Airbnb High Booking Rate Predictions

The goal of the project was to predict the highest booking rate for Airbnb listings. Therefore, this report summarizes the initial data exploratory findings and the various models implemented and tested to obtain the highest accuracy.

Exploratory Data Analysis/ Feature Engineering

The dataset provided had nearly 70 features, each feature uniquely describing the listings. To analyze relationship among various predictors and discover hidden trends, we visualized percentage of weekly and monthly listings, average price per night of listing per city, average price per night for each property type, number of listings per state, number of business travel ready listings per city, number of listings vs host_response_time. From these visualizations, we found interesting facts about the data set. For example, New York had the maximum number of business-ready listings which indicates that New York is a business hub. Also, Austin, capital of Texas had the highest average price per night.

Before implementing any model to predict our target variable, we wanted to cleanse our data. By selecting important features based on our intuition, we checked for missing values and anomalies. We observed that variables such as cleaning_fee, price had NAs and outliers, so we imputed those values with mean and MICE package which leverages a predictive mean modelling method for numerical variables. Additionally, we replaced all the negative values in minimum_nights with 1 and all values greater than 1125 in maximum_nights with 1125.

Furthermore, to avoid overfitting and reduce the data file size from noisy data, we binned a few features. For example, we binned host_response_rate into three particular bins (Low, Mid, High). We performed text analysis on access, transit, interaction, neighborhood columns. The access text had the information containing which parts of the listing can the guest access (eg. living room, kitchen, bathroom), so we retained these keywords. Similarly, for variables like ‘neighborhood_overview’, ‘transit’, ‘interaction’, we retained the keywords that were relevant to the listing using 90% as the sparseTerm cut-off and getting rid of the stopwords.

We removed some variables having majority empty strings and missing values because they did not add any value to our models. A few of these features were Is_business_travel_ready, security_deposit, require_guest_profile_pic. Some features had a high correlation with another such as beds and accommodates, so we decided to remove beds and keep accommodates instead.

We created a few binary variables like monthy_available (full month booking), weekly_available (full week booking), Is_host_here (host resides in same city), guest_cost_extra (extra cost for additional guests) and numerical variables like host_verifications_count (number of host verifications), amenities_count (number of amenities per listing), host_age (current date - host since). We created these variables to learn how the factors logically affect the listings and produce the inferences from the data.

At last, we added interaction terms to our model: price_per_guest (‘price’ / ‘guests_included’), beds_per_guest and bathrooms_per_guest. We also added what percentage of the price is the security and cleaning fee respectively. These would logically affect the quality and pricing of a listing.

Modeling

With the target variable high booking rate, our overall goal was to determine a binary outcome. In our first modelling process we tried a logistic regression because our target variable was a categorical value and we can model the probability of a listing being high or not. Also, another valid reason we tried logistic regression was the simplicity of the coefficient relationships within the model. It would have been easier for us to communicate the business value of each feature from the logistic regression.

To control for variables that may not have been as informative for the predictive model, we thought we would run a L1 and L2 regularization logistic regression so that the model would give a penalty for larger coefficients. We performed this model in order to prevent overfitting. To tune for the penalty value, we tuned both the L1 and L2 model's lambda by creating a list of 100 lambda values between 10^10 and 10^-2 and allowed R to determine which lambda value gave the best accuracy. Between the L1 and L2 models, the L1 had an accuracy of 78.6% which was slightly better than the L2 model. L1 is shown in the appendix (Appendix C). The best lambda for the L1 model was 0.01. This model was obtained by using a cutoff of 0.5, which was determined as the best cutoff from a standard logistic regression model that was run.

Intuitively we decided that the ensemble methods generally perform better because of the randomness in picking up variables and its ability to decrease variance without increasing bias. Random Forest is one good example of a bagging method. The only downside of Random Forest is that it is very hard to interpret. So, we ran a logistic on top of random forest to see how each variable affects 'high_booking_rate'.

Therefore, we decided to use the random forest as our final model. One of the advantages of random forests is that it performs well with many instances. And with 100,000 instances we can potentially achieve a steeper learning curve before it eventually plateaus. Now, with many features and also many rows the data has a lot of noise. But, as an ensemble method, random forests implement bootstrap sampling which resamples and replaces data while using a subset of the overall features. Thus, random forests randomize the trees, create independence and therefore increase overall performance. We tuned our random forest using feature importance functions and plots showing us which feature significantly decreased the accuracy if removed.

Model Evaluation

After cleaning we combined our unlabeled feature data frame to the labels and in total, we have 100,00 rows with 159 features in our dataframe. This can get computationally intensive so we split our training data 25% validation and 75% training. We tried combinations of features in our modeling techniques to get the best model. When we first started modeling, we used a reasonable number of features to make our model concise, however, that came at the expense of our feature list being relatively small, and thus our models did not perform as well. We then tuned our feature selection to find the best possible features for a high performing model while also giving insight into some feature importance for business value. We used less than half of the total feature list with around 55 features included in our models.

With 55 features, most of our models had accuracies above the baseline of 71.4%, which assumes the majority class of all zeros. However, our best model was the random forest which received an accuracy of 86.8% on the validation data. Eventually, we decided that the random forest was our best model and we used that model instead of our logistic regression models. There are some advantages to choosing the random forest besides the accuracy, including leveraging modeling techniques such as bootstrap sampling and decorrelation/independence. Using bootstrap sampling, the random forest iteratively resamples and replaces and thus it quantifies any unknowns and reduces variance. Also, another advantage of random forest is the decorrelation of classification trees by randomizing trees and thus keeps the bias of our model low. We should note that random forests are not perfect since they can be hard to interpret and are prone to overfitting like any other model. However, running a random forest on our data not only offered the best predictions but also offered practical values on the Airbnb listing by grouping the best features.

Baseline accuracy (assuming the majority class of 0): 71.4%

Model Accuracies in comparison:

- Standard logistic regression: 75.6%
- L1 logistic regression: 78.6%
- Random Forest:
  - 73.8%
  - 79.4%

- **Final Random Forest on validation data**: 86.8%

Business Value

While these predictions are important for Airbnb and the hosts, other businesses can benefit from these predictions as well. For example, by grouping Airbnb's by their location, and seeing which regions have the most number of Airbnbs with a 'high_booking_rate', another company, **ZipCar,** which rents out cars for driving, can then place cars near those hotspots. One of the biggest advantages is that **ZipCars** have their own parking spots and once the rental period is over, the car is parked back in its spot ensuring its booking availability.

After evaluating our model and tuning, we concluded that the Random Forest was making the best predictions. Random Forest also gives out the feature importance (Appendix D). We took the most important features from the plot and ran a logistic model on them. We can see that 'high_booking_rate' is highly influenced by how the host responds.

For Airbnb hosts and property owners, this information is very useful. They can be informed about the variables that most affect their booking rate. If they respond within an hour their 'high_booking_rate' goes up by a big factor.

This model can also be implemented as a part of the Airbnb host dashboard by the engineers at Airbnb. They can use the Random Forest model to determine if listing will have a 'high_booking_rate'. They can further use the Logistic Regression model and its coefficients to look at which features to improve upon.

In general, the advice we would give to the hosts is to increase the number of days prior to the booking date that the property is available to rent. In addition, hosts should respond quickly to the customer's inquiries, provide simple descriptions for listings and upload a picture of themselves.

Appendix

Appendix A

```
rfMod <- randomForest(high_booking_rate~ accommodates + availability_365 + availability_60 +
                      bathrooms + availability_90+ availability_30 + bedrooms + bed_type +
                        host_identity_verified + host_is_superhost +
                        host_listings_count + maximum_nights +
               minimum_nights + amenities_count + host_verification_count + host_age + review_age +
price + guests_included + extra_people + property_type + instant_bookable + cancellation_policy +
is_location_exact + requires_license + cleaning_fee_zero + price/guests_included +
bedrooms/guests_included + bathrooms/guests_included + host_response_rate_binned + marketCheck + help
+ need + phone + questions + available + around + block + blocks + easy + minute + minutes + subway +
walk + walking + private + use + will + bathroom + entire + kitchen + living  +  PriceMean +
security_deposit_zero/price + weekly_available + monthly_available + cleaning_fee_zero/price +
security_deposit_binned + host_response_time,
                      data=airbnbTrainDF,
                      mtry=3,
                      ntree=1000,
                      importance=TRUE,
               na.action = na.roughfix
               )
```

Appendix B

```
model_log <- glm(high_booking_rate~ availability_365 + bathrooms +
                    bedrooms + host_identity_verified + host_is_superhost +
                    host_listings_count + maximum_nights +
                    minimum_nights + amenities_count + host_verification_count + review_age,
                    data=airbnbTrainDF,
                    family = 'binomial')

log_train_preds <- predict(model_log, newdata = airbnbTrainDF, type="response")
log_valid_preds <- predict(model_log, newdata = airbnbValidDF, type="response")

class_performance(confusion_matrix(log_valid_preds, airbnbValidDF$high_booking_rate, 0.5))[1]
```

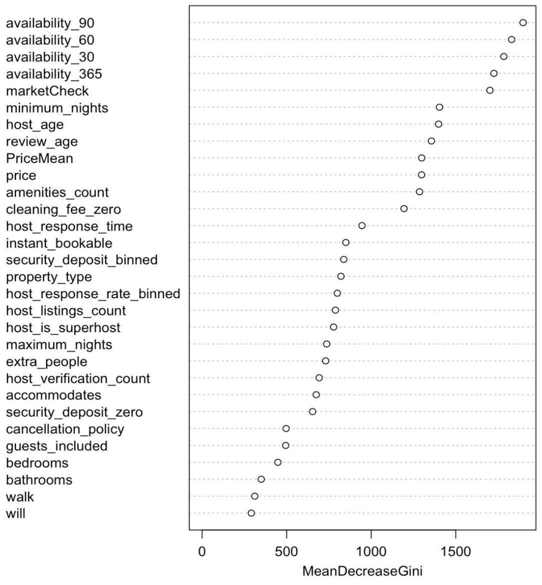
Cutoff	Accuracy
0.5	0.7560712
0.65	0.7530306
0.7	0.7510302
0.4	0.7460292
0.55	0.7553111

Appendix C

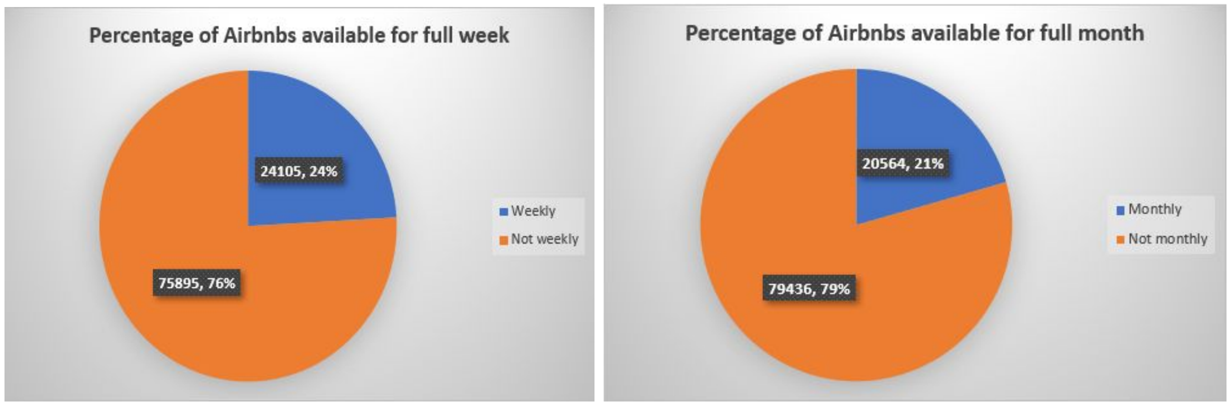
```
# options(na.action="na.pass")
airbnb_x <- model.matrix(high_booking_rate~ accommodates + availability_365 +
availability_60
             + bathrooms + availability_90+ availability_30 + bed_type + bedrooms
             + bed_type + host_identity_verified + host_is_superhost +
               host_listings_count + maximum_nights + minimum_nights +
               amenities_count + host_verification_count + host_age + review_age +
               price + guests_included + extra_people + property_type +
               instant_bookable + cancellation_policy + is_location_exact +
               requires_license + cleaning_fee + price/guests_included +
               bedrooms/guests_included + bathrooms/guests_included +
               host_response_rate_binned + marketCheck + help + need + phone +
               questions + available + around + block + blocks + easy + minute +
               minutes + subway + walk + walking + private + use + will + bathroom
             + entire + kitchen + living + security_deposit + weekly_price_binned +
               monthly_price_binned,
                    data=airbnbTrain)


airbnb_y <- airbnbTrain$high_booking_rate
```

```
grid <- 10^seq(10,-2,length=100)
# grid
# seq(10,-2,length=100)
k <-5
cv.out.lasso <- cv.glmnet(x_train, y_train, family="binomial", alpha=0, lambda=grid,
nfolds=k)
# cv.out.ridge
bestlam <- cv.out.lasso$lambda.min
bestlam

lasso.pred <- predict(cv.out.lasso, s=bestlam, newx=x_valid, type='response')

lasso.logistic.matrix <- confusion_matrix(lasso.pred, y_valid, 0.5)
lasso.log.accuracy <- class_performance(lasso.logistic.matrix)[1]
lasso.log.accuracy
```

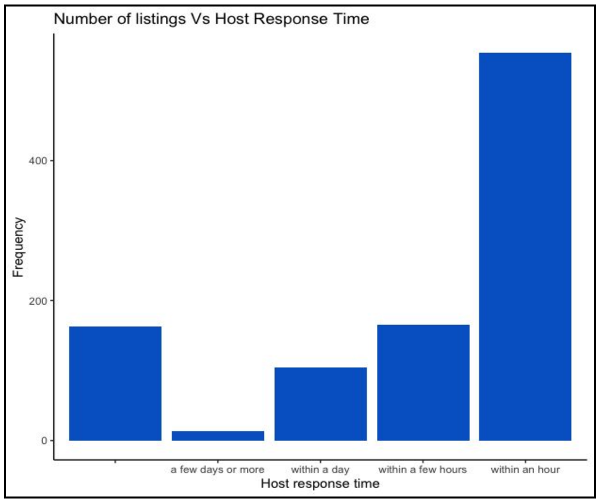
Appendix D

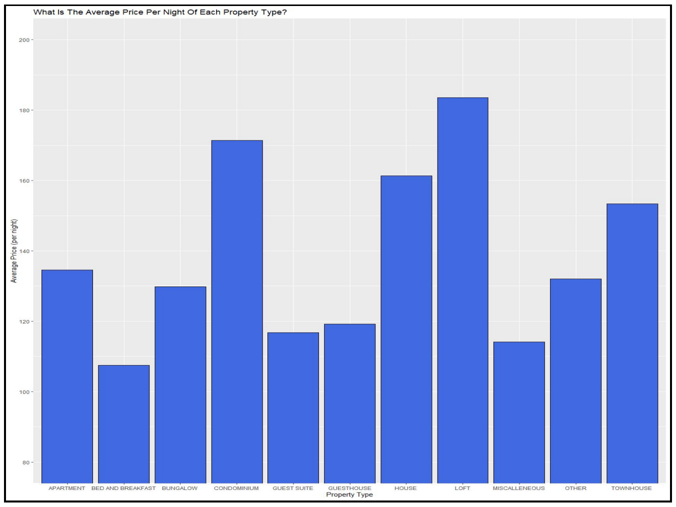


Signed Visualizations



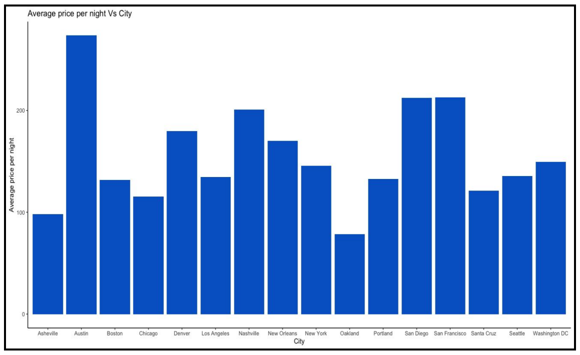
Neel

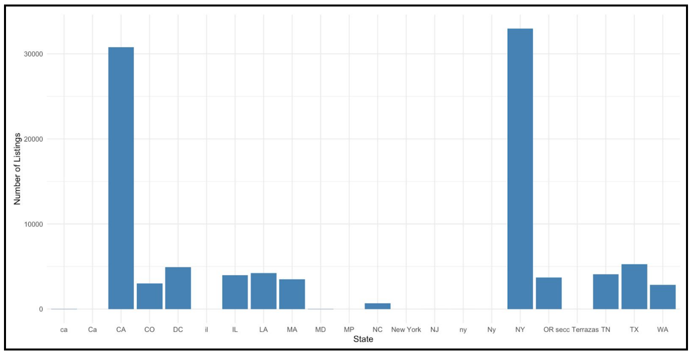




Samantha

Marc





Juhi

Abhishek