', 'kba']
                                        7294
         ['email', 'phone', 'facebook', 'reviews', 'jumio']
                                        4214
         ['email', 'phone', 'facebook', 'reviews']
                                        3762
         ['email', 'phone', 'reviews', 'kba', 'work_email']
                                        2861
         ['email', 'phone', 'reviews', 'jumio', 'government_id']
                                        2184
         ['email', 'phone', 'reviews', 'jumio', 'offline_government_id', 'selfie',
         'government_id', 'identity_manual']
                                        2126
         ['email', 'phone', 'reviews', 'jumio', 'work_email']
                                        1998
         ['email', 'phone', 'reviews', 'jumio', 'offline_government_id', 'governmen
         t_id']
                                        1831
         ['email', 'phone', 'facebook', 'reviews', 'kba', 'work_email']
                                        1523
         ['email', 'phone', 'facebook', 'reviews', 'jumio', 'government_id']
                                        1461
         ['phone', 'reviews']
                                        1442
         ['email', 'phone', 'google', 'reviews', 'kba']
                                        1206
         ['email', 'phone', 'facebook', 'reviews', 'jumio', 'offline_government_i
         d', 'government_id']
                                          1003
         ['email', 'phone', 'facebook', 'reviews', 'jumio', 'work_email']
                                         905
         ['email', 'phone', 'reviews', 'work_email']
                                         825
         ['email', 'phone', 'reviews', 'jumio', 'kba']
                                         824
```

```
['email', 'phone', 'google', 'reviews', 'jumio', 'government_id']

                       801
['email', 'phone', 'facebook', 'reviews', 'jumio', 'offline_government_i
d', 'selfie', 'government_id', 'identity_manual']
                         658
['email', 'phone', 'linkedin', 'reviews', 'kba']

                       584
['email', 'phone', 'google', 'reviews', 'jumio']

                       584
['email', 'phone', 'reviews', 'manual_offline', 'jumio']

                       476
['email', 'phone', 'facebook', 'google', 'reviews', 'kba']

                       429
['email', 'phone', 'google', 'reviews', 'jumio', 'offline_government_id',
'government_id']
                       428
['phone', 'facebook', 'reviews']

                       424
['email', 'phone']

                       382
['email', 'phone', 'google', 'reviews']

                       375
['email', 'phone', 'reviews', 'jumio', 'government_id', 'work_email']

                       368



                     ...
['email', 'phone', 'facebook', 'google', 'linkedin', 'amex', 'reviews', 'j
umio', 'offline_government_id', 'kba', 'government_id']
                         1
['email', 'phone', 'linkedin', 'reviews', 'jumio', 'selfie', 'government_i
d']
                         1
['email', 'phone', 'manual_online', 'linkedin', 'reviews', 'manual_offlin
e', 'jumio', 'offline_government_id', 'selfie', 'government_id', 'identity
_manual']                 1
['email', 'phone', 'google', 'linkedin', 'jumio', 'offline_government_id',
'selfie', 'government_id', 'identity_manual']
                       1
['email', 'phone', 'facebook', 'google', 'amex', 'reviews', 'work_email']

                         1
['phone', 'facebook', 'reviews', 'jumio', 'selfie', 'government_id', 'iden
tity_manual']
                         1
['email', 'manual_online', 'facebook', 'reviews', 'manual_offline', 'sent_
id', 'kba']
                         1
```

```
['email', 'phone', 'facebook', 'amex', 'reviews']

                         1
['email', 'phone', 'facebook', 'google', 'amex', 'reviews', 'jumio', 'kb
a', 'government_id', 'work_email']
                           1
['phone', 'linkedin', 'reviews', 'kba', 'work_email']

                         1
['email', 'phone', 'facebook', 'reviews', 'manual_offline', 'kba']

                         1
['email', 'phone', 'google', 'linkedin', 'reviews', 'sesame', 'sesame_offl
ine']
                         1
97.0

                         1
['email', 'phone', 'facebook', 'amex', 'reviews', 'jumio', 'kba', 'governm
ent_id', 'work_email']
                         1
['email', 'phone', 'facebook', 'google', 'linkedin', 'amex', 'reviews', 'j
umio', 'selfie', 'government_id', 'identity_manual', 'work_email']
                         1
['phone', 'reviews', 'jumio', 'offline_government_id', 'kba', 'government_
id']
                         1
['email', 'phone', 'facebook', 'google', 'linkedin', 'amex', 'reviews', 'j
umio', 'offline_government_id', 'selfie', 'government_id', 'identity_manua
l', 'work_email']        1
['reviews', 'jumio', 'government_id']

                         1
['phone', 'facebook', 'linkedin', 'reviews', 'kba']

                         1
['email', 'phone', 'google', 'linkedin', 'reviews', 'jumio', 'selfie', 'go
vernment_id', 'identity_manual', 'work_email']
                         1
['email', 'phone', 'manual_online', 'facebook', 'reviews', 'manual_offlin
e', 'jumio', 'offline_government_id', 'government_id', 'work_email']
                          1
['email', 'phone', 'facebook', 'google', 'linkedin', 'amex', 'reviews', 'j
umio', 'offline_government_id', 'government_id', 'work_email']
                         1
['email', 'phone', 'linkedin', 'amex', 'reviews', 'jumio', 'kba', 'work_em
ail']
                         1
['email', 'phone', 'manual_online', 'reviews', 'manual_offline', 'jumio',
'offline_government_id', 'kba', 'government_id', 'work_email']
                         1
['email', 'phone', 'facebook', 'jumio', 'offline_government_id', 'selfie',
'government_id', 'identity_manual', 'work_email']
                        1
['phone', 'facebook', 'jumio', 'offline_government_id', 'government_id']

                         1
```

```
['email', 'phone', 'amex', 'reviews', 'jumio', 'government_id', 'work_emai
l']
                                1
['email', 'phone', 'facebook', 'google', 'linkedin', 'reviews', 'sent_id']

                                1
['email', 'phone', 'facebook', 'reviews', 'jumio', 'government_id', 'ident
ity_manual']
                                1
['email', 'phone', 'google', 'reviews', 'jumio', 'kba', 'selfie', 'governm
ent_id', 'identity_manual', 'work_email']
                                1
Name: host_verifications, Length: 729, dtype: int64
```

In [36]:

```
#variable-listing verifications
train_x['Reviews'] = np.where(train_x['host_verifications'].str.contains('revi
ew'), 'yes', 'no')
train_x['Email'] = np.where(train_x['host_verifications'].str.contains('email'
), 'yes', 'no')
train_x['Phone'] = np.where(train_x['host_verifications'].str.contains('phone'
), 'yes', 'no')
train_x['Facebook'] = np.where(train_x['host_verifications'].str.contains('fac
ebook'), 'yes', 'no')
```

In [37]:

```
#variable-41-house_rules
train_x.house_rules.fillna('f', inplace = True)
train_x.house_rules = train_x.house_rules.str.replace('f','0')
train_x.house_rules = pd.to_numeric(train_x['house_rules'], errors='coerce')
train_x.house_rules.fillna(1, inplace = True)
train_x.house_rules.value_counts()
```

Out[37]:

```
1.0    69607
0.0    30393
Name: house_rules, dtype: int64
```

In [38]:

```
#variable-42-instant_bookable
train_x.instant_bookable.fillna('f', inplace = True)
train_x.instant_bookable = train_x.instant_bookable.str.replace('f','0')
train_x.instant_bookable = train_x.instant_bookable.str.replace('t','1')
train_x.instant_bookable = pd.to_numeric(train_x['instant_bookable'], errors=
'coerce')
train_x.instant_bookable.fillna(0, inplace = True)
```

In [39]:

```
#variable-44-is_business_travel_ready
train_x.is_business_travel_ready.value_counts()
train_x.is_business_travel_ready = train_x.is_business_travel_ready.str.replac
e('f','0')
train_x.is_business_travel_ready = train_x.is_business_travel_ready.str.replac
e('t','1')
train_x.is_business_travel_ready = pd.to_numeric(train_x['is_business_travel_r
eady'], errors='coerce')
train_x.is_business_travel_ready.fillna(0, inplace = True)
```

In [40]:
```
#variable-45-is_location_exact
train_x.is_location_exact.value_counts()
train_x.is_location_exact.fillna('t', inplace = True)
train_x.is_location_exact.value_counts()
```

Out[40]:
```
t    83933
f    16067
Name: is_location_exact, dtype: int64
```

In [41]:
```
#variable-51-maximum nights
train_x.maximum_nights = pd.to_numeric(train_x['maximum_nights'], errors='coerce')
train_x.maximum_nights.fillna(1125, inplace = True)
train_x['maximum_nights'] = np.where(train_x['maximum_nights'] >1125,1125,train_x['maximum_nights'])
```

In [42]:
```
#variable-52-minimum nights
train_x.minimum_nights = pd.to_numeric(train_x['minimum_nights'], errors='coerce')
train_x.minimum_nights.fillna(1, inplace = True)
```

In [43]:
```
#variable-53-monthly_price
train_x.monthly_price = train_x.monthly_price.str.replace('$','')
train_x.monthly_price = pd.to_numeric(train_x['monthly_price'], errors='coerce')
train_x.monthly_price.fillna('f', inplace = True)
train_x['monthly_price'] = np.where(~train_x['monthly_price'].astype(str).str.contains('f'), 't', 'f')
train_x.monthly_price.value_counts()
```

Out[43]:
```
f    98641
t     1359
Name: monthly_price, dtype: int64
```

In [44]:
```
#variable-58-price
train_x.price = train_x.price.str.replace('$','')
train_x.price = pd.to_numeric(train_x['price'], errors='coerce')
train_x.price.fillna(train_x.price.mode()[0], inplace = True)
train_x.price.isnull().value_counts()
```

Out[44]:
```
False    100000
Name: price, dtype: int64
```

In [45]:
```
#variable-60-require_guest_phone_verification
train_x.require_guest_phone_verification.fillna('f', inplace = True)
train_x.require_guest_phone_verification.isnull().value_counts()
train_x.require_guest_phone_verification.value_counts()
```

Out[45]:
```
f    95445
t     4555
Name: require_guest_phone_verification, dtype: int64
```

In [46]:
```
#variable-61-require_guest_profile_picture
train_x.require_guest_profile_picture.fillna('f', inplace = True)
train_x.require_guest_profile_picture.isnull().value_counts()
train_x.require_guest_profile_picture.value_counts()
```

Out[46]:
```
f    96521
t     3479
Name: require_guest_profile_picture, dtype: int64
```

In [47]:
```
#variable-62-requires_license
train_x.requires_license.fillna('f', inplace = True)
train_x.requires_license.isnull().value_counts()
train_x.requires_license.value_counts()
```

Out[47]:
```
f    80534
t    19466
Name: requires_license, dtype: int64
```

In [48]:
```
#variable-64-security_deposit
train_x.security_deposit = train_x.security_deposit.str.replace('$','')
train_x.security_deposit = pd.to_numeric(train_x['security_deposit'], errors=
'coerce')
train_x.security_deposit.fillna('f', inplace = True)
train_x['security_deposit'] = np.where(~train_x['security_deposit'].astype(str
).str.contains('f'), 't', 'f')
train_x.security_deposit.value_counts()
```

Out[48]:
```
t    55712
f    44288
Name: security_deposit, dtype: int64
```

In [49]:
```
state = pd.read_csv('state-property_type.csv')
```

In [50]:
```
#variable-59 - state and variable 68 - property type
train_x.state = state.state
train_x.property_type = state.property_type
```

In [51]:
```
train_x['property_type'] = np.where(train_x['property_type'].astype(str).str.c
ontains('bed & breakfast'),
                                 'bed and breakfast', train_x['property_type'
])
```

In [52]:
```
train_x.property_type.value_counts()
```

Out[52]:
```
apartment                 54709
house                     31447
condominium                3701
townhouse                  2776
loft                       1866
guesthouse                  891
other                       829
bed and breakfast           696
bungalow                    653
guest suite                 567
cabin                       225
dorm                        215
villa                       213
camper/rv                   162
serviced apartment          162
in-law                      160
boutique hotel              153
hostel                      139
boat                         88
timeshare                    72
vacation home                50
tent                         45
resort                       30
treehouse                    28
castle                       18
yurt                         14
cottage                      14
hotel                        12
hut                          12
chalet                       10
earth house                  10
tipi                          5
aparthotel                    5
tiny house                    5
cave                          5
island                        3
train                         3
farm stay                     2
barn                          1
casa particular (cuba)        1
lighthouse                    1
plane                         1
nature lodge                  1
Name: property_type, dtype: int64
```

In [53]:
```
#variable-63-room_type
train_x.room_type.value_counts()
train_x.room_type.fillna('Entire home/apt', inplace = True)
train_x.room_type.value_counts()
```

Out[53]:
```
Entire home/apt    60911
Private room       36617
Shared room         2472
Name: room_type, dtype: int64
```

In [54]:
```
train_x['transit'] = train_x['transit'].str.lower()
```

In [55]:
```
#variable-71-transit
train_x['transit_available'] = np.where(train_x['transit'].str.contains('publi
c transportation' or  'metro' or'bus'
                                                                  or 'line'), 'yes',
'no')
train_x.transit_available.value_counts()
```

Out[55]:
```
no     63901
yes    36099
Name: transit_available, dtype: int64
```

In [56]:
```
train_x.weekly_price.isnull().value_counts()
```

Out[56]:
```
True     75895
False    24105
Name: weekly_price, dtype: int64
```

In [57]:
```
#variable-72-weekly_price
train_x.weekly_price = train_x.weekly_price.str.replace('$','')
train_x.weekly_price = pd.to_numeric(train_x['weekly_price'], errors='coerce')
train_x.weekly_price.fillna('f', inplace = True)
train_x['weekly_price'] = np.where(~train_x['weekly_price'].astype(str).str.co
ntains('f'), 't', 'f')
train_x.weekly_price.value_counts()
```

Out[57]:
```
f    82962
t    17038
Name: weekly_price, dtype: int64
```

In [58]:
```
train_x.to_csv('train_x_may07_11pm.csv', encoding='utf-8', index=False)
```

In [61]:
```
train_x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Columns: 103 entries, Unnamed: 0 to transit_available
dtypes: datetime64[ns](2), float64(22), int64(2), object(77)
memory usage: 78.6+ MB
```

In [59]:
```
test_x = pd.read_csv("train_x_may06_5pm.csv")
```

```
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2
717: DtypeWarning: Columns (28,43,45,69) have mixed types. Specify dtype opti
on on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [60]:
```
test_x.year_host.value_counts()
```

Out[60]:
```
2015.0    20666
2014.0    18985
2016.0    16144
2013.0    15143
2012.0    11557
2017.0     6984
2011.0     6719
2010.0     2391
2009.0      959
2018.0      314
2008.0      138
Name: year_host, dtype: int64
```

RANDOM FOREST

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [193]:
```
data = pd.read_csv("train_x_may07_11pm.csv")
```

```
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2
717: DtypeWarning: Columns (28,43,45,69) have mixed types. Specify dtype opti
on on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [194]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Columns: 103 entries, Unnamed: 0 to transit_available
dtypes: float64(23), int64(2), object(78)
memory usage: 78.6+ MB
```

In [252]:
```
data['24-hour-check'] = data['24-hour-check'].astype('category')
data["24-hour-check"] = data["24-hour-check"].cat.codes
data['24-hour-check'] = data['24-hour-check'].astype('category')

data.bed_type = data.bed_type.astype('category')
data["bed_type"] = data["bed_type"].cat.codes
data.bed_type = data.bed_type.astype('category')

data.cancellation_policy = data.cancellation_policy.astype('category')
data["cancellation_policy"] = data["cancellation_policy"].cat.codes
data.cancellation_policy = data.cancellation_policy.astype('category')

data.Wifi = data.Wifi.astype('category')
data["Wifi"] = data["Wifi"].cat.codes
data.Wifi = data.Wifi.astype('category')

data.parking = data.parking.astype('category')
data["parking"] = data["parking"].cat.codes
data.parking = data.parking.astype('category')

data.Kitchen = data.Kitchen.astype('category')
data["Kitchen"] = data["Kitchen"].cat.codes
data.Kitchen = data.Kitchen.astype('category')

data.Breakfast = data.Breakfast.astype('category')
data["Breakfast"] = data["Breakfast"].cat.codes
data.Breakfast = data.Breakfast.astype('category')

data.ac = data.ac.astype('category')
data["ac"] = data["ac"].cat.codes
data.ac = data.ac.astype('category')

data.Heating = data.Heating.astype('category')
data["Heating"] = data["Heating"].cat.codes
data.Heating = data.Heating.astype('category')

data.host_has_profile_pic = data.host_has_profile_pic.astype('category')
data["host_has_profile_pic"] = data["host_has_profile_pic"].cat.codes
data.host_has_profile_pic = data.host_has_profile_pic.astype('category')

data.host_identity_verified = data.host_identity_verified.astype('category'
)
data["host_identity_verified"] = data["host_identity_verified"].cat.codes
data.host_identity_verified = data.host_identity_verified.astype('category'
)

data.host_is_superhost = data.host_is_superhost.astype('category')
data["host_is_superhost"] = data["host_is_superhost"].cat.codes
data.host_is_superhost = data.host_is_superhost.astype('category')

data.host_response_time = data.host_response_time.astype('category')
data["host_response_time"] = data["host_response_time"].cat.codes
data.host_response_time = data.host_response_time.astype('category')

data.host_verifications = data.host_verifications.astype('category')
data["host_verifications"] = data["host_verifications"].cat.codes
```

```
data.host_verifications = data.host_verifications.astype('category')

data.security_deposit = data.security_deposit.astype('category')
data["security_deposit"] = data["security_deposit"].cat.codes
data.security_deposit = data.security_deposit.astype('category')

data.state = data.state.astype('category')
data["state"] = data["state"].cat.codes
data.state = data.state.astype('category')

data.property_type=data.property_type.astype('category')
data["property_type"] = data["property_type"].cat.codes
data.property_type=data.property_type.astype('category')

data.instant_bookable = data.instant_bookable.astype('category')
data["instant_bookable"] = data["instant_bookable"].cat.codes
data.instant_bookable = data.instant_bookable.astype('category')

data.is_business_travel_ready = data.is_business_travel_ready.astype('categ
ory')
data["is_business_travel_ready"] = data["is_business_travel_ready"].cat.cod
es
data.is_business_travel_ready = data.is_business_travel_ready.astype('categ
ory')

data.is_location_exact = data.is_location_exact.astype('category')
data["is_location_exact"] = data["is_location_exact"].cat.codes
data.is_location_exact = data.is_location_exact.astype('category')

data.city = data.city.astype('category')
data["city"] = data["city"].cat.codes
data.city = data.city.astype('category')
```

In [253]:

```
data.room_type = data.room_type.astype('category')
data["room_type"] = data["room_type"].cat.codes
data.room_type = data.room_type.astype('category')
```

In [254]:
```
data.room_type = data.room_type.astype('category')
data["room_type"] = data["room_type"].cat.codes
data.room_type = data.room_type.astype('category')

data.year_first_review = data.year_first_review.astype('category')
data["year_first_review"] = data["year_first_review"].cat.codes
data.year_first_review = data.year_first_review.astype('category')

data.year_host  = data.year_host .astype('category')
data["year_host"] = data["year_host"].cat.codes
data.year_host  = data.year_host .astype('category')

data.transit_available = data.transit_available.astype('category')
data["transit_available"] = data["transit_available"].cat.codes
data.transit_available = data.transit_available.astype('category')

data.Reviews = data.Reviews.astype('category')
data["Reviews"] = data["Reviews"].cat.codes
data.Reviews = data.Reviews.astype('category')

data.weekly_price = data.weekly_price.astype('category')
data["weekly_price"] = data["weekly_price"].cat.codes
data.weekly_price = data.weekly_price.astype('category')

data.monthly_price = data.monthly_price.astype('category')
data["monthly_price"] = data["monthly_price"].cat.codes
data.monthly_price = data.monthly_price.astype('category')

data.house_rules = data.house_rules.astype('category')
data["house_rules"] = data["house_rules"].cat.codes
data.house_rules = data.house_rules.astype('category')

data.verification = data.verification.astype('category')
data["verification"] = data["verification"].cat.codes
data.verification = data.verification.astype('category')

data.Email = data.Email.astype('category')
data["Email"] = data["Email"].cat.codes
data.Email = data.Email.astype('category')

data.Phone = data.Phone.astype('category')
data["Phone"] = data["Phone"].cat.codes
data.Phone = data.Phone.astype('category')

data.Facebook = data.Facebook.astype('category')
data["Facebook"] = data["Facebook"].cat.codes
data.Facebook = data.Facebook.astype('category')

data.require_guest_phone_verification = data.require_guest_phone_verificati
on.astype('category')
data["require_guest_phone_verification"] = data["require_guest_phone_verifi
cation"].cat.codes
data.require_guest_phone_verification = data.require_guest_phone_verificati
on.astype('category')

data.require_guest_profile_picture  = data.require_guest_profile_picture.as
```

```
type('category')
data["require_guest_profile_picture"] = data["require_guest_profile_pictur
e"].cat.codes
data.require_guest_profile_picture  = data.require_guest_profile_picture.as
type('category')

data.Reviews = data.Reviews.astype('category')
data["Reviews"] = data["Reviews"].cat.codes
data.Reviews = data.Reviews.astype('category')
```

In [255]:
```
data.hair_dryer = data.hair_dryer.astype('category')
data["hair_dryer"] = data["hair_dryer"].cat.codes
data.hair_dryer = data.hair_dryer.astype('category')

data.hangers = data.hangers.astype('category')
data["hangers"] = data["hangers"].cat.codes
data.hangers = data.hangers.astype('category')

data.iron = data.iron.astype('category')
data["iron"] = data["iron"].cat.codes
data.iron = data.iron.astype('category')

data.laptop = data.laptop.astype('category')
data["laptop"] = data["laptop"].cat.codes
data.laptop = data.laptop.astype('category')

data.shampoo = data.shampoo.astype('category')
data["shampoo"] = data["shampoo"].cat.codes
data.shampoo = data.shampoo.astype('category')

data.essentials = data.essentials.astype('category')
data["essentials"] = data["essentials"].cat.codes
data.essentials = data.essentials.astype('category')

data.washer = data.washer.astype('category')
data["washer"] = data["washer"].cat.codes
data.washer = data.washer.astype('category')

data.dryer = data.dryer.astype('category')
data["dryer"] = data["dryer"].cat.codes
data.dryer = data.dryer.astype('category')

data.tv = data.tv.astype('category')
data["tv"] = data["tv"].cat.codes
data.tv = data.tv.astype('category')

data.pool = data.pool.astype('category')
data["pool"] = data["pool"].cat.codes
data.pool = data.pool.astype('category')

data.gym = data.gym.astype('category')
data["gym"] = data["gym"].cat.codes
data.gym = data.gym.astype('category')

data['hot tub'] = data['hot tub'].astype('category')
data["hot tub"] = data["hot tub"].cat.codes
data['hot tub'] = data['hot tub'].astype('category')

data.cable = data.cable.astype('category')
data["cable"] = data["cable"].cat.codes
data.cable = data.cable.astype('category')

data.wireless = data.wireless.astype('category')
data["wireless"] = data["wireless"].cat.codes
data.wireless = data.wireless.astype('category')
```

```
data.internet = data.internet.astype('category')
data["internet"] = data["internet"].cat.codes
data.internet = data.internet.astype('category')

data['family/kid friendly'] = data['family/kid friendly'].astype('category'
)
data['family/kid friendly'] = data['family/kid friendly'].cat.codes
data['family/kid friendly'] = data['family/kid friendly'].astype('category'
)

data.elevator = data.elevator.astype('category')
data["elevator"] = data["elevator"].cat.codes
data.elevator = data.elevator.astype('category')
```

In [199]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

In [200]:
```
train_y = pd.read_csv("airbnb_train_y.csv")
```

In [201]:
```
train_y.high_booking_rate = pd.to_numeric(train_y['high_booking_rate'], errors
='coerce')
train_y.high_booking_rate .fillna(train_y.high_booking_rate.mode()[0], inplace
 = True)
train_y.high_booking_rate.isnull().value_counts()
```

Out[201]:
```
False    100000
Name: high_booking_rate, dtype: int64
```

In [68]:
```
X = data[['accommodates','availability_30', 'availability_90', 'year_host',
          'availability_365','cleaning_fee','bathrooms', 'Kitchen', '24-hour-c
heck','Heating','bedrooms',
          'cancellation_policy', 'price', 'year_host', 'state','year_first_rev
iew',
         'host_has_profile_pic', 'host_is_superhost', 'host_response_time',
     'minimum_nights','instant_bookable', 'is_business_travel_ready',
         'is_location_exact', 'room_type','monthly_price', 'require_guest_phon
e_verification', 'Phone', 'Email', 'verification',
        'weekly_price','transit_available', 'hair_dryer','hangers','iron','lap
top','shampoo','essentials',
         'washer','dryer','tv','family/kid friendly','elevator']]
y = train_y['high_booking_rate']
```

In [244]:
```
X = data[['accommodates', 'availability_30',  'availability_90', 'availability
_365','bathrooms','bed_type',
             'cleaning_fee', 'Wifi', 'parking', 'Kitchen', '24-hour-check', 'B
reakfast', 'ac', 'Heating', 'property_type', 'price',
             'bedrooms','security_deposit','state', 'cancellation_policy', 'ex
tra_people', 'guests_included',
         'host_has_profile_pic', 'host_identity_verified', 'host_is_superhost'
, 'host_response_rate', 'host_response_time',
        'host_total_listings_count', 'maximum_nights', 'minimum_nights', 'inst
ant_bookable', 'is_business_travel_ready',
         'is_location_exact', 'room_type', 'city', 'monthly_price','year_host'
,'year_first_review',
           'require_guest_phone_verification', 'Facebook', 'Phone', 'Email',
'verification',
          'house_rules', 'weekly_price','transit_available', 'hair_dryer','han
gers','iron','laptop','shampoo','essentials',
         'washer','dryer','tv','pool','gym','hot tub','cable','wireless','inte
rnet','family/kid friendly','elevator']]
y = train_y['high_booking_rate']
```

In [245]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, rand
om_state=42)
```

In [246]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=600)
from sklearn.model_selection import train_test_split
```

In [247]:
```
rfc.fit(X_train,y_train)
```

Out[247]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=600, n_jobs=1, oob_score=False, random_state=None,
            verbose=0, warm_start=False)
```

In [ ]:
```
predictions = rfc.predict(X_test)
```

In [249]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, predictions)
```

Out[249]:
```
0.83296666666666663
```

In [250]:
```
data = pd.read_csv('test_may08-12am.csv')
```

In [256]:
```
X_test_data = data[['accommodates', 'availability_30',  'availability_90', 'av
ailability_365','bathrooms','bed_type',
             'cleaning_fee', 'Wifi', 'parking', 'Kitchen', '24-hour-check', 'B
reakfast', 'ac', 'Heating', 'property_type', 'price',
             'bedrooms','security_deposit','state', 'cancellation_policy', 'ex
tra_people', 'guests_included',
         'host_has_profile_pic', 'host_identity_verified', 'host_is_superhost'
, 'host_response_rate', 'host_response_time',
        'host_total_listings_count', 'maximum_nights', 'minimum_nights', 'inst
ant_bookable', 'is_business_travel_ready',
         'is_location_exact', 'room_type', 'city', 'monthly_price','year_host'
,'year_first_review',
           'require_guest_phone_verification', 'Facebook', 'Phone', 'Email',
'verification',
          'house_rules', 'weekly_price','transit_available', 'hair_dryer','han
gers','iron','laptop','shampoo','essentials',
         'washer','dryer','tv','pool','gym','hot tub','cable','wireless','inte
rnet','family/kid friendly','elevator']]
```

In [257]:
```
test_pred = rfc.predict(X_test_data)
```

In [258]:
```
test_pred_df = pd.DataFrame(test_pred)
```

In [259]:
```
test_pred_df[0].value_counts()
```

Out[259]:
```
0.0    10221
1.0     1987
Name: 0, dtype: int64
```

In [260]:
```
test_pred_df[0].to_csv('Outliers_high_booking_rate_predictions-may08.csv', enc
oding='utf-8', index=False)
```

ADABOOSTING

In [ ]:
```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.cross_validation import train_test_split
from sklearn.datasets import make_hastie_10_2
import matplotlib.pyplot as plt
from sklearn.ensemble import AdaBoostClassifier #For Classification
from sklearn.tree import DecisionTreeClassifier
```

In [2]:
```
data = pd.read_csv("train_x_april29_7pm.csv")
```

```
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2
728: DtypeWarning: Columns (29,44,46,70) have mixed types. Specify dtype opti
on on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [3]:
```
#variable-listing verifications
data['Reviews'] = np.where(data['host_identity_verified'].str.contains('Review
s' or 'review' or 'reviews'), 'yes', 'no')
data['Email'] = np.where(data['host_identity_verified'].str.contains('email' o
r 'Email'), 'yes', 'no')
data['Phone'] = np.where(data['host_identity_verified'].str.contains('phone' o
r 'Telephone'), 'yes', 'no')
data['Facebook'] = np.where(data['host_identity_verified'].str.contains('Faceb
ook'), 'yes', 'no')
```

In [4]:
```
#availability_30,60
data.availability_30 = (data.availability_30/30)*100
data.availability_60 = (data.availability_60/60)*100
data['interaction_availability'] = (data.availability_30/30) - (data.availabil
ity_60/60)
```

In [5]:
```
#availability_90,365
data.availability_90 = (data.availability_90/90)*100
data.availability_365 = (data.availability_365/365)*100
data['interaction_availability1'] = (data.availability_90/90) - (data.availabi
lity_365/365)
```

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 84 columns):
Unnamed: 0                    100000 non-null int64
Unnamed: 0.1                  100000 non-null int64
access                        65564 non-null object
accommodates                  100000 non-null float64
amenities                     100000 non-null object
availability_30               100000 non-null float64
availability_365              100000 non-null float64
availability_60               100000 non-null float64
availability_90               100000 non-null float64
bathrooms                     100000 non-null float64
bed_type                      100000 non-null object
bedrooms                      100000 non-null float64
beds                          100000 non-null float64
cancellation_policy           100000 non-null object
city                          100000 non-null object
city_name                     99999 non-null object
cleaning_fee                  100000 non-null float64
country                       99999 non-null object
country_code                  99993 non-null object
description                   99973 non-null object
experiences_offered           99997 non-null object
extra_people                  100000 non-null float64
first_review                  99992 non-null object
guests_included               100000 non-null float64
host_about                    69179 non-null object
host_acceptance_rate          8242 non-null object
host_has_profile_pic          100000 non-null object
host_identity_verified        100000 non-null object
host_is_superhost             100000 non-null float64
host_listings_count           99855 non-null object
host_location                 99589 non-null object
host_name                     99853 non-null object
host_neighbourhood            82767 non-null object
host_response_rate            100000 non-null float64
host_response_time            100000 non-null float64
host_since                    99858 non-null object
host_total_listings_count     100000 non-null float64
host_verifications            99990 non-null object
house_rules                   69607 non-null object
instant_bookable              100000 non-null float64
interaction                   64120 non-null object
is_business_travel_ready      100000 non-null float64
is_location_exact             100000 non-null object
jurisdiction_names            47310 non-null object
latitude                      99990 non-null object
license                       14279 non-null object
longitude                     99989 non-null object
market                        99567 non-null object
maximum_nights                100000 non-null float64
minimum_nights                100000 non-null float64
monthly_price                 20564 non-null object
name                          99967 non-null object
neighborhood_overview         69257 non-null object
neighbourhood                 86447 non-null object
```

```
notes                              51466 non-null object
price                              100000 non-null float64
property_type                      100000 non-null object
require_guest_phone_verification   99981 non-null object
require_guest_profile_picture      100000 non-null object
requires_license                   100000 non-null object
room_type                          100000 non-null object
security_deposit                   100000 non-null object
smart_location                     99980 non-null object
space                              78955 non-null object
square_feet                        1581 non-null float64
state                              100000 non-null object
street                             99981 non-null object
summary                            97036 non-null object
transit                            70833 non-null object
weekly_price                       24105 non-null object
zipcode                            98951 non-null object
Wifi                               100000 non-null object
parking                            100000 non-null object
Kitchen                            100000 non-null object
24-hour-check                      100000 non-null object
Breakfast                          100000 non-null object
ac                                 100000 non-null object
Heating                            100000 non-null object
Reviews                            100000 non-null object
Email                              100000 non-null object
Phone                              100000 non-null object
Facebook                           100000 non-null object
interaction_availability           100000 non-null float64
interaction_availability1          100000 non-null float64
dtypes: float64(23), int64(2), object(59)
memory usage: 64.1+ MB
```

In [7]:
```
train_y = pd.read_csv("airbnb_train_y.csv")
```

In [8]:
```
data['24-hour-check'] = data['24-hour-check'].astype('category')
data["24-hour-check"] = data["24-hour-check"].cat.codes
data['24-hour-check'] = data['24-hour-check'].astype('category')

data.bed_type = data.bed_type.astype('category')
data["bed_type"] = data["bed_type"].cat.codes
data.bed_type = data.bed_type.astype('category')

data.cancellation_policy = data.cancellation_policy.astype('category')
data["cancellation_policy"] = data["cancellation_policy"].cat.codes
data.cancellation_policy = data.cancellation_policy.astype('category')

data.Wifi = data.Wifi.astype('category')
data["Wifi"] = data["Wifi"].cat.codes
data.Wifi = data.Wifi.astype('category')

data.parking = data.parking.astype('category')
data["parking"] = data["parking"].cat.codes
data.parking = data.parking.astype('category')

data.Kitchen = data.Kitchen.astype('category')
data["Kitchen"] = data["Kitchen"].cat.codes
data.Kitchen = data.Kitchen.astype('category')

data.Breakfast = data.Breakfast.astype('category')
data["Breakfast"] = data["Breakfast"].cat.codes
data.Breakfast = data.Breakfast.astype('category')

data.ac = data.ac.astype('category')
data["ac"] = data["ac"].cat.codes
data.ac = data.ac.astype('category')

data.Heating = data.Heating.astype('category')
data["Heating"] = data["Heating"].cat.codes
data.Heating = data.Heating.astype('category')

data.host_has_profile_pic = data.host_has_profile_pic.astype('category')
data["host_has_profile_pic"] = data["host_has_profile_pic"].cat.codes
data.host_has_profile_pic = data.host_has_profile_pic.astype('category')

data.host_is_superhost = data.host_is_superhost.astype('category')
data["host_is_superhost"] = data["host_is_superhost"].cat.codes
data.host_is_superhost = data.host_is_superhost.astype('category')

data.host_response_time = data.host_response_time.astype('category')
data["host_response_time"] = data["host_response_time"].cat.codes
data.host_response_time = data.host_response_time.astype('category')

data.Reviews = data.Reviews.astype('category')
data["Reviews"] = data["Reviews"].cat.codes
data.Reviews = data.Reviews.astype('category')

data.Facebook = data.Facebook.astype('category')
data["Facebook"] = data["Facebook"].cat.codes
data.Facebook = data.Facebook.astype('category')
```

```
data.Phone = data.Phone.astype('category')
data["Phone"] = data["Phone"].cat.codes
data.Phone = data.Phone.astype('category')

data.Email = data.Email.astype('category')
data["Email"] = data["Email"].cat.codes
data.Email = data.Email.astype('category')


data.security_deposit = data.security_deposit.astype('category')
data["security_deposit"] = data["security_deposit"].cat.codes
data.security_deposit = data.security_deposit.astype('category')

data.state = data.state.astype('category')
data["state"] = data["state"].cat.codes
data.state = data.state.astype('category')

data.property_type=data.property_type.astype('category')
data["property_type"] = data["property_type"].cat.codes
data.property_type=data.property_type.astype('category')

data.instant_bookable = data.instant_bookable.astype('category')
data["instant_bookable"] = data["instant_bookable"].cat.codes
data.instant_bookable = data.instant_bookable.astype('category')

data.is_business_travel_ready = data.is_business_travel_ready.astype('categ
ory')
data["is_business_travel_ready"] = data["is_business_travel_ready"].cat.cod
es
data.is_business_travel_ready = data.is_business_travel_ready.astype('categ
ory')

data.is_location_exact = data.is_location_exact.astype('category')
data["is_location_exact"] = data["is_location_exact"].cat.codes
data.is_location_exact = data.is_location_exact.astype('category')

data.city = data.city.astype('category')
data["city"] = data["city"].cat.codes
data.city = data.city.astype('category')
```

In [9]:
```
data.room_type = data.room_type.astype('category')
data["room_type"] = data["room_type"].cat.codes
data.room_type = data.room_type.astype('category')
```

In [10]:
```
train_y.high_booking_rate = pd.to_numeric(train_y['high_booking_rate'], errors
='coerce')
train_y.high_booking_rate .fillna(train_y.high_booking_rate.mode()[0], inplace
 = True)
train_y.high_booking_rate.isnull().value_counts()
```

Out[10]:
```
False    100000
Name: high_booking_rate, dtype: int64
```

In [13]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 84 columns):
Unnamed: 0                          100000 non-null int64
Unnamed: 0.1                        100000 non-null int64
access                              65564 non-null object
accommodates                        100000 non-null float64
amenities                           100000 non-null object
availability_30                     100000 non-null float64
availability_365                    100000 non-null float64
availability_60                     100000 non-null float64
availability_90                     100000 non-null float64
bathrooms                           100000 non-null float64
bed_type                            100000 non-null category
bedrooms                            100000 non-null float64
beds                                100000 non-null float64
cancellation_policy                 100000 non-null category
city                                100000 non-null category
city_name                           99999 non-null object
cleaning_fee                        100000 non-null float64
country                             99999 non-null object
country_code                        99993 non-null object
description                         99973 non-null object
experiences_offered                 99997 non-null object
extra_people                        100000 non-null float64
first_review                        99992 non-null object
guests_included                     100000 non-null float64
host_about                          69179 non-null object
host_acceptance_rate                8242 non-null object
host_has_profile_pic                100000 non-null category
host_identity_verified              100000 non-null object
host_is_superhost                   100000 non-null category
host_listings_count                 99855 non-null object
host_location                       99589 non-null object
host_name                           99853 non-null object
host_neighbourhood                  82767 non-null object
host_response_rate                  100000 non-null float64
host_response_time                  100000 non-null category
host_since                          99858 non-null object
host_total_listings_count           100000 non-null float64
host_verifications                  99990 non-null object
house_rules                         69607 non-null object
instant_bookable                    100000 non-null category
interaction                         64120 non-null object
is_business_travel_ready            100000 non-null category
is_location_exact                   100000 non-null category
jurisdiction_names                  47310 non-null object
latitude                            99990 non-null object
license                             14279 non-null object
longitude                           99989 non-null object
market                              99567 non-null object
maximum_nights                      100000 non-null float64
minimum_nights                      100000 non-null float64
monthly_price                       20564 non-null object
name                                99967 non-null object
neighborhood_overview               69257 non-null object
neighbourhood                       86447 non-null object
```

```
notes                              51466 non-null object
price                              100000 non-null float64
property_type                      100000 non-null category
require_guest_phone_verification   99981 non-null object
require_guest_profile_picture      100000 non-null object
requires_license                   100000 non-null object
room_type                          100000 non-null category
security_deposit                   100000 non-null category
smart_location                     99980 non-null object
space                              78955 non-null object
square_feet                        1581 non-null float64
state                              100000 non-null category
street                             99981 non-null object
summary                            97036 non-null object
transit                            70833 non-null object
weekly_price                       24105 non-null object
zipcode                            98951 non-null object
Wifi                               100000 non-null category
parking                            100000 non-null category
Kitchen                            100000 non-null category
24-hour-check                      100000 non-null category
Breakfast                          100000 non-null category
ac                                 100000 non-null category
Heating                            100000 non-null category
Reviews                            100000 non-null category
Email                              100000 non-null category
Phone                              100000 non-null category
Facebook                           100000 non-null category
interaction_availability           100000 non-null float64
interaction_availability1          100000 non-null float64
dtypes: category(24), float64(19), int64(2), object(39)
memory usage: 48.2+ MB
```

In [11]:
```
X = data[['accommodates', 'interaction_availability1', 'interaction_availabili
ty','bathrooms','bed_type',
             'cleaning_fee', 'Wifi', 'parking', 'Kitchen', '24-hour-check', 'B
reakfast', 'ac', 'Heating', 'property_type', 'price',
             'bedrooms', 'beds','security_deposit','state', 'cancellation_poli
cy', 'extra_people', 'guests_included',
         'host_has_profile_pic', 'Reviews', 'Email', 'Phone', 'Facebook', 'hos
t_is_superhost', 'host_response_rate', 'host_response_time',
        'host_total_listings_count', 'maximum_nights', 'minimum_nights', 'inst
ant_bookable', 'is_business_travel_ready',
         'is_location_exact', 'room_type', 'city']]
y = train_y['high_booking_rate']
```

In [12]:
```
   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, r
andom_state=42)
```

In [22]:
```
dt = DecisionTreeClassifier()
```

In [54]:
```
clf = AdaBoostClassifier(n_estimators=400, base_estimator=dt,learning_rate=0.2
)
clf.fit(X_train,y_train)
```

Out[54]:
```
AdaBoostClassifier(algorithm='SAMME.R',
          base_estimator=DecisionTreeClassifier(class_weight=None, criterion
='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'),
          learning_rate=0.2, n_estimators=400, random_state=None)
```

In [55]:
```
predictions = clf.predict(X_test)
```

In [56]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, predictions)
```

Out[56]:
```
0.8219666666666666
```

In [37]:
```
data = pd.read_csv('test_x_may3_11am.csv')
```

In [38]:
```
4-
data["state"] = data["state"].cat.codes
data.state = data.state.astype('category')

data.property_type=data.property_type.astype('category')
data["property_type"] = data["property_type"].cat.codes
data.property_type=data.property_type.astype('category')

data.instant_bookable = data.instant_bookable.astype('category')
data["instant_bookable"] = data["instant_bookable"].cat.codes
data.instant_bookable = data.instant_bookable.astype('category')

data.is_business_travel_ready = data.is_business_travel_ready.astype('categor
y')
data["is_business_travel_ready"] = data["is_business_travel_ready"].cat.codes
data.is_business_travel_ready = data.is_business_travel_ready.astype('categor
y')

data.is_location_exact = data.is_location_exact.astype('category')
data["is_location_exact"] = data["is_location_exact"].cat.codes
data.is_location_exact = data.is_location_exact.astype('category')

data.city = data.city.astype('category')
data["city"] = data["city"].cat.codes
data.city = data.city.astype('category')
```

In [39]:
```
data.room_type = data.room_type.astype('category')
data["room_type"] = data["room_type"].cat.codes
data.room_type = data.room_type.astype('category')
```

In [40]:
```
X_test_data = data[['accommodates', 'availability_30', 'availability_60', 'ava
ilability_90', 'availability_365','bathrooms','bed_type',
             'cleaning_fee', 'Wifi', 'parking', 'Kitchen', '24-hour-check', 'B
reakfast', 'ac', 'Heating', 'property_type', 'price',
             'bedrooms', 'beds','security_deposit','state', 'cancellation_poli
cy', 'extra_people', 'guests_included',
         'host_has_profile_pic', 'host_identity_verified', 'host_is_superhost'
, 'host_response_rate', 'host_response_time',
        'host_total_listings_count', 'maximum_nights', 'minimum_nights', 'inst
ant_bookable', 'is_business_travel_ready',
         'is_location_exact', 'room_type', 'city']]
```

In [41]:
```
test_pred = clf.predict(X_test_data)
```

In [42]:
```
test_pred_df = pd.DataFrame(test_pred)
```

In [43]:
```
test_pred_df[0].value_counts()
```

Out[43]:
```
0.0    10752
1.0     1456
Name: 0, dtype: int64
```

In [ ]:
```
learning_rate = df[0.1, 0.2, 0.3, 0.3, 0.5]
```

GRADIENT BOOSTING

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
data =pd.read_csv("train_x_april29_7pm.csv")
```

```
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2
728: DtypeWarning: Columns (29,44,46,70) have mixed types. Specify dtype opti
on on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [3]:
```
#variable-listing verifications
data['Reviews'] = np.where(data['host_identity_verified'].str.contains('Review
s' or 'review' or 'reviews'), 'yes', 'no')
data['Email'] = np.where(data['host_identity_verified'].str.contains('email' o
r 'Email'), 'yes', 'no')
data['Phone'] = np.where(data['host_identity_verified'].str.contains('phone' o
r 'Telephone'), 'yes', 'no')
data['Facebook'] = np.where(data['host_identity_verified'].str.contains('Faceb
ook'), 'yes', 'no')
```

In [4]:
```
#availability_interaction
data['availability_interaction_1'] = (data.availability_30*data.availability_6
0)*100
data['availability_interaction_2'] = (data.availability_30*data.availability_9
0)*100
data['availability_interaction_3'] = (data.availability_60*data.availability_9
0)*100
```

In [5]:
```
#interaction accomodates-bedrooms
data['interaction_accommodates_1']= data.accommodates* data.bedrooms
data['interaction_accommodates_2'] = data.accommodates*data.beds
```

In [6]:
```
#variable-25-first_review
data['date_first_review'] = pd.to_datetime(data['first_review'],errors='coerc
e')
data['year_first_review'] = pd.DatetimeIndex(data['date_first_review']).year
data.year_first_review = pd.to_numeric(data['year_first_review'], errors='coer
ce')
data.year_first_review.fillna(data.year_first_review.mode()[0], inplace = True
)
data.year_first_review.isnull().value_counts()
```

Out[6]:
```
False    100000
Name: year_first_review, dtype: int64
```

In [7]:
```
#variable-71-transit
data['transit_available'] = np.where(data['transit'].str.contains('public tran
sportation' or 'Metro' or 'metro' or'bus'
                                                          or 'Line' or 'line'
), 'yes', 'no')
data.transit_available.value_counts()
```

Out[7]:
```
no     65873
yes    34127
Name: transit_available, dtype: int64
```

In [8]:
```
#variable host since
data['date'] = pd.to_datetime(data['host_since'],errors='coerce')
data['year'] = pd.DatetimeIndex(data['date']).year
data.year = pd.to_numeric(data['year'], errors='coerce')
data.year.fillna(data.year.mode()[0], inplace = True)
data.year.isnull().value_counts()
```

Out[8]:
```
False    100000
Name: year, dtype: int64
```

In [9]:
```
##host verifications
#variable-listing verifications
data['Reviews'] = np.where(data['host_verifications'].str.contains('Reviews' o
r 'review' or 'reviews'), 'yes', 'no')
data['Email'] = np.where(data['host_verifications'].str.contains('email' or 'E
mail'), 'yes', 'no')
data['Phone'] = np.where(data['host_verifications'].str.contains('phone' or 'T
elephone'), 'yes', 'no')
data['Facebook'] = np.where(data['host_verifications'].str.contains('Facebook'
), 'yes', 'no')
```

In [10]:
```
data['24-hour-check'] = data['24-hour-check'].astype('category')
data["24-hour-check"] = data["24-hour-check"].cat.codes
data['24-hour-check'] = data['24-hour-check'].astype('category')

data.bed_type = data.bed_type.astype('category')
data["bed_type"] = data["bed_type"].cat.codes
data.bed_type = data.bed_type.astype('category')

data.cancellation_policy = data.cancellation_policy.astype('category')
data["cancellation_policy"] = data["cancellation_policy"].cat.codes
data.cancellation_policy = data.cancellation_policy.astype('category')

data.Wifi = data.Wifi.astype('category')
data["Wifi"] = data["Wifi"].cat.codes
data.Wifi = data.Wifi.astype('category')

data.parking = data.parking.astype('category')
data["parking"] = data["parking"].cat.codes
data.parking = data.parking.astype('category')

data.Kitchen = data.Kitchen.astype('category')
data["Kitchen"] = data["Kitchen"].cat.codes
data.Kitchen = data.Kitchen.astype('category')

data.Breakfast = data.Breakfast.astype('category')
data["Breakfast"] = data["Breakfast"].cat.codes
data.Breakfast = data.Breakfast.astype('category')

data.ac = data.ac.astype('category')
data["ac"] = data["ac"].cat.codes
data.ac = data.ac.astype('category')

data.Heating = data.Heating.astype('category')
data["Heating"] = data["Heating"].cat.codes
data.Heating = data.Heating.astype('category')

data.host_has_profile_pic = data.host_has_profile_pic.astype('category')
data["host_has_profile_pic"] = data["host_has_profile_pic"].cat.codes
data.host_has_profile_pic = data.host_has_profile_pic.astype('category')

data.host_identity_verified = data.host_identity_verified.astype('category'
)
data["host_identity_verified"] = data["host_identity_verified"].cat.codes
data.host_identity_verified = data.host_identity_verified.astype('category'
)

data.host_is_superhost = data.host_is_superhost.astype('category')
data["host_is_superhost"] = data["host_is_superhost"].cat.codes
data.host_is_superhost = data.host_is_superhost.astype('category')

data.host_response_time = data.host_response_time.astype('category')
data["host_response_time"] = data["host_response_time"].cat.codes
data.host_response_time = data.host_response_time.astype('category')


data.Reviews = data.Reviews.astype('category')
```

```
data["Reviews"] = data["Reviews"].cat.codes
data.Reviews = data.Reviews.astype('category')

data.Facebook = data.Facebook.astype('category')
data["Facebook"] = data["Facebook"].cat.codes
data.Facebook = data.Facebook.astype('category')

data.Phone = data.Phone.astype('category')
data["Phone"] = data["Phone"].cat.codes
data.Phone = data.Phone.astype('category')

data.Email = data.Email.astype('category')
data["Email"] = data["Email"].cat.codes
data.Email = data.Email.astype('category')


data.security_deposit = data.security_deposit.astype('category')
data["security_deposit"] = data["security_deposit"].cat.codes
data.security_deposit = data.security_deposit.astype('category')

data.state = data.state.astype('category')
data["state"] = data["state"].cat.codes
data.state = data.state.astype('category')

data.property_type=data.property_type.astype('category')
data["property_type"] = data["property_type"].cat.codes
data.property_type=data.property_type.astype('category')

data.instant_bookable = data.instant_bookable.astype('category')
data["instant_bookable"] = data["instant_bookable"].cat.codes
data.instant_bookable = data.instant_bookable.astype('category')

data.is_business_travel_ready = data.is_business_travel_ready.astype('categ
ory')
data["is_business_travel_ready"] = data["is_business_travel_ready"].cat.cod
es
data.is_business_travel_ready = data.is_business_travel_ready.astype('categ
ory')

data.is_location_exact = data.is_location_exact.astype('category')
data["is_location_exact"] = data["is_location_exact"].cat.codes
data.is_location_exact = data.is_location_exact.astype('category')

data.city = data.city.astype('category')
data["city"] = data["city"].cat.codes
data.city = data.city.astype('category')

data.year_first_review = data.year_first_review.astype('category')
data["year_first_review"] = data["year_first_review"].cat.codes
data.year_first_review = data.year_first_review.astype('category')

data.host_since = data.host_since.astype('category')
data["host_since"] = data["host_since"].cat.codes
data.host_since = data.host_since.astype('category')
```

In [11]:
```
data.room_type = data.room_type.astype('category')
data["room_type"] = data["room_type"].cat.codes
data.room_type = data.room_type.astype('category')
```

In [12]:
```
%matplotlib inline

import pandas as pd
import numpy as np
from IPython.display import display

from sklearn import metrics
```

In [13]:
```
train_y = pd.read_csv("airbnb_train_y.csv")
```

In [14]:
```
train_y.high_booking_rate = pd.to_numeric(train_y['high_booking_rate'], errors
='coerce')
train_y.high_booking_rate .fillna(train_y.high_booking_rate.mode()[0], inplace
 = True)
train_y.high_booking_rate.isnull().value_counts()
```

Out[14]:
```
False    100000
Name: high_booking_rate, dtype: int64
```

In [15]:
```
X = data[['availability_60', 'availability_365', 'bathrooms','bed_type',
             'cleaning_fee', 'Wifi', 'parking', 'Kitchen', '24-hour-check', 'B
reakfast', 'ac', 'Heating',
          'property_type', 'price', 'bedrooms', 'beds','security_deposit', 'ca
ncellation_policy', 'state',
          'extra_people', 'guests_included','host_has_profile_pic', 'Reviews',
 'year_first_review',
          'host_is_superhost', 'host_response_rate',  'host_response_time', 'h
ost_total_listings_count',
          'maximum_nights', 'minimum_nights', 'instant_bookable', 'is_business
_travel_ready', 'is_location_exact',
          'room_type', 'city', 'availability_interaction_1', 'availability_int
eraction_2', 'availability_interaction_3',
          'interaction_accommodates_1',  'interaction_accommodates_2']]
y = train_y['high_booking_rate']
```

In [16]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=300)
from sklearn.model_selection import train_test_split
```

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, rand
om_state=42)
```

In [18]:
```
import pandas
from sklearn import
from sklearn.ensemble import GradientBoostingClassifier
```

In [22]:
```
##gradient boosting
X = X_train
Y = y_train
seed = 7
num_trees = 700
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = GradientBoostingClassifier(n_estimators=num_trees, random_state=seed)
results = model_selection.cross_val_score(model, X, Y, cv=kfold)
print(results.mean())
```

```
0.8366571428571428
```