QF301. Homework #5.

2022-11-27

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: Arjun Koshal

CWID: 10459064

Date: 11/30/2022

Instructions

In this assignment, you should use R markdown to answer the questions below. Simply type your R code into embedded chunks as shown above. When you have completed the assignment, knit the document into a PDF file, and upload both the .pdf and .Rmd files to Canvas. If you use Python, you will need to include your .ipynb and prinout as .pdf as well.

```r
CWID = 10459064 #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)#You can reset the seed at any time in your code,
#but please always set it to this seed.
```

Question 1 (100pt)

In this assignment, you will be required to find a set of data to run a regression or classification on.

Question 1.1 (10pt)

For this task, use the quantmod package to obtain the daily adjusted close prices of at least 3 different stocks. You should have at least 5 years of data for all assets. You should inspect the dates to make sure you are including everything appropriately. Find the daily log returns of all stocks along with (at least) 3 lags for each stock. Create a data frame of your desired output (whether as a regression of returns or classification) and the lagged returns. Print the first 6 lines of your data frame.

Solution:

```r
library(quantmod)

getSymbols(c("AMZN", "MSFT", "TSLA"),from="2017-11-15",to="2022-11-16")
```

```
## [1] "AMZN" "MSFT" "TSLA"
```

```
rAMZN = as.numeric(dailyReturn(AMZN$AMZN.Adjusted,type="log"))
rMSFT = as.numeric(dailyReturn(MSFT$MSFT.Adjusted,type="log"))
rTSLA = as.numeric(dailyReturn(TSLA$TSLA.Adjusted,type="log"))

rAMZN1 = as.numeric(lag(rAMZN,k=1))[-(1:3)]
rAMZN2 = as.numeric(lag(rAMZN,k=2))[-(1:3)]
rAMZN3 = as.numeric(lag(rAMZN,k=3))[-(1:3)]
rAMZN = as.numeric(rAMZN)[-(1:3)]
rAMZN.dir = (rAMZN > 0) + 0

rMSFT1 = as.numeric(lag(rMSFT,k=1))[-(1:3)]
rMSFT2 = as.numeric(lag(rMSFT,k=2))[-(1:3)]
rMSFT3 = as.numeric(lag(rMSFT,k=3))[-(1:3)]
rMSFT = as.numeric(rMSFT)[-(1:3)]
rMSFT.dir = (rMSFT > 0) + 0

rTSLA1 = as.numeric(lag(rTSLA,k=1))[-(1:3)]
rTSLA2 = as.numeric(lag(rTSLA,k=2))[-(1:3)]
rTSLA3 = as.numeric(lag(rTSLA,k=3))[-(1:3)]
rTSLA = as.numeric(rTSLA)[-(1:3)]
rTSLA.dir = (rTSLA > 0) + 0

df = data.frame(rAMZN,rAMZN.dir,rMSFT,rMSFT.dir,rTSLA,rTSLA.dir,
                rAMZN1,rMSFT1,rTSLA1,rAMZN2,rMSFT2,
                rTSLA2,rAMZN3,rMSFT3,rTSLA3)
head(df)
```

```
##          rAMZN rAMZN.dir        rMSFT rMSFT.dir        rTSLA rTSLA.dir
## 1 -0.003164648         0  0.001576608         1 -0.020231827         0
## 2  0.011634009         1  0.014316061         1  0.028954191         1
## 3  0.014523375         1 -0.007312979         0 -0.016529239         0
## 4  0.025482132         1  0.001803131         1  0.009392746         1
## 5  0.008254221         1  0.007299751         1  0.003985077         1
## 6 -0.001866554         0  0.011970450         1  0.002333045         1
##         rAMZN1       rMSFT1       rTSLA1       rAMZN2       rMSFT2       rTSLA2
## 1 -0.003164648  0.001576608 -0.020231827 -0.003164648  0.001576608 -0.020231827
## 2  0.011634009  0.014316061  0.028954191  0.011634009  0.014316061  0.028954191
## 3  0.014523375 -0.007312979 -0.016529239  0.014523375 -0.007312979 -0.016529239
## 4  0.025482132  0.001803131  0.009392746  0.025482132  0.001803131  0.009392746
## 5  0.008254221  0.007299751  0.003985077  0.008254221  0.007299751  0.003985077
## 6 -0.001866554  0.011970450  0.002333045 -0.001866554  0.011970450  0.002333045
##         rAMZN3       rMSFT3       rTSLA3
## 1 -0.003164648  0.001576608 -0.020231827
## 2  0.011634009  0.014316061  0.028954191
## 3  0.014523375 -0.007312979 -0.016529239
## 4  0.025482132  0.001803131  0.009392746
## 5  0.008254221  0.007299751  0.003985077
## 6 -0.001866554  0.011970450  0.002333045
```

Question 1.2 (10pt)

Provide a description of the data below: what is your desired prediction and why do you think your data will aid in this task?

Solution:

```
The data consists of 5 year stock prices of Amazon, Microsoft, and Tesla. The goal is to see from 4 dif
which is the best model that fits the data. The data will aid in the task because we have sufficient amo
from 5 years and we can solve the classification problem.
```

Question 1.3 (60pt)

Fit at least four different models in order to run your prediction. You will need to confirm the models you try are as good a fit as you can find for that model type (i.e., feature selection or cross-validation to find model hyperparameters). You need to convince the grader that you have chosen the best model fits, so provide comments as to why you choose the models you use.

If you use neural networks, make reference in your solution below and provide the Python code with your submission.

Solution:

Model 1:

```
# Logistic Regression
train = sample(length(rAMZN),0.8*length(rAMZN),replace=FALSE)

logistic.reg = glm(rAMZN.dir ~ rAMZN1 + rAMZN2 + rMSFT1 + rMSFT2 + rTSLA1 +
                     rTSLA2, data=df, subset=train, family=binomial)
summary(logistic.reg)
```

```
##
## Call:
## glm(formula = rAMZN.dir ~ rAMZN1 + rAMZN2 + rMSFT1 + rMSFT2 +
##     rTSLA1 + rTSLA2, family = binomial, data = df, subset = train)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.006357   0.000000   0.000000   0.000000   0.004535
##
## Coefficients: (3 not defined because of singularities)
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.502e+00  1.360e+02  -0.055    0.956
## rAMZN1       2.742e+05  2.433e+06   0.113    0.910
## rAMZN2              NA         NA      NA       NA
## rMSFT1       1.682e+01  2.645e+04   0.001    0.999
## rMSFT2              NA         NA      NA       NA
## rTSLA1      -5.425e+02  7.687e+03  -0.071    0.944
## rTSLA2              NA         NA      NA       NA
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.3870e+03  on 1003  degrees of freedom
## Residual deviance: 7.3444e-05  on 1000  degrees of freedom
## AIC: 8.0001
##
## Number of Fisher Scoring iterations: 25
```

```
logistic.prob=predict(logistic.reg,df[-train,])

logistic.pred=rep(0,length(logistic.prob))
logistic.pred[logistic.prob>.5] = 1

table(predict=logistic.pred , truth=df$rAMZN.dir[-train])
```

```
##        truth
## predict   0   1
##       0 118   0
##       1   0 134
```

```
mean(logistic.pred==df$rAMZN.dir[-train])
```

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

Model 2:

```
# Naive Bayes Classifier
library("e1071")
df.x = data.frame(rAMZN1,rMSFT1,rTSLA1,rAMZN2,rMSFT2,rTSLA2)

nb = naiveBayes(df.x[train,] , df$rAMZN.dir[train])

nb.prob = predict(nb , newdata=df.x[-train,] , type="raw")
head(nb.prob)
```

```
##               0          1
## [1,] 0.66272797 0.33727203
## [2,] 0.92999947 0.07000053
## [3,] 0.05216017 0.94783983
## [4,] 0.01969091 0.98030909
## [5,] 0.17156329 0.82843671
## [6,] 0.14397933 0.85602067
```

```
nb.pred = (nb.prob[,1] < nb.prob[,2])+0

mean(nb.pred==df$rAMZN.dir[-train])
```

```
## [1] 0.8928571
```

```r
table(predict=nb.pred , truth=df$rAMZN.dir[-train])
```

```
##        truth
## predict   0   1
##       0  94   3
##       1  24 131
```

```r
nb.prob.train = predict(nb , newdata=df.x[train,] , type="raw")
nb.pred.train = (nb.prob.train[,1] < nb.prob.train[,2]) + 0
mean(nb.pred.train==df$rAMZN.dir[train])
```

```
## [1] 0.9003984
```

Model 3:

```r
# Random Forest classifier with 300 trees and 2 predictors in each tree
library(randomForest)

rAMZN.factor = as.factor(rAMZN.dir)
df.tree = data.frame(rAMZN.dir=rAMZN.factor , rAMZN1,rMSFT1,rTSLA1,rAMZN2,
                     rMSFT2,rTSLA2)

rf.class = randomForest(rAMZN.dir ~ .,data=df.tree,subset=train,mtry=2,
                        ntree=300,importance=TRUE)
rf.class
```

```
##
## Call:
##  randomForest(formula = rAMZN.dir ~ ., data = df.tree, mtry = 2,      ntree = 300, importance = TRUE
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0%
## Confusion matrix:
##     0   1 class.error
## 0 467   0           0
## 1   0 537           0
```

```r
rf.pred = predict(rf.class,newdata=df.tree[-train,])

mean(rf.pred==df.tree$rAMZN.dir[-train])
```

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

```r
table(predict=rf.pred,truth=df.tree$rAMZN.dir[-train])
```

```
##        truth
## predict   0   1
##       0 118   0
##       1   0 134
```

```r
rf.pred.train = predict(rf.class,newdata=df.tree[train,])
mean(rf.pred.train==df$rAMZN.dir[train])
```

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

Model 4:

```r
# Support vector machine using a radial basis kernel
rAMZN.factor = as.factor(rAMZN.dir)
df.svm = data.frame(rAMZN.dir=rAMZN.factor , rAMZN1,rMSFT1,rTSLA1,rAMZN2,
                    rMSFT2,rTSLA2)

tune.out = tune(svm,rAMZN.dir ~ .,data=df.svm[train,],kernel="radial",
                ranges=list(cost=c(0.001,.01,.1,1,5,10,100),
                            gamma=c(.5,1,2,3,4)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.5
##
## - best performance: 0.01693069
##
## - Detailed performance results:
##     cost gamma      error  dispersion
## 1  1e-03   0.5 0.46505941 0.050224918
## 2  1e-02   0.5 0.11551485 0.035218714
## 3  1e-01   0.5 0.03387129 0.018958289
## 4  1e+00   0.5 0.02490099 0.011757909
## 5  5e+00   0.5 0.02193069 0.006366858
## 6  1e+01   0.5 0.01693069 0.010561778
## 7  1e+02   0.5 0.01890099 0.017754147
## 8  1e-03   1.0 0.46505941 0.050224918
## 9  1e-02   1.0 0.29072277 0.057222661
## 10 1e-01   1.0 0.03980198 0.020367181
## 11 1e+00   1.0 0.02191089 0.006304762
## 12 5e+00   1.0 0.02495050 0.014364286
## 13 1e+01   1.0 0.02394059 0.016428538
## 14 1e+02   1.0 0.02393069 0.017100491
## 15 1e-03   2.0 0.46505941 0.050224918
## 16 1e-02   2.0 0.46505941 0.050224918
## 17 1e-01   2.0 0.04579208 0.019403597
## 18 1e+00   2.0 0.02893069 0.016681776
## 19 5e+00   2.0 0.03192079 0.016915355
## 20 1e+01   2.0 0.03091089 0.015260363
## 21 1e+02   2.0 0.03490099 0.015846826
```

```
## 22 1e-03   3.0 0.46505941 0.050224918
## 23 1e-02   3.0 0.46505941 0.050224918
## 24 1e-01   3.0 0.05183168 0.023043690
## 25 1e+00   3.0 0.03291089 0.017724290
## 26 5e+00   3.0 0.03590099 0.017826194
## 27 1e+01   3.0 0.03589109 0.016510179
## 28 1e+02   3.0 0.03790099 0.016918227
## 29 1e-03   4.0 0.46505941 0.050224918
## 30 1e-02   4.0 0.46505941 0.050224918
## 31 1e-01   4.0 0.08165347 0.028773581
## 32 1e+00   4.0 0.03983168 0.018746691
## 33 5e+00   4.0 0.03686139 0.014958693
## 34 1e+01   4.0 0.03985149 0.017017242
## 35 1e+02   4.0 0.04384158 0.018965719
```

```r
best.svm.class = tune.out$best.model
best.svm.class$cost
```

```
## [1] 10
```

```r
best.svm.class$gamma
```

```
## [1] 0.5
```

```r
summary(best.svm.class)
```

```
##
## Call:
## best.tune(method = svm, train.x = rAMZN.dir ~ ., data = df.svm[train,
##     ], ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100), gamma = c(0.5,
##     1, 2, 3, 4)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  147
##
##  ( 73 74 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
best.svm.pred=predict(best.svm.class,newdata = df.svm[-train,])

mean((best.svm.pred==df.svm$rAMZN.dir[-train]))
```

```
## [1] 0.9761905
```

```
table(predict=best.svm.pred,truth=df.svm$rAMZN.dir[-train])
```

```
##        truth
## predict   0   1
##       0 117   5
##       1   1 129
```

```
best.svm.pred.train = predict(best.svm.class,newdata = df.svm[train,])
mean(best.svm.pred.train==df$rAMZN.dir[train])
```

```
## [1] 0.9950199
```

Question 1.4 (20pt)

Determine which of your four (or more) 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 need to convince the grader that you have chosen the best model.

Solution:

Although Random forest appears to perform the best, it had a training accuracy of 100%. This implies that the training data was over fitting and could lead to poor performance on new data. I would recommend SVM due to the slightly lower training accuracy, and SVM provides more balanced predictions.