Predicting Price Range of New York City Airbnb listings using KNN Classification

Anushka Nair
10/19/2020

Introduction and Description of classification task:

Airbnb is a service that has completely disrupted the travel & hospitality markets, and this disruption has clearly been welcomed by many travelers, as Airbnb has sustained massive growth since their inception in 2009. Any Airbnb insights that could be gained from analyzing user data may translate into a more efficient travel market overall. This could mean a better experience for both travelers and hosts, as well as a deeper understanding of hospitality services in today's world.

I obtained a large, New York City (NYC) Airbnb listing dataset from kaggle.com, a crowd-sourced, data-sharing platform. This dataset consisted of 16 variables with 48,895 listings, which I ultimately reduced to 10 variables with 38843 listings. This paper presents the considerations, processes, and conclusions derived from one project goal; to predict the price range of an Airbnb listing based on different features. A description of the dataset is provided below.

Data Description:
Number of observations: 38,843
Number of predictors:9
Number of predictors after one-hot encoding: 15

Categorical:

1) Neighbourhood_group: This predictor represents which New York city borough the listing is from. It has 5 categories, namely Bronx, Brooklyn, Manhattan, Queens, and Staten Island.
2) Room_type: The room type of the listing, it can either be Entire Home, Shared Room, or a Private Room.

Numerical:

1) Latitude :The latitude coordinates of the listing
2) Longitude : The longitude coordinates of the listing
3) Minimum_nights: The minimum amount of nights required to book this listing.
4) Number_of_reviews: The number of reviews
5) Reviews_per_month: The number of reviews per month
6) Calculated_host_listings_count: The number of listings per host.
7) Availability_365: The number of days in a year when the listing is available for booking
8) Price: Price in Dollars

My target variable is Price. The price variable in the dataset was in dollars, so I created 3 categories in order to be able to build a classification model. The classes and the number of observations per class are listed below.

Target
Low($0-$79) : 12,804 cases
Medium($80-$149): 13,319 cases
High (>=$150) : 12,720 cases

Cleaning the Dataset:

I deleted certain features because they did not provide any useful information or were hard to integrate into my model. I deleted the *ID*, *Name*, *Host ID*, and *Host Name* because they do not help in the classification task.
I deleted *Neighborhood* because it consistently threw a missing value error. The error stated that missing values are not allowed in the knn model. I created a function to check if any variables consisted of missing values, and I found 3 of the neighborhoods did not have any values associated with it. This was because I used na.omit in the original dataset, due to which 3 neighborhood groups that were only associated with one observation each were deleted. This led to the one-hot encoded variable neighborhood to retain those three neighborhood groups but associated with no observations. I got rid of the three variables from the predictor dataset, and confirmed that there were no NA values, however the knn still did not run. The data no longer had a missing value error, but a new error that said the training set and class values were not of the same length. I checked the number of rows in each, and they both had 31,074 values. I tried changing it to a data frame and re-running, but the error persisted. Ultimately, I decided to drop the variable neighborhood.
Lastly, I deleted the date of the *last review*, because it was too complex to work with.
I also deleted observations with missing data, thus reducing the number of observations from 48,895 to 38,843.

Balancing the size of Classes:

The target class Price had the following classes: Class low consisting of prices <$79, Class Medium consisting of prices <$150 and Class High consisting of Prices >=$150

I chose these specific ranges based on the following:

1. New York hotels range from $74 to $506 per night, with an average cost of $196. The first price range could be thought of as competitive listings that beat the average hotel cost, since most of the values fall below the lower end of an average New York hotel. The medium price range was chosen to be roughly consistent with the first price range(spanning approximately $70-$80 dollars). The high price ranges from $150- $10,000, which is a huge range, however approximately 10,000 observations out of the 12,720 cases that fall under the category High are in the price range $150-$300.
2. The exact cutoff values were picked based on which values provided the best balance between classes. I tried several different values for each cutoff and ultimately decided on the ones stated above.

Categorical to Numerical:

The two categorical variables, neighborhood_group and room_type, were dealt with by using the process of one-hot encoding. One hot encoding transforms a given categorical variable with *n* values into *n* different features where the values can either 1 or 0, 1 indicating the presence of the corresponding category and 0 the absence.

- The feature neighborhood_group consisted of 5 categories: Bronx, Brooklyn, Manhattan, Queens, and Staten Island. The 5 features were one hot encoded and took on the following ID's :
  neighbourhood_group_Bronx, neighbourhood_group_Brooklyn, neighbourhood_group_Manhattan, neighbourhood_group_Queens and neighbourhood_group_Staten Island.

- The feature Room_type consisted of 3 categories: Entire home/apt, Private room and Shared Room. The 3 features were one hot encoded and took on the following ID's :
  room_type_Entire home/apt, room_type_Private room, room_type_Shared room

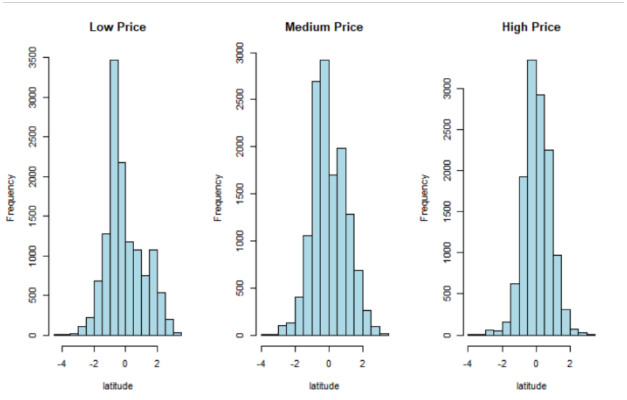
Standardizing each feature:

I standardized all the features, so each feature has a center of 0 and standard deviation of 1 using an inbuilt R function. Normalizing the data is an important step, because it reduces any bias introduced due to variables being measured at different scales.

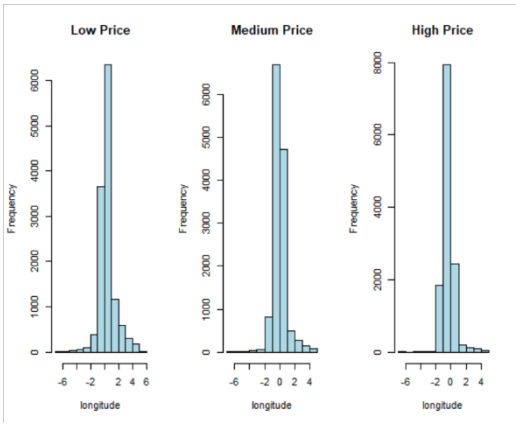
Evaluating Discriminating power: Comparing Features Visually & using the t-test:

In order to compare the features visually, I divided the data into three subsets corresponding to the three classes: LOWPRICE, MEDPRICE and HIGHPRICE.

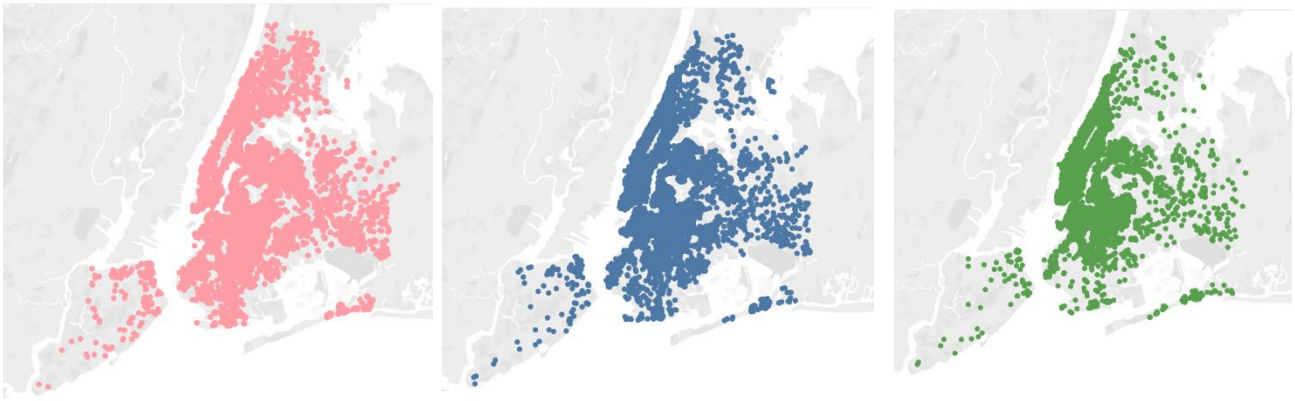
Below are histograms of a few features:



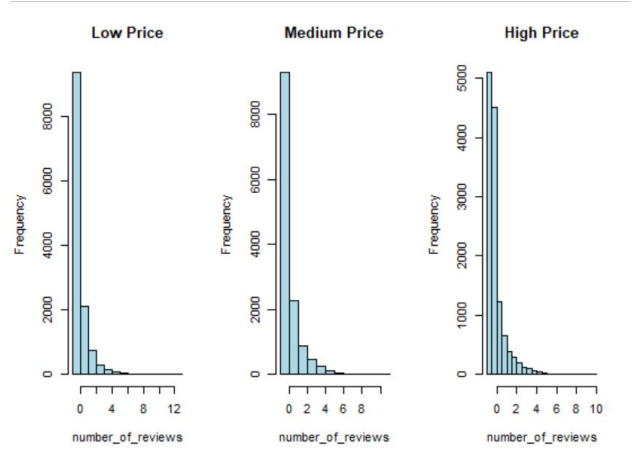
Latitude: The three histograms seem to follow a generally normal distribution. Price Low seems to be peak around x= -1 and then immediately drops. Medium price also peaks around x=-1, but it doesn't have a sharp drop in frequency like price low, instead it has a gradual drop in frequency. Price medium also looks bimodal, with a smaller second peak at around x=1. Price High has a peak x=0 and does not drop as sharply as the histogram for low price. The variable latitude seems to have high discriminating power. This is confirmed by the t.test I conducted for the three pairs of classes, which indicate that the means are significantly different for each of the three classes.



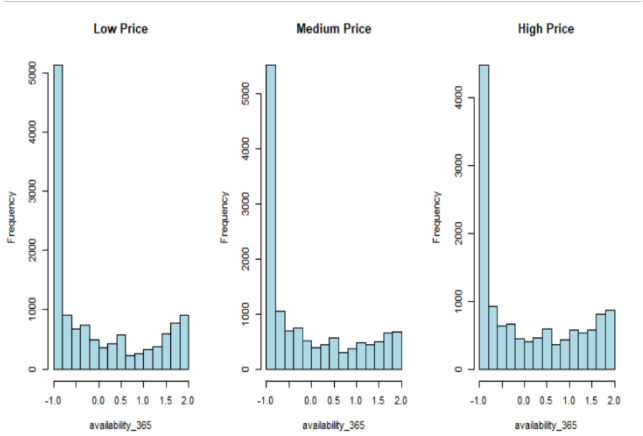
Longitude: The three histograms seem approximately normal. All three histograms peak at x=0. The histograms do look slightly different, suggesting high discriminating power. The t-test suggested that the three means are significantly different.



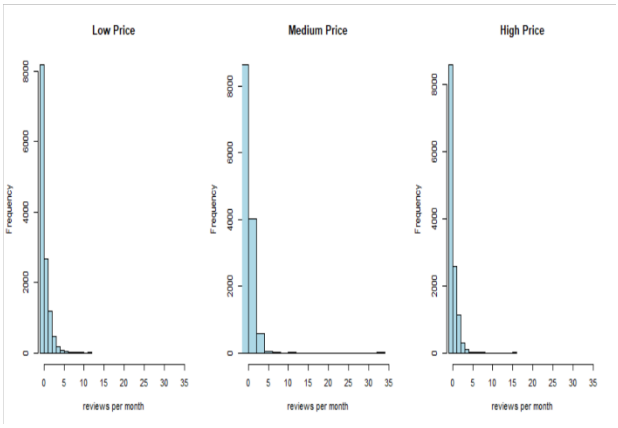
Latitude and Longitude: The two predictors latitude and longitude paired together tell us exactly where the location of the listing is. Thus, I used tableau, a data visualizing tool, to plot the latitude and longitude versus the target price Low, Medium and High. Plotting the values on a map makes it easier to visualize any trends.
Pink represents Low Price; Blue represents Medium Price; Green represents High Price. The latitude and longitude variable seem to be useful in predicting price, because the three plots above seem to indicate that certain locations have more/less listings across the three classes.



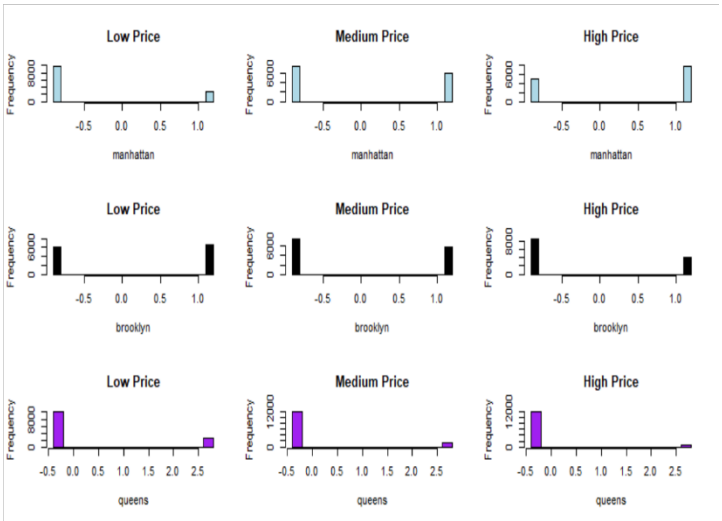
Number_of_reviews: Visually, number of reviews of low and medium price don't seem that different, however high price has a different pattern compared to low and medium. I conducted a t-test to compare the mean number of reviews between pairs of the three classes, and all three pairs had means that were significantly different.
The variable number of reviews seems to have some discriminating power.

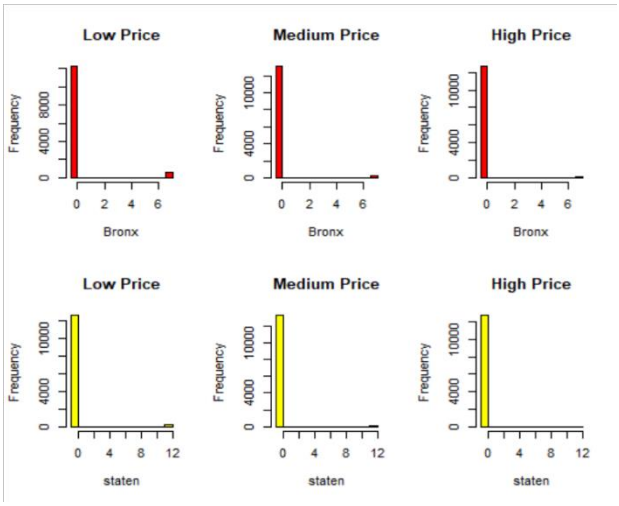


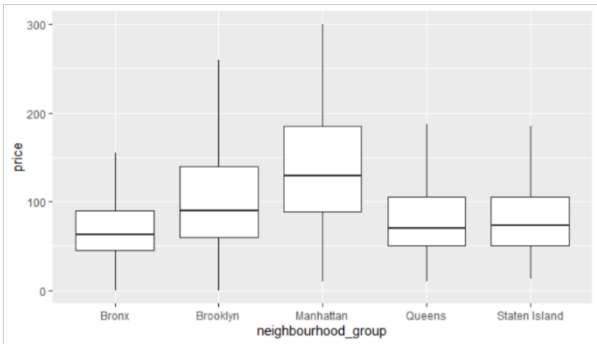
Availability_365: All three histograms peak at the lower end of the plot and seem to have some underlying quadratic pattern. I conducted a t-test for the three pairs of features, and all three pairs of features had significantly different means. Availability 365 seems to have discriminating power.



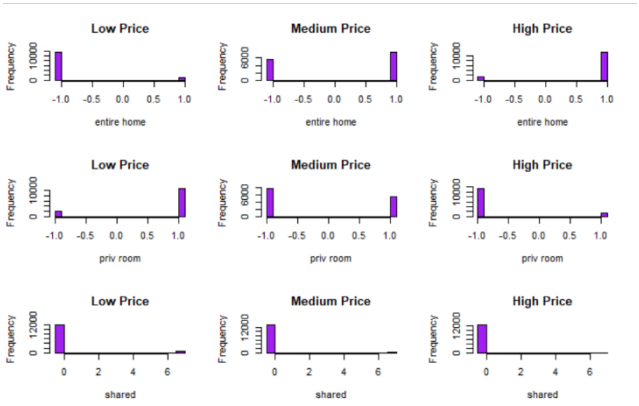
Reviews per month: The histograms for Low price, Medium Price and High price look almost identical. Conducting the t-test suggested that all three pairs of classes had significantly different means. Visually, reviews per month does not seem to have strong discriminating power.

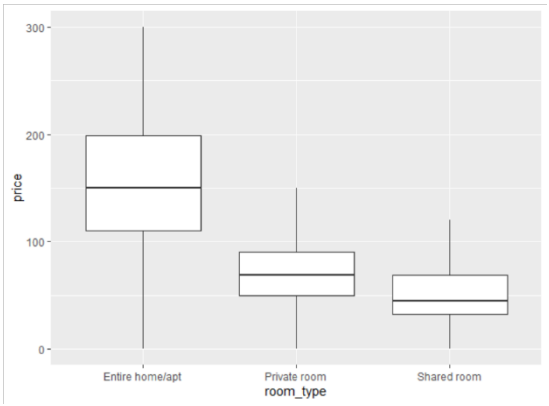






Neighborhood_Location: The location does seem to have discriminating power. In the histogram, the lower end of the x axis indicates absence, and the higher end indicates presence. Manhattan seems to have more high-priced listings than lower. Brooklyn generally looks the same across the three classes, except there seem to be fewer Brooklyn listings when the Price is high. Queens listings seem to decrease when the price goes up. Bronx and Staten Island don't have much of a pattern in the three classes, except that there do not seem to be that many listings in all three classes. Overall, location seems like a good feature in terms of discriminating power. Since this feature is categorical and the histograms are not as easy to read, I also plotted the box plot to get a better look at the 5 neighborhood locations versus the original numerical price variable. The boxplot indicates that this feature has discriminating power. Listings in Manhattan and Brooklyn are, on average, higher priced than the other three neighborhoods. The t test indicates that all the means are significantly different across classes.





Room Type: The histogram for entire home and private room seem to have a clear pattern across the classes. 0 indicates absence and 1 indicates presence, so in the case of entire home, as price goes up, there are more listings and private room follows the opposite trend. This makes sense practically because we would expect an entire house listing to be priced higher than a single room. The shared room does not have any pattern across classes, and in fact it seems as though there aren't many shared room listings. The box plot also confirms the inference, with entire home priced higher than private room, and private room priced higher than shared room. The t test indicates the three means are significantly different.

Trainset and test-set distribution:

- Prior to plotting the histograms, I split the dataset into three subsets corresponding to the target classes: LOWPRICE, MEDPRICE, HIGHPRICE.
- I randomly split each subset into a training and testing set, with each train set consisting of 80% of the corresponding subset and each test set consisting of 20% of the corresponding subset.
- I combined the three train sets into one and the three test sets into one, naming them TRAIN_SET and TEST_SET respectively.

Principles of KNN algorithm:

- The KNN classifier identifies K points in the training data that are closest to x_0 (where x_0 denotes the predictor values (x_1, …., x_p)
- It then estimates the conditional probability $P(Y=j| X =x_0)$ where j = 1,2,.. C (where C=# of classes) i.e the probability that Y=j given the observed predictor vector x_0.
- Finally, it classifies the observation to the class with highest estimated probability. C(x0) = j , if $\hat{p}j(x0) = \max\{\hat{p}1(x0), ..., \hat{p}J(x0)\}$.

KNN-Classification for multiple values:

- I ran the knn for multiple values of K, namely K= 5,10,15,20,30,40,60,80,100. Prior to running the knn, I set the seed in order to ensure reproducible results. The training accuracy for low K values were extremely high, which is to be expected due to bias. The training set accuracy ranged from approximately 69%-76% while the testing set accuracy ranged from 66%-69%. Looking at the accuracies, I decided to rerun the knn with values K=5,15,20-40,60,80,100 because the best K seemed to be in the range of 20 to 40.

- Figure 1 on the right displays the train accuracy and test accuracy values for k= 5,10,15-30,50,80,100. The blue line is the training set accuracy, while the red line is the testing set accuracy. The plot indicates some level of overfitting, since our train accuracy is consistently 2-3% more accurate than our test set. Ultimately, the best k from the plot seems to be k=21.

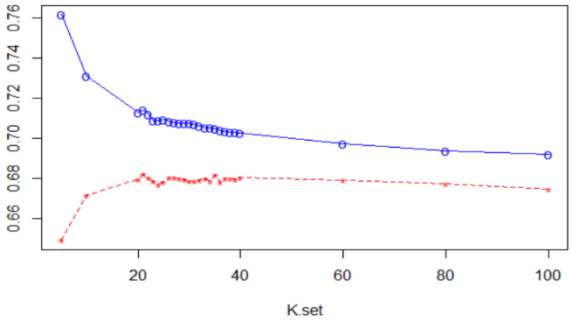


***Figure 1**: percent accuracy for train vs test set*

KNN-Classification for best K:

- After carefully looking at the plot, I choose k=21 to be the best k value since it has the highest testing set accuracy.
- The training and testing accuracy for K= 21 are as follows:
  - $TrainAccuracy^{21} = 0.7088$
  - $TestAccuracy^{21} = 0.6912$
- I also ran a 95% confidence interval for the train and test set, with the following intervals. The intervals indicate that the training accuracy will almost always be greater than the testing accuracy since there is no overlap.
  - We can be 95% confident that the training accuracy for k=21 lies between the interval (0.7049, 0.7150)
  - We can be 95% confident that the test set accuracy for k=21 lies between the i nterval (0.6749, 0.6955)

- Below are the 3x3 confusion matrices for the train-set and test-set :

TRAIN SET CONFUSION MATRIX

TRAIN-SETK=21		ACTUAL CLASS		
		LOW	MEDIUM	HIGH
KNN	LOW	8550	2190	341
PREDICTION	MEDIUM	1494	5900	2108
	HIGH	199	2565	7727

TRAIN-SETK=21		ACTUAL CLASS		
		LOW	MEDIUM	HIGH
KNN	LOW	83.47%	20.55%	3.35%
PREDICTION	MEDIUM	14.59%	55.37%	20.72%
	HIGH	1.94%	24.07%	75.93%

TEST SET CONFUSION MATRIX

TEST-SETK=21		ACTUAL CLASS		
		LOW	MEDIUM	HIGH
KNN	LOW	2084	613	81
PREDICTION	MEDIUM	427	1324	631
	HIGH	50	727	1832

TEST-SETK=21		ACTUAL CLASS		
		LOW	MEDIUM	HIGH
KNN	LOW	81.37%	23.01%	3.18%
PREDICTION	MEDIUM	16.67%	49.7%	24.8%
	HIGH	1.95%	27.29%	72.01%

- Global Train-set accuracy=71.59%
- Global Test-set accuracy=67.69%
- We lose predicting power in all three classes when we move from the training set to the testing set.
- Looking at the confusion matrix, we can see that the knn performs poorly in classifying the class Medium in the train and test set, with an accuracy of 55.37% and 49.7% respectively. The model does a fairly good job in classifying Low and High class in both, the train and test set. Taking this into consideration, after conducting the weighted knn, I will also build a knn model dropping the class Medium in order to improve accuracy(See last section titled **Reducing to a 2 class problem)** .

Weighted KNN:

- In an attempt to improve accuracy of my knn model, I performed a weighted KNN. The non-weighted set assumes that all features are equally important when they are not. Hence, we weigh the features to account for the importance of different features.
- The first step in performing a weighted knn is to obtain the weights to apply to each feature.
- I tried applying knn using each individual feature/packs of categorical features, but the knn would not run due to many observations being equidistant, so I used the ks.test to obtain the weights.
- The KS test, which stands for the Kolmogorov-Smirnov Goodness of Fit Test compares two datasets and tells you if the datasets come from the same distribution.
- I conducted a ks-test for each of the 15 features across the three classes. I compared Low versus Medium, Low versus High, and Medium Versus High for each feature, obtaining 3 different p values, namely PV12,PV13 and PV23. Next, to calculate the weight, I used the formula:
  $\text{Weight} = \frac{(1-\text{PV12})+(1-\text{PV13})+(1-\text{PV23})}{3}$
- If the ks-test determines significance for a feature across all three classes, the above formula will have a weight of 100% .
- After conducting the 15 ks-tests, I looked at the 15 corresponding weights. 11 of my features had weight=1, indicating that those predictors are important for the knn classification. The remaining 4 weights are as follows:
  - Weight1 = 0.916, Bronx
  - Weight5 = 0.213, Staten Island
  - Weight10 = 0.6671, Room type: Shared
  - Weight13 = 0.9974, Reviews per month
  - When I applied the t-test earlier, these predictors were deemed significant. The ks test however confirmed my analysis. When I analyzed the histograms for these particular features, I pointed out that these variables did not seem to have any trend across the classes.
  - The four predictors above will be considered less important in the weighted knn model.
- Next, I multiplied each weight to its corresponding predictor using the standardized dataset, after which I renormalized the data
- I followed the same steps listed in the section "Trainset and test-set distribution" to obtain my train and test dataset, using the weighted standardized dataset.
- I applied knn with K=21 and obtained the following train and test-set results:

WEIGHTED KNN-TRAIN SET

TRAIN-SETK=21		ACTUAL CLASS		
WEIGHTED		LOW	MEDIUM	HIGH
KNN	LOW	8537	2156	335
PREDICTION	MEDIUM	1499	5904	2243
	HIGH	207	2595	7598

TRAIN-SETK=21		ACTUAL CLASS		
WEIGHTED		LOW	MEDIUM	HIGH
KNN	LOW	83.34%	20.23%	3.29%
PREDICTION	MEDIUM	14.63%	55.41%	22.04%
	HIGH	2.02%	24.35%	74.67%

WEIGHTED KNN TEST SET

TEST-SETK=21		ACTUAL CLASS		
WEIGHTED		LOW	MEDIUM	HIGH
KNN	LOW	2140	646	74
PREDICTION	MEDIUM	375	1297	592
	HIGH	46	721	1878

TEST-SETK=21		ACTUAL CLASS		
WEIGHTED		LOW	MEDIUM	HIGH
KNN	LOW	83.56%	24.25%	2.91%
PREDICTION	MEDIUM	14.64%	48.69%	23.27%
	HIGH	1.8%	27.06%	73.82%

- Weighted KNN Global Train Set Accuracy = 71.14%
- Weighted KNN Global Test set Accuracy = 68.69%
- As we move from the train set to the test set, the accuracy reduces by approximately 3%. The test set accuracy for classifying low increases slightly, but decreases for Medium and High.

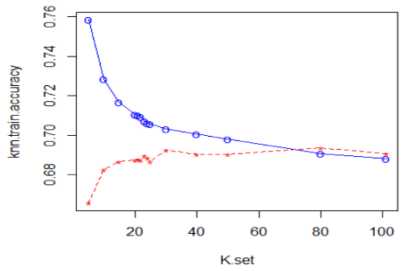
Comparing Ordinary KNN and Weighted KNN performance in the testing set:

TEST-SETK=21		ACTUAL CLASS		
		LOW	MEDIUM	HIGH
KNN	LOW	81.37%	23.01%	3.18%
PREDICTION	MEDIUM	16.67%	49.7%	24.8%
	HIGH	1.95%	27.29%	72.01%

TEST-SETK=21		ACTUAL CLASS		
WEIGHTED		LOW	MEDIUM	HIGH
KNN	LOW	83.56%	24.25%	2.91%
PREDICTION	MEDIUM	14.64%	48.69%	23.27%
	HIGH	1.8%	27.06%	73.82%

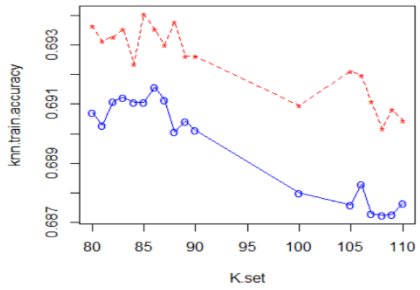
- Ordinary KNN global test-set accuracy = 67.69%
- Weighted KNN global test-set accuracy = 68.69%
- The accuracy went up exactly by 1%. The weighted KNN improved in classifying Low and High, but got worse in classifying Medium.
- Overall, conducting the weighted KNN did not seem to significantly improve the knn.
    - The 95% confidence interval for the weighted KNN testing set performance is (0.6803, 0.7009).
    - The 95% confidence interval for the ordinary KNN testing set performance that we calculated earlier was (0.6749, 0.6955).
    - The weighted knn confidence interval does improve by 1%, however there is overlap between the two intervals suggesting that the two knn's perform similarly.
- Next, I also applied KNN to multiple K values, with K= 5,10,15,20:25,30,40,50,80,100.

Below is a plot with the train versus test accuracies for the weighted KNN.



Weighted KNN 5<K<100

The blue line represents the training set accuracy, and the red line is the testing set accuracy. Applying the weights to the knn model seemed to make the testing set accuracy greater than the training set accuracy for K values>80 so I ran the knn model for values k=80:90,100,105:110.



Weighted KNN 80<K<110

The plot on the left are the train and test set accuracies for the weighted KNN. The red plot is the testing accuracy and the blue plot is the training accuracy. It seems that once k crosses 80, the testing set consistently does better than the training set. The testing accuracy for 80<k<110 is approximately 69% for all these values, which is just 1% more than the best K I used earlier i.e K=21.

Reducing to a 2 class problem:

- This step is not part of the midterm requirements, however I conducted this before the step 4 instructions, and the results were interesting, so I decided to leave this in my report.
- When I conducted the ordinary KNN, I noticed that the knn algorithm did a fairly decent job in classifying Low Class and High Class, with testing set accuracies >75%, but performed poorly in classifying Medium with a testing set accuracy of 49.7%. This suggested that the knn algorithm was confusing some of the low and high price listings with the medium listings.
- I followed the same steps as earlier to obtain the KNN results. I deleted the class Medium, and built a train-set and test-set consisting of just Low and High, using the same 80-20 split as before. Below are the results for the 2 class KNN for predicting Low versus High:

Train-SETK=21		ACTUAL CLASS	
		LOW	HIGH
KNN	LOW	92.35%	6.57%
PREDICTION	HIGH	7.65%	93.43%

TEST-SETK=21		ACTUAL CLASS	
		LOW	HIGH
KNN	LOW	91.21%	7.43%
PREDICTION	HIGH	8.79%	92.57%

- Train set global accuracy = 92.89%
- Test set global accuracy = 91.89%
- The 2 class KNN test set accuracy increased by approximately 24% compared to the ordinary 3 class KNN and the weighted 3 class KNN.
- An increase in test set accuracy is expected when reducing classes, since the knn algorithm now has a 50% chance of accidentally predicting the right class. However, to confirm that this dramatic increase wasn't just because of luck, I also ran a KNN to classify Low versus Medium and Medium versus High. The confusion Matrices are below:

Train-SETK=21		ACTUAL CLASS	
		LOW	MEDIUM
KNN	LOW	83.57%	20.96%
PREDICTION	MEDIUM	16.43%	79.04%

TEST-SETK=21		ACTUAL CLASS	
		LOW	MEDIUM
KNN	LOW	81.45%	21.06%
PREDICTION	MEDIUM	18.55%	78.94%

- Low versus Medium: Train Set global accuracy = 81.31%
- Low versus Medium: Test Set global accuracy = 80.19%

Train-SETK=21		ACTUAL CLASS	
		MEDIUM	HIGH
KNN	MEDIUM	75.72%	25.24%
PREDICTION	HIGH	24.28%	74.76%

Test-SETK=21		ACTUAL CLASS	
		MEDIUM	HIGH
KNN	MEDIUM	72.71%	26.42%
PREDICTION	HIGH	27.29%	73.58%

- Medium versus High: Train set global accuracy = 75.24%
- Medium versus High: Test set global accuracy = 73.15%

The test set accuracies for Low versus Medium and Medium versus High are also better than their performance in the ordinary and weighted KNN, but the performance of Low versus High is improved dramatically. This suggests that there may be some Medium-priced listings that are similar to Low and High, which is understandable. My hypothesis is that these similar listings may be at the extreme ends of class Medium, which means they could be considered either Low or High. I tried changing the cut-off values for the three classes, but the class sizes changed dramatically, but exploring that option further might be the solution to a better KNN model.

R code

```
######GET RID OF PRICE COLUMN

NYC_DATA= na.omit(AB_NYC_2019)##No NA values
NYC_DATA$neighbourhood<-as.factor(NYC_DATA$neighbourhood)
NYC_DATA_CLEAN= NYC_DATA[,-c(1:4,6,13)]
View(NYC_DATA_CLEAN)
#adding the target variable
NYC_TARGET<-NYC_DATA_CLEAN
NYC_TARGET$target = NA

#Creating the target class
for (i in 1:nrow(NYC_TARGET)){
  if(NYC_TARGET$price[i]<80){
    NYC_TARGET$target[i]="Low"
  }else if(NYC_TARGET$price[i]<150){
    NYC_TARGET$target[i]="Medium"
  }else{
    NYC_TARGET$target[i]="High"
  }
}
NYC_TARGET$target=as.factor(NYC_TARGET$target)

summary(NYC_TARGET$target)

##Data visualization
library(ggplot2)
ggplot(data = NYC_DATA_CLEAN,aes(x=room_type,y=price))+geom_boxplot(outlier.shape=NA)+ylim(0,300)
ggplot(data = NYC_DATA_CLEAN,aes(x=neighbourhood_group,y=price))+geom_boxplot(outlier.shape=NA)+ylim(0,300)

#one hot encoding categorical
library(data.table)
install.packages("mltools")
library(mltools)
S_NYC<-one_hot(as.data.table(NYC_TARGET), cols = c("room_type","neighbourhood_group"))

##Standardizing

library(standardize)
library("dplyr")
S_NYC1<-S_NYC%>%mutate_if(is.numeric, function (x) as.vector(scale(x)))
S_NYC1<-S_NYC1[,-11]#get rid of original price var

#split into LOW, MED AND HIGH

set.seed(1)
LOWPRICE = S_NYC1[which(S_NYC1$target =="Low"),]

set.seed(1)
MEDPRICE = S_NYC1[which(S_NYC1$target =="Medium"),]
```

```
set.seed(1)
HIGHPRICE = S_NYC1[which(S_NYC1$target =="High"),]

###Compare histograms visually and conduct t-test

par(mfrow = c(3,3))
hist(LOWPRICE$availability_365, main = "Low Price" , col = "lightblue", xlab = "availability_365")
hist(MEDPRICE$availability_365, main = "Medium Price" , col = "lightblue", xlab = "availability_365")
hist(HIGHPRICE$availability_365, main = "High Price" , col = "lightblue", xlab = "availability_365")

t.test(LOWPRICE$availability_365,MEDPRICE$availability_365)
t.test(LOWPRICE$availability_365,HIGHPRICE$availability_365)
t.test(MEDPRICE$availability_365,HIGHPRICE$availability_365)

par(mfrow = c(1,3))
hist(LOWPRICE$reviews_per_month, main = "Low Price" , col = "lightblue", xlab = "reviews per month", xlim = c(0,35))
hist(MEDPRICE$reviews_per_month, main = "Medium Price" , col = "lightblue", xlab = "reviews per month", xlim = c(0,35))
hist(HIGHPRICE$reviews_per_month, main = "High Price" , col = "lightblue", xlab = "reviews per month",xlim =c(0,35))

t.test(LOWPRICE$reviews_per_month,MEDPRICE$reviews_per_month)
t.test(LOWPRICE$reviews_per_month,HIGHPRICE$reviews_per_month)
t.test(MEDPRICE$reviews_per_month,HIGHPRICE$reviews_per_month)

par(mfrow = c(1,3))
hist(LOWPRICE$number_of_reviews, main = "Low Price" , col = "lightblue", xlab = "number_of_reviews")
hist(MEDPRICE$number_of_reviews, main = "Medium Price" , col = "lightblue", xlab = "number_of_reviews")
hist(HIGHPRICE$number_of_reviews, main = "High Price" , col = "lightblue", xlab = "number_of_reviews")

t.test(LOWPRICE$number_of_reviews,MEDPRICE$number_of_reviews)
t.test(LOWPRICE$number_of_reviews,HIGHPRICE$number_of_reviews)
t.test(MEDPRICE$number_of_reviews,HIGHPRICE$number_of_reviews)



par(mfrow = c(3,5))
hist(LOWPRICE$minimum_nights, main = "Low Price" , col = "lightblue", xlab = "min nights")
hist(MEDPRICE$minimum_nights, main = "Medium Price" , col = "lightblue", xlab = "min nights")
hist(HIGHPRICE$minimum_nights, main = "High Price" , col = "lightblue", xlab = "min nights")

par(mfrow = c(3,3))
hist(LOWPRICE$neighbourhood_group_Manhattan, main = "Low Price" , col = "lightblue", xlab = "manhattan", yim=c(0,8000))
hist(MEDPRICE$neighbourhood_group_Manhattan, main = "Medium Price" , col = "lightblue", xlab = "manhattan", yim=c(0,8000))
hist(HIGHPRICE$neighbourhood_group_Manhattan, main = "High Price" , col = "lightblue", xlab = "manhattan", yim=c(0,8000))

hist(LOWPRICE$neighbourhood_group_Bronx, main = "Low Price" , col = "red", xlab = "Bronx")
hist(MEDPRICE$neighbourhood_group_Bronx, main = "Medium Price" , col = "red", xlab = "Bronx")
hist(HIGHPRICE$neighbourhood_group_Bronx, main = "High Price" , col = "red", xlab = "Bronx")

t.test(LOWPRICE$neighbourhood_group_Bronx,MEDPRICE$neighbourhood_group_Bronx)
t.test(LOWPRICE$neighbourhood_group_Bronx,HIGHPRICE$neighbourhood_group_Bronx)
t.test(MEDPRICE$neighbourhood_group_Bronx,HIGHPRICE$neighbourhood_group_Bronx)

t.test(LOWPRICE$`neighbourhood_group_Staten Island`,MEDPRICE$`neighbourhood_group_Staten Island`)
t.test(LOWPRICE$`neighbourhood_group_Staten Island`,HIGHPRICE$`neighbourhood_group_Staten Island`)
t.test(MEDPRICE$`neighbourhood_group_Staten Island`,HIGHPRICE$`neighbourhood_group_Staten Island`)

t.test(LOWPRICE$neighbourhood_group_Queens,MEDPRICE$neighbourhood_group_Queens)
t.test(LOWPRICE$neighbourhood_group_Queens,HIGHPRICE$neighbourhood_group_Queens)
t.test(MEDPRICE$neighbourhood_group_Queens,HIGHPRICE$neighbourhood_group_Queens)

hist(LOWPRICE$neighbourhood_group_Brooklyn, main = "Low Price" , col = "black", xlab = "brooklyn", ylim = c(0,8000))
hist(MEDPRICE$neighbourhood_group_Brooklyn, main = "Medium Price" , col = "black", xlab = "brooklyn", yim=c(0,8000))
hist(HIGHPRICE$neighbourhood_group_Brooklyn, main = "High Price" , col = "black", xlab = "brooklyn", yim=c(0,8000))
```

```
hist(LOWPRICE$`neighbourhood_group_Staten Island`, main = "Low Price" , col = "yellow", xlab = "staten")
hist(MEDPRICE$`neighbourhood_group_Staten Island`, main = "Medium Price" , col = "yellow", xlab = "staten")
hist(HIGHPRICE$`neighbourhood_group_Staten Island`, main = "High Price" , col = "yellow", xlab = "staten")

hist(LOWPRICE$neighbourhood_group_Queens, main = "Low Price" , col = "purple", xlab = "queens", yim=c(0,10000))
hist(MEDPRICE$neighbourhood_group_Queens, main = "Medium Price" , col = "purple", xlab = "queens", yim=c(0,10000))
hist(HIGHPRICE$neighbourhood_group_Queens, main = "High Price" , col = "purple", xlab = "queens", yim=c(0,10000))

##Room type
par(mfrow = c(3,3))
hist(LOWPRICE$`room_type_Entire home/apt`, main = "Low Price" , col = "purple", xlab = "entire home")
hist(MEDPRICE$`room_type_Entire home/apt`, main = "Medium Price" , col = "purple", xlab = "entire home")
hist(HIGHPRICE$`room_type_Entire home/apt`, main = "High Price" , col = "purple", xlab = "entire home")

hist(LOWPRICE$`room_type_Private room`, main = "Low Price" , col = "purple", xlab = "priv room")
hist(MEDPRICE$`room_type_Private room`, main = "Medium Price" , col = "purple", xlab = "priv room")
hist(HIGHPRICE$`room_type_Private room`, main = "High Price" , col = "purple", xlab = "priv room")

hist(LOWPRICE$`room_type_Shared room`, main = "Low Price" , col = "purple", xlab = "shared")
hist(MEDPRICE$`room_type_Shared room`, main = "Medium Price" , col = "purple", xlab = "shared")
hist(HIGHPRICE$`room_type_Shared room`, main = "High Price" , col = "purple", xlab = "shared")

t.test(LOWPRICE$`room_type_Shared room`,MEDPRICE$`room_type_Shared room`)
t.test(LOWPRICE$`room_type_Shared room`,HIGHPRICE$`room_type_Shared room`)
t.test(MEDPRICE$`room_type_Shared room`,HIGHPRICE$`room_type_Shared room`)

t.test(LOWPRICE$`room_type_Entire home/apt`,MEDPRICE$`room_type_Entire home/apt`)
t.test(LOWPRICE$`room_type_Entire home/apt`,HIGHPRICE$`room_type_Entire home/apt`)
t.test(MEDPRICE$`room_type_Entire home/apt`,HIGHPRICE$`room_type_Entire home/apt`)
```

```
t.test(LOWPRICE$`room_type_Private room`,MEDPRICE$`room_type_Private room`)
t.test(LOWPRICE$`room_type_Private room`,HIGHPRICE$`room_type_Private room`)
t.test(MEDPRICE$`room_type_Private room`,HIGHPRICE$`room_type_Private room`)

par(mfrow = c(1,3))
hist(LOWPRICE$latitude, main = "Low Price" , col = "lightblue", xlab = "latitude")
hist(MEDPRICE$latitude, main = "Medium Price" , col = "lightblue", xlab = "latitude")
hist(HIGHPRICE$latitude, main = "High Price" , col = "lightblue", xlab = "latitude")

t.test(LOWPRICE$latitude,MEDPRICE$latitude)
t.test(LOWPRICE$latitude,HIGHPRICE$latitude)
t.test(MEDPRICE$latitude,HIGHPRICE$latitude)

par(mfrow = c(1,3))
hist(LOWPRICE$longitude, main = "Low Price" , col = "lightblue", xlab = "longitude")
hist(MEDPRICE$longitude, main = "Medium Price" , col = "lightblue", xlab = "longitude")
hist(HIGHPRICE$longitude, main = "High Price" , col = "lightblue", xlab = "longitude")

##TRAIN/TEST SET: LOW
n<-nrow(LOWPRICE)
train<-sample(1:n, 0.8*n)
trainset_LOW <- LOWPRICE[train,]
testset_LOW <- LOWPRICE[-train,]

##TRAIN/TEST SET: MEDIUM
n<-nrow(MEDPRICE)
train<-sample(1:n, 0.8*n)
trainset_MED <- MEDPRICE[train,]
testset_MED <- MEDPRICE[-train,]
```

```
n<-nrow(HIGHPRICE)
train<-sample(1:n, 0.8*n)
trainset_HIGH <- HIGHPRICE[train,]
testset_HIGH <- HIGHPRICE[-train,]
#combined train set
TRAIN_SET = rbind(trainset_LOW,trainset_MED,trainset_HIGH)
TEST_SET = rbind(testset_LOW,testset_MED,testset_HIGH)

library(class)
TRAIN_predictor <- TRAIN_SET[,-c(16)]
TRAIN_label <- TRAIN_SET[, "target"]
TEST_predictor <- TEST_SET[,-c(16)]
TEST_label <- TEST_SET[,"target"]

TRAIN_predictor=as.data.frame(TRAIN_predictor)
TEST_predictor=as.data.frame(TEST_predictor)
TRAIN_label=as.data.frame(TRAIN_label)
TEST_label=as.data.frame(TEST_label)

K.set = c(5,10,15:35,50,80,101)
knn.train.accuracy <- numeric(length(K.set))
traincl = TRAIN_label[,1, drop = TRUE]
#train set knn
for (j in 1:length(K.set)){
  set.seed(1)
  knn.pred <- knn(train=TRAIN_predictor,
                  test=TRAIN_predictor,
                  cl=traincl,
                  k=K.set[j])
  knn.train.accuracy[j] <- mean(knn.pred == traincl)
}
knn.train.accuracy
#test set knn
knn.test.accuracy <- numeric(length(K.set))
cl = TRAIN_label[,1, drop = TRUE]
testcl = TEST_label[,1, drop = TRUE]
for (j in 1:length(K.set)){
  set.seed(1)
  knn.pred <- knn(train= TRAIN_predictor,
                  test=TEST_predictor,
                  cl= cl,
                  k=K.set[j])
  knn.test.accuracy[j] <- mean(knn.pred == testcl)

}
knn.test.accuracy

#plot for multiple k vs train and test
plot(K.set, knn.train.accuracy, type="o", col="blue", pch="o", lty=1, ylim=range( c(knn.train.accuracy,knn.test.accuracy) ) )
points(K.set, knn.test.accuracy, col="red", pch="*")
lines(K.set, knn.test.accuracy, col="red",lty=2)

##knn using bestk
knn.predtrain <- knn(train= data.frame(TRAIN_SET[,-c(16)]),
                     test=TRAIN_SET[,-c(16)],
                     cl= traincl,
                     k=21)

knn.predtest <- knn(train= TRAIN_SET[,-16],
                    test=TEST_SET[,-16],
                    cl= traincl,
                    k=21)
```

```
#confusion matrix
conftrain<-table(data.frame(knn.predtrain,traincl))
conftest<-table(data.frame(knn.predtest,testcl))
library(DescTools)
#confidence intervals
Conf(conftrain)
Conf(conftest)
mean(knn.predtrain == traincl)
mean(knn.predtest == testcl)

#####STEP 4
#finding the weights for each feature using ks.test p value
#feature 1 neighbourhoodgroupbronx
ks.test(LOWPRICE$neighbourhood_group_Bronx,MEDPRICE$neighbourhood_group_Bronx)
#low vs medium PV12 value= 0.0001286
ks.test(LOWPRICE$neighbourhood_group_Bronx,HIGHPRICE$neighbourhood_group_Bronx)
#low vs high PV13, P VALUE = 0
ks.test(MEDPRICE$neighbourhood_group_Bronx,HIGHPRICE$neighbourhood_group_Bronx)
#medium vs high pv23 -> 0.2522
W1 = ((1-0.0001286)+(1-0)+(1-0.2522))/3

#feature 2 neighbourhoodgroupbrooklyn
ks.test(LOWPRICE$neighbourhood_group_Brooklyn,MEDPRICE$neighbourhood_group_Brooklyn)
#low vs medium PV12 value= 0
ks.test(LOWPRICE$neighbourhood_group_Brooklyn,HIGHPRICE$neighbourhood_group_Brooklyn)
#low vs high PV13, P VALUE = 0
ks.test(MEDPRICE$neighbourhood_group_Brooklyn,HIGHPRICE$neighbourhood_group_Brooklyn)
#medium vs high pv23 -> 0
W2 = ((1-0)+(1-0)+(1-0))/3
```

```
#feature 3 neighbourhoodgroupmanhattan
ks.test(LOWPRICE$neighbourhood_group_Manhattan,MEDPRICE$neighbourhood_group_Manhattan)
#low vs medium PV12 value= 0
ks.test(LOWPRICE$neighbourhood_group_Manhattan,HIGHPRICE$neighbourhood_group_Manhattan)
#low vs high PV13, P VALUE = 0
ks.test(MEDPRICE$neighbourhood_group_Manhattan,HIGHPRICE$neighbourhood_group_Manhattan)
#medium vs high pv23 -> 0
W3 = ((1-0)+(1-0)+(1-0))/3

#feature 4 neighbourhoodgroupqueens
ks.test(LOWPRICE$neighbourhood_group_Queens,MEDPRICE$neighbourhood_group_Queens)#low vs medium PV12 value= 0
ks.test(LOWPRICE$neighbourhood_group_Queens,HIGHPRICE$neighbourhood_group_Queens)#low vs high PV13, P VALUE = 0
ks.test(MEDPRICE$neighbourhood_group_Queens,HIGHPRICE$neighbourhood_group_Queens)#medium vs high pv23 -> 0
W4 = ((1-0)+(1-0)+(1-0))/3

#feature 5 neighbourhoodgroupstaten
ks.test(LOWPRICE$`neighbourhood_group_Staten Island`,MEDPRICE$`neighbourhood_group_Staten Island`)
#low vs medium PV12 value= 0.9195
ks.test(LOWPRICE$`neighbourhood_group_Staten Island`,HIGHPRICE$`neighbourhood_group_Staten Island`)
#low vs high PV13, P VALUE = 0.442
ks.test(MEDPRICE$`neighbourhood_group_Staten Island`,HIGHPRICE$`neighbourhood_group_Staten Island`)
#medium vs high pv23 -> 0.9999
W5 = ((1-.9195)+(1-.442)+(1-.9999))/3

#feature 6 latitude
ks.test(LOWPRICE$latitude,MEDPRICE$latitude)
#low vs medium PV12 value= 0
ks.test(LOWPRICE$latitude,HIGHPRICE$latitude)
#low vs high PV13, P VALUE = 0
ks.test(MEDPRICE$latitude,HIGHPRICE$latitude)
#medium vs high pv23 -> 0
W6 = ((1-0)+(1-0)+(1-0))/3
```

```
#feature 7 longitude
ks.test(LOWPRICE$longitude,MEDPRICE$longitude)#low vs medium PV12 value= 0
ks.test(LOWPRICE$longitude,HIGHPRICE$longitude)#low vs high PV13, P VALUE = 0
ks.test(MEDPRICE$longitude,HIGHPRICE$longitude)#medium vs high pv23 -> 0
W7 = ((1-0)+(1-0)+(1-0))/3

#feature 8 roomtype entire home
ks.test(LOWPRICE$`room_type_Entire home/apt`,MEDPRICE$`room_type_Entire home/apt`)
#low vs medium PV12 value= 0
ks.test(LOWPRICE$`room_type_Entire home/apt`,HIGHPRICE$`room_type_Entire home/apt`)
#low vs high PV13, P VALUE = 0
ks.test(MEDPRICE$`room_type_Entire home/apt`,HIGHPRICE$`room_type_Entire home/apt`)
#medium vs high pv23 -> 0
W8 = ((1-0)+(1-0)+(1-0))/3

#feature 9 roomtype priv
ks.test(LOWPRICE$`room_type_Private room`,MEDPRICE$`room_type_Private room`)
#low vs medium PV12 value= 0
ks.test(LOWPRICE$`room_type_Private room`,HIGHPRICE$`room_type_Private room`)
#low vs high PV13, P VALUE = 0
ks.test(MEDPRICE$`room_type_Private room`,HIGHPRICE$`room_type_Private room`)
#medium vs high pv23 -> 0
W9 = ((1-0)+(1-0)+(1-0))/3

#feature 10 roomtype shared
ks.test(LOWPRICE$`room_type_Shared room`,MEDPRICE$`room_type_Shared room`)
#low vs medium PV12 value= 0
ks.test(LOWPRICE$`room_type_Shared room`,HIGHPRICE$`room_type_Shared room`)
#low vs high PV13, P VALUE = 0
ks.test(MEDPRICE$`room_type_Shared room`,HIGHPRICE$`room_type_Shared room`)
#medium vs high pv23 -> 0.9985
W10 = ((1-0)+(1-0)+(1-.9985))/3
```

```
#feature 11 min nights
ks.test(LOWPRICE$minimum_nights,MEDPRICE$minimum_nights)#low vs medium PV12 value= 0
ks.test(LOWPRICE$minimum_nights,HIGHPRICE$minimum_nights)#low vs high PV13, P VALUE = 0
ks.test(MEDPRICE$minimum_nights,HIGHPRICE$minimum_nights)#medium vs high pv23 ->0
W11 = ((1-0)+(1-0)+(1-0))/3

#feature 12 number_of_reviews
ks.test(LOWPRICE$number_of_reviews,MEDPRICE$number_of_reviews)#low vs medium PV12 value= 0
ks.test(LOWPRICE$number_of_reviews,HIGHPRICE$number_of_reviews)#low vs high PV13, P VALUE = 0
ks.test(MEDPRICE$number_of_reviews,HIGHPRICE$number_of_reviews)#medium vs high pv23 -> 0
W12 = ((1-0)+(1-0)+(1-0))/3

#feature 13 reviews per month
ks.test(LOWPRICE$reviews_per_month,MEDPRICE$reviews_per_month)#low vs medium PV12 value= 0.007786
ks.test(LOWPRICE$reviews_per_month,HIGHPRICE$reviews_per_month)#low vs high PV13, P VALUE = 0
ks.test(MEDPRICE$reviews_per_month,HIGHPRICE$reviews_per_month)#medium vs high pv23 -> 0
W13 = ((1-0.007786)+(1-0)+(1-0))/3

#feature 14 calculated host listing count
ks.test(LOWPRICE$calculated_host_listings_count,MEDPRICE$calculated_host_listings_count)#low vs medium PV12 value= 0
ks.test(LOWPRICE$calculated_host_listings_count,HIGHPRICE$calculated_host_listings_count)#low vs high PV13, P VALUE = 0
ks.test(MEDPRICE$calculated_host_listings_count,HIGHPRICE$calculated_host_listings_count)#medium vs high pv23 -> 0
W14 = ((1-0)+(1-0)+(1-0))/3

#feature 15 availability 365
ks.test(LOWPRICE$availability_365,MEDPRICE$availability_365)#low vs medium PV12 value= 0
ks.test(LOWPRICE$availability_365,HIGHPRICE$availability_365)#low vs high PV13, P VALUE = 0
ks.test(MEDPRICE$availability_365,HIGHPRICE$availability_365)#medium vs high pv23 ->0
W15 = ((1-0)+(1-0)+(1-0))/3
```

```
#add the weights to original scaled data
SNYC_weighted = rbind(data.frame(W1*S_NYC1[,1],W2*S_NYC1[,2],W3*S_NYC1[,3],
                    W4*S_NYC1[,4],W5*S_NYC1[,5],W6*S_NYC1[,6],
                    W7*S_NYC1[,7],W8*S_NYC1[,8],W9*S_NYC1[,9],
                    W10*S_NYC1[,10],W11*S_NYC1[,11],W12*S_NYC1[,12],
                    W13*S_NYC1[,13],W14*S_NYC1[,14],W15*S_NYC1[,15],S_NYC1[,16]))

#rescale the weighted data
S_NYCWS<-SNYC_weighted%>%mutate_if(is.numeric, function (x) as.vector(scale(x)))

#split into train and test set
LOWPRICEw = S_NYCWS[which(S_NYCWS$target =="Low"),]
n<-nrow(LOWPRICEw)
train<-sample(1:n, 0.8*n)
trainset_LOW_w <- LOWPRICEw[train,]
testset_LOW_w <- LOWPRICEw[-train,]

##TRAIN/TEST SET: MEDIUM
MEDPRICEw = S_NYCWS[which(S_NYCWS$target =="Medium"),]
n<-nrow(MEDPRICEw)
train<-sample(1:n, 0.8*n)
trainset_MED_w <- MEDPRICEw[train,]
testset_MED_w <- MEDPRICEw[-train,]

##TRAIN/TEST SET: HIGH
HIGHPRICEw = S_NYCWS[which(S_NYCWS$target =="High"),]
n<-nrow(HIGHPRICEw)
train<-sample(1:n, 0.8*n)
trainset_HIGH_w <- HIGHPRICE[train,]
testset_HIGH_w <- HIGHPRICE[-train,]
```

```
#Train set weighted and test set weighted
TRAIN_SETWS = rbind(trainset_LOW_w,trainset_MED_w,trainset_HIGH_w)
TEST_SETWS = rbind(testset_LOW_w,testset_MED_w,testset_HIGH_w)
#split into labels and predictors
TRAIN_predictor <- TRAIN_SETWS[,-c(16)]
TRAIN_label <- TRAIN_SETWS[, "target"]
TEST_predictor <- TEST_SETWS[,-c(16)]
TEST_label <- TEST_SETWS[,"target"]
TRAIN_predictor=as.data.frame(TRAIN_predictor)
TEST_predictor=as.data.frame(TEST_predictor)
TRAIN_label=as.data.frame(TRAIN_label)
TEST_label=as.data.frame(TEST_label)
cl = TRAIN_label[,1, drop = TRUE]
testcl = TEST_label[,1, drop = TRUE]

library(class)
#run the knn with best k on weighted standardized dataset
knn.predtrainWS <- knn(train= TRAIN_predictor,
                       test=TRAIN_predictor,
                       cl= cl,
                       k=21)

knn.predtestWS<- knn(train= TRAIN_predictor,
                     test=TEST_predictor ,
                     cl= cl,
                     k=21)
conftrainWS<-table(data.frame(knn.predtrainWS,cl))
conftestWS<-table(data.frame(knn.predtestWS,testcl))
mean(knn.predtrainWS == cl)
mean(knn.predtestWS == testcl)
```

```r
#confidence intervals
library(DescTools)
Conf(conftrainWS)
Conf(conftestWS)
#weighted knn multiple k values to obtain the plot
K.set = c(5,10,15,20:25,30,40,50,80,100)
knn.train.accuracy <- numeric(length(K.set))
traincl = TRAIN_label[,1, drop = TRUE]
for (j in 1:length(K.set)){
  set.seed(1)
  knn.predWS <- knn(train=TRAIN_predictor,
                    test=TRAIN_predictor,
                    cl=traincl,
                    k=K.set[j])
  knn.train.accuracy[j] <- mean(knn.predWS == traincl)
}
knn.train.accuracy
#test set multiple values
knn.test.accuracy <- numeric(length(K.set))

cl = TRAIN_label[,1, drop = TRUE]
testcl = TEST_label[,1, drop = TRUE]

for (j in 1:length(K.set)){
  set.seed(1)
  knn.predWStest <- knn(train= TRAIN_predictor,
                    test=TEST_predictor,
                    cl= cl,
                    k=K.set[j])
  knn.test.accuracy[j] <- mean(knn.predWStest == testcl)

}
```

```r
#plot the K values versus the train and test set accuracy
plot(K.set, knn.train.accuracy, type="o", col="blue", pch="o", lty=1, ylim=range( c(knn.train.accuracy,knn.test.accuracy) ) )
points(K.set, knn.test.accuracy, col="red", pch="*")
lines(K.set, knn.test.accuracy, col="red",lty=2)
###investigating K>80 since the training set performs better than the testing set
K.set = c(80:90,100,105:110)
knn.train.accuracy <- numeric(length(K.set))
traincl = TRAIN_label[,1, drop = TRUE]
for (j in 1:length(K.set)){
  set.seed(1)
  knn.predWS <- knn(train=TRAIN_predictor,
                    test=TRAIN_predictor,
                    cl=traincl,
                    k=K.set[j])
  knn.train.accuracy[j] <- mean(knn.predWS == traincl)
}
knn.train.accuracy
#test
knn.test.accuracy <- numeric(length(K.set))

cl = TRAIN_label[,1, drop = TRUE]
testcl = TEST_label[,1, drop = TRUE]

for (j in 1:length(K.set)){
  set.seed(1)
  knn.predWStest <- knn(train= TRAIN_predictor,
                        test=TEST_predictor,
                        cl= cl,
                        k=K.set[j])
  knn.test.accuracy[j] <- mean(knn.predWStest == testcl)

}
```

```
#plot with 80<k<110(weighted knn)
plot(K.set, knn.train.accuracy, type="o", col="blue", pch="o", lty=1, ylim=range( c(knn.train.accuracy,knn.test.accuracy) ) )
points(K.set, knn.test.accuracy, col="red", pch="*")
lines(K.set, knn.test.accuracy, col="red",lty=2)

##Individual investigation
#make a train test set with only low price and high price

TRAINLOWHIGHp=as.data.frame(rbind(trainset_LOW[,-16],trainset_HIGH[,-16]))
TESTLOWHIGHp=as.data.frame(rbind(testset_LOW[,-16],testset_HIGH[,-16]))
TRAINLOWHIGHlab=as.data.frame(rbind(trainset_LOW[,16],trainset_HIGH[,16]))
TESTLOWHIGHlab=as.data.frame(rbind(testset_LOW[,16],testset_HIGH[,16]))
cl1 = TRAINLOWHIGHlab[,1, drop = TRUE]
testcl1 = TESTLOWHIGHlab[,1, drop = TRUE]

knn.predtrain <- knn(train= TRAINLOWHIGHp,
                     test=TRAINLOWHIGHp,
                     cl= cl1,
                     k=21)

knn.predtest <- knn(train= TRAINLOWHIGHp,
                    test=TESTLOWHIGHp,
                    cl= cl1,
                    k=21)

conftrain1<-table(data.frame(knn.predtrain,cl1))
conftest1<-table(data.frame(knn.predtest,testcl1))
mean(knn.predtrain == cl1)
mean(knn.predtest == testcl1)  #91%
##################Med vs high

TRAINMEDHIGHp=as.data.frame(rbind(trainset_MED[,-16],trainset_HIGH[,-16]))
TESTMEDHIGHp=as.data.frame(rbind(testset_MED[,-16],testset_HIGH[,-16]))

TRAINMEDHIGHlab=as.data.frame(rbind(trainset_MED[,16],trainset_HIGH[,16]))
TESTMEDHIGHlab=as.data.frame(rbind(testset_MED[,16],testset_HIGH[,16]))



cl2 = TRAINMEDHIGHlab[,1, drop = TRUE]
testcl2 = TESTMEDHIGHlab[,1, drop = TRUE]


knn.predtrain2 <- knn(train= TRAINMEDHIGHp,
                      test=TRAINMEDHIGHp,
                      cl= cl2,
                      k=21)

knn.predtest2 <- knn(train= TRAINMEDHIGHp,
                     test=TESTMEDHIGHp,
                     cl= cl2,
                     k=21)

conftrain2<-table(data.frame(knn.predtrain2,cl2))
conftest2<-table(data.frame(knn.predtest2,testcl2))
mean(knn.predtrain2 == cl2)
mean(knn.predtest2 == testcl2)
```

```
#################LOW vs Med

TRAINLOWMEDp=as.data.frame(rbind(trainset_LOW[,-16],trainset_MED[,-16]))
TESTLOWMEDp=as.data.frame(rbind(testset_LOW[,-16],testset_MED[,-16]))

TRAINLOWMEDlab=as.data.frame(rbind(trainset_LOW[,16],trainset_MED[,16]))
TESTLOWMEDlab=as.data.frame(rbind(testset_LOW[,16],testset_MED[,16]))


cl3 = TRAINLOWMEDlab[,1, drop = TRUE]
testcl3 = TESTLOWMEDlab[,1, drop = TRUE]


knn.predtrain3 <- knn(train= TRAINLOWMEDp,
                      test=TRAINLOWMEDp,
                      cl= cl3,
                      k=21)

knn.predtest3 <- knn(train= TRAINLOWMEDp,
                     test=TESTLOWMEDp,
                     cl= cl3,
                     k=21)

conftrain3<-table(data.frame(knn.predtrain3,cl3))
conftest3<-table(data.frame(knn.predtest3,testcl3))
mean(knn.predtrain3 == cl3)
mean(knn.predtest3 == testcl3)
```