Playoffs Prediction Challenge

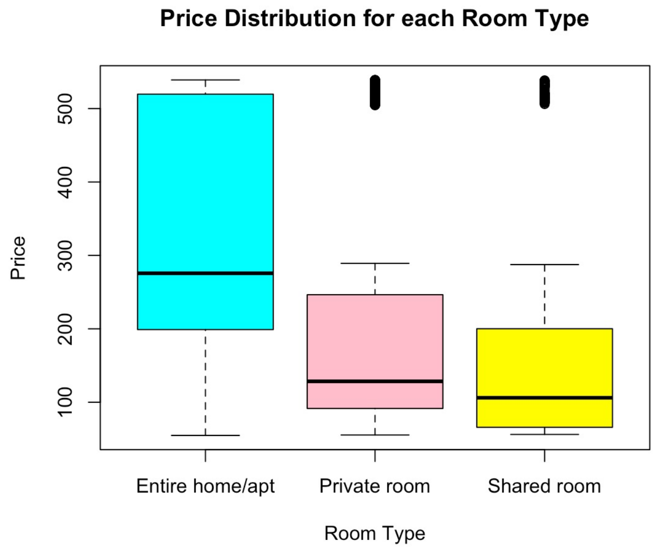
Presentation by:

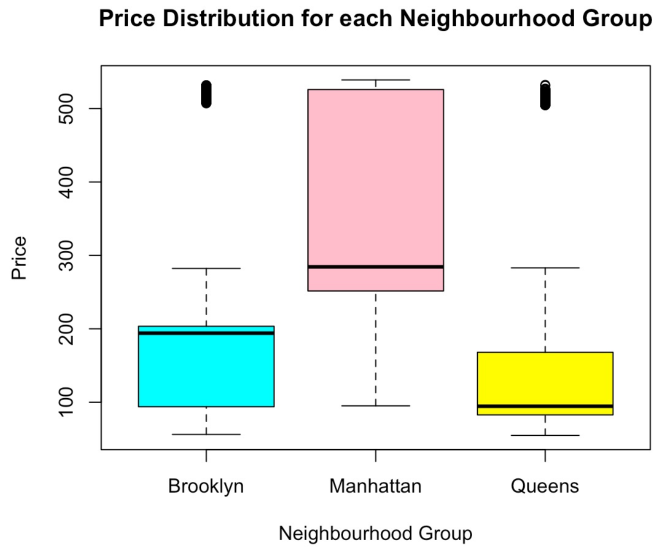
Muskan Burman

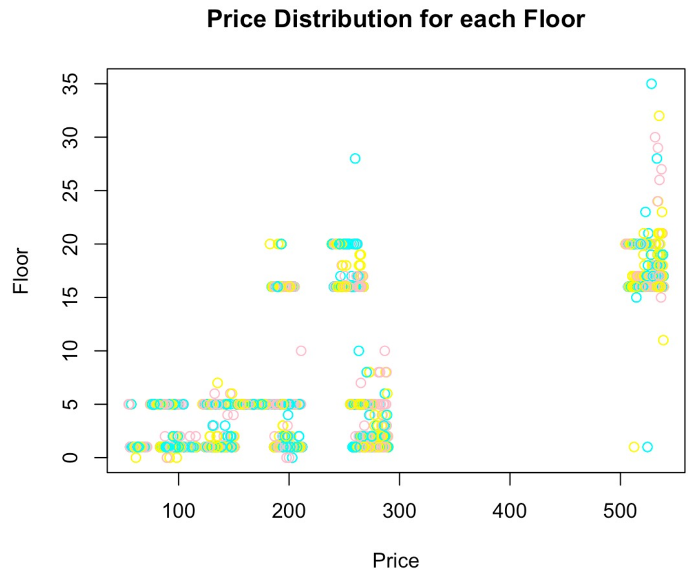
Data Science Capstone Project

Plots

- To observe the patterns within the dataset, I started by plotting the relationships between various attributes, and the following plots stood out to me the most.



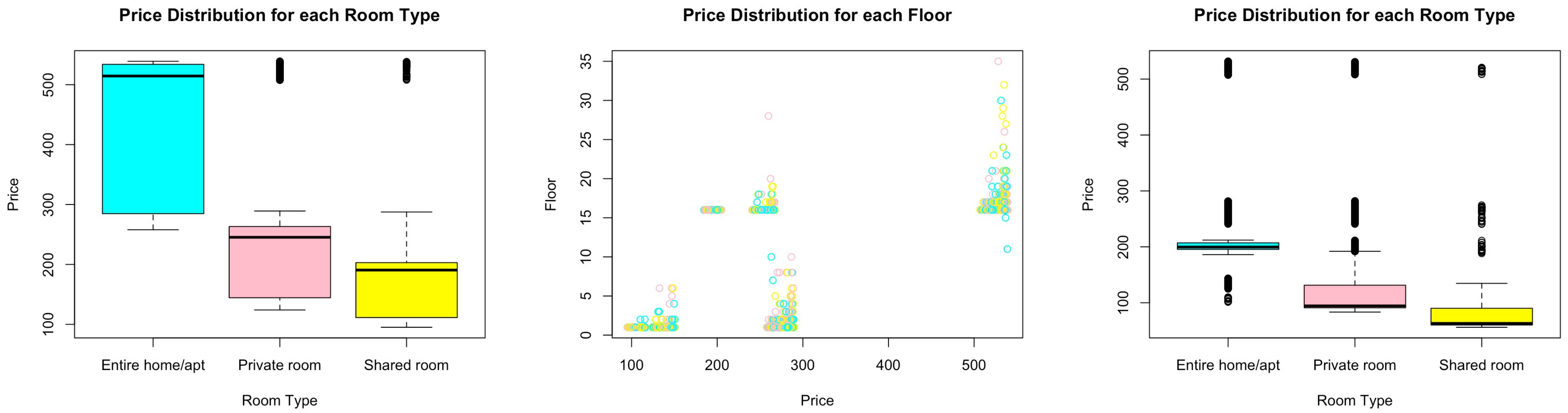




- Looking at these plots, we can see that Entire home/apt is the most expensive room type and Manhattan is the most expensive neighborhood group.
- Price increases with the floor number as there are only a few outliers with a higher price below floor ~15.

Plots

- 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.

Developing a Prediction Model

- I started by randomly dividing the given dataset into training and testing data – around 80% for training and 20% for testing.
- I started with Random Forest, and initially included all the attributes, just 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_playoffs), replace = TRUE, prob = c(.8, .2))
train <- airbnb_data_playoffs[idx == 1,]
test <- airbnb_data_playoffs[idx == 2,]
```

- From the plots, the most important attributes were:
  - Neighborhood group
  - Room Type
  - Floor
- And so I decided to use these for my model.

Developing a Prediction Model

- 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
```

Final Prediction model using SVM

```
> random_forest_play <- randomForest(price ~ neighbourhood_group + room_type + floor,
+                                      data = train)
>
> pred_train_play <- predict(random_forest_play,newdata = train)
> mse(train$price,pred_train_play)
[1] 3791.47
> pred_test_play <- predict(random_forest_play,newdata = test)
> mse(test$price,pred_test_play)
[1] 3723.566
```

Example using RF

Incorporation of POIs

- To try improve my model further, I tried using some feature engineering by introducing some POIs in the data and their relation to the price.
- I looked at Airbnbs near the 9/11 memorial, Central Park, Empire State Building, Statue of Liberty, Madison Square Garden, and the JFK Airport, and developed plots to find the relationship between different attributes.

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: After trying multiple different methods 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
```

Incorporation of POIs

Result: Using the above implementation, I added all new 6 columns to my SVM model, however the results were not significantly different.

```
#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)
```

I thus decided to stick to my original SVM model for my Kaggle submission and got a final score of 3022.80756 on Kaggle.