Airbnb Data Prediction

Sanskriti Kansal

1. Business Understanding

We have used cutting-edge machine learning techniques to create a prediction model that has significant business benefits, especially for the hospitality industry. Our approach focuses on streamlining processes, boosting visitor experiences, and giving businesses a competitive edge in the following ways:

- Property Optimization: Businesses may strategically improve the client experience by utilizing our data-driven insights. Our approach identifies opportunities for development and directs property owners in putting quality-raising measures into place for their listings, resulting in increased guest satisfaction and ratings.
- Competitive Advantage: By emphasizing key elements that help firms achieve a flawless rating score, our strategy gives companies a clear advantage over their rivals. Businesses may stand out from the competition and draw customers looking for outstanding experiences by emphasizing and prioritizing these crucial components.
- Operational Enhancement: Our approach highlights places where business operations could be improved. Businesses should proactively implement operational adjustments to ensure a flawless visitor experience and promote positive reviews and ratings by focusing on areas like amenities, listing details, and pricing.
- Growth & Benchmarking: Our approach makes it easy to compare properties across industries or within a company's portfolio. Businesses can discover underperforming properties, take corrective action, and boost overall business development and performance by comparing expected and actual ratings.

In addition to the value we bring to businesses, we have identified key players who directly benefit from our predictive model:

- Property Owners: Our predictive model enables owners of real estate to make data-driven decisions about how to best utilize their properties, improve visitor experiences, and raise the probability of receiving a perfect rating score.
- Property Management Companies: Companies who manage several properties can make use of our approach to evaluate and enhance the potential of those properties in order to attain a flawless rating score. This enables them to properly allocate resources, put operational changes into place, and maximize marketing initiatives.
- Real Estate Investors: Real estate investors can use our methodology to assess possible profits based on the criteria for the optimal rating score if they are interested in short-term rental properties. This enables people to choose wisely while buying and improving real estate.
- Property Listing Businesses: Platforms like Airbnb can use our model's predictions to give property owners advice on getting a perfect rating. These platforms can promote user engagement and happiness by encouraging better customer experiences.

Businesses and major players in the hospitality sector can achieve new heights of success, improve guest experiences, and stay one step ahead of the competition by utilizing our predictive model.

2. Data Understanding and Data Preparation

2.1 Datasets

For prediction, we used the provided Airbnb dataset. However, to gain a better understanding of the localities, we obtained an external dataset [1] from Kaggle, which gave us insights on the top 100 cities in the US, the living wages in each city along with the population and density and with a rank.

2.2 Data Understanding

After getting a quick glance of the data set, we can see that the above data set contains a total of 70 columns and 100,000 rows.

We further went through each column to understand how relevant each column is for the prediction of perfect rating score.

For the purpose of prediction, the team brainstormed, plotted charts, tried different features by trial and error, cleaned over 75 features and finally decided on the following 63 features to be used as the dependent variables:

ID	Feature Name	Brief Description	R Code line numbers
1	accommodates	Original feature from Airbnb dataset	67
2	availability_30	Original feature from Airbnb dataset	68
3	availability_60	Original feature from Airbnb dataset	69
4	bathrooms	Original feature from Airbnb dataset	70,91
5	bed_type	Original feature from Airbnb dataset	97
6	bedrooms	Original feature from Airbnb dataset	71,92
7	beds	Original feature from Airbnb dataset	72,93
8	cancellation_policy	Original feature from Airbnb dataset	96

9	city_name	Original feature from Airbnb dataset	94
10	cleaning_fee	Original feature from Airbnb dataset	56,88
11	extra_people	Original feature from Airbnb dataset	73
12	first_review	Original feature from Airbnb dataset	N/A
13	guests_included	Original feature from Airbnb dataset	79
14	host_acceptance_rate	Original feature from Airbnb dataset	76
15	host_is_superhost	Original feature from Airbnb dataset	75
16	host_listings_count	Original feature from Airbnb dataset	77
17	host_response_time	Original feature from Airbnb dataset	82
18	host_since	Original feature from Airbnb dataset	101
19	instant_bookable	Original feature from Airbnb dataset	N/A
20	is_location_exact	Original feature from Airbnb dataset	N/A
21	maximum_nights	Original feature from Airbnb dataset	N/A
22	minimum_nights	Original feature from Airbnb dataset	N/A
23	property_type	Original feature from Airbnb dataset	122-123
24	require_guest_phone_verification	Original feature from Airbnb dataset	N/A
25	require_guest_profile_picture	Original feature from Airbnb dataset	N/A
26	requires_license	Original feature from Airbnb dataset	N/A
27	room_type	Original feature from Airbnb dataset	98
28	security_deposit	Original feature from Airbnb dataset	61-63

29	density	Original feature from Living wages dataset	113
30	one_adult_no_kids_living_wage	Original feature from Living wages dataset	115
31	population_2020	Original feature from Living wages dataset	108,121
32	rank_2020	Original feature from Living wages dataset	116-117
33	two_adults_both_working_no_kids_living_wage	Original feature from Living wages dataset	114
34	airconditioning	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
35	breakfast	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
36	cabletv	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
37	carbonmonoxidedetector	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
38	dryer	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
39	en	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
40	fireextinguisher	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
41	firstaidkit	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
42	freeparkingonpremises	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
43	hairdryer	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127

44	hangers	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
45	indoorfireplace	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
46	internet	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
47	iron	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
48	kidfriendly	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
49	laptopfriendlyworkspace	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
50	lockonbedroomdoor	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
51	petsallowed	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
52	shampoo	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
53	tv	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
54	V1	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
55	washer	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
56	wifi	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127

57	wirelessinternet	Dummy feature created from the amenities feature in Airbnb dataset	38-51,127
58	amenities_n	A numeric field indicating the number of characters in amenities field, derived from the amenities feature in Airbnb dataset	102
59	desc_score	A numeric field generated by performing sentiment analysis on the description column in Airbnb dataset	135-177
60	months_diff_scale	A numeric field derived by scaling the first_review feature in Airbnb dataset	118
61	price_per_person	A numeric field derived by dividing the price by accommodates feature in Airbnb dataset	100
62	minimum_price	A numeric field created by multiplying minimum nights and price_per_person features in Airbnb dataset	111
63	charges_for_extra	A binary field indicating if charges exist in from extra_people feature in Airbnb dataset	74

In addition to the aforementioned variables, as part of our feature engineering efforts, we investigated extra features, hypothesizing that their inclusion could improve the predictive capacity of our model. Nonetheless, these additional features were dropped from the final model because they did not make a significant contribution to improving our model's performance metrics.

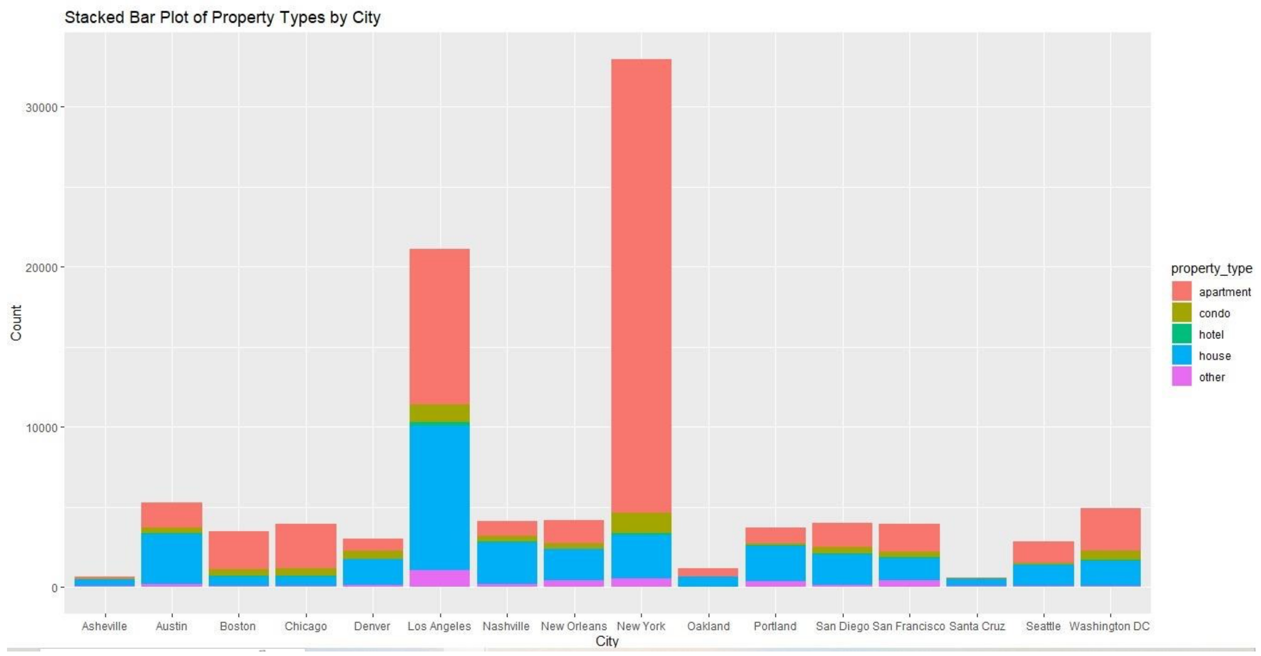
ID	Feature Name	Brief Description	R Code line numbers
1	pop_change	A numeric field calculated as the difference between population in 2020 and 2010 of a city from the Living wages dataset, indicating the expansion of each city.	110

2	fare_high	A binary field indicating if the price_per_person is greater than the living wage of one adult, combining data from Living wage and Airbnb dataset.	106-107
3	desc_n	A numeric field, which is the length of the characters in description field from the Airbnb dataset, which indicates the effort a host puts up in their listing.	103-104
4	trust	A numeric field, which assigns a score to the strength of ‘trust’ using sentiment analysis in house rules column from the Airbnb dataset.	715-774
5	anticipation	A numeric field, which assigns a score to the strength of ‘anticipation’ using sentiment analysis in the house rules column from the Airbnb dataset.	715-774
6	monthly_discount	A numeric field, which calculates the difference between the price*30 days and the monthly_price column in Airbnb dataset.	785-787

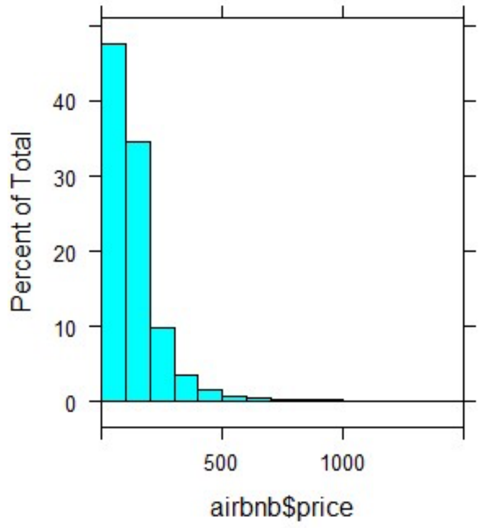
7	months_diff	A numeric field which is the number of months in between the first review and today's date from the Airbnb dataset.	777-781
8	host_response_rate	Original feature from Airbnb dataset	80
9	maximum_price	A numeric field created by multiplying maximum nights and price_per_person features in Airbnb dataset	112
10	price_per_bedroom	A numeric field derived by dividing the price by bedrooms feature in Airbnb dataset	69*
11	country_code	Original feature from Airbnb dataset	40-42*

*Can be found in Logistic_regression.r file

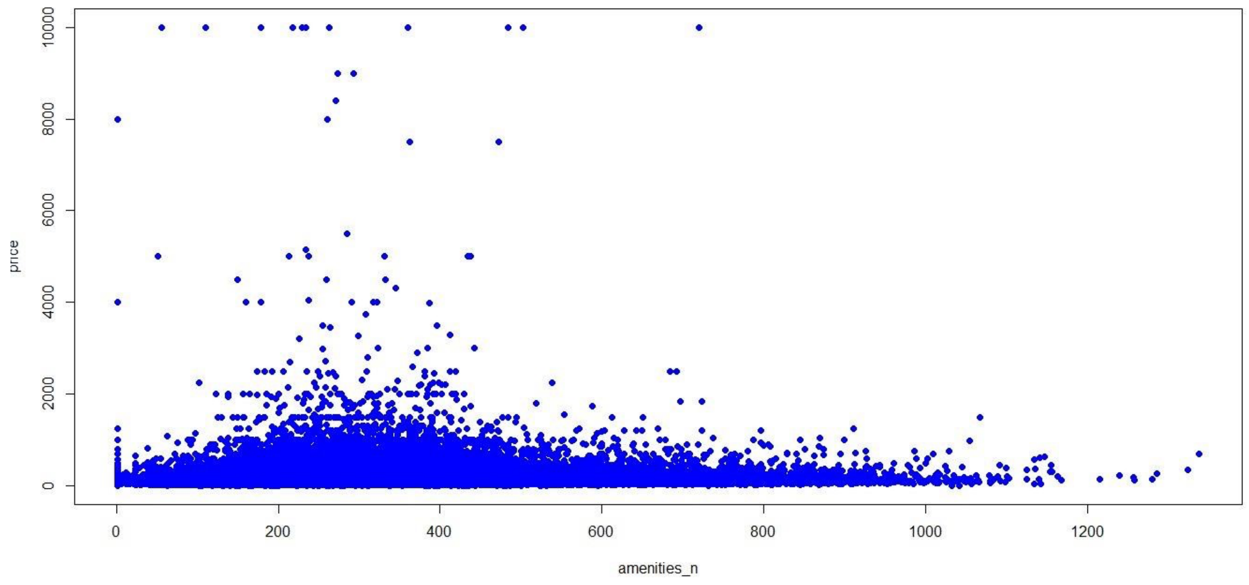
2.3 Data Insights



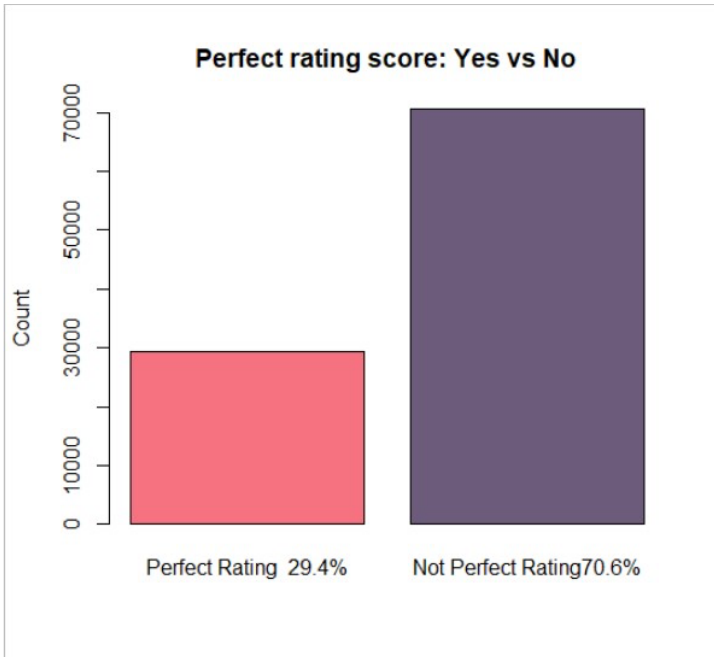
Property type: From the above graph, we can clearly see that most of the airbnbs are present in New York. Moreover, in each city, most properties are of the type 'Apartment'. Following New York, Los Angeles also contributes a significant portion of data, with relatively equal proportions of apartments and houses.



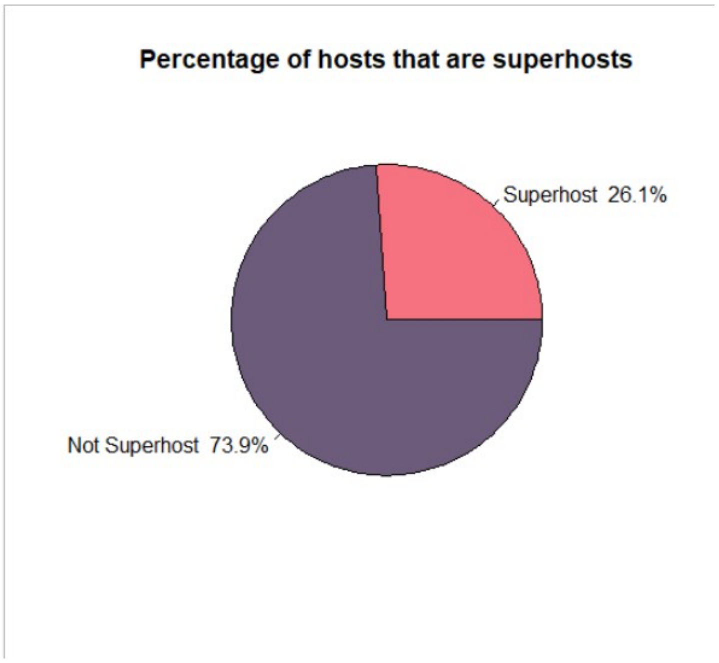
Price: The data set exhibits a positive skew, indicating that the majority of its values are concentrated towards the left side. The majority of properties in the data set fall within the price range of up to $500 per night, while only a small number of expensive properties are included. Because of which, the model might have issues in understanding the pattern/trend of expensive properties.



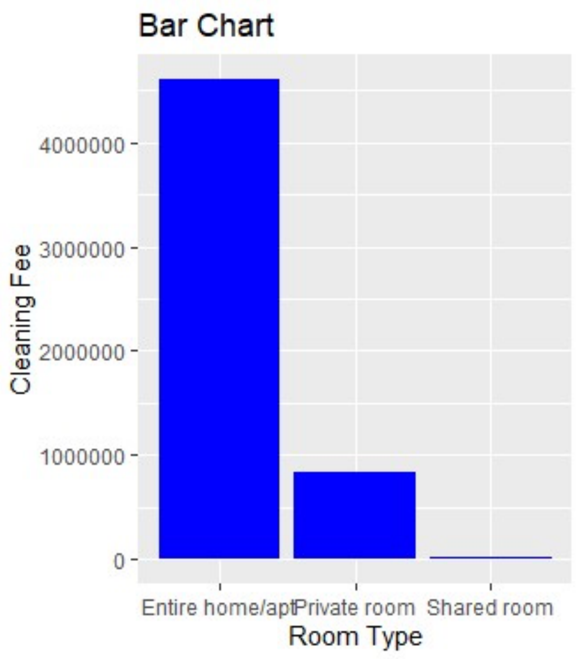
Amenities: The majority of the dataset generally falls within a price range of up to 4000, irrespective of the number of amenities. Nevertheless, there are certain outliers where the highest price is observed for specific numbers of amenities, while the majority of the dataset maintains prices lower than 4000.



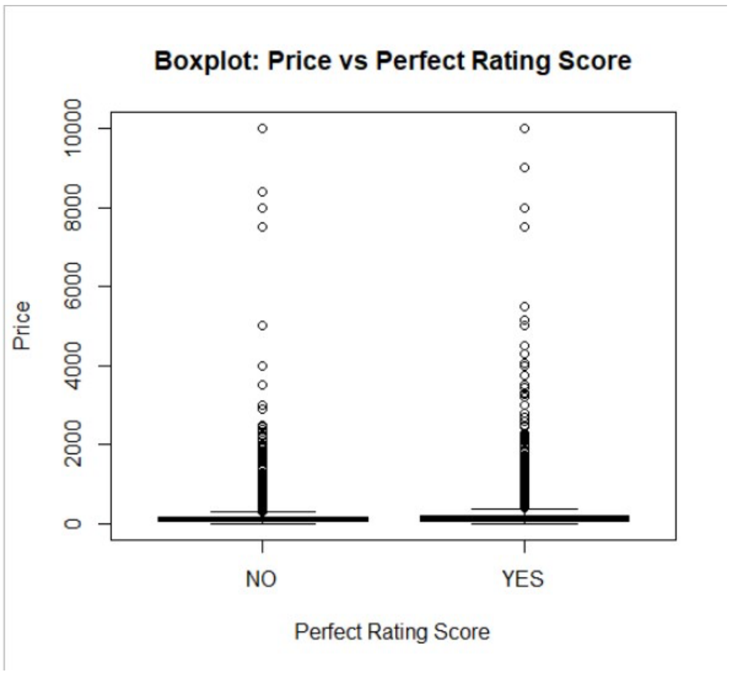
perfect_rating_score: From the above bar plot, we can see the about 30000 listings have a perfect rating score while almost 70000 do not have it, which makes it roughly 1 in 4 listings that have a perfect rating.



host_is_superhost: From the host_is_superhost column, we can plot the above piechart to analyze the percentage of hosts that are superhosts. We can see that only about 1/4th of the hosts are superhosts, which is probably why only 1 in 4 listings have a perfect rating score.



Room type: From the above bar chart, we can see that the combined cleaning fee is highest for the room type ‘Entire home/apt’ and lowest for shared room. This is probably due to the reason that there are a greater number of listings that are entire homes while only a very few are shared rooms.



From the above plot, we can see that when the perfect rating score is “Yes”, there are comparatively a greater number of listings that are priced higher. This is probably due to the reason that higher paid properties are well maintained and hence, gets better ratings.

2.4 Sentiment Analysis

For using textual data in our predictive models, we converted some of the textual data into numeric values. For this purpose, we used the get_sentiments() function and used the 'bing' parameter, that returns the sentiment of each word present in the dictionary. We clean the data present in the description column and pass it through this function to get the number of positive and negative words present and assign a score to each row, which is the difference between the positive and negative words. The model accuracy was 77.01 without this feature, but it jumped to 77.22% after using the sentiment score. Moreover, the importance score assigned to this variable was 737.7, which is among the top 10 features.

3. Evaluation and Modelling

3.1 Winning model

Random Forest

Random Forest is an ensemble method that combines the results of multiple decision trees to make a final prediction. It helped us to reduce overfitting and increased the generalization of the model, which was important as our target variable, i.e., perfect rating score, is a binary classifier.

Second, it handles both categorical and numerical data, missing data, and outliers. It used the process of bootstrapping by using a subset of the available features and samples for each decision tree. This helped us to reduce the impact of noisy data.

Third, it provided a measure of feature importance, which helped us in identifying which features are most relevant in making prediction. We used this information to refine the model and improve its performance.

Overall, Random Forest was well-suited for the given complex dataset as it provided good accuracy and generalization performance with a FPR of less than 10 percent.

We included a total of 63 features(24 dummies) in the winning model which are as follows:
first_review, guests_included, require_guest_profile_picture, room_type, security_deposit, charges_for_extra, host_is_superhost, host_listings_count, host_response_time, instant_bookable,is_location_exact,require_guest_phone_verification, bed_type,cancellation_policy, city_name, property_type,host_acceptance_rate, security_deposit, price_per_person, extra_people, cleaning_fee, availability_30,availability_60, accommodates, amenities_n,bathrooms,bedrooms, beds, host_since,minimum_price, requires_license,minimum_nights, maximum_nights, desc_score,density,population_2020, one_adult_no_kids_living_wage, two_adults_both_working_no_kids_living_wage, rank_2020, wifi, wirelessinternet, washer, tv, shampoo, petsallowed, lockonbedroomdoor, laptopfriendlyworkspace,kidfriendly, internet, iron, indoorfireplace, hangers, hairdryer, freeparkingonpremises, firstaidkit, fireextinguisher, dryer, cabletv, en, carbonmonoxidedetector, breakfast, airconditioning, V1, months_diff_scale

3.2 Models used

3.2.1 Baseline model

A baseline model is one that predicts all of its outputs as the majority class. In the perfect_rating_score of the Airbnb dataset, there are 70551 “NO”s and 29430 “YES”s and if a model were to predict all of the 99981 values as ‘NO’, which is the majority class, then the accuracy would be 70551/99981 = 0.7056.

This is demonstrated in the R file ‘other models.r’ in line 133-138

3.2.2 Logistic Regression

The first model that we used for the prediction was Regularized logistic regression. We used the library glmnet and function glm() to train the model. After completing the data cleaning process, we proceeded to implement Lasso regression by creating a dummy variable. To ensure a reliable evaluation of our model's performance, we further divided the training dataset into a 70% training set and a 30% validation set. In order to determine the best lambda value for our Lasso regression, we employed a grid search approach. We generated a list of lambda values using the sequence function, covering a range from 10^-1 to 10^-4 in 100 evenly spaced steps. These lambda values were selected through an iterative process of trial and error. Subsequently, we trained the Lasso model using the training data and made predictions on the validation data. Upon obtaining the best lambda value, we printed the corresponding coefficients for the Lasso model.

Moving forward, we performed a "Model Specification" process to identify the most effective set of features for predicting the target variable while mitigating the risk of overfitting. To achieve this, we randomly shuffled our training dataset and further divided it into training and validation data subsets. The logistic model was trained using these datasets.

Finally, leveraging the insights from the "Model Specification," we applied the best set of features to our training dataset once again. Employing the "Simple Partition Logistic Regression" approach, we split the training data into a 70% training subset and a 30% validation subset to train and evaluate the logistic regression model.

From the model, we could achieve a training accuracy, TPR and FPR of 71.39%, 25.98% and 97.6% respectively and a validation accuracy, TPR and FPR of 71.04%, 25.86% and 9.9% respectively.

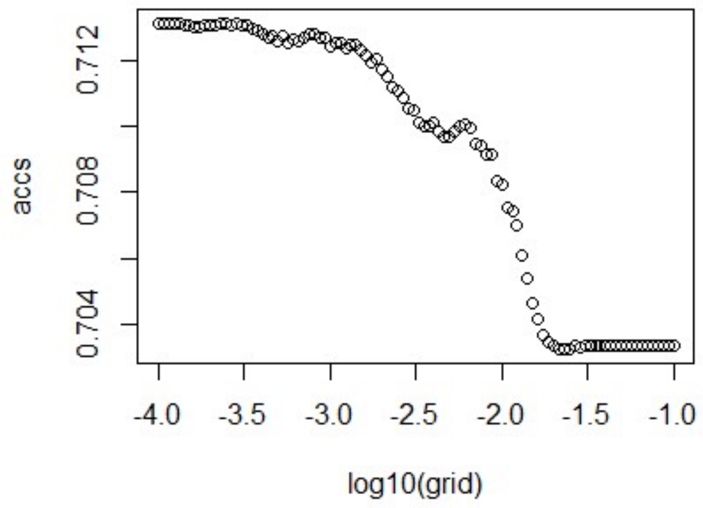
The code can be found on lines 280-361 in ‘Logistic_regression.r’ file.

The best performing features in this model were: amenities_n, price_per_person, availability_30, bed_type, cancellation_policy, city_name, cleaning_fee, guests_included, host_is_superhost,

host_listings_count, host_response_time, instant_bookable, is_location_exact, require_guest_phone_verification, require_guest_profile_picture, room_type, security_deposit, charges_for_extra, host_acceptance_rate, property_category, bathrooms, bedrooms, beds

Fitting Curves

In our logistic model, we employed L1 Regularization as a tuning technique. This regularization method introduces a penalty term on the logistic regression coefficients, encouraging them towards zero. To determine the optimal lambda value, we conducted a thorough 'grid search' where we explored different lambda values and evaluated their impact on the model's performance. After identifying the best lambda value through this search, we utilized it for subsequent analyses, ensuring that the model's coefficients were appropriately regularized.



3.2.3 Neural Network

A neural network works like a human brain, which is a set of complex neurons. It creates a complex network after learning patterns from the learning dataset. This model belongs to the deep learning family.

We have used the library ‘nnet’ to learn using the neural network algorithm and the code can be found in ‘other_models.r’ file on line 216-256. For the purpose of measuring metrics of this model, we decided to go with a simple split and split the data into 70:30 ratio for training and testing.

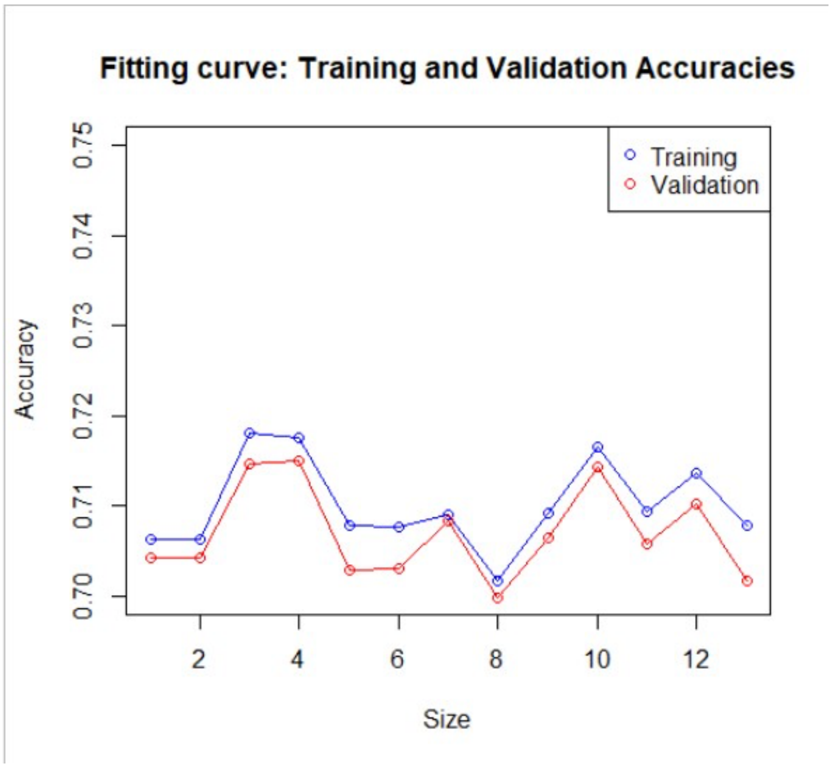
The training accuracy, TPR and FPR were 71.6%, 25.8 and 9.39 respectively while the accuracy, TPR and FPR on the validation data were 71.48%, 25.61 and 9.18 respectively.

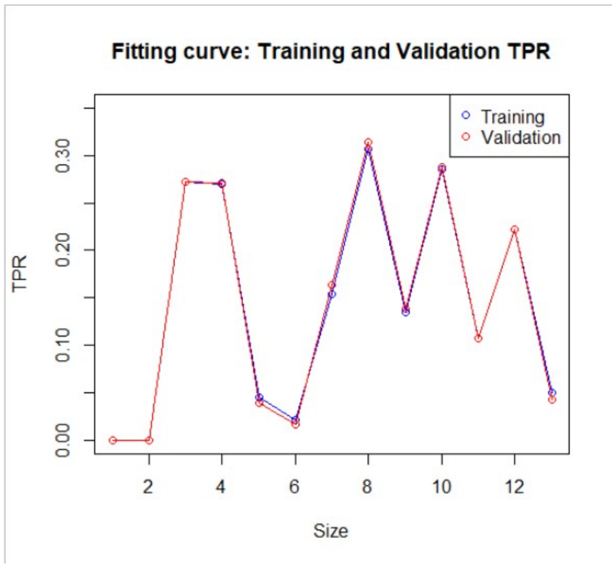
The dependent variables used are: amenities_n ,price_per_person , availability_30 , bed_type , cancellation_policy , city_name , cleaning_fee , guests_included , host_is_superhost ,

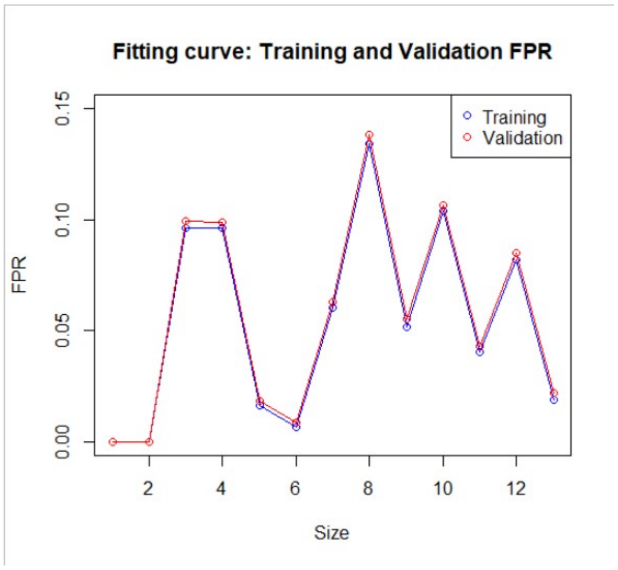
host_listings_count , host_response_time, instant_bookable , is_location_exact , require_guest_phone_verification, require_guest_profile_picture , room_type , security_deposit ,charges_for_extra , host_acceptance_rate , property_type , bathrooms , bedrooms , beds **Fitting curves**

Our neural networks have only one parameter, ‘size’. This parameter controls the number of nodes or neurons in the model and the higher the size, the model may be able to capture more complicated details, by increasing the complexity of the model.

To determine the best possible size, we passed in different values of size from 1 to 10, in increments of 1. The plots obtained were as follows:







From the graphs, we can see that none of the metrics clearly increases or decreases along with the size parameters. However, increasing the size increases the time consumed for the prediction. Hence, we limit the size to 3, the point at which we achieved the highest accuracy.

3.2.4 Naïve Bayes

Naïve Bayes uses probability theory to come up with the classification. It uses historic data to analyze what inputs lead to a particular output and calculates the probability of the outputs in validation data based on the pattern learnt. This model belongs to the probabilistic model's family.

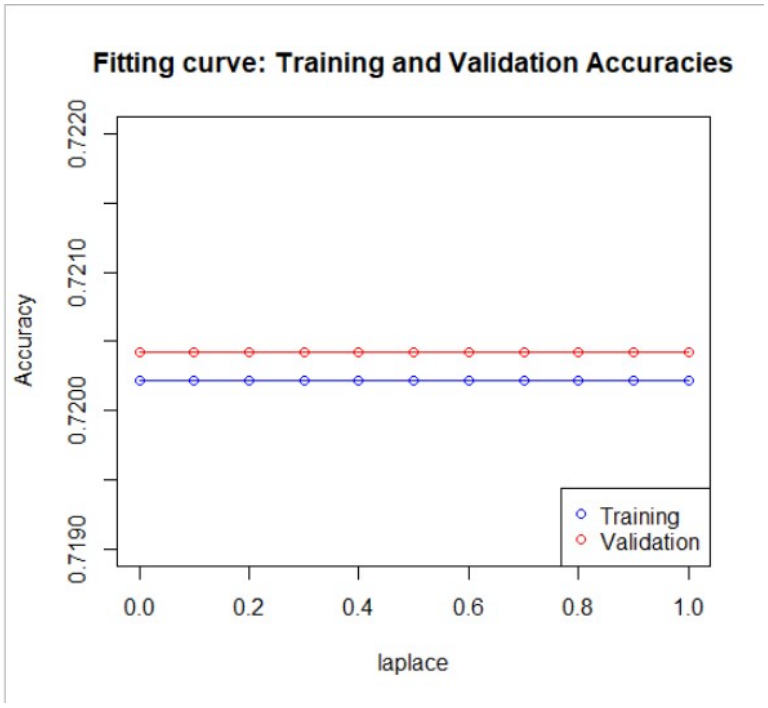
For the purpose of measuring metrics of this model, we decided to go with a simple split and split the data into 70:30 ratio for training and testing.

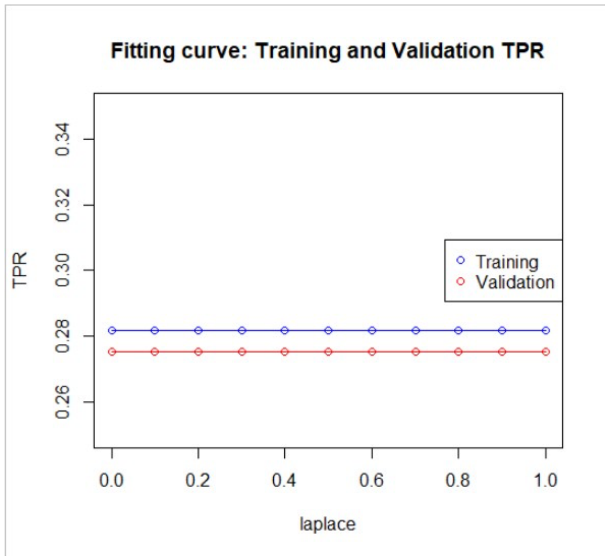
The training accuracy, TPR and FPR were 72.02%, 28.19 and 9.64 respectively while the accuracy, TPR and FPR on the validation data were 72.04%, 27.51 and 9.5 respectively.

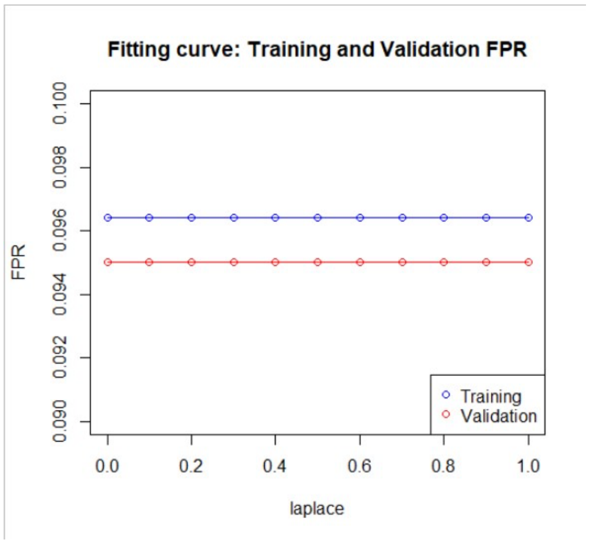
We have used the naive_bayes function in R and has been implemented in the 'other_models.r' file on line 334-369.

The dependent variables used in this model are: amenities_n ,price_per_person , availability_30 , bed_type , cancellation_policy , city_name , cleaning_fee , guests_included , host_is_superhost , host_listings_count , host_response_time, instant_bookable , is_location_exact , require_guest_phone_verification, require_guest_profile_picture , room_type , security_deposit ,charges_for_extra , host_acceptance_rate , property_type , bathrooms , bedrooms , beds **Fitting curves**

In certain cases where there is no probability due to a lack of entry in the training data, Laplace smoothing technique can be used to fill in these probabilities. The parameter 'laplace' adds a constant value to the features such that a small laplace value gives more weight to the data while a higher laplace value gives more weight to the assumptions. To find the best possible laplace value for our model, we use a list of laplaces from 0 to 1, in increments of 0.1 and plot the corresponding accuracy, TPR and FPR for both training and validation data.







From the graph, we see no significant increase in any of the metric as laplace value changes. This could be due to the fact that our dataset has no missing probablilites. Hence, we go with the default value of laplace and do not assign a value.

3.2.5 Support Vector Machine

Support Vector Machine (SVM) is a powerful machine learning model that separates the classes by drawing boundaries. It belongs to the family of discriminative models.

The r library that we used to run this algorithm is 'library(e1071)' and the function is svm(). For the purpose of measuring metrics of this model, we decided to go with a simple split and split the data into 70:30 ratio for training and testing. However, SVM uses cross validation while training and the value of k can be modified.

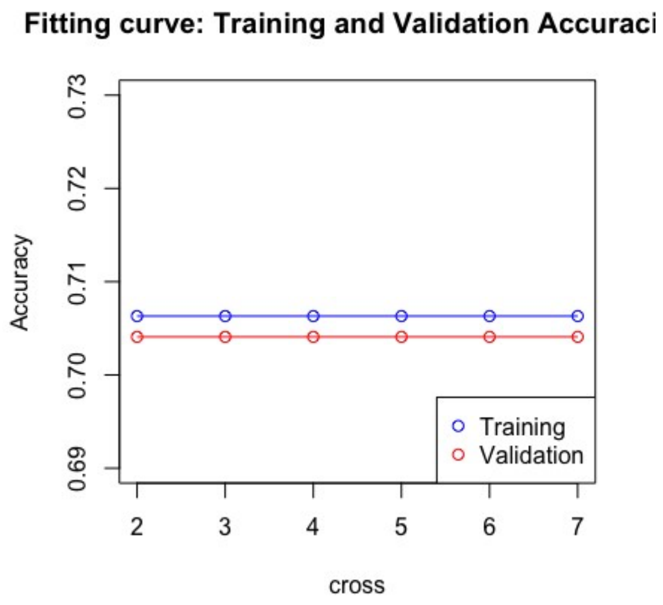
The accuracy of the model on training and validation datasets were 70.63% and 70.4%. This model takes a huge amount of time to run and since the accuracies did not show a significant improvement from the baseline accuracy, we decided to not proceed with this model.

The code can be found on lines 450-463 in 'other_models.r' file.

The dependent variables used in this model are amenities_n, price_per_person, availability_30, host_is_superhost, property_type **Fitting curves**

- **Cross**

The parameter 'cross' is the value of k in the cross-validation process. We fine-tuned the value of this parameter by passing in different values of cross from 2 to 7, in increments of 1 and the graph is as follows:



From the graph, we see that the value of 'cross' has no significant impact on the accuracies and hence, choose a value of 5.

3.2.6 Random Forest

Random Forest is a machine learning algorithm that uses decision trees to arrive at the prediction. Each decision tree uses a subset of the features available and tries to predict the outcome. Using a subset reduces the complexity and hence, tends not to overfit, giving better results. The model is based on the ensemble learning family.

The r library that we used to run this algorithm is ‘library(ranger)’. For the purpose of measuring metrics of this model, we decided to go with a simple split and split the data into 70:30 ratio for training and testing.

The accuracy, TPR and FPR obtained on the validation data were 77.21%, 44.90 and 9.17 respectively while for the training data, the accuracy, TPR and FPR were 98.44%, 94.89 and 0.08 respectively.

The code can be found on lines 349-401 in ‘Project_final_submission.r’ file.

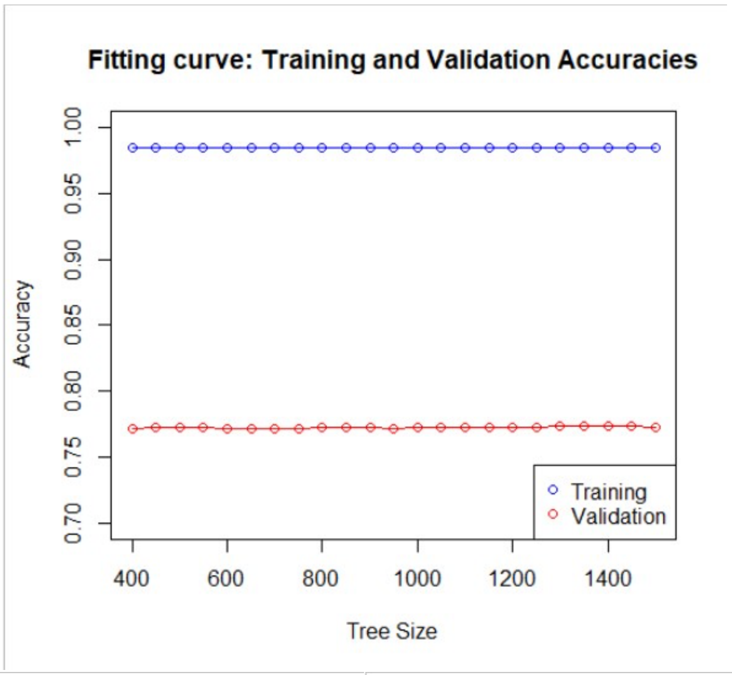
The dependent variables used in this model were: first_review, guests_included, require_guest_profile_picture, room_type, security_deposit, charges_for_extra, host_is_superhost, host_listings_count, host_response_time, instant_bookable,is_location_exact,require_guest_phone_verification, bed_type,cancellation_policy, city_name, property_type,host_acceptance_rate, security_deposit, price_per_person, extra_people, cleaning_fee, availability_30,availability_60, accommodates, amenities_n,bathrooms,bedrooms, beds, host_since,minimum_price, requires_license,minimum_nights, maximum_nights, desc_score,density,population_2020, one_adult_no_kids_living_wage, two_adults_both_working_no_kids_living_wage, rank_2020, wifi, wirelessinternet, washer, tv, shampoo, petsallowed, lockonbedroomdoor, laptopfriendlyworkspace,kidfriendly, internet, iron, indoorfireplace, hangers, hairdryer, freeparkingonpremises, firstaidkit, fireextinguisher, dryer, cabletv, en, carbonmonoxidedetector, breakfast, airconditioning, V1, months_diff_scale

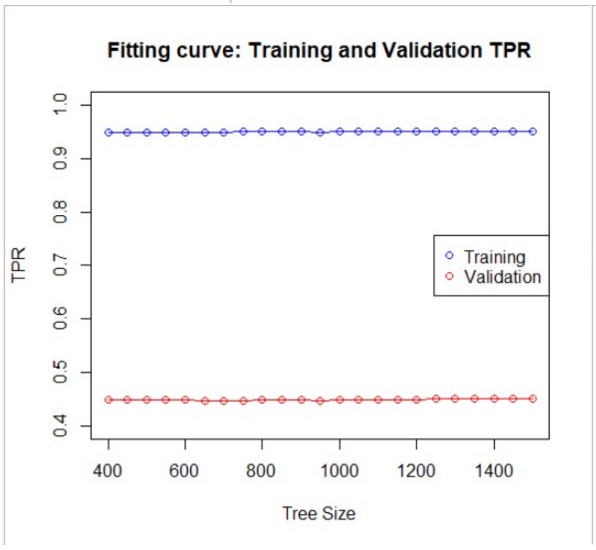
Fitting curves

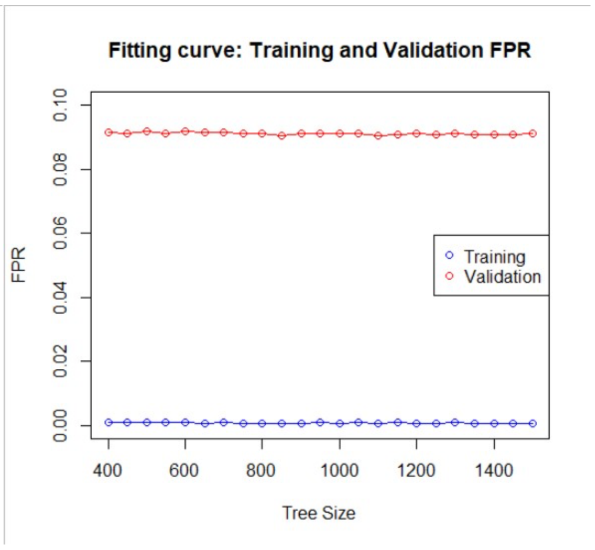
The hyperparameters used within Random forest are ‘mtry’ and ‘num.trees’. mtry is the number of random variables to be considered at each split while num.trees is the number of decision trees that are created in order to make the prediction. We tuned these parameters in order to get the best possible model.

- **Tree Size**

To analyze the best possible tree size, a curve was plotted between the training and validation accuracy of the model for various tree sizes, which is as follows which was obtained by passing in different tree sizes from 400 to 1500 in increments of 50. For each size, we noted the accuracy, TPR and FPR of both the training and validation data. The plots for the same are as follows:



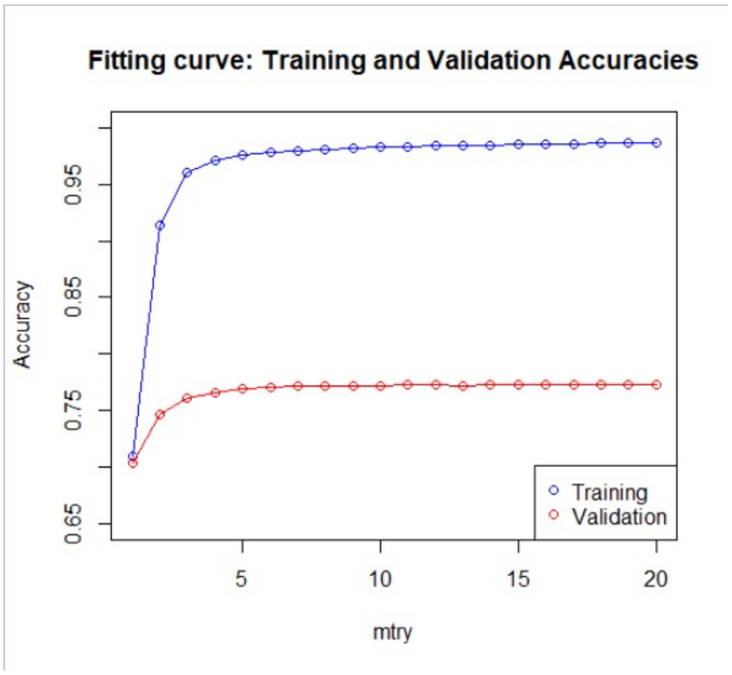


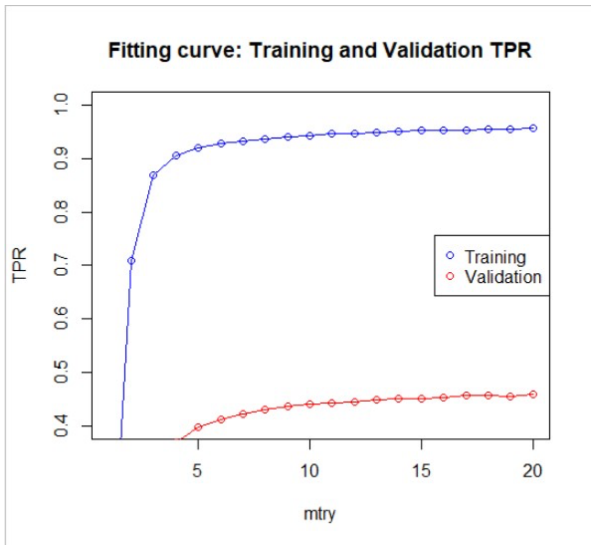


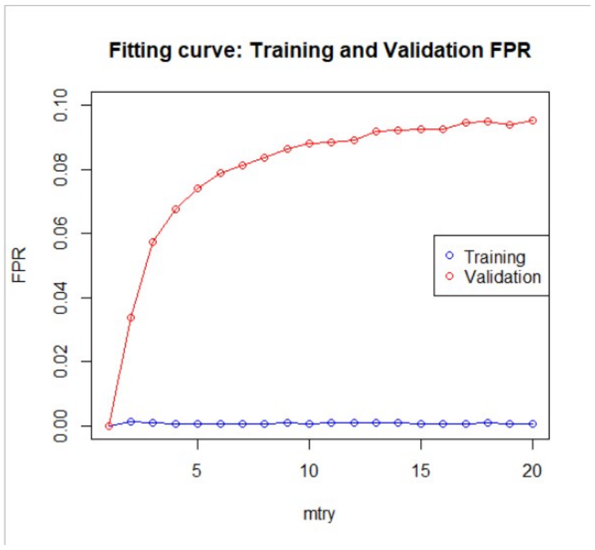
From the graphs, we can see that between the tree size makes no difference in either of the metric and hence, we shall proceed with a tree size of 500.

- **mtry**

Similar to the tree size, we also plot fitting curves to tune the 'mtry' parameter. The following plots were obtained:







From the graphs, it is evident that as the mtry value increases from 0 to about 10, the values of TPR and FPR both rise and the value of TPR remains nearly constant after this point, while the FPR has a small jump after a value of mtry=13. Hence, we cap the value of mtry at 13.

3.2.7 Bagging

Bagging is very similar to Random forest, the only difference being that there is no subset of features in this model and as a result, the value of mtry is equal to the number of variables present in the training dataset. The model is based on the ensemble learning family.

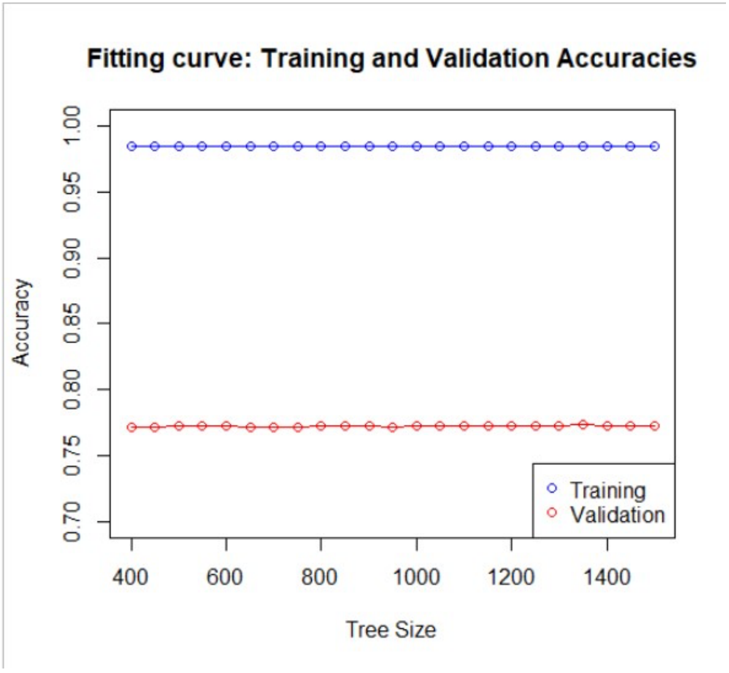
We have again used the ranger function, as in Random forest for this model. However, the mtry value is set to 63, which is the total number of dependent variables.

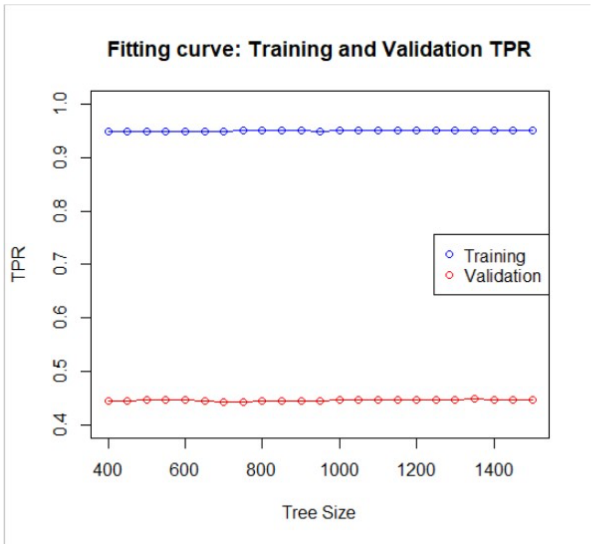
For this model, we used a simple train/test split, where the data was split in the 70:30 ratio for training and validation. The training accuracy was found out to be 99.05% and the validation accuracy was 77.08%. The TPR and FPR for training were 96.9 and 0.05 respectively and for validation were 46.07 and 9.85 respectively.

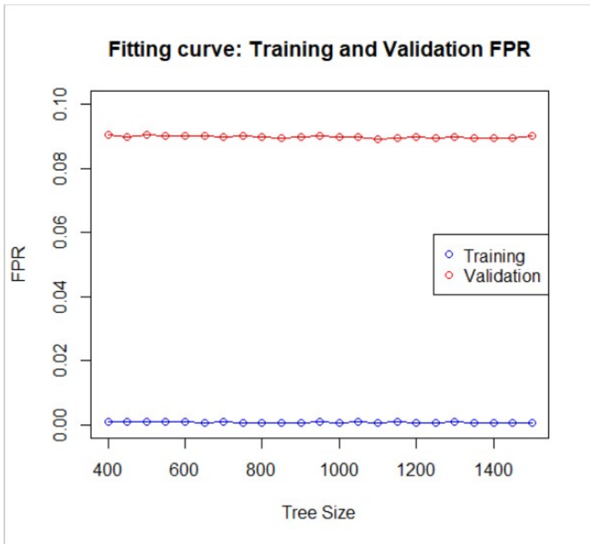
The code can be found on lines 482-527 in ‘Project_final_submission.r’ file.

The dependent variables used in this model were: first_review, guests_included, require_guest_profile_picture, room_type, security_deposit, charges_for_extra, host_is_superhost, host_listings_count, host_response_time, instant_bookable,is_location_exact,require_guest_phone_verification, bed_type,cancellation_policy, city_name, property_type,host_acceptance_rate, security_deposit, price_per_person, extra_people, cleaning_fee, availability_30,availability_60, accommodates, amenities_n,bathrooms,bedrooms, beds, host_since,minimum_price, requires_license,minimum_nights, maximum_nights, desc_score,density,population_2020, one_adult_no_kids_living_wage, two_adults_both_working_no_kids_living_wage, rank_2020, wifi, wirelessinternet, washer, tv, shampoo, petsallowed, lockonbedroomdoor, laptopfriendlyworkspace,kidfriendly, internet, iron, indoorfireplace, hangers, hairdryer, freeparkingonpremises, firstaidkit, fireextinguisher, dryer, cabletv, en, carbonmonoxidedetector, breakfast, airconditioning, V1, months_diff_scale **Fitting curves**

Since the hyperparameter ‘mtry’ remains constant for bagging, we tried different values of Number of trees to find the best possible model. The result was as follows:







Increase in the tree size has no significant impact on any of the metric, hence, we cap the tree_size at 500.

References:

[1] Living Wage - Top 100 Cities | Kaggle