**Written by Jia Yu**
**Step 0:**

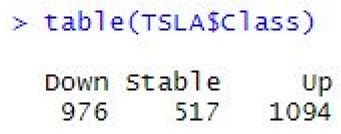
Dataset source/link
https://finance.yahoo.com/quote/TSLA/history?period1=1413158400&period2=1602547200&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true

Description:
This dataset contains daily stock prices for Tesla. The practical use of the classes is that it is indicative of the stocks performance. It is useful to automatically classify a given day's performance so real world users can understand the overall performance of TSLA stock and make decisions based on the patterns perceived.

Features description and meaning:
All numerical features
Date: Date of recording
Open: Day’s opening price
High: Day’s peak price
Low: Day’s lowest price
Close: Day’s closing price
Adj. Close: Day’s closing price including dividends and splits
Volume: Day’s trading volume
Daily Return: Days Open- [Day - 1]’s opening price
EarnLossGapTrade: Day’s Opening - [Day - 1]’s closing price
threshold <- [Day - 1]’s Opening price * 0.006 = 0.6% threshold
Class <- Up, Down, or Stable indicating the stocks performance compared to previous day

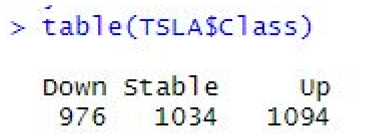
Each case within this dataset is a given day from 2014-2020.
We compare our Daily Return with our “Threshold” to determine the trend of the market for that given day. That is if our Daily Return is greater than our threshold value for that day, it is the case is classified as “UP”. Conversely, if the daily return is less than our threshold value, it is classified as “DOWN”. Any other instance would fall between -0.6% to 0.6% and is classified as “STABLE”.

Number of Cases: 2588
Number of features: 9
Number of classes: 3

Size of classes:

```
> table(TSLA$Class)

  Down Stable     Up
   976    517   1094
```

**Step 1:**
**Clean Up**:
Because the first row(first day) of the stock does not have a daily return, earn loss gap grade, or threshold value, I set the values equal to the mean values to not skew any data on the mean, and standard deviations.

**Balancing the class sizes**: I observed “Stable” had about half of the entries has Down and Up, so doubled the class by cloning. The new size of the classes can be observed:



```
> table(TSLA$Class)

  Down Stable     Up
   976   1034   1094
```

**Binary Recoding**: The class value “Up”, “Down”, and “Stable” are recoded to -1, 0, and 1.

**Step 2:**
**1.1** The classification task is to be able to automatically classify a day's stock trend based on T - X days. The practicality of this classification is many fold and can be interpreted in different ways but moreover, it is useful in understanding the overall movement of stock based on historical data. That is, if trends continue, we can expect results to be as predicted; as a result, well informed decisions are supported through this research.

**1.2** . Tesla Inc. designs and manufactures electric vehicles globally. With the rise of popularity with electric vehicles, Tesla has emerged as a leading force within the industry. As a result, their stock prices have reflected this rise. Within this dataset is daily stock data from 2014 - 2020 pertinent to classifying a market trend. This report uses K-Nearest Neighbors to analyze the trend of the market for a given day based on the movement since the previous day.

**1.3**

**Histograms**

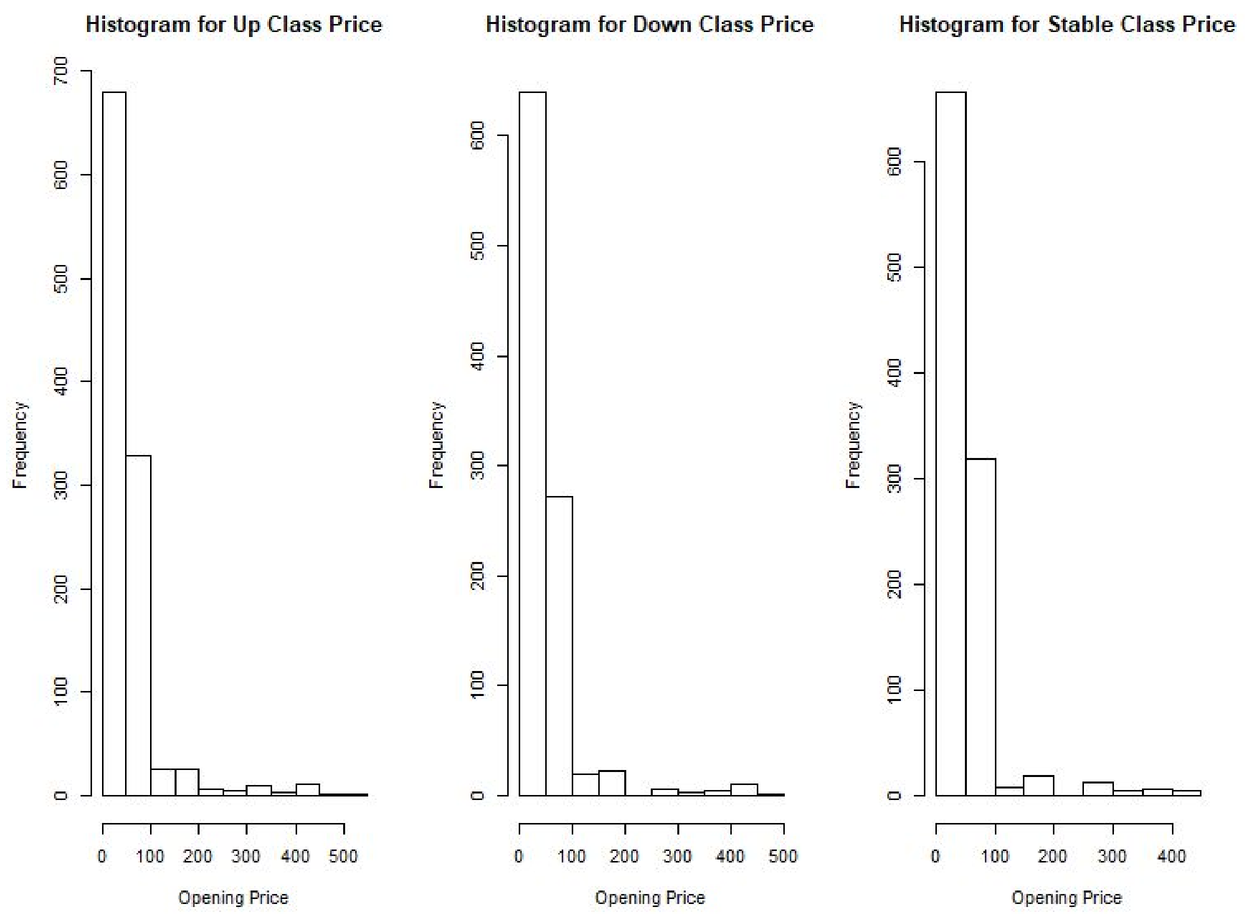
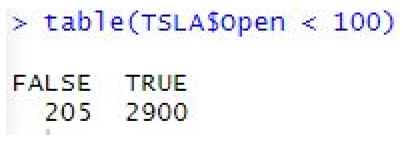


***Figure 1.3a***
The histograms for the three classes observed share similar distributions. It is easily observed that the opening price for Tesla stock spent a majority of time between 2014-2020 being under

$100, where the highest percentages of distributions are. As a result, the histograms are skewed right. In fact we can observe this with the following and deduce that 93.4% of all the days had a stock price of lower than $100.



```
> table(TSLA$Open < 100)

FALSE  TRUE 
  205  2900 
```

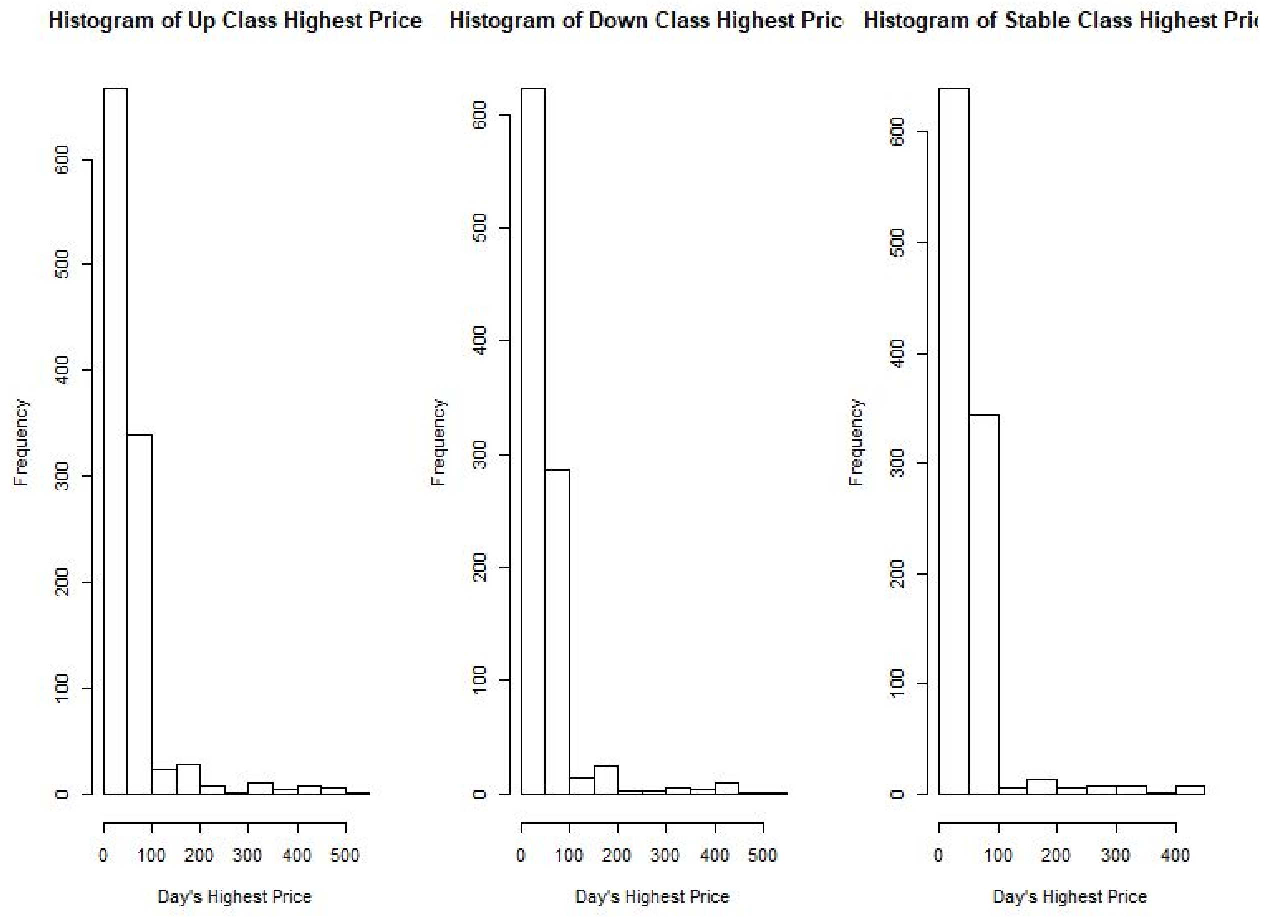


*Figure 1.3b*
Similarly, as the day’s highest price is usually close to the day’s opening price, we see similar distributions in each of the classes.

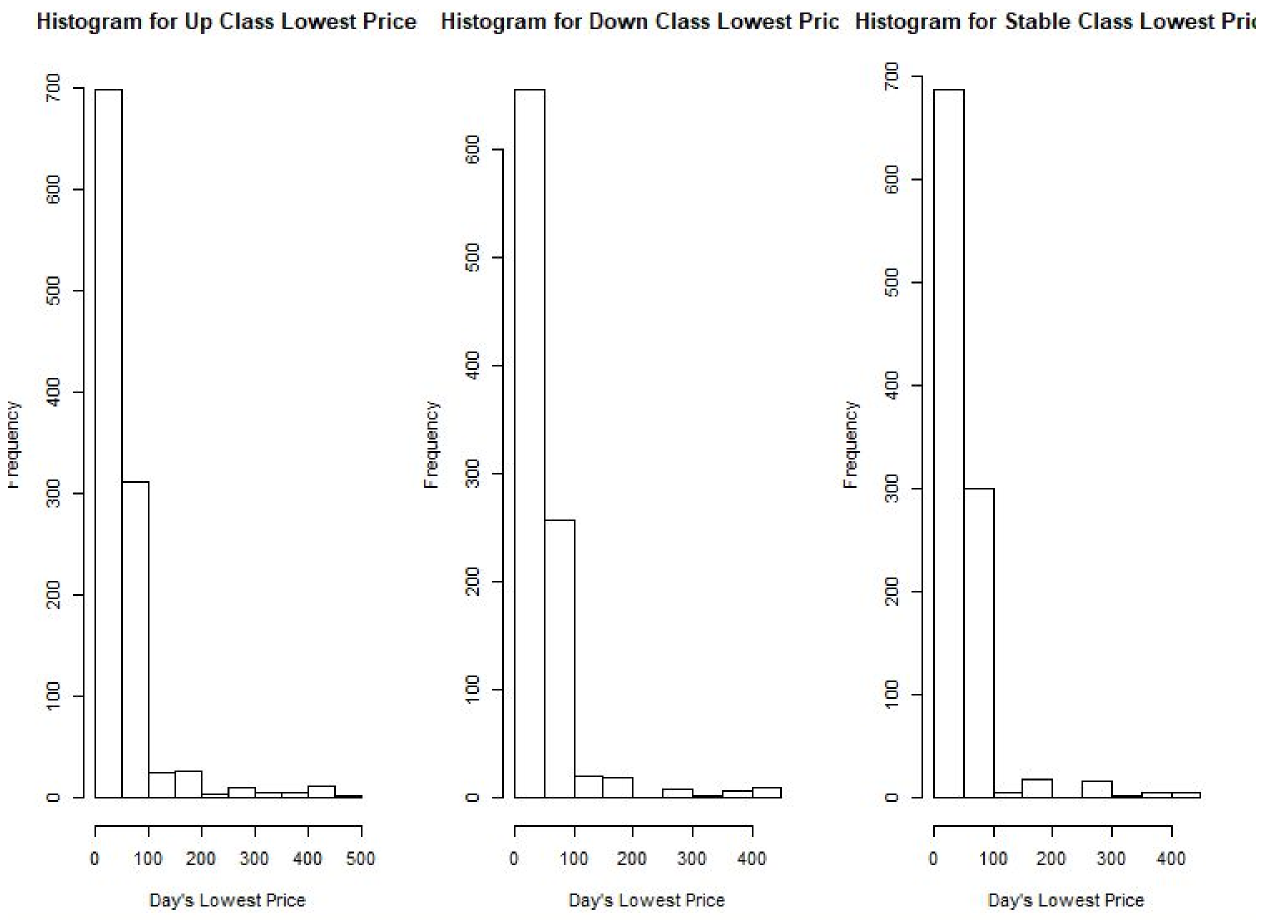


***Figure 1.3c***
The daily lowest price does not deviate far from the day’s opening price, or the highest price, so we also see similar distributions in each of the classes.

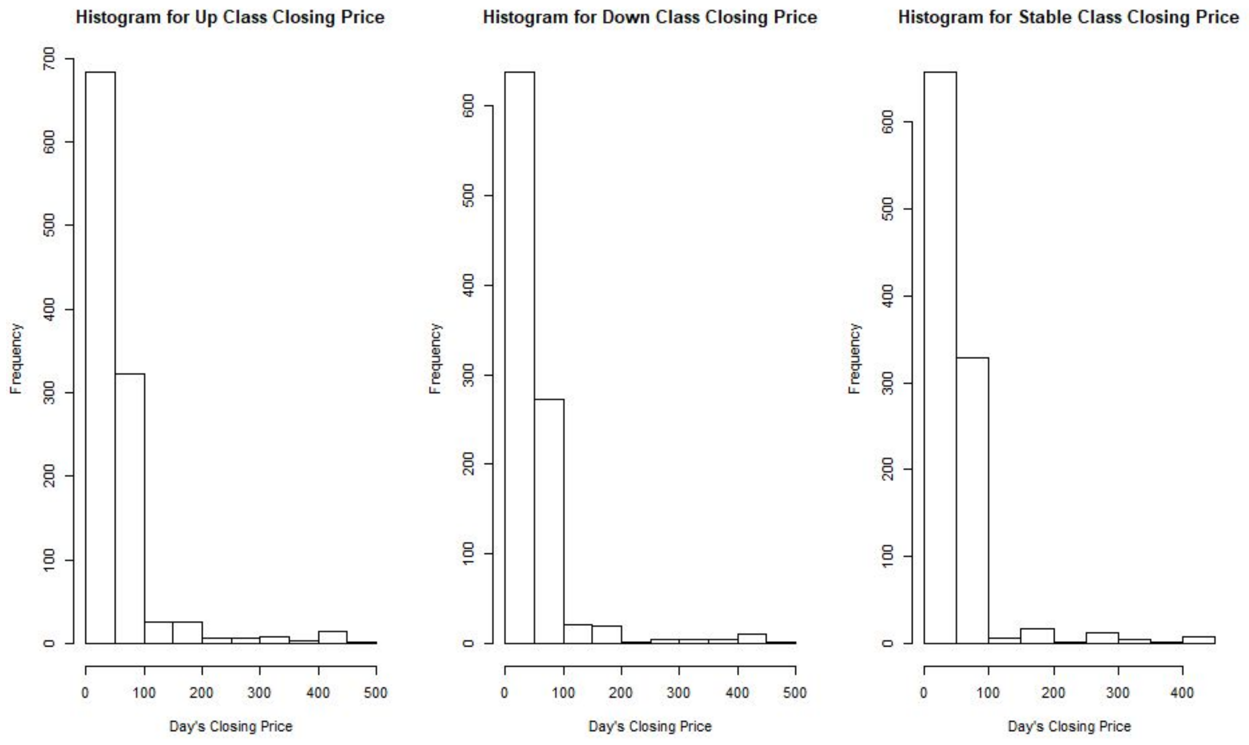


***Figure 1.3d***

The daily closing price does not deviate far from the day's opening price, highest price, or lowest price so we also see similar distributions in each of the classes.

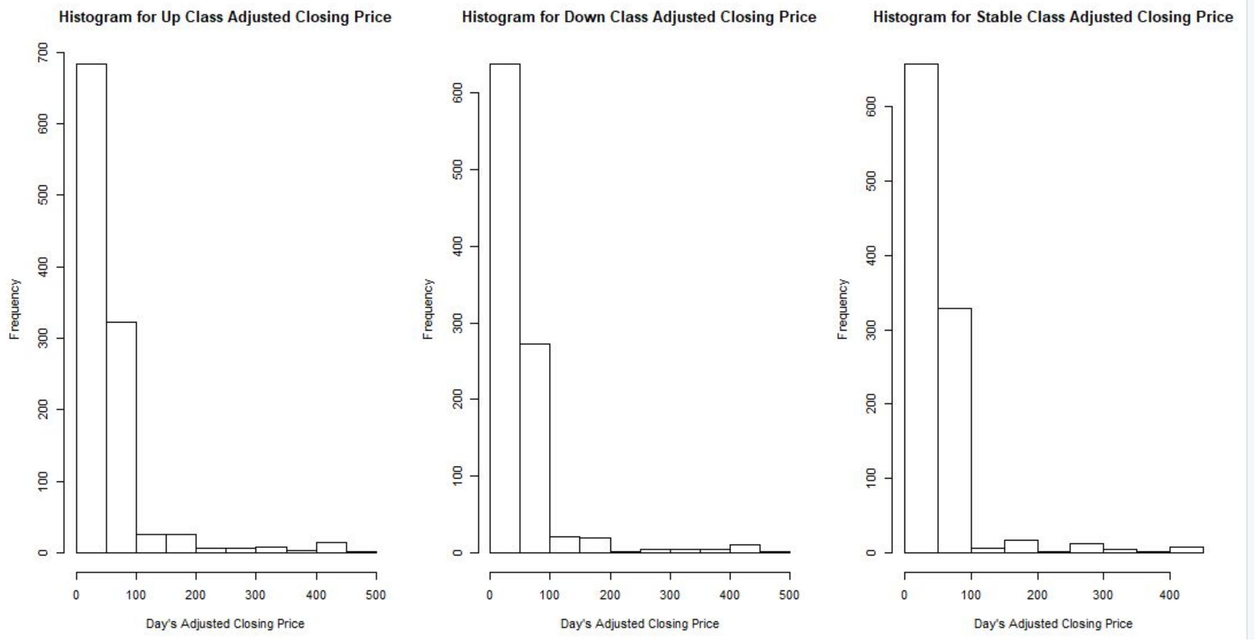


***Figure 1.3e***

The daily adjusted closing price does not deviate far from the day's opening price, highest price, lowest price, or closing price so we also see similar distributions in each of the classes.

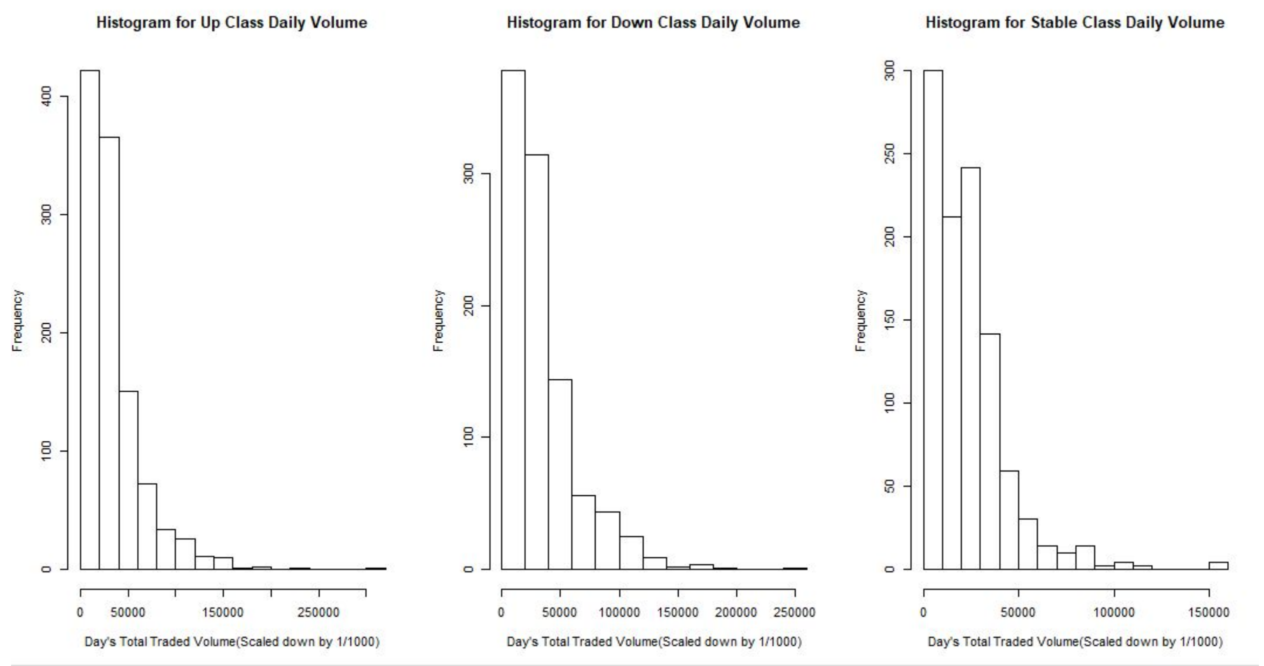
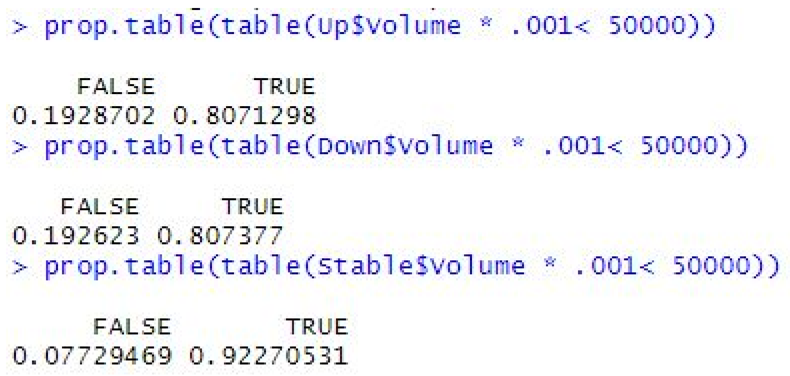


***Figure 1.3f***

We can clearly distinguish some glaring differences in the histograms for volume traded. After observing, the histogram for “Up” class and “Down” class are extremely similar, while “Stable” class differs. That is, we can clearly observe that when the stock price is stable, users trade less for that given day. This proposition is supported if we further observe:



```
> prop.table(table(Up$Volume * .001< 50000))

    FALSE      TRUE 
0.1928702 0.8071298 
> prop.table(table(Down$Volume * .001< 50000))

   FALSE     TRUE 
0.192623 0.807377 
> prop.table(table(Stable$Volume * .001< 50000))

     FALSE       TRUE 
0.07729469 0.92270531 
```

We can see that for both up and down class, 19.2% of cases (days) traded more than 50000000 shares. For stable class, only 7.7% of the days had traded volume of  more than 50000000 shares. From this data, we can say that users are more inclined to trade large amounts when the market is dropping or rising significantly (greater or less than 0.6%)

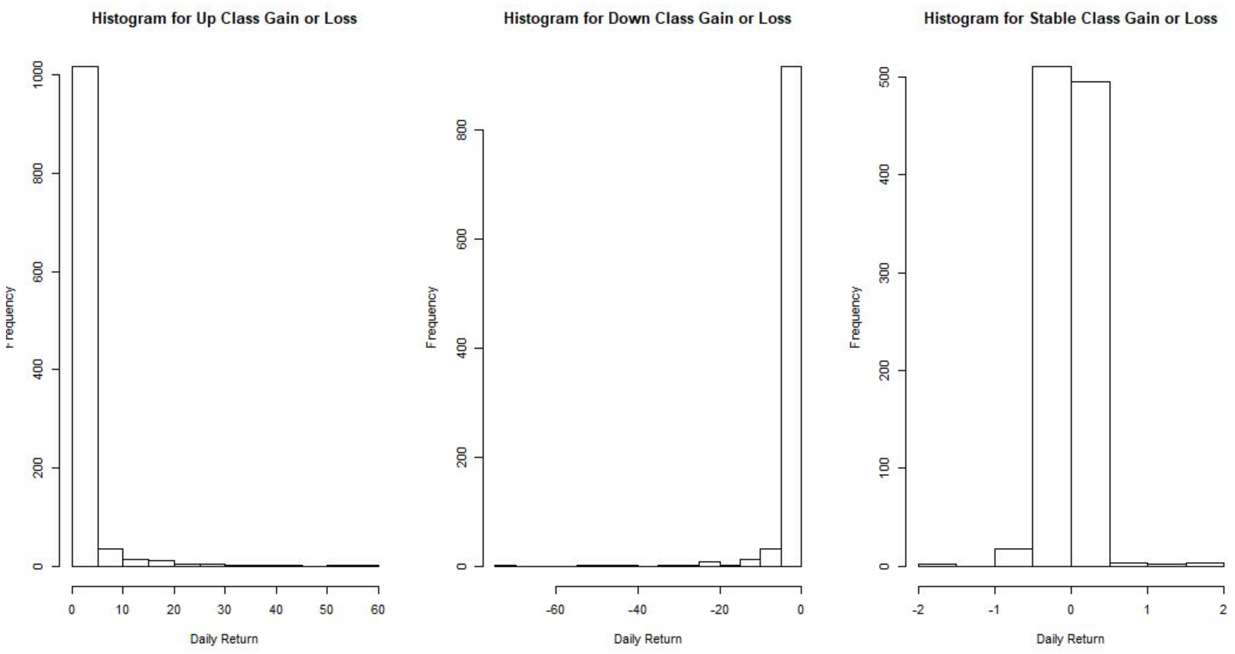


***Figure 1.3g***

The histograms for Daily Return differ greatly, but as expected. For “Up” Class, we expect a return of greater than 0.6%, thus the daily return will always be greater than 0, as shown on the. histogram Similarly, the daily return for the Down Class will always be less than 0. The Stable class, is the -0.6% to 0.6% change in opening price, so we can expect values to be around 0. Histogram for Up class is skewed right, down class is skewed left, and stable class follows a more normal distribution.

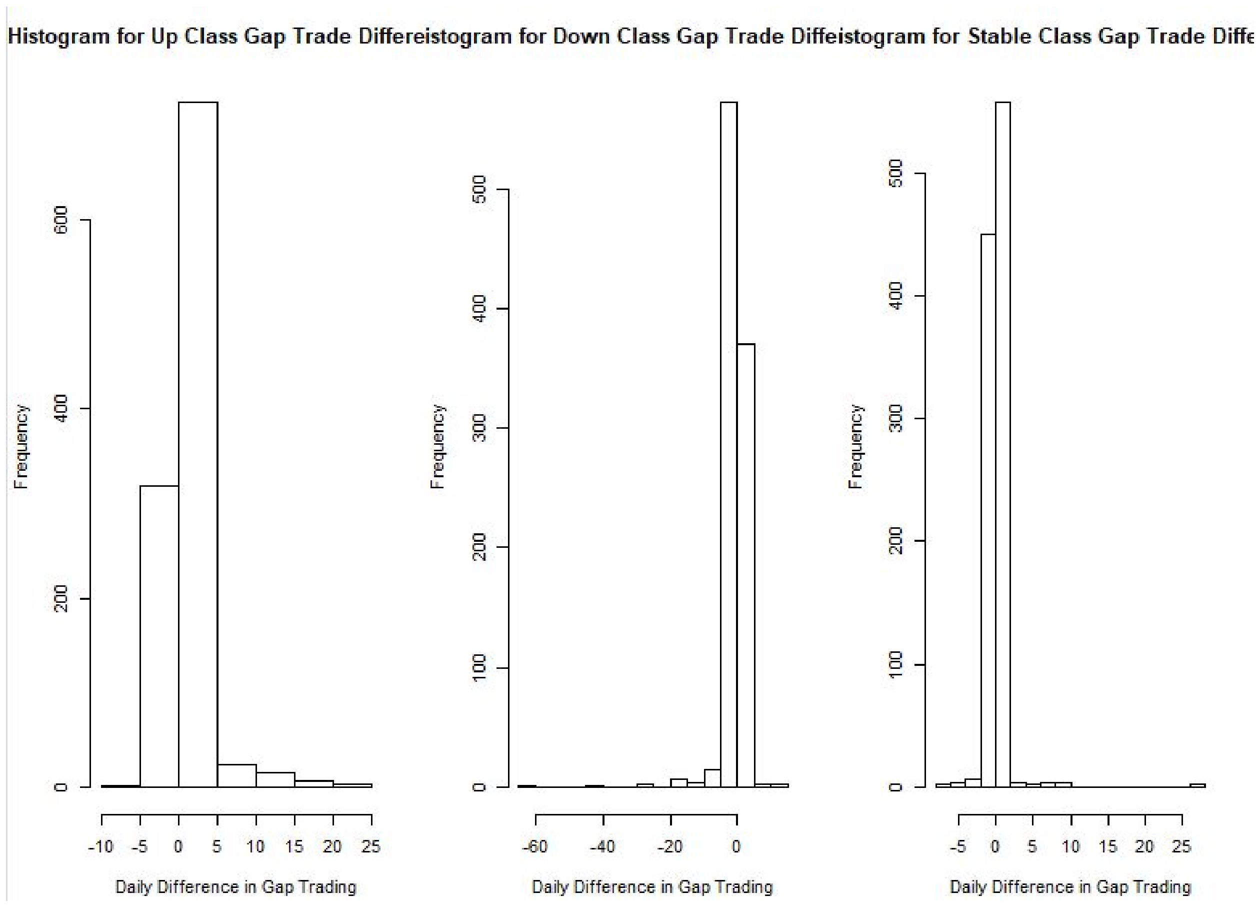


***Figure 1.3h***

The histograms for Earn Loss Gap Trade differ greatly. That is, on days that are “Up”, we generally see a gain from yesterday’s closing price and today’s opening price. On days that are “Down”, we see high frequencies in values less than 0, indicating that we see a loss from yesterday’s closing and todays opening. On days that are “stable”, we see high frequencies around 0.

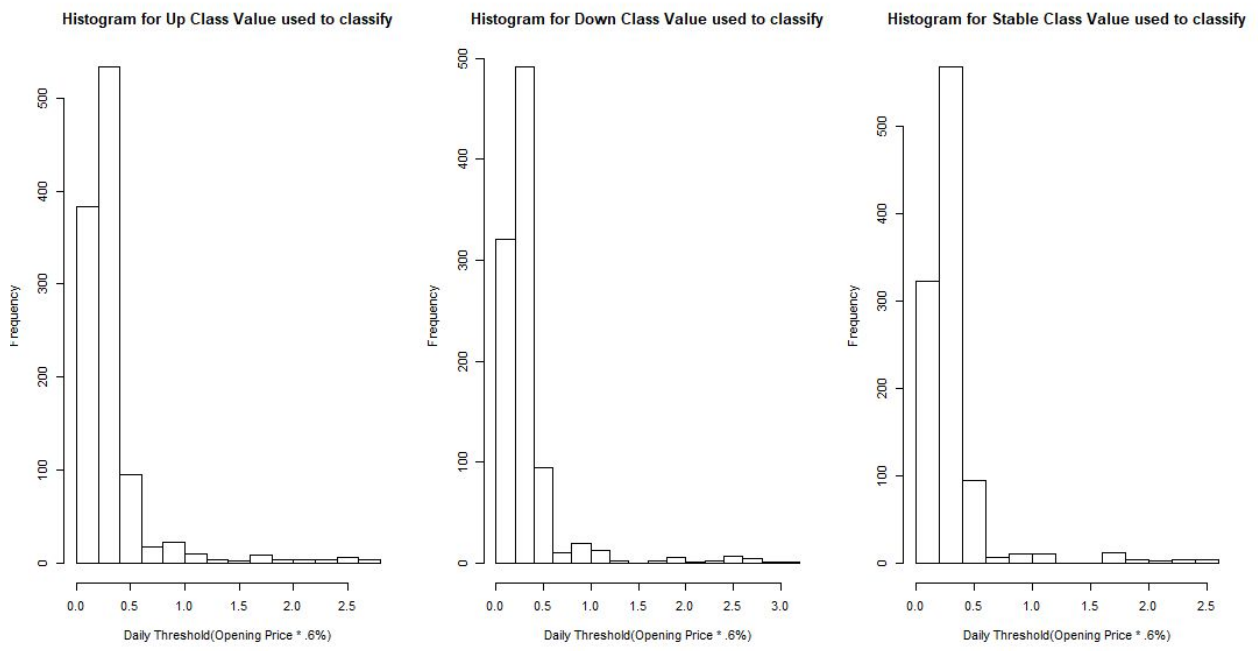


***Figure 1.3i***
All of the threshold levels are skewed right, and understandably match the Opening Price's histogram since the threshold values are directly derived from Opening price multiplied by .006.

**T-test Comparison and Discriminating Power:**
**Note:** The t-test statistic is used to determine the discriminating power for each pairing of histograms. That is, we can determine the true difference between the means of selected classes for that given feature by using the p-value to reject or fail to reject our null hypothesis.

For the following comparisons, the null hypothesis is denoted as: the true difference between in the means of the given classes is equal to 0. The alternative hypothesis is denoted as: the true difference in the means of the given classes is not equal to 0.

A p-value less than .05 means that we reject our null hypothesis, and conclude that the true difference between the mean of the two classes is not equal to 0. That is, there is a noticeable difference between the two means and the distribution of the feature for that class should not be similar to the distribution of the corresponding class. If our p-value is greater than .05, we fail to reject the null hypothesis and state that the true difference in means is equal to 0. That is, the distribution for that feature and class should be similar to the distribution to the corresponding class.

Furthermore, we can derive our discriminatory power with the equation: 1 - p-value. That is, when we have a low p-value < 0.5:
1. Conclude that the true difference in means is not equal to 0, and thus, distributions are expected to differ.
2. We will have a discriminatory power > .95

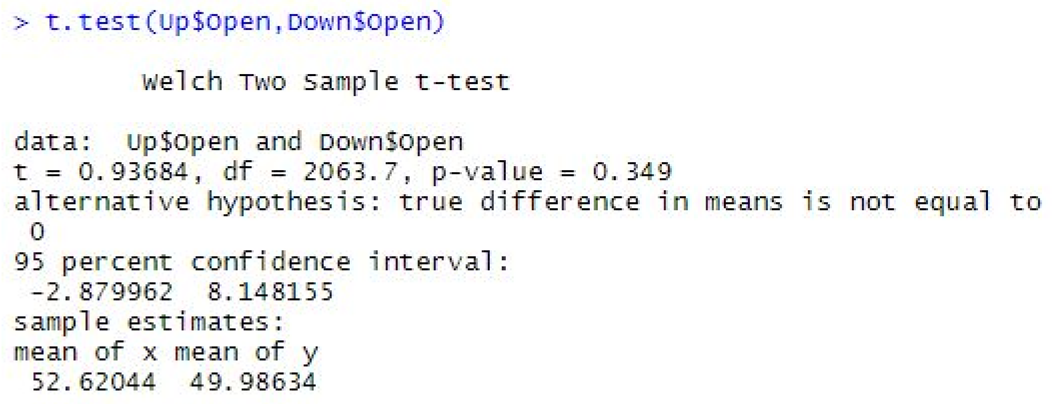
3. Observe which features discriminate more, that is- which features and class pairings influence the spread of a distribution. The closer the value is to 1, the greater the discrimination.

**Discriminatory Power Table**

| | Open Price | Highest Price | Lowest Price | Closing Price | Adj. Closing Price | Volume | Daily Return | Earn Loss Gap Trade | Threshold |
|---|---|---|---|---|---|---|---|---|---|
| Up Down | .651 | .63 | .677 | .639 | .639 | .2246 | 1 | 1 | .306 |
| Up Stable | .926 | .922 | .908 | .898 | .898 | 1 | 1 | 1 | .707 |
| Down Stable | .570 | .579 | .477 | .494 | .494 | 1 | 1 | 1 | .840 |

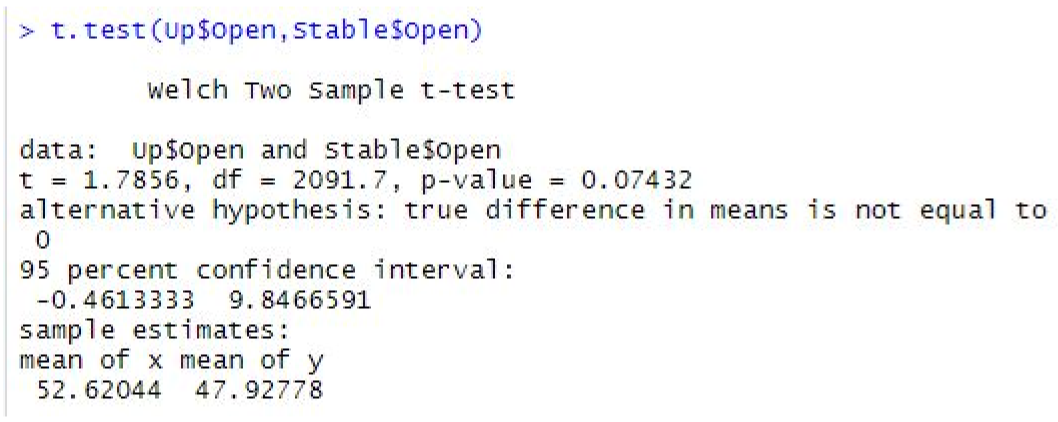
**Opening Price**

**Up and Down Class**



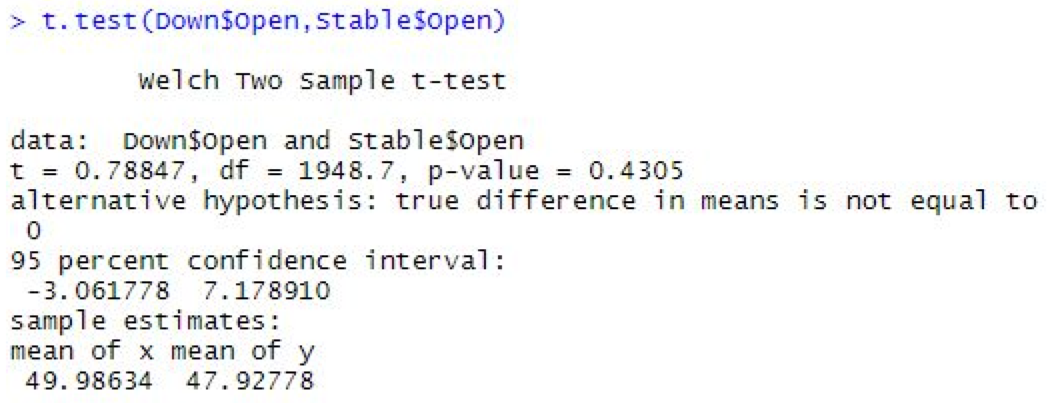
```
> t.test(Up$Open,Down$Open)

        Welch Two Sample t-test

data:  Up$Open and Down$Open
t = 0.93684, df = 2063.7, p-value = 0.349
alternative hypothesis: true difference in means is not equal to
 0
95 percent confidence interval:
 -2.879962  8.148155
sample estimates:
mean of x mean of y
 52.62044  49.98634
```

**Up and Stable Class**



```
> t.test(Up$Open,Stable$Open)

        Welch Two Sample t-test

data:  Up$Open and Stable$Open
t = 1.7856, df = 2091.7, p-value = 0.07432
alternative hypothesis: true difference in means is not equal to
 0
95 percent confidence interval:
 -0.4613333  9.8466591
sample estimates:
mean of x mean of y
 52.62044  47.92778
```

**Down and Stable Class**

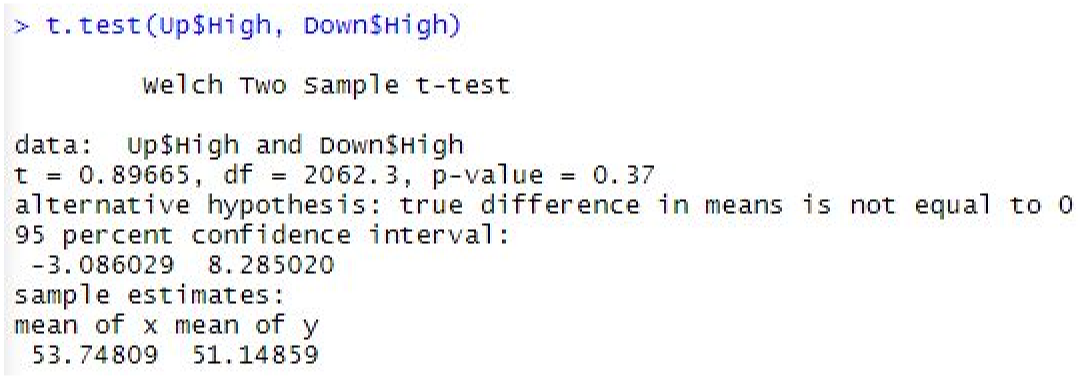


```
> t.test(Down$Open,Stable$Open)

        Welch Two Sample t-test

data:  Down$Open and Stable$Open
t = 0.78847, df = 1948.7, p-value = 0.4305
alternative hypothesis: true difference in means is not equal to
 0
95 percent confidence interval:
 -3.061778  7.178910
sample estimates:
mean of x mean of y
 49.98634  47.92778
```

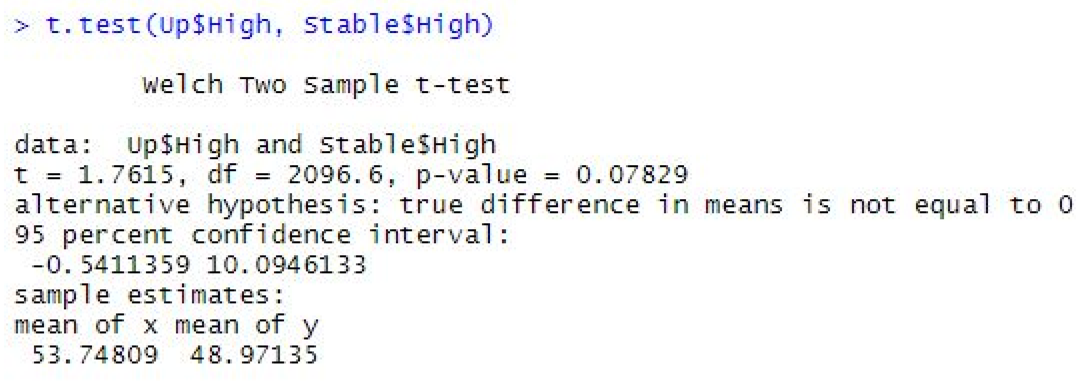
When comparing the p-values for the histograms for *Figure 1.3a,* we fail to reject the null hypothesis and conclude that the true difference in means for all three classes in the Opening Price feature is 0. That is, there is no significant difference and the feature does not discriminate by class. We obtain discriminatory values of .651, .926, and .570 for UpDown, UpStable, and DownStable. With values of .651 and .570, UpDown and DownStable's Opening Price is clearly not discriminatory, and shares a very similar correlation. With a value of .926, Opening Price for UpStable is not as discriminatory but still shares a similar correlation as we conclude that the true difference in means is 0.

**Highest Price**

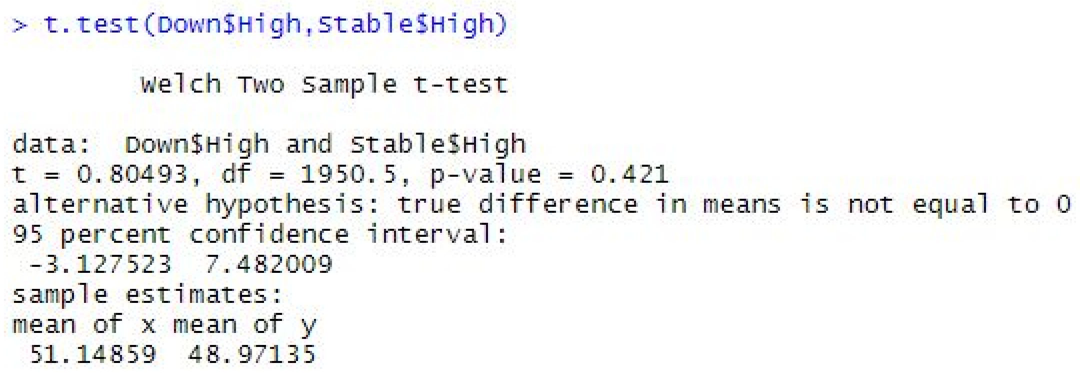
**Up and Down Class**


```
> t.test(Up$High, Down$High)

        Welch Two Sample t-test

data:  Up$High and Down$High
t = 0.89665, df = 2062.3, p-value = 0.37
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.086029  8.285020
sample estimates:
mean of x mean of y
 53.74809  51.14859
```

### Up and Stable Class


```
> t.test(Up$High, Stable$High)

        Welch Two Sample t-test

data:  Up$High and Stable$High
t = 1.7615, df = 2096.6, p-value = 0.07829
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.5411359 10.0946133
sample estimates:
mean of x mean of y
 53.74809  48.97135
```

### Down and Stable Class

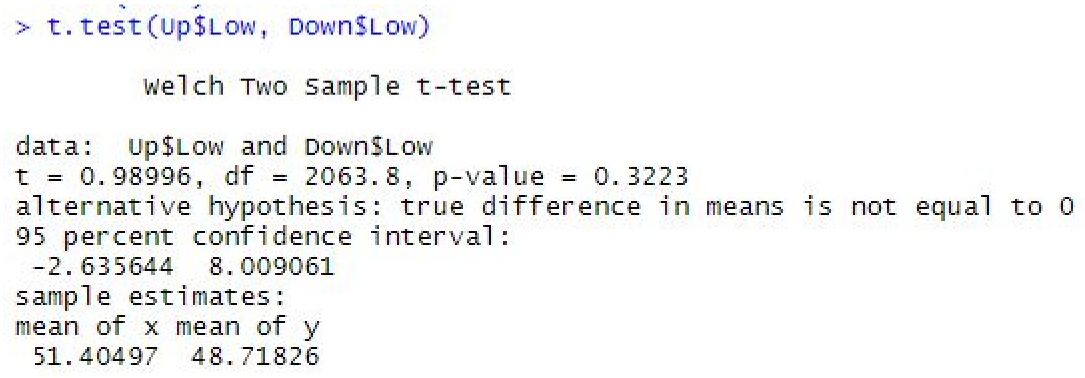

```
> t.test(Down$High,Stable$High)

        Welch Two Sample t-test

data:  Down$High and Stable$High
t = 0.80493, df = 1950.5, p-value = 0.421
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.127523  7.482009
sample estimates:
mean of x mean of y
 51.14859  48.97135
```

When comparing the p-values for the histograms for *Figure 1.3b,* we fail to reject the null hypothesis and conclude that the true difference in means for all three classes in the Highest Price feature is 0. That is, there is no significant difference and the feature does not discriminate by class. We obtain discriminatory values of .63, .922, and .579 for UpDown, UpStable, and DownStable. With values of .63 and .579, UpDown and DownStable's Highest Price is clearly not discriminatory, and shares a very similar correlation. With a value of .922, Highest Price for UpStable is not as discriminatory but still shares a similar correlation as we conclude that the true difference in means is 0.

As expected, much like the histograms and interpretation of histograms, the highest price's p-values and discriminatory powers are close to the opening price. With this assumption, we can expect the discriminatory powers for Low, Close, Adjusted Close, and Threshold to follow the same trend and have similar p-values and discriminatory powers.

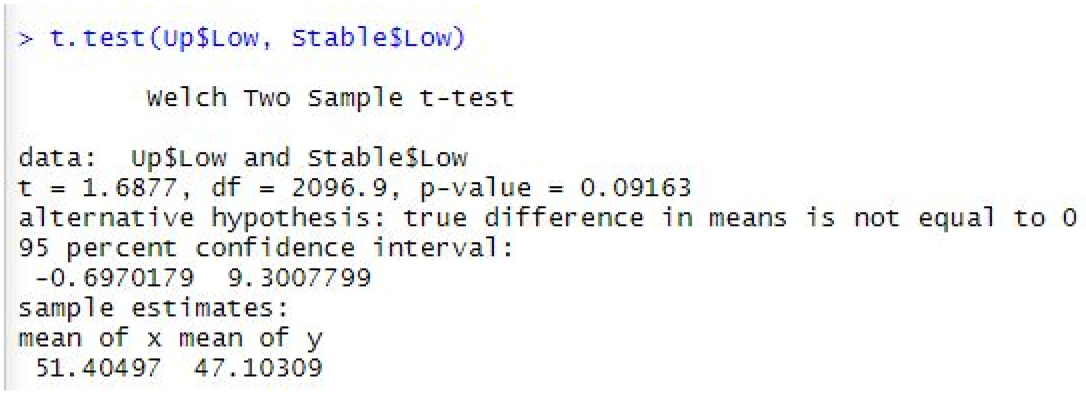
**Lowest Price**

**Up and Down Class**



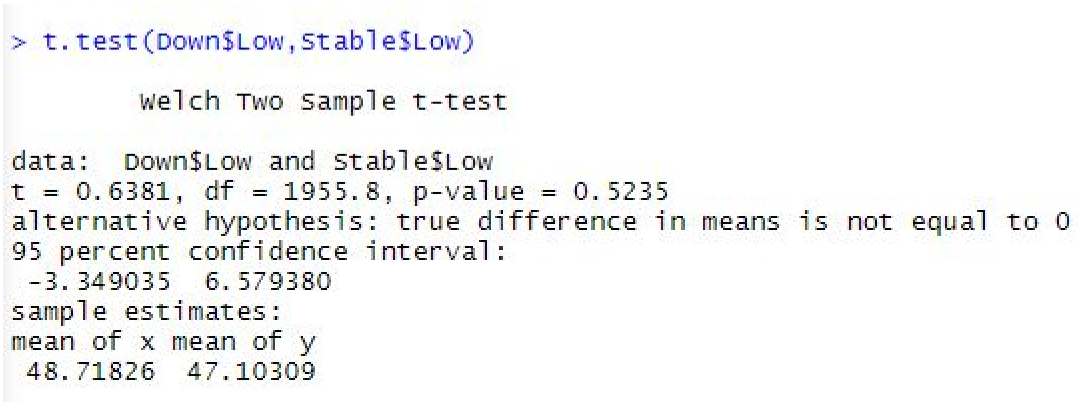
```
> t.test(Up$Low, Down$Low)

        Welch Two Sample t-test

data:  Up$Low and Down$Low
t = 0.98996, df = 2063.8, p-value = 0.3223
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.635644  8.009061
sample estimates:
mean of x mean of y
 51.40497  48.71826
```

**Up and Stable Class**



```
> t.test(Up$Low, Stable$Low)

        Welch Two Sample t-test

data:  Up$Low and Stable$Low
t = 1.6877, df = 2096.9, p-value = 0.09163
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.6970179  9.3007799
sample estimates:
mean of x mean of y
 51.40497  47.10309
```

**Down and Stable Class**

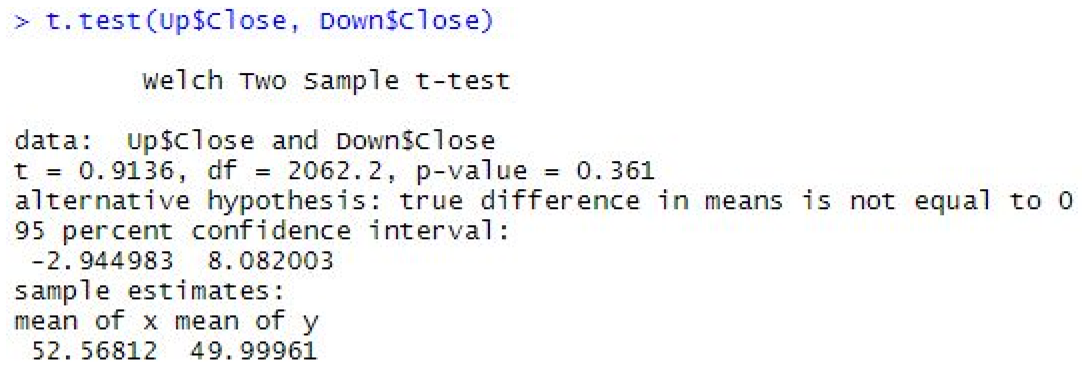


```
> t.test(Down$Low,Stable$Low)

        Welch Two Sample t-test

data:  Down$Low and Stable$Low
t = 0.6381, df = 1955.8, p-value = 0.5235
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.349035  6.579380
sample estimates:
mean of x mean of y
 48.71826  47.10309
```

When comparing the p-values for the histograms for *Figure 1.3c,* we fail to reject the null hypothesis and conclude that the true difference in means for all three classes in the Lowest Price feature is 0. That is, there is no significant difference and the feature does not discriminate by class. We obtain discriminatory values of .677, .908, and .477 for UpDown, UpStable, and DownStable. With values of .677 and .477, UpDown and DownStable's Lowest Price is clearly not discriminatory, and shares a very similar correlation. With a value of .908, Lowest Price for UpStable is not as discriminatory but still shares a similar correlation as we conclude that the true difference in means is 0.

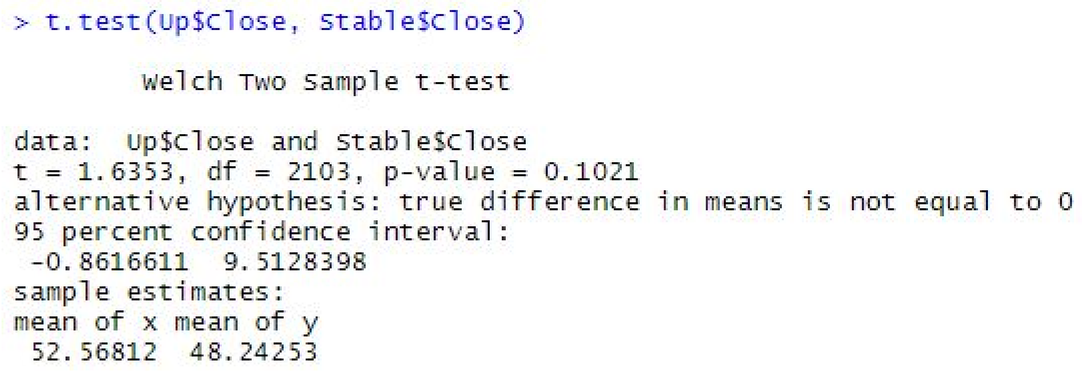
**Closing**

**Up and Down Class**



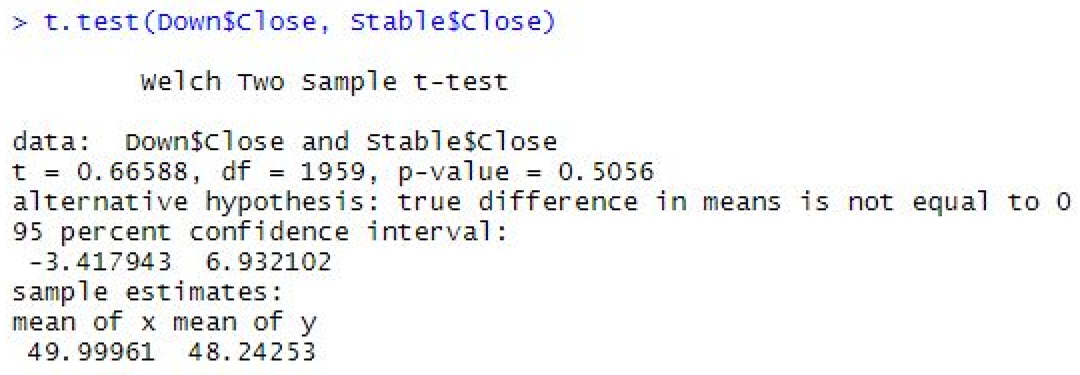
```
> t.test(Up$Close, Down$Close)

        Welch Two Sample t-test

data:  Up$Close and Down$Close
t = 0.9136, df = 2062.2, p-value = 0.361
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.944983  8.082003
sample estimates:
mean of x mean of y 
 52.56812  49.99961 
```

**Up and Stable Class**



```
> t.test(Up$Close, Stable$Close)

        Welch Two Sample t-test

data:  Up$Close and Stable$Close
t = 1.6353, df = 2103, p-value = 0.1021
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.8616611  9.5128398
sample estimates:
mean of x mean of y 
 52.56812  48.24253 
```

**Down and Stable Class**

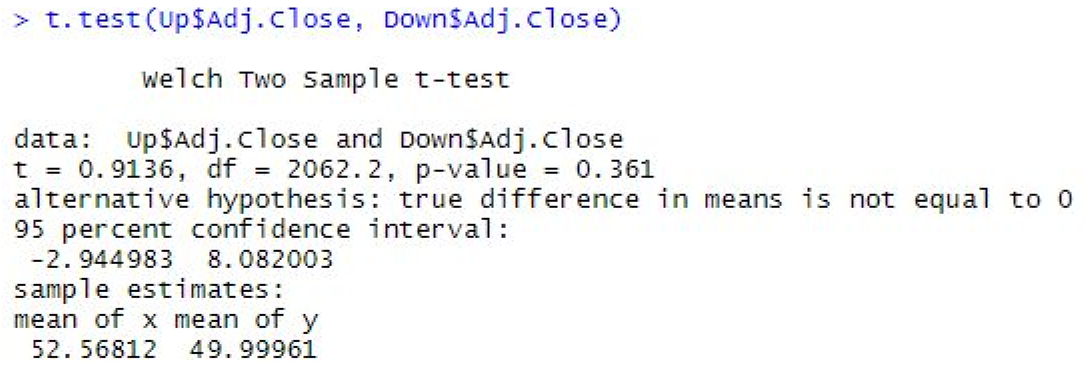


```
> t.test(Down$Close, Stable$Close)

        Welch Two Sample t-test

data:  Down$Close and Stable$Close
t = 0.66588, df = 1959, p-value = 0.5056
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.417943  6.932102
sample estimates:
mean of x mean of y 
 49.99961  48.24253 
```

When comparing the p-values for the histograms for *Figure 1.3d,* we fail to reject the null hypothesis and conclude that the true difference in means for all three classes in the Closing Price feature is 0. That is, there is no significant difference and the feature does not discriminate by class. We obtain discriminatory values of .639, .898, and .494 for UpDown, UpStable, and DownStable. With values of .639 and .494, UpDown and DownStable's Closing Price is clearly not discriminatory, and shares a very similar correlation. With a value of .898, Closing Price for UpStable is not as discriminatory but still shares a similar correlation as we conclude that the true difference in means is 0.

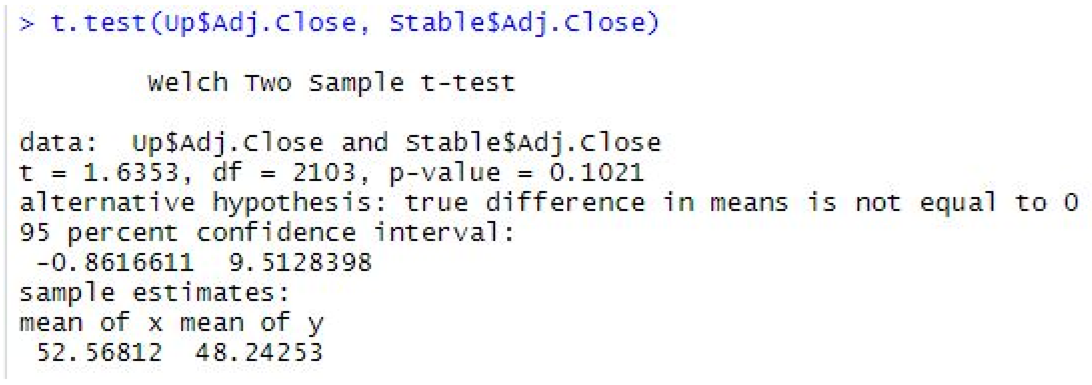
## Adjusted Closing

### Up and Down Class



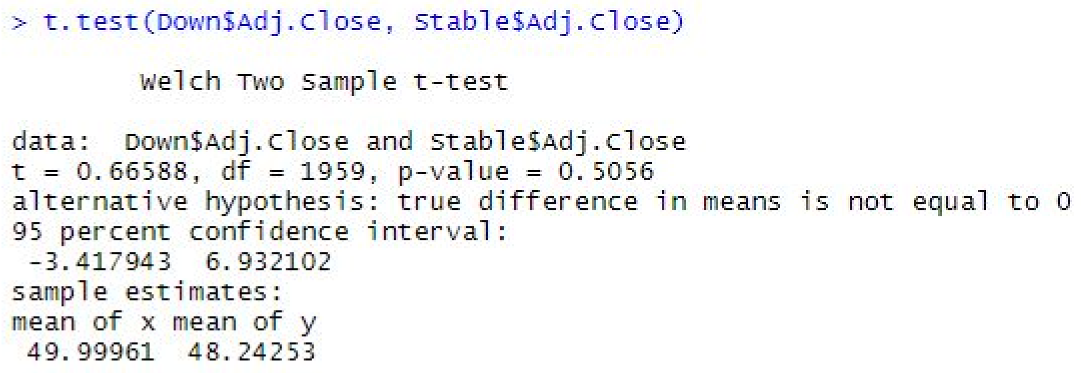
```
> t.test(Up$Adj.Close, Down$Adj.Close)

	Welch Two Sample t-test

data:  Up$Adj.Close and Down$Adj.Close
t = 0.9136, df = 2062.2, p-value = 0.361
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.944983  8.082003
sample estimates:
mean of x mean of y 
 52.56812  49.99961 
```

### Up and Stable Class



```
> t.test(Up$Adj.Close, Stable$Adj.Close)

	Welch Two Sample t-test

data:  Up$Adj.Close and Stable$Adj.Close
t = 1.6353, df = 2103, p-value = 0.1021
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.8616611  9.5128398
sample estimates:
mean of x mean of y 
 52.56812  48.24253 
```

### Down and Stable Class



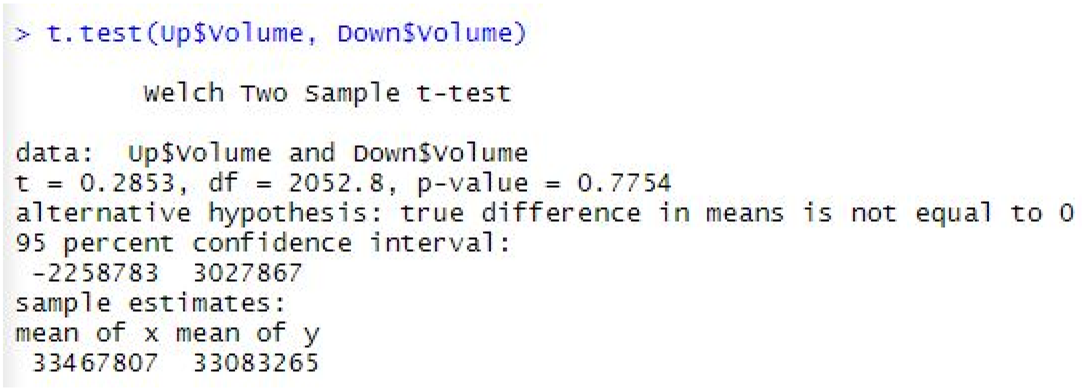
```
> t.test(Down$Adj.Close, Stable$Adj.Close)

        Welch Two Sample t-test

data:  Down$Adj.Close and Stable$Adj.Close
t = 0.66588, df = 1959, p-value = 0.5056
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.417943  6.932102
sample estimates:
mean of x mean of y 
 49.99961  48.24253 
```

When comparing the p-values for the histograms for *Figure 1.3e,* we fail to reject the null hypothesis and conclude that the true difference in means for all three classes in the Adjusted Closing Price feature is 0. That is, there is no significant difference and the feature does not discriminate by class. We obtain discriminatory values of .639, .898, and .494 for UpDown, UpStable, and DownStable. With values of .639 and .494, UpDown and DownStable's Adjusted Closing Price is clearly not discriminatory, and shares a very similar correlation. With a value of .898, Adjusted Closing Price for UpStable is not as discriminatory but still shares a similar correlation as we conclude that the true difference in means is 0.

We can further observe that these values are exactly identical to the values generated in the "Closing" features. From this observation, we can conclude that the histograms for closing and adjusted closing is identical.

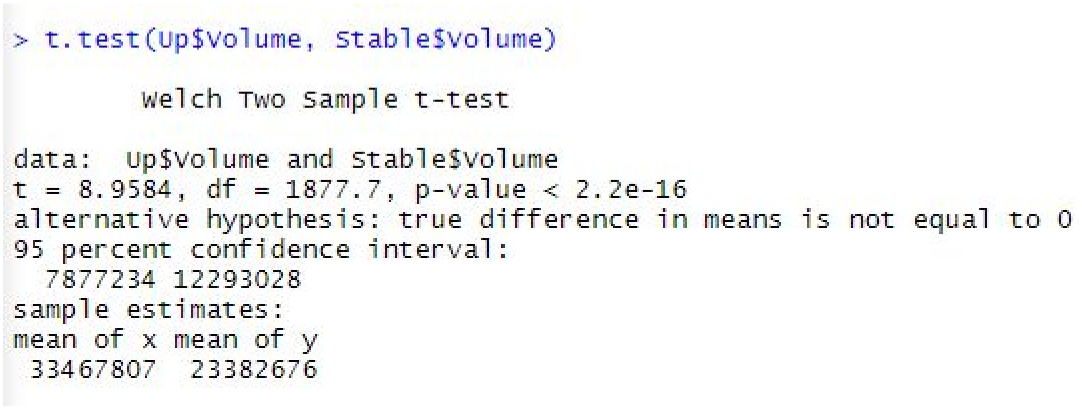
**Volume**

**Up and Down Class**



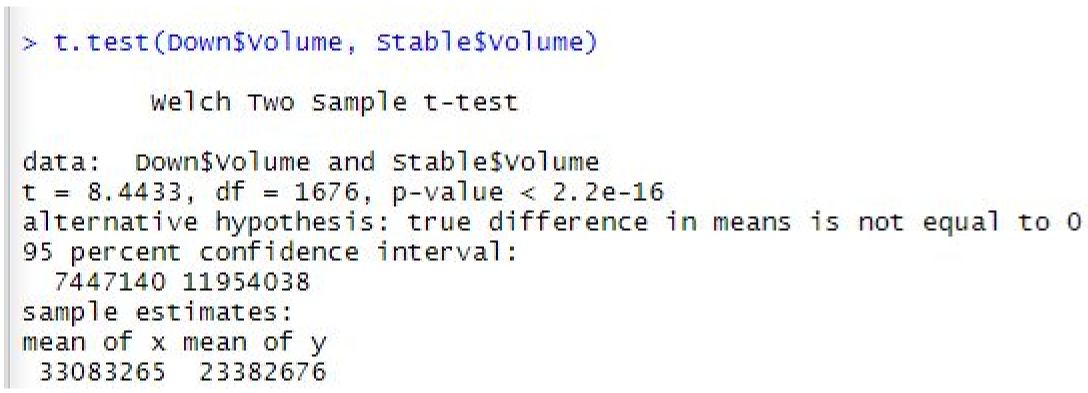
```
> t.test(Up$Volume, Down$Volume)

        Welch Two Sample t-test

data:  Up$Volume and Down$Volume
t = 0.2853, df = 2052.8, p-value = 0.7754
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2258783  3027867
sample estimates:
mean of x mean of y 
 33467807  33083265 
```

**Up and Stable Class**



```
> t.test(Up$Volume, Stable$Volume)

        Welch Two Sample t-test

data:  Up$Volume and Stable$Volume
t = 8.9584, df = 1877.7, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  7877234 12293028
sample estimates:
mean of x mean of y 
 33467807  23382676 
```

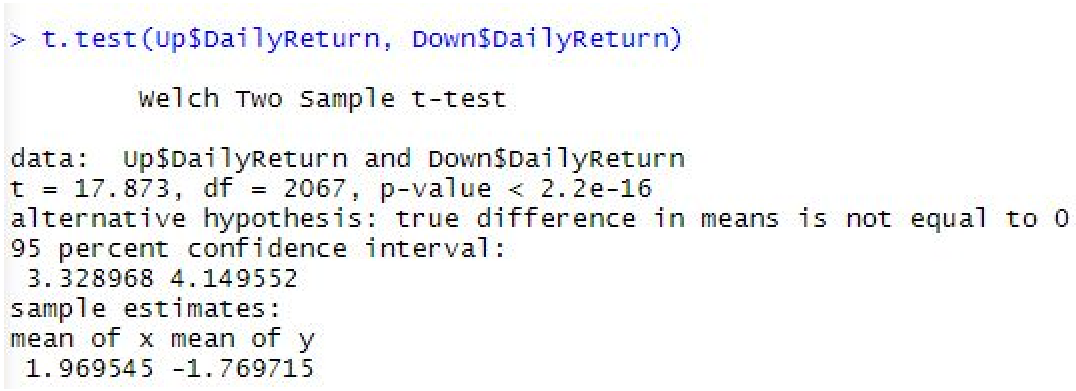
**Down and Stable Class**



```
> t.test(Down$Volume, Stable$Volume)

        Welch Two Sample t-test

data:  Down$Volume and Stable$Volume
t = 8.4433, df = 1676, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  7447140 11954038
sample estimates:
mean of x mean of y 
 33083265  23382676 
```

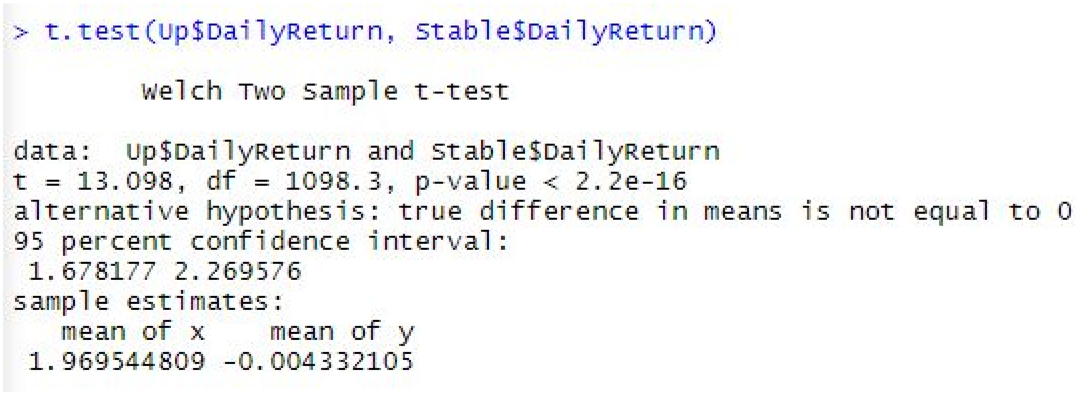
When comparing the p-values for the histograms in *Figure 1.3f,* we have a few different conclusions to be made. Firstly, when comparing volume in the Up and Down classes, we have a pvalue of .7754, so we fail to reject the null hypothesis and conclude that the true difference in means between the volume for up class and the volume for down class is 0. That is, we can expect the distributions to be similar, and the average amount of shares within a given daily for Up and Down days are generally the same. For Up and Stable, and Down and stable, we have a pvalue that is less than .05 and thus we can reject the null hypothesis and conclude that the true difference in means is not equal to 0. That is, the amount of shares on Stable days is significantly different than the amount of shares traded on Up or Down days. If we observe the means, we can see the average volume traded on Stable days is about 1000000 less than Up or Down days. Reflecting our findings, the discriminatory power for Up and Down in the volume feature is .23, indicating that the volume does not discriminate between Up or Down days. The discriminatory powers for the other two pairs is 1, meaning that the volume for those pairings are discriminated greatly based on the class. That is, we can expect less shares to be traded on days that are stable.

**Daily Return**
**Up and Down Class**



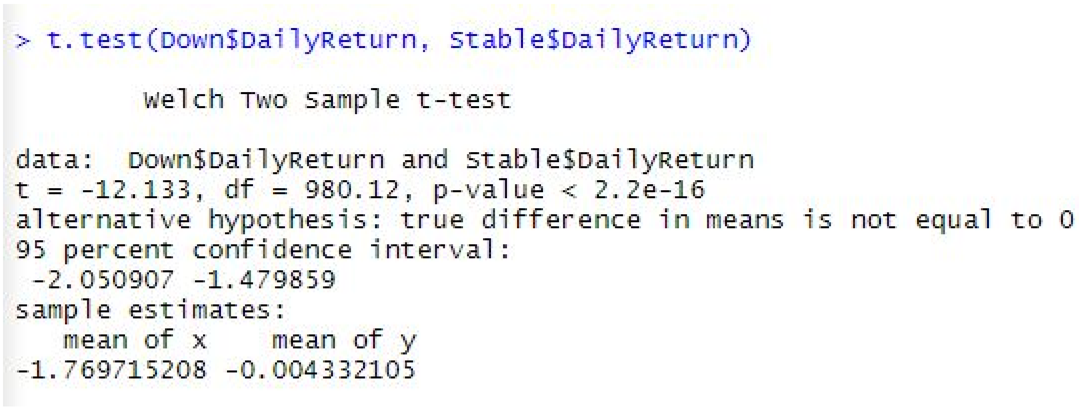
```
> t.test(Up$DailyReturn, Down$DailyReturn)

        Welch Two Sample t-test

data:  Up$DailyReturn and Down$DailyReturn
t = 17.873, df = 2067, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 3.328968 4.149552
sample estimates:
mean of x mean of y 
 1.969545 -1.769715 
```

**Up and Stable Class**



```
> t.test(Up$DailyReturn, Stable$DailyReturn)

	Welch Two Sample t-test

data:  Up$DailyReturn and Stable$DailyReturn
t = 13.098, df = 1098.3, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.678177 2.269576
sample estimates:
   mean of x    mean of y 
 1.969544809 -0.004332105 
```

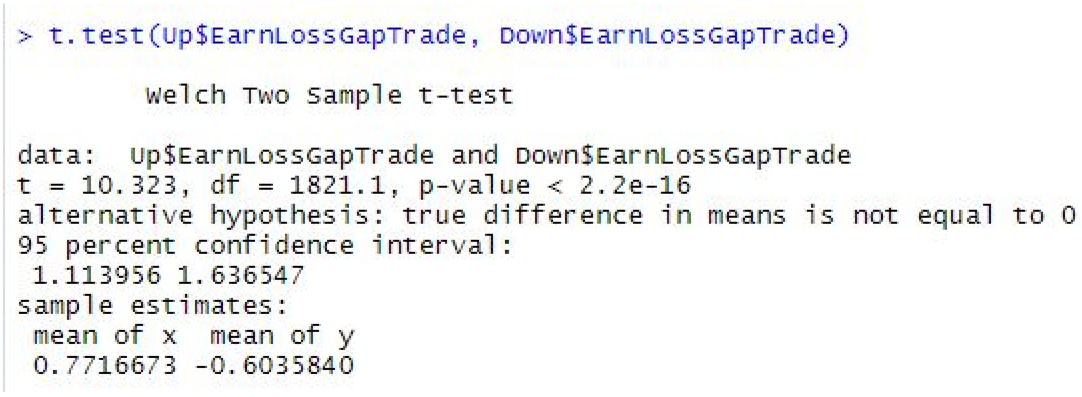
**Down and Stable Class**



```
> t.test(Down$DailyReturn, Stable$DailyReturn)

	Welch Two Sample t-test

data:  Down$DailyReturn and Stable$DailyReturn
t = -12.133, df = 980.12, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.050907 -1.479859
sample estimates:
   mean of x    mean of y 
-1.769715208 -0.004332105 
```

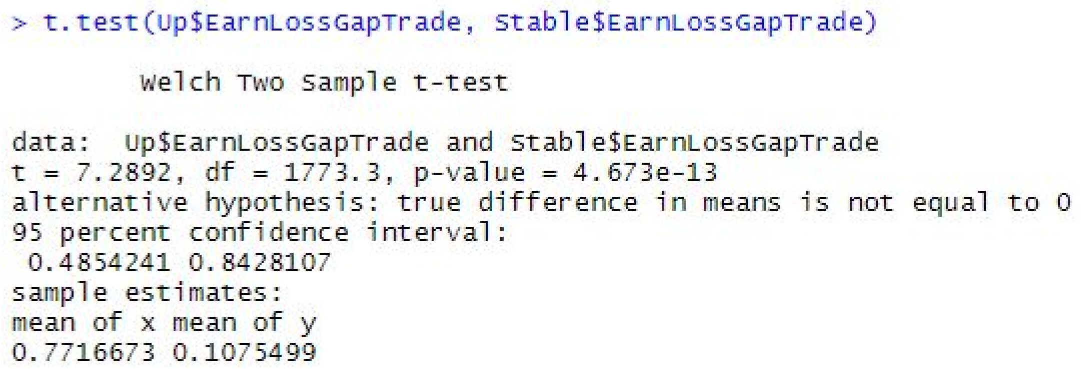
When comparing the histograms for *Figure 1.3g,* we make a single general conclusion. With pvalues of practically 0 for all 3 pairings for Daily Return, we reject the null hypothesis and conclude that the true difference in means is not equal to 0. That is, our discriminatory power is 1 and Daily Return is completely discriminatory and leads to differences in distributions by class. This is obvious in the histograms because they look nothing alike, and even more obvious when we observe the means of the classes for this feature and find that Up is always positive, Down is always negative, and Stable is extremely close to 0.

**Earn Loss Gap Trade**
**Up and Down Class**



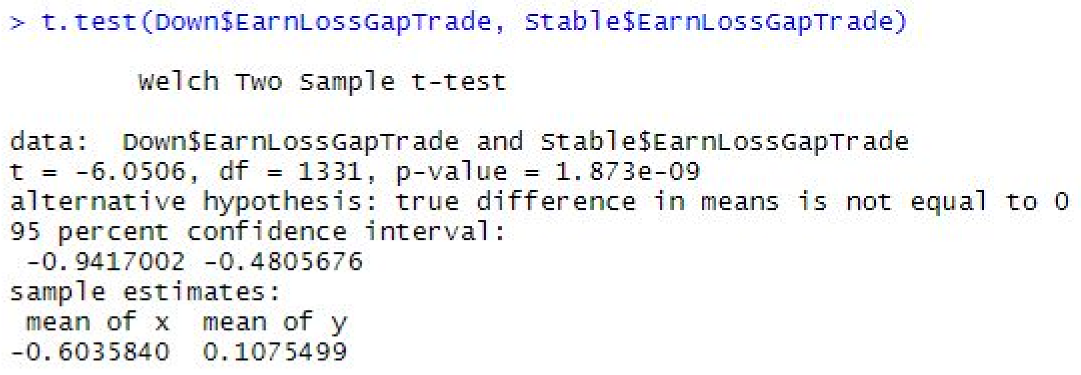
```
> t.test(Up$EarnLossGapTrade, Down$EarnLossGapTrade)

	Welch Two Sample t-test

data:  Up$EarnLossGapTrade and Down$EarnLossGapTrade
t = 10.323, df = 1821.1, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.113956 1.636547
sample estimates:
 mean of x  mean of y 
 0.7716673 -0.6035840 
```

**Up and Stable Class**



```
> t.test(Up$EarnLossGapTrade, Stable$EarnLossGapTrade)

        Welch Two Sample t-test

data:  Up$EarnLossGapTrade and Stable$EarnLossGapTrade
t = 7.2892, df = 1773.3, p-value = 4.673e-13
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.4854241 0.8428107
sample estimates:
mean of x mean of y
0.7716673 0.1075499
```

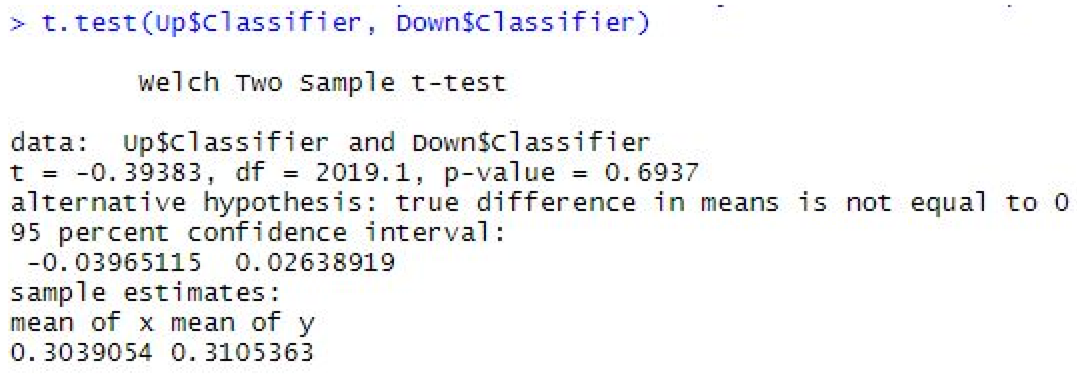
**Down and Stable Class**



```
> t.test(Down$EarnLossGapTrade, Stable$EarnLossGapTrade)

        Welch Two Sample t-test

data:  Down$EarnLossGapTrade and Stable$EarnLossGapTrade
t = -6.0506, df = 1331, p-value = 1.873e-09
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.9417002 -0.4805676
sample estimates:
 mean of x  mean of y
-0.6035840  0.1075499
```

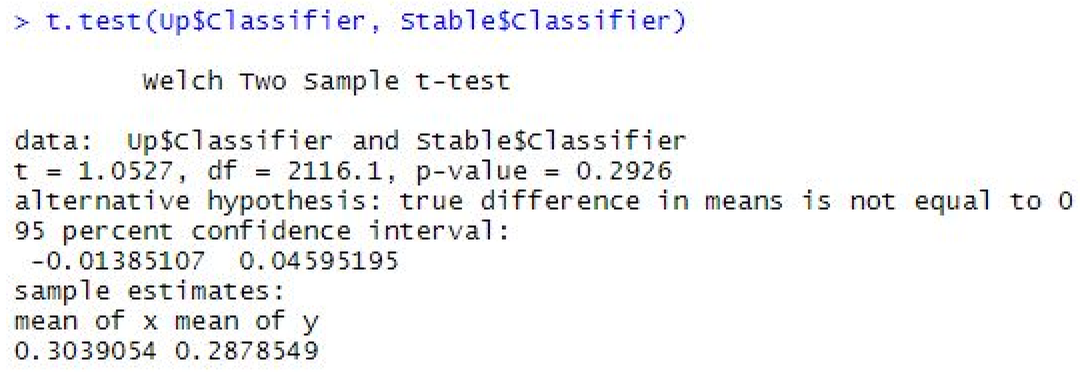
When comparing the histograms for *Figure 1.3h,* we make a single general conclusion. With pvalues of practically 0 for all 3 pairings for Earn Loss Gap Trade, we reject the null hypothesis and conclude that the true difference in means is not equal to 0. That is, our discriminatory power is 1 and Earn Loss Gap Trade is completely discriminatory and leads to differences in distributions by class. This is obvious in the histograms because they look nothing alike, and even more obvious when we observe the means of the classes for this feature and find the means supports our initial proposition that Up is positive, Down is negative, and Stable is close to 0.

**Threshold**
**Up and Down Class**



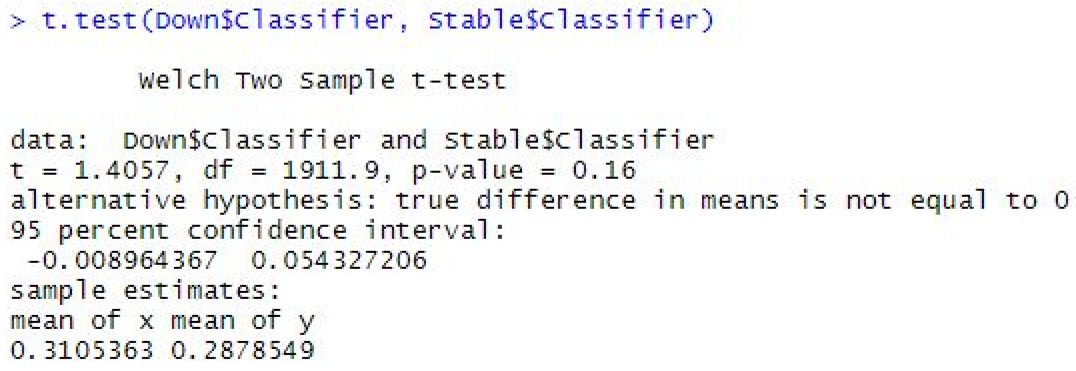
```
> t.test(Up$Classifier, Down$Classifier)

        Welch Two Sample t-test

data:  Up$Classifier and Down$Classifier
t = -0.39383, df = 2019.1, p-value = 0.6937
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.03965115  0.02638919
sample estimates:
mean of x mean of y
0.3039054 0.3105363
```

**Up and Stable Class**



```
> t.test(Up$Classifier, Stable$Classifier)

        Welch Two Sample t-test

data:  Up$Classifier and Stable$Classifier
t = 1.0527, df = 2116.1, p-value = 0.2926
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.01385107  0.04595195
sample estimates:
mean of x mean of y
0.3039054 0.2878549
```

**Down and Stable Class**



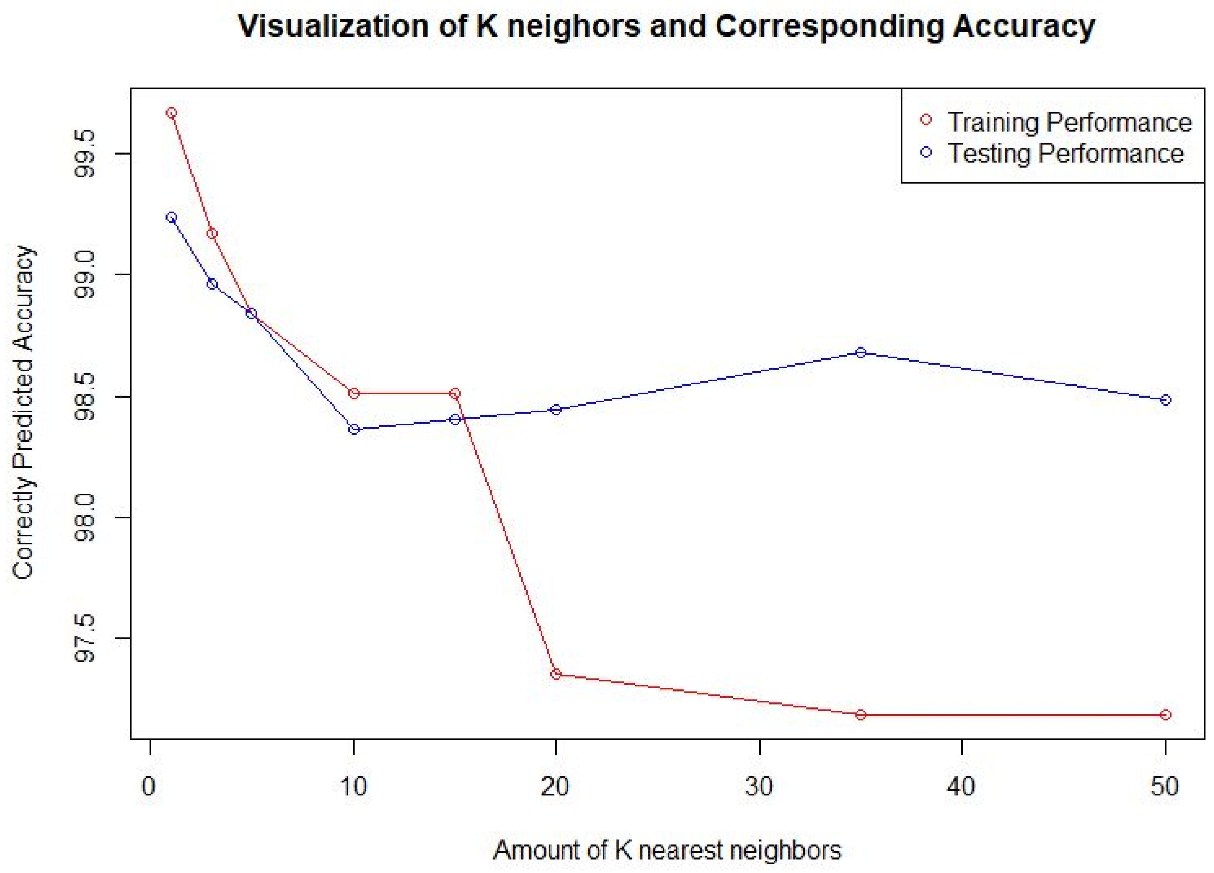
```
> t.test(Down$Classifier, Stable$Classifier)

        Welch Two Sample t-test

data:  Down$Classifier and Stable$Classifier
t = 1.4057, df = 1911.9, p-value = 0.16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.008964367  0.054327206
sample estimates:
mean of x mean of y
0.3105363 0.2878549
```

When comparing the histograms for *Figure 1.3i*, we make a general conclusion for all 3 pairings.Since the pvalues of each pairing is greater than .05, we fail to reject the null hypothesis and conclude that the true difference in means is equal to 0. That is, Threshold is not discriminatory and we can expect to see similar distributions in each of the histograms. With discriminatory powers of .3063, .7074, and .84, and mean values that are generally close to each other, we can conclude that the threshold value does not have an impact on classification. This becomes even more obvious when we realize that threshold is just the value of Opening Price scaled down, and as a result, will generally follow the same distribution as Opening Price.

**Step 3**
KNN is a supervised learning algorithm that relies on data to learn and classify different cases. KNN assumes similar things are near each other and classifies data points based on the K amount of “nearest” neighbors. That is, we find the surrounding data points and based on the classification of those data points, we can determine the classification of our own data point.

For our KNN algorithm, we split the data into 80% train and 20% test . Notice, before we do the split we have to set.seed() in order to ensure the same starting values are used throughout the experiment if we start with the same seed, set.seed(1). To maintain a similarly balanced set of classes within train and test set, we split each class by 80/20 splits. To achieve this we sample each class dataset for a training set and test set for that class and then,upon obtaining a perfectly balanced split, we combine the 3 class training and testing sets into one TRAININGSET, and TESTSET.
The following class splits are denoted as: Class(Training cases, Testing Cases)
Down(788, 188)
Stable(833,202)
Up(880,214)

For the following KNN evaluation, we start from 1 K value and gradually increase to test performance. K=1,3,5,10,15,20,35,50 are used for evaluation on both the TRAINING and TESTSET and obtain accuracies as plotted below.

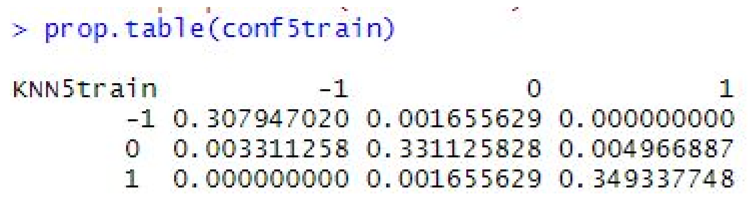


Based on the graph above, we can perceive that the highest accuracies start from K =1, and slowly diminish. Training set generally performs better than the testing set until around K = 15, where the training set takes an obvious dip in accuracy by 1%.

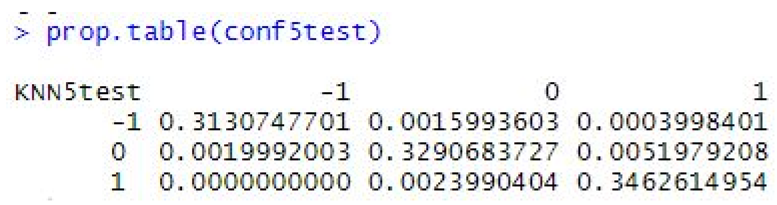
Clearly, the highest performancing K value for both training set and testing set is 1, with an accuracy of 99.66% for training set and 99.24% for the testing set. However, with observing the graphical results, we can see that 5 may be a better K value to use. That is, while it is less accurate overall, we will obtain consistency regardless of with set we use to train or test.
For K = 5, for both the training set and testing set, we obtain an accuracy of 98.84% of both sets. Thus we can say the “kbest” is 5, in terms of consistency.

**Confusion Matrices for K = 5**

Using kbest = 5, we obtain the 3x3 confusion matrix for KNN classification, namely “conf5train”, and “conf5test”. The confusion matrix provides us with percentages of correctly predicted cases by class. In the following confusion matrices, and as stated in Binary Recoding of Step 1:
-1: Down Class
0: Stable Class
1: Up Class



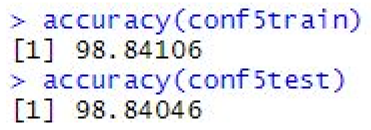
```
> prop.table(conf5train)

KNN5train          -1           0           1
       -1 0.307947020 0.001655629 0.000000000
       0  0.003311258 0.331125828 0.004966887
       1  0.000000000 0.001655629 0.349337748
```



```
> prop.table(conf5test)

KNN5test           -1            0            1
      -1 0.3130747701 0.0015993603 0.0003998401
      0  0.0019992003 0.3290683727 0.0051979208
      1  0.0000000000 0.0023990404 0.3462614954
```

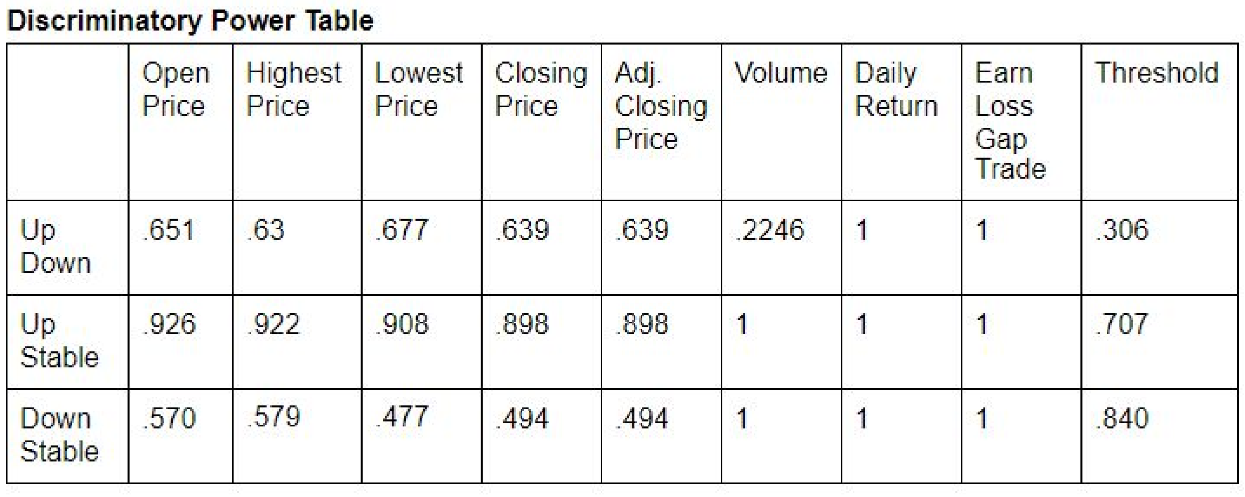
We can obtain the total accuracies with the following formula:
sum(diagonal(SET))/ Sum(SET)

That is, we take the proportion of correctly predicted classes and divide it by the entire amount of classes (1 in this case). Following this formula, we obtain the follow accuracies for conf5train and conf5test:



```
> accuracy(conf5train)
[1] 98.84106
> accuracy(conf5test)
[1] 98.84046
```

**Step 4:**
In determining the weights for each feature F for each class , we obtain each class pairing's discriminating power and compute the arithmetic mean. That is, when a class is MORE significant it is MORE discriminatory, and should have a higher weight in determining the class. A class with low discriminatory power does not contribute in obtaining the class as the distribution is relatively similar for that feature.



**Discriminatory Power Table**

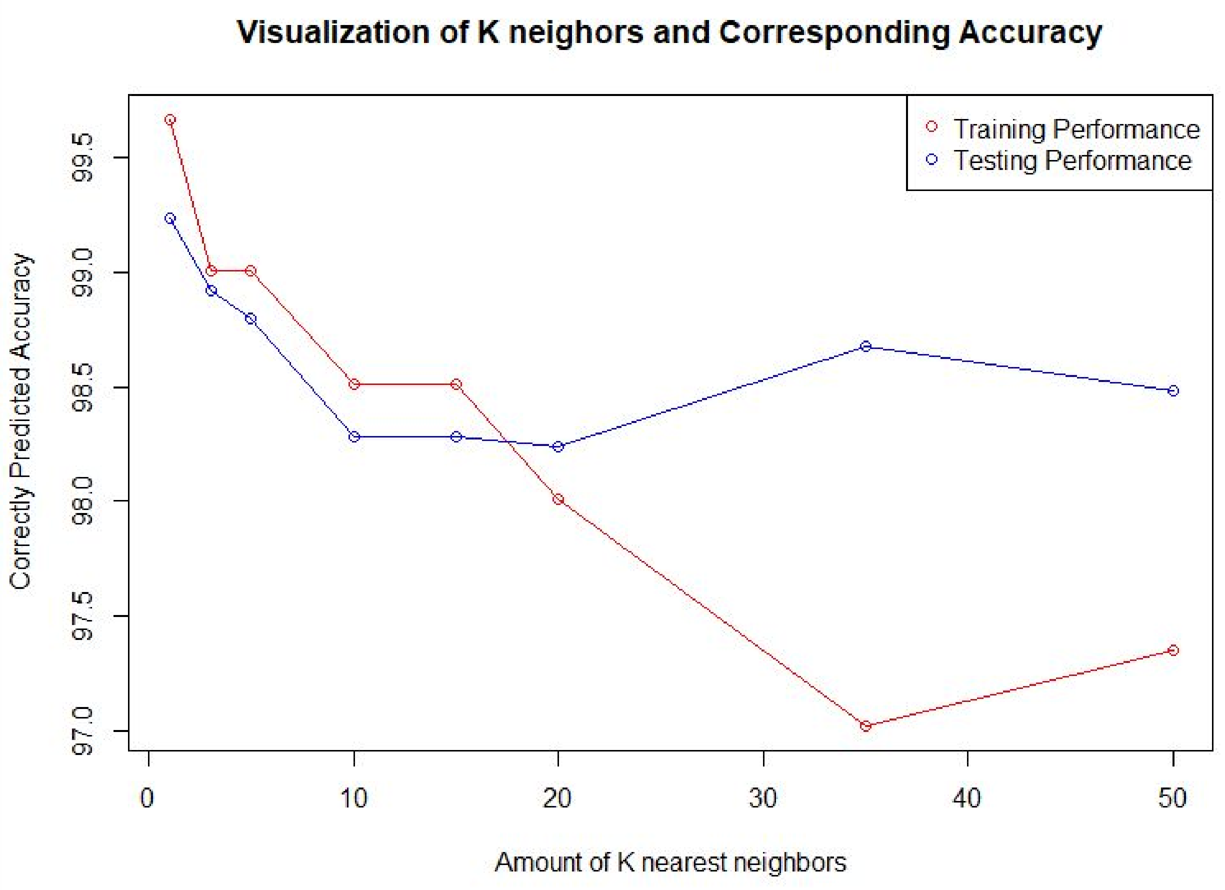
| | Open Price | Highest Price | Lowest Price | Closing Price | Adj. Closing Price | Volume | Daily Return | Earn Loss Gap Trade | Threshold |
|---|---|---|---|---|---|---|---|---|---|
| Up Down | .651 | .63 | .677 | .639 | .639 | .2246 | 1 | 1 | .306 |
| Up Stable | .926 | .922 | .908 | .898 | .898 | 1 | 1 | 1 | .707 |
| Down Stable | .570 | .579 | .477 | .494 | .494 | 1 | 1 | 1 | .840 |

The weights of each feature is computed by
Weight = ((DP(UpDown) + DP(UpStable) + DP(DownStable)/3) where DP = discriminating power

We obtain the following weights for each feature:

| Feature | Weight |
|---|---|
| Open Price | .716 |
| Highest Price | .710 |
| Lowest Price | .687 |
| Closing Price | .677 |
| Adjusted Closing Price | .677 |
| Volume | .742 |
| Daily Return | 1 |
| Earn Loss Gap Grade | 1 |
| Threshold | .618 |

Upon applying the weights to each of the features, we run the KNN classification for the same K values and obtain the following graph:



Graphically, we can see that the training set performed better than the testing set until K = 20. We can also recall that the lines converged at K = 5, but after apply our weights, they do not coverage at K = 5 anymore. That is because the accuracy for K = 5 improved to 99% from 98.84%. In this case, since we do not have consistency in training set vs testing set anymore, it may be better to utilize K = 1 as our Kbest, as the accuracies for both are higher. While they did not improve after applying our weights, the accuracies still stand at 99.67% for training set and 99.24% for the testing set.

## R Code

```
library(dplyr)
library(dslabs)
library(ISLR)
names <- c("Open","High", "Low", "Close", "AdjustedClose","Volume", "DailyReturn","ELGT","Classifier")

TSLA <- read.csv("C:/Users/Kevin/Desktop/fall 2020/6350/midterm 1/TSLA.csv")
for (i in 1:nrow(TSLA)) {
  if(i > 1) {
    TSLA$DailyReturn[i] <- TSLA$Open[i] - TSLA$Open[i - 1]
    TSLA$EarnLossGapTrade[i] <- TSLA$Open[i] - TSLA$Close[i-1]
    TSLA$Classifier[i] <- TSLA$Open[i - 1] * 0.006

    if (TSLA$DailyReturn[i] > TSLA$Classifier[i]) {
      TSLA$Class[i] <- 1
    }
```

```
      else if (TSLA$DailyReturn[i] < (TSLA$Classifier[i] * -1)) {
        TSLA$Class[i] <- -1
      }
      else {
        TSLA$Class[i] <- 0
      }
    }
}

table(TSLA$Class)

orgtablelength = nrow(TSLA)
count = 0
for (i in 1:orgtablelength) {
  if (i > 1) {
    if (TSLA$Class[i] == 0) {
      count = count + 1
      TSLA[(orgtablelength + count),] = TSLA[i,]
    }
  }

}

TSLA$DailyReturn[1] <- mean(TSLA$DailyReturn[2:nrow(TSLA)])
TSLA$EarnLossGapTrade[1] <- mean(TSLA$EarnLossGapTrade[2:nrow(TSLA)])
TSLA$Classifier[1] <- mean(TSLA$Classifier[2:nrow(TSLA)])
TSLA$Class[1] <- 0

MeanTable <- data.frame(matrix(ncol = 9, nrow = 3))
colnames(MeanTable) <- c("Open","High", "Low", "Close", "AdjustedClose","Volume", "DailyReturn","ELGT","Classifier")
StdDevTable <- data.frame(matrix(ncol = 9, nrow = 3))
colnames(StdDevTable) <- c("Open","High", "Low", "Close", "AdjustedClose","Volume", "DailyReturn","ELGT","Classifier")
SofFTable <- data.frame(matrix(ncol = 9, nrow = 3))
colnames(SofFTable) <- c("Open","High", "Low", "Close", "AdjustedClose","Volume", "DailyReturn","ELGT","Classifier")
DiscTable <- data.frame(matrix(ncol = 9, nrow = 3))
colnames(DiscTable) <- c("Open","High", "Low", "Close", "AdjustedClose","Volume", "DailyReturn","ELGT","Classifier")

Up <- data.frame(TSLA %>% filter(TSLA$Class == 1))
Down <- data.frame(TSLA %>% filter(TSLA$Class == -1 ))
Stable <- data.frame(TSLA %>% filter(TSLA$Class == 0))

par(mfrow = c(1,3))
hist(Up$Open,xlab = "Opening Price", main = "Histogram for Up Class Price")
hist(Down$Open,xlab = "Opening Price", main = "Histogram for Down Class Price")
hist(Stable$Open,xlab = "Opening Price", main = "Histogram for Stable Class Price")

hist(Up$High,xlab = "Day's Highest Price", main = "Histogram of Up Class Highest Price")
hist(Down$High,xlab = "Day's Highest Price", main = "Histogram of Down Class Highest Price")
hist(Stable$High,xlab = "Day's Highest Price", main = "Histogram of Stable Class Highest Price")

hist(Up$Low,xlab = "Day's Lowest Price", main = "Histogram for Up Class Lowest Price")
hist(Down$Low,xlab = "Day's Lowest Price", main = "Histogram for Down Class Lowest Price")
hist(Stable$Low,xlab = "Day's Lowest Price", main = "Histogram for Stable Class Lowest Price")

hist(Up$Close,xlab = "Day's Closing Price", main = "Histogram for Up Class Closing Price")
hist(Down$Close,xlab = "Day's Closing Price", main = "Histogram for Down Class Closing Price")
hist(Stable$Close,xlab = "Day's Closing Price", main = "Histogram for Stable Class Closing Price")

hist(Up$Adj.Close,xlab = "Day's Adjusted Closing Price", main = "Histogram for Up Class Adjusted Closing Price")
hist(Down$Adj.Close,xlab = "Day's Adjusted Closing Price", main = "Histogram for Down Class Adjusted Closing Price")
hist(Stable$Adj.Close,xlab = "Day's Adjusted Closing Price", main = "Histogram for Stable Class Adjusted Closing Price")

hist(Up$Volume * .001,xlab = "Day's Total Traded Volume(Scaled down by 1/1000)", main = "Histogram for Up Class Daily Volume")
hist(Down$Volume * .001,xlab = "Day's Total Traded Volume(Scaled down by 1/1000)", main = "Histogram for Down Class Daily Volume")
hist(Stable$Volume * .001,xlab = "Day's Total Traded Volume(Scaled down by 1/1000)", main = "Histogram for Stable Class Daily Volume")

hist(Up$DailyReturn ,xlab = "Daily Return", main = "Histogram for Up Class Gain or Loss")
hist(Down$DailyReturn ,xlab = "Daily Return", main = "Histogram for Down Class Gain or Loss")
hist(Stable$DailyReturn ,xlab = "Daily Return", main = "Histogram for Stable Class Gain or Loss")

hist(Up$EarnLossGapTrade, xlab = "Daily Difference in Gap Trading", main = "Histogram for Up Class Gap Trade Difference")
hist(Down$EarnLossGapTrade, xlab = "Daily Difference in Gap Trading", main = "Histogram for Down Class Gap Trade Difference")
hist(Stable$EarnLossGapTrade, xlab = "Daily Difference in Gap Trading", main = "Histogram for Stable Class Gap Trade Difference")

hist(Up$Classifier, xlab = "Daily Threshold(Opening Price * .6%)", main = "Histogram for Up Class Value used to classify")
hist(Down$Classifier, xlab = "Daily Threshold(Opening Price * .6%)", main = "Histogram for Down Class Value used to classify")
hist(Stable$Classifier, xlab = "Daily Threshold(Opening Price * .6%)", main = "Histogram for Stable Class Value used to classify")

prop.table(table(Up$Volume * .001< 50000))
prop.table(table(Down$Volume * .001< 50000))
prop.table(table(Stable$Volume * .001< 50000))
```

```
UpDown <- rbind(Up,Down)
UpStable <- rbind(Up,Stable)
DownStable <- rbind(Down,Stable)

t.test(Up$Open,Down$Open)
t.test(Up$Open,Stable$Open)
t.test(Down$Open,Stable$Open)

t.test(Up$High, Down$High)
t.test(Up$High, Stable$High)
t.test(Down$High,Stable$High)

t.test(Up$Low, Down$Low)
t.test(Up$Low, Stable$Low)
t.test(Down$Low,Stable$Low)

t.test(Up$Close, Down$Close)
t.test(Up$Close, Stable$Close)
t.test(Down$Close, Stable$Close)

t.test(Up$Adj.Close, Down$Adj.Close)
t.test(Up$Adj.Close, Stable$Adj.Close)
t.test(Down$Adj.Close, Stable$Adj.Close)

t.test(Up$Volume, Down$Volume)
t.test(Up$Volume, Stable$Volume)
t.test(Down$Volume, Stable$Volume)

t.test(Up$DailyReturn, Down$DailyReturn)
t.test(Up$DailyReturn, Stable$DailyReturn)
t.test(Down$DailyReturn, Stable$DailyReturn)

t.test(Up$EarnLossGapTrade, Down$EarnLossGapTrade)
t.test(Up$EarnLossGapTrade, Stable$EarnLossGapTrade)
t.test(Down$EarnLossGapTrade, Stable$EarnLossGapTrade)

t.test(Up$Classifier, Down$Classifier)
t.test(Up$Classifier, Stable$Classifier)
t.test(Down$Classifier, Stable$Classifier)

StTSLA <- data.frame(scale(TSLA[2:9]))
StTSLA['Class'] <- TSLA['Class']
Up <- data.frame(StTSLA %>% filter(StTSLA$Class == 1))
Down <- data.frame(StTSLA %>% filter(StTSLA$Class == -1 ))
Stable <- data.frame(StTSLA %>% filter(StTSLA$Class == 0))

set.seed(1)
Upindexes = sample(1:nrow(Up), size = 0.2*nrow(Up))
Uptest <- Up[indexes,]
Uptrain <- Up[-indexes,]

Downindexes = sample(1:nrow(Down), size = 0.2*nrow(Down))
Downtest <- Down[indexes,]
Downtrain <- Down[-indexes,]

Stableindexes = sample(1:nrow(Stable), size = 0.2*nrow(Stable))
Stabletest <- Stable[indexes,]
Stabletrain <- Stable[-indexes,]

TRAINSET <- bind_rows(Uptrain, Downtrain, Stabletrain)
TESTSET <- bind_rows(Uptest,Downtest,Stabletest)

table(TESTSET$Class)
table(TRAINSET$Class)

TESTSET <- na.omit(TESTSET)

library(caret)
library(class)
accuracy <- function(x) {sum(diag(x))/(sum(rowSums(x))) * 100}

KNN1train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 1)
conf1train <- table(KNN1train, TESTSET$Class)
KNN1test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 1)
conf1test <- table(KNN1test, TRAINSET$Class)
train1perf <- accuracy(conf1train)
test1perf <- accuracy(conf1test)

KNN3train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 3)
conf3train <- table(KNN3train, TESTSET$Class)
```

```
KNN3test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 3)
conf3test <- table(KNN3test, TRAINSET$Class)
train3perf <- accuracy(conf3train)
test3perf <- accuracy(conf3test)

KNN5train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 5)
conf5train <- table(KNN5train, TESTSET$Class)
KNN5test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 5)
conf5test <- table(KNN5test, TRAINSET$Class)
train5perf <- accuracy(conf5train)
test5perf <- accuracy(conf5test)

KNN10train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 10)
conf10train <- table(KNN10train, TESTSET$Class)
KNN10test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 10)
conf10test <- table(KNN10test, TRAINSET$Class)
train10perf <- accuracy(conf10train)
test10perf <- accuracy(conf10test)

KNN15train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 15)
conf15train <- table(KNN15train, TESTSET$Class)
KNN15test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 15)
conf15test <- table(KNN15test, TRAINSET$Class)
train15perf <- accuracy(conf15train)
test15perf <- accuracy(conf15test)

KNN20train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 20)
conf20train <- table(KNN20train, TESTSET$Class)
KNN20test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 20)
conf20test <- table(KNN20test, TRAINSET$Class)
train20perf <- accuracy(conf20train)
test20perf <- accuracy(conf20test)

KNN35train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 35)
conf35train <- table(KNN35train, TESTSET$Class)
KNN35test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 35)
conf35test <- table(KNN35test, TRAINSET$Class)
train35perf <- accuracy(conf35train)
test35perf <- accuracy(conf35test)

KNN50train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 50)
conf50train <- table(KNN50train, TESTSET$Class)
KNN50test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 50)
conf50test <- table(KNN50test, TRAINSET$Class)
train50perf <- accuracy(conf50train)
test50perf <- accuracy(conf50test)

par(mfrow=c(1,1))
TrainPlotPerf <- rbind(data.frame(c(1,3,5,10,15,20,35,50),(c(train1perf,train3perf,train5perf,train10perf,train15perf,train20perf,train35perf, train50perf))))
PlotPerf <- rbind(data.frame(c(1,3,5,10,15,20,35,50),(c(test1perf,test3perf,test5perf,test10perf,test15perf,test20perf, test35perf,test50perf))))

plot(TrainPlotPerf, type = "o", xlab = "Amount of K nearest neighbors", ylab = "Correctly Predicted Accuracy", col = "red", main = "Visualization of K neighors and Corresponding
Accuracy")
lines(PlotPerf, type = "o", col= "blue")
legend("topright", legend = c("Training Performance","Testing Performance"), col = c("red","blue"), pch = 1)

set.seed(132)
highesttestpref <- 0
testperf <- 0
bestK <- 0
for (i in 1:5) {
  KNN <- knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class , k =i)
  conftrain <- prop.table(table(KNN, TESTSET$Class))
  testperf <- accuracy(conftrain)
  if (testperf > highesttestpref ) {
    highesttestpref <- testperf
    bestK <- i
  }
}
bestK

highesttestpref <- 0
testperf <- 0
bestK <- 0
for (i in 1:5) {
  KNN <- knn(train = TESTSET, test = TRAINSET, cl = TESTSET$Class , k =i)
  conftrain <- prop.table(table(KNN, TRAINSET$Class))
  testperf <- accuracy(conftrain)
  if (testperf > highesttestpref ) {
    highesttestpref <- testperf
    bestK <- i
  }
}
bestK

prop.table(conf1train)
accuracy(conf1train)
accuracy(conf1test)

prop.table(conf5train)
prop.table(conf5test)
accuracy(conf5train)
accuracy(conf5test)

TSLA$Open <- TSLA$Open * .716
TSLA$High <- TSLA$High * .710
TSLA$Low <- TSLA$Low * .687
TSLA$Close <- TSLA$Close * .677
TSLA$Adj.Close <- TSLA$Adj.Close * .677
TSLA$Volume <- TSLA$Volume * .742
TSLA$Classifier <- TSLA$Classifier * .618

StTSLA <- data.frame(scale(TSLA[2:9]))
StTSLA['Class'] <- TSLA['Class']
Up <- data.frame(StTSLA %>% filter(StTSLA$Class == 1))
Down <- data.frame(StTSLA %>% filter(StTSLA$Class == -1 ))
Stable <- data.frame(StTSLA %>% filter(StTSLA$Class == 0))

set.seed(1)
Upindexes = sample(1:nrow(Up), size = 0.2*nrow(Up))
Uptest <- Up[indexes,]
Uptrain <- Up[-indexes,]

Downindexes = sample(1:nrow(Down), size = 0.2*nrow(Down))
Downtest <- Down[indexes,]
Downtrain <- Down[-indexes,]

Stableindexes = sample(1:nrow(Stable), size = 0.2*nrow(Stable))
Stabletest <- Stable[indexes,]
Stabletrain <- Stable[-indexes,]

TRAINSET <- bind_rows(Uptrain, Downtrain, Stabletrain)
TESTSET <- bind_rows(Uptest,Downtest,Stabletest)
TESTSET <- na.omit(TESTSET)

KNN1train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 1)
conf1train <- table(KNN1train, TESTSET$Class)
KNN1test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 1)
conf1test <- table(KNN1test, TRAINSET$Class)
train1perf <- accuracy(conf1train)
test1perf <- accuracy(conf1test)

KNN3train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 3)
conf3train <- table(KNN3train, TESTSET$Class)
KNN3test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 3)
conf3test <- table(KNN3test, TRAINSET$Class)
train3perf <- accuracy(conf3train)
test3perf <- accuracy(conf3test)

KNN5train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 5)
conf5train <- table(KNN5train, TESTSET$Class)
KNN5test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 5)
conf5test <- table(KNN5test, TRAINSET$Class)
train5perf <- accuracy(conf5train)
test5perf <- accuracy(conf5test)

KNN10train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 10)
conf10train <- table(KNN10train, TESTSET$Class)
KNN10test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 10)
conf10test <- table(KNN10test, TRAINSET$Class)
train10perf <- accuracy(conf10train)
test10perf <- accuracy(conf10test)

KNN15train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 15)
conf15train <- table(KNN15train, TESTSET$Class)
KNN15test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 15)
conf15test <- table(KNN15test, TRAINSET$Class)
train15perf <- accuracy(conf15train)
test15perf <- accuracy(conf15test)

KNN20train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 20)
conf20train <- table(KNN20train, TESTSET$Class)
```

```
KNN20test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 20)
conf20test <- table(KNN20test, TRAINSET$Class)
train20perf <- accuracy(conf20train)
test20perf <- accuracy(conf20test)

KNN35train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 35)
conf35train <- table(KNN35train, TESTSET$Class)
KNN35test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 35)
conf35test <- table(KNN35test, TRAINSET$Class)
train35perf <- accuracy(conf35train)
test35perf <- accuracy(conf35test)

KNN50train <-knn(train = TRAINSET, test = TESTSET, cl = TRAINSET$Class, k = 50)
conf50train <- table(KNN50train, TESTSET$Class)
KNN50test <- knn(train = TESTSET, test= TRAINSET, cl = TESTSET$Class, k = 50)
conf50test <- table(KNN50test, TRAINSET$Class)
train50perf <- accuracy(conf50train)
test50perf <- accuracy(conf50test)

par(mfrow=c(1,1))
TrainPlotPerf <- rbind(data.frame(c(1,3,5,10,15,20,35,50),(c(train1perf,train3perf,train5perf,train10perf,train15perf,train20perf,train35perf, train50perf))))
PlotPerf <- rbind(data.frame(c(1,3,5,10,15,20,35,50),(c(test1perf,test3perf,test5perf,test10perf,test15perf,test20perf, test35perf,test50perf))))

plot(TrainPlotPerf, type = "o", xlab = "Amount of K nearest neighbors", ylab = "Correctly Predicted Accuracy", col = "red", main = "Visualization of K neighors and Corresponding
Accuracy")
lines(PlotPerf, type = "o", col= "blue")
legend("topright", legend = c("Training Performance","Testing Performance"), col = c("red","blue"), pch = 1)
```