Muskan Burman
May 2, 2022
Data Science Capstone
Prof. Imielinski

NYC Airbnb Dataset: Final Report

Introduction

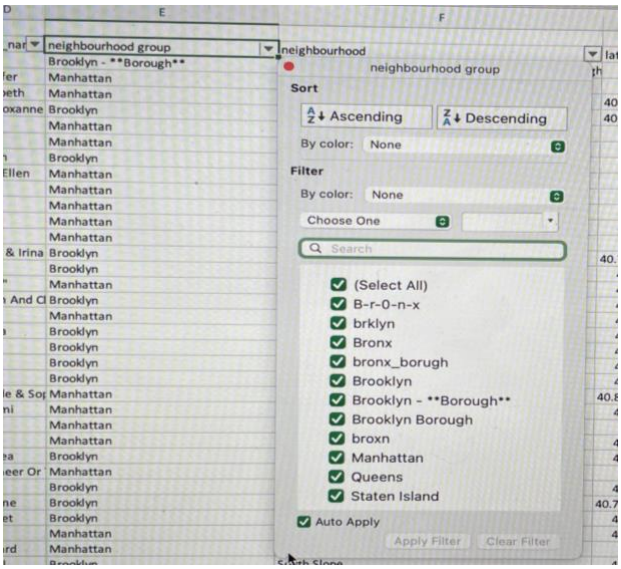
Through the Data Science Capstone Project, we were tasked to tell a story through data. We were provided with a dataset of Airbnbs in NYC that contained multiple attributes such as noise levels, neighbourhood groups, room types, etc. and we were tasked to find the relationship of these attributes with the price of the Airbnbs and come up with a prediction model that best described those relationships. We had multiple checkpoints throughout the semester, starting with Data Cleaning wherein we had to effectively clean our given dataset by identifying and solving problems to make our data importable to R for further inspection, leading us into checkpoint 2, that was Exploratory Data Analysis. Our task here was to look for trends and patterns within the data and find relationships between the various attributes and visualize them using plots, charts, tables, etc. We then moved into checkpoint 3, which was developing a Prediction Model for our dataset that utilized the relationships between price and other attributes found through EDA in the best possible way and apply them to create an effective model. All of us then participated in a Kaggle competition to check the accuracy of our models and compare it with the rest of the class. Finally, we all participated in Playoffs hosted on Kaggle wherein we were given the same dataset but with new patterns and we had to reimplement all work done throughout the semester on this new dataset.

Data Cleaning

I was introduced to the dataset in this section. We were given an unclean dataset and were tasked with identifying problems and finding solutions to effectively clean the dataset so that it can be imported to R for exploration and prediction purposes.

Identifying the problems:

I used various functions of Python and Excel to give a proper look over to the data. For example, I used Excel's filter function on each of the columns to see the different values/tuples of the columns, and whether they needed cleaning.



Here, we can see that the “neighbourhood group” column has many unnecessary values that need replacing.

I came up with the following list of modifications:

- Make Headers (Title Rows).
- Remove the column full of just NA values.
- Clean the”[host_name]” column name.
- Clean ”id” values.
- Clean ”host id” values.
- Clean “neighbourhood group” values.
- Clean “neighbourhood” values.
- Clean the latitude and longitude values.
- Clean the “room and type” values.
- Clean the “minimum nights” values.
- Replace the NA values in the “reviews per month” column.
- Clean the “floor” values.
- Clean the ”noise.dB.” values.
- Clean the “price” values.

Overcoming the problems:

I used python through Jupyter Notebooks and made use of tools like Pandas and Numpy.

Some of the functions that I used included:

- **iloc** for creating headers for my dataframe
- **dropna** for removing the one column of NA values
- **rename** for changing column names
- **replace** for updating values of the neighbourhood group, neighbourhood, room type etc., columns
- **map** and **lambda** for removing unnecessary trailing characters from the price column

- **astype** for converting all values of the floor column to numeric

After making all these changes and some others, my dataset was ready to be imported to R for the next step, i.e., Data Exploration and Analysis.

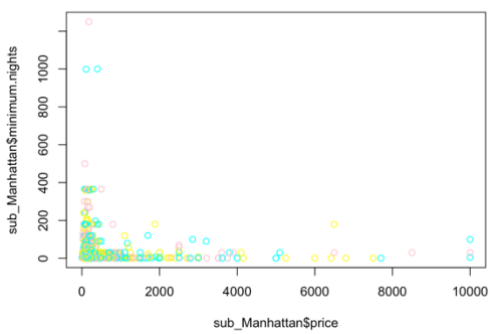
Data Exploration

In this section, I started by subsetting my dataset to look at the relationships of the various attributes with the price variable. I first subsetted based on the neighbourhood groups, resulting in 5 different sets, and then further based on the room type, resulting in 15 more sets to get a total of 20. I then used these subsets, as required, to identify the factors that affected the price of an Airbnb room.

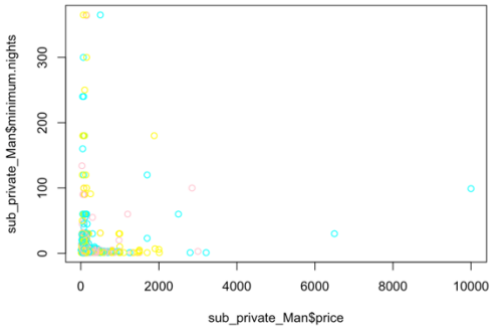
```
#Subsetting based on neighbourhood group

sub_Manhattan <- subset(airbnb_data, neighbourhood.group == "Manhattan")
sub_Bronx <- subset(airbnb_data, neighbourhood.group == "Bronx")
sub_Brooklyn <- subset(airbnb_data, neighbourhood.group == "Brooklyn")
sub_Queens <- subset(airbnb_data, neighbourhood.group == "Queens")
sub_Staten <- subset(airbnb_data, neighbourhood.group == "Staten Island")

#private rooms
sub_private_Man <- subset(sub_Manhattan, room.and.type == "Private room")
sub_private_Bronx <- subset(sub_Bronx, room.and.type == "Private room")
sub_private_Brooklyn <- subset(sub_Brooklyn, room.and.type == "Private room")
sub_private_Queens <- subset(sub_Queens, room.and.type == "Private room")
sub_private_Staten <- subset(sub_Staten, room.and.type == "Private room")

#shared rooms
sub_shared_Man <- subset(sub_Manhattan, room.and.type == "Shared room")
sub_shared_Bronx <- subset(sub_Bronx, room.and.type == "Shared room")
sub_shared_Brooklyn <- subset(sub_Brooklyn, room.and.type == "Shared room")
sub_shared_Queens <- subset(sub_Queens, room.and.type == "Shared room")
sub_shared_Staten <- subset(sub_Staten, room.and.type == "Shared room")

#entire homes/apts
sub_entire_Man <- subset(sub_Manhattan, room.and.type == "Entire home/apt")
sub_entire_Bronx <- subset(sub_Bronx, room.and.type == "Entire home/apt")
sub_entire_Brooklyn <- subset(sub_Brooklyn, room.and.type == "Entire home/apt")
sub_entire_Queens <- subset(sub_Queens, room.and.type == "Entire home/apt")
sub_entire_Staten <- subset(sub_Staten, room.and.type == "Entire home/apt")
```

Since Manhattan had the greatest number of rows among the 5 neighborhoods, I decided to research it further.

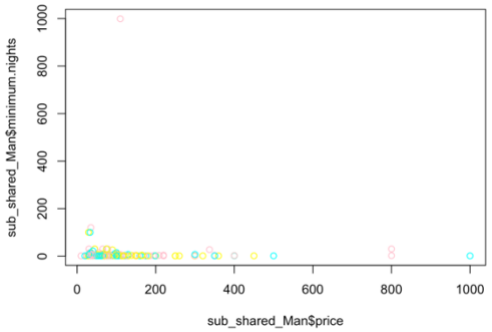
Price vs Minimum nights for Manhattan



Places that cost less usually have lower number of minimum nights.



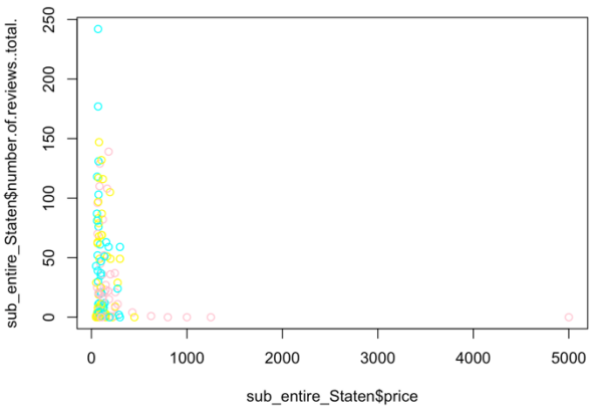
While the observations remain same while looking specifically at private rooms in Manhattan, we observe many outliers here.



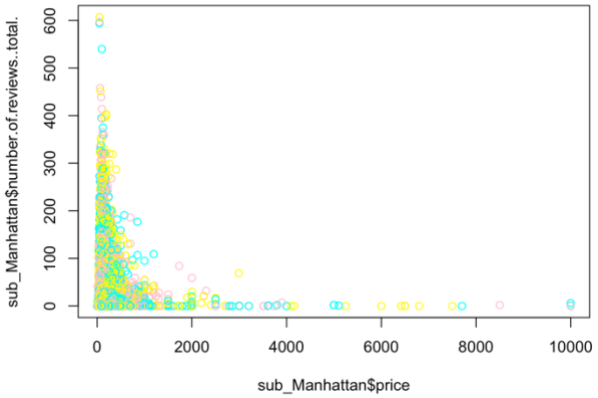
However, when we specifically look at all the shared spaces in Manhattan, we can see that regardless of price, the minimum nights remain the same.

I found that this was a general trend across all neighbourhood groups.

Price vs Total number of review for Manhattan



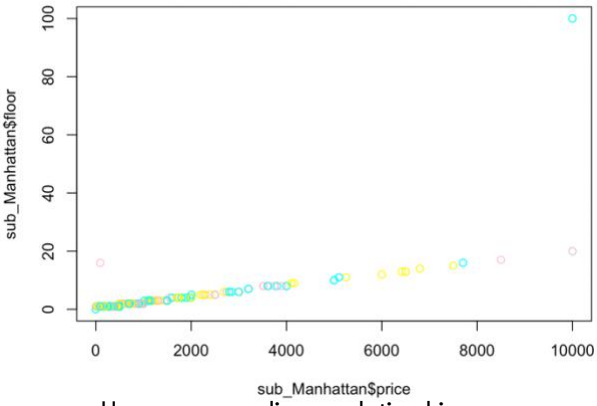
Places that cost less usually have a higher number of total reviews. This might be a result of the fact that since these places are more affordable, more people are able to get them, leading to a greater number of reviews.



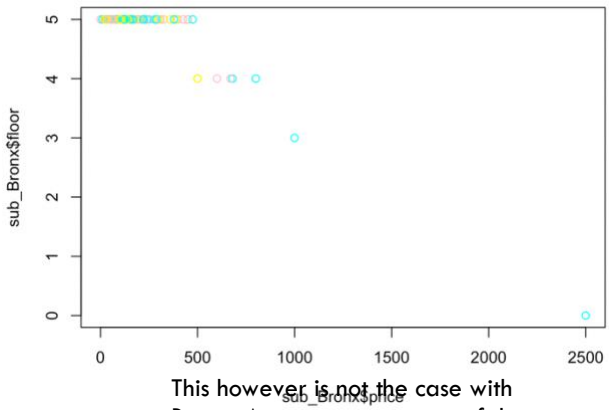
Similar is the case for entire houses in Staten Island.

Yet again, I found that this was a general trend across all neighbourhood groups.

Price vs Floor for Manhattan

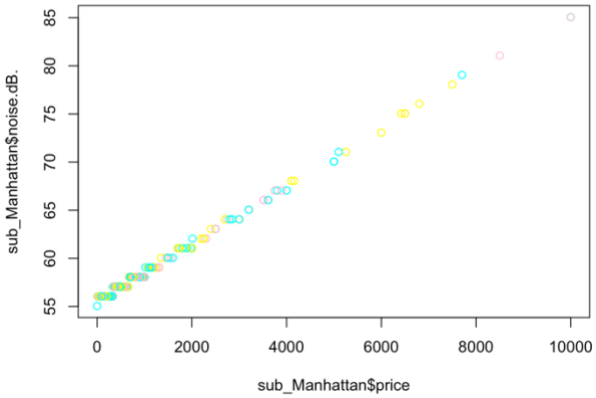


Here, we see a linear relationship between the price and the floors, indicating that the price increases as the level of floor increases – the higher the floor, the higher is the price.

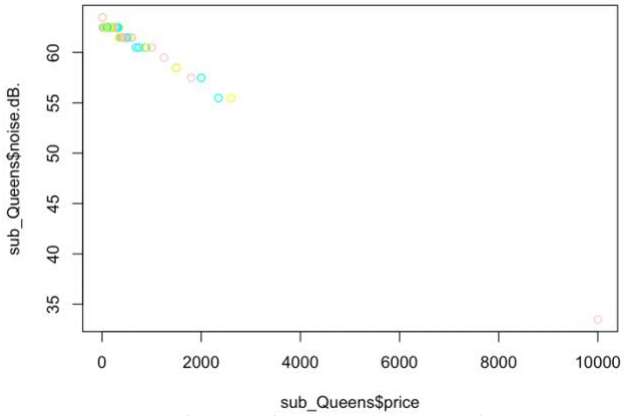


This however is not the case with Bronx. As we can see, most of the Airbnbs in Bronx are on the fifth floor, with a few outliers, thus giving us a very vague

Price vs Noise for Manhattan



Here, we again see a linear relationship between the price and the noise levels, indicating that the price increases as the noise levels do, which although odd, makes sense in a city like Manhattan where everything is very lively.



Such is not the case in Queens, because prices increase as noise levels decrease.

Points of Interest

To find more patterns within the data, I looked at different places in NYC that are popular tourist/transportation spots to see if their location affected the prices of Airbnbs in that area. My list of points of interest included the 9/11 memorial, Central Park, Empire State Building, Statue of Liberty, Madison Square Garden, and the JFK Airport. I will talk more in depth about how I incorporated these POIs in my prediction models in the next section.

Prediction challenge 1

To develop my prediction model, I started by dividing the given dataset into training (80%) and testing (20%) sets.

```
# splitting the dataset into training and testing.
idx <- sample( 1:2, size = nrow(airbnb_data), replace = TRUE, prob = c(.8, .2))
train <- airbnb_data[idx == 1,]
test <- airbnb_data[idx == 2,]
```

The first ML algorithm that I used was Random Forest, and I initially included all the attributes to get a basic idea of the relationship of price with different attributes. After acquiring an initial mse, I narrowed down my list of attributes to the following, according to the plots generated in the previous section:

neighbourhood.group + room.and.type + minimum.nights + number.of.reviews..total. + last.review..date.+ reviews.per.month + floor + noise.dB.

```
> random_forest <- randomForest::randomForest(factor(price) ~ neighbourhood.group +
+                                               room.and.type + minimum.nights +
+                                               number.of.reviews..total. + last.review..date.+
+                                               reviews.per.month + floor + noise.dB.,
+                                             data = train)
> random_forest

> mse(factor(train$price),pred.train)
[1] 48560.3
> pred.test <- predict(random_forest,newdata = test)
> mse(factor(test$price),pred.test)
[1] 35456.45
```

This combination of attributes resulted in producing an mse of 48560.3 for the training data, and an mse of 35456.45 for the testing portion of the data.

I also tried using other ML algorithms such as SVM, Linear Regression and Rpart, but none of them resulted in a better mse and error percentage, and so I decided to move forward with Random Forest.

Incorporation of POIs

Since this mse was still too large, I decided to implement some feature engineering and make use of the POIs identified in the previous section.

Idea: To check the proximity of every single listing to the above-mentioned 6 POIs, and see if that affects the accuracy of my model.

Implementation: To do so, I used the hutils package and the haversine_distance function. I made 2 lists of all the values of the latitudes and longitudes respectively, and then used the lats and longs of the different POIs to compare them, and created 6 new columns in my dataset with True or False values, representing if a particular listing is within 500m from a POI.

```
data_copy$proximitytomemorial <- hutils::haversine_distance(lat, long, memlat, memlon) < 500
data_copy$proximitytocpl <- hutils::haversine_distance(lat, long, cplat, cplon) < 500
data_copy$proximitytoemps <- hutils::haversine_distance(lat, long, empslat, empslon) < 500
data_copy$proximitytostatue <- hutils::haversine_distance(lat, long, statuelat, statuelon) < 500
data_copy$proximitytojfk <- hutils::haversine_distance(lat, long, jfklat, jfklon) < 500
data_copy$proximitytomsg <- hutils::haversine_distance(lat, long, msglat, msglon) < 500
```

Result: Using the above implementation, I added all new 6 columns to my Random Forest Model, and got an mse of 7580.843 for the training data, and an mse of 5220.389 for the testing portion of the data, which was a improvement from the older mses.

Problems: Even though these new mses were an improvement from the older ones, I had to remove

```
> random_forest2 <- randomForest::randomForest(factor(price) ~ neighbourhood_group +
+                                                room_type
+
+                                                 + floor + proximitytomemorial
+                                               + proximitytocpl + proximitytoemps
+                                               + proximitytostatue + proximitytojfk + proximitytomsg,
+                                              data = train)
> pred.train2 <- predict(random_forest2,newdata = train)
> mse(factor(train$price),pred.train2)
[1] 7580.843
> pred.test2 <- predict(random_forest2,newdata = test)
> mse(factor(test$price),pred.test2)
[1] 5220.389
```

many attributes like minimum nights, noise, etc. because of the large size of the dataset resulting in implementation errors.

Prediction Model on subsets

I then made 5 different prediction models using the same attributes for each of the 5 subsets. The Manhattan subsett produced the best results of the 5, resulting in an mse of 10832.87 for the training dataset and an mse of 6028.235 for the testing portion.

```
> random_forest_man <- randomForest::randomForest(factor(price) ~ neighbourhood_group +
+                       room_type + floor+ proximitytomemorial
+                       + proximitytocpl + proximitytoemps
+                       + proximitytostatue + proximitytojfk + proximitytomsg,
+                       data = train_man)
>
> pred.train_man <- predict(random_forest_man,newdata = train_man)
> mse(factor(train_man$price),pred.train_man)
[1] 10832.87
> pred.test_man <- predict(random_forest_man,newdata = test_man)
> mse(factor(test_man$price),pred.test_man)
[1] 6028.235
```

Since these mses were more than those produced by older prediction model, I decided to stick to my older model for my submission.

My final score on Kaggle upon submission of this model was 47680.98445

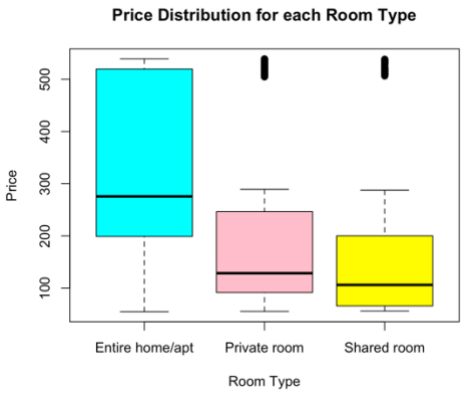
Prediction challenge 2 – Playoffs

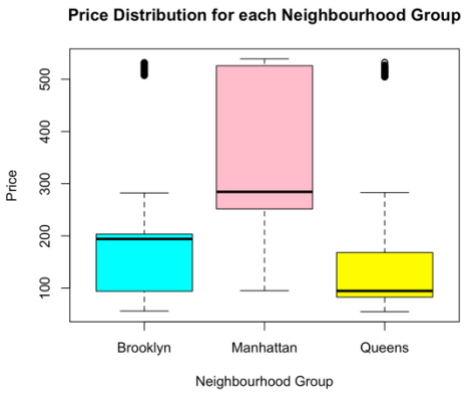
After the initial Kaggle competition, we were given the same dataset with new patterns for the Playoffs.

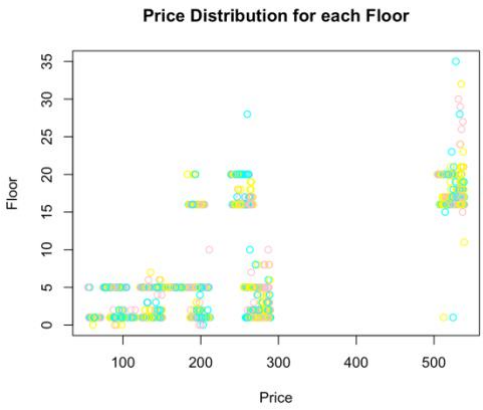
I followed similar steps for this dataset as the previous one, and since it was already clean, I started by making some plots to get some insight of the data.

Plots

I started by plotting the relationships between various attributes, and the following plots stood out to me the most.

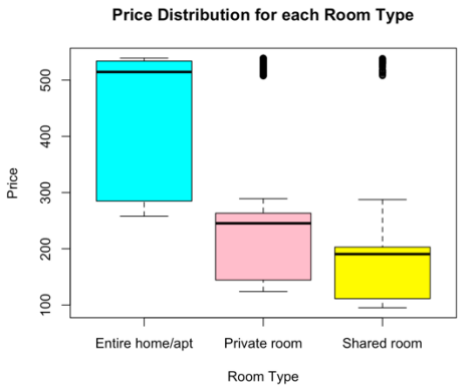


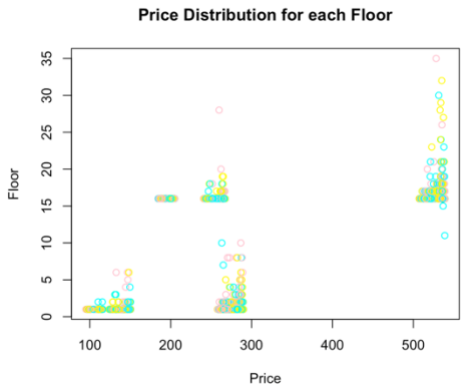


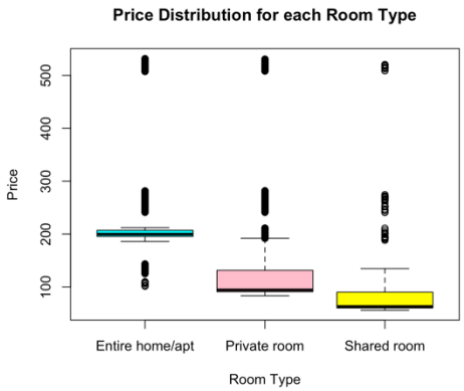


Looking at these plots, we can see that Entire home/apt is the most expensive room type and Manhattan is the most expensive neighborhood group. Price increases with the floor number as there are only a few outliers with a higher price below floor ~15.

To further examine the relationships obtained through the previous graphs, I decided to subset the dataset based on neighborhood groups. Below are the graphs that show the relationship between price and other attributes for Manhattan (the rightmost one is for Brooklyn):







From these plots, we can safely conclude that Entire home/apt in Manhattan are some of the most expensive AirBnbs and higher floors (assumingly for those apartments) are the most expensive ones.

Prediction Model

Just like my last model, I started this one by dividing the given dataset into training (80%) and testing (20%) sets as well, and using Random Forest with all the attributes to get a basic idea of the relationship of price with different attributes.

```
# splitting the dataset into training and testing.
idx <- sample( 1:2, size = nrow(airbnb_data), replace = TRUE, prob = c(.8, .2))
train <- airbnb_data[idx == 1,]
test <- airbnb_data[idx == 2,]
```

From the plots, I used the most important attributes which were:

Neighborhood group + Room Type + Floor

Upon using this combination of attributes on various ML algorithms including Random Forest, Rpart and SVM, I found that SVM worked the best and resulted in the least mse of 3185.992 for the training portion, and 3169.211 for the testing portion of the data. I thus decided to use that as my final model

```
> svm_fit <- svm(price ~ neighbourhood_group + room_type + floor,
+                data = airbnb_data_playoffs)
>
> pred_train_svm <- predict(svm_fit,newdata = train)
> mse(train$price,pred_train_svm)
[1] 3185.992
> pred_test_svm <- predict(svm_fit,newdata = test)
> mse(test$price,pred_test_svm)
[1] 3169.211
```

Incorporation of POIs

Similar to my previous model, I introduced POIs in my dataset to see if they help improve my mses at all.

```
#SVM
svm_fit2 <- svm(price ~ neighbourhood_group + room_type + floor +
                  proximitytomsg + proximitytojfk + proximitytostatue + proximitytoemps + proximitytocpl
                + proximitytomemorial,
                data = train2)

pred_train_svm <- predict(svm_fit2,newdata = train2)
mse(train2$price,pred_train_svm)
pred_test_svm <- predict(svm_fit2,newdata = test2)
mse(test2$price,pred_test_svm)
```

The results, however, were not significantly different thus, I decided to stick to my original SVM model for my final submission.

My final score on Kaggle upon submission of this model was 3022.80756

Conclusion

This was a very fun project! I loved going through all the stages, working on my dataset and learning something new about it at each checkpoint. Although I tried my best throughout, some of the things, as mentioned by professor, that I could have improved on were:

- Making my prediction model with the human in the loop and build a model which was a combination of several models for different data subsets.
- I don't believe my implementation of the incorporation of POIs is the most effective one. I tried improving it using various methods including the use of geosphere package and making separate lists and trying to compare them, but my R Session kept aborting. I could've tried even more ways to best incorporate POIs into my dataset.