# *Predicting NYC Airbnb Listing Price*

*Muen Chen*
*Fordham University*
*Bronx, NY*
*mchen118@fordham.edu*

***Abstract* — Airbnb is the predominant housing-sharing (lodging) business in the United States. The company publicly offers a dataset on all of its listings in major cities like the New York City. Data mining methods (regression methods, specifically) can be used on this data to explore what features of a listing matter most to market and is a good exercise to push for the limitation of these methods on a complex dataset with high dimensionality.**

**Using manual feature selection, one-hot category encoder and random forest regression algorithm, I was able to build a regression model on New York City's Airbnb listing dataset that could predict the listing price with an absolute 23% margin of error on average. Ordinary least squares linear regression was the baseline algorithms, which random forest was able to out-perform along with k-nearest-neighbor regression and multi-layer perceptron regression. Multi-layer perceptron was surprisingly bad on this dataset.**

**The complexity of New York City's neighborhood forced me to entirely drop the neighborhood feature for the model. In future works, some feature importance analysis should be used so neighborhood info can be incorporated into dataset using one-hot encoding without blowing up the dimensionality. In addition, further fine tuning of models using a validation set has to be done to maximize prediction accuracy. Lastly, a comparison between the city-wide model and neighborhood-specific model is worth exploring to see if one has better accuracy than the other.**

## I. Introduction

### A. Background

New York City's housing-sharing (lodging) business is booming since the first Airbnb listing appeared a decade ago. Luckily Airbnb, the leading company in this industry, offers an expansive dataset of all of its listings in major cities across the globe, including NYC. It allows us to explore whether we can accurately predict listing price using data mining and statistical models, and to perhaps learn what factors affect the listing price along the way.

### B. Problem Statement

To predict Airbnb listing price in New York City.

## II. Experiment Methodology

### A. Data

The dataset comes from insideairbnb.com, which scrapes publicly available date from Airbnb's site. For each major city, data is craped monthly, and the monthly data is divided into three types – listings data, calendar data, and reviews data. For this paper's purpose, I use only the listings data, specifically those scraped in August, 2019 and September 2019. Combining the August and September data, there are 97272 records in total, and each record has 106 features.

### B. Data Pre-Processing

#### I. REMOVING USELESS FEATURES

Most features in the dataset are either 1) irrelevant to pricing, like "listing_url", "scrape_id", 2) beyond the scope of my abilities, like "description", which is a host-written string that presents the host's listing, meaning it requires text mining to be useful, or 3) redundant, like "city". Those features are removed right away, leaving 47 of the original 106 features.

#### II. ENCODING CATEGORICAL FEATURES

For regression algorithms, all features have to be numeric or they can be used in the algorithm at all. However in reality a lot of dataset contains many categorical features with string values.

There are many ways to convert categorical features in to numeric ones, and the most common methods are ordinal, one-hot, and binary. The ordinal method maps each unique string value of the categorical feature to a unique integer starting from 0. This method does not increase the dataset's dimensionality but it implies the feature is ratio, although it is definitely not, in most cases. So I did not use the ordinal method. The one-hot method creates as many "dummy features" as the number of distinct values in the categorical feature. A "1" in the dummy feature means whatever category this dummy feature represents is present in the original categorical feature, and a "0" means opposite. So for each row there should only be one "1" and the rest should be "0" among generated dummy features. The binary method encodes each distinct categorical feature value to a binary code. Let the number of distinct values in the original categorical feature equals N, the number of dummy features created would be $\log_2 N$, which is significantly fewer than the one-hot method, which is N.

However, I chose the one-hot method over the binary in the end, because the dummy features it creates are more interpretable.

The next step is to drop one dummy feature for each group of dummy features generated to avoid perfect collinearity, because the linear model assumes its absence, and the presence of it would throw the linear model off. Collinearity means you can predict one independent variable with other

independent variables, and perfect collinearity means the prediction is always correct.

One last essential step in the pre-processing is to drop all rows with missing values. Although some algorithms like random forest can deal with missing values, the base line algorithm ordinary least squares cannot, so they have to be dropped.

### *C. Model Building*

#### *I. ALGORITHMS*

Ordinary least squares linear regression is used as the base line. Besides, k-nearest-neighbor regression, random forest regression and multi-layer perceptron regression are also used.

#### *II. METRICS*

Since the project's focus is regression, usual regression metrics like root mean squared error [RMSE] and $R^2$ are used to evaluate model performance. In addition, I included a metric of my own – mean absolute percentage error [MAPE], which measures the average percentage difference between the predicted value of the true value, because root mean square error does not measure the error relative to the true value, which makes it less interpretable.

#### *III. FURTHER FILTERING OF FEATURES*

After encoding using the one-hot method the categorical features in the 47-feature dataset, I ran the four algorithms, and they performed terribly except random forest regression. Ordinary least squares linear regression, k-nearest-neighbor regression, and multi-layer perceptron regression all had negative $R^2$ values.

The problem partly lies in the peculiarity of the New York City data, because user "mdzhang" on kyso.io [2] was able to easily fit a simple linear model on the Austin, Texas Airbnb dataset with $R^2$ equals 0.594. The problem also partly lies in that the dataset needs further cleaning and dimension reduction.

From the 47 remaining features, I removed "host_since", "host_has_profile_pic", "host_identity_verified", "calendar_updated", "first_review", "require_guest_profile_picure", "require_guest_phone_verification", "calculated_host_listings_count", and "last_review" for they bear little relevence.

"review_scores_accuracy", "review_scores_cleanliness", "review_scores_checkin", "review_scores_communication", "review_scores_location", and "review_scores_value", were dropped and only the aggregate review score ("review_score") were kept.

"availability_60", "availability_90", and "availability_356" were dropped because they appear to have high collinearity with "availabitliy_30". "beds" was dropping due to high collinearity with "accommodates". "square_feet" was dropped because it has too many missing values. "weekly_price", "monthly_price", "security_deposit", and "cleaning_fee" were dropped because I thought it was unfair to regress the listing price using other related prices.

In the end, there were 22 features left along with feature "sept" which I created to distinguish whether the listing was from the August scraping or September scraping when I appended the August and September dataset together to form one large dataset.

## III. Results

Because "neighborhood_cleansed" creates such a big overhead in dimensionality when encoded using one-hot encoder, I ran the algorithm with and without the feature using 10-fold cross validation, and juxtaposed the different test set results.

| "neighborhood_cleansed" dropped | | | |
|---|---|---|---|
| | RMSE | $R^2$ | MAPE |
| OLS (baseline) | 147.062 | 0.338 | 49.505% |
| KNN (k=11) | 163.692 | 0.172 | 60.440% |
| **RF (21 trees)** | **103.931** | **0.662** | **23.920%** |
| MLP | 6250.015 | -4366.106 | 138.878% |

| "neighborhood_cleansed" included | | | |
|---|---|---|---|
| | RMSE | $R^2$ | MAPE |
| OLS (baseline) | 6945.110 | -11717.508 | 212.860% |
| KNN (k=11) | 163.532 | 0.173 | 60.269% |
| **RF (21 trees)** | **102.420** | **0.673** | **23.142%** |
| MLP | 7388.512 | -6242.207 | 134.509% |

Evidently, including "neighborhood_cleansed" generally makes the models perform worse. It renders the ordinary least squares linear regression model completely useless and decreases the $R^2$ score for all four algorithms. When it does improve the metrics, it only does so by an insignificantly small amount.

Linear regression does well enough as a benchmark. Out of the other 3 algorithms, random forest out-performs the rest in both cases by a large margin. The reason is perhaps its ability to form complex decision boundaries and it being the only ensemble algorithm. K-nearest-neighbor does not perform as well as I had expected it to be. One possible explanation is that New York City's Airbnb listings have a lot of outliers.

As to the reason multi-layer perceptron does not work, I cannot give one, because I am not familiar with how the algorithms works exactly.

## IV. Related Work

There are many projects that regress Airbnb listing price online, but Stanford University's Kalehbasti, Nikolenko, and Rezaei's *Airbnb Price Prediction Using Machine Learning and Sentiment Analysis* [1] is the one to look to when it comes to New York City's Airbnb listings. They tuned "the coefficient of linear regression model with Lasso Regularization trained on the train split", and picked the best

performing model. Using that model, they selected "78 features with non-zero values". With this feature selection method, they were able to reach an $R^2$ score of 0.690, and mean squared error of 0.147. Because they used natural log to normalize the predicted and true prices before computing the MSE, I cannot make a direct comparison between my model's score and theirs. Nonetheless, Kalehbasti, Nikolenko, and Rezaei's automated feature importance analysis does seem to be a promising way to improve regression accuracy.

## V. Conclusion

Using manual feature selections and random forest algorithm, I was able to predict New York City's Airbnb listing's price with 23% deviation from the true price on average and with an $R^2$ score of 0.67.

Realistically, the easiest way to improve prediction accuracy is probably by building a model for each neighborhood instead of one across the entire city. This way each model faces less variance in the dataset. There might be a problem of insufficient number of samples in some neighborhoods in Staten Island, however.

To improve on the city-wide model, the "neighborhood_cleansed" feature needs to be incorporated model in some way, probably by using certain feature importance analysis method to reduce the number of dummy features created from it. Currently my best-performing model disregards neighborhood information completely, and regresses on "host_response_time", "host_response_rate", "host_is_superhost", "neighborhood_group_cleansed", "latitude", "longitude", "is_location_exact", "property_type", "room_type", "accommodates", "bathrooms", "bedrooms", "bed_type", "guests_included", "minimum_nights", "maximum_nights", "avalibliity_30", "number_of_reviews_ltm", "review_scores_rating", "instant_bookable", "cancellation_prolicy", and "sept".

Besides, tuning my model using a validation set is also needed.

## VI. References


[1] P. R. Kalehbasti, L. Nikolenko, and H. Rezaei, "Airbnb Price Prediction Using Machine Learning and Sentiment Analysis," *arXiv.org*, 29-Jul-2019. [Online]. Available at: https://arxiv.org/abs/1907.12665. [Accessed: 12-Dec-2019].

[2] mdzhang (2019). Predicting Airbnb Prices in Austin. [online] Kyso. Available: https://kyso.io/mdzhang/predicting-airbnb-prices-in-austin [Accessed 13 Dec. 2019].