# Executive Summary

## Introduction

### The Problem

Airbnb is a revolutionary platform that has been a disruptive force in the hotel industry, due to their peer to peer model, which allows hosts and customers to have a seamless operation in terms of finding short-term lodging in over 65,000 cities and 191 countries across the globe. Currently, in the US alone, there are over 660K listings. Airbnb receives commissions from two sources upon every booking, namely from the hosts and guests. For every booking, Airbnb charges the guest 6-12% of the booking fee. Moreover Airbnb charges the host 3% for every successful transaction. For a win-win situation for Airbnb and the hosts, the listings should ideally have a high booking rate. Given the infinite diversity in the types of accommodations as well as the needs/wants of guests, a one-size-fits-all formula to determine what can make a listing get a high booking rate may not be easy to derive. However, a viable solution can be obtained through the application of data mining and predictive analytics wherein data of existing listings along with the knowledge of whether they have a high booking rate or not can help in building models to a) draw inferences as to what aspects of a listing have a stronger influence on the target variable, and b) predict if a new listing will have a high booking rate or not. For this project, the preliminary goal is the latter.

### The Data

The training set has 100K records of listings described by 70 features along with the target variable about whether the respective listing had a high booking rate or not (Y = 1 or 0). A second dataset without the target variable has 12K instances for testing accuracy of models created using the training set. The data has features of numeric (price, cleaning fee, guests_included), text (description, house rules), categorical (room type, property type), categorical list (list of host verifications, amenities), and date (date of first review, date of host joining Airbnb) types.

### Our Approach

To prepare the data for modeling, we explored the distributions of different variables and their influence on the target variable, cleaned wherever we found discrepancies, and created new features using domain knowledge to better describe the data. Next, we tried logistic regression, Naive Bayes, Random Forest, and XG Boost. While XGBoost yielded the highest prediction accuracy of 83.45%, the Random Forest model was helpful in understanding which features were most useful in contributing to the prediction accuracy.

### Key Takeaways and Future Scope

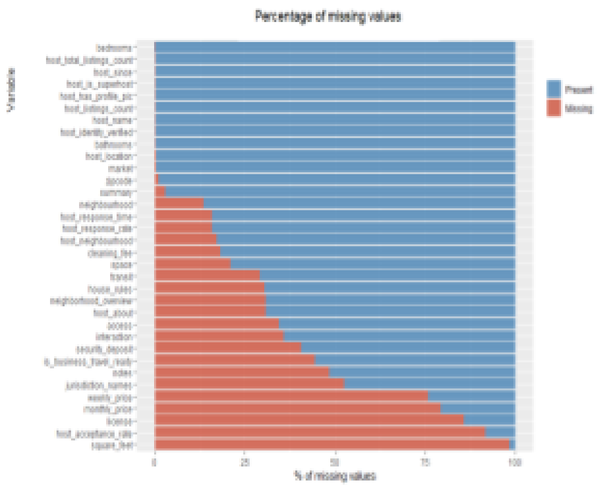
From our Random Forest model, we have found that the availability of the listing for booking within the next 30, 60, and 90 days as well as the host's status as a superhost and time taken by the host to respond are features that play an important role in predicting if a listing will have a high booking rate or not. However, to understand how changes in different features can impact the likelihood of a listing having a high booking rate, we used the output of our logistic regression model. For the inference model, we used the top 20 features from the feature importance plot of our Random Forest model. Overall, superhost status, responding to guest's queries within an hour, keeping the listing available for booking 30 days in advance, as well as making the listing instantly bookable seem to have a positive impact on the target variable (i.e., increase the likelihood of listing getting a high booking rate).

Ultimately, a combination of predictive and inference models can be used by Airbnb when a host enters the information of a new listing on the application such that, if the predictive model finds that the listing is unlikely to have a high booking rate, the application utilizes the inference model to make customized recommendations to the host of the best possible ways to increase the booking rate.

## Exploratory Data Analysis/Data Cleaning/Feature Engineering

### Missingness

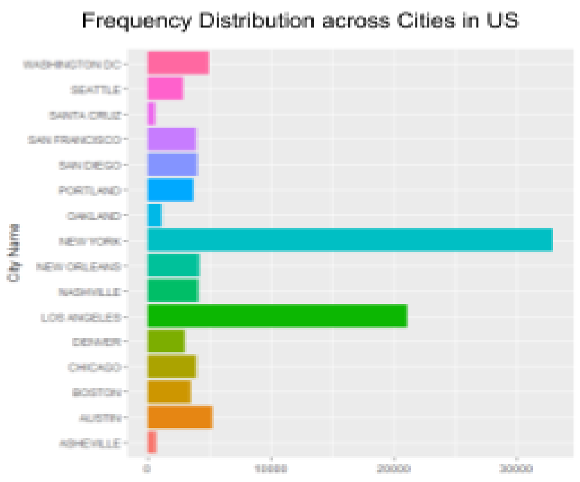
Firstly, while importing the data into R, we replaced all the blank values with NA. Next, we plotted the percentage of missing values across different variables as this will help with our planning for data processing - which features we could invest more time into exploring and possibly create new features out of; which features we wanted to make use of but has too many missing values and therefore, need to make imputations. While the majority of the columns were fully populated, there were a few like square_feet, host_acceptance_rate, and monthly_price which were almost completely empty.



Sahitya Angara

### Location Distribution

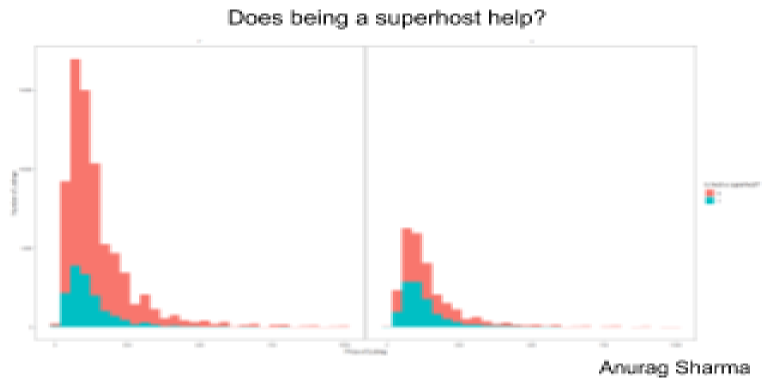
We were curious to find out how the data was distributed across the US for it could give insight about skewness. And rightly so, we found that almost 50% of the listings in the training set were located in New York and Los Angeles. This made us suspect that the prediction accuracy for test instances located in these two cities will be relatively higher.



Neethi Reddy

## The Superhost Impact

From past experience of searching/ultimately booking Airbnbs, we felt that there is an intuitive tendency to give preference to listings hosted by superhosts. So we wanted to explore the percentage of listings where the host is a superhost in both listings labelled by negative and positive classes. We found that less than a quarter of the listings of the negative class were hosted by a superhost while almost or even more than half of the listings of the positive class were hosted by a superhost.

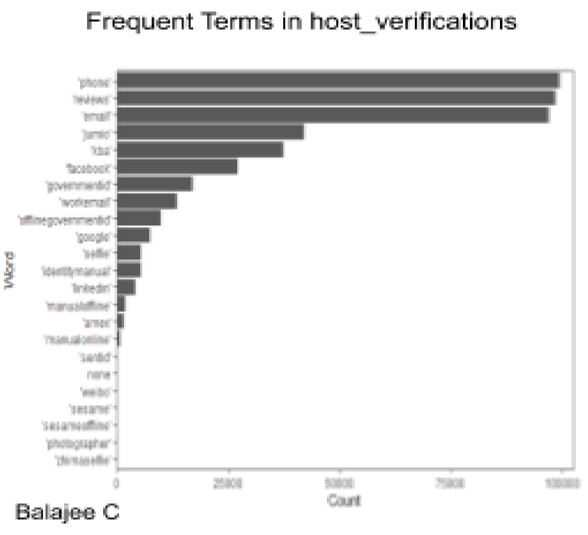


## Data Cleaning & Feature Engineering

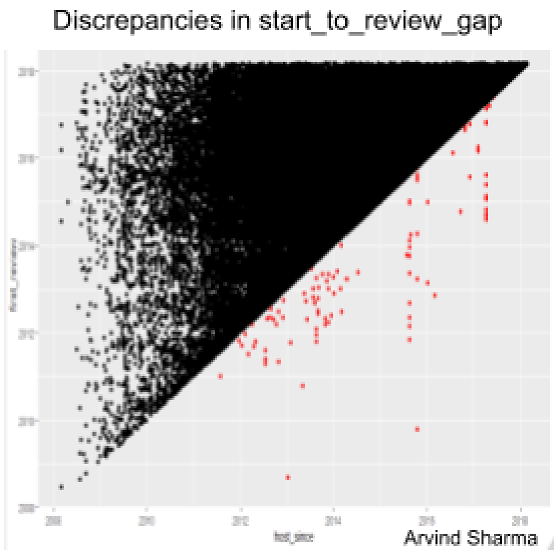
Among categorical variables, there were few like host_is_superhost and instant_bookable which assumed values of ‘t’ or ‘f’. We converted them to 1s and 0s respectively. We used one-hot encoding on a few columns with a manageable number of categories with even/reasonable distributions like host_response_time and city_name. When it came to categories like property_type, bed_type, and cancellation_policy, there were several related categories with very few occurrences so we grouped them together (using domain knowledge and some intuition) before performing one-hot encoding. For example, bed_type had 5 categories - airbed, couch, futon, pull-out sofa, and real bed. The information we essentially wanted to feed into the model from this feature was whether the bed was comfortable or not. So we created a variable which indicated if the bed was comfortable (bed_type = Real Bed) or not.

As for numerical variables, we used multiple approaches. There were cases where we created new ones by combining existing variables such as price per guest (which is (price + cleaning_fee)/guests_included), beds per guest, bedrooms per guest, and bathrooms per guest. Further, with the help of distributions across quantiles, we binned and one-hot encoded these numerical variables into new ones like low, medium, and very expensive for price per guest and cramped, comfortable, and excess for beds/bathrooms/bedrooms per guest. There were some numerical variables like availability (30, 60, 90, and 365), accommodates, maximum_nights, and price where it made sense to leave them as they are to capture the variability. However, to deal with outliers in the data, we applied YeoJohnson transformation on these columns.

We tokenized the two categorical list variables - host_verifications and amenities and found the most frequent categories for both of them. We created new binary variables to indicate the presence (or absence) of certain amenities like kitchen, tv, WiFi, and privacy (private entrance, lock on bedroom door) which could sway a customer towards the listing. A similar approach was used for the presence/absence of certain types of verifications for the host like government id, Facebook, and jumio. We did not create new variables for the phone, reviews, and email because almost all listings had them. We also created variables based on the number of verifications/amenities for each listing using quantile binning into very low, low, medium, high, and very high.



There were two variables of the date type - host_since and first_review. We thought that capturing the gap between the year that the host started out and the year he got the first review. This new variable was called start_to_review_gap. We once again performed quantile binning and one-hot encoded new variables to indicate low/medium/high start_to_review_gap. We found that there were some listings where the first_review date was earlier than the host_since date which thereby made the start_to_review_gap negative. So we corrected these discrepancies by replacing the negative values in this new column with 0 which is the most popular start_to_review_gap. We also computed the years of experience the host has by subtracting the year in host_since from 2020. Then, we once again created binary variables like host_lowExp, host_medExp, host_highExp based using quantile binning.



Out of the plain text variables, we checked for the presence of the phrase - ‘no smoking’ in the house_rules column and created a new binary variable to indicate whether smoking is not allowed or otherwise.

Our final feature set had 110 variables out of which 92 were newly generated. From the original feature set, we did away with most of the text columns owing to our limited competency in text mining and time constraints, and columns like zipcode, latitude, and longitude. We also chose city_name over state as we believed that a) the two columns are heavily correlated and b) the variability between Los Angeles and San Francisco might be missed if we choose state instead. All missing values in each column were imputed with the respective mean, median, or mode based on what made the most sense. All columns with text (including categorical) were trimmed and converted to a common case (lower or upper). The entire feature set used for building models is attached in Appendix A. The code for feature engineering is available in Appendix B.

## Model Evaluation

The aim is to find the model which can yield the best prediction accuracy. Hence, the most important metric for evaluating each model was accuracy.

### Baseline

We used the majority class classifier as our baseline model (with the majority class being 0) which gave an accuracy of 75.532%. At the very basic level, whatever model we built had to beat this baseline.

Out of the four models we tried, except for Naive Bayes, all the others were able to beat this baseline by at least 4%.

### Data Partitioning

For evaluating the models, we did a 75:25 train-validation split on the 100K training instances we had.  We fit our models on the first half of the split and used the model to make predictions on the validation set and then calculated accuracy by comparing the predictions with the actual values of the target variable for the validation instances.

### Inference?

As stated previously, the primary goal is to build a model with high prediction accuracy. However, in the larger scheme of things, inferences can provide hosts with actionable insights which can help improve likelihood of having a high booking rate.

## Modeling

As this is a classification problem, we started off with the go-to classification algorithm, logistic regression which gave a reasonably decent above-the-baseline accuracy of 80.1% for a cutoff of 0.5. We did check the accuracy for slightly higher and lower cutoffs but it was highest for 0.5. We chose to go for logistic regression first because of the algorithm’s simplicity, room for interpretability (when it comes to finding business value), and how it works well with classification problems across diverse domains. On predictions of the test instances, this model gave an accuracy of 78.02261%. We also ran logistic regression with the 20 most important features from the results of our Random Forest model to gather inferences which we will discuss in the next section.

Next, we tried Naive Bayes owing to the model's simplicity and the low run-time. However, we obtained an accuracy of 68.1% which was way below the baseline and not acceptable.

With a major proportion of time spent before the due date on data cleaning and feature engineering, we chose to go for the models known for their powerful performances - the ensemble methods - Random Forest and XGBoost. They are known to resist the temptation to overfit data and work well with large datasets that have multiple unrelated variables.

Coming to Random Forest, as mentioned previously, we did not invest sufficient time into tuning the model, except for running it for different numbers of trees. We got the following accuracies on the validation set -

| **Number of Trees** | **Accuracy** |
|---|---|
| 300 | 82.56667% |
| 700 | 83.06129% |
| 1000 | 82.53% |

With the high run-time and decreasing accuracy with increase in number of trees, we decided not to dwell on Random Forest further. However, we did want to utilize the importance function of the Random Forest library on R Studio and plotted the 20 most important features. The plot is available in Appendix C.

The next model we tried was XGBoost. This turned out to be our best model. Without tuning, the model after running for 210 iterations, yielded an accuracy of 83.267% on the validation instances and 81.30734% on the test instances.

We performed parameter tuning and got the following accuracies for different combinations of parameters -

| eta | subsample | max_depth | gamma | min_child_width | Accuracy |
|---|---|---|---|---|---|
| 0.1 | 0.75 | 6 | 0.1 | 10 | 83.2% |
| 0.1 | 0.75 | 6 | 0.1 | 1 | 83.175% |
| 0.1 | 0.75 | 8 | 0.1 | 10 | 83.365% |
| 0.1 | 0.75 | 8 | 0.1 | 1 | 83.453% |

(*We obtained optimal values of eta, subsample, max_depth, gamma, and min_child_width by running multiple combinations of values for each parameter and selected the ones which corresponded to the lowest rmse. We chose not to increase the max_depth further as we might run the risk of overfitting.*)

The parameter combination in the last row of the above table gave the highest accuracy on the validation set out of all the models we tried. On applying the model to predict on the test instances, we got an accuracy of 82.24115%.

Given the accuracy, this was our final model choice.

As much as XGBoost is reliable for generating excellent prediction accuracies, it is far too complex for making inferences between the features and the target variable.

## Interpretations

Out of curiosity to explore what variables might have more weight in swaying the likelihood of a listing having a high booking rate, we used the output of the importance function to get the 20 most important variables of the Random Forest model and used only those to run a logistic model which is better known for its interpretability. The logistic output is attached in Appendix D.

The output provides some interesting insights -

- ➔ If the host responds to a guest's queries within an hour, the odds of the listing having a high booking rate can be expected to go up by a factor of 2.13
- ➔ If the host is a superhost, the odds of the listing having a high booking rate can be expected to go up by a factor of 1.468
- ➔ If the minimum number of nights required for guests to book the listing is 1, the odds of the listing having a high booking rate can be expected to go up by a factor of 1.477
- ➔ Making the listing instantly bookable also increases the odds by a factor of 1.28.

## Potential

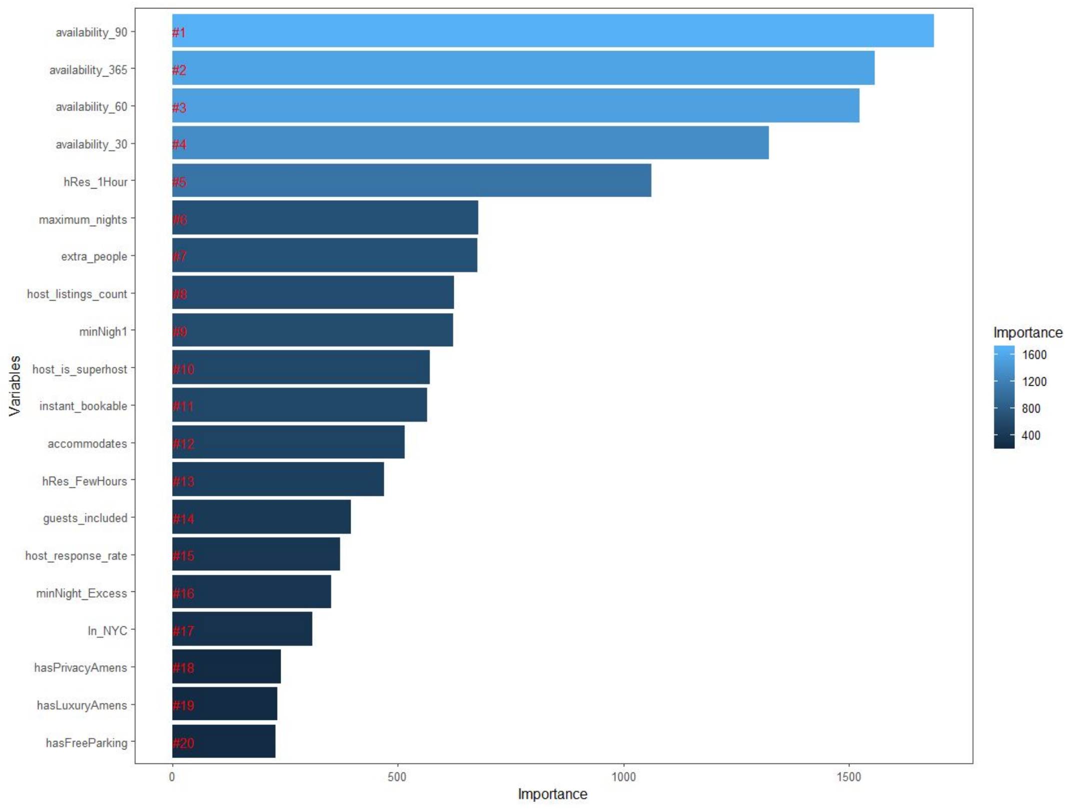
To reiterate what has been mentioned in the Future Scope section of the Introduction, such granular insights can prove to be immensely useful for Airbnb to give customized recommendations on tangible improvements to hosts to help increase their probability of a high booking rate. Further, this analysis can help in setting the initial price of a listing and/or customize promotional offers for the listings based on probability of high booking rate.

## Appendices

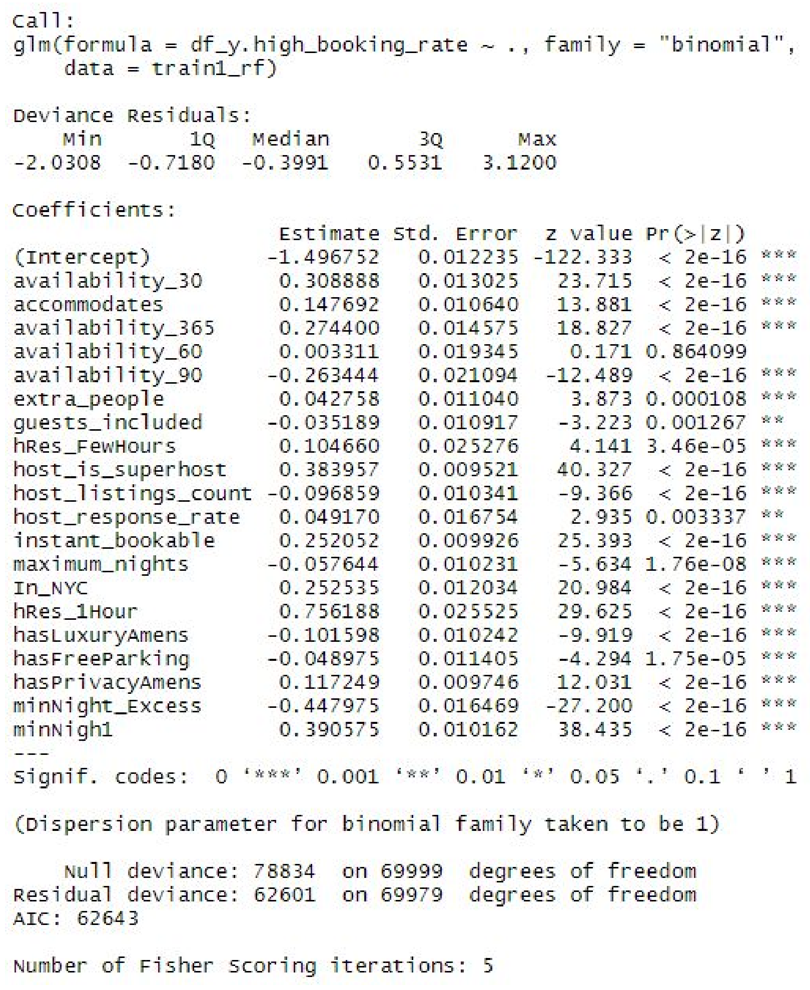
Appendix A: Feature Set used for models

Appendix B: Code for feature engineering

Appendix C: 20 most important variables contributing to prediction of high booking rate in Random Forest model



Appendix D: Summary of logistic regression model run using 20 most important features from Random Forest model.



```
Call:
glm(formula = df_y.high_booking_rate ~ ., family = "binomial",
    data = train1_rf)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0308  -0.7180  -0.3991   0.5531   3.1200

Coefficients:
                     Estimate Std. Error  z value Pr(>|z|)
(Intercept)         -1.496752   0.012235 -122.333  < 2e-16 ***
availability_30      0.308888   0.013025   23.715  < 2e-16 ***
accommodates         0.147692   0.010640   13.881  < 2e-16 ***
availability_365     0.274400   0.014575   18.827  < 2e-16 ***
availability_60      0.003311   0.019345    0.171 0.864099
availability_90     -0.263444   0.021094  -12.489  < 2e-16 ***
extra_people         0.042758   0.011040    3.873 0.000108 ***
guests_included     -0.035189   0.010917   -3.223 0.001267 **
hRes_FewHours        0.104660   0.025276    4.141 3.46e-05 ***
host_is_superhost    0.383957   0.009521   40.327  < 2e-16 ***
host_listings_count -0.096859   0.010341   -9.366  < 2e-16 ***
host_response_rate   0.049170   0.016754    2.935 0.003337 **
instant_bookable     0.252052   0.009926   25.393  < 2e-16 ***
maximum_nights      -0.057644   0.010231   -5.634 1.76e-08 ***
In_NYC               0.252535   0.012034   20.984  < 2e-16 ***
hRes_1Hour           0.756188   0.025525   29.625  < 2e-16 ***
hasLuxuryAmens      -0.101598   0.010242   -9.919  < 2e-16 ***
hasFreeParking      -0.048975   0.011405   -4.294 1.75e-05 ***
hasPrivacyAmens      0.117249   0.009746   12.031  < 2e-16 ***
minNight_Excess     -0.447975   0.016469  -27.200  < 2e-16 ***
minNigh1             0.390575   0.010162   38.435  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 78834  on 69999  degrees of freedom
Residual deviance: 62601  on 69979  degrees of freedom
AIC: 62643

Number of Fisher Scoring iterations: 5
```

Appendix E: All charts - Maximized for better viewing