11/25/2021

MATH 6350

Predict Stock Direction with SVM

Chia-Hung Chien, Tola Ouk, Tyler Stinson
UNIVERSITY OF HOUSTON

Table of Contents

1 Scope of the Report and Data Description

This report is going to predict a particular company's stock price direction of a certain day in the future based on the rate of return of the stock price for the 10 previous days' data of itself and other 19 companies. Below is the list of top 20 companies with highest revenue in the US in 2020.

The period of these 20 companies is from 01/01/2015 to 12/31/2018 which is prior to the pandemic and give a more stable stock price. Also, the data will only focus on the stock price at closing time. After excluding the market closed dates, there are 1006 dates in the data set. We will denote the stock price for company j on day t as $S_j(t)$.

The target company is General Motor, which is the last one in Table 1. On each day "t", $S_{20}(t+1)$ is unknown to all stock traders and the goal is to predict whether the stock price will be up or down for the next day i.e., day "$t+1$" with respect to the currently known $S_{20}(t)$. Thus, all the data from 20 companies will be used. Since the true classification is available, they will be used to build the model.

Table 1 Top 20 Companies with Highest Revenue in the US in 2020

1	2	3	4	5
Walmart	Amazon	Apple Inc.	CVS Health	ExxonMobil
6	7	8	9	10
UnitedHealth Group	Berkshire Hathaway	McKesson Corporation	Amerisource Bergen	Alphabet Inc.
11	12	13	14	15
AT&T	Cigna	Ford Motor Company	Costco	FedEx
16	17	18	19	20
Chevron Corporation	Cardinal Health	Microsoft	JPMorgan Chase	General Motor

Support Vector Machine (SVM) is the targeted algorithm to perform the task. Linear SVM and Radial Kernel SVM will both be implemented and compared on their performances.

2 Data Pre-processing

After examining the data set, there are no missing values. Since each company has a different stock price, it is reasonable to compute the rate of return instead of using stock price. The rate of return, $Y_j(t)$, is calculated as below.

$$Y_j(t) = \frac{S_j(t) - S_j(t-1)}{S_j(t-1)}$$

There is no rate of return for day 1 since we do not have data for day 0. Thus, the only time series that will be used is from $2 \leq t \leq 1006$. This will create a data frame as:

$$\begin{matrix} Y_1(2) & Y_1(3) & \cdots & \cdots & Y_1(1006) \\ Y_2(2) & Y_2(3) & \cdots & \cdots & Y_2(1006) \\ Y_3(2) & Y_2(3) & \cdots & \cdots & Y_3(1006) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ Y_{19}(2) & Y_{19}(3) & \vdots & \vdots & Y_{19}(1006) \\ Y_{20}(2) & Y_{20}(3) & \cdots & \cdots & Y_{20}(1006) \end{matrix}$$

3 Cases and Feature Creation

To predict the stock price direction of day $t+1$, the feature will include the rate of return of all the 20 companies from day $t-9$ to day t. Thus, for each company j, it will create a line vector called $V_j(t)$ with length of 10. Noted that t shall start from 11 since there is no $Y_j(1)$.

$$V_j(t) = [Y_j(t-9), Y_j(t-8), \ldots\ldots, Y_j(t-1), Y_j(t)]$$

Combine the $V_j(t)$ from all the 20 companies will get a long line vector called X_t for each day t with length of 200.

$$X_t = [V_1(t), V_2(t), \ldots\ldots, V_{20}(t)] = [X_t(1), X_t(2), \ldots\ldots, X_t(200)]$$

In the other word, for each day t with $11 \leq t \leq 1005$, we can define the case "t" by its feature vector X_t which have 200 features. This will give us a total 995 cases in the data set. Thus, the dimension of the complete data set is 995×200.

4 Classes Split

On day t, the goal is to predict the stock price direction for the next day, $t+1$. This information is currently unknown to all the stock traders.

$$TARG(t) = Y_{20}(t+1) = \frac{S_{20}(t+1) - S_{20}(t)}{S_{20}(t)}$$

Since the data set is historical, all the target values are available. They will be used to build the model and check for prediction accuracy. Two classes are defined as:

- True Class of Case "t" = HIGH if $TARG(t) \geq 0.6\%$
- True Class of Case "t" = LOW if $TARG(t) < 0.6\%$

This will generate a column vector as true class for each case as below

$$TRUC = Z = [Z(11), Z(12), \ldots\ldots, Z(1005)]^T$$

Table 2 shows the number of cases of each class and their respective percentage in entire data set.

Table 2 Classes size

	LOW	HIGH	Total
# Of Cases	688	307	995
Percentage	69.2%	30.8%	100%

5 Principal Component Analysis (PCA)

Since there will be information loss between the data compression and decompression, it is important to keep as much information as possible. Thus, the Mean Squared Error of Decompression (MSE) i.e., information loss shall be as small as possible. Percentage of Explain Variance (PEV) which is a function of MSE is set to be 90% in this report.

$$PEV(h) = 1 - \frac{MSE(h)}{k} = \frac{\sum_{i=1}^{h} L_i}{k}$$

where

- k is the number of original features
- L_i is the sorted eigenvalue

In our case, there are 200 features in the original data set. A 200×200 eigenvector will be computed together with 200 eigenvalues. Figure 1 shows that 200 eigenvalues of the correlation matrix of the data set with dimension of 995×200 acquired in previous sections.

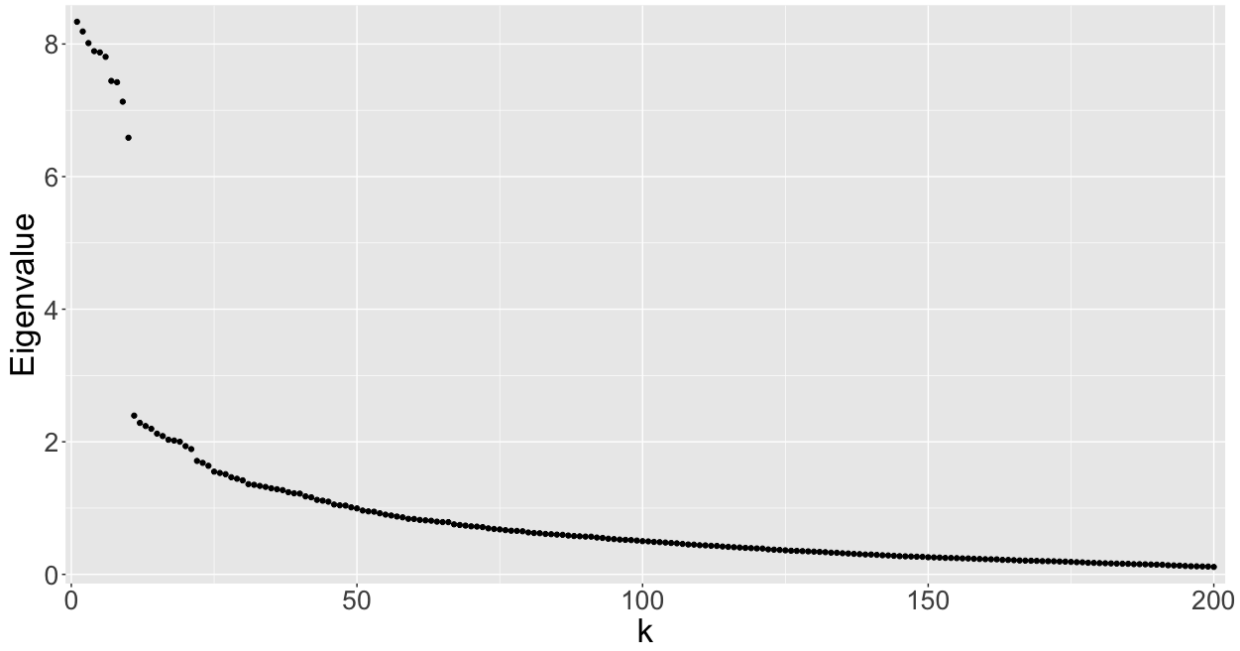

Figure 1 Eigenvalue vs k

In this report, PEV is set to be equal to or larger than 90%. Thus, the smallest integer "h" such that $PEV(h) \geq 90\%$ is 118. Figure 2 shows the PEV versus k.

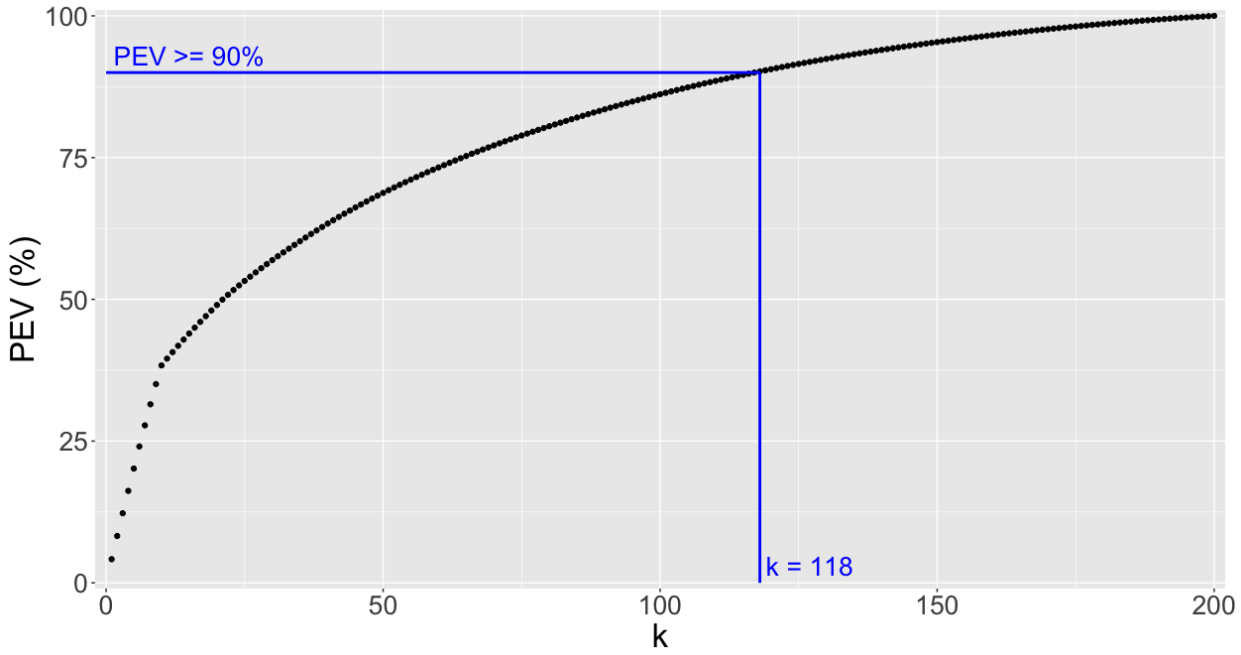

Figure 2 Percentage of Explained Variance (PEV) vs k

The new features after PCA are produced by the multiplication of the Original Data and the Eigenvector (W).

$$PCA\ Data_{995\times200} = Oringal\ Data_{995\times200} \times W_{200\times200}$$

The final dimension of the data after PCA is still 995×200. However, since we have $h = 118$ to get $PEV \geq 90\%$, only the first 118 principal components will be taken from PCA Data to have a data set with size of 995×118.

6 Data Splitting

To reduce the bias toward any class for either training set or test set, evenly and randomly sampling the data shall be performed. The complete test set will be randomly selected from 15% of all the cases in each class and then combined as a complete test set. The remaining 85% cases from each class are combined as complete training set. Table 3 shows that there are 149 cases of test set and 847 cases of training set.

Table 3 Size of Training Set and Test Set

	Training Set	Test Set
LOW	585	103
HIGH	261	46
Total	846	149

The ratio of the two classes is around 47%. Thus, further treatment such as cloning, perturbation, or SMOTE which are used to deal with imbalanced data are not required.

7 Linear SVM Application

First, the linear SVM classifier is applied. To minimize the total cost, $Cost(H)$, of selected hyperplane $H: < w, x > -b - 1 = 0$, several error weight coefficients, C, are tested.

$$Cost(H) = \frac{\|w\|^2}{2} + C \times \sum_{misclassified\ \ cases} |< w, x > -b \pm 1|$$

There is no good way to estimate C. One must try different value to identify the best one. Several values should be tested and the results are compared to identify the best value of C. Initially, 10 values of C are tested: 0.01, 0.25, 0.5, 0.75, 1, 2, 2.5, 5, 7.5, 10. The results are in Table 4.

Table 4 Linear SVM result with different values of C

C	Training Accuracy	Test Accuracy	Accuracy Ratio	%Support
0.01	90.4	87.2	96.5	51.1
0.25	94.6	87.2	92.2	27.4
0.5	95.3	87.2	91.5	24.8
0.75	95.6	88.6	92.7	24.1
1	95.7	89.3	93.3	23.3
2	96.6	88.6	91.7	21.2
2.5	96.7	88.6	91.6	21.2
5	96.8	89.3	92.3	20.3
7.5	96.8	89.9	92.9	19.9
10	96.8	89.9	92.9	19.6

8 Further Linear SVM Application

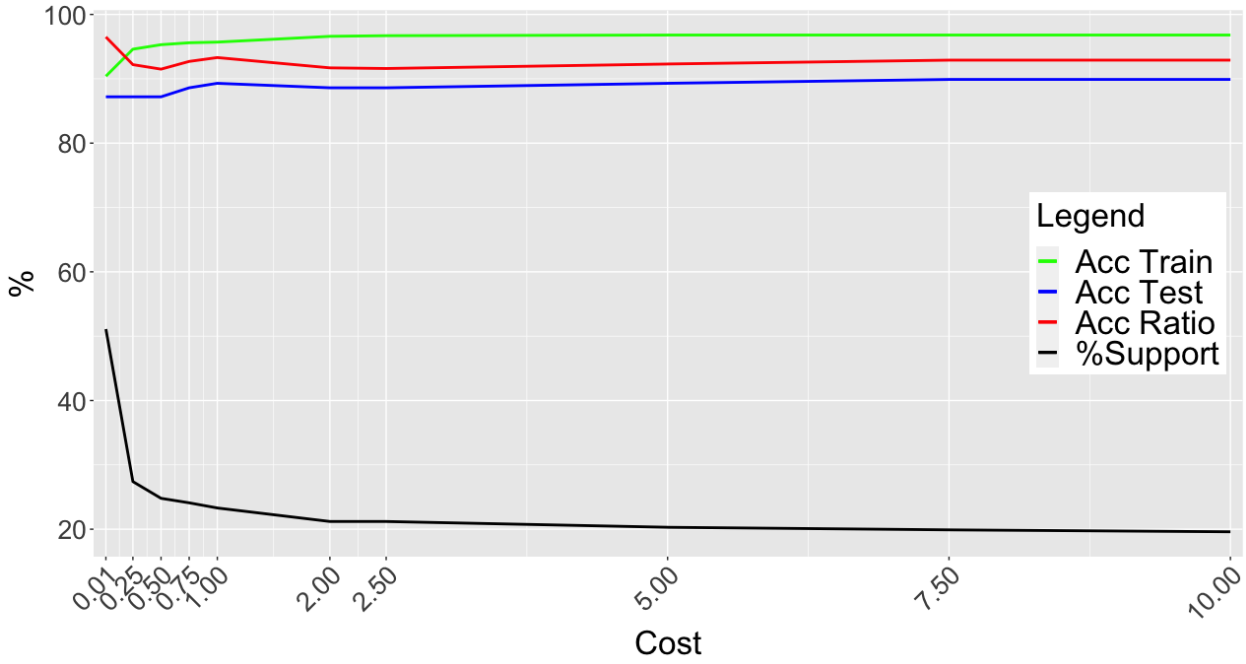

Figure 3 Test Accuracy, Test Accuracy, Accuracy Ratio, and % of Support Vector with 10 initial values of C

Figure 3 shows that the 4 curves of Test Accuracy, Test Accuracy, Accuracy Ratio, and % of Support Vector with the 10 initial values of C. Here are several criteria of selecting the best C.

1. Accuracy Ratio < 1 and closer to 1.
2. Higher C means higher training accuracy but weaker generalization power.
3. Lower C means lower training accuracy but better generalization power.
4. Smaller %support means better generalization.

From these criteria, three of the C values are selected as below:

1. C = 1 $\Rightarrow$ Second Highest Accuracy Ratio
2. C = 5 $\Rightarrow$ High Test Accuracy and Low % of Support Vector
3. C = 10 $\Rightarrow$ Highest Test Accuracy and Lowest % of Support Vector

Based on the 3 temporarily best values of C listed above, 12 new values of C = {0.1, 0.8, 3, 3.5, 4, 6, 7, 8, 9, 15, 20, 25} will be selected to conduct a further Linear SVM classifier hyperparameter selection.

Table 5 Linear SVM result with newly selected values of C

C	Training Accuracy	Test Accuracy	Accuracy Ratio	%Support
0.1	94	87.2	92.8	32.2
0.8	95.3	88.6	93	23.8
3	96.8	88.6	91.5	21.3
3.5	96.9	89.3	92.2	20.8
4	96.9	88.6	91.4	20.8
6	96.5	89.9	93.2	19.9
7	96.9	89.9	92.8	19.9
8	96.9	89.9	92.8	20
9	96.8	89.9	92.9	19.7
15	96.7	89.9	93	19.6
20	96.7	90.6	93.7	19.4
25	96.9	89.9	92.8	19.3

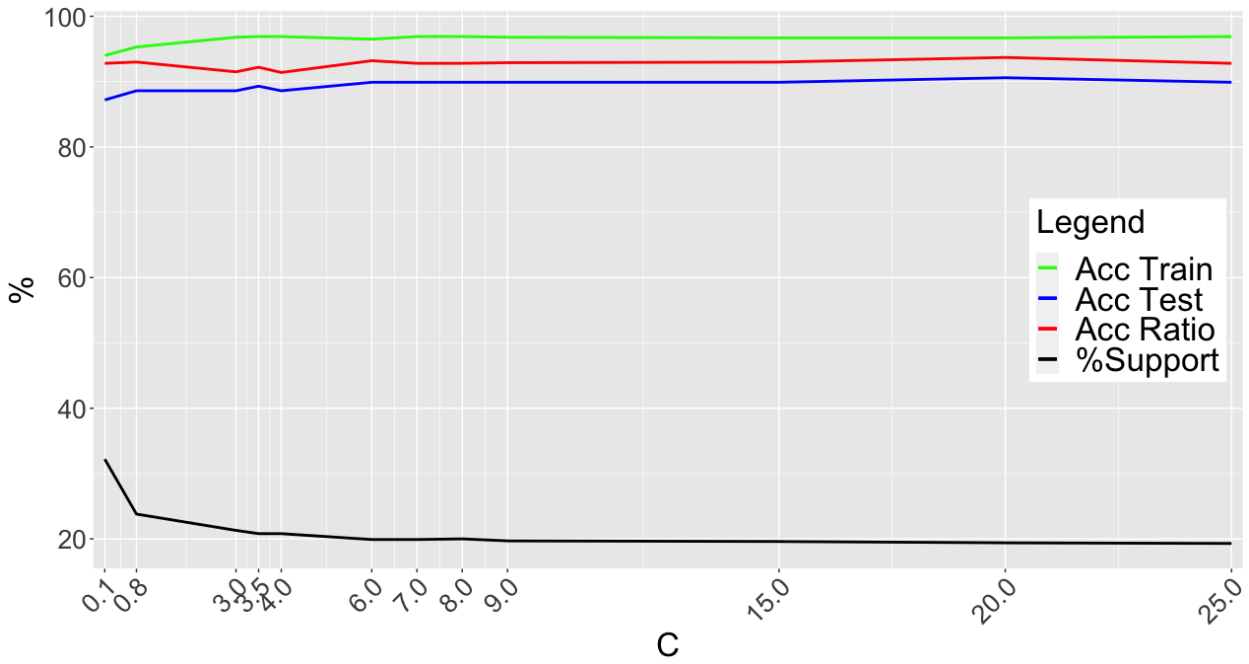

Figure 4 Test Accuracy, Test Accuracy, Accuracy Ratio, and % of Support Vector with 12 new values of C

Table 5 and Figure 4 shows that result of newly selected 12 values of C. It can be observed that all the 4 curves are stable after $C \geq 6$. Training Accuracies settle around 97%, Test Accuracies settle around 90%, Accuracy Ratios settle around 93%, and percentages of Support Vectors settle around 20%. However, increasing C will also increase the computation time. Since the margin is very small while increasing the C, one would like to select the hyperparameter cost with less computation time. In conclusion, C = 6 will be selected for comparison in Section 11.

9 Radial Kernel SVM Application

In this section, positive definite kernel SVM will be introduced. In practice, only 3 types of positive definite kernels are used. The selected one for this report is Radial Kernel:

$$K(x,y) = e^{-gamma \times \|x-y\|^2}$$

9.1 Hilbert Distance

To initialize this hierarchical process, there is a method to give an initial range of gamma selection. The similarity matrix $SIM(i,j)$ for all the cases between two classes with several different values of gamma will be computed.

$$SIM(i,j) = K(CL0_i, CL1_j) = e^{-gamma \times \|CL0_i - CL1_j\|^2}$$

The range of $SIM(i,j)$ is between 0 and 1. From the similarity matrix, compute Hilbert Distance $H(i,j)$ as below:

$$H(i,j) = \sqrt{[2 - 2 \times SIM(i,j)]}$$

Then, display the histograms of all $H(i,j)$. For an acceptable gamma, the histogram must not concentrate near 0 or $\sqrt{2}$. Figure 5 shows the histogram of 4 different values of gamma. It can be observed that gamma = 0.001 and 100 are poor choices, as they are located at 0 and $\sqrt{2}$ respectively. Thus, we will select four values between 0.005 and 10 as gamma which are 0.005, 0.01, 0.1, and 1.

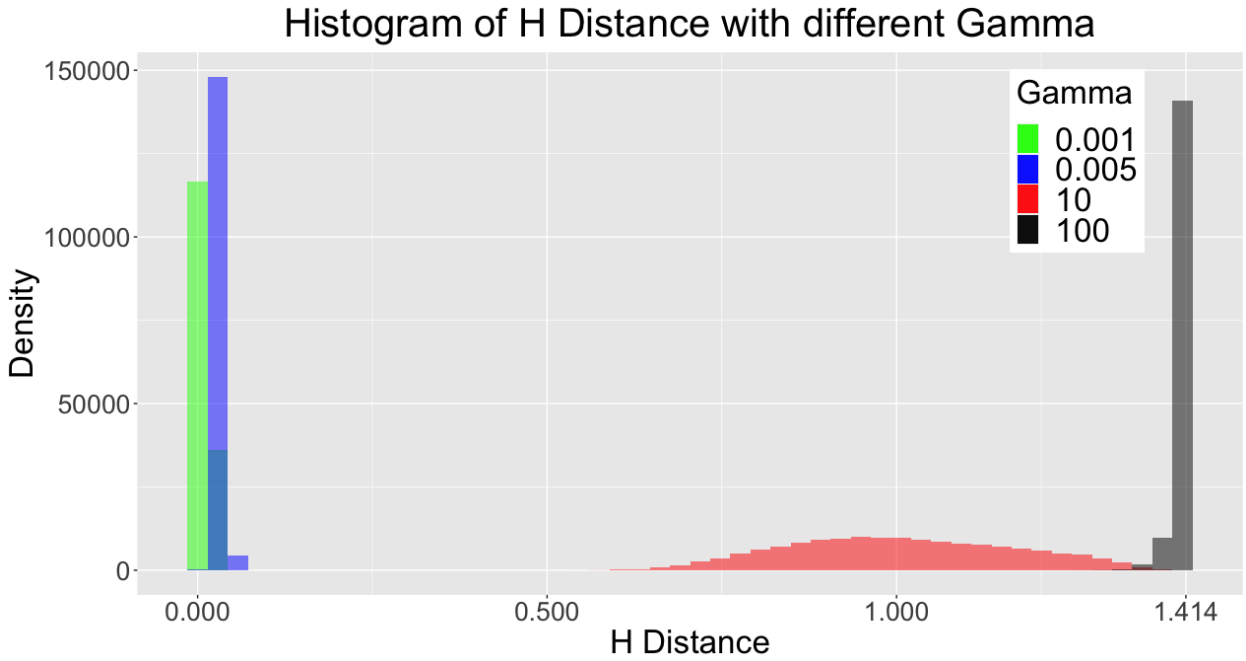

Figure 5 Histogram of Hilbert Distance with different Gamma

9.2 Error Weight Coefficient Selection

As mentioned in Section 7, a good SVM model should minimize the total cost, $Cost(H)$. Thus, 10 values of C are selected: 0.0001, 0.001, 0.01, 0.1, 0.5, 0.75, 1, 5, 10, 20. These values covered a wide range. Pairing with gamma selected in the previous section, this will generate 40 pairs of C and gamma.

9.3 Apply Radial SVM and Result

Table 6 shows the 40 pairs of result for radial kernel SVM. It can be observed that when gamma = 0.1 and 1, no matter which the C value is, the test accuracy is always fixed at 69.1%. Also, the training set are perfectly predicted when C value is higher than 0.75 which means worse generalization power is provided by these models. Comparing gamma = 0.005 and gamma = 0.001, they both provide similar results. Thus, it is reasonable not to go with gamma = 0.1 or 1 which means we prefer a lower gamma value.

Below five pairs of C and gamma (C, gamma) are selected:

1. (0.0001, 0.005) $\Rightarrow$ Lowest % of Support Vector with 100% Accuracy Ratio
2. (0.5, 0.005) $\Rightarrow$ Both Test Accuracy and Training Accuracy start to increase comparing lower C at same gamma.
3. (1, 0.005) $\Rightarrow$ Highest Test Accuracy over all pairs

4. (10, 0.005) ⇒ High Test and Training Accuracy and lower % of Support Vector
5. (1, 0.01) ⇒ Almost perfect Training Accuracy but low Accuracy Ratio

Table 6 Radial Kernel SVM result with (a) gamma = 0.005, (b) gamma = 0.01, (c) gamma = 0.1, (d) gamma = 1

C	Training Accuracy	Test Accuracy	Accuracy Ratio	%Support
0.0001	69.1	69.1	100	61.7
0.001	69.1	69.1	100	62.5
0.01	69.1	69.1	100	71.9
0.1	69.1	69.1	100	73.2
0.5	77.8	73.2	94.1	73.2
0.75	89.8	79.9	89	72.9
1	94	86.6	92.1	72
5	100	85.9	85.9	63.7
10	100	85.2	85.2	62.3
20	100	84.6	84.6	62.3

(a)

C	Training Accuracy	Test Accuracy	Accuracy Ratio	%Support
0.0001	69.1	69.1	100	61.7
0.001	69.1	69.1	100	62.6
0.01	69.1	69.1	100	79.2
0.1	69.1	69.1	100	81.2
0.5	78.4	71.1	90.7	81.1
0.75	95.7	79.9	83.5	81.2
1	99.3	83.9	84.5	81.7
5	100	83.9	83.9	79.4
10	100	83.9	83.9	79.4
20	100	83.9	83.9	79.4

(b)

C	Training Accuracy	Test Accuracy	Accuracy Ratio	%Support
0.0001	69.1	69.1	100	61.7
0.001	69.1	69.1	100	62.5
0.01	69.1	69.1	100	100
0.1	69.1	69.1	100	100
0.5	69.1	69.1	100	100
0.75	100	69.1	69.1	100
1	100	69.1	69.1	100
5	100	69.1	69.1	100
10	100	69.1	69.1	100
20	100	69.1	69.1	100

(c)

C	Training Accuracy	Test Accuracy	Accuracy Ratio	%Support
0.0001	69.1	69.1	100	61.7
0.001	69.1	69.1	100	92.6
0.01	69.1	69.1	100	100
0.1	69.1	69.1	100	100
0.5	69.1	69.1	100	100
0.75	100	69.1	69.1	100
1	100	69.1	69.1	100
5	100	69.1	69.1	100
10	100	69.1	69.1	100
20	100	69.1	69.1	100

(d)

10 Further Radial Kernel SVM Application

From the selected pairs in Section 9.3, further trial and error around these values will be conducted to check if there is better choice of C and gamma. It can be observed that gamma = 0.005 gives almost always the best results (higher test accuracies and lower percentages of support vectors). However, comparing gamma = 0.01 and gamma = 0.005, higher gamma value will give higher training accuracies. Thus, 4 pairs of (C, gamma) will be selected around each best potential pair found in Section 9.3. Table 7 to Table 11 show the results of the additional 20 pairs of (C, gamma) applied to Radial Kernel SVM.

Table 7 Result of 4 pairs of (C, gamma) around C = 0.0001 and gamma = 0.005

C	gamma	Training Accuracy	Test Accuracy	Accuracy Ratio	%Support
0.0003	0.003	69.2	69.1	99.9	61.6
0.0002	0.004	69.2	69.1	99.9	61.6
0.00009	0.006	69.2	69.1	99.9	61.6
0.00008	0.007	69.2	69.1	99.9	61.6

Table 8 Result of 4 pairs of (C, gamma) around C = 0.5 and gamma = 0.005

C	gamma	Training Accuracy	Test Accuracy	Accuracy Ratio	%Support
0.52	0.003	74.9	72.5	96.8	69.7
0.51	0.004	76.6	72.5	94.6	70.6
0.49	0.006	77.7	72.5	93.3	73.6
0.48	0.007	77.1	71.8	93.1	75.2

Table 9 Result of 4 pairs of (C, gamma) around C = 1 and gamma = 0.005

C	gamma	Training Accuracy	Test Accuracy	Accuracy Ratio	%Support
1.2	0.003	90.9	87.2	95.9	67.2
1.1	0.004	93	87.2	93.8	69.3
0.9	0.006	94.8	82.6	87.1	73.1
0.8	0.007	94	81.2	86.4	73.1

Table 10 Result of 4 pairs of (C, gamma) around C = 10 and gamma = 0.005

C	gamma	Training Accuracy	Test Accuracy	Accuracy Ratio	%Support
12	0.003	100	89.3	89.3	52.9
11	0.004	100	87.9	87.9	57.4
9	0.006	100	87.9	87.9	67.8
8	0.007	100	85.9	85.9	71.4

Table 11 Result of 4 pairs of (C, gamma) around C = 1 and gamma = 0.01

C	gamma	Training Accuracy	Test Accuracy	Accuracy Ratio	%Support
1.2	0.008	98.8	85.2	86.2	75.3
1.1	0.009	99.2	83.2	83.9	78.3
0.9	0.011	98.9	81.2	82.1	81.2
0.8	0.012	97.9	78.5	80.2	83.7

It can be observed that when C = 12 and gamma = 0.003, the training accuracy is perfect, test accuracy is around 90% and percentage of support vectors drastically drops to around 50%. In the next section, the results of both Linear SVM and Radial Kernel SVM will be compared.

11 SVM Comparison

Comparing the Linear SVM and Radial Kernel SVM in Section 7, 8, and 9, it can be found that Linear SVM classifier is better than Radial Kernel SVM based on the tested hyperparameters. Linear SVM has high Training Accuracy, high Test Accuracy, and a low Percentage of Support Vectors. Especially, the Percentage of Support Vector, which need to be lower to have better generalization power, has huge difference (50% vs 20%). Table 12 and Table 13 are the confusion matrices for both training set and test set generated by Linear SVM classifier with C = 6. Table 14 shows the percentages of Support Vector in

each class. Both HIGH and LOW classes provide same number of support vector. Low Percentage of Support Vector represent a better generalization power for the selected SVM.

It can be observed that the best SVM classifier did a great job for predicting both the training set and test set. Also, this classifier predicted the test set with 9.7% for misclassifying LOW as HIGH and 10.9% for misclassifying HIGH as LOW. For stock market traders, it is quite acceptable that they only lose money 1 day out of 10 because there is no one who can always gain the profit. Even Warren Buffet cannot achieve this goal.

Table 12 Confusion Matrix of Training Set

		True Class	
		HIGH	LOW
Prediction	HIGH	94.6	2.7
	LOW	5.4	97.3

Table 13 Confusion Matrix of Test Set

		True Class	
		HIGH	LOW
Prediction	HIGH	89.1	9.7
	LOW	10.9	90.3

Table 14 Percentage of Support Vector in each class

		%Support
Class	HIGH	9.9
	LOW	9.9

12 Graphs

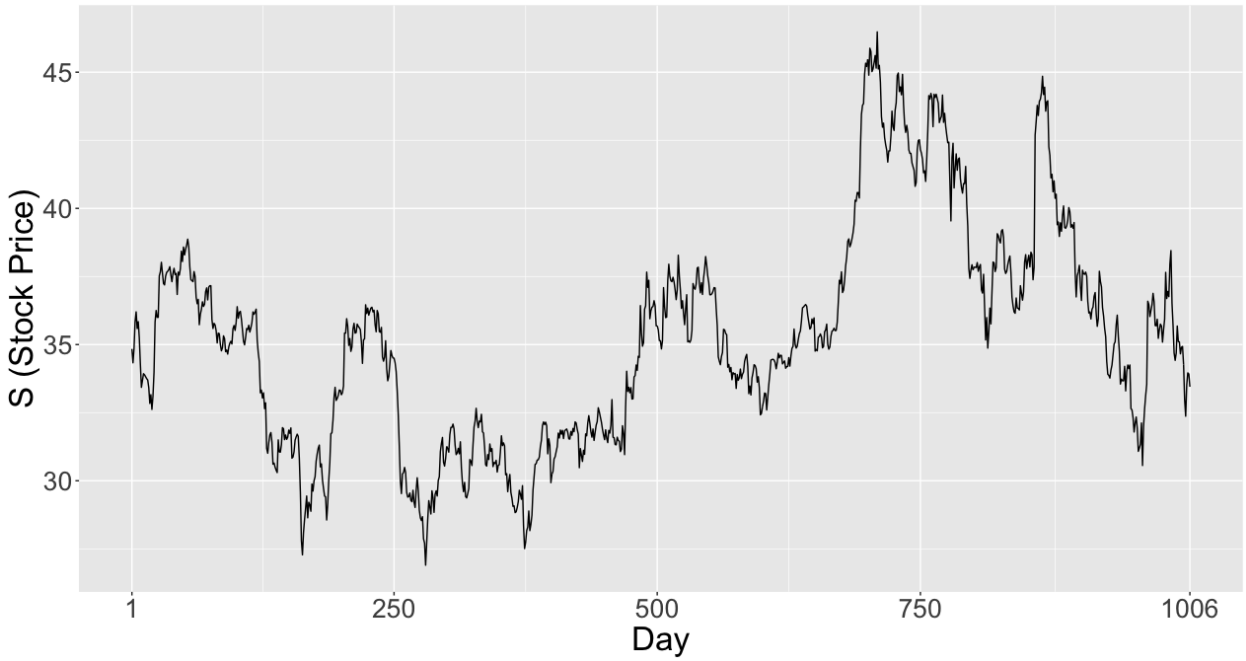

Figure 6 Stock Price at each day

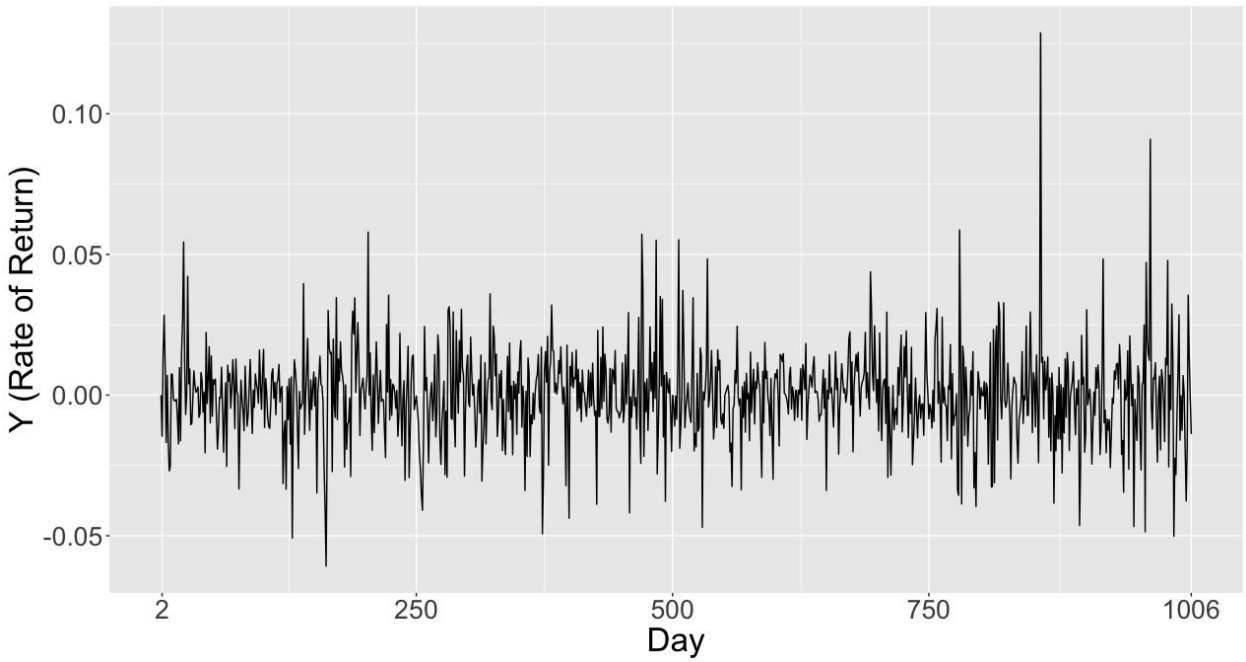

Figure 7 Rate of Return at each day

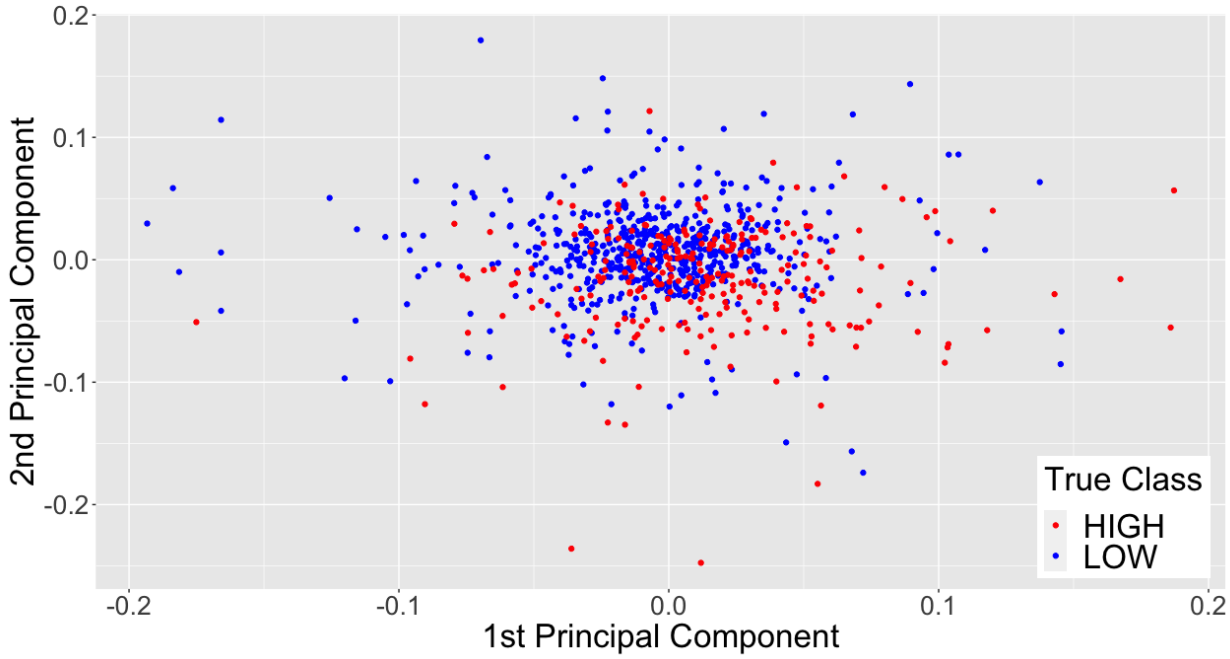

Figure 8 True Class of Training Set under 2D (x-axis: 1st Principal Component, y-axis: 2nd Principal Component)

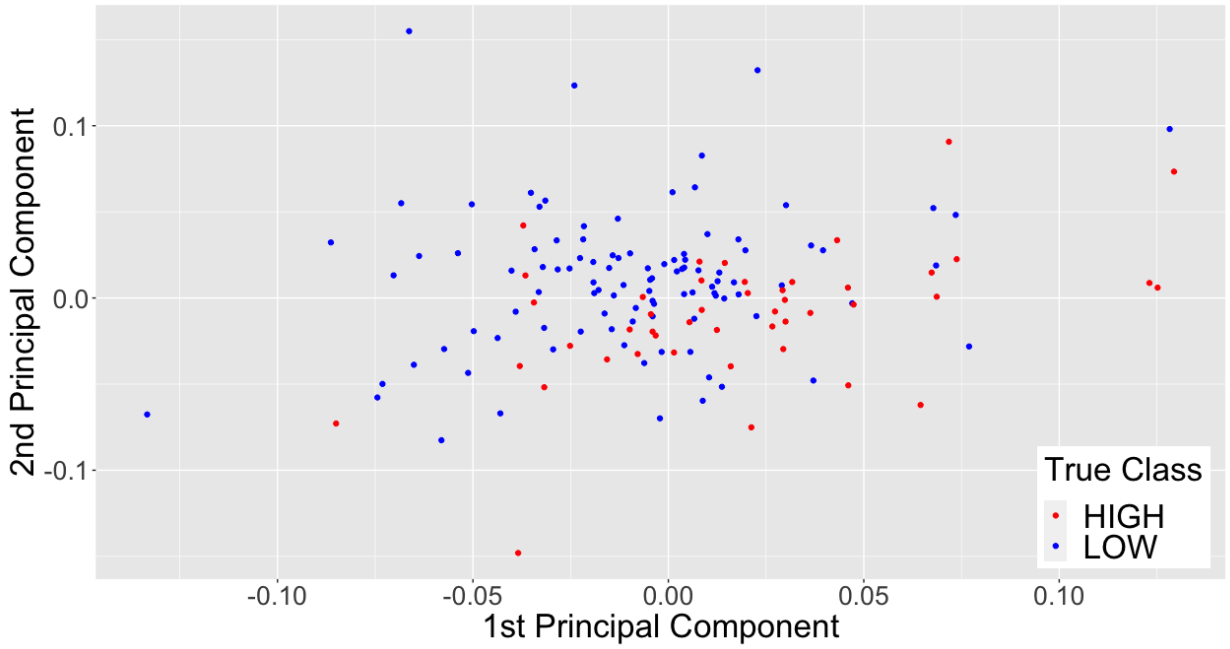

Figure 9 True Class of Test Set under 2D (x-axis: 1st Principal Component, y-axis: 2nd Principal Component)

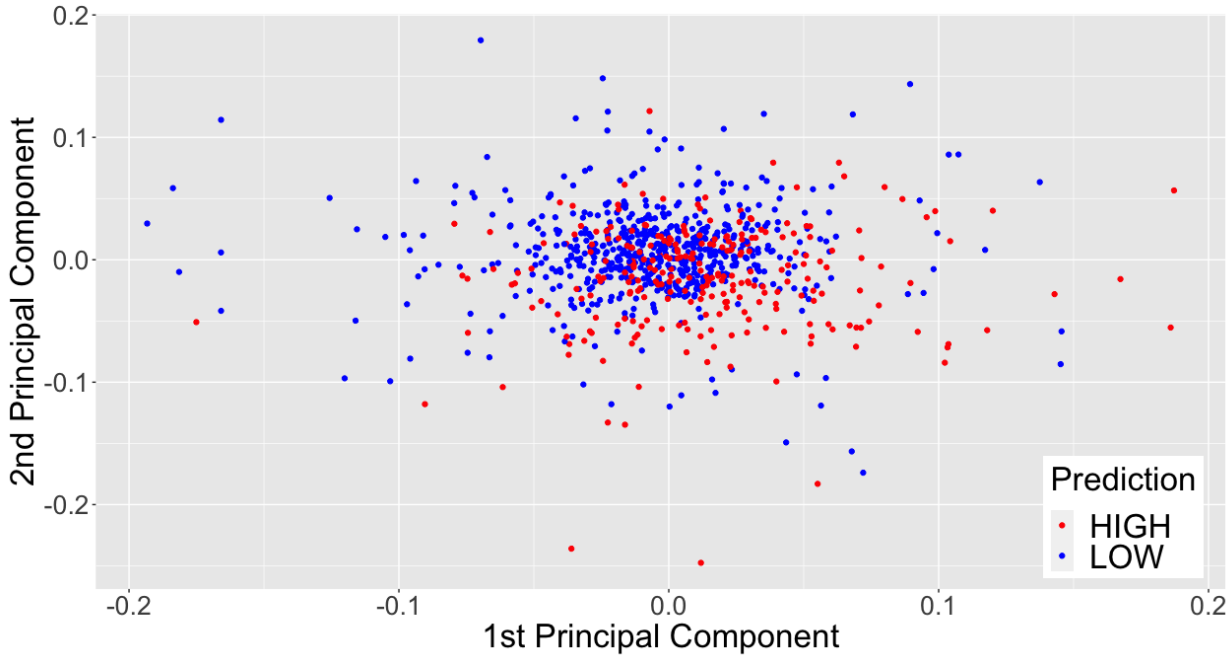

Figure 10 Predicted Class of Training Set under 2D (x-axis: 1st Principal Component, y-axis: 2nd Principal Component)

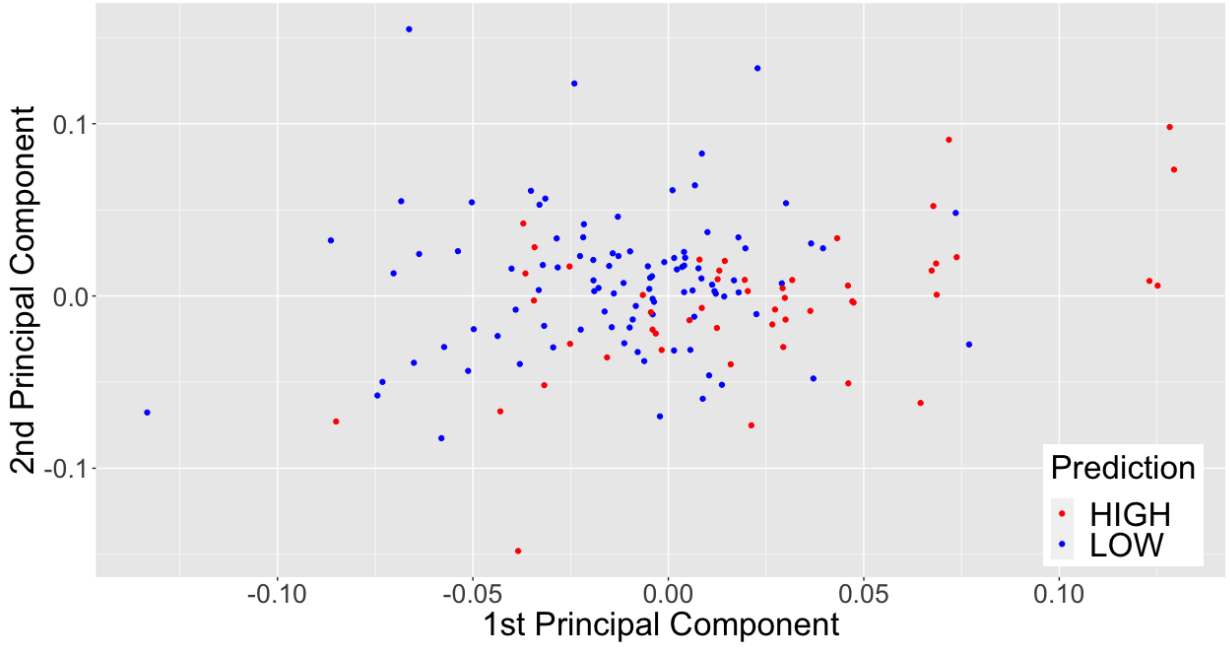

Figure 11 Predicted Class of Test Set under 2D (x-axis: 1st Principal Component, y-axis: 2nd Principal Component)

13 Interpretation

13.1 Graph Comments

Figure 6 shows that Stock Price of the targeted company, General Motor, from day 1 to day 1006. Since the goal is to predict the stock price direction, Figure 7 shows the rate of return i.e., true classes from day 2 to day 1006.

Figure 8 and Figure 9 show the true classes of training and test set respectively. Figure 10 and Figure 11 show the predictions of training and test set respectively. Even though we have used PCA to conduct the feature reduction, there are still 118 principal components. It is impossible to demonstrate a graph of this dimension; therefore, we only use 1st principal component as x-axis and 2nd principal component as y-axis to represent the graph. Noted that the first two principal components only contain 8.26% of information of the original 200 features.

Basically, we can observe that for the true class in Figure 8 and 9, the "LOW" class data points spread wider than the "HIGH" class data points for both training set and test set. Figure 10 and 11 for prediction shows same pattern as true class. Thus, we can guess that both the training accuracy and test accuracy will be high which is consistent with the result.

13.2 Support Vector

Figure 12 shows the support vectors (circled in green) of the best SVM found in Section 11. It is not so meaningful to visually inspect the support vector on a 2D plot when the true dimension is 118. However, it still can be observed that most of the support vectors are concentrated around the original point.

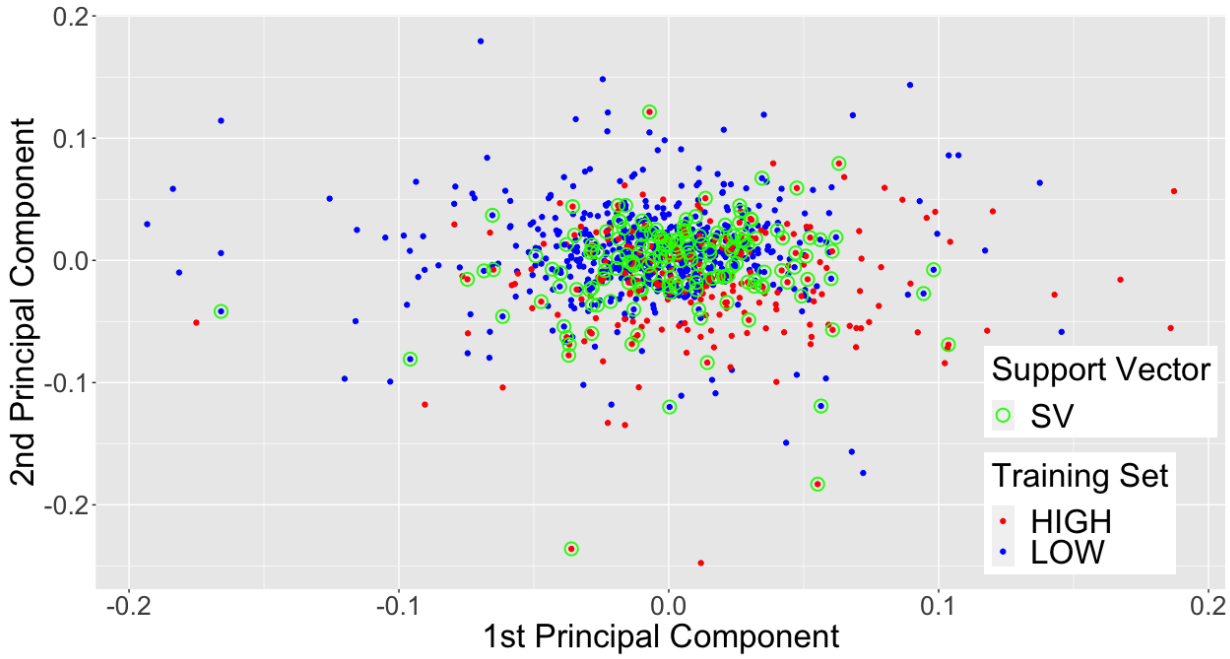

Figure 12 Support Vectors of Best SVM (Linear SVM with cost value equals to 6)

13.3 Misclassified Cases

As mentioned in Section 11, around 10% of data points in each class are misclassified. Also, we have 103 cases as HIGH class and 46 cases as LOW class in test set. It is difficult to identify the result of misclassification for Figure 13. However, we can guess that most of the misclassified cases in Figure 14 are close to the original point which is the location that most of the Support Vectors located.

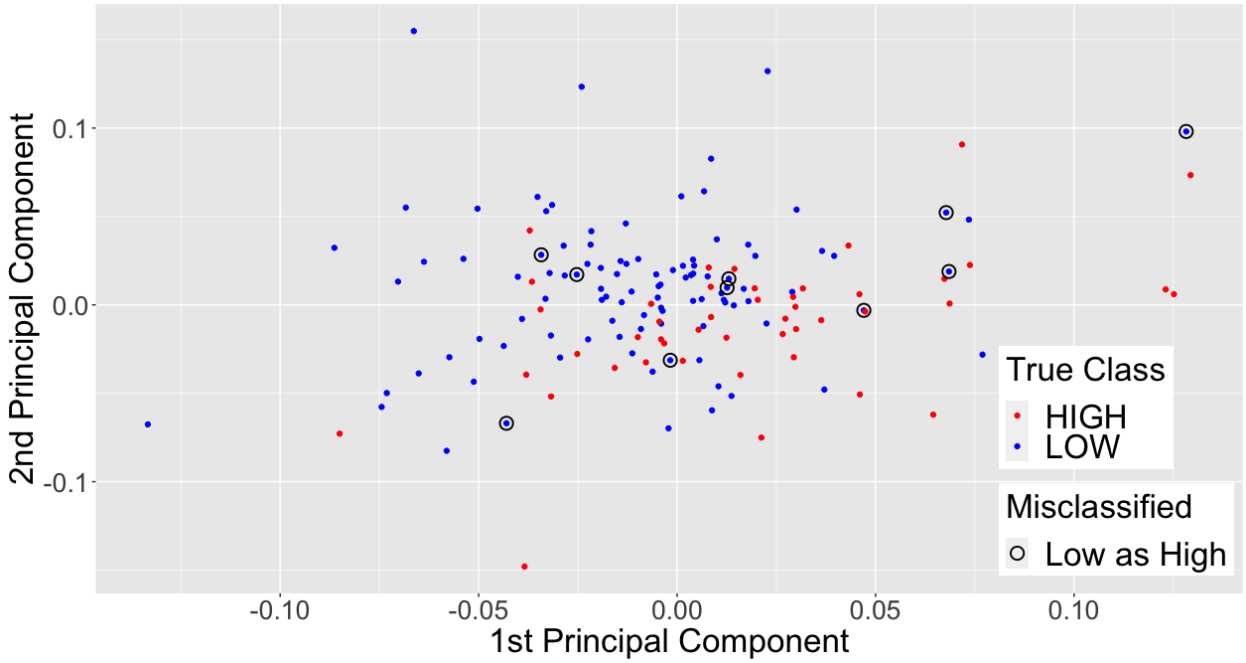

Figure 13 Misclassified Data Points circled in green (True Class = LOW but Prediction = HIGH)

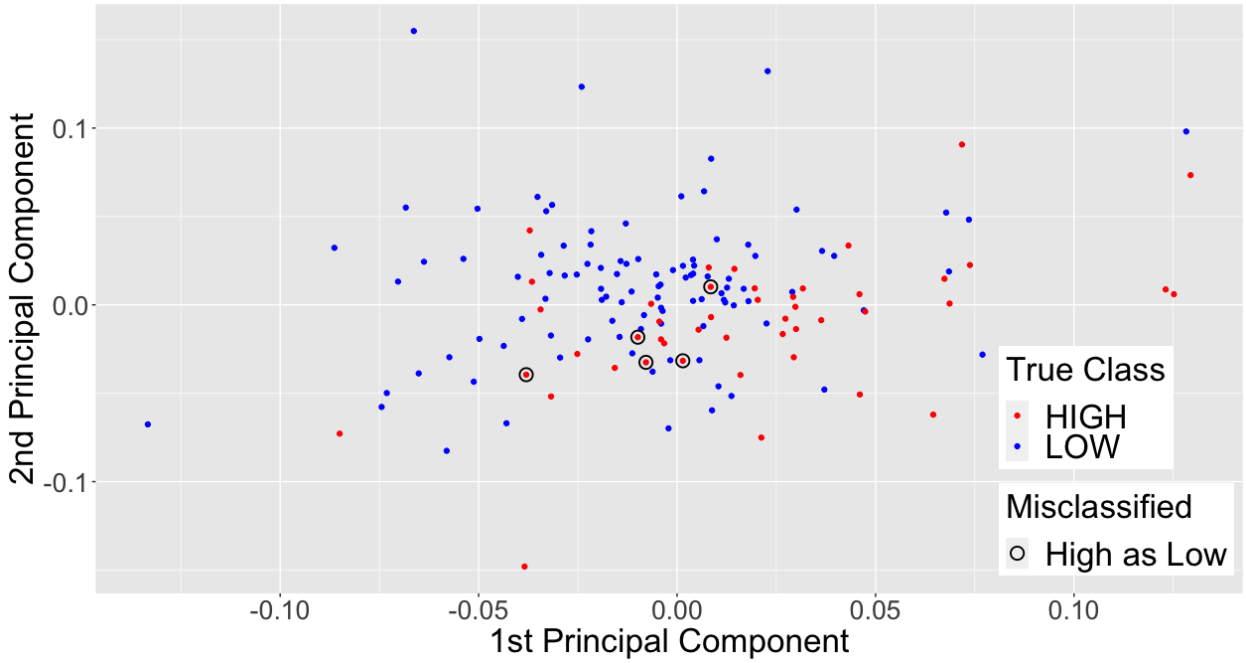

Figure 14 Misclassified Data Points circled in green (True Class = HIGH but Prediction = LOW)

14 Computation Time

Since several two different SVM are performed, it is wise to compare the computation time for each of them to check if it is worthy to have a longer computation time to obtain a higher accuracy i.e., better classifier. Table 15 shows all the candidate classifier used in Section 7 and 9. It can be observed that for all the SVM including Linear and Radial Kernel,

computation time increases as C goes higher. However, gamma is less correlated with the computation time compared to the C values. But it does require more time to tune two hyperparameter than one to find a best model.

Table 15 Computation Time (a) Linear SVM, (b) Radial Kernel SVM with gamma = 0.005, (c) Radial Kernel SVM with gamma = 0.01, (d) Radial Kernel SVM with gamma = 0.1, (e) Radial Kernel SVM with gamma = 1

C	Computing Time (s)
0.01	0.47
0.25	0.22
0.5	0.21
0.75	0.22
1	0.55
2	0.57
2.5	0.38
5	1.22
7.5	1.29
10	3.29

(a)

C	Computing Time (s)
0.0001	0.71
0.001	0.24
0.01	0.29
0.1	0.28
0.5	0.27
0.75	0.29
1	0.28
5	0.28
10	0.28
20	0.28

(b)

C	Computing Time (s)
0.0001	0.25
0.001	0.25
0.01	0.28
0.1	0.5
0.5	0.29
0.75	0.29
1	0.29
5	0.31
10	0.31
20	0.3

(c)

C	Computing Time (s)
0.0001	0.25
0.001	0.24
0.01	0.34
0.1	0.36
0.5	0.35
0.75	0.34
1	0.34
5	0.34
10	0.34
20	0.35

(d)

C	Computing Time (s)
0.0001	0.24
0.001	0.33
0.01	0.34
0.1	0.35
0.5	0.34
0.75	0.34
1	0.35
5	0.35
10	0.35
20	0.35

(e)

R Script

```
rm(list = ls())
#================================================================================
library(e1071)
library(ggplot2)
library(smotefamily)
library(flexclust)
library(caret)
#================================================================================

# Part 1 Data Set
wd = "/Users/jerrychien/Desktop/OneDrive - University Of Houston/6350 - Stati
stical Learning and Data Mining/HW/HW 4/Stock Data/"
comp = c("WMT", "AMZN", "AAPL", "CVS", "XOM", "UNH", "BRK-A", "MCK", "ABC", "
GOOG",
         "T", "CI", "F", "COST", "FDX", "CVX", "CAH", "MSFT", "JPM", "GM")
for (i in 1:20){
  temp = read.csv(paste(wd, comp[i], ".csv", sep = ""))[, 5]
  temp = cbind(c(1:length(temp)), temp)
  colnames(temp) = c("T", "S")
  assign(comp[i], as.data.frame(temp))
}

# Part 2 Pre-Processing
for (i in 1:20){
  temp = get(comp[i])
  temp["Y"] = 0
  for (j in 2:1006){
    temp[j, "Y"] = (temp[j, "S"] - temp[j - 1, "S"]) / temp[j - 1, "S"]
  }
  assign(comp[i], temp)
}

# Part 3 Create cases and features vectors
df = data.frame(matrix(ncol = 200, nrow = 995))
colnames(df) = c(paste("X", c(1:200), sep = ""))
for (i in 1:20){
  temp = get(comp[i])
  for (j in 2:996){
    df[j - 1, ((i - 1)*10 + 1):(i*10)] = temp[c(j:(j+9)), "Y"]
  }
}

# Part 4 Create two disjoint class of cases
comp20 = get(comp[20])
TRUC = as.factor(ifelse(comp20[c(11:1005), "Y"] >= 0.006, "HIGH", "LOW"))
class_size = data.frame()
for (i in c("LOW", "HIGH")){
  class_size["Number", i] = table(TRUC)[i]
```

```
  class_size["Percentage", i] = round(prop.table(table(TRUC))[i] * 100, 1)
}

# Part 5 PCA Analysis
CORR = cor(df)
E = eigen(CORR)
L = E$values
W = E$vector
ggplot() +
  geom_point(aes(x = c(1:200), y = L)) +
  ylab("Eigenvalue") + xlab("k") +
  scale_y_continuous(expand = c(0, 0.25)) + scale_x_continuous(expand = c(0,
2)) +
  theme(text = element_text(size = 25), plot.margin = margin(10, 15, 10, 10,
"point"))

PEV = NULL
for (m in 1:200){
  PEV[m] = sum(L[c(1:m)]) / 200 * 100
}

h = sum(PEV < 90) + 1 # h to have PEV >= 90%

ggplot() +
  geom_point(aes(x = c(1:200), y = PEV)) +
  geom_segment(aes(x = h, xend = h, y = 0, yend = 90), size = 1, colour="blue
") +
  geom_segment(aes(x = 0, xend = h, y = 90, yend = 90), size = 1, colour="blu
e") +
  geom_text(aes(x = h + 9, y = 3, label = paste("k =", h)), size = 7, color =
 "blue") +
  geom_text(aes(x = 15, y = 93, label = "PEV >= 90%"), size = 7, color = "blu
e") +
  ylab("PEV (%)") + xlab("k") +
  scale_y_continuous(expand = c(0, 1)) + scale_x_continuous(expand = c(0, 2))
 +
  theme(text = element_text(size = 25), plot.margin = margin(10, 15, 10, 10,
"point"))

U =  as.data.frame(as.matrix(df) %*% as.matrix(W))[, c(1:h)]

final_df = cbind(U, TRUC)

# Part 6 Training and Test Sets
set.seed(20211125)
training_set = NULL
test_set = NULL
for (i in as.factor(c("LOW", "HIGH"))){
  temp = subset(final_df, TRUC == i)
  sample_number = sample(x = dim(temp)[1], size = round(dim(temp)[1] * 0.85))
```

```
  training_set = rbind(training_set, temp[sample_number, ])
  test_set = rbind(test_set, temp[-sample_number, ])
}

size_test_training = data.frame()
for (i in c("training_set", "test_set")){
  size_test_training["LOW", i] = dim(subset(get(i), TRUC == "LOW"))[1]
  size_test_training["HIGH", i] = dim(subset(get(i), TRUC == "HIGH"))[1]
  size_test_training["Total", i] = dim(get(i))[1]
}

# Part 7 Linear SVM
cost_list_linear = c(0.01, 0.25, 0.5, 0.75, 1, 2, 2.5, 5, 7.5, 10)
result_linear = data.frame(matrix(nrow = length(cost_list_linear), ncol = 5))
colnames(result_linear) = c("Cost", "AccTrain", "AccTest", "AccTest/AccTrain
", "%Support")
for (i in 1:length(cost_list_linear)){
  start_time = Sys.time()
  linear_svm = svm(x = training_set[, -119], y = training_set[, 119], kernel
= "linear", cost = cost_list_linear[i])
  end_time = Sys.time()
  linear_pred_training = predict(linear_svm, training_set[, -119])
  linear_pred_test = predict(linear_svm, test_set[, -119])
  AccTrain = round(mean(linear_pred_training == training_set[, 119]) * 100,
1)
  AccTest = round(mean(linear_pred_test == test_set[, 119]) * 100, 1)
  result_linear[i, "Cost"] = linear_svm$cost
  result_linear[i, "AccTrain"] = AccTrain
  result_linear[i, "AccTest"] = AccTest
  result_linear[i, "AccTest/AccTrain"] = round((AccTest / AccTrain) * 100, 1)
  result_linear[i, "%Support"] = round((sum(linear_svm$nSV) / dim(training_se
t)[1]) * 100, 1)
  result_linear[i, "Computing Time"] = round(end_time - start_time, 2)
}

# Part 8 Plot
ggplot() +
  geom_line(aes(x = cost_list_linear, y = unlist(result_linear["AccTrain"]),
col = "Acc Train"), size = 1) +
  geom_line(aes(x = cost_list_linear, y = unlist(result_linear["AccTest"]), c
ol = "Acc Test"), size = 1) +
  geom_line(aes(x = cost_list_linear, y = unlist(result_linear["AccTest/AccTr
ain"]), col = "Acc Ratio"), size = 1) +
  geom_line(aes(x = cost_list_linear, y = unlist(result_linear["%Support"]),
col = "%Support"), size = 1) +
  ylab("%") + xlab("Cost") +
  scale_color_manual(name = "Legend", values = c("Acc Train" = "green", "Acc
Test" = "blue", "Acc Ratio" = "red", "%Support" = "black")) +
  scale_x_continuous(breaks = cost_list_linear, expand = c(0.01, 0.01)) +
  theme(text = element_text(size = 25), legend.position = c(0.9, 0.5), legen
```

```
d.text = element_text(size = 25), legend.title = element_text(size = 25)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

cost_list_linear_new = c(0.1, 0.8, 3, 3.5, 4, 6, 7, 8, 9, 15, 20, 25)
result_linear_new = data.frame(matrix(nrow = length(cost_list_linear_new), nc
ol = 5))
colnames(result_linear_new) = c("Cost", "AccTrain", "AccTest", "AccTest/AccTr
ain", "%Support")
for (i in 1:length(cost_list_linear_new)){
  start_time = Sys.time()
  linear_svm_new = svm(x = training_set[, -119], y = training_set[, 119], ker
nel = "linear", cost = cost_list_linear_new[i])
  end_time = Sys.time()
  linear_pred_training_new = predict(linear_svm_new, training_set[, -119])
  linear_pred_test_new = predict(linear_svm_new, test_set[, -119])
  AccTrain = round(mean(linear_pred_training_new == training_set[, 119]) * 10
0, 1)
  AccTest = round(mean(linear_pred_test_new == test_set[, 119]) * 100, 1)
  result_linear_new[i, "Cost"] = linear_svm_new$cost
  result_linear_new[i, "AccTrain"] = AccTrain
  result_linear_new[i, "AccTest"] = AccTest
  result_linear_new[i, "AccTest/AccTrain"] = round((AccTest / AccTrain) * 10
0, 1)
  result_linear_new[i, "%Support"] = round((sum(linear_svm_new$nSV) / dim(tra
ining_set)[1]) * 100, 1)
  result_linear_new[i, "Computing Time"] = round(end_time - start_time, 2)
}

ggplot() +
  geom_line(aes(x = cost_list_linear_new, y = unlist(result_linear_new["AccTr
ain"]), col = "Acc Train"), size = 1) +
  geom_line(aes(x = cost_list_linear_new, y = unlist(result_linear_new["AccTe
st"]), col = "Acc Test"), size = 1) +
  geom_line(aes(x = cost_list_linear_new, y = unlist(result_linear_new["AccTe
st/AccTrain"]), col = "Acc Ratio"), size = 1) +
  geom_line(aes(x = cost_list_linear_new, y = unlist(result_linear_new["%Supp
ort"]), col = "%Support"), size = 1) +
  ylab("%") + xlab("C") +
  scale_color_manual(name = "Legend", values = c("Acc Train" = "green", "Acc
Test" = "blue", "Acc Ratio" = "red", "%Support" = "black")) +
  scale_x_continuous(breaks = cost_list_linear_new, expand = c(0.01, 0)) +
  theme(text = element_text(size = 25), legend.position = c(0.9, 0.5), legen
d.text = element_text(size = 25), legend.title = element_text(size = 25)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Part 9 Radial Kernel
## Part 9.1 Hilbert Distance
training_HIGH = subset(training_set, TRUC == "HIGH")
training_LOW = subset(training_set, TRUC == "LOW")
gamma_list = c(0.001, 0.005, 10, 100)
```

```
for (i in 1:length(gamma_list)){
  SIM = exp(-gamma_list[i] * (dist2(training_HIGH[, -119], training_LOW[, -11
9]))^2)
  Hdist = sqrt(2 - 2*SIM)
  assign(paste("H_dist_", i, sep = ""), c(Hdist))
}

ggplot() +
  geom_histogram(aes(x = H_dist_1, fill = "0.001"), alpha = 0.5, bins = 50) +
  geom_histogram(aes(x = H_dist_2, fill = "0.005"), alpha = 0.5, bins = 50) +
  geom_histogram(aes(x = H_dist_3, fill = "10"), alpha = 0.5, bins = 50) +
  geom_histogram(aes(x = H_dist_4, fill = "100"), alpha = 0.5, bins = 50) +
  ggtitle("Histogram of H Distance with different Gamma") +
  scale_fill_manual(name = "Gamma", values = c("0.001" = "green", "0.005" = "
blue", "10" = "red", "100" = "black")) +
  scale_x_continuous(breaks = c(0, 0.5, 1, 1.414)) +
  theme(text = element_text(size = 25), legend.position = c(0.85, 0.8), legen
d.text = element_text(size = 25), legend.title = element_text(size = 25), plo
t.title = element_text(hjust = 0.5)) +
  ylab("Density") + xlab("H Distance")

gamma_list_radial = c(0.005, 0.01, 0.1, 1)

## Part 9.2 Error Weight Coefficient
cost_list_radial = c(0.0001, 0.001, 0.01, 0.1, 0.5, 0.75, 1, 5, 10, 20)

## Part 9.3 Apply Radial Kernel SVM
for (i in 1:length(gamma_list_radial)){
  temp = data.frame()
  for (j in 1:length(cost_list_radial)){
    start_time = Sys.time()
    radial_svm = svm(x = training_set[, -119], y = training_set[, 119], kerne
l = "radial", gamma = gamma_list_radial[i], cost = cost_list_radial[j])
    end_time = Sys.time()
    radial_pred_training = predict(radial_svm, training_set[, -119])
    radial_pred_test = predict(radial_svm, test_set[, -119])
    AccTrain = round(mean(radial_pred_training == training_set[, 119]) * 100,
 1)
    AccTest = round(mean(radial_pred_test == test_set[, 119]) * 100, 1)
    temp[j, "Cost"] = radial_svm$cost
    temp[j, "AccTrain"] = AccTrain
    temp[j, "AccTest"] = AccTest
    temp[j, "AccTest/AccTrain"] = round((AccTest / AccTrain) * 100, 1)
    temp[j, "%Support"] = round((sum(radial_svm$nSV) / dim(training_set)[1])
* 100, 1)
    temp[j, "Computing Time"] = round(end_time - start_time, 2)
  }
  assign(paste("result_radial_gamma_", i, sep = ""), temp)
}
```

```r
# Part 10
## First pair (C = 0.0001, Gamma = 0.005) Reason: Lowest % of Support Vector
with 100% Accuracy Ratio
gamma_pair_1 = c(0.003, 0.004, 0.005, 0.006, 0.007)
C_pair_1 = c(0.0003, 0.0002, 0.0001, 0.00009, 0.00008)
pair_1 = data.frame()
for (i in 1:5){
  pair_1_svm = svm(x = training_set[, -119], y = training_set[, 119], kernel
= "radial", gamma = gamma_pair_1[i], cost = C_pair_1[i])
  radial_pred_training = predict(pair_1_svm, training_set[, -119])
  radial_pred_test = predict(pair_1_svm, test_set[, -119])
  AccTrain = round(mean(radial_pred_training == training_set[, 119]) * 100,
1)
  AccTest = round(mean(radial_pred_test == test_set[, 119]) * 100, 1)
  pair_1[i, "Cost"] = pair_1_svm$cost
  pair_1[i, "Gamma"] = pair_1_svm$gamma
  pair_1[i, "AccTrain"] = AccTrain
  pair_1[i, "AccTest"] = AccTest
  pair_1[i, "AccTest/AccTrain"] = round((AccTest / AccTrain) * 100, 1)
  pair_1[i, "%Support"] = round((sum(pair_1_svm$nSV) / dim(training_set)[1])
* 100, 1)
  pair_1[i, "Computing Time"] = round(end_time - start_time, 2)
}

## Second pair (C = 0.5, Gamma = 0.005) Reason: Both Test Accuracy and Traini
ng Accuracy start to increase comparing lower C at same gamma.
gamma_pair_2 = c(0.003, 0.004, 0.005, 0.006, 0.007)
C_pair_2 = c(0.52, 0.51, 0.5, 0.49, 0.48)
pair_2 = data.frame()
for (i in 1:5){
  pair_2_svm = svm(x = training_set[, -119], y = training_set[, 119], kernel
= "radial", gamma = gamma_pair_2[i], cost = C_pair_2[i])
  radial_pred_training = predict(pair_2_svm, training_set[, -119])
  radial_pred_test = predict(pair_2_svm, test_set[, -119])
  AccTrain = round(mean(radial_pred_training == training_set[, 119]) * 100,
1)
  AccTest = round(mean(radial_pred_test == test_set[, 119]) * 100, 1)
  pair_2[i, "Cost"] = pair_2_svm$cost
  pair_2[i, "Gamma"] = pair_2_svm$gamma
  pair_2[i, "AccTrain"] = AccTrain
  pair_2[i, "AccTest"] = AccTest
  pair_2[i, "AccTest/AccTrain"] = round((AccTest / AccTrain) * 100, 1)
  pair_2[i, "%Support"] = round((sum(pair_2_svm$nSV) / dim(training_set)[1])
* 100, 1)
  pair_2[i, "Computing Time"] = round(end_time - start_time, 2)
}

## Third pair (C = 1, Gamma = 0.005) Reason: Highest Test Accuracy over all p
airs
gamma_pair_3 = c(0.003, 0.004, 0.005, 0.006, 0.007)
```

```
C_pair_3 = c(1.2, 1.1, 1, 0.9, 0.8)
pair_3 = data.frame()
for (i in 1:5){
  pair_3_svm = svm(x = training_set[, -119], y = training_set[, 119], kernel
= "radial", gamma = gamma_pair_3[i], cost = C_pair_3[i])
  radial_pred_training = predict(pair_3_svm, training_set[, -119])
  radial_pred_test = predict(pair_3_svm, test_set[, -119])
  AccTrain = round(mean(radial_pred_training == training_set[, 119]) * 100,
1)
  AccTest = round(mean(radial_pred_test == test_set[, 119]) * 100, 1)
  pair_3[i, "Cost"] = pair_3_svm$cost
  pair_3[i, "Gamma"] = pair_3_svm$gamma
  pair_3[i, "AccTrain"] = AccTrain
  pair_3[i, "AccTest"] = AccTest
  pair_3[i, "AccTest/AccTrain"] = round((AccTest / AccTrain) * 100, 1)
  pair_3[i, "%Support"] = round((sum(pair_3_svm$nSV) / dim(training_set)[1])
* 100, 1)
  pair_3[i, "Computing Time"] = round(end_time - start_time, 2)
}

## Fourth pair (C = 10, Gamma = 0.005) Reason: High Test Accuracy with perfec
t Training Accuracy and lower % of Support Vector
gamma_pair_4 = c(0.003, 0.004, 0.005, 0.006, 0.007)
C_pair_4 = c(12, 11, 10, 9, 8)
pair_4 = data.frame()
for (i in 1:5){
  pair_4_svm = svm(x = training_set[, -119], y = training_set[, 119], kernel
= "radial", gamma = gamma_pair_4[i], cost = C_pair_4[i])
  radial_pred_training = predict(pair_4_svm, training_set[, -119])
  radial_pred_test = predict(pair_4_svm, test_set[, -119])
  AccTrain = round(mean(radial_pred_training == training_set[, 119]) * 100,
1)
  AccTest = round(mean(radial_pred_test == test_set[, 119]) * 100, 1)
  pair_4[i, "Cost"] = pair_4_svm$cost
  pair_4[i, "Gamma"] = pair_4_svm$gamma
  pair_4[i, "AccTrain"] = AccTrain
  pair_4[i, "AccTest"] = AccTest
  pair_4[i, "AccTest/AccTrain"] = round((AccTest / AccTrain) * 100, 1)
  pair_4[i, "%Support"] = round((sum(pair_4_svm$nSV) / dim(training_set)[1])
* 100, 1)
  pair_4[i, "Computing Time"] = round(end_time - start_time, 2)
}

## Fifth pair (C = 1, Gamma = 0.01) Reason: Almost perfect Training Accuracy
but low Accuracy Ratio
gamma_pair_5 = c(0.008, 0.009, 0.01, 0.011, 0.012)
C_pair_5 = c(1.2, 1.1, 1, 0.9, 0.8)
pair_5 = data.frame()
for (i in 1:5){
  pair_5_svm = svm(x = training_set[, -119], y = training_set[, 119], kernel
```

```
= "radial", gamma = gamma_pair_5[i], cost = C_pair_5[i])
  radial_pred_training = predict(pair_5_svm, training_set[, -119])
  radial_pred_test = predict(pair_5_svm, test_set[, -119])
  AccTrain = round(mean(radial_pred_training == training_set[, 119]) * 100,
1)
  AccTest = round(mean(radial_pred_test == test_set[, 119]) * 100, 1)
  pair_5[i, "Cost"] = pair_5_svm$cost
  pair_5[i, "Gamma"] = pair_5_svm$gamma
  pair_5[i, "AccTrain"] = AccTrain
  pair_5[i, "AccTest"] = AccTest
  pair_5[i, "AccTest/AccTrain"] = round((AccTest / AccTrain) * 100, 1)
  pair_5[i, "%Support"] = round((sum(pair_5_svm$nSV) / dim(training_set)[1])
* 100, 1)
  pair_5[i, "Computing Time"] = round(end_time - start_time, 2)
}

# Part 11
best_svm = svm(x = training_set[, -119], y = training_set[, 119], kernel = "l
inear", cost = 6)
best_pred_training = predict(best_svm, training_set[, -119])
best_pred_test = predict(best_svm, test_set[, -119])
conf_training= round(prop.table(confusionMatrix(data = best_pred_training, re
ference = training_set[, 119])$table, margin = 2) * 100, 1)
conf_test= round(prop.table(confusionMatrix(data = best_pred_test, reference
= test_set[, 119])$table, margin = 2) * 100, 1)

sv_percentage = data.frame()
sv_percentage["HIGH", "%Support"] = round(best_svm$nSV[1] / dim(training_set)
[1] * 100, 1)
sv_percentage["LOW", "%Support"] = round(best_svm$nSV[2] / dim(training_set)
[1] * 100, 1)

# Part 12
ggplot() +
  geom_line(aes(x = comp20[, "T"], y = comp20[, "S"])) +
  scale_x_continuous(breaks = c(1, 250, 500, 750, 1006)) +
  ylab("S (Stock Price)") + xlab("Day") +
  theme(text = element_text(size = 25))

ggplot() +
  geom_line(aes(x = comp20[, "T"], y = comp20[, "Y"])) +
  scale_x_continuous(breaks = c(2, 250, 500, 750, 1006)) +
  ylab("Y (Rate of Return)") + xlab("Day") +
  theme(text = element_text(size = 25))

ggplot() +
  geom_point(aes(x = training_set[, 1], y = training_set[, 2], col = training
_set[, 119])) +
  scale_color_manual(name ="True Class", values = c("HIGH" = "red", "LOW" = "
blue")) +
```

```
  ylab("2nd Principal Component") + xlab("1st Principal Component") +
  theme(text = element_text(size = 25)) +
  theme(legend.position = c(0.91, 0.125), legend.text = element_text(size = 2
5), legend.title = element_text(size = 25), plot.title = element_text(hjust =
 0.5), plot.margin = margin(10, 15, 10, 10, "point"))

ggplot() +
  geom_point(aes(x = test_set[, 1], y = test_set[, 2], col = test_set[, 11
9])) +
  scale_color_manual(name ="True Class", values = c("HIGH" = "red", "LOW" = "
blue")) +
  ylab("2nd Principal Component") + xlab("1st Principal Component") +
  theme(text = element_text(size = 25)) +
  theme(legend.position = c(0.91, 0.125), legend.text = element_text(size = 2
5), legend.title = element_text(size = 25), plot.title = element_text(hjust =
 0.5), plot.margin = margin(10, 15, 10, 10, "point"))

ggplot() +
  geom_point(aes(x = training_set[, 1], y = training_set[, 2], col = best_pre
d_training)) +
  scale_color_manual(name ="Prediction", values = c("HIGH" = "red", "LOW" = "
blue")) +
  ylab("2nd Principal Component") + xlab("1st Principal Component") +
  theme(text = element_text(size = 25)) +
  theme(legend.position = c(0.91, 0.125), legend.text = element_text(size = 2
5), legend.title = element_text(size = 25), plot.title = element_text(hjust =
 0.5), plot.margin = margin(10, 15, 10, 10, "point"))

ggplot() +
  geom_point(aes(x = test_set[, 1], y = test_set[, 2], col = best_pred_test))
 +
  scale_color_manual(name ="Prediction", values = c("HIGH" = "red", "LOW" = "
blue")) +
  ylab("2nd Principal Component") + xlab("1st Principal Component") +
  theme(text = element_text(size = 25)) +
  theme(legend.position = c(0.91, 0.125), legend.text = element_text(size = 2
5), legend.title = element_text(size = 25), plot.title = element_text(hjust =
 0.5), plot.margin = margin(10, 15, 10, 10, "point"))

# Part 13
## Part 13.2
best_svm_sv = training_set[rownames(best_svm$SV), ]

ggplot() +
  geom_point(aes(x = training_set[, 1], y = training_set[, 2], col = best_pre
d_training)) +
  geom_point(aes(x = best_svm_sv[, 1], y = best_svm_sv[, 2], shape = "SV"), s
ize = 4, col ="green", stroke = 1) +
  scale_color_manual(name = "Training Set", values = c("HIGH" = "red", "LOW"
= "blue")) +
  scale_shape_manual(name = "Support Vector", values = c("SV" = 1)) +
```

```
  ylab("2nd Principal Component") + xlab("1st Principal Component") +
  theme(text = element_text(size = 25)) +
  theme(legend.position = c(0.89, 0.225), legend.text = element_text(size = 2
5), legend.title = element_text(size = 25), plot.title = element_text(hjust =
 0.5), plot.margin = margin(10, 15, 10, 10, "point"))

## Part 13.3
TRUC_pred = cbind(test_set[119], best_pred_test)
mis_low_as_high = test_set[rownames(subset(TRUC_pred, TRUC == "LOW" & TRUC !=
 best_pred_test)), ]

ggplot() +
  geom_point(aes(x = test_set[, 1], y = test_set[, 2], col = test_set[, 11
9])) +
  geom_point(aes(x = mis_low_as_high[, 1], y = mis_low_as_high[, 2], shape =
"Low as High"), size = 4, col ="black", stroke = 1) +
  scale_color_manual(name = "True Class", values = c("HIGH" = "red", "LOW" =
"blue")) +
  scale_shape_manual(name = "Misclassified", values = c("Low as High" = 1)) +
  ylab("2nd Principal Component") + xlab("1st Principal Component") +
  theme(text = element_text(size = 25)) +
  theme(legend.position = c(0.89, 0.225), legend.text = element_text(size = 2
5), legend.title = element_text(size = 25), plot.title = element_text(hjust =
 0.5))

## Part 13.4
mis_high_as_low = test_set[rownames(subset(TRUC_pred, TRUC == "HIGH" & TRUC !
= best_pred_test)), ]

ggplot() +
  geom_point(aes(x = test_set[, 1], y = test_set[, 2], col = test_set[, 11
9])) +
  geom_point(aes(x = mis_high_as_low[, 1], y = mis_high_as_low[, 2], shape =
"High as Low"), size = 4, col ="black", stroke = 1) +
  scale_color_manual(name = "True Class", values = c("HIGH" = "red", "LOW" =
"blue")) +
  scale_shape_manual(name = "Misclassified", values = c("High as Low" = 1)) +
  ylab("2nd Principal Component") + xlab("1st Principal Component") +
  theme(text = element_text(size = 25)) +
  theme(legend.position = c(0.89, 0.225), legend.text = element_text(size = 2
5), legend.title = element_text(size = 25), plot.title = element_text(hjust =
 0.5))

# Part 14
linear_computing = result_linear[, c("Cost", "Computing Time")]
radial_computing_1 = result_radial_gamma_1[, c("Cost", "Computing Time")]
radial_computing_2 = result_radial_gamma_2[, c("Cost", "Computing Time")]
radial_computing_3 = result_radial_gamma_3[, c("Cost", "Computing Time")]
radial_computing_4 = result_radial_gamma_4[, c("Cost", "Computing Time")]
```