Airbnb Listing Success Prediction

Collin Mui
Master of Science Business Analytics (Student)
California State University East Bay
Hayward, USA

Kanmani Natarajan
Master of Science Business Analytics (Student)
California State University East Bay
Hayward, USA

Madison Schiller
Master of Science Business Analytics (Student)
California State University East Bay
Hayward, USA

Matthew Smithey
Master of Science Business Analytics (Student)
California State University East Bay
Hayward, USA

I. Introduction

Airbnb changed the hospitality market in a major way by introducing a platform where homeowners can rent out their space. On the other end, people can search through this platform to book those listings with ease. This has become a popular alternative compared to booking a hotel / motel since it gives guests more options whether it be location, room size, price, etc. Airbnb has millions of listings around the world and that number is growing every day. A major part of the growth of the company is having a large variety of listings for consumers to choose from. So Airbnb needs to promote new hosts to sign up and add their homes.

II. Problem / Research Question

When a new host creates a listing for the first time, they must research other listings and compare location, size, amenities, etc. to determine a price that will sell. This research can deter potential hosts away. With so many listings, Airbnb has collected a large amount of data. All this data can be used to help the host and answer the question below.

Will a listing be successful based on the specifications given?

If there is a tool or model, it can help answer that question. This can result in a new host having the confidence to post, which will benefit the consumer by having more options to pick from and the company since they make their money by taking a percentage from the total cost.

III. How This Will Be Done

To answer the question above, we will use web scraped listing data off Airbnb's website. Since there are millions of listings, it will require a lot of computer power and time, so the focus is on listings in San Francisco. This model can be applied to any location around the world for future use. In order to use the data as the source of our models, the data needs to be cleaned. Once that is completed, the data will be explored to discover patterns or valuable information at a high level. For this project, three models have been chosen which are Logistic Regression, Random Forest / Classification, and Deep Neural Network. These models will predict a binary value: yes the listing is successful or no it is not. Then the models accuracy scores will be compared and the model with the best score will be the optimal model which will be used to predict a sample listing.

IV. Data / Data Preprocessing

A. Data

The data was web scraped off the Airbnb site by a site called “Inside Airbnb”[1]. The scraping was done on 1/6/22. Luckily, we were able to use their service to have a large dataset for our project as finding “good” data was a challenge. The dataset consists of 74 columns and 6413 rows. Each row is an individual listing posted in the San Francisco Area. Each column is a specific attribute of the listing such as price, size, location, amenities, beds, etc. One of the biggest challenges of this project was preprocessing the dataset. At a glance, there was a lot of information which is why it was chosen but after really exploring the data it wasn’t very clean.

B. Data Preprocessing

The first step of the process was looking at the data and getting a better understanding of what could be used for our analysis. Majority of the variables didn’t add value to our models, some of the variables were URLs, names, dates, etc. Once removed, the dataset was then reduced from 74 variables to 42 variables. Next was focusing on dealing with the missing data. The first approach was dropping rows that had missing values. This was chosen first because it was easy to do. After dropping the rows, our dataset was reduced significantly to the point where we felt the models would be affected. The alternative approach was to go through each column and fill in the N/As with a value whether it be the mean, zero to represent no value, or false. The value was determined by two factors:

type and meaning. Types of values could be categorical, numerical, etc.

Another step of the process was simplifying a few variables to fit our model. This was done by converting a few categorical variables into binary. One example is the variable host_location, this column showed where the host was located. There were multiple cities listed but the idea was to see if living in the area you are hosting would be better for your listing so we replaced it with a yes/no value, yes meaning living in SF and no meaning not living in SF.

Majority of the variables in the dataset were straight forward, with one single value for each cell. There were a couple of variables that had a list for each row. An example would be amenities. Each listing had its amenities in free text format. The frequency count of each amenity is calculated, and some redundant amenities are removed. Out of all the amenities, the count (presence) of the top 20 most frequent amenities in each listing is taken as one of the input features for our model. This method was applied to another variable host verification. The count (presence) of the top 11 most-frequent host verification methods is taken as one of the input features for our model.

Lastly, an dependent variable was needed for models. The question being answered in our project is how successful a listing will be? Keeping that in mind, there needed to be some measure to show if listings are doing well or not. This was created by using the four columns which are the availability for the next 30,60,90,365 days. These variables were good indicators to see if the listings were being booked and how often. Using these numbers, we created an equation (shown below)

*Success Rate = 1-((0.25 * "availability_30"/30)+ (0.25 * ("availability_60"-"availability_30")/30) + (0.25 * ("availability_90" - "availability_60")/30) + (0.25 * ("availability_365" - "availability_90")/275))*

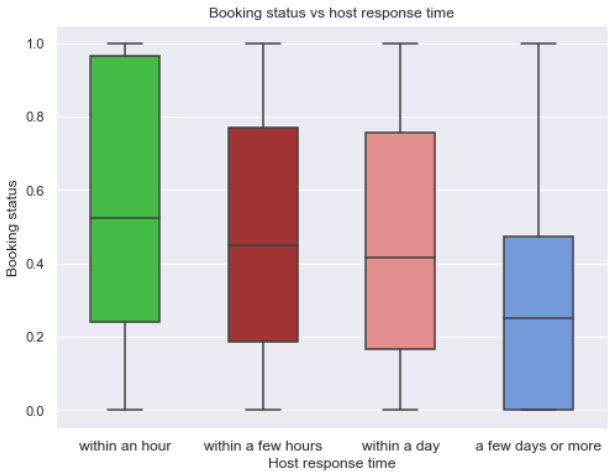
This gave us a success rate on each listing with a minimum value of zero and a maximum value of one. The threshold rate that we considered successful was anything above 0.5. This was then used to create the binary variable successful or not.

V. Feature Selection and Dimensionality Reduction using Statistical Testing

Airbnb dataset had too many features and some of them may be redundant, some of them may be highly correlated and some may be irrelevant. If all the features are used, it will take a lot of time to train the model, and the accuracy of the predictions may also get reduced. Hence it was decided to select features that have a significant effect on the model's output. Statistical testing is used to select the features for our model.

Let's take a look at statistical testing on host response time to check whether host response time has a significant impact on booking status. Host response time is a categorical variable with 4 types of responses (within an hour/ within a few hours/within a day/ a few days or more). The procedure we follow is as follows.

A. Create a box plot:



From the above box plot, we can observe a significant difference in booking status between 4 different host responses. The booking status of hosts who respond within a few days is not that appealing compared to the booking status of hosts who respond within an hour.

B. Check If The Variance of Different Responses are the same:

As a rule of thumb, we check if the ratio of variance is below 4, with the larger variance in the numerator. If the ratio is less than 4, we can assume that the variance of the two samples is the same. We can also run an F test to confirm the equality of variances. But it is not recommended.

In our case, even though the variances of the different samples are the same, we still prefer Welch's t-test as it is more robust than Student's t-test and maintains type I error rates close to nominal for unequal variances and for unequal sample sizes under normality [2].

T-test Assumptions

- The independent variable is categorical with at least two levels or groups
- The dependent variable is continuous which is measured on an interval or ratio scale
- The distribution of the two groups should follow the normal distribution

Here we assume that the two groups are normally distributed as the number of samples is greater than 50.

C. *Hypothesis Testing*

- *Null hypothesis:* There is no effect of host response time on the booking status
- *Alternate hypothesis*: Host response time influences booking status

Running four tests on the same data increases the probability of observing some significant results from 5% to 19%. To remedy this problem, a Bonferroni correction factor is applied.

	Within an hour	Within a day	Within a few hours	A few days or more
Within an hour	1	7.54×10^{-16}	6.901×10^{-11}	3.42×10^{-18}
Within a day	7.54×10^{-16}	1	7.238×10^{-3}	1.25×10^{-6}
Within a few hours	6.9×10^{-11}	7.23×10^{-3}	1	1.29×10^{-10}
A few days or more	3.42×10^{-18}	1.25×10^{-6}	1.29×10^{-10}	1

Table: Welch T-test with Bonferroni Correction Factor

The p-values are far less than 0.01 (Here we assume, alpha=0.01) for all the response types. Hence, we reject the null hypothesis.

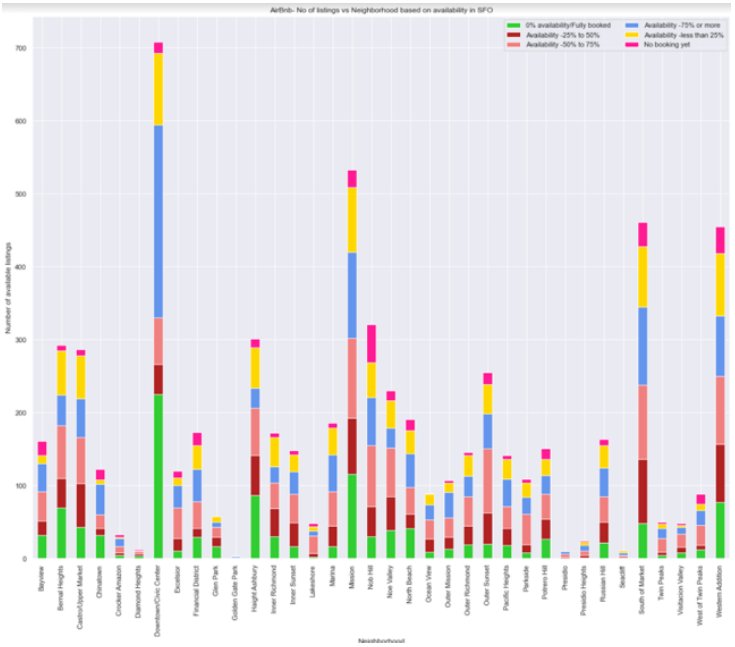
To conclude, all the response types are statistically significant in the above case. We can say with 99% confidence that the host response time is linked to the booking status and only a 1% chance of coincidence.

In a similar way, statistical tests were performed on host status, Neighborhood, and room type. The response types were grouped together if their p-values were statistically insignificant. By this method, neighborhood types got reduced from 36 to 2 and response type for bathroom were reduced from 30 to 5.

VI. Data Exploration

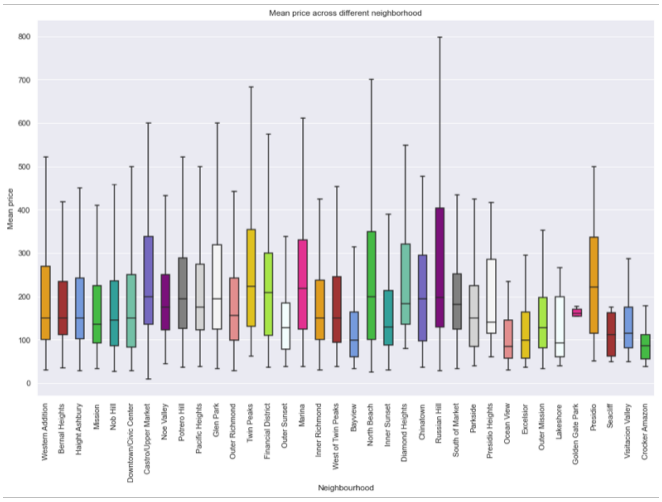
The goal in exploration was to uncover more information and visualize the data. Below are a few visualizations chosen to help see if there is a pattern on why different listings are successful.

A. *Number of Listings vs Neighorhoods*



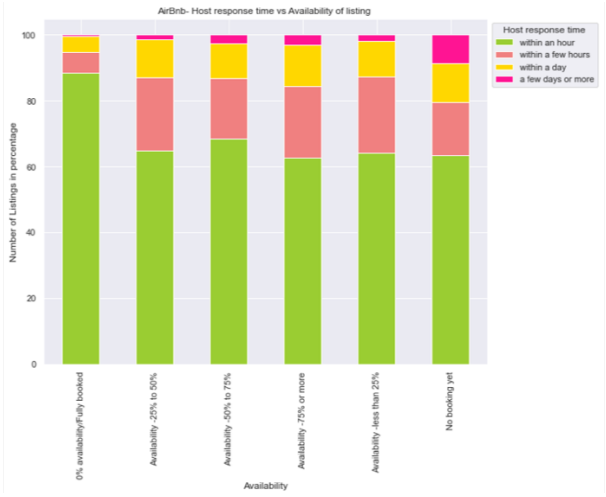
Above is a stacked bar chart showing the number listings for each neighborhood. The different colors represent the count of availability of listings within the neighborhood. When deciding what is an important factor in determining the success of the listing, one of the first thoughts that came to mind was location. Breaking it up by neighborhood allows us to see which area has the most listings and how successful they are. Downtown/Civic Center has the most number of listings compared to the rest with Mission having the second most. Focusing on the booking availability, Downtown/Civic Center has the most fully booked listings with Mission being second. This shows that these two locations are desirable places where consumers want to stay. Our assumption is that the locations tailor to tourists who are visiting the city because they are near the tourist attractions the city offers.

B. *Mean Price vs Neighborhoods*



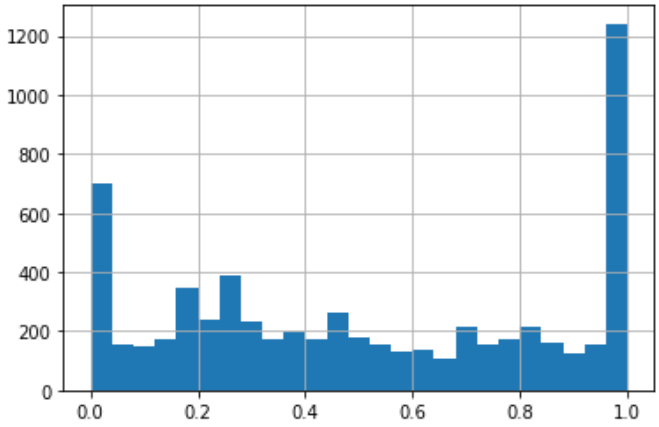
Since location does have some effect on the numbers of listings that a neighborhood receives, we wanted to see if the price correlates with the popular locations shown in the bar chart. The boxplot above shows the price per night in each neighborhood. Focusing on the mean price of each neighborhood, Presidio has the highest price with Twin Peaks having the second highest. Presidio and Twin Peaks are very nice areas where home prices are very high. Surprisingly, Downtown/ Civic Center and Mission prices are average compared to the rest of the neighborhoods. Even if the location is very popular for tourists to book an Airbnb, it seems that the price has more correlation to the neighborhood comp price.

C. *Host Response Time vs Availability of Listing*



The next focus was to see how a host can have an effect on the frequency of a place being booked. One major part of being a host is the ability to respond to booking requests and questions the guest has. The chart above shows a 100% stacked bar chart divided by availability with the percentage of each host response time value. The "fully booked" category for avaiability has the highest within an hour response time compared to the rest of the availability variables. Also, "no booking yet" has the highest number in the "few days or more" category for response time. This shows there is some correlation to response time and bookings.

D. *Histogram of Success Rate*



The success rate is the target variable. It is unevenly distributed, weighing heavily on opposite extremes, either being 100% successful or 0% successful. Over 1200 units have a success rate of 1.0, corresponding to 100% booking. About 700 have a success rate of 0.0. The various intermediate rates between 0.0 and 1.0 range from 100 to 400 units and are unevenly distributed. Other than weighing heavily on the extremes, there is no clearly discernible pattern to the success rates.

E. *Variable Correlations*

Based on correlations between predictor variables and the target variables. A few recommendations can be made for persons who are listing their property on Airbnb. First and foremost, they should keep their rental unit clean and respond promptly to renters. Cleanliness and response rate were the 3rd and 5th highest-ranking predictor variables in terms of correlation. Review scores on communication ranked 4th, which indicates that not only the rapidity of responses is important, but also the quality. Good spelling, grammar, etiquette, and clarity in communication on the part of the property owner could help improve success rates on their listings. If someone is able to list an entire home or apartment, that can also increase success rate, though that may not be an option for some folks. Being a superhost, providing childrens' books and toys, and having a license can also help to increase chances of success. These last three fill out the 8th, 9th, and 10th positions in rank based on correlation to the success rate.

Rank	Predictor Variable	Correlation
1	review_scores_rating	0.127119
2	review_scores_location	0.120320
3	review_scores_cleanliness	0.118876
4	review_scores_communication	0.116001
5	host_response_rate	0.114939
6	review_scores_checkin	0.111098
7	room_type_Entire home/apt	0.101002
8	host_is_superhost_t	0.090050
9	children\u2019s_books_and_toys	0.087207
10	licensed_1.0	0.086234

Table: Predictor variables with highest direct (positive) correlation to the target variable

In addition to these recommendations about what to do, there are also a few about what not to do. These are features that are likely to decrease the overall success rate of a listing in Airbnb. Minimum number of nights is the top ranking predictor variable in terms of having an inverse or negative correlation with success rate. Maximum number of nights ranks 8th. Based on these, we recommend that property owners not constrain renters to any specific number of nights, insofar as they are able to do so, given their particular circumstances. They should also be sure to respond in a timely manner. The longer the response time, the lesser the chances of success are on renting the property, though we already know that from the positive correlation of response rate. Interestingly, having a profile picture is likely to decrease chances of success. We recommend either not having a profile picture at all, or using some kind of personal avatar. As much as we would like to think that people don't judge a person based on their appearance in their profile picture, this seems to not be the case in real life. If the owner can avoid having their listing fall into the category of "private room in a residential home", then that might improve their rate of success. If it's possible to partition off the rented space with a wall or a locked door, that might be one solution, if it is not already a detached unit on its own.

Rank	Predictor Variable	Correlation
1	minimum_nights	-0.164004
2	long_term_stays_allowed	-0.144584
3	host_response_time	-0.142904
4	has_availability_t	-0.142385
5	host_has_profile_pic_t	-0.109972
6	property_type_Private room in residential home	-0.103716
7	minimum_minimum_nights	-0.101431
8	maximum_nights	-0.093191
9	property_type_Shared room in residential home	-0.089935
10	number_of_reviews_ltm	-0.089086

Table: Predictor variables with highest inverse (negative) correlation to the target variable

VII. Decision Tree / Random Forest

A. Decision Tree Classifier

Decision Tree is a tree-structured Supervised Machine Learning Algorithm that uses a set of rules to make decisions. The rules will continually split the dataset until you isolate the data into each class. After each split, the decision tree algorithm minimizes the loss function as much as possible, like other machine learning algorithms. Decision Trees can be used for classification and regression problems.

Another benefit of Decision Trees is interpretability. Other machine learning algorithms are often black boxes with no explanation, however recommendations can be made from decision trees, providing useful information to Airbnb hosts in this project.

The baseline accuracy is determined by the naive approach of classification which is always choosing the bigger class. In this case, non-successful has 3277 data points while successful has 3063. Therefore the baseline accuracy, to determine if the model is useful, is 51%. The model must have a higher accuracy than the baseline to be more useful.

In this project, we will use a Binary Classification Decision Tree to classify the success of Airbnb listings. The cut off success rate for the decision tree is 50% booked to maintain consistency with other models. With this cut off rate of 50%, the dataset is balanced with 3,277 unsuccessful listings and 3,063 successful listings.

In order to train and test the decision tree models, the data was split into 80% training and 20% validation with 5,072 rows for training and 1,268 rows for testing.

The first model is the basic decision tree classifier model. The model hyperparameter "max_depth" was kept at 5 because the ideal tree is the smallest tree possible, with fewer splits, that can accurately classify all data points. Keeping the max depth small, also allows the decision tree to be interpreted better and prevent overfitting.

The next hyperparameter is "max_features", the number of features to consider when looking for the best split. For this model, "max_features" was set to auto.

If "auto", then max_features=sqrt(n_features).

The data has 68 features due to the one hot encoding of categorical variables in the preprocessing. Therefore, max_features is approximately 8 features for the model.

The 'random_state' hyperparameter makes the model's output replicable. The model will always produce the same results when it has a definite value of random_state and if it has been given the same hyperparameters and the same training data.

The loss function that determines the splitting for the decision tree is Gini Impurity, which is a measure of variance across the different classes, ranging from 0.0 to 0.5. Gini works with binary, categorical target variables such as "success" and "failure". Lower the Gini Impurity, higher is the homogeneity of the node. For example, the Gini Impurity of a pure node meaning the same class, is zero. The formula for Gini impurity is as follows:

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

To evaluate the decision tree model's performance, we will use the confusion matrix. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making. The matrix is as follows:

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Besides accuracy, other metrics are used in the confusion matrix to further evaluate the classification model.

- **Accuracy:** the ratio of correct predictions to total predictions made.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision:** The ratio of correct positive predictions to the total predicted positives. Precision is a measure of quality, the percentage of results that are relevant. Precision is a good measure to determine when the cost of False Positive is high.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

- **Recall:** The ratio of correct positive predictions to the total predicted positives. Recall is a measure of quantity, the percentage of total relevant results correctly classified. Recall is a good measure to determine when the cost of False Negative is high.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

- **F1-score:** the harmonic mean of precision and recall.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

	Precision	Recall	F1-Score	Support
0	0.68	0.45	0.54	626
1	0.60	0.79	0.68	642
Accuracy			0.62	1268
Macro Avg	0.64	0.62	0.61	1268
Weighted Avg	0.64	0.62	0.61	1268

Table: Classifier Model Confusion Matrix on Validation Set

The accuracy of the model is 62%, which is higher than the baseline but still not very good. The recall is 79% and the precision is 60%. With recall being the highest metric, the model is best at predicting positives. The F-1 score is 68%, which is an average of precision and recall. Because our project does not have a bias towards the importance of precision or recall, F-1 score is the best metric.

Decision Classifier Model Tree Graph [3]

B. Random Forest Model

The Random Forest model randomly builds several decision trees from randomly selected features and observations and then averages the results to build a more accurate and complete model. Random Forest model limits the risk of overfitting in decision trees because it creates random subsets of the features and builds smaller trees using the subsets. Random Forest also uses Gini index as the loss function to split the trees.

This Random Forest model will use the same training and validation data as the Decision Tree Classifier model. Keeping the data consistent, will allow the model's performance to be compared accurately.

In the Random Forest model, the first hyperparameter is 'n_estimators' , which is the number of trees the algorithm builds before taking the averages of predictions. This model will build 1000 trees.

The next hyperparameters remain consistent with the decision tree classifier model: 'max_features'='auto', 'max_depth'='5', and 'random_state'='1'.

	Precision	Recall	F1-Score	Support
0	0.67	0.79	0.72	626
1	0.75	0.62	0.68	642
Accuracy			0.70	1268
Macro Avg	0.71	0.70	0.70	1268
Weighted Avg	0.71	0.70	0.70	1268

Table: Random Forest Confusion Matrix on Validation Set

The accuracy increased by approximately 8% by using the random forest model, making the accuracy 70%. Recall decreased to 62% and precision increased to 75%, making precision the highest metric. The F-1 Score remained 68%.

Random Forest Model Tree Graph [4]

C. *Deeper Random Forest Model*

The last model is a deeper Random Forest model. In this model, the 'max_depth' hyperparameter is increased to 10. The max_depth hyperparameter will increase accuracy however it is limited to prevent overfitting and increase interpretability of the tree. Finding the balance of this hyperparameter is ideal.

	Precision	Recall	F1-Score	Support
0	0.83	0.85	0.84	626
1	0.85	0.83	0.84	642
Accuracy			0.84	1268
Macro Avg	0.84	0.84	0.84	1268
Weighted Avg	0.84	0.84	0.84	1268

Table: Deeper Model Confusion Matrix on Validation Set

All metrics increased using a deeper random forest model. The accuracy increased to 84%, precision increased to 85%, recall increased to 83% and the F-1 score increased to 84%. Overall, this model performed the best with all metrics above 83%.

Deeper Random Forest Tree Graph [5]

VIII. Neural Network

The program loads a dataset from a CSV file with the title 'modeldata.csv'. The data in this file is based on the raw data taken from Airbnb. It has already been cleaned and mostly preprocessed by another program, prior to being loaded to this one. Categorical variables are converted by the program natively into binary variables using a dummies function. The data is then passed into a Pandas dataframe. All of the columns of the dataframe must be numerical types in order to be loaded into the model. At this point in the process, the dataframe column types are 'float', 'int', and 'uint'. It has 6340 rows and 75 columns. Of the 75 columns, 74 are for predictor variables and one is for the target variable.

A. *Preprocessing*

The data frame was partitioned with 80% used for training and 20% for validation. The shuffling parameter is set to 'True' so that the results of the model will vary each time it is run. This would hopefully prevent any kind of clustering of similar or related data points from having a disproportionate impact on training and testing the neural network model.

The input data is standardized using the StandardScalar function. This contains all of the predictor variables. The standard deviation for all of the variables, after standardization, is effectively 1.0. The table below shows the first five variables for reference. Standard deviation is highlighted. The same process is used for both regression and binary models. After standardization, all of the predictor variables are type 'float'.

index	host_response_time	host_response_rate	host_acceptance_rate	host_total_listings_count	accommodates
count	4755	4755	4755	4755	4755
mean	2.00E-16	1.08E-15	2.28E-16	7.63E-17	2.07E-16
std	1.000105169	1.000105169	1.000105169	1.000105169	1.000105169
min	-0.6580011481	-6.716197605	-3.843004572	-0.227269587	-1.145748809
25%	-0.6580011481	-0.05806519483	-0.1977120787	-0.2248361622	-0.6063927467
50%	-0.6580011481	0.3669219803	0.3946479515	-0.2224027375	-0.6063927467
75%	0.6046281736	0.3669219803	0.7136110447	-0.1968517779	0.4723193785
max	3.129886817	0.3669219803	0.7136110447	5.693252767	6.94459213

B. *Activation Function*

Both models use the sigmoid activation function. Based on the z-scores of the standardized input data, it predicts a probability of the success rate between the range of 0.0 and 1.0. This is used for both regression and binary models. In the regression model, the prediction is equal to the probability given by the model's output. In the binary model, the model's output is rounded from a proportion to an integer 0 or 1.

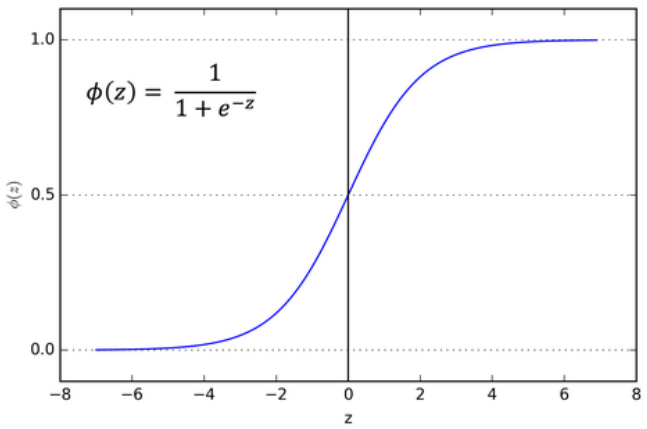


Figure: Activation Function[6]

C. *Regularization*

Both models use regularization on the hidden layer. The regression model uses an L1 regularizer. The binary model uses an L2 regularizer. This was determined by trial and error, with different regularizers and different numerical parameters tried on each model. L1 and L2 regularization both regulate the weights parameters of each node on the layer with the aim of reducing overfitting. The major difference between them is that the L1 tends to "force" the weights towards zero, whereas L2 will tend to push them close to zero but not all the way.

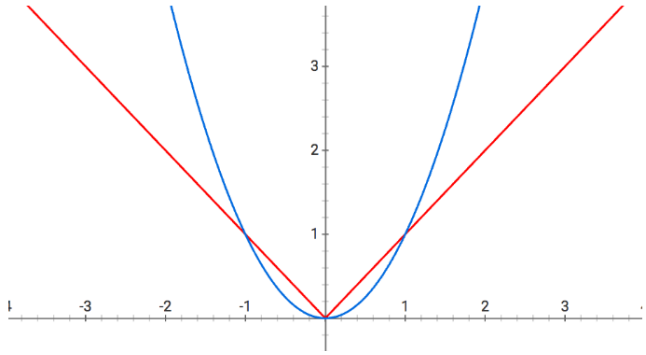


Figure: L1 function (red), L2 function (blue) [7]

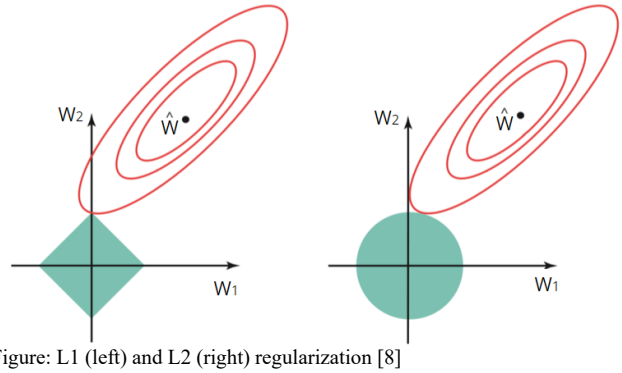


Figure: L1 (left) and L2 (right) regularization [8]

D. *Regression Model*

The regression model uses a dense neural network (DNN) with four layers. The input layer is set to receive a number of variables equal to the number of columns in the dataframe containing the predictor variables. From there, it is passed to a hidden layer with 16 nodes, another hidden layer with 8 nodes, and finally an output layer with a single node. Activation on the hidden layers is set to 'relu'. Activation on the output layer is set to 'sigmoid', which produces a proportion value between 0.0 and 1.0. This is used as the prediction for the success rate.

One hidden layer includes an L1 regularizer with a parameter of 0.01. The optimizer has a learning rate of 0.001. Both the regularizer and the learning rate aim to reduce the degree to which the model overfits itself to the training data. The loss function on the regression model is set to mean absolute error (MAE), customary for regression models. Root mean squared error (RMSE) is used for measuring model performance.

Layer (type)	**Output Shape**	**Param #**
dense (Dense)	(None, 16)	1200
dense_1 (Dense)	(None, 8)	136
dense_2 (Dense)	(None, 1)	9
Total Parameters: 1,345		
Trainable Parameters: 1,345		
Non-Trainable Parameters: 0		

Table: Regression Model Summary

The model was run over 120 epochs. For training data, the loss rate or MAE dropped sharply from 0 to 20 epochs, and then continued to drop gradually and steadily from there, even all the way up to 120th epoch.

For validation data, the MAE also dropped sharply from 0 to 20 epochs. It continued to drop gradually up until about 60 epochs, and then leveled off after that. Additional epochs would not significantly improve the model's performance on validation data.

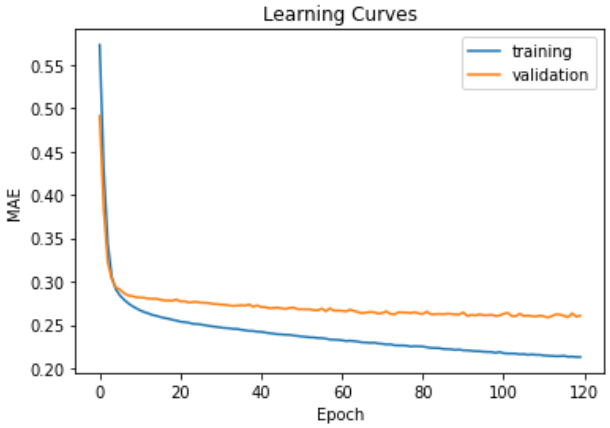


Figure: MAE on Regression Model over 120 Epochs

The model was evaluated on both training and validation data. It performed slightly better on training data, as should be expected. The minimum loss achieved on validation data was about 0.26. This is considered satisfactory for the purposes of this study.

Metric	Value
MAE on training	0.213
RMSE on training	0.277
MAE on validation	0.242
RMSE on validation	0.318

Table: Evaluation of the regression model with loss (MAE) and accuracy (RMSE) measures

To further evaluate performance, the residuals were collected for each data point on the validation data. The residual used here was the absolute value of the difference between the predicted value and the actual value of the success rate. The residuals are distributed in an exponential fashion, sloping downward from left to right, with a very steep slope at the beginning, and gradually tapering off towards the end.

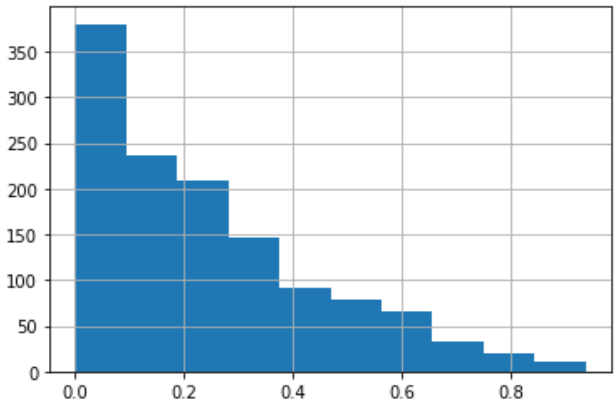


Figure: Histogram of showing absolute value of the residuals on validation data

The regression statistics for the validation data provide more specific information. Based on these, a level of confidence can be provided for each prediction. Based on this analysis, it can be stated with 50% confidence that the actual success rate will be within about 0.19 of the predicted rate. Similarly, it can be stated with 75% confidence that the actual success rate will be within about 0.36 of the predicted rate. This information can be useful for making an informed business decision. For example, someone might be considering the purchase of a particular property to rent out for profit on Airbnb. Another might already own some property, and they want to predict how often it will be booked. Whatever the circumstances, this information could be helpful in making business decisions.

Metric	Value
count	1268
mean	0.2418
std	0.2069
min	0.0000
25%	0.0697
50%	0.1943
75%	0.3600
max	0.9372

Table: Standard statistics for regression residuals

Some additional statistics for the regression residuals show that the model can predict the actual success rate within about 0.65 with 95% confidence. This is a very wide range, which means that a high degree of accuracy cannot be expected from this model. Nevertheless, being able to predict success rate within 0.19 of the actual value with 50% confidence is moderately good and may be useful to some property owners.

Metric	Value
residuals average	0.2417724625
95% C.I.	0.6538938891

Table: Additional statistics for regression residuals

A set of sample residuals has results that are similar to the histogram. Four of the ten data points in the sample have residuals under 0.1, which is considered as “good accuracy”. Two of the residuals are over 0.5, which is considered “poor accuracy”. The rest are moderate.

Prediction	Actual	Residual	Subjective Accuracy
0.085025	0.0417	0.043325	Good
0.76326	1	0.23674	Moderate

0.134905	0.05	0.084905	Good
0.99608	1	0.00392	Good
0.801864	0.05	0.751864	Poor
0.420656	1	0.579344	Poor
0.207834	0.1691	0.038734	Good
0.886238	0.4858	0.400438	Moderate
0.330825	0.6879	0.357075	Moderate
0.884679	1	0.115321	Moderate

Table: Sample residuals of predicted success rates on ten rentals

E. Binary Model

The binary model is used to predict whether or not the success rate will be over 0.5 (or 50%). The parameter can be changed, depending on the wishes of the property owner and the circumstances of each business decision.

$$min\_success\_rate = 0.50$$

The model uses a dense neural network (DNN) with four layers, just like the regression model. The input layer has a number of nodes equal to the number of columns in the dataframe. Each of these corresponds to a predictor variable. There are two hidden layers: one with 16 nodes, and another with 8 nodes. The output layer has a single node, and predicts a proportion between 0.0 and 1.0. The predicted value is rounded by the compiler to 0 or 1 for a final prediction. The loss function being set to "binary cross-entropy" dictates that the output will be binary: either 0 or 1.

The binary model is slightly different from the regression model on a few points. It uses an L2 regularizer with a parameter of 0.001 on the second hidden layer. The learning rate on the optimizer is set to 0.0001. This is done in order to prevent the model from overfitting on the training data.

Layer (type)	**Output Shape**	**Param #**
dense (Dense)	**(None, 16)**	**1200**
dense_1 (Dense)	**(None, 8)**	**136**
dense_2 (Dense)	**(None, 1)**	**9**
Total Parameters: 1,345		
Trainable Parameters: 1,345		
Non-Trainable Parameters: 0		

Table: Binary Model Summary

The loss drops significantly on validation data up to about 60 epochs. After that, it continues to drop, but gradually, and in a more linear fashion, rather than exponential. The loss continues to drop significantly on training data over all epochs.

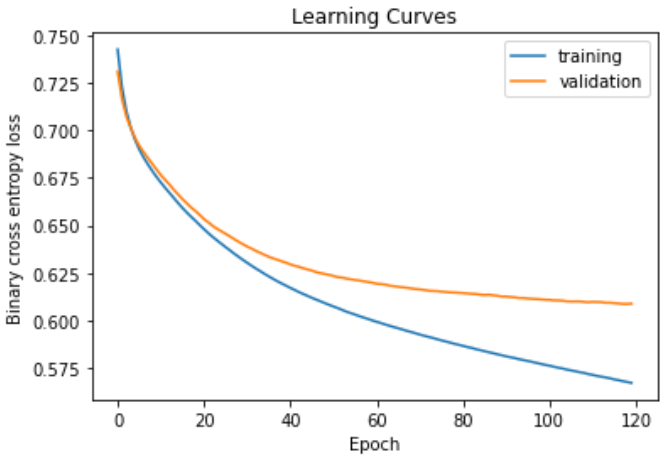


Figure: Loss on binary model over 120 epochs

The loss of 0.61 and accuracy of 0.65 on validation data is considered satisfactory on this model.

Metric	**Value**
training loss	0.5648
training accuracy	0.7003
validation loss	0.6151
validation accuracy	0.6522

Table: Evaluation of the binary model with loss and accuracy measures

A confusion matrix shows a more specific breakdown of the results. Interestingly, the accuracy was 65% on both actual positives and actual negatives. The accuracy is satisfactory, though less than excellent. Nevertheless, at least the model is consistent between predicting positive and negative results for success.

Event	Count	Percent of Category	Percent of Total
true positives	375	65%	30%
false positives	201	35%	16%
total positives	576	100%	45%
true negatives	452	65%	36%
false negatives	240	35%	19%
total negatives	692	100%	55%
all data points	1268	NA	100%

Table: Results of the confusion matrix

A sample of ten data points had slightly better accuracy than the validation dataset as a whole. All categories are represented except false positives. In this way, a property owner could use the model to predict whether or not one or more of their units will be rented out over the minimum success rate. Again, the parameter for minimum success rate could be increased or decreased depending on the wishes of the property owner and the particular circumstances of their business decision.

Prediction	Actual	Classification
0	0	true negative
1	1	true positive
1	1	true positive
0	1	false negative
1	1	true positive
0	1	false negative
0	0	true negative
1	1	true positive
1	1	true positive
0	0	true negative

IX. Picking Best Model

A. Summary of Model Performance

The results of the model are viewed side by side for comparison. Two performance measures are shown here: root-mean-square error (RMSE) and F1 score. The accuracy on classifying success rate using input from validation data ranged between 0.59 and 0.74 for all models. The model with the best accuracy was the ensemble trees, followed closely by the boosted trees.

Model	Regression		Classification	
	Training	Validation	Training (0/1)	Validation (0/1)
	RMSE		F1 SCORE	
KNN	0.2724	0.325/0.316 (valid/test)	0.75/0.73	0.63/0.59
Logistic Regression	NA		0.65/0.62	0.65/0.61
Deeper Random Forest	NA		0.84/0.84	0.84/0.84
DNN	0.305	0.316	0.68	0.65

Table: A summary of the results of each model, showing accuracy measures RMSE and F1 score

B. Prediction for Sample Listings

index	host_response_rate	host_is_superhost	accommodates	beds	minimum_nights	maximum_nights	number_of_reviews	top_amenities_count	host_verification_count	bedrooms
607	27	0	2	1	30	365	18	2	4	1
1812	100	1	6	3	3	75	60	2	8	3
354	94	0	4	2	180	180	1	2	5	2
1620	100	1	2	1	2	7	64	2	8	1

Table: 4 sample listing for Deeper Random Forest Model

Listing	Actual	Predicted
607	0	0
1812	1	1
354	0	0
1620	1	1

Table: Results of Deeper Random Forest on Sample

From this small sample set, we can infer a few recommendations on the features that hosts can focus on to make their listing have a higher probability of success. Below are some examples of the features and the recommendations:

- High Host verification count, i.e. 8
- Host response rate, i.e. closest to 100%
- Low minimum nights, i.e. under 5
- Low maximum nights, i.e. under 80
- Is a super host
- High number of reviews, i.e. 60 or above

X. Challenges

A. Low Correlation Between Target and Predictor Variables

When selecting variables to use in our model, one criteria we considered was the correlation between the predictor variables and the target variable. Dealing with real-world data, rather than data that has already gone through some sort of vetting process, such as might be used for demonstration purposes or during a class lesson, correlation between the predictor variables and the target variable is not guaranteed. There really is no way of knowing whether or not there will be any correlation at all, or if so, to what degree, until after the data is cleaned, and the analysis reaches the exploration and visualization phase. In the dataset used for this study, the highest absolute value of any correlation was about 0.13. This caused some uncertainty as to whether or not any good predictive model could be produced with the data provided. In spite of this, most of the models we tested performed reasonably well. All of them achieved at least 60% accuracy on validation data as a whole, which is well over the accuracy of the naive model, which was about 50%. An accuracy of 85% on the best-performing model was quite pleasing, given the circumstances. It was able to predict at a high-level of accuracy relative to the low level of the correlation between the predictor variables and the target variable.

B. Blocking Issue in Data

The other main issue that we faced in this project was that of property owners blocking their listing on Airbnb. For example, if an owner had their listing available for only one month out of the year, and it was rented for that entire month, then the success rate would be 1.0 or 100%. Another listing might be available for all twelve months of the year, but rented out for only two, making the success rate 0.2 or 20%. Even though the first listing was rented out for only half the time as the second, its success rate is five times more. This could potentially skew the data, causing the listings to have a disproportionately high or low success rate relative to the predictor variables. The original data from the Airbnb website included a category for “blocked” listings, but this was not carried over to the dataset while being scraped from the web. With additional time, we would work to remedy this issue by pulling the data directly from the Airbnb site itself, and then compiling our own dataset, being sure to include the “blocked” category. Further analysis may or may not reveal disproportionality related to this, but at least we would have certainty about it. In the case where it did affect success rate, listings could be parsed into buckets, based on the days per year that they are blocked, with separate analyses being run on each bucket.

XI. Recommendations

A. Location

Location plays a huge role in the booking. Generally, locations in the downtown area have better booking numbers. Of course, the host can’t just move their existing property to another location but if individuals are looking to buy a property to host guests, they can really focus on what area they choose. Downtown location would be a good area to start.

B. Match Price

When looking at the mean price in the neighborhoods around the city, the locations that had the most listings didn’t have the highest prices. The price was really determined by the neighborhood and how much value it has. An example was Russian Hill, which is an expensive neighbor so even if the listing isn’t booked frequently the price can be set high. There are many factor such as size of the property but in a high level view neighborhood value determines the price.

C. Be a Good Host

As a host, guest want to be accommodated in a professional fashion. In data exploration, the fully booked listing had a higher number of hosts who respond within an hour. In the deeper random forest, 4 out of 6 recommendations were host related. As a host being verified, a superhost, and have a high number of reviews (positive reviews) will give guests the confidence that the host is professional, and the place will reflect that.

XX. Conclusion

We learned a lot from this project, and hopefully produced some data that could be useful to property owners who are interested in listing their properties on Airbnb, as well as to staff and management at Airbnb. We measured the success of a listing using the booking status. But “one size fits all” doesn’t apply to rental property as there are so many factors to account for. Some property owners might be content to rent out their property only 20% of the time, while others might expect to have it rented out over 90% of the time. Still others might not be interested in how many nights per year they can rent it out, but rather how much money they can get per night. Whatever the case may be, we are confident that a good model can be created to make predictions, that it can be adapted to the particular circumstances of various stakeholders, that meaningful results can be derived from it, and that these will be useful in making informed decisions.

REFERENCES

[1] [Online]. Available: http://insideairbnb.com/get-the-data/ [Accessed Jan. 23, 2022]

[2] "Welch's t-test" [Online]. Available:https://en.wikipedia.org/wiki/Welch%27s_t-test [Accessed Apr. 20, 2022]

[3] M. Schiller, "Decision Classifier" [Online]. Available:https://drive.google.com/file/d/1tYmD5hE8d0qqo3f1c8n-RX2p9XVmjj8k/view [Accessed Apr. 20, 2022]

[4] M. Schiller, "Random Forest" [Online]. Available:https://drive.google.com/file/d/1gk6qPRXJ0B4tf9zQZbb5_Jmvh6b5ayQg/view [Accessed Apr. 20, 2022]

[5] M. Schiller, "Deeper Random Forest" [Online]. Available:https://drive.google.com/file/d/1HctvwQsE7BSrc3K9GHoH-jRdwFKLd5xy/view?usp=sharing [Accessed Apr. 20, 2022]

[6] S. Sharma, "Activation Functions in Neural Networks" [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 [Accessed Apr. 21, 2022]

[7] A. Oppermann, "Regularization in Deep Learning-L1,L2, and Dropout" [Online]. Available: https://towardsdatascience.com/regularization-in-deep-learning-l1-l2-and-dropout-377e75acc036 [Accessed Apr. 21, 2022]