Math 6350 - Statistical Learning and Data Mining

FINAL

JIA YU

Introduction

In this report, the dataset containing daily stock price fluctuations for Tesla will be explored. Tesla Inc. designs and manufactures electric vehicles globally. 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 different methods in k-means, random forest, and SVM to analyze the market based on the movement of the previous day.

The data used for this report can be downloaded from the following link:

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

Below we describe the different features from the dataset, and additional features derived:

Features description and meaning:

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 factored as 1 for Up, 2 for Down, and 3 for Stable

IClass <- The previous day’s class factored as 1 for Up, 2 for Down, and 3 for Stable

IMovement <- The previous day’s daily return

IClassifier <- The previous day’s threshold

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: 12

Number of classes: 3

The breakdown of our classes are:

Class 1 (Up): 1094 cases

Class 2 (Down): 976 cases

Class 3 (Stable): 1035 cases

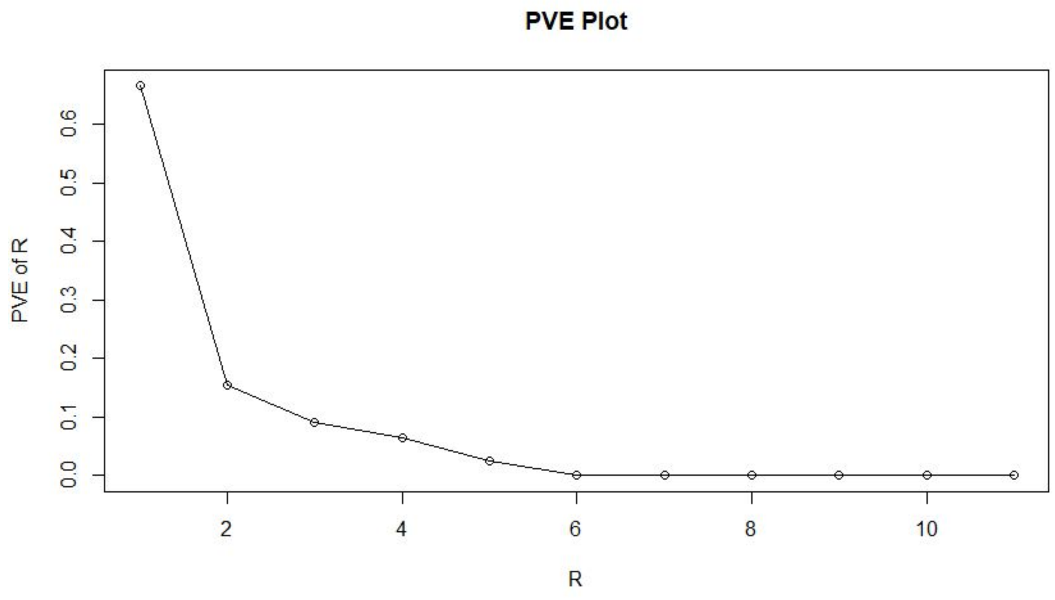
Question 1:

The data is standardized with the use of the scale() function and assigned to SDATA.

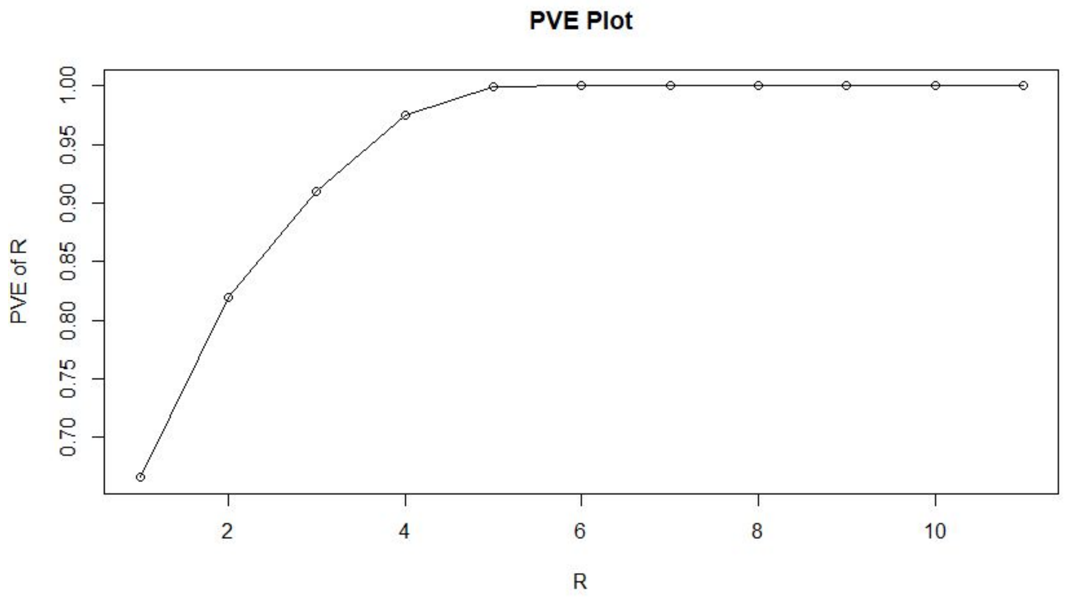
Principal Component Analysis is then used on SDATA to obtain our principal components.

The eigenvectors and eigenvalues of a covariance (or correlation) matrix represent the "core" of a PCA: The eigenvectors determine the directions of the new feature space, and the eigenvalues determine their magnitude. In other words, the eigenvalues explain the variance of the data along the new feature axis. Eigenvalues are simply the coefficients attached to eigenvectors, which give the amount of variance carried in each Principal Component. The goal of a PCA is to reduce the dimensionality of feature spaces by projecting it onto smaller subspace, these eigenvectors will form the axis, but they only define the directions of the new axis, to decide which eigenvector(s) can dropped without losing too much information. In other words, we are attempting to reduce the dimensionality while keeping as much of the variation as possible.

A percent variance is the change in an account during a period of from one period to the next expressed as a ratio. In other words, it shows the increase or decrease in an account over time as a percentage of the total account value. The higher the percentage of variance a proposed model manages to explain, the more valid the model seems to be. Below, we display the percentage of variance explained by each principal component



Our PVE plot expressed the percentage of variance explained for the given R, denoted Lr, in which our highest Lr values are closer to 0. Below, we observe our increasing function of cumulative PVE plot, in which we can clearly observe that the majority of the variance within our dataset occurs within the first 5 or so R values.

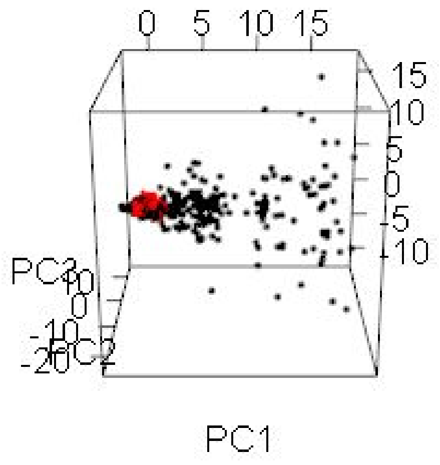


We further explore the eigenvalues associated with each principal component and examine a clear pattern in the percentage of variance explained and the eigenvalue itself.

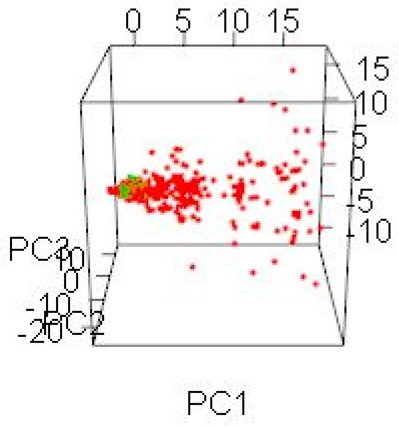
First 3 eigenvalues: 7.33, 1.69, 1.00

First 3 eigenvalues as a percentage of total variance: 66.58%, 15.35%, 9.08%

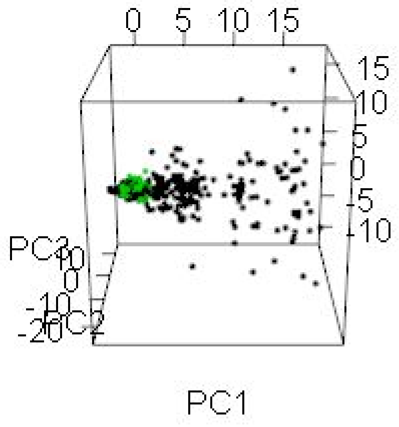
We then plot the 3D projection of the first 3 PCA eigenvectors by class below:



In the above 3D plot, we observe two classes, Up and Down on three PCA eigenvectors with the highest variability. In that sense, the three eigenvectors represent about 90% of the projection. We can see that there is low separability between the classes on lower values for PC1, as the data points tend to cluster between 0-5 on the PC1 scale.



In the above 3D plot, we observe two classes, Down and Stable on three PCA eigenvectors with the highest variability. In that sense, the three eigenvectors represent about 90% of the projection. Much like our projection for Up and Down, we can see that there is low separability between the classes on lower values for PC1, as the data points tend to cluster between 0-5 on the PC1 scale.



Finally, in the above 3D plot, we observe Down and Stable on three PCA eigenvectors with the highest variability. In that sense, the three eigenvectors represent about 90% of the projection. We can see that there is low separability between the classes on lower values for PC1, as the data points tend to cluster between 0-5 on the PC1 scale.

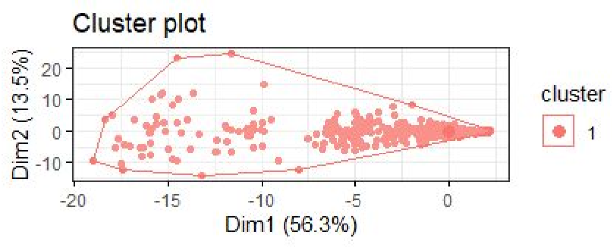
Later in the report, we will see that these PCA projections support the clustering we get from k-means.

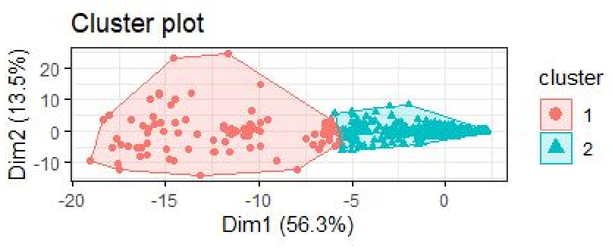
Question 2

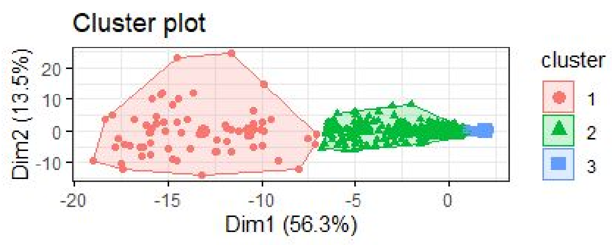
K-means clustering is a method used for clustering analysis, especially in data mining and statistics. It aims to partition a set of observations into a number of clusters (k), resulting in the partitioning of the data into k clusters. It can be considered a method of finding out which group a certain object really belongs to.

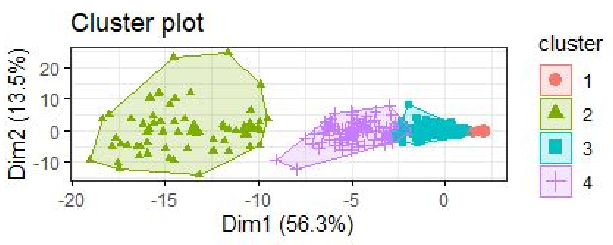
The K-means algorithm divides a set of samples into disjoint clusters , each described by the mean of the samples in the cluster. The means are commonly called the cluster “centroids”. The K-means algorithm aims to choose centroids that minimise the inertia, or within-cluster sum-of-squares criterion

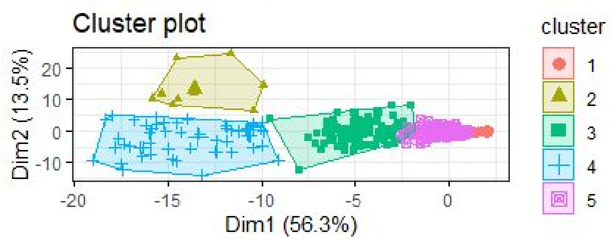
In attempting to find our best number of clusters, we apply k-means clustering on our standardized features from k =1 to k = 10 resulting the following disjointed clusters.

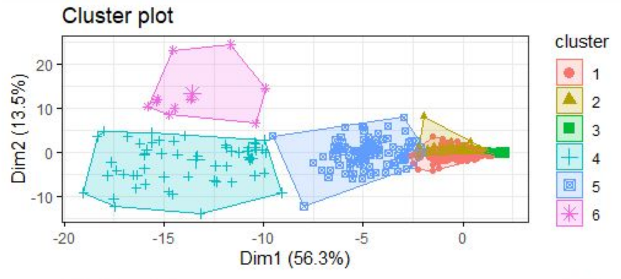

Cluster plot
Dim2 (13.5%)
Dim1 (56.3%)
cluster
1

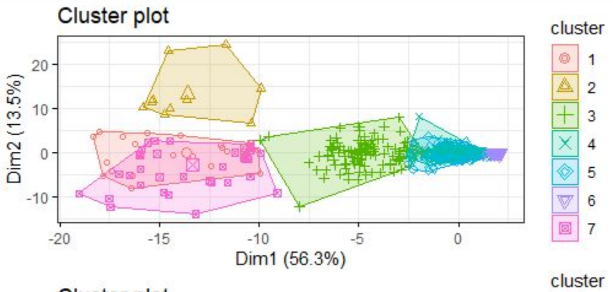

Cluster plot
Dim2 (13.5%)
Dim1 (56.3%)
cluster
1
2

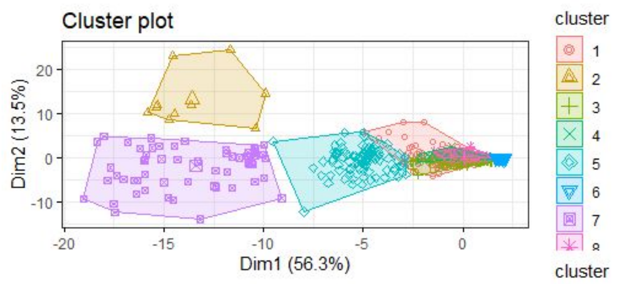

Cluster plot
Dim2 (13.5%)
Dim1 (56.3%)
cluster
1
2
3

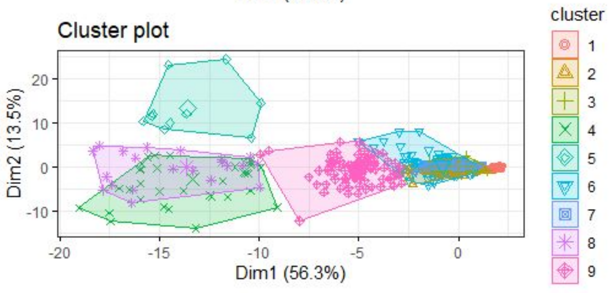

Cluster plot
Dim2 (13.5%)
Dim1 (56.3%)
cluster
1
2
3
4

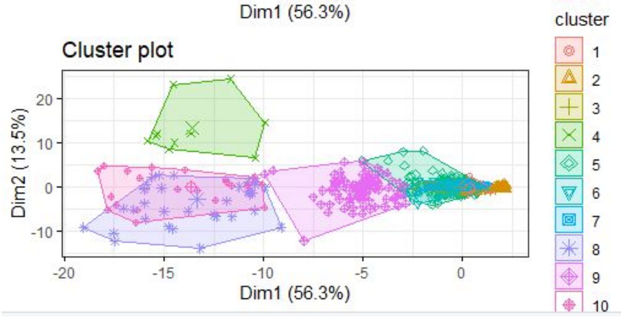

Cluster plot
Dim2 (13.5%)
Dim1 (56.3%)
cluster
1
2
3
4
5


Cluster plot
Dim2 (13.5%)
Dim1 (56.3%)
cluster
1
2
3
4
5
6


Cluster plot
Dim2 (13.5%)
Dim1 (56.3%)
cluster
1
2
3
4
5
6
7


Cluster plot
Dim2 (13.5%)
Dim1 (56.3%)
cluster
1
2
3
4
5
6
7
8


Cluster plot
Dim2 (13.5%)
Dim1 (56.3%)
cluster
1
2
3
4
5
6
7
8
9


Cluster plot
Dim2 (13.5%)
Dim1 (56.3%)
cluster
1
2
3
4
5
6
7
8
9
10

Having one cluster virtual makes no sense because we will not be able to differentiate the different classes.

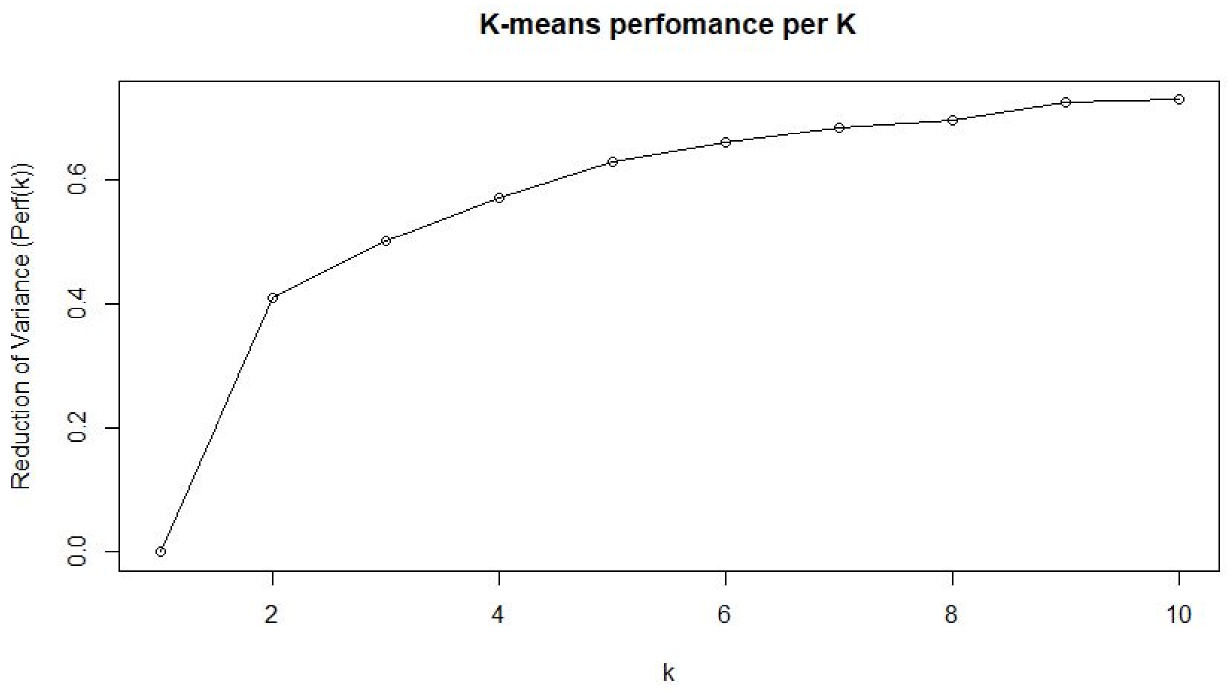
Having two clusters we can see the partitioning of the data set into non-overlapping coordinates in a 2d plot which is good but we want to see the best number of clusters so we keep partitioning.

Clusters = 3 looks good even though there seems to be some overlapping this seems like the best cluster especially knowing that there are 3 different groups but again we keep going.

Clusters = 4 is now starting to show a lot of overlap on the 2-d plot. But in general the more cluster we have the less the dispersion will be so we keep going to find the right balance.

Cluster 5-10 clearly shows a lot of overlapping, but that is expected as we are attempting to plot a 400 dimensional cluster on a 2 dimensional plane.

We further explore the performance of each k number of clusters to help determine the best number of clusters for our model.

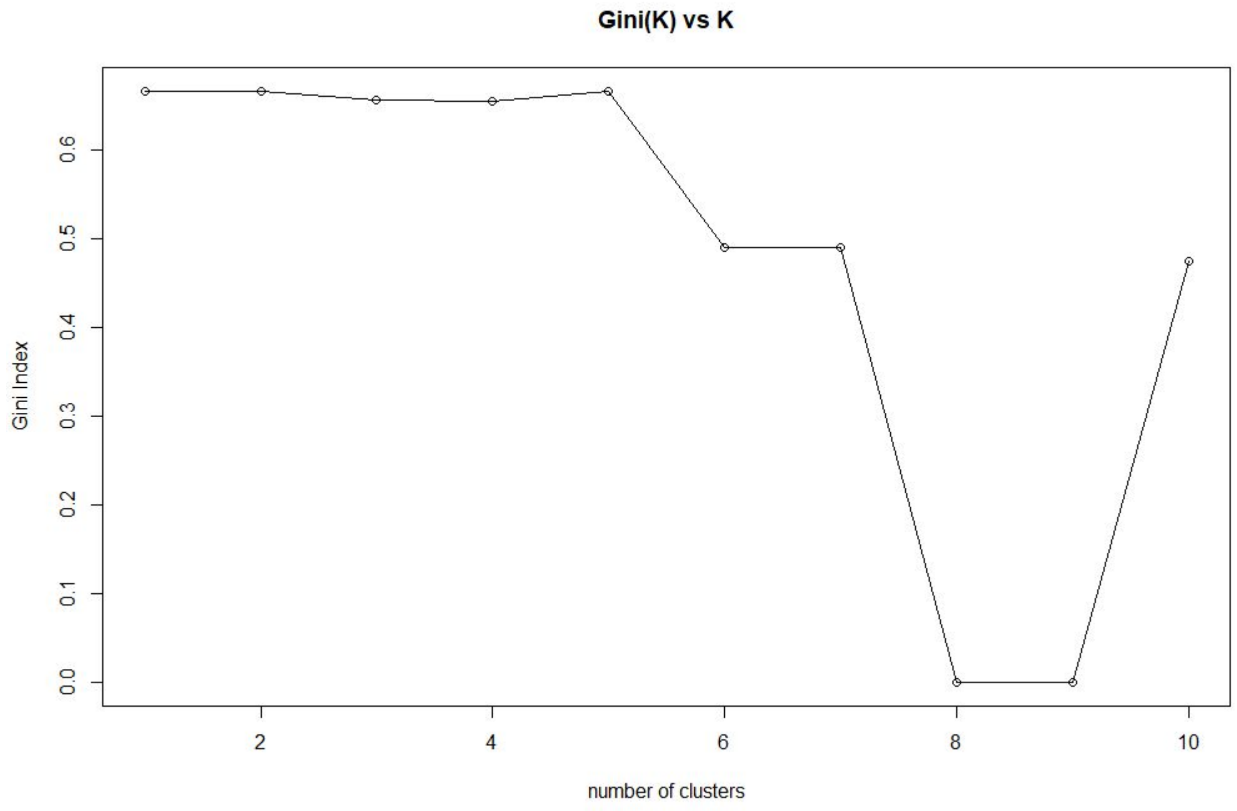


One method to validate the number of clusters is the elbow method. The idea of the elbow method is to run k-means clustering on the dataset for a range of values of k, and for each value of k calculate the sum of squared errors (SSE). We then plot a line chart of the SSE for each value of k. If the line chart looks like an arm, then the "elbow" on the arm is the value of k that is the best. The idea is that we want a small SSE, but that the SSE tends to decrease toward 0 as we increase k. Our goal is to choose a small value of k that still has a low SSE, and the elbow usually represents where we start to have diminishing returns by increasing k. In referencing our plotted k graph against performance, we identify our best k as 6.

To further select our best k value, we explore our gini index for each k number of clusters below.

Gini index or Gini impurity measures the degree or probability of a particular variable being wrongly classified when it is randomly chosen. If all the elements belong to a single class, then it can be called pure. The degree of Gini index varies between 0 and 1, where 0 denotes that all elements belong to a certain class or if there exists only one class, and 1 denotes that the elements are randomly distributed across various classes. A Gini Index of 0.5 denotes equally distributed elements into some classes.

Gini Impurity is a measurement of the likelihood of an incorrect classification of a new instance of a random variable, if that new instance were randomly classified according to the distribution of class labels from the data set. We aim to construct clusters with smaller impurity.

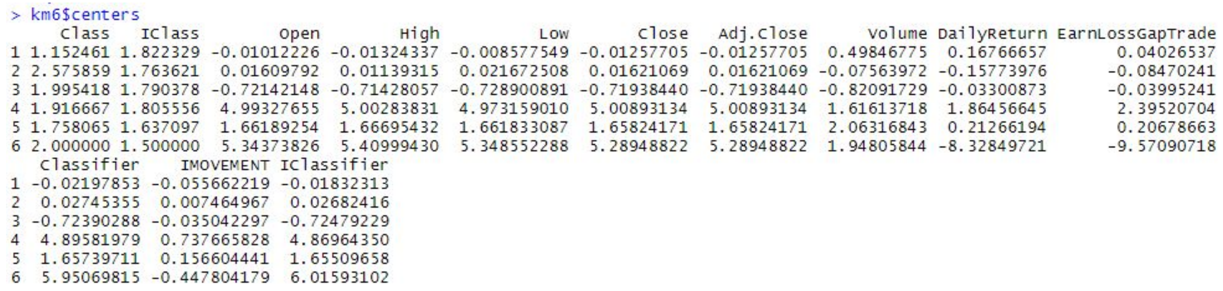


Clearly, we observe our best gini index that is 0 comes at k = 6, so we select 6 as our best k.

Question 3

The K-means algorithm divides a set of samples into disjoint clusters , each described by the mean of the samples in the cluster. The means are commonly called the cluster "centroids". The K-means algorithm aims to choose centroids that minimise the inertia, or within-cluster sum-of-squares criterion

A centroid is a data point at the center of a cluster. The resulting classifier is used to classify the data and thereby produce an initial randomized set of clusters. Each centroid is thereafter set to the arithmetic mean of the cluster it defines. In the section below, we explore the prominent features of k-means clustering: centers, size, dispersion, gini impurity, and frequencies.



```
> km6$centers
     Class   IClass        Open        High          Low       Close   Adj.Close      Volume DailyReturn EarnLossGapTrade
1 1.152461 1.822329 -0.01012226 -0.01324337 -0.008577549 -0.01257705 -0.01257705  0.49846775  0.16766657       0.04026537
2 2.575859 1.763621  0.01609792  0.01139315  0.021672508  0.01621069  0.01621069 -0.07563972 -0.15773976      -0.08470241
3 1.995418 1.790378 -0.72142148 -0.71428057 -0.728900891 -0.71938440 -0.71938440 -0.82091729 -0.03300873      -0.03995241
4 1.916667 1.805556  4.99327655  5.00283831  4.973159010  5.00893134  5.00893134  1.61613718  1.86456645       2.39520704
5 1.758065 1.637097  1.66189254  1.66695432  1.661833087  1.65824171  1.65824171  2.06316843  0.21266194       0.20678663
6 2.000000 1.500000  5.34373826  5.40999430  5.348552288  5.28948822  5.28948822  1.94805844 -8.32849721      -9.57090718
   Classifier    IMOVEMENT IClassifier
1 -0.02197853 -0.055662219 -0.01832313
2  0.02745355  0.007464967  0.02682416
3 -0.72390288 -0.035042297 -0.72479229
4  4.89581979  0.737665828  4.86964350
5  1.65739711  0.156604441  1.65509658
6  5.95069815 -0.447804179  6.01593102
```

Size

Cluster	1	2	3	4	5	6
Size	833	1193	873	72	124	10

We observe the size of each cluster, and identify our biggest cluster as 4. We then observe the biggest cluster to distinguish if there is a distinct class that clearly holds a higher proportion in that cluster.

Dispersion

Cluster	1	2	3	4	5	6
Dispersion	2250.42	1888.75	1158.83	4599.76	2397.36	675.57

Dispersion reflects the compactness of a cluster when employed at the intra-cluster level and reveals the separation when measured at the inter-cluster level. So it results in a vector with a number for each cluster. One expects this ratio to be as low as possible for each cluster, since we would like to have homogeneity within the clusters. We can observe that cluster 3, the most balanced class has a low dispersion score.

Gini

Cluster	1	2	3	4	5	6
Gini	.26	.49	.67	.63	.64	0

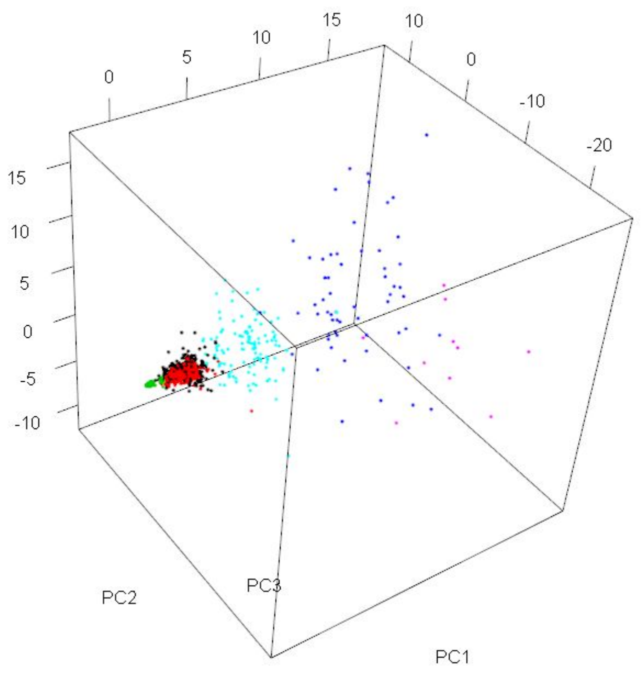
We obtained the above gini values for each cluster and observed that most of the clusters have decent gini values with the exception of cluster 1 and cluster 6. Upon further investigation, we recognize the classes with lower gini scores are more unbalanced. This notion is supported by the frequency table in the next section.

Impurity is the sum of the gini index for every cluster. We obtain an impurity score of 2.68.

Frequency

Class/Cluster	Up	Down	Stable
1	708	123	2
2	0	506	687
3	300	277	296
4	32	14	26
5	54	46	24
6	0	10	0

On our frequency table, we observe what we expect in relation to our gini indexes for each cluster. That is, Cluster 1 is extremely unbalanced so we expect a low gini score, followed by cluster 2, with cluster 3,4,5 being relatively more balanced so we observe high gini scores. Finally cluster 6 is not balanced whatsoever, so we observe a gini score of 0.



Lastly, we display the 3D projection of our kmeans for 6 clusters, on the three most prominent PCA eigenvectors. That is, we can observe a large number of green, red, and black data points; which is reflected by our size chart.

Question 4

In preparation for further classification methods, we obtain training and test sets for each class within our dataset.

The sample split used for deriving the training and test set is 80/20% split with 80% of the random sampling being for training purposes and 20% of the random sampling being for testing purposes. After obtaining the training and test sets for each class we observe the dimensions of each:

Class Up Train size: 876 rows

Class Down Train size: 781 rows

Class Stable Train size: 828 rows

Class Up Test Size: 218 rows

Class Down Test Size: 195 rows

Class Stable Test Size: 207 rows

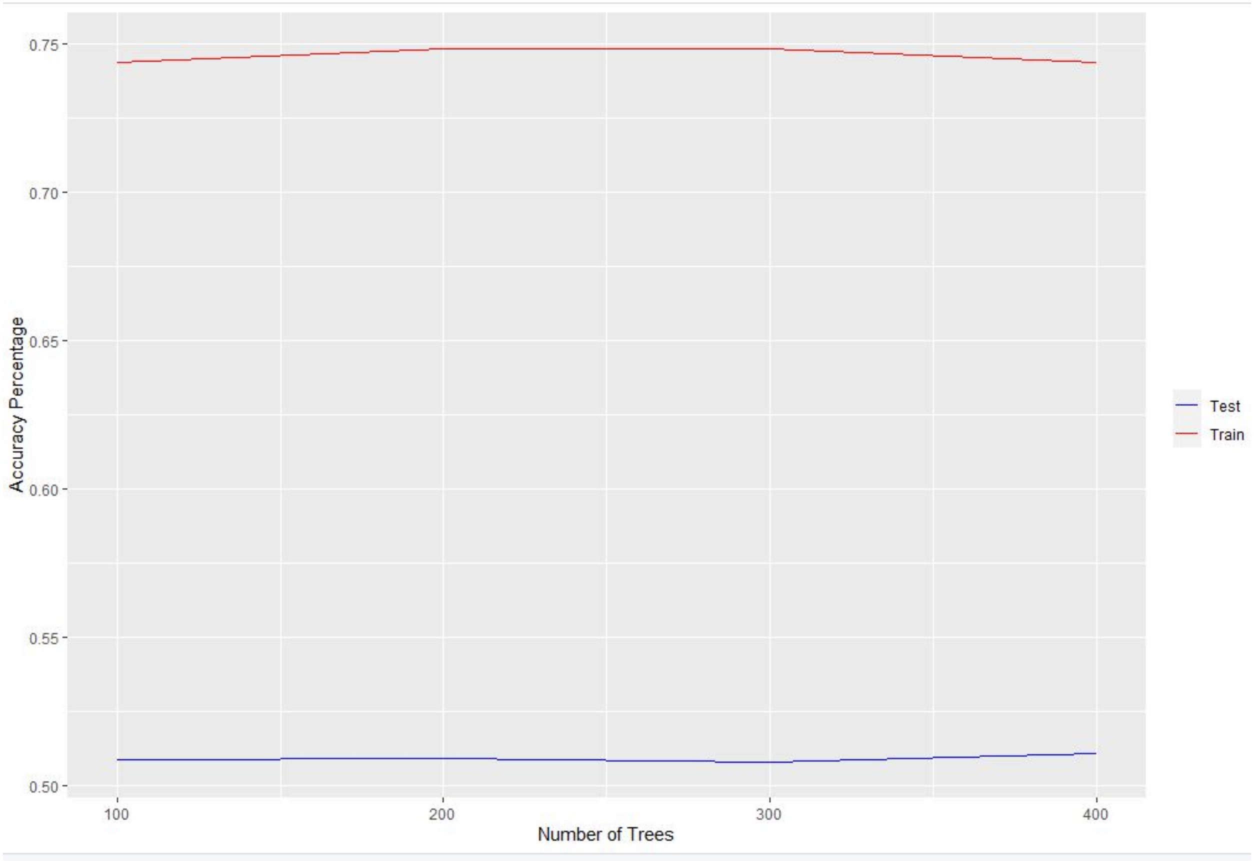
Furthermore, each class is relatively balanced, no cloning needs to be done on the training and test sets.

Question 5

Random Forest is a robust machine learning algorithm that can be used for a variety of tasks including regression and classification. It is an ensemble method, meaning that a random forest model is made up of a large number of small decision trees, called estimators, which each produce their own predictions. The random forest model combines the predictions of the estimators to produce a more accurate prediction. It works by aggregating the predictions made by multiple decision trees of varying depth. Every decision tree in the forest is trained on a subset of the dataset; this is called the bootstrapped dataset. The portion of samples that were left out of each decision tree in the forest is referred to as the Out-Of-Bag (OOB) dataset. As we'll see later, the model will automatically evaluate its own performance by running each of the samples in the OOB dataset through the forest.

Since the practical classification task is to determine the movement of "Class" (i.e. Today's stock movement) based on today's stock price, and yesterday's movement, we omit the following features of the formula when building the model: Class, DailyReturn, and Classifier(threshold). That is, because including those features in the model would result in 100% accuracy in predicting class, since the model will know what class is already. By excluding that, we are effectively trying to predict class, with only the knowledge of today's stock prices, and yesterday's stock's movement (IClass)

Below, we plot the performance of the training and test set of the random forest algorithm for the following number of trees [100,200,300,400].



The random forest performance accuracy lied generally around 75% for the training set, and around 52-53% for the test set. We further explore in detail the performance of each ntrees model by class with confusion matrices.

NTREES=100	Predicted:Up	Predicted:Down	Predicted:Stable
True:Up	55%	19%	26%
True:Down	25%	46%	29%
True:Stable	28%	22%	50%

NTREES=200	Predicted:Up	Predicted:Down	Predicted:Stable
True:Up	55%	19%	26%
True:Down	25%	47%	29%
True:Stable	28%	22%	50%

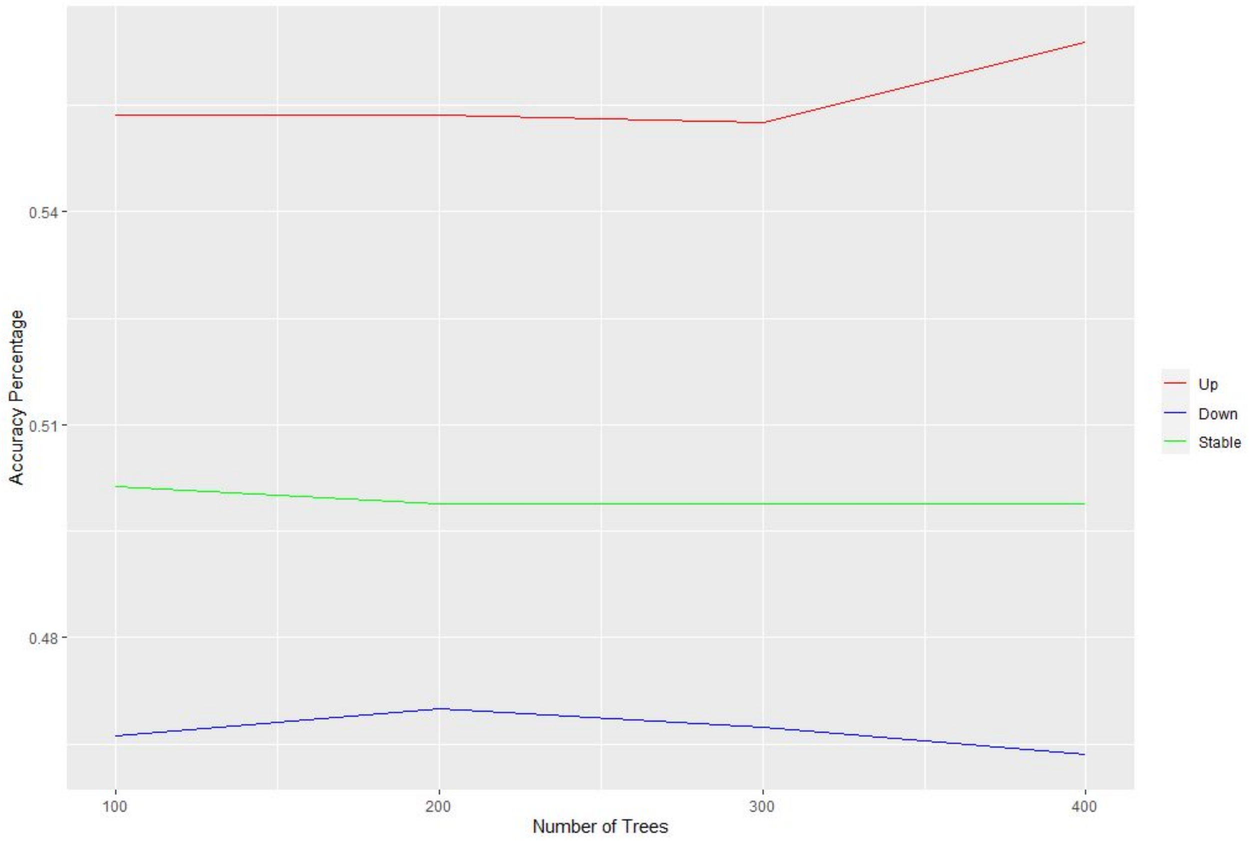
NTREES=300	Predicted:Up	Predicted:Down	Predicted:Stable
True:Up	55%	19%	26%
True:Down	25%	47%	29%
True:Stable	27%	23%	50%

NTREES=400	Predicted:Up	Predicted:Down	Predicted:Stable
True:Up	56%	18%	25%
True:Down	25%	46%	28%
True:Stable	27%	23%	50%

Based on our confusion matrices, we can see that regardless of ntrees, the accuracy of each class prediction remains around the same value, as indicated by the number of trees vs accuracy performance plot above. That is, Up is generally predicted correctly ~ 55% of the time, Down is predicted correctly ~ 46% of the time and Stable is predicted correctly around 50% of the time.

Question 6

In this section, we observe the performance by class for each number of ntrees in 100, 200, 300, and 400.



The random forest performance accuracy is generally between 45% to 56% for ntrees = 100,200,300, and 400. When we explore three curves, with each curve representing the diagonal coefficients for each class for n trees = 100, 200, 300, 400, we observe that our best random forest model occurs when ntrees = 400 with a very miniscule advantage.

Question 7

There are two measures of importance given for each variable in the random forest. The first measure is based on how much the accuracy decreases when the variable is excluded. This is further broken down by outcome class. The second measure is based on the decrease of Gini impurity when a variable is chosen to split a node.Each tree has its own out-of-bag sample of data that was not used during construction. This sample is used to calculate the importance of a specific variable. First, the prediction accuracy on the out-of-bag sample is measured. Then, the values of the variable in the out-of-bag-sample are randomly shuffled, keeping all other variables the same. Finally, the decrease in prediction accuracy on the shuffled data is measured.

The mean decrease in accuracy across all trees is reported. This importance measure is also broken down by outcome class.When a tree is built, the decision about which variable to split at each node uses a calculation of the Gini impurity. For each variable, the sum of the Gini decreases across every tree of the forest and is accumulated every time that variable is chosen to split a node. The sum is divided by the number of trees in the forest to give an average. The scale is irrelevant: only the relative values matter. The importance of each feature is shown below:

Feature	**MeanDecreaseGini (Importance)**
IClass	35.8
Open	148.2
High	145.6
Low	145.6
Close	131.5
Adj.Close	133.8
Volume	201.8

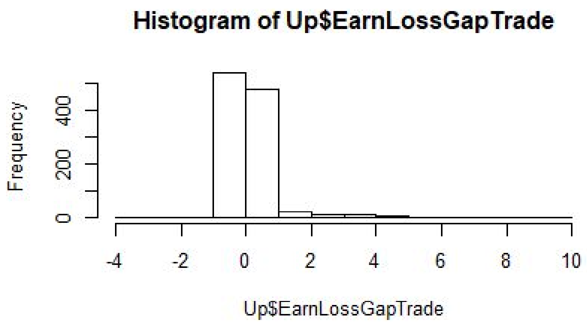
EarnLossGapTrade	305.0
IMovement	182.3
IClassifier	224.5

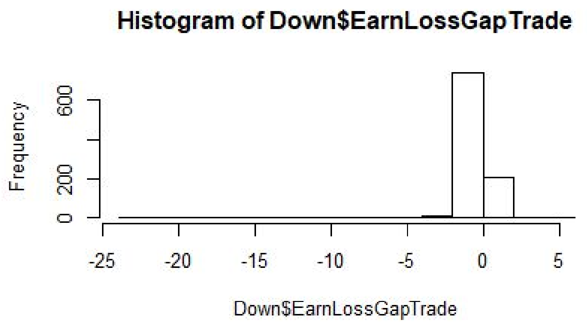
For better visualization, the features are sorted by importance and displayed below:

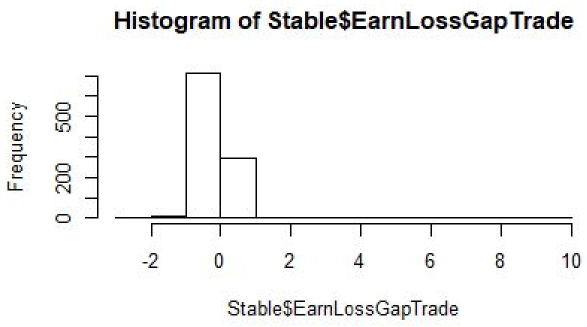
Sorted Feature	**MeanDecreaseGini (Importance)**
EarnLossGapTrade	305.0
IClassifier	224.5
Volume	201.8
IMovement	182.3
Open	148.2
High	145.6
Low	145.6
Adj.Close	133.8
Close	131.5
IClass	35.8

Question 8

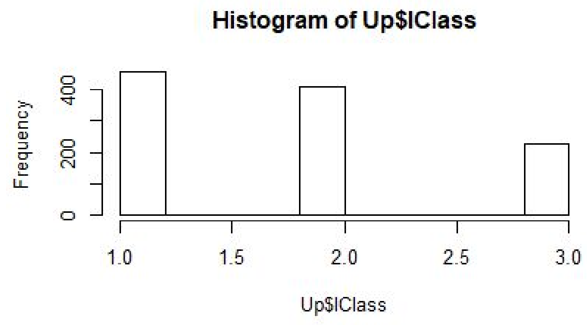
After identifying the highest and lowest features from the previous question, we can assume that Earn Loss Gap trade has the highest importance and IClass is the lowest. To understand how the highest and lowest features differ, a Histogram and the K-S test will be performed. The histogram on each feature will display how distinct each class is in that feature and the K- S test is a numerical confirmation. The D- value of the K-S test represents the distance of each class pair. The larger the D- value, the more distinct the class pair.

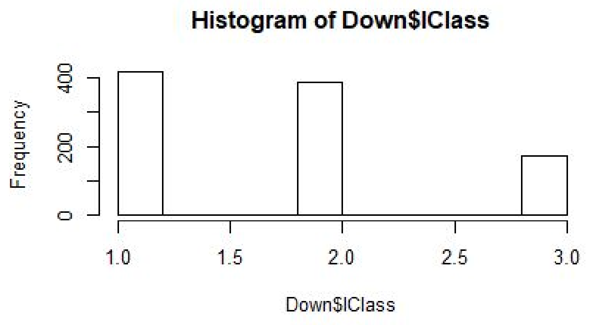


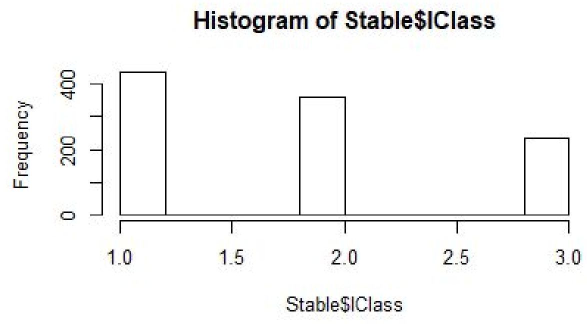




As we can see, the three histograms differ in trend. We can assume that each class is distinct in this feature. This represents that Earn Loss Gap Trade is a good feature to predict classes.

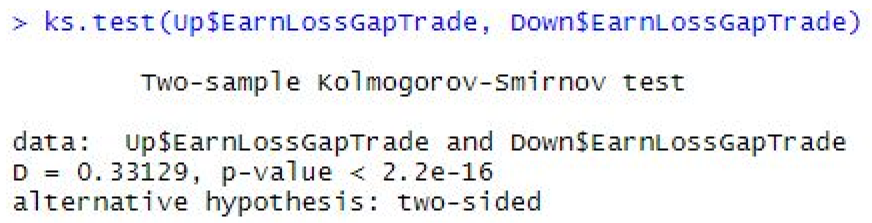




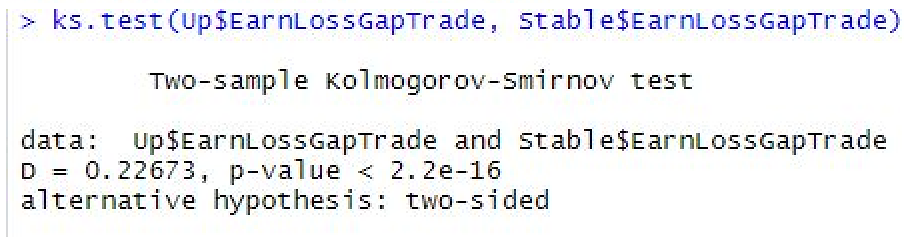


Here we can see that IClass has the same trend. We can assume that IClass features will have a difficult time in differentiating classes. It is possible that extracting this feature may improve accuracy in the model.

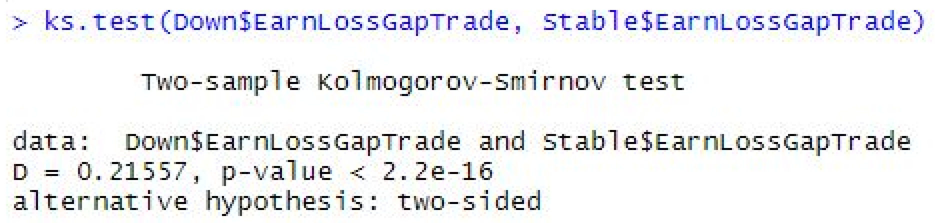
Comparatively, we can conclude that EarnLossGapTrade is most prominent in predicting Today's stock movement, as the formula to derive that is Today's opening price - Yesterday's closing price, which one can imagine would be a good predictor compared to Yesterday's stock movement which is denoted as T -1 Open (Yesterday's Opening price) - [T-2] Open (The day before's opening price).



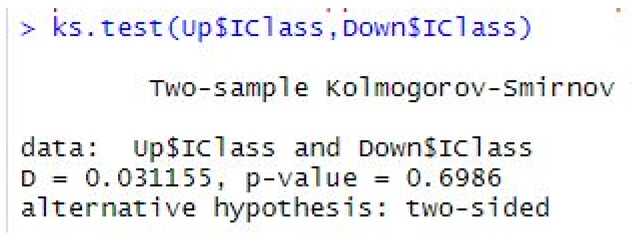
```
> ks.test(Up$EarnLossGapTrade, Down$EarnLossGapTrade)

        Two-sample Kolmogorov-Smirnov test

data:  Up$EarnLossGapTrade and Down$EarnLossGapTrade
D = 0.33129, p-value < 2.2e-16
alternative hypothesis: two-sided
```



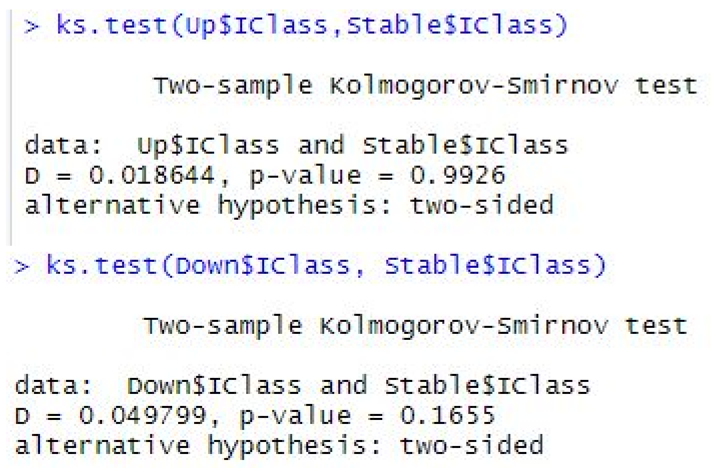
```
> ks.test(Up$EarnLossGapTrade, Stable$EarnLossGapTrade)

        Two-sample Kolmogorov-Smirnov test

data:  Up$EarnLossGapTrade and Stable$EarnLossGapTrade
D = 0.22673, p-value < 2.2e-16
alternative hypothesis: two-sided
```



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

        Two-sample Kolmogorov-Smirnov test

data:  Down$EarnLossGapTrade and Stable$EarnLossGapTrade
D = 0.21557, p-value < 2.2e-16
alternative hypothesis: two-sided
```



```
> ks.test(Up$IClass,Down$IClass)

        Two-sample Kolmogorov-Smirnov test

data:  Up$IClass and Down$IClass
D = 0.031155, p-value = 0.6986
alternative hypothesis: two-sided
```



```
> ks.test(Up$IClass,Stable$IClass)

        Two-sample Kolmogorov-Smirnov test

data:  Up$IClass and Stable$IClass
D = 0.018644, p-value = 0.9926
alternative hypothesis: two-sided

> ks.test(Down$IClass, Stable$IClass)

        Two-sample Kolmogorov-Smirnov test

data:  Down$IClass and Stable$IClass
D = 0.049799, p-value = 0.1655
alternative hypothesis: two-sided
```

KS test is used to test the null hypothesis that two datasets come from the same distribution. The p-value is <0.01 across the three pairs of classes for the feature EarnLossGapTrade, thus we conclude that the three datasets come from different distributions. The p-value>0.01 across the three pairs of classes for the feature IClass, thus we conclude that the three datasets come from the same distribution. We can also see that Earn Loss Gap trade has a larger distance in each class pair compared to the IClass pairs.

Question 9

In this section we attempt to train a new random forest classifier dedicated only to cases belonging in the cluster with the lowest gini score. Initially, cluster 6 was the most prime candidate with a gini index of 0, however it seems faulty as it has only 10 cases, and all 10 of those cases exist within a single class. We then move to the next legitimate gini index cluster, cluster 1.

Since this cluster is extremely imbalanced as expected, we conduct SMOTE cloning technique to rebalance the cases for good training and test sets.This function handles unbalanced classification problems using the SMOTE (Synthetic Minority Over-sampling Technique) method. Namely, it can generate a new "SMOTEd" data set that addresses the class imbalance problem. Now that we obtained balanced classes, we ran the best random forest function on our dataset with the balanced classes.

With the cloned balanced data, we split the training and test set in a random 80% and 20% split for our training and test set, and run same Random Forest conditions to obtain the following results:

Question 10

The RF classifier is 91% accurate in predicting class Up, with a false positive rate of 0%, and false negative rate of 9%. It is 100% accurate in predicting class Down and Stable.

New RF on Gini cluster

	Predicted:Up	Predicted:Down	Predicted:Stable
True:Up	91%	9%	0%
True:Down	0%	100%	0%
True:Stable	0%	0%	100%

Confusion matrix from section 5

NTREES=400	Predicted:Up	Predicted:Down	Predicted:Stable
True:Up	56%	18%	25%
True:Down	25%	46%	28%
True:Stable	27%	23%	50%

From the previous results in question five we can see overall the accuracy of the model in New RF is increased. We can assume that targeting the least impure cluster does influence the model by increasing accuracy. Because the Gini differ in a large range we can not assume that there is an advantage to training each cluster separately. However, if the Gini values were similar there could be an advantage to training and testing each cluster separately.

Question 11

The support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. It uses a decision boundary to classify two classes by distance. The SVM algorithm has many pro's such as less computation power. However, it is very sensitive to outliers. This could cause the accuracy to be low.
The two classes chosen for this section are Up, and Down in which we use new training sets of those classes to train a linear support vector machine model to classify whether a case is "Up" or "Down". The svm function in R is used to run this algorithm, with several cost values tested, namely {0.01,0.1,1,5}. Ultimately, cost = 0.1 is used to train the svm model.

Train performance

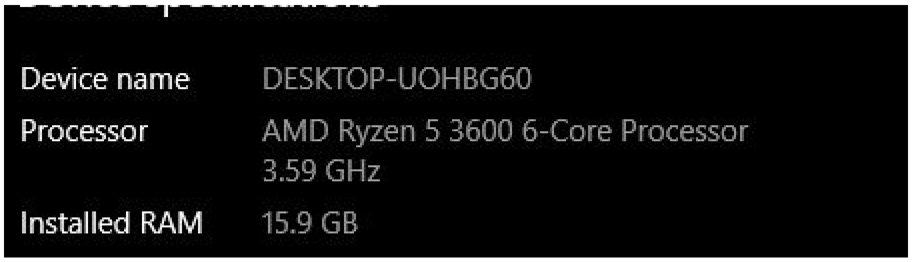
	Predicted:Up	Predicted:Down
True:Up	95%	5%
True:Down	53%	47%

Test performance

	Predicted:Up	Predicted:Down
True:Up	90%	10%
True:Down	60%	40%

Predicting power is lost in both classes when we move from the train set to the test set. Based on the confusion matrix, we can see that the SVM algorithm performs well in predicting Class Up with an accuracy of 90%, but not as well in predicting class Down. This result indicates that Class Up and Down are distinct. A way to possibly improve these results could be by testing a polynomial or radial kernel. However, this is a trial- and - error process and finding the correct parameters is important.

Computer :



Device name DESKTOP-UOHBG60
Processor AMD Ryzen 5 3600 6-Core Processor 3.59 GHz
Installed RAM 15.9 GB

Time Computations:
CompEst(SDATA,km6): 2s
CompEst(SDATA,model400):1s
CompEst(SDATA,svmodel): 13s

We can see that the random forest model performed best, followed by kmeans, and then svm.

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")

ncol(TSLA)
#Data Setup
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] <- 2
    }
    else {
      TSLA$Class[i] <- 3
    }
  }
```

```
}

TSLA$DailyReturn[which(is.na(TSLA$DailyReturn))] <-
mean(TSLA$DailyReturn[2:nrow(TSLA)])
TSLA$EarnLossGapTrade[which(is.na(TSLA$EarnLossGapTrade))] <-
mean(TSLA$EarnLossGapTrade[2:nrow(TSLA)])
TSLA$Classifier[which(is.na(TSLA$Classifier))] <-
mean(TSLA$Classifier[2:nrow(TSLA)])
TSLA$Class[which(is.na(TSLA$Class))] <- 3

TSLA$IMOVEMENT <- NA
TSLA$IClass <- NA
TSLA$IClassifier <- NA

for (i in 1:nrow(TSLA)) {
  if(i > 2) {
    TSLA$IMOVEMENT[i] <- TSLA$Open[i - 1] - TSLA$Open[i - 2]
    #TSLA$IEarnLossGapTrade[i] <- TSLA$Open[i] - TSLA$Close[i-1]
    TSLA$IClassifier[i] <- TSLA$Open[i - 2] * 0.006

    if (TSLA$IMOVEMENT[i] > TSLA$IClassifier[i]) {
      TSLA$IClass[i] <- 1
    }
    else if (TSLA$IMOVEMENT[i] < (TSLA$IClassifier[i] * -1)) {
      TSLA$IClass[i] <- 2
    }
    else {
      TSLA$IClass[i] <- 3
    }
  }
}

table(TSLA$Class)

orgtablelength = nrow(TSLA)
count = 0
for (i in 1:orgtablelength) {
  if (i > 1) {
    if (TSLA$Class[i] == 3) {
      count = count + 1
```

```
     TSLA[(orgtablelength + count),] = TSLA[i,]
   }
  }

}

TSLA$IMOVEMENT[which(is.na(TSLA$IMOVEMENT))] <-
mean(TSLA$IMOVEMENT[3:nrow(TSLA)])
TSLA$IClassifier[which(is.na(TSLA$IClassifier))] <-
mean(TSLA$IClassifier[3:nrow(TSLA)])
TSLA$IClass[which(is.na(TSLA$IClass))] <- 3

#Q1 standardization
SDATA <- scale(TSLA[,c(2:10,12,14)])
SDATA <- cbind(TSLA[,c(11,13)],SDATA)

#Q1 PCA
library(factoextra)
pr.out <- prcomp(SDATA[3:13], scale = FALSE, center = TRUE)
variance <- data.frame(summary(pr.out)$importance[2,])

pr.var=pr.out$sdev^2
pve=pr.var/sum(pr.var)
pve
#plot curves
plot(cumsum(pve),xlab="PC",ylab="Cumulative Propotion of Var
Explained",ylim=c(0,1),type='b')
plot((1:11), variance$summary.pr.out..importance.2..., type = 'o', xlab = 'R', ylab = 'PVE of
R', main = "PVE Plot")

#First 3 eigenvalues
corr = cor(SDATA[3:ncol(SDATA)])
corr = round(corr,2)
EV = eigen(corr)
eigenvalues = round(EV$values,2)
eigenvectors = EV$vectors
round((eigenvalues/sum(eigenvalues))  * 100,2)

library(rgl)
library(ggplot2)
library(plotly)
centpca <- pr.out$x[,1:3]

variance <- data.frame(summary(pr.out)$importance[2,])
plot((1:11), variance$summary.pr.out..importance.2..., type = 'o', xlab = 'R', ylab = 'PVE of
R', main = "PVE Plot")
plot((1:11), cumsum(variance$summary.pr.out..importance.2...), type = 'o', xlab = 'R', ylab
= 'PVE of R', main = "PVE Plot")

index = 0
sum = 0
for (i in 1:nrow(variance)) {
  if (sum < .95){
    sum = sum + variance$summary.pr.out..importance.2...[i]
  }
  else {
    index = i - 1
    break
  }

}

sum
index
newdata <- pr.out$x[,1:5]
centpca <- pr.out$x[,1:3]
NDATA <- cbind(SDATA[,c('Class','IClass')],newdata)

Up <- data.frame(NDATA %>% filter(NDATA$Class == 1))
Down <- data.frame(NDATA %>% filter(NDATA$Class == 2 ))
Stable <- data.frame(NDATA %>% filter(NDATA$Class == 3))

UpDown <- bind_rows(Up,Down)
plot3d(centpca, col = UpDown$Class)
DownStable <- bind_rows(Down,Stable)
plot3d(centpca, col = DownStable$Class)
UpStable <- bind_rows(Up,Stable)
plot3d(centpca, col = UpStable$Class)

#Question 2
```

```
library(factoextra)
library(gridExtra)
set.seed(1)
km <- kmeans(SDATA, centers = 5, nstart = 50)
km$size

km1 <- kmeans(SDATA, centers = 1, nstart = 50)
kmeans1 <- fviz_cluster(km1, data= SDATA, geom = "point", ellipse.type = "convex",
ggtheme = theme_bw())
km2 <- kmeans(SDATA, centers = 2, nstart = 50)
kmeans2 <- fviz_cluster(km2, data= SDATA, geom = "point", ellipse.type = "convex",
ggtheme = theme_bw())
km3 <- kmeans(SDATA, centers = 3, nstart = 50)
kmeans3 <- fviz_cluster(km3, data= SDATA, geom = "point", ellipse.type = "convex",
ggtheme = theme_bw())
km4 <- kmeans(SDATA, centers = 4, nstart = 50)
kmeans4 <- fviz_cluster(km4, data= SDATA, geom = "point", ellipse.type = "convex",
ggtheme = theme_bw())
km5 <- kmeans(SDATA, centers = 5, nstart = 50)
kmeans5 <- fviz_cluster(km5, data= SDATA, geom = "point", ellipse.type = "convex",
ggtheme = theme_bw())
grid.arrange(kmeans1,kmeans2,kmeans3,kmeans4,kmeans5)

km6 <- kmeans(SDATA, centers = 6, nstart = 50)
kmeans6 <- fviz_cluster(km6, data= SDATA, geom = "point", ellipse.type = "convex",
ggtheme = theme_bw())
km7 <- kmeans(SDATA, centers = 7, nstart = 50)
kmeans7 <- fviz_cluster(km7, data= SDATA, geom = "point", ellipse.type = "convex",
ggtheme = theme_bw())
km8 <- kmeans(SDATA, centers = 8, nstart = 50)
kmeans8 <- fviz_cluster(km8, data= SDATA, geom = "point", ellipse.type = "convex",
ggtheme = theme_bw())
km9 <- kmeans(SDATA, centers = 9, nstart = 50)
kmeans9 <- fviz_cluster(km9, data= SDATA, geom = "point", ellipse.type = "convex",
ggtheme = theme_bw())
kmx <- kmeans(SDATA, centers = 10, nstart = 50)
kmeansx <- fviz_cluster(kmx, data= SDATA, geom = "point", ellipse.type = "convex",
ggtheme = theme_bw())
grid.arrange(kmeans6,kmeans7,kmeans8,kmeans9,kmeansx)
RV = data.frame(matrix(ncol = 1, nrow = 10))
```

```
colnames(RV) = c("Perf")
RV$Perf = c(1 - km1$tot.withinss/km1$totss, 1 - km2$tot.withinss/km2$totss,1 -
km3$tot.withinss/km3$totss,1 - km4$tot.withinss/km4$totss,1 -
km5$tot.withinss/km5$totss,
            1 - km6$tot.withinss/km6$totss,1 - km7$tot.withinss/km7$totss,1 -
km8$tot.withinss/km8$totss,1 - km9$tot.withinss/km9$totss,1 - kmx$tot.withinss/kmx$totss
)
plot(RV$Perf, xlab = "k", ylab = "Reduction of Variance (Perf(k))", main = "K-means
perfomance per K")
lines(RV$Perf)

SDATA$cluster <- as.factor(km6$cluster)
ClusterTable = with(SDATA, table(cluster,Class))

ginie_index = c()
for(a in 1:10){
  CLUSTERS = kmeans(SDATA,iter.max = 10, a)
  CLUSTERS$cluster
  SDATAG <- cbind(SDATA,CLUSTERS$cluster)
  gini = c()
  k = seq(1,a)
  for(i in k){
    cluster_i = SDATAG[SDATAG$`CLUSTERS$cluster` == 1,]
    Fs_1 = dim(cluster_i[cluster_i$Class == 1,])[1]/dim(cluster_i)[1]
    Fs_2 = dim(cluster_i[cluster_i$Class == 2,])[1]/dim(cluster_i)[1]
    Fs_3 = dim(cluster_i[cluster_i$Class == 3,])[1]/dim(cluster_i)[1]
    gini[i] = Fs_1*(1-Fs_1) + Fs_2*(1-Fs_2) + Fs_3*(1-Fs_3)
  }
  ginie_index[a] = sum(gini)/a
}
par(mfrow = c(1,1))
plot(1:10, ginie_index, type = "o", main = "Gini(K) vs K", xlab = "number of clusters",
ylab = "Gini Index")
library(reldist)
gini(ClusterTable)

F <- function(cluster, i) {
  f <- 0

  f <- table(cluster$Class)[i]/sum(table(cluster$Class))
  return(f)
}

gini <- function(F1,F2,F3) {
  gini = F1 * (1 - F1) + F2 * (1 - F2) + F3 * (1 - F3)

  return(gini)
}

cluster1 = SDATA$cluster == 1
cluster2 = SDATA$cluster == 2
cluster3 = SDATA$cluster == 3
cluster4 = SDATA$cluster == 4
cluster5 = SDATA$cluster == 5
cluster6 = SDATA$cluster == 6
Clu1 = SDATA[cluster1,]
Clu2 = SDATA[cluster2,]
Clu3 = SDATA[cluster3,]
Clu4 = SDATA[cluster4,]
Clu5 = SDATA[cluster5,]
Clu6 = SDATA[cluster6,]

ClusterTable
CL1C1 <- F(Clu1,1)
CL1C2 <- F(Clu1,2)
CL1C3 <- F(Clu1,3)
giniC1 <- gini(CL1C1, CL1C2, CL1C3)

CL2C1 <- F(Clu2,1)
CL2C2 <- F(Clu2,2)
CL2C3 <- 0
giniC2 <- gini(CL2C1, CL2C2, CL2C3)

CL3C1 <- F(Clu3,1)
CL3C2 <- F(Clu3,2)
CL3C3 <- F(Clu3,3)
giniC3 <- gini(CL3C1, CL3C2, CL3C3)

CL4C1 <- F(Clu4,1)
CL4C2 <- F(Clu4,2)
```

```
CL4C3 <- F(Clu4,3)
giniC4 <- gini(CL4C1, CL4C2, CL4C3)

CL5C1 <- F(Clu5,1)
CL5C2 <- F(Clu5,2)
CL5C3 <- F(Clu5,3)
giniC5 <- gini(CL5C1, CL5C2, CL5C3)

CL6C1 <- F(Clu6,1)
CL6C2 <- F(Clu6,2)
CL6C3 <- F(Clu6,3)
giniC6 <- 1 * (1 - 1) + 0 * (1 - 0) + 0 * (1 - 0)


giniarr <- c(giniC1,giniC2,giniC3,giniC4,giniC5,giniC6)
sum(giniarr)

#Q3 K* = 6
km6$centers
km6$size
giniarr <- c(giniC1,giniC2,giniC3,giniC4,giniC5,giniC6)
km6$withinss
ClusterTable #frequencies of each class
plot3d(centpca, col = SDATA$cluster)

#Q4
#Training/Testsets for each class cj
set.seed(1)
Up <- data.frame(SDATA %>% filter(SDATA$Class == 1))
Down <- data.frame(SDATA %>% filter(SDATA$Class == 2 ))
Stable <- data.frame(SDATA %>% filter(SDATA$Class == 3))

UpIndexes <- sample(1:nrow(Up), size = 0.2 * nrow(Up))
DownIndexes <- sample(1:nrow(Down), size = 0.2 * nrow(Down))
StableIndexes <- sample(1:nrow(Stable), size = 0.2 * nrow(Stable))

UpTest <- Up[UpIndexes,]
DownTest <- Down[DownIndexes,]
StableTest <- Stable[StableIndexes,]
```

```
UpTrain <- Up[-UpIndexes,]
DownTrain <- Down[-DownIndexes,]
StableTrain <- Stable[-StableIndexes,]

nrow(UpTest)
nrow(DownTest)
nrow(StableTest)
nrow(UpTrain)
nrow(DownTrain)
nrow(StableTrain)


TRAINSET <- bind_rows(UpTrain,DownTrain,StableTrain)
TESTSET<- bind_rows(UpTest, DownTest, StableTest)

TRAINSET <- transform(TRAINSET, Class = as.factor(Class))
TESTSET <- transform(TESTSET, Class = as.factor(Class))

table(TRAINSET$Class) #does not need cloning
table(TESTSET$Class) #does not need cloning

#Q5 random forest
set.seed(1)
library(randomForest)
library(party)
library(caret)

model1 <- randomForest(Class~ IMOVEMENT + DailyReturn, data = TRAINSET,
na.action = na.omit)
plot(model1)

model100<- randomForest(Class ~ . - Class- DailyReturn - Classifier - cluster, data=
TRAINSET, ntrees = 100, ntry = (ncol(TRAINSET)))
modeltest100<- randomForest(Class ~. - Class- DailyReturn - Classifier- cluster, data=
TESTSET, ntrees = 100, ntry = (ncol(TESTSET)))

trainpred100 <- predict(model100, newdata= TESTSET)
CONFTRAIN100 <- confusionMatrix(as.factor(TESTSET$Class),as.factor(trainpred100))

testpred100 <- predict(modeltest100, newdata= TRAINSET)
```

```
CONFTEST100 <- confusionMatrix(as.factor(TRAINSET$Class),as.factor(testpred100))

model200<- randomForest(Class ~ . - Class- DailyReturn - Classifier- cluster, data=
TRAINSET, ntrees = 200, ntry = (ncol(TRAINSET)))
modeltest200<- randomForest(Class ~. - Class- DailyReturn - Classifier- cluster, data=
TESTSET, ntrees = 200, ntry = (ncol(TESTSET)))

trainpred200 <-predict(model200, newdata= TESTSET)
CONFTRAIN200 <- confusionMatrix(as.factor(TESTSET$Class),as.factor(trainpred200))

testpred200 <- predict(modeltest200, newdata= TRAINSET)
CONFTEST200 <- confusionMatrix(as.factor(TRAINSET$Class),as.factor(testpred200))


model300<- randomForest(Class ~. - Class- DailyReturn - Classifier- cluster, data=
TRAINSET, ntrees = 300, ntry = (ncol(TRAINSET)))
modeltest300<- randomForest(Class ~.- Class- DailyReturn - Classifier- cluster, data=
TESTSET, ntrees = 300, ntry = (ncol(TESTSET)))

trainpred300 <- predict(model300, newdata= TESTSET)
CONFTRAIN300 <- confusionMatrix(as.factor(TESTSET$Class),as.factor(trainpred300))

testpred300 <- predict(modeltest300, newdata= TRAINSET)
CONFTEST300 <- confusionMatrix(as.factor(TRAINSET$Class),as.factor(testpred300))

model400<- randomForest(Class ~.- Class- DailyReturn - Classifier- cluster, data=
TRAINSET, ntrees = 400, ntry = (ncol(TRAINSET)))
modeltest400<- randomForest(Class ~.- Class- DailyReturn - Classifier- cluster, data=
TESTSET, ntrees = 400, ntry = (ncol(TESTSET)))

trainpred400 <- predict(model400, newdata= TESTSET)
CONFTRAIN400 <- confusionMatrix(as.factor(TESTSET$Class),as.factor(trainpred400))

testpred400 <- predict(modeltest400, newdata= TRAINSET)
CONFTEST400 <- confusionMatrix(as.factor(TRAINSET$Class),as.factor(testpred400))

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

CONFTRAIN100$overall['Accuracy']
ntrees <- c(100,200,300,400)
```

```
library(ggplot2)
b = bind_cols(ntrees,
c(CONFTEST100$overall['Accuracy'],CONFTEST200$overall['Accuracy'],CONFTEST30
0$overall['Accuracy'],CONFTEST400$overall['Accuracy']),c(CONFTRAIN100$overall['A
ccuracy'],CONFTRAIN200$overall['Accuracy'],CONFTRAIN300$overall['Accuracy'],CO
NFTRAIN400$overall['Accuracy']))
ggplot() +
  geom_line(data = b, aes(x = ...1, y = ...2, color = "Test")) +
  geom_line(data = b, aes(x = ...1, y = ...3, color = "Train")) +
  xlab("Number of Trees") + ylab("Accuracy Percentage") +
  scale_colour_manual("", breaks = c("Test","Train"), values = c("blue","red"))

#Display 4 test set confusion matrcies
CONFTEST100
CONFTEST200
CONFTEST300
CONFTEST400

#Question 6
CONF100 <-
prop.table(CONFTEST100$table)/rowSums(prop.table(CONFTEST100$table))
CONF200 <-
prop.table(CONFTEST200$table)/rowSums(prop.table(CONFTEST200$table))
CONF300 <-
prop.table(CONFTEST300$table)/rowSums(prop.table(CONFTEST300$table))
CONF400 <-
prop.table(CONFTEST400$table)/rowSums(prop.table(CONFTEST400$table))
UpPerf <- c(CONF100[1],CONF200[1],CONF300[1],CONF400[1])
DownPerf <- c(CONF100[5],CONF200[5],CONF300[5],CONF400[5])
StablePerf <- c(CONF100[9],CONF200[9],CONF300[9],CONF400[9])

a <- bind_cols(ntrees, UpPerf,DownPerf, StablePerf)
ggplot() +
  geom_line(data = a, aes(x = ntrees, y = UpPerf, color = "Up")) +
  geom_line(data = a, aes(x = ntrees, y = DownPerf, color = "Down")) +
  geom_line(data = a, aes(x = ntrees, y = StablePerf, color = "Stable")) +
  xlab("Number of Trees") + ylab("Accuracy Percentage") +
  scale_colour_manual("", breaks = c("Up","Down","Stable"), values =
c("red","blue","green"))
```

```
#question 7
BestRF <- model400
model400$importance
sort(model400$importance, decreasing = TRUE)

#question 8
par(mfrow=c(2,2))
hist(Up$EarnLossGapTrade)
hist(Down$EarnLossGapTrade)
hist(Stable$EarnLossGapTrade)

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

hist(Up$IClass)
hist(Down$IClass)
hist(Stable$IClass)

ks.test(Up$IClass,Down$IClass)
ks.test(Up$IClass,Stable$IClass)
ks.test(Down$IClass, Stable$IClass)

#Question 9
#cluster 6 -> cluster1
km6$size
Clu1

library(DMwR)
table(Clu1$Class)
Clu1$Class = as.factor(Clu1$Class)
set.seed(1)
SMOTECLU1 <- SMOTE(Class ~ ., Clu1, perc.over = 2000, perc.under = 100)
na.omit(SMOTECLU1)
table(SMOTECLU1$Class)

SMOTECLU1 <- SMOTE(Class ~ ., SMOTECLU1, perc.over = 3000, perc.under = 200)
table(SMOTECLU1$Class)

Clu1Indexes <- sample(1:nrow(SMOTECLU1), size = 0.2 * nrow(SMOTECLU1))
```

```
Clu1Test <- SMOTECLU1[Clu1Indexes,]
Clu1Train <- SMOTECLU1[-Clu1Indexes,]

table(Clu1Train$Class)
table(Clu1Test$Class)

newmodel<- randomForest(Class ~ . - Class- DailyReturn - Classifier- cluster, data=
Clu1Train, ntrees = 300, ntry = (ncol(Clu1Train)), na.action = na.exclude)

#Question 10

newmodelpred <- predict(newmodel, newdata= Clu1Test)
NEWCONFTEST300 <-
confusionMatrix(as.factor(Clu1Test$Class),as.factor(newmodelpred))
NEWCONF300 <-
prop.table(NEWCONFTEST300$table)/rowSums(prop.table(NEWCONFTEST300$table))
#Question 11
library(e1071)
NEWTRA <- bind_rows(UpTrain,DownTrain)
NEWTST <- bind_rows(UpTest, DownTest)

NEWTRA <- transform(NEWTRA, Class = as.factor(Class))
NEWTST <- transform(NEWTST, Class = as.factor(Class))

svmmodel<- svm(Class~. - Class- DailyReturn - Classifier- cluster, data= NEWTRA ,
kernel = "linear")
svmpred <- predict(svmmodel, newdata = NEWTST)
SVMTRAIN <- confusionMatrix(as.factor(NEWTST$Class),as.factor(svmpred))
SVMTRAINCONF <-
prop.table(SVMTRAIN$table)/rowSums(prop.table(SVMTRAIN$table))

svmmodel<- svm(Class~. - Class- DailyReturn - Classifier- cluster, data= NEWTST , kernel
= "linear")
svmpred <- predict(svmmodel, newdata = NEWTRA)
SVMTEST <- confusionMatrix(as.factor(NEWTRA$Class),as.factor(svmpred))
SVMTESTCONF <- prop.table(SVMTEST$table)/rowSums(prop.table(SVMTEST$table))

install.packages("GuessCompx")
library(GuessCompx)
```

```
CompEst(SDATA,km6)
CompEst(SDATA,model400)
CompEst(SDATA, svmmodel)
```