Predicting Airbnb Ratings: A Machine Learning Approach

Audrey Chu ac8839

Connor Reed cr3221

Dawood Abbas Abdul Malik da2729

Faizan Kanji fnk9850

Abstract

The path to success as a host on Airbnb is often characterized by trial and error, with the experience of any guest influenced by a variety of factors including the behaviours of the host, the qualities of the home, and the location of the listing. Using a machine learning approach, we developed two models to predict the guest rating for a given listing in New York City—one random forest using features of a listing that are more immediately within the control of the host (AUC = 0.716) and one random forest using features derived from a listing's locale (AUC = 0.596). We use the probability predictions of our two models to assign both a "property score" and a "location score" to each listing. Together, these scores establish a useful framework for current and potential hosts to optimize their potential for high guest ratings given the constraints of their property and specific location, and hence improve their revenue earned from the platform. By offering insight to improve the experiences of both guests and hosts, this tool may also be useful for Airbnb in its efforts to reduce churn across both userbases.

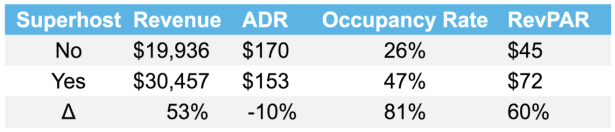
1 Business Understanding

Hosting guests via Airbnb represents a significant economic opportunity for many who have additional room to share in their homes. Since its founding in 2008, Airbnb has served more than 150 million guests, and there are currently about 2 million people who stay in an Airbnb every night. Despite what seems to be an abundance of demand, there is a significant amount of variation and competition among hosts that make becoming successful through the platform potentially difficult. To date, 2.9 million people have become hosts through Airbnb worldwide and in 2020, 14,000 new hosts are expected to join each month (Deane, 2020).[1] Given this expanding reach, it is important for Airbnb to be able to provide meaningful insight to hosts about their potential to be successful on the platform in order to minimize host dissatisfaction with the service and thereby reduce churn.

There are countless factors that influence a listing's success such as amenities in the space and popularity of the location. The difficulty in understanding how to host a successful Airbnb becomes more apparent when hosts lack knowledge about their competition and opportunities to improve stay experiences. While some factors like proximity to public transportation or to tourist attractions cannot be controlled by the host, other factors like host response time and cleaning deposit fees can be. These factors can often make all the difference in a host's potential to receive a positive rating, let alone many.

Receiving consistent positive ratings on one's listings can have a significant impact on a host's Airbnb revenue and profits. Maintaining an overall rating of 96 or higher is a key eligibility criterion for a host to attain "Superhost" status within the Airbnb Platform (Airbnb, 2020). According to an analysis done by AirDNA, attaining superhost status is directly tied to an 81% lift in Occupancy Rate and a 60% lift in RevPAR (primary lodging profitability metric) as shown in Figure 1 (Shatford, 2020).

[1]Projected as of May 2020.



Superhost	Revenue	ADR	Occupancy Rate	RevPAR
No	$19,936	$170	26%	$45
Yes	$30,457	$153	47%	$72
Δ	53%	-10%	81%	60%

Figure 1: Impact of superhost status

Currently a host can use the Smart Pricing tool on Airbnb to determine the optimal price for their listing. Smart Pricing determines the best price for a listing based on factors like lead-time, listing views, neighborhood demand, and booking history (Airbnb, 2017). Over time, Smart Pricing can also be used to update suggested pricing based on factors like review ratings. While Smart Pricing is helpful in suggesting listing prices, outside of their own guests' reviews—which vary in terms of the amount of information they share and their usefulness—hosts still lack much insight on how to improve guest experiences at large and thus the average rating for their listing(s). For example, Smart Pricing might suggest a host increase their listing price by $5 should they add a washer or dryer amenity. However, the host does not know if taking the cost of adding this amenity will boost their ratings or impact their likelihood of being booked. Furthermore, Smart Pricing will also not be directly helpful for hosts who are looking for a tool to help them attain "superhost" status.

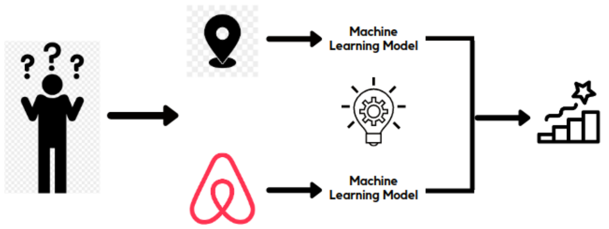


Figure 2: Symbolic overview of data science solution

Using a machine learning approach, we developed two models to predict the average guest rating for a given listing in New York City, one of the most popular Airbnb cities. We explored five types of machine learning models for this task, ultimately creating one random forest using features of a listing that are more immediately within the control of the host, and another random forest using features derived from a listing's locale. Hosts can use this tool to evaluate how much uncontrollable location factors versus controllable property factors affect their ratings, which they can leverage to optimize their hosting strategy and profits. This tool will also be useful for those people who are considering hosting on Airbnb but are unsure of how successful their listing will be. By helping hosts succeed on the platform, this tool will also be helpful for Airbnb in its efforts to reduce churn across its host and guest userbases.

2 Data Understanding

2.1 Data Collection and Sources

The data science problem we are trying to solve is to predict the average rating of an Airbnb listing given a set of controllable and uncontrollable features. Controllable features are those aspects of a property which are, to a certain extent, in the hands of the host, like price per night, number of bedrooms, and types of amenities available. Whereas the uncontrollable features are those that describe the location of the listing and the qualities of places that are accessible nearby. By focusing on these sets of features, we are able to dissect the performance of an Airbnb listing along location-based uncontrollable features and property-based controllable features.

2.1.1 Controllable Data

The primary data source we leveraged is a comprehensive dataset of Airbnb Listings. This dataset is collected from the website "Inside Airbnb", which provides monthly snapshots of scraped Airbnb listings across several regions (Inside Airbnb, 2020). We focused on Airbnb listings in New York City over 2019 as the scope of our analysis.

The raw listings dataset for New York City from 2019 contains 106 columns and 530k rows – which comprise of 80k unique listings. The dataset includes a combination of continuous variables (such as price of a listing), binary variables (such as whether the listing is "instant bookable" or not), simple categorical variables (such as the host's cancellation policy) as well as raw text data (such as the listing's specified amenities).

Our exploratory data analysis revealed that several of the categorical variables have a highly skewed distribution. As an example, there are 30 distinct property types but 77% of the listings have the type "Apartment" and the top 5 types account for 97% of the listings as shown in Figure 3. This would indicate both a need for grouping several less frequent property types in an "other" bucket as well as a need for standardizing the features if we plan to use model types where the scale and skew of features can influence model selection.

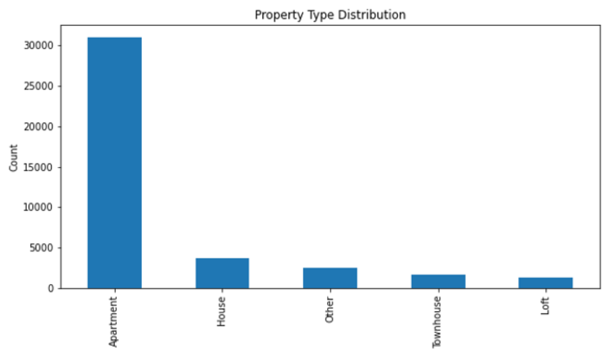


Figure 3: Property type has a skewed distribution

2.1.2 Uncontrollable Data

The location of each listing in our listings dataset is encoded in longitude/latitude coordinates. We decided to leverage this information to compute features for each listing derived from other georeferenced data.

Points of interest: We used Foursquare's Places API to gather information about venues within New York City that could be relevant to any particular listing in our listings data set (Foursquare Labs, Inc., 2020). Each request sent to the API returns a JSON object of up to 50 venues (i.e., points of interest), including their names and coordinates in longitude/latitude, selected according to a set of parameters. Given that each listing in our listings data set corresponds to a particular neighborhood, we decided to query the API for the top 50 venues in each neighborhood represented in the listings. These venues were ordered in terms of decreasing popularity (ranked by Foursquare) for four different categories of activity: (1) general (i.e., non-specific); (2) food; (3) nightlife; and (4) travel (e.g., hotels, airports, trains, ferries, etc.). Some of the neighborhoods represented by listings did not match available neighborhoods in the Foursquare Places database, so we also collected the top 50 venues in each of the above categories for the boroughs to which those "missing" neighborhoods belong as well as for all of New York City. In total, we collected a total of 10,500 points of interest that could be relevant to a guest's experience at any listing in the city.

Local demographics: Publicly available zipcode-level data from the 2010 census provides a rich dataset of micro-geographic demographic indicators (US Census Bureau, 2010). From this dataset, we extracted relevant features that might help explain the performance of an Airbnb location such as median age, income, population density and education levels in New York City zip codes.

Subway proximity: In order to more accurately understand the location of an Airbnb, we opted to include subway transportation in our model. We used the Subway Stations data set from NYC Open Data to calculate the number of subway stations and the number of subway lines within a 0.2 miles, or approximately a 4 block, radius of a given listing by using longitude/latitude coordinates. This subway data set contains 473 subway stations and 105 unique subway lines as of August 2019 (Metropolitan Transportation Authority (MTA), 2020).

Competition: Intuitively if there are a lot of choices of Airbnb's in a particular area then attracting customers and having good reviews is going to be tougher. Hence, one of the most important features is the number of choices a customer has when they look for a place in a particular area. If there are more options then the host has to compete with a bigger crowd in getting good reviews. We used the Airbnb Data mentioned earlier to obtain a feature describing the competing Airbnbs. We have also created another feature which describes the competition existing outside of Airbnb. For this we have used the data set obtained from NYC Open Data which contains the coordinates of 2,731 hotel properties in NYC (Department of Finance (DOF), 2020).

3 Data Preparation

The raw Airbnb Listings data is in a semi-panel format, with each listing appearing multiple times in the dataset depending on how often it was scraped. All of our features as well as the average review score of a listing, however, do not vary substantially with time over the course of 1 year. We therefore decided to eliminate the time dimension in preparing the dataset for our data mining exercise by deduping the listings and selecting the latest scraped version of each listing in 2019. This simplifies our model building and testing process by allowing us to have one instance per listing in our final dataset.

Additionally, to avoid having listings with relatively less accurate and noisier mean review scores, we filtered to only those listings that had at least 5 reviews. While this does introduce some selection bias by filtering out newer or less frequented listings, we can mitigate it by thinking of the 5 review criteria as a maturity threshold and interpret our model scores to reflect mean rating for a mature listing only.

Finally, we then reviewed the distribution of missing values in our de-duped and filtered dataset. While a few features with substantial missing values (such as host response rate) were imputed using their mean, we identified a small set of instances (about 330 rows) that had no data on any host related features. We decided to drop these rows given they represented a very small proportion of the dataset. This gave us 39,801 listing instances that we used for our final modeling dataset.

3.1 Target Variable

The target variable for our model is a binary ratings variable where ratings above the median value equal 1 and ratings below or equal to the median value equal 0, where the median score is 95. As shown in Figure 4, the distribution of raw review score ratings is very left-skewed, suggesting that most guests have a neutral or positive experience. We opted for a 95 score threshold for two reasons:

1. To ensure the classes in our dataset were equally balanced.
2. To reflect an important eligibility criterion that, in part, distinguishes superhosts from regular hosts.

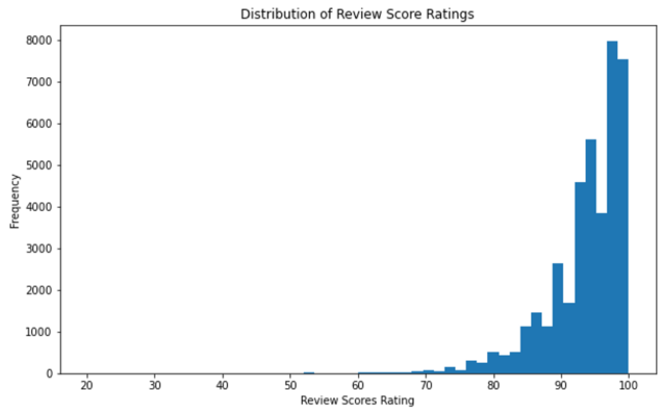


Figure 4: Raw target variable distribution

3.2 Feature Engineering

Listing Data: While the listing data was fairly rich and robust in its raw form, we engineered a few additional features based on the inputs extracted. Firstly, any skewed simple categorical variables were first binned through the creation of an "Other" bucket and then converted into binary features using one-hot encoding. We also noticed that even though the Amenities information was primarily in free-form text format, the feature includes several common words (such as "air conditioning" or "coffee maker"). We believe this contains information that could be predictive of our target variable so we extracted this by identifying the 15 most common amenities by parsing through the text, and created binary features for whether a listing has each of those amenities or not.

We also hypothesize that the date a host first joined Airbnb can provide information about their ratings. We leveraged that date to engineer two additional and more useful features - a feature indicating the tenure of a host and a binary feature indicating if the host is completely new in 2019 or not.

Finally, we also identified that nulls in some features give us useful information about those features. For example, a null in the cleaning fee or security deposit column indicates the host does not charge a cleaning fee or security deposit. We created additional binary features to indicate whether the original feature was null or not to capture this information.

Points of interest: We wanted to construct features that would provide information about a listing's location relative to highly popular points of interest within its neighborhood and across the city. To do this, we created the following metric inspired by inverse-distance similarity:

$$H_c(l,n) = \sum_{v \in n_c} \frac{1}{r(v) * d(v,l)} \tag{1}$$

Where c is the venue category and n is the neighborhood such that n_c is the set of venues in neighborhood n of category c. v is a venue, $r \in \{1, \ldots, 50\}$ is the popularity rank, d is the geodesic distance function in kilometers (calculated using geopy.distance.distance), and l is the Airbnb listing. The idea behind this feature is that it is higher for listings that are closer to the most popular venues in its neighborhood, without discarding information about a listing's proximity to other venues that are less popular.

For each listing in the dataset and for each category of venue, we calculated this metric using the top 50 venues both within that listing's neighborhood and across the entire city. This resulted in each listing having a "local" score and a "citywide" score reflecting its proximity to popular general points of interest, food, nightlife, and travel.

Finally, we calculated the number of the top 50 general venues within a listing's neighborhood that were within 500 meters of the listing. This was computationally intensive, and so we were only able to compute this count for the general category of venues.

Competition: In order to formalize the aspect of competition as a parameter, we have taken the number of active Airbnb listings in a 2-mile radius of the new property. On a similar note, we have also formalized a feature with the number of hotels (which are not on Airbnb) within a 2-mile radius. Since these features are obtained either from a self-join with the Airbnb data set or a join with NYC Hotel data set, calculating distance for all the combinations using the geographic coordinates and then computing these features is time-intensive. In order to overcome this, we have found the geographic coordinates of points within 2 miles and in the 45 degrees NE, 135 degrees SE, 225 degrees SW and 315 degrees NW directions, and computed the number of locations within the square with these coordinates.

4 Modeling & Evaluation

4.1 Suitable algorithms

We considered five algorithms for the task of classifying the ratings of our available listings: decision tree (DT), logistic regression (LR), support vector machine (SVM), random forest (RF), and gradient boosting machine (GBM). Each of these models has their own advantages and disadvantages in classification. For instance, DTs, RFs, GBMs may have an advantage in some tasks in that they operate without any parametric assumptions of the data. As a result, they are better equipped to handle unspecified non-linearity and/or interactions among our features if they exist. Additionally, these algorithms are useful for ranking the predictive importances of different features by their normalized information gain, which is meaningful in our use case as we hope to be able to make recommendations to current and future hosts about how to improve their ratings in addition to predicting how well they will be rated.

LR and SVM have similar advantages and disadvantages. Both are more effective for smaller datasets than ours and reach performance asymptotes with greater amounts of data. While we hypothesized that the tree variants would perform better for our classification task, we considered LR and SVM to be thorough in our search.

One of the advantages of RF is that the default hyperparameters it uses often produce good prediction results. It is also easy to view the relative importance it assigns to input features, which will be helpful in our use case. RFs also handle overfitting well, meaning that if there are more trees in the forest, the classifier won't overfit the model. A disadvantage of RF is that large number of trees will make the algorithm slow and increase the computation time in creating predictions.

4.2 Evaluation Framework

We relied on an empirical framework to evaluate the relative efficacy of these different learning algorithms for classifying Airbnb ratings with the following steps:

1. Split data into train, validation, and test sets along a 60/20/20 ratio.
2. Split the train, validation, and test sets by feature type (controllable or uncontrollable), yielding two train sets, two validation sets, and two test sets with identical instances but non-overlapping features. Scale all feature sets to mean 0 and variance 1.
3. For each feature type:
   i. Build a baseline model for each algorithm using the appropriate training set and the default parameters in `scikit-learn` and evaluate its performance in terms of AUC on the associated validation set.
   ii. Search set of available hyperparameters for each model using a mixture of manual searching, `RandomizedSearchCV`, and `GridSearchCV`. Select the hyperparameters that yield the highest AUC on the validation set, identifying the best version for each model type.
   iii. Compare the AUC of each of the best model types on the validation set to identify the overall best performing model.
   iv. Evaluate the top model's performance on the Test set across AUC and value dimensions to

get an overall evaluation of our data mining exercise.

We decided to use AUC as our evaluation metric because it provides a comprehensive summary of how well each classifier ranks over all decision thresholds, allowing for nice comparison between the models. Additionally, AUC is a useful evaluation metric if the ultimate goal is ranking instead of just classification - this will be useful in giving us the flexibility to generate accurate scores that can help rank different listings across location and property dimensions. Finally, we care about classifying both good and bad locations accurately which further makes AUC an appropriate choice because it prioritizes true positives and true negatives equally.

4.3 Baseline model

Given the relative advantages of tree-based models identified in the previous section, as well as its ability to do implicit feature selection, for our baseline model we leveraged an out-of-the-box Decision Tree. This algorithm produced a validation AUC of **0.519** for the uncontrollable model and an AUC of **0.565** for the controllable model.

The out of box decision tree has very lenient hyperparameters that allow the tree to go as deep as possible. This makes the model very prone to over-fitting on the training data. Therefore, we hypothesize we can significantly improve upon these baseline models. In doing so, we will test out the other models identified earlier as well as tune the hyperparameters of each model to attempt to achieve an optimal bias-variance trade off. We describe this approach in more detail in the next section.

4.4 Model training and tuning

To improve upon our DT baseline model we decided to tune its hyparameters, as well as model and tune hyperparameters for algorithms we seem fit i.e., RF, LR, SVM, and GBM. Since we followed the same approach for tuning parameters across these models, we have explained the process with a deep dive of RF tuning process for the uncontrollable model.

4.4.1 Tuning Random Forest

While our baseline RF works reasonably well with the default values of the hyperparameters specified in the `sklearn.ensemble` package, tuning the hyperparameters can improve the performance of our RF.

Adjusting parameters in random forest are either to increase the predictive power of the model or to make it easier to train the model. RF hyperparameters include forest level and tree level parameters. Forest level parameters include number of trees and number of features to sample. Increasing the number of trees decreases variance, but also increases training times. Reducing the number of features samples in each tree increases the bias, but decreases the RF variance. Tree level parameters include number of intermediate nodes, or max depth, and the size of the intermediate nodes. Usually, we don't want to limit max depth, and so for our model, did not set this parameter.

To find the optimal number of trees and features to sample, we used `GridSearchCV` and `RandomizedSearchCV`. We first began with `GridSearchCV` by setting a wide range of parameters. However, this process took much longer than expected since RF has at least four hyparameters and runtime increases substantially with the increase in number of trees. We then opted for a step-wise method in which we searched for one parameter at a time and at each next step, searched for a new hyperparameters conditional on the previously defined best estimator. Through this method, we discovered that AUC performance plateaus after 50 trees and the best AUC is achieved at 101 number of trees. Using 101 trees, we then searched for the highest AUC across various min_sample_split values, finding that the optimal number of splits occur at 1600 (5). We continued this process with the remaining hyperparameters to produce a model with a AUC of 0.584.

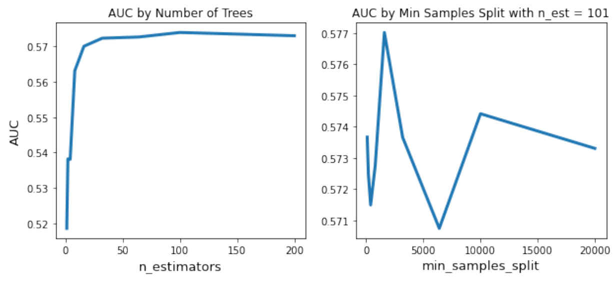


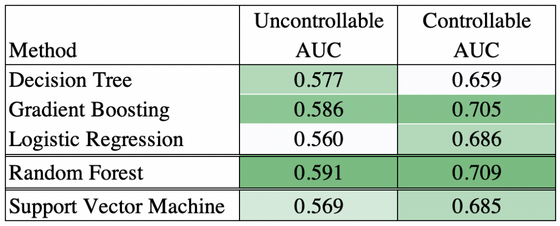
Figure 5: Step-wise tuning of RF

After gaining a better understanding the hyperparameter ranges in which the model worked better, we decided to use `RandomizedSearchCV` to search more parameters in ranges that had higher AUCS from the step-wise method. From our earlier exploration, we expected to find a model that has around 100 trees. Using `RandomizedSearchCV`, we reduced the search time significantly and were able to find a model that had an AUC of 0.591, which is an improvement from both the baseline model (0.575) and the exploratory step-wise model (0.584). The best parameters are are n_estimators = 101, max_features = 161, min_samples_split = 300, and min_samples_leaf = 1.

This tuning methodology was applied to both uncontrollable and controllable models across all methods based on each method's respective hyperparameters.

5 Results

Our final AUCs for our best models after tuning are listed in Table 6. After tuning each method and model, we find that a tuned RF performs the best. For both the uncontrollable and controllable models, our final validation AUCs of **0.591** and **0.709** respectively are a significant improvement over our baseline model (with AUC's 0.519 and 0.565 respectively).



Method	Uncontrollable AUC	Controllable AUC
Decision Tree	0.577	0.659
Gradient Boosting	0.586	0.705
Logistic Regression	0.560	0.686
Random Forest	0.591	0.709
Support Vector Machine	0.569	0.685

Figure 6: Best model performance

5.1 Combined Evaluation and Value Framework

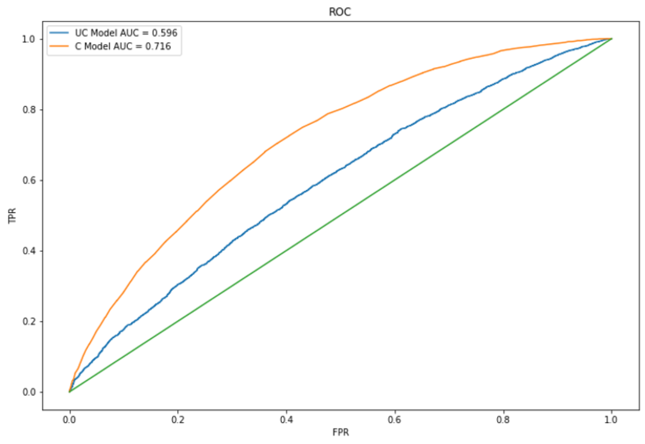
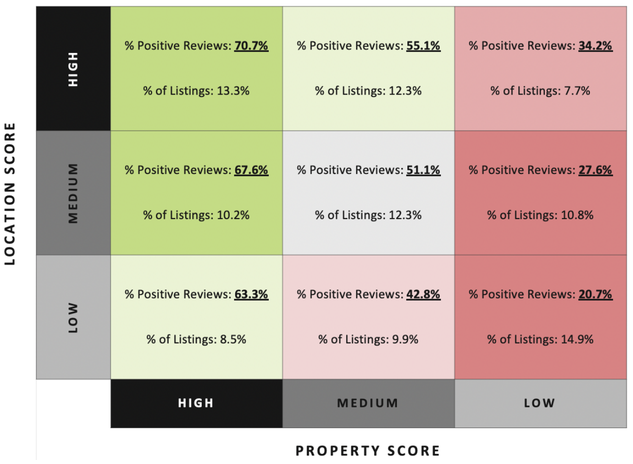


Figure 7: ROC and AUC evaluated on test set

With our final Random Forest models determined through the validation exercise described above, we now test these models on the previously unused Test data. To do so, we retrained the model on the Training+Validation data and evaluate it on the Test data. Figure 7 shows the ROC's and the Test AUC's of our final Controllable and Uncontrollable models. We note a few key observations:

- Our Test AUC's of 0.596 for the Uncontrollable model and 0.716 for the Controllable model are fairly comparable to our validation AUCs suggesting our model selection process did not overfit on the validation data.

- The Controllable model is significantly more accurate than the Uncontrollable model suggesting that property level controllable characteristics of a listing have a stronger influence in determining high vs. low ratings than location-based characteristics of the listing. This is an encouraging insight for hosts who can potentially significantly improve their rating despite being in a less desirable location through changing controllable property features.



LOCATION SCORE	PROPERTY SCORE: HIGH	PROPERTY SCORE: MEDIUM	PROPERTY SCORE: LOW
HIGH	% Positive Reviews: **70.7%** % of Listings: 13.3%	% Positive Reviews: **55.1%** % of Listings: 12.3%	% Positive Reviews: **34.2%** % of Listings: 7.7%
MEDIUM	% Positive Reviews: **67.6%** % of Listings: 10.2%	% Positive Reviews: **51.1%** % of Listings: 12.3%	% Positive Reviews: **27.6%** % of Listings: 10.8%
LOW	% Positive Reviews: **63.3%** % of Listings: 8.5%	% Positive Reviews: **42.8%** % of Listings: 9.9%	% Positive Reviews: **20.7%** % of Listings: 14.9%

Figure 8: 9-Box evaluation and value framework

To further evaluate potential practical applications of this model, we propose a 9-box value framework. The 9-box framework is a 2-dimensional bucketed variant inspired by the Lift evaluation metric and it helps more closely tie our model predictions to both the potential economic value generated from the model as well as a decision making framework to help hosts optimize their review score. We construct this 9-box as follows:

1. Generate probability predictions using our Random Forest models on the Test data. We will consider the Uncontrollable probability predictions to be the "Location Score" for a listing and the controllable probability predictions as the "Property Score".
2. For each score, we create two thresholds using the 33.33% and 66.66% percentiles on the Test predictions allowing us to create a "Low", "Medium" and "High bucket for each score.
3. The cross tabulation of these buckets give us 9 boxes. A new listing will therefore be scored, bucketed and receive a 9-box score comprising of its Property score bucket and Location score bucket.

To evaluate our model against this framework, we score and classify each of our Test listing into the 9-box grid. The results are in Figure 8. A few observations based on this framework:

- Generally, we see the trends we expect across the 9-boxes (i.e., across both property and location dimensions, higher scores correspond to higher actual probability of having a high rating).
- It is exciting to see that our (high, high) box has a score of 70.7% while our (low, low) box has a score of 20.7%. Considering that in the overall dataset, the distribution of positive and negative listings is 50-50, our model is doing very well in accurately identifying both really good and really bad listings.
- Consistent with our observations from the ROC curve, improving their property score gives hosts a much greater bang for their buck than moving to a better location. Even across the buckets with the lowest location scores, having a high property score gets you to an average 63.3% probability of having a high rating - which is better than the overall average probability of 50%.

This overall evaluation suggests that our models combined with the 9-box evaluation framework can be really helpful for hosts in understanding the potential performance of their listing and identifying ways to improve the listing. We will build on this further by diving deeper into feature importances to identify which factors will have the most influence in improving a listing's property and location scores, allowing us to give more specific recommendations to hosts on how to improve their ratings.

5.2 Feature Importances

In addition to the model scores themselves, an analysis of feature importances will help hosts understand not only how their current or potential listing would score, but also how they can improve their score. By focusing and prioritizing the most important controllable and uncontrollable features, a host can identify the most efficient way (i.e., requiring the fewest modifications) to improve their 9-box bucket.

From our best uncontrollable model, we calculated feature importances for our RF classifier. In Figure 13, we find that that the Airbnb competition variable is the most important feature in predicting a high Airbnb rating, followed by several key points of interest scores.

From our best controllable model, we find that price, host tenure, cleaning fees, and maximum nights offered are the most important features in predicting a good rating. For hosts who have listings in less popular areas based on Airbnb competition or population, they can focus on these controllable features to optimize their hosting strategy to obtain more or better ratings.

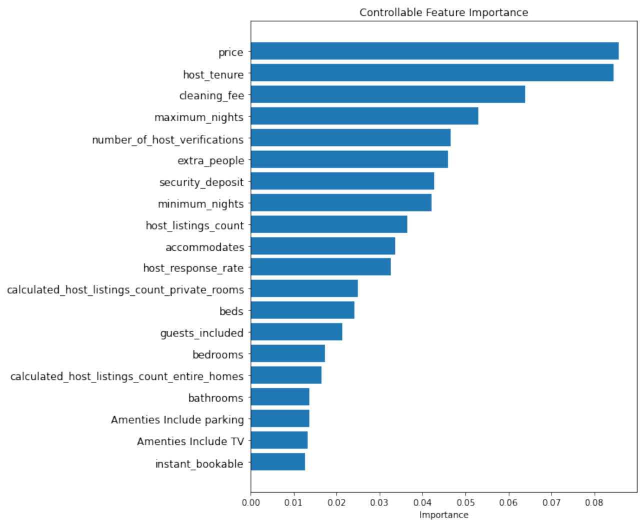


Figure 9: RF controllable feature importances

6 Deployment

6.1 Deployment infrastructure and evaluation

Our model provides a very useful evaluative framework for Airbnb listings along location and property dimensions. There are a variety of ways this model can be deployed and leveraged in a production environment. Primarily, our model can be deployed as part of a standalone platform that can serve as a listing evaluation calculator for potential and current hosts. In addition to the scoring logic, we will also have to deploy feature creation scripts that, given a latitude and longitude, can create all location-based features (e.g., points of interest features from foursquare API or zip level census demographics). Once we have that infrastructure built out, potential hosts that are considering whether their property would be a good Airbnb location or not can input their location (i.e., address or lat/long), and characteristics of the property (e.g., number of beds, price being considered, etc.), and this calculator can provide them with location and property scores – specifically highlighting where within the 9-box their considered listing falls. These scores can be an input into their decision process regarding whether to list their property on Airbnb or not. Existing hosts can also leverage this calculator to dissect their property's performance along uncontrollable location characteristics and controllable property characteristics. Building on this evaluation, they can leverage this as a tool to understand the impact of changing certain controllable characteristics (e.g., charging a different price or offering additional amenities) to their listing's rating. Additionally, we can also simulate different changes and re-score the model to propose recommendations based on the minimum changes a user needs to do to improve their score. In this way, current hosts can identify what changes can help them move their list-

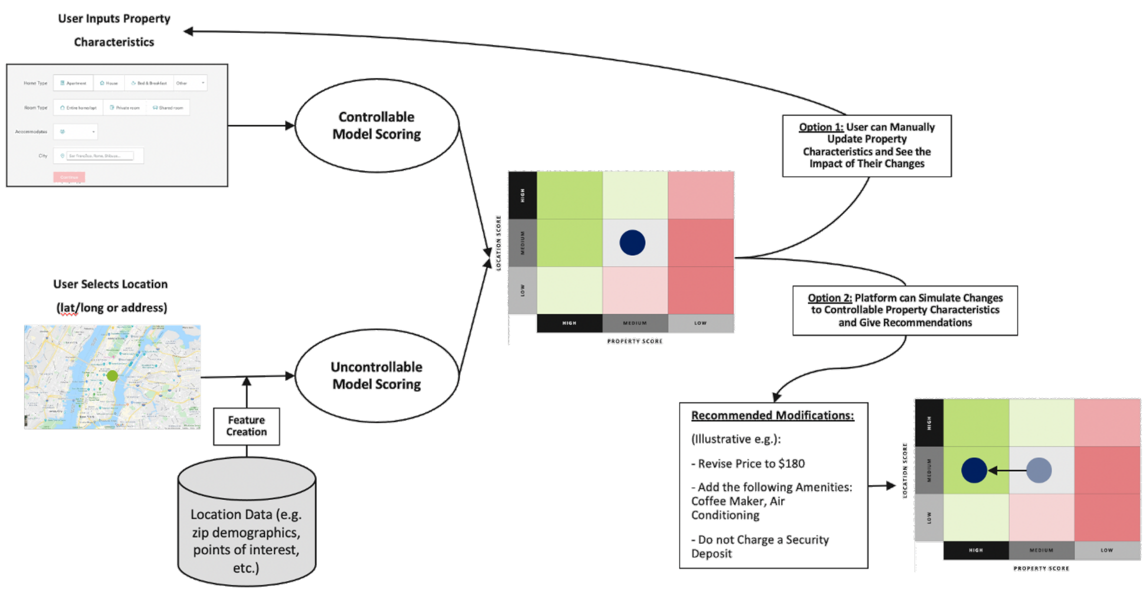

User Inputs Property Characteristics
Controllable Model Scoring
User Selects Location (lat/long or address)
Feature Creation
Location Data (e.g. zip demographics, points of interest, etc.)
Uncontrollable Model Scoring
LOCATION SCORE
HIGH
MEDIUM
LOW
PROPERTY SCORE
Option 1: User can Manually Update Property Characteristics and See the Impact of Their Changes
Option 2: Platform can Simulate Changes to Controllable Property Characteristics and Give Recommendations
Recommended Modifications:
(Illustrative e.g.):
- Revise Price to $180
- Add the following Amenities: Coffee Maker, Air Conditioning
- Do not Charge a Security Deposit

Figure 10: Proposed Deployment Flow

ing to a more preferable 9-box bucket or attain superhost status, resulting in improved ratings and revenue. The flow of this proposed deployment approach is summarized in Figure 10.

As an alternative to deployment as a standalone platform, this can also be incorporated within tools Airbnb provides hosts. By providing a tool to help hosts evaluate and improve their listings, Airbnb can benefit by lower host churn as well as higher quality listings leading to additional revenue as they capture additional travel and lodging spend share from hotels.

Once deployed, we have to consider how frequently we will refresh our model as well. We hypothesize that the factors that determine whether a location or property will be rated high or not are unlikely to change very frequently. As a result, we likely don't have to worry about very short-term concept drift and therefore do not need to update our model very frequently – we believe a quarterly or bi-annual cadence of refreshing the model would be appropriate.

Finally, we want to ensure we are consistently evaluating model performance in the production environment. We propose evaluating the model using new properties that have crossed the 5-review mark to ensure we are not evaluating on the same set of properties we trained the model on. This will also help us catch concept drift because if our model is consistently misclassifying newer properties, it could indicate a change in the market dynamics or consumer preferences that would highlight the need for a model refresh.

6.2 Potential Issues, Risks and Considerations

A risk and potential issue that anyone deploying this model should be aware of is that, as stated in the business understanding section, ratings are only one aspect of what makes an Airbnb location successful and may not fully capture metrics such as profitability. As a hypothetical example, if I offer an Airbnb in a luxurious apartment in Times Square for $20 a night, our model will give it very high property and location scores. However, that location will not be a profitable Airbnb location given the significantly low price. It will be important to ensure that as this model is deployed, additional models are also created and deployed that can help hosts evaluate other dimensions of success of their listings (e.g., RevPAR, occupancy rate, etc.).

We should also be aware of the potential of a negative feedback loop on the system. If our model helps existing hosts improve their ratings or discourages those potential hosts from joining Airbnb whose listings would have gotten bad ratings to the point where there are very few low rated listings left, we will no longer be able to update and effectively leverage this model. To mitigate this we should consistently evaluate the distribution of high and low ratings we see in the data and update our modeling process if we see that distribution significantly changing.

Finally, we should keep in mind an important ethical consideration when deploying this model. Within the location evaluation, our model learns what someone considers a good or bad location within New York City. This can lead to the model learning and perpetuating implicit biases people have about locations based on factors such as racial or ethnic demographics in those locations. The use of this tool can then discourage hosts in such locations from listing their properties on Airbnb. Evaluation of how these implicit biases can be learned by the model should be incorporated into the model-building and evaluation process.

References

[Airbnb2017] Airbnb. 2017. What's smart about smart pricing? https://blog.atairbnb.com/smart-pricing/.

[Airbnb2020] Airbnb. 2020. How do i become a superhost? https://www.airbnb.com/help/article/829/how-do-i-become-a-superhost.

[Deane2020] Steve Deane. 2020. 2020 airbnb statistics: Usage, demographics, and revenue growth. https://www.stratosjets.com/blog/airbnb-statistics/.

[Department of Finance (DOF)2020] Department of Finance (DOF). 2020. Hotels properties citywide. data retrieved from NYC OpenData, https://data.cityofnewyork.us/City-Government/Hotels-Properties-Citywide/tjus-cn27.

[Foursquare Labs, Inc.2020] Foursquare Labs, Inc. 2020. Places database. data retrieved from Places API, https://api.foursquare.com/v2/venues/search.

[Inside Airbnb2020] Inside Airbnb. 2020. data retrieved from Inside Airbnb, http://insideairbnb.com/get-the-data.html.

[Metropolitan Transportation Authority (MTA)2020] Metropolitan Transportation Authority (MTA). 2020. Subway stations. data retrieved from NYC Open Data, https://data.cityofnewyork.us/Transportation/Subway-Stations/arq3-7z49.

[Shatford2020] Scott Shatford. 2020. What is airbnb's superhost status really worth? https://www.airdna.co/blog/airbnb_superhost_status.

[US Census Bureau2010] US Census Bureau. 2010. Decennial census tables. data retrieved from US Census Bureau, https://www.census.gov/programs-surveys/decennial-census/data/tables.2010.html.

Appendices

A Additional Figures

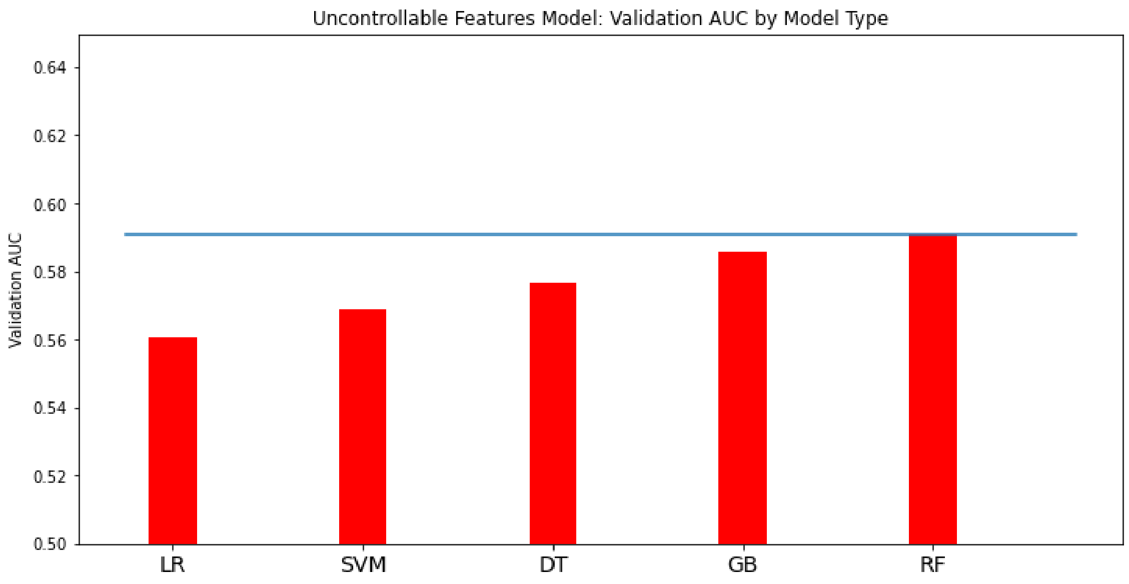


Figure 11: Uncontrollable Validation AUC Across Model Types

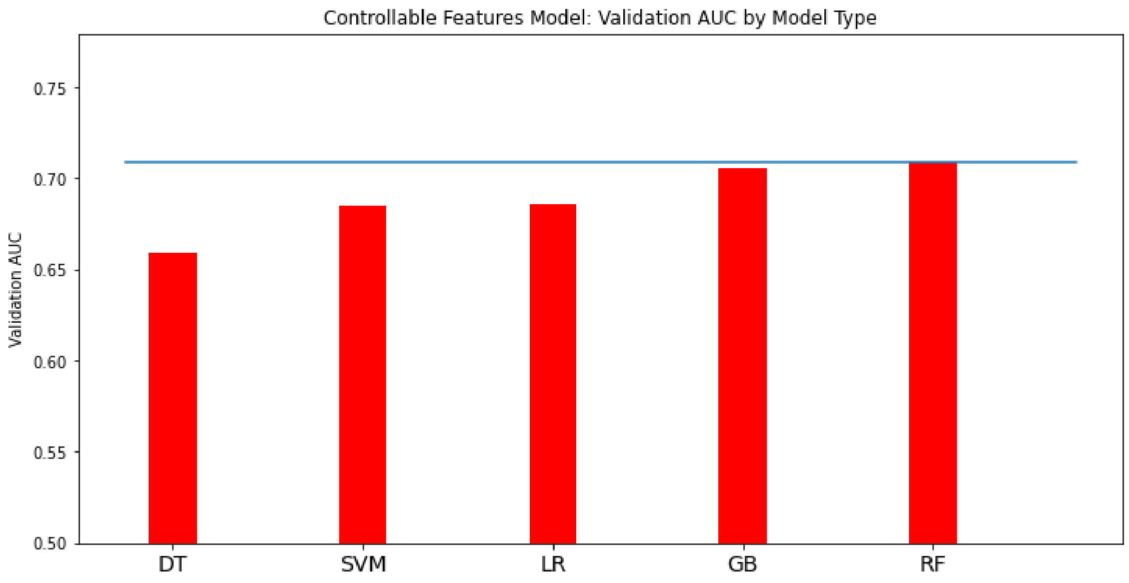


Figure 12: Controllable Validation AUC Across Model Types

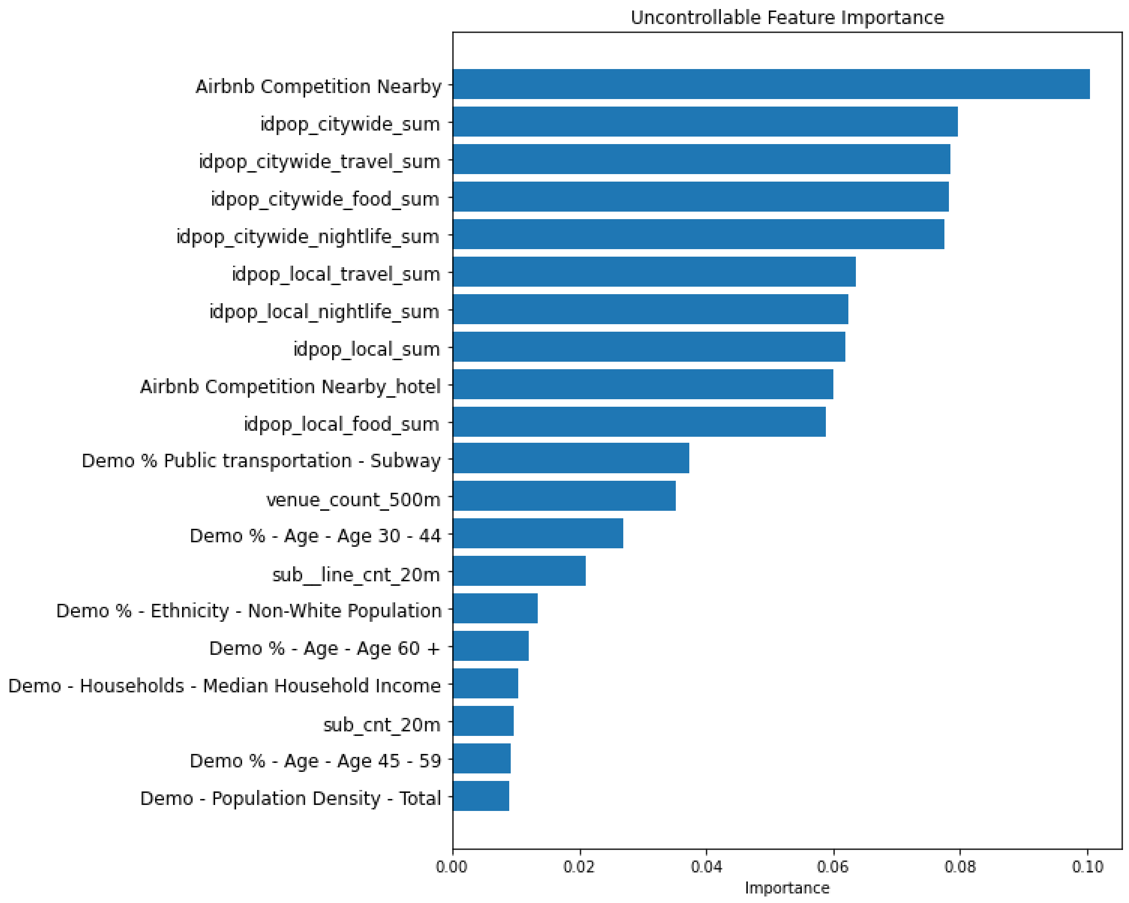


Figure 13: RF Uncontrollable Feature Importances

B Individual contributions

- Audrey Chu: Random Forest tuning, subway data cleaning and feature engineering
- Connor Reed: Logistic regression and support vector machine tuning, Foursquare data cleaning and feature engineering
- Dawood Abbas: Gradient boosting tuning, competition data cleaning and feature engineering
- Faizan Kanji: Decision tree tuning, Airbnb and Census data cleaning and feature engineering

C GitHub Repository

The code for this project can be found at `https://github.com/audreychu/Airbnb_Optimization`.