Stevens Institute of Technology

FA590. Assignment #4.

This content is protected and may not be shared, uploaded, or distributed

Enter Your Name Here, or "Anonymous" if you want to remain anonymous..

2022-04-30

I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination. I further pledge that I have not copied any material from a book, article, the Internet or any other source except where I have expressly cited the source.

By filling out the following fields, you are signing this pledge. No assignment will get credit without being pledged.

Name:Zongqi Shen

CWID:10479206

Date:04/20/2022

Instructions

When you have completed the assignment, knit the document into a PDF file, and upload *both* the .pdf and .Rmd files to Canvas.

Note that you must have LaTeX installed in order to knit the equations below. If you do not have it installed, simply delete the questions below.

```
CWID = 10479206 #Place here your Campus wide ID number, this will personalize
#your results, but still maintain the reproducible nature of using seeds.
#If you ever need to reset the seed in this assignment, use this as your seed
#Papers that use -1 as this CWID variable will earn 0's so make sure you change
#this value before you submit your work.
personal = CWID %% 10000
set.seed(personal)
```

Question 1:

In this assignment, you will be required to find a set of data to run regression on. This data set should be financial in nature, and of a type that will work with the models we have discussed this semester (hint: we didn't look at time series) You may not use any of the data sets in the ISLR package that we have been looking at all semester. Your data set that you choose should have both qualitative and quantitative variables. (or has variables that you can transform)

Provide a description of the data below, where you obtained it, what the variable names are and what it is describing.

Description: I obtained this data from Kaggle, the time column is Number of seconds elapsed between this transaction and the first transaction in the dataset, columne v1-v10 may be result of a PCA Dimensionality reduction to protect user identities and sensitive features(v1-v10), Amount :Transaction amount, class: 1 for fraudulent transactions, 0 otherwise

Question 2:

Pick a quantitative variable and fit at least four different models in order to predict that variable using the other predictors. Determine which of the models is the best fit. You will need to provide strong reasons as to why the particular model you chose is the best one. You will need to confirm the model you have selected provides the best fit and that you have obtained the best version of that particular model (i.e. subset selection or validation for example). You need to convince the grader that you have chosen the best model.

DataSummary

```
library(gdata)
```

```
## gdata: read.xls support for 'XLS' (Excel 97-2004) files ENABLED.
```

```
##
```

```
## gdata: read.xls support for 'XLSX' (Excel 2007+) files ENABLED.
```

```
##
## Attaching package: 'gdata'
```

```
## The following object is masked from 'package:stats':
##
##     nobs
```

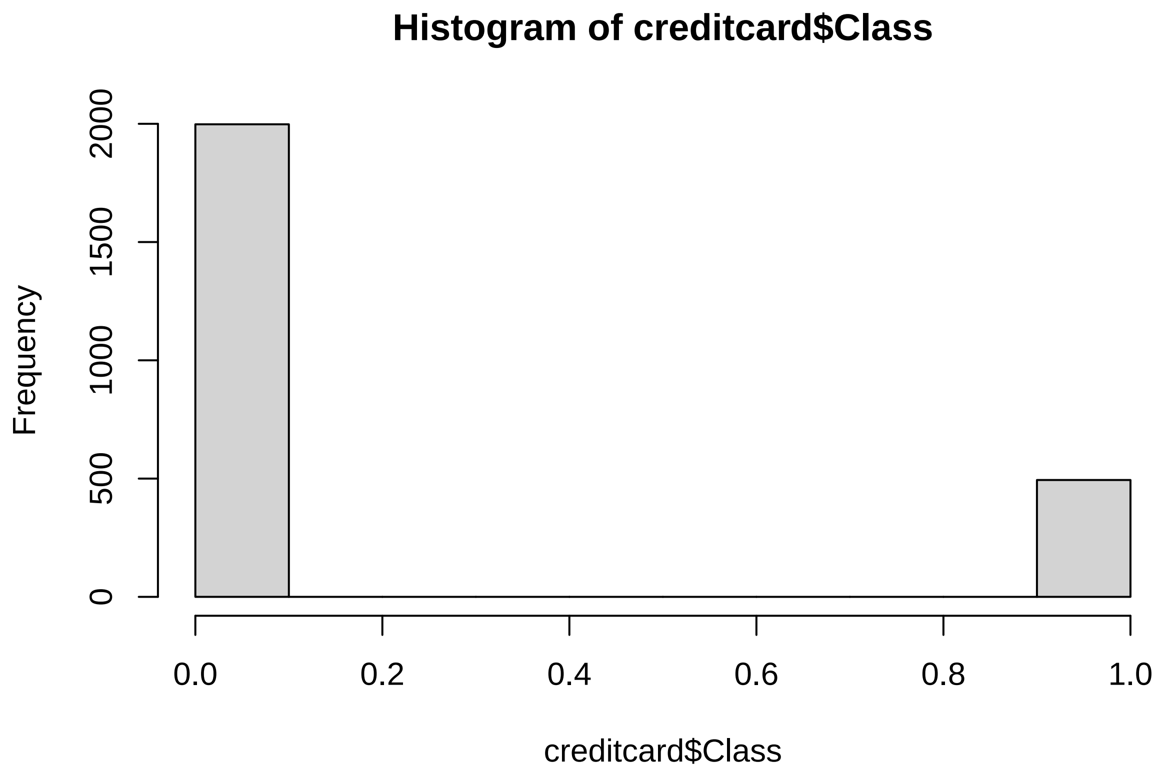
```
## The following object is masked from 'package:utils':
##
##     object.size
```

```
## The following object is masked from 'package:base':
##
##     startsWith
```

```
creditcard <- read.csv("creditcard.csv")
creditcard <- creditcard[, -c(1,2)]
creditcard <- na.omit(creditcard)
summary(creditcard$Amount)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    0.000    2.988   14.995   78.987   71.765 7712.430
```

```
hist(creditcard$Class)
```



Pickbestvariables

```
library(leaps)
Q2_data <- creditcard[, -12]
regfit_full <- regsubsets(Amount~., Q2_data)
reg_sum <- summary(regfit_full)
names(reg_sum)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

```
par(mfrow = c(2,2))
plot(reg_sum$rss, xlab = "number of Variables", tlab = "RSS", type = "l")
plot(reg_sum$adjr2, xlab = "number of Variables", tlab = "adjr2", type = "l")
which.max(reg_sum$adjr2)
```

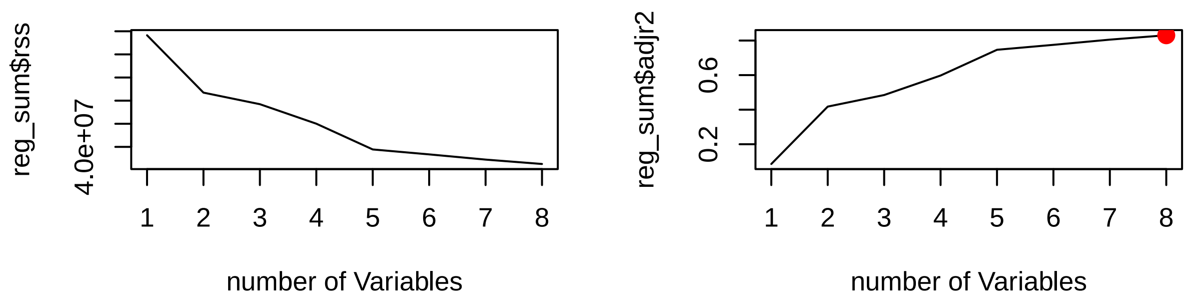
```
## [1] 8
```

```
points(8, reg_sum$adjr2[8], col = "red", cex = 2, pch = 20)
#pick v1-v8 variables
reg_sum$which
```

```
##   (Intercept)    V1    V2    V3    V4    V5    V6    V7    V8    V9   V10
## 1        TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## 2        TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
```

```
## 3        TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE
## 4        TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE
## 5        TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE
## 6        TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE
## 7        TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE
## 8        TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
```

```
Q2_data <- Q2_data[, -c(9,10)]
```



$Model1 : simplelinear Regression$

```
train <- sample(nrow(Q2_data),nrow(Q2_data)*0.7)
Q2_train_set <- Q2_data[train,]
Q2_test_set <- Q2_data[-train,]
Model1 <- lm(Amount~., data = Q2_train_set)
Model1_predicted <- predict(Model1, newdata = Q2_test_set)
mean((Model1_predicted - Q2_test_set$Amount)^2)
```

```
## [1] 7564.423
```

$Model2 Polyregression$

```
library(boot)
set.seed(10086)
cv.error_M2 <- rep(0,5)
```

```r
#It takes too long time to run so I just record it here, poly3 is the lowest cv.error

#for (i in 1:5){
#  Model_2 <- glm(Amount~poly(V1,i)+poly(V2,i)+poly(V3,i)+poly(V4,i)+poly(V5,i)+poly(V6,i)+poly(V7,i)+p
#  cv.error_M2[i] = cv.glm(Q2_train_set,Model_2)$delta[1]
#}

#cv.error_M2 : 12078.25    14623.62    10056.64  2739210.78 38486977.03
Model2 <- glm(Amount~poly(V1,3)+poly(V2,3)+poly(V3,3)+poly(V4,3)+poly(V5,3)+poly(V6,3)+poly(V7,3)+poly(V
Model2_predicted <- predict(Model2, newdata = Q2_test_set)
mean((Model2_predicted - Q2_test_set$Amount)^2)
```

```
## [1] 4517.114
```

Obviously, Model 2 has the lowest mean squared error compared to model1.

Model3Tree

```r
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

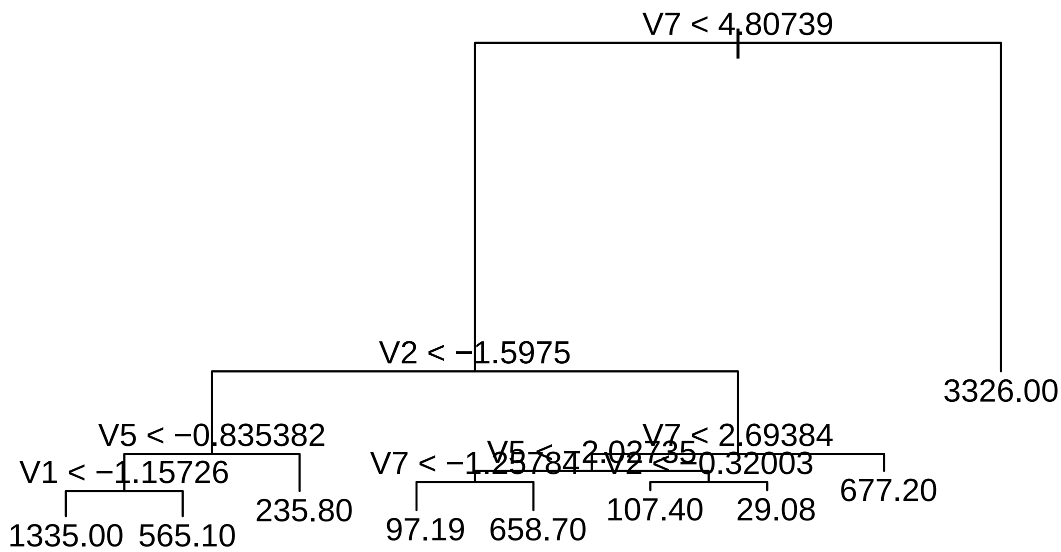
```r
library(ISLR)
summary(Q2_data$Amount)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    0.000    2.988   14.995   78.987   71.765 7712.430
```

```r
Model3 <- tree(Amount~., Q2_train_set)
summary(Model3)
```

```
##
## Regression tree:
## tree(formula = Amount ~ ., data = Q2_train_set)
## Variables actually used in tree construction:
## [1] "V7" "V2" "V5" "V1"
## Number of terminal nodes:  9
## Residual mean deviance:  24990 = 43360000 / 1735
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2497.00   -28.08   -19.91     0.00    10.10  4386.00
```

```r
plot(Model3)
text(Model3, pretty = 0)
```




```r
Model3_predict <- predict(Model3, Q2_test_set)
mean((Model3_predict - Q2_test_set$Amount)^2)
```

```
## [1] 20794.59
```

Not a good model

$Model4 : K - Means$

```r
set.seed(2)
library(e1071)
Model4 <- svm(Amount~., data=Q2_train_set, kernel="linear", cost=10, scale = FALSE)
plot(Model4, Q2_train_set)
Model4$index
```

```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14
##  [15]  15  16  17  18  19  20  21  22  23  24  25  26  27  28
##  [29]  29  30  31  32  33  34  35  36  37  38  39  40  41  42
##  [43]  43  44  45  46  47  48  49  50  51  52  53  54  55  56
##  [57]  57  58  59  60  61  62  63  64  65  66  67  68  69  70
##  [71]  71  72  73  74  75  76  77  78  79  80  81  82  83  84
##  [85]  85  86  87  88  89  90  91  92  93  94  95  96  97  98
##  [99]  99 100 101 102 103 104 105 106 107 108 109 110 111 112
## [113] 113 114 115 116 117 118 119 120 121 122 123 124 125 126
## [127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140
```

```
## [1653] 1656 1657 1658 1659 1660 1661 1662 1663 1664 1665 1666 1667 1668 1669
## [1667] 1670 1671 1672 1673 1674 1675 1676 1677 1678 1679 1680 1681 1682 1683
## [1681] 1684 1685 1686 1687 1688 1689 1690 1691 1692 1693 1694 1695 1696 1697
## [1695] 1698 1699 1700 1701 1702 1703 1704 1705 1706 1707 1708 1709 1710 1711
## [1709] 1712 1713 1714 1715 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725
## [1723] 1726 1727 1728 1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739
## [1737] 1740 1741 1742 1743 1744
```

```r
set.seed(1)
Model4_turn <- tune(svm, Amount~. ,data = Q2_train_set, kernel="linear", ranges = list(cost=c(0.001, 0.
summary(Model4_turn)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 12770.82
##
## - Detailed performance results:
##    cost    error dispersion
## 1 1e-03 65890.50  85918.345
## 2 1e-01 13621.78   7512.988
## 3 1e-01 13621.78   7512.988
## 4 1e+00 12833.85   7756.201
## 5 5e+00 12790.68   7714.632
## 6 1e+01 12791.28   7714.430
## 7 1e+02 12770.82   7698.255
```

```r
#when cost = 100 Model4 has lowest error
bestmodel <- Model4_turn$best.model
summary(bestmodel)
```

```
##
## Call:
## best.tune(method = svm, train.x = Amount ~ ., data = Q2_train_set,
##     ranges = list(cost = c(0.001, 0.1, 0.1, 1, 5, 10, 100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  100
##       gamma:  0.125
##     epsilon:  0.1
##
##
## Number of Support Vectors:  849
```

```r
Model4_predict <- predict(bestmodel, Q2_test_set)
mean((Model4_predict - Q2_test_set$Amount)^2)
```

```
## [1] 7404.733
```

$Overallthebestmodelisthesecondmodel$

#Question 3: Do the same approach as in question 2, but this time for a qualitative variable.

$Model1 - Logisticregression$

```r
Q3_data <- creditcard[, -11]
Q3_train_set <- Q3_data[train, ]
Q3_test_set <- Q3_data[-train, ]
glm.fit <- glm(Class~., data = Q3_train_set, family = binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Class ~ ., family = binomial, data = Q3_train_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5088  -0.2137  -0.1187  -0.0371   3.4319
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -4.1116     0.2909 -14.133  < 2e-16 ***
## V1           -0.2601     0.1029  -2.526   0.0115 *
## V2            0.2018     0.1191   1.694   0.0903 .
## V3           -0.9142     0.1389  -6.581 4.69e-11 ***
## V4            1.3070     0.1532   8.531  < 2e-16 ***
## V5            0.2478     0.1323   1.873   0.0611 .
## V6           -0.3918     0.1389  -2.820   0.0048 **
## V7           -0.1215     0.1361  -0.893   0.3719
## V8           -0.1531     0.1318  -1.162   0.2453
## V9            0.2390     0.2141   1.117   0.2642
## V10          -0.7987     0.1851  -4.316 1.59e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1715.04  on 1743  degrees of freedom
## Residual deviance:  333.16  on 1733  degrees of freedom
## AIC: 355.16
##
## Number of Fisher Scoring iterations: 9
```

```r
#Pick 5 most important variables
glm.fit <- glm(Class~V1+V3+V4+V6+V10, data = Q3_train_set, family = binomial)
glm.predicted <- predict(glm.fit, Q3_test_set, type = "response")
glm.predicted <- ifelse(glm.predicted>0.5, 1,0)
table(glm.predicted, Q3_test_set$Class)
```

```
##
## glm.predicted   0   1
##             0 584  12
##             1   8 144
```

```
mean(glm.predicted == Q3_test_set$Class)
```

```
## [1] 0.973262
```

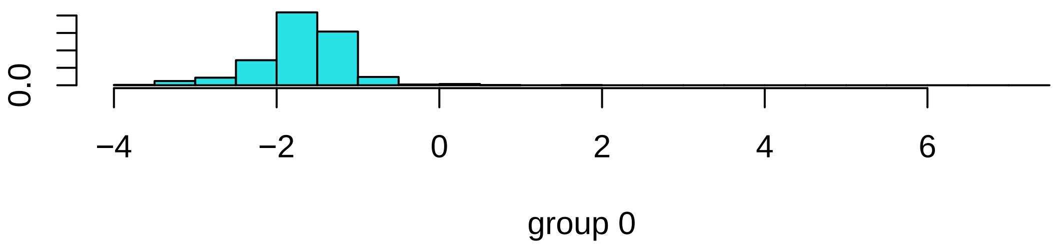
It is a very good model

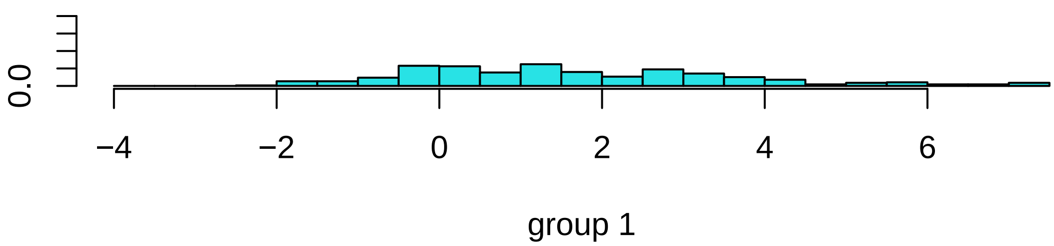
$Model2 - LDA$

```
library(MASS)
lda.fit <- lda(Class~., data = Q3_train_set)
lda.fit
```

```
## Call:
## lda(Class ~ ., data = Q3_train_set)
##
## Prior probabilities of groups:
##         0         1
## 0.8061927 0.1938073
##
## Group means:
##          V1        V2         V3        V4         V5          V6         V7
## 0 -0.271377 0.2619635  0.8392109 0.1419466 -0.0999791  0.06958173  0.1541654
## 1 -4.815448 3.5973071 -6.8582967 4.4395196 -3.0913455 -1.36294752 -5.5412402
##            V8          V9        V10
## 0 -0.05940998  0.01639195  0.0222728
## 1  0.49822148 -2.49017750 -5.5617438
##
## Coefficients of linear discriminants:
##             LD1
## V1  -0.01149477
## V2  -0.02910594
## V3  -0.28077739
## V4   0.32334106
## V5   0.20781283
## V6  -0.17367886
## V7   0.15528779
## V8  -0.07539447
## V9   0.08173098
## V10 -0.24240384
```

```
plot(lda.fit)
```






```r
lda.predict <- predict(lda.fit, Q3_test_set)
table(lda.predict$class, Q3_test_set$Class)
```

```
##
##       0   1
##   0 591  37
##   1   1 119
```

```r
mean(lda.predict$class==Q3_test_set$Class)
```

```
## [1] 0.9491979
```

$Model3 - QDA$

```r
qda.fit <- qda(Class~., data = Q3_train_set)
qda.fit
```

```
## Call:
## qda(Class ~ ., data = Q3_train_set)
##
## Prior probabilities of groups:
##         0         1
## 0.8061927 0.1938073
```

```
##
## Group means:
##          V1        V2         V3        V4         V5          V6         V7
## 0 -0.271377 0.2619635  0.8392109 0.1419466 -0.0999791  0.06958173  0.1541654
## 1 -4.815448 3.5973071 -6.8582967 4.4395196 -3.0913455 -1.36294752 -5.5412402
##            V8          V9        V10
## 0 -0.05940998  0.01639195  0.0222728
## 1  0.49822148 -2.49017750 -5.5617438
```

```
qda.class <- predict(qda.fit, Q3_test_set)$class
table(qda.class, Q3_test_set$Class)
```

```
##
## qda.class   0   1
##         0 566  15
##         1  26 141
```

```
mean(qda.class == Q3_test_set$Class)
```

```
## [1] 0.9451872
```

$Model4 - Knn$

```
library(class)
set.seed(1)
knn.pred <- knn(Q3_train_set, Q3_test_set,Q3_train_set$Class)
mean(Q3_test_set$Class==knn.pred)
```

```
## [1] 0.9893048
```

The best model is Knn

#Question 4:

In this problem, you will use support vector approaches in order to predict the direction of your ETFs in your data set from homework 2.

##(a) Create two different data frames, one for each ETF. Each data frame should include the log returns of your assets as well as a binary classifier for the direction of each ETF.

```
FA590_hw2dataset <- read.csv("~/Documents/Stevens_second_semester/FA590/Homework/Homework2/FA590_hw2data
IWS <- FA590_hw2dataset[, -12]

n <- nrow(IWS)
iws_return <- log(IWS$IWS.Adjusted[-1] / IWS$IWS.Adjusted[-n])
iws_return <- c(NA, iws_return)
IWS <- cbind(IWS, iws_return)
IWS$binary <- ifelse(IWS$iws_retur>0, 1, 0)
head(IWS)
```

```
##   AMD.Adjusted AMZN.Adjusted NKE.Adjusted VALE.Adjusted NOK.Adjusted
## 1        18.83       1539.13     72.05029      10.68235     5.621394
```

```
## 2        17.05       1500.28      70.77586      10.17480      5.454907
## 3        19.00       1575.39      72.62429      11.11853      5.807468
## 4        20.57       1629.51      73.66526      10.96785      5.895608
## 5        20.75       1656.58      74.64784      11.15025      6.022923
## 6        20.19       1659.42      74.51163      11.50712      6.081682
##   AAL.Adjusted CCL.Adjusted TGT.Adjusted DQ.Adjusted BAC.Adjusted IWS.Adjusted
## 1     31.96316     47.31749     61.98927       4.810     23.31846     71.74081
## 2     29.58167     44.96349     61.14024       4.660     22.94477     70.92140
## 3     31.53016     47.50733     61.97995       4.948     23.89768     72.96524
## 4     32.42568     48.06736     65.01222       5.312     23.87900     73.64336
## 5     31.90411     49.26335     64.94691       5.210     23.83229     74.52869
## 6     32.88819     49.34878     65.58137       5.164     24.06585     75.04672
##     iws_return binary
## 1           NA     NA
## 2 -0.011487471      0
## 3  0.028410914      1
## 4  0.009250874      1
## 5  0.011950169      1
## 6  0.006926674      1
```

```
IWS <- na.omit(IWS)

IWN <- FA590_hw2dataset[, -11]
n <- nrow(IWN)
iwn_return <- log(IWN$IWN.Adjusted[-1] / IWN$IWN.Adjusted[-n])
iwn_return <- c(NA, iwn_return)
IWN <- cbind(IWN, iwn_return)

IWN$binary <- ifelse(IWN$iwn_return>0, 1, 0)
head(IWN)
```

```
##   AMD.Adjusted AMZN.Adjusted NKE.Adjusted VALE.Adjusted NOK.Adjusted
## 1        18.83       1539.13     72.05029      10.68235      5.621394
## 2        17.05       1500.28     70.77586      10.17480      5.454907
## 3        19.00       1575.39     72.62429      11.11853      5.807468
## 4        20.57       1629.51     73.66526      10.96785      5.895608
## 5        20.75       1656.58     74.64784      11.15025      6.022923
## 6        20.19       1659.42     74.51163      11.50712      6.081682
##   AAL.Adjusted CCL.Adjusted TGT.Adjusted DQ.Adjusted BAC.Adjusted IWN.Adjusted
## 1     31.96316     47.31749     61.98927       4.810     23.31846     102.4427
## 2     29.58167     44.96349     61.14024       4.660     22.94477     101.2897
## 3     31.53016     47.50733     61.97995       4.948     23.89768     104.7202
## 4     32.42568     48.06736     65.01222       5.312     23.87900     105.8637
## 5     31.90411     49.26335     64.94691       5.210     23.83229     107.5081
## 6     32.88819     49.34878     65.58137       5.164     24.06585     108.3303
##     iwn_return binary
## 1           NA     NA
## 2 -0.011318362      0
## 3  0.033307405      1
## 4  0.010860327      1
## 5  0.015413504      1
## 6  0.007618599      1
```

```r
IWN <- na.omit(IWN)
```

##(b) Fit a support vector classifier to the data using linear kernels. You should use the tune function to determine an optimal cost for each SVM. What do you see in these results? Is one ETF more accurately predicted over the other?

```r
library(e1071)
svmfit <- svm(IWS$binary~., data = IWS[,-c(11,12)], kernel="linear", cost=10 )
summary(svmfit)
```

```
##
## Call:
## svm(formula = IWS$binary ~ ., data = IWS[, -c(11, 12)], kernel = "linear",
##     cost = 10)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  749
```

```r
plot(svmfit, IWS)
tunr.out <- tune(svm, binary~., data = IWS[,-c(11,12)], kernel="linear", ranges = list(cost=c(0.1,1,10,1
summary(tunr.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##  1000   0.5
##
## - best performance: 0.4018213
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1  1e-01   0.5 0.4025640 0.05641980
## 2  1e+00   0.5 0.4025678 0.05641248
## 3  1e+01   0.5 0.4025916 0.05639783
## 4  1e+02   0.5 0.4024471 0.05643113
## 5  1e+03   0.5 0.4018213 0.05686153
## 6  1e-01   1.0 0.4025640 0.05641980
## 7  1e+00   1.0 0.4025678 0.05641248
## 8  1e+01   1.0 0.4025916 0.05639783
## 9  1e+02   1.0 0.4024471 0.05643113
```

```
## 10 1e+03   1.0 0.4018213 0.05686153
## 11 1e-01   2.0 0.4025640 0.05641980
## 12 1e+00   2.0 0.4025678 0.05641248
## 13 1e+01   2.0 0.4025916 0.05639783
## 14 1e+02   2.0 0.4024471 0.05643113
## 15 1e+03   2.0 0.4018213 0.05686153
## 16 1e-01   3.0 0.4025640 0.05641980
## 17 1e+00   3.0 0.4025678 0.05641248
## 18 1e+01   3.0 0.4025916 0.05639783
## 19 1e+02   3.0 0.4024471 0.05643113
## 20 1e+03   3.0 0.4018213 0.05686153
## 21 1e-01   4.0 0.4025640 0.05641980
## 22 1e+00   4.0 0.4025678 0.05641248
## 23 1e+01   4.0 0.4025916 0.05639783
## 24 1e+02   4.0 0.4024471 0.05643113
## 25 1e+03   4.0 0.4018213 0.05686153
```

```r
mean((IWS$binary-predict(tunr.out$best.model, newx=IWS$binary))^2)
```

```
## [1] 0.4024656
```

```r
svmfit2 <- svm(IWN$binary~., data = IWN[,-c(11,12)], kernel="linear", cost=10 )
summary(svmfit2)
```

```
##
## Call:
## svm(formula = IWN$binary ~ ., data = IWN[, -c(11, 12)], kernel = "linear",
##     cost = 10)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  10
##       gamma:  0.1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  759
```

```r
plot(svmfit2, IWN)
tunr.out2 <- tune(svm, binary~., data = IWN[,-c(11,12)], kernel="linear", ranges = list(cost=c(0.1,1,10
summary(tunr.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##  1000   0.5
```

```
##
## - best performance: 0.4018213
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1  1e-01   0.5 0.4025640 0.05641980
## 2  1e+00   0.5 0.4025678 0.05641248
## 3  1e+01   0.5 0.4025916 0.05639783
## 4  1e+02   0.5 0.4024471 0.05643113
## 5  1e+03   0.5 0.4018213 0.05686153
## 6  1e-01   1.0 0.4025640 0.05641980
## 7  1e+00   1.0 0.4025678 0.05641248
## 8  1e+01   1.0 0.4025916 0.05639783
## 9  1e+02   1.0 0.4024471 0.05643113
## 10 1e+03   1.0 0.4018213 0.05686153
## 11 1e-01   2.0 0.4025640 0.05641980
## 12 1e+00   2.0 0.4025678 0.05641248
## 13 1e+01   2.0 0.4025916 0.05639783
## 14 1e+02   2.0 0.4024471 0.05643113
## 15 1e+03   2.0 0.4018213 0.05686153
## 16 1e-01   3.0 0.4025640 0.05641980
## 17 1e+00   3.0 0.4025678 0.05641248
## 18 1e+01   3.0 0.4025916 0.05639783
## 19 1e+02   3.0 0.4024471 0.05643113
## 20 1e+03   3.0 0.4018213 0.05686153
## 21 1e-01   4.0 0.4025640 0.05641980
## 22 1e+00   4.0 0.4025678 0.05641248
## 23 1e+01   4.0 0.4025916 0.05639783
## 24 1e+02   4.0 0.4024471 0.05643113
## 25 1e+03   4.0 0.4018213 0.05686153
```

```
mean((IWN$binary-predict(tunr.out$best.model, newx=IWN$binary))^2)
```

```
## [1] 0.4319535
```

Yes,the IWN ETF is more accurately predicted over the other

##(c) Now repeat (b), this time using SVMs with radial and polynomial basis kernels, with different values of gamma and degree and cost. Comment on your results.

```
library(e1071)
svmfit <- svm(IWS$binary~., data = IWS[,-c(11,12)], kernel="radial", cost=10 )
summary(svmfit)
```

```
##
## Call:
## svm(formula = IWS$binary ~ ., data = IWS[, -c(11, 12)], kernel = "radial",
##     cost = 10)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
```

```
##        cost:  10
##       gamma:  0.1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  719
```

```
plot(svmfit, IWS)
tunr.out <- tune(svm, binary~., data = IWS[,-c(11,12)], kernel="radial", ranges = list(cost=c(0.1,1,10,
summary(tunr.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1     4
##
## - best performance: 0.3462825
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1  1e-01   0.5 0.3827197 0.04466034
## 2  1e+00   0.5 0.3529337 0.03081888
## 3  1e+01   0.5 0.3534453 0.03730579
## 4  1e+02   0.5 0.3796875 0.04721241
## 5  1e+03   0.5 0.5117646 0.07317371
## 6  1e-01   1.0 0.3799727 0.04377706
## 7  1e+00   1.0 0.3551550 0.02920140
## 8  1e+01   1.0 0.3519129 0.04654478
## 9  1e+02   1.0 0.4181379 0.06334316
## 10 1e+03   1.0 0.6270038 0.11991541
## 11 1e-01   2.0 0.3784171 0.04239157
## 12 1e+00   2.0 0.3546528 0.03262552
## 13 1e+01   2.0 0.3567597 0.05647084
## 14 1e+02   2.0 0.4640220 0.08363519
## 15 1e+03   2.0 0.6441839 0.11051988
## 16 1e-01   3.0 0.3774620 0.04160475
## 17 1e+00   3.0 0.3499791 0.03551792
## 18 1e+01   3.0 0.3681139 0.06696216
## 19 1e+02   3.0 0.4739610 0.07574507
## 20 1e+03   3.0 0.5776554 0.08995785
## 21 1e-01   4.0 0.3767414 0.04115726
## 22 1e+00   4.0 0.3462825 0.03855581
## 23 1e+01   4.0 0.3702410 0.06573958
## 24 1e+02   4.0 0.4719224 0.07031363
## 25 1e+03   4.0 0.5168931 0.07427616
```

```
mean((IWS$binary-predict(tunr.out$best.model, newx=IWS$binary))^2)
```

```
## [1] 0.1794741
```

```r
svmfit2 <- svm(IWN$binary~., data = IWN[,-c(11,12)], kernel="radial", cost=10 )
summary(svmfit2)
```

```
##
## Call:
## svm(formula = IWN$binary ~ ., data = IWN[, -c(11, 12)], kernel = "radial",
##     cost = 10)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.1
##     epsilon:  0.1
##
##
## Number of Support Vectors:  754
```

```r
plot(svmfit2, IWN)
tunr.out2 <- tune(svm, binary~., data = IWN[,-c(11,12)], kernel="radial", ranges = list(cost=c(0.1,1,10
summary(tunr.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1     4
##
## - best performance: 0.3462825
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1  1e-01   0.5 0.3827197 0.04466034
## 2  1e+00   0.5 0.3529337 0.03081888
## 3  1e+01   0.5 0.3534453 0.03730579
## 4  1e+02   0.5 0.3796875 0.04721241
## 5  1e+03   0.5 0.5117646 0.07317371
## 6  1e-01   1.0 0.3799727 0.04377706
## 7  1e+00   1.0 0.3551550 0.02920140
## 8  1e+01   1.0 0.3519129 0.04654478
## 9  1e+02   1.0 0.4181379 0.06334316
## 10 1e+03   1.0 0.6270038 0.11991541
## 11 1e-01   2.0 0.3784171 0.04239157
## 12 1e+00   2.0 0.3546528 0.03262552
## 13 1e+01   2.0 0.3567597 0.05647084
## 14 1e+02   2.0 0.4640220 0.08363519
## 15 1e+03   2.0 0.6441839 0.11051988
## 16 1e-01   3.0 0.3774620 0.04160475
```

```
## 17 1e+00   3.0 0.3499791 0.03551792
## 18 1e+01   3.0 0.3681139 0.06696216
## 19 1e+02   3.0 0.4739610 0.07574507
## 20 1e+03   3.0 0.5776554 0.08995785
## 21 1e-01   4.0 0.3767414 0.04115726
## 22 1e+00   4.0 0.3462825 0.03855581
## 23 1e+01   4.0 0.3702410 0.06573958
## 24 1e+02   4.0 0.4719224 0.07031363
## 25 1e+03   4.0 0.5168931 0.07427616
```

```
mean((IWN$binary-predict(tunr.out$best.model, newx=IWN$binary))^2)
```

```
## [1] 0.2305031
```