Springboard--Data Science Career Track
Airbnb price in New York City
Capstone Project 1 Final Project
By Kin Sun
June 2020

1. Introduction:

With so many Airbnb available in NYC, determining the price of one's Airbnb listing or understanding what drives the price of an Airbnb listing before purchasing an investment property can be a headache. The 48,000-samples dataset I am planning to use—which I will obtain from Kaggle[1]—contains some features that can be used to predict the price per night of a listing in NYC for the year of 2019. Prospective clients are any individuals, property management companies, existing Airbnb hosts, new investors—anyone who just wants to start getting involved with Airbnb. These clients would be able to use the results of this project to find out how much they can reasonably charge per night and also get an idea of what is driving the price up.

The codes and results are all uploaded to my Github repository:

https://github.com/kinlongsun/Springboard/tree/master/Capstone%20Project%201

2. Approach:

(2.1) Data Description:

Feature	Description
id	listing ID
name	name of the listing
host_id	host ID
host_name	name of the host
neighbourhood_group	location
neighbourhood	area

[1] https://www.kaggle.com/dgomonov/new-york-city-airbnb-

latitude	latitude coordinates
longitude	longitude coordinates
room_type	listing space type
price	price in dollars, per night
minimum_nights	amount of nights minimum
number_of_reviews	number of reviews
last_review	latest review
reviews_per_month	number of reviews per month
calculated_host_listings_count	amount of listing per host
availability_365	number of days when listing is available for booking

(2.2) Data Wrangling

1 - Dropping features

The dataset has some features that serve no purpose for my analysis. For example, id and host_id are identifiers of the listing and the host. Another two features that will be removed are name and host_name. Lastly, the last_review feature, which is the date of the last review left to the listing, may be useful for NLP analysis but for our purpose, but was not used in this project.

2 - Missing Variables

By utilizing Panda's ".isnull().sum()," the following result was generated:

```
neighbourhood_group                 0
neighbourhood                       0
latitude                            0
longitude                           0
room_type                           0
price                               0
minimum_nights                      0
number_of_reviews                   0
reviews_per_month               10052
calculated_host_listings_count      0
availability_365                    0
dtype: int64
```

There are 10,052 null values for the feature "reviews_per_month."

Below, rows with null values were selected:

	number_of_reviews	reviews_per_month
2	0	NaN
19	0	NaN
26	0	NaN
36	0	NaN
38	0	NaN
...	...	...
48890	0	NaN
48891	0	NaN
48892	0	NaN
48893	0	NaN
48894	0	NaN

10052 rows × 2 columns

By summing the "number_of_reviews" reveals that 0 "number_of_reviews" is the reason for "reviews_per_month" to be a null value. One option to deal with these null values is to remove those rows. However, the size is over 20% of our dataset. Luckily, there is a better and wiser way to deal with the null values here, and that is to simply replace them with 0's. This option is viable since 0 number of reviews should have an average of 0 reviews per month.

Another issue was that the dataset contains 0 for price, showed below:

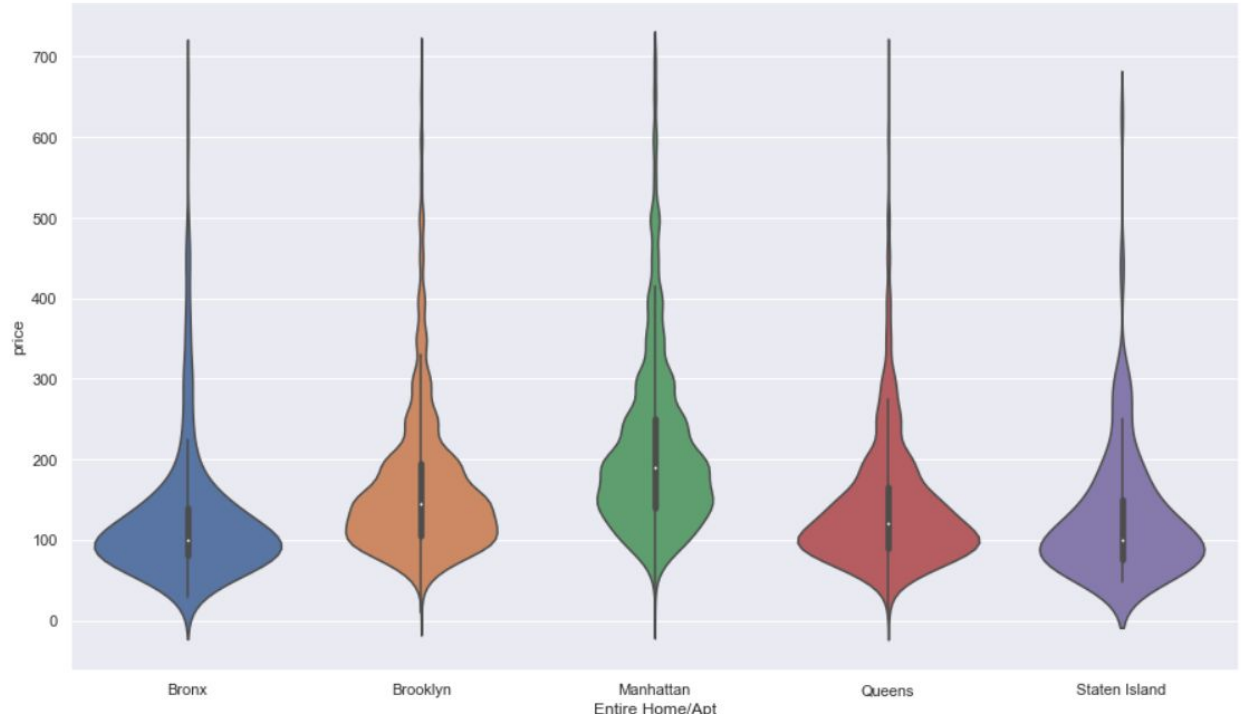
```
raw_data.describe(include='all')
```

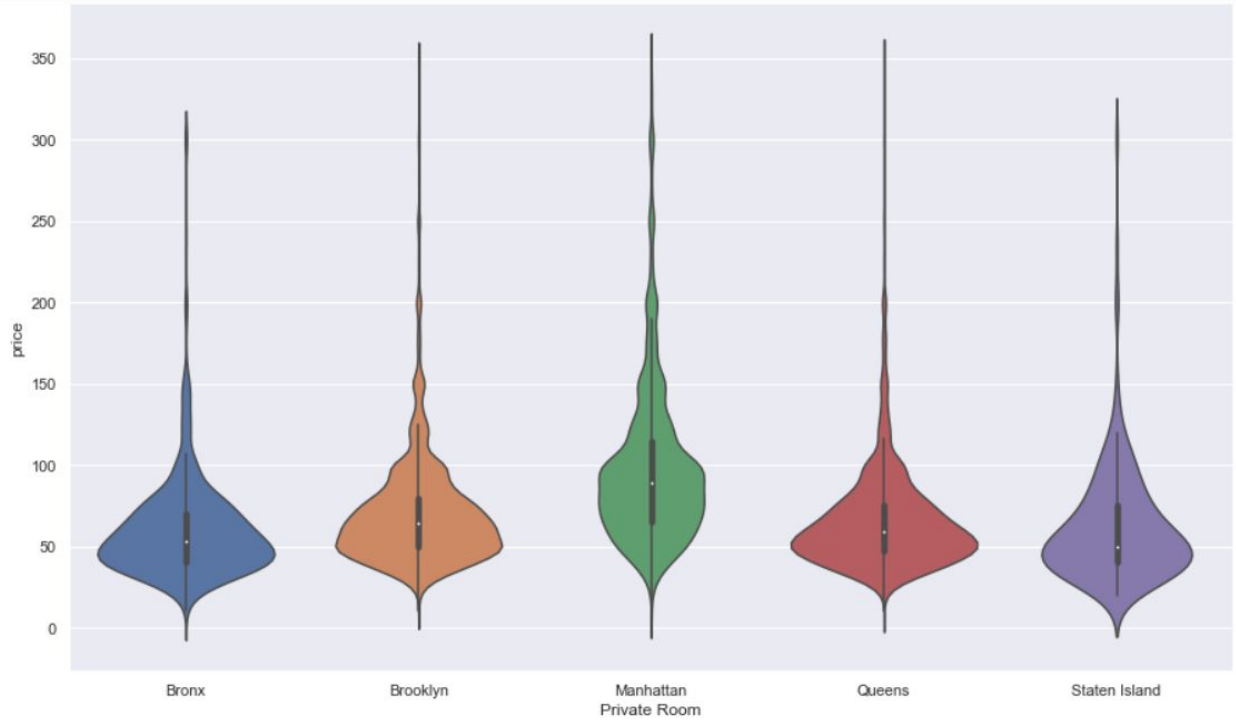
	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price
count	48895	48895	48895.000000	48895.000000	48895	48895.000000
unique	5	221	NaN	NaN	3	NaN
top	Manhattan	Williamsburg	NaN	NaN	Entire home/apt	NaN
freq	21661	3920	NaN	NaN	25409	NaN
mean	NaN	NaN	40.728949	-73.952170	NaN	152.720687
std	NaN	NaN	0.054530	0.046157	NaN	240.154170
min	NaN	NaN	40.499790	-74.244420	NaN	0.000000
25%	NaN	NaN	40.690100	-73.983070	NaN	69.000000
50%	NaN	NaN	40.723070	-73.955680	NaN	106.000000
75%	NaN	NaN	40.763115	-73.936275	NaN	175.000000
max	NaN	NaN	40.913060	-73.712990	NaN	10000.000000

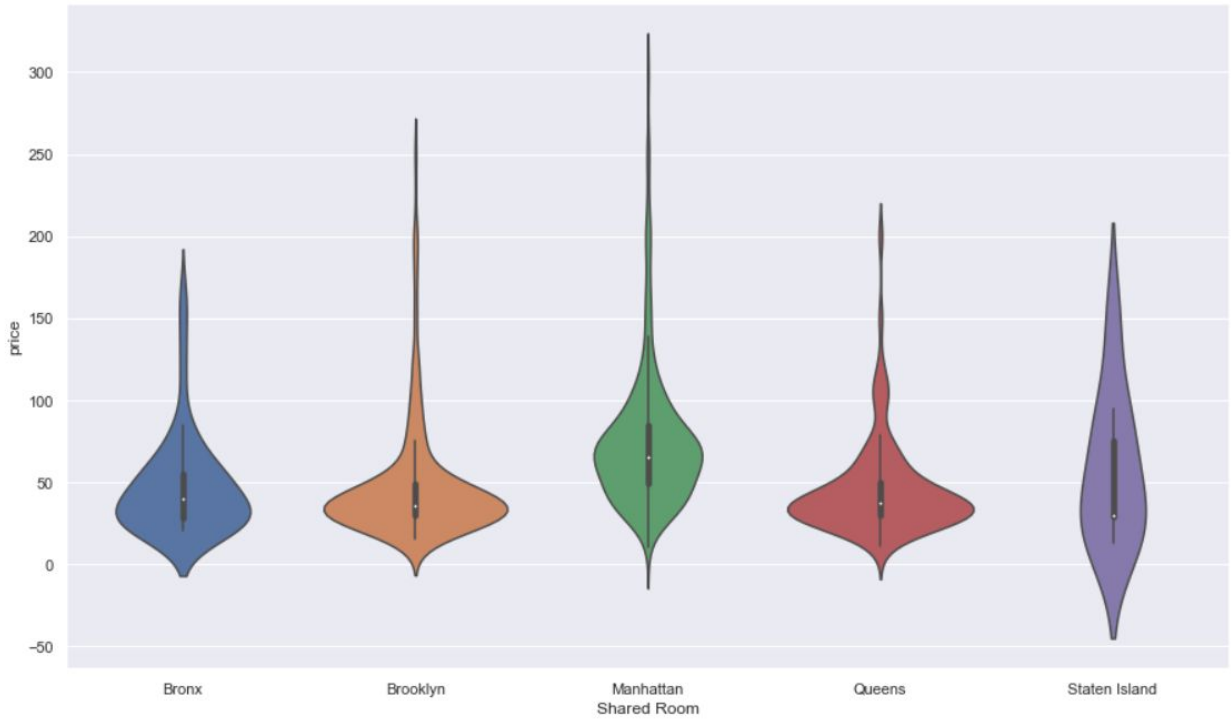
This is an issue because the problem we are interested in solving is how much people can charge on their listings. There are only 11 of those instances and they will be removed from our final data.

(2.3) Storytelling and Inferential Statistics

Below are 3 violin plots, separated by room types, showing the price distribution of the 5 New York City boroughs.







We can easily see that the order of price is Entire Home/Apt > Private Room > Shared Room. Note that the visual effect may be subtle, due to the long tail end, but price at Manhattan is the highest out of all 5 boroughs.

Take a look at the heatmap below.

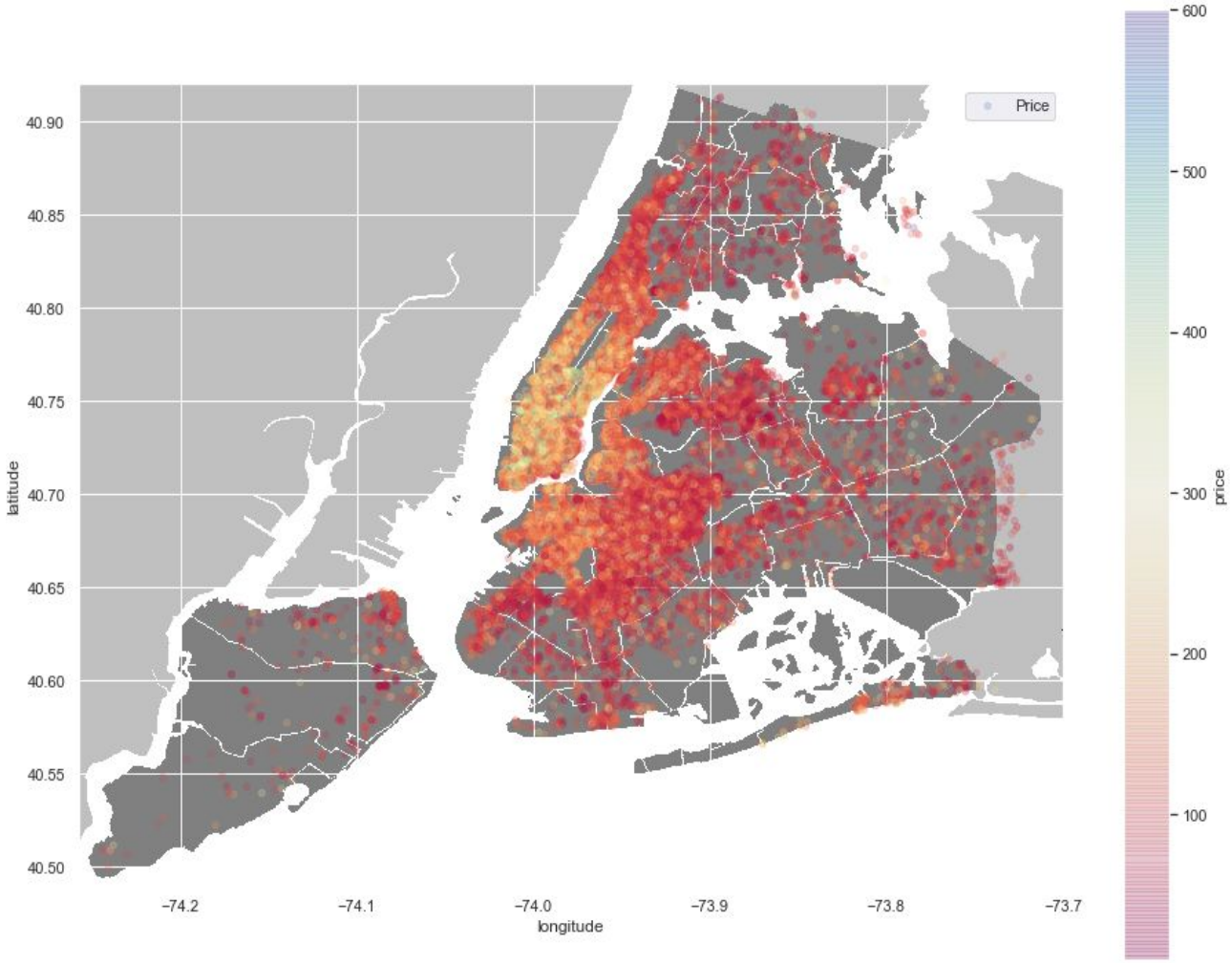
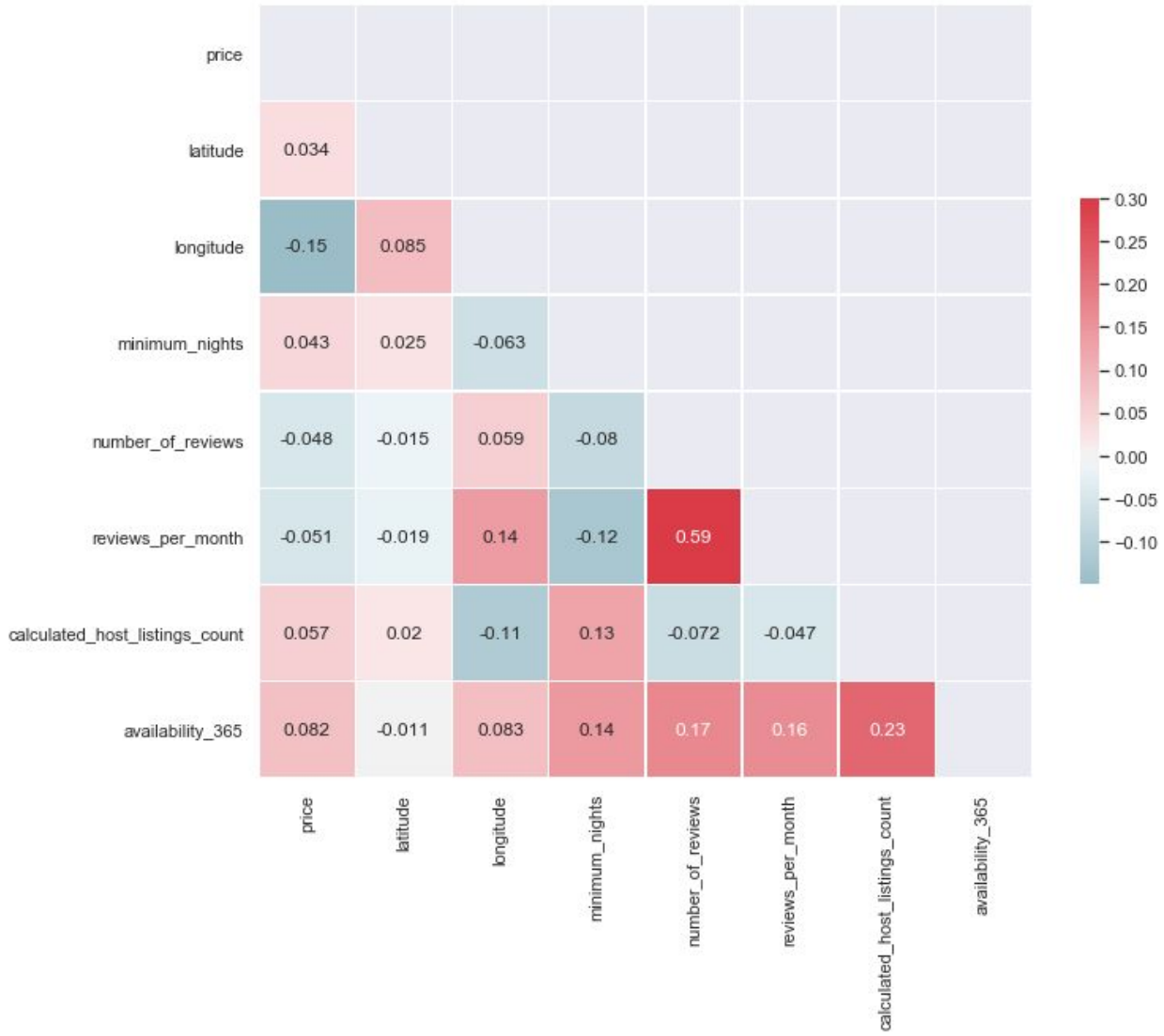


Figure 0 - NYC Heat Map

Dataset was slightly modified to remove any price higher than $600.00, which affected roughly 778 rows of data. The reason to do that is to prevent distortion on the heatmap caused by distinctly high price listings. What the heat map shows is that the closer to downtown Manhattan, the more expensive the Airbnb listing prices are. Note that we also observe the higher price in Brooklyn as well, as it gets closer to Manhattan.

Below is a heatmap for correlation coefficient of the numerical (noncategorical) features:



We see that the correlations are not strong between any of those variables, with exception of "reviews_per_month" and "number_of_reviews," but it is still not very strong. This is a good thing and it will be interesting when doing feature selection later on.

(2.4) <u>Baseline Modeling</u>

Before building my baseline models, I used train_test_split and split 75% of my data into training set and 25% into my testing set. I also utilized Pandas' "get_dummies" on my categorical variables.

The first model was built using the training data and Sklearn Linear Regression class. Below are the features used and the target variable is the price.

['minimum_nights',
 'number_of_reviews',
 'reviews_per_month',
 'calculated_host_listings_count',
 'availability_365',
 'neighbourhood_group_Brooklyn',
 'neighbourhood_group_Manhattan',
 'neighbourhood_group_Queens',
 'neighbourhood_group_Staten Island',
 'room_type_Private room',
 'room_type_Shared room']

The figure below is a scatterplot of y_test(price of testing set) and the y_hat_test (predicted value of testing set).

Figure 1 - LR scatterplot of testing set's Actual Price vs Predicted Price

Figure 2 below is a scatterplot of y_hat_test(predicted value of testing set) vs the Residual.

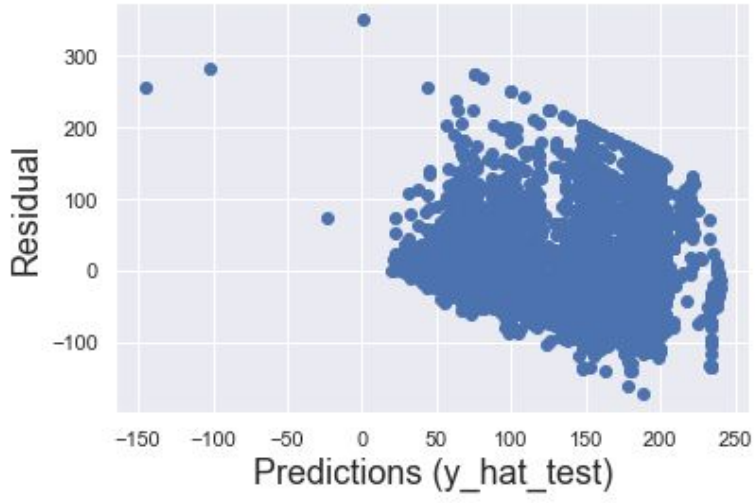


Figure 2 - LR scatterplot of Predicted Price vs Residual

Figure 3 & 4 below are QQ-plot of the Residuals and histogram of Residuals

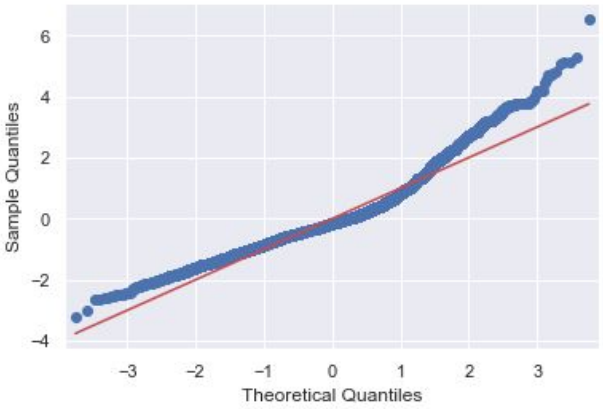


Figure 3 - QQ-plot of residuals(Linear Regression Model)

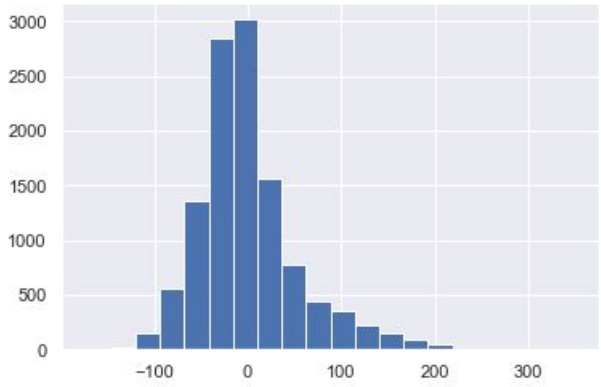


Figure 4 - Histogram of residuals

In figure 1 and 2, we can see that the model is not flawless, but it is solid. In figure 3 and 4, we see that the residuals look Normal.

Note that all results were evaluated on the testing dataset. The Root Mean Square Error(RMSE), R-Squared and Mean Absolute Percentage Error (MAPE) are 53.490615, 0.451247 and 37.961281%, respectively. In the later section, we will evaluate our model using these metrics, but loosely speaking, we want to minimize errors while maximizing R-Squared.

I also dug deeper into other models like Ridge Regression and Lasso Regression with hyperparameter tuning on Alpha using GridSearchCV. What I found was that the RMSE, R-Squared and MAPE were almost identical to the Linear Regression Model.

(2.5) Extended Modeling

The Linear Regression Model's evaluation metrics left a lot of room for improvement, so I decided to use a more advanced model, the Random Forest Regression.

To start, I used the same training and testing set, and selected a few parameters for hyperparameter tuning using RandomizedSearchCV. I also used the 'neg_root_mean_squared_error' scoring option. The random grid is as follow:

```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [100, 600, 1100, 1600, 2100, 2600, 3100, 3600]}
```

After fitting the random search, the best parameters are:

```
{'n_estimators': 600,
 'min_samples_split': 5,
 'min_samples_leaf': 2,
 'max_features': 'auto',
 'max_depth': 10,
 'bootstrap': True}
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=10, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=2,
                      min_samples_split=5, min_weight_fraction_leaf=0.0,
                      n_estimators=600, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

Note that all results were evaluated on the testing dataset (after finding the optimal hyperparameters). Upon checking the MAPE, there were some improvements from the Linear Regression model. I wanted to see if I can quickly improve my model a little more by doing a quick GridSearchCV, near the tuned best parameters from the RandomizedSearchCV, again, 'neg_root_mean_squared_error' as the scoring option..

With minor change in parameters, my evaluation metrics actually got worse off than the randomized search result on the testing data. Since the difference occurred in the 10th and 100th decimal, I decided to just stick to this Random Forest model.

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=10, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=5, min_weight_fraction_leaf=0.0,
```

n_estimators=600, n_jobs=None, oob_score=False,
random_state=None, verbose=0, warm_start=False)

(3) Findings

Below are the summary of evaluation metrics. Recall that a model tends to be more desirable if the RMSE and MAPE are lower, and R-Squared is higher. We can easily see in *figure 5* that the Random Forest model is noticeably better than the Linear Regression model even though it is not by much.

	RMSE	R2	MAPE
LR_Test	53.490615	0.451247	37.961281
Ridge_Test	53.490610	0.451247	37.964141
Lasso_Test	53.489367	0.451273	37.967859
RF_Test	51.004630	0.501069	35.520892

Figure 5 - Metrics Summary Table

It is also worth looking at the 5th percentile and 95th percentile of the testing dataset residuals for our models:

Linear Regression:
0.05 -72.439109
0.95 109.387873
Name: price, dtype: float64

Random Forest
0.05 -71.255439
0.95 102.666099
Name: price, dtype: float64

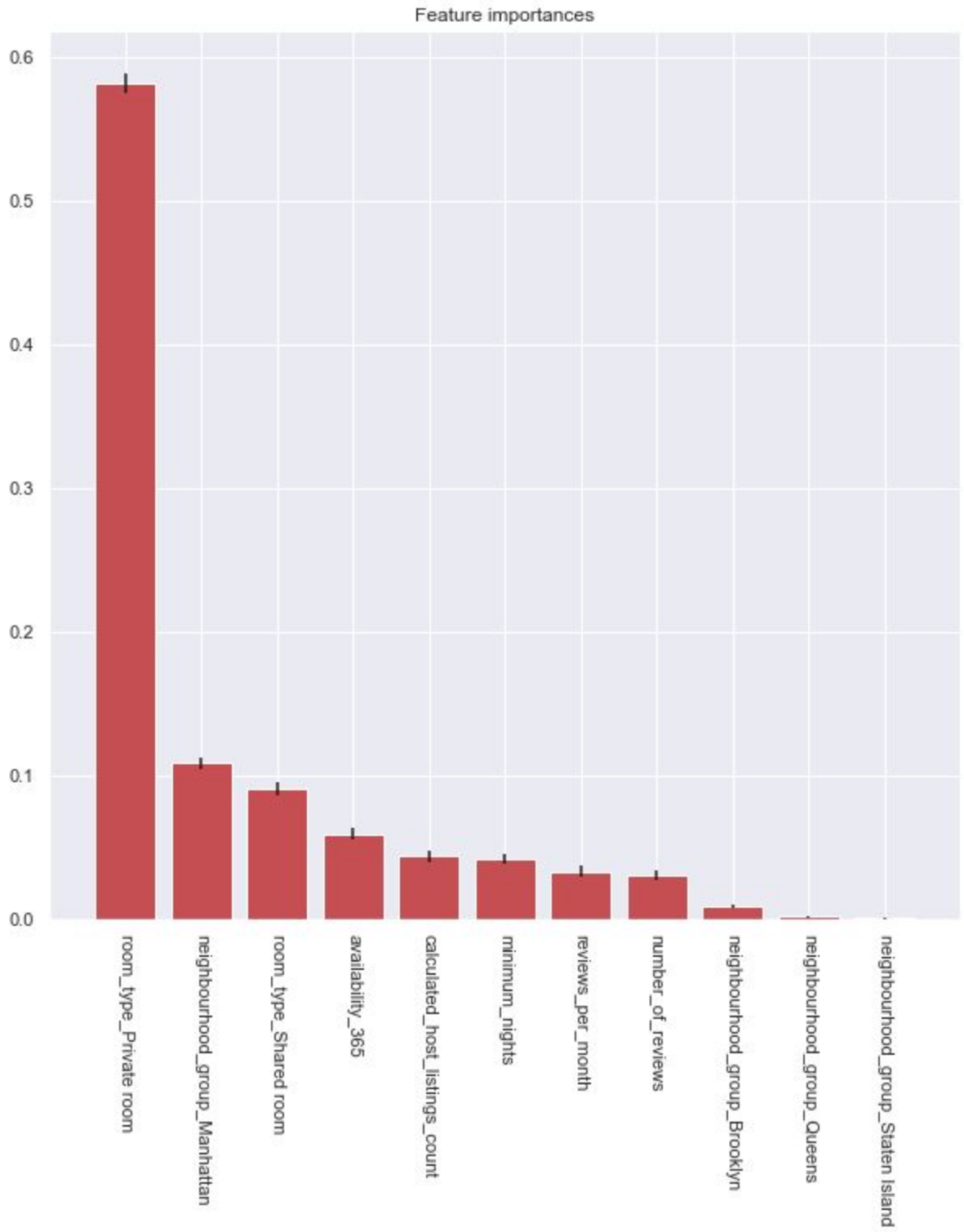


Figure 6 - Feature Importance

	Features	Coefficients	F-statistics	P-values
0	minimum_nights	-0.242368	32.901	0.0
1	number_of_reviews	-0.047594	37.329	0.0
2	reviews_per_month	-0.825077	93.175	0.0
3	calculated_host_listings_count	0.127663	1036.314	0.0
4	availability_365	0.055885	133.450	0.0
5	neighbourhood_group_Brooklyn	22.851811	1158.846	0.0
6	neighbourhood_group_Manhattan	54.261456	4140.439	0.0
7	neighbourhood_group_Queens	11.299690	1000.244	0.0
8	neighbourhood_group_Staten Island	-1.378366	63.120	0.0
9	room_type_Private room	-80.827102	16462.094	0.0
10	room_type_Shared room	-104.342369	655.038	0.0

Figure 7 - Summary of coefficients of Linear Regression Model.

In *Figure 6*, we can clearly see the importance of a private room, and shared room, as opposed to the default entire house/apt. It also looks like locating in Manhattan also has a high impact on the price as well.

In figure 7, the coefficients of the linear regression model were summarized by the chart. Holding all else constant, by locating in Manhattan, as opposed to the default Bronx, the price of the house will be higher. If the listing is a private room or a shared room, the estimated price will be lower.

(4) Conclusions and Future Work

Overall, the Linear Regression Model and Random Forest Model have no overfitting problem, but the R-Squared, RMSE, and MAPE aren't stellar. There are other models or implementations I could try:

- Split the data by boroughs and run models for each of them
- Since we know the coordinate of the listing, we can determine the distance of the closest:
    - Downtown
    - Park
    - Beach
    - Shopping center
    - Crime rate associated with the coordinates' zip code

(5) Recommendations for the Clients

Referring to the figures and findings in **Section (3)**, here are some recommendations for my clients:

1. Location: In *figure 0*, we noted that Manhattan listings tend to have higher listing prices, and our models support that claim. We can see that in our Linear Regression model and *figure 7*, holding all else constant, listings located in Manhattan increase the highest relative to all other boroughs.. This importance of being in Manhattan is also high in our Random Forest Model. So for my clients who are interested in an investment property, be sure to put this into consideration.

2. Room type: Refer by back *figure 6* and *figure 7*. We see that in *figure 6*, a private room has significant importance while a shared room also has high importance. This is consistent with *figure 7*, where we see that in the Linear Regression Model, a listing of private room and shared room will lead to a decrease in price--This is all while holding other variables constant.

3. Recall the residuals 5th percentile and 95th percentile mentioned earlier,

   Linear Regression:
   0.05 -72.439109
   0.95 109.387873
   Name: price, dtype: float64

   Random Forest
   0.05 -71.255439
   0.95 102.666099
   Name: price, dtype: float64

   If we look at the Linear Regression model's residuals, the worst case scenario is that the predicted value is undershooting by $109.39 or overshooting by $72.44. Likewise, the worst case scenario for the Random Forest model is that the predicted value undershoot by $102.67 or overshoot by $71.26. My clients can use this information to adjust their listing price.

(6) Consulted Resources

Scikit-learn.org - Used for sklearn packages documentations
Stackoverflow.com - Read questions asked by others who had similar coding questions