Stevens Institute of Technology

FA590. Assignment #4.

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

#"1/2/17"-4/25/22

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

2022-05-04

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:Darsh Kachhara

CWID:10474181

Date:5/4/22

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 = 10474181 #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)
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.1.2
```

```
library(leaps)
library(splines)
library(gam)
```

```
## Warning: package 'gam' was built under R version 4.1.2
```

```
## Loading required package: foreach

## Warning: package 'foreach' was built under R version 4.1.2

## Loaded gam 1.20.1
```

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 4.1.2
```

```
library(tree)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.1.2

## randomForest 4.7-1

## Type rfNews() to see new features/changes/bug fixes.
```

```
library(boot)
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
library(class)
```

```
## Warning: package 'class' was built under R version 4.1.2
```

```
library(e1071)
library(glmnet)
```

```
## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 4.1.2

## Loaded glmnet 4.1-3
```

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.

The project will determine the relationship between the Indian StockMarket Index NIFty 50 and other sectors of the same stock market by their Indexes.

The data I have used for this project includes NIFTY50 (The largest and most recognized Index of National Stock Exchange India.) It is made of a cumulative of 50 stocks(BLUE CHIP) ranging from all sectors and is considered a proxy for the entire Indian stock market.The other 10 Indexes represent their respectve sectors.The data has been downloaded from Indian Data website "Trendlyne."

The Varaibles are Explanatory varaibles : NIFTYAUTO : Represents Auto sector NIFTYBANK :Represents Banking sector NIFTYENERGY : Represents Energy sector NIFTYFMCG: Represents FMCG sector NIFTYIT :Represents IT sector NIFTYMEDIA : Represents Media sector NIFTYMETAL :Represents Metal sector NIFTYPHARMA :Represents Pharma sector NIFTYREALTY : Represents Reality sector NIFTY-INFRA : Represents Infrastructure sector Explained NIFTY50: CNX composite Nifty 50 index

```
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.1.2
```

```
MAIN_DATA <- read_excel("DARSHp4data/MAIN_DATA.xlsx")
all_data <- MAIN_DATA
all_data <- log(all_data[c(2:nrow(all_data)),]/all_data[c(1:nrow(all_data)-1),])
all_data <- cbind(all_data[c(1:dim(all_data)[1]-1),
c(1:dim(all_data)[2]-1)], all_data[c(2:dim(all_data)[1]),
dim(all_data)[2]])
colnames(all_data)[ncol(all_data)] <- c("NIFTY50")
dim(all_data)
```

```
## [1] 1310   11
```

```
summary(all_data)
```

```
##    NIFTYAUTO            NIFTYBANK           NIFTYENERGY
##  Min.   :-0.1490552   Min.   :-0.1831300   Min.   :-0.1021668
##  1st Qu.:-0.0068461   1st Qu.:-0.0064952   1st Qu.:-0.0063313
##  Median : 0.0005452   Median : 0.0009507   Median : 0.0011685
##  Mean   : 0.0001369   Mean   : 0.0005339   Mean   : 0.0007871
##  3rd Qu.: 0.0072968   3rd Qu.: 0.0079165   3rd Qu.: 0.0085394
##  Max.   : 0.0989966   Max.   : 0.0999515   Max.   : 0.0828181
##    NIFTYFMCG              NIFTYIT             NIFTYMEDIA
##  Min.   :-0.1119978   Min.   :-0.1006498   Min.   :-0.1788166
##  1st Qu.:-0.0049448   1st Qu.:-0.0056610   1st Qu.:-0.0090792
##  Median : 0.0005183   Median : 0.0009693   Median : 0.0003053
##  Mean   : 0.0004606   Mean   : 0.0008566   Mean   :-0.0001044
##  3rd Qu.: 0.0059086   3rd Qu.: 0.0078592   3rd Qu.: 0.0094381
##  Max.   : 0.0799060   Max.   : 0.0864042   Max.   : 0.1345096
##    NIFTYMETAL           NIFTYPHARMA           NIFTYREALTY
##  Min.   :-0.1233179   Min.   :-9.351e-02   Min.   :-0.120524
##  1st Qu.:-0.0092737   1st Qu.:-7.515e-03   1st Qu.:-0.009542
##  Median : 0.0009857   Median :-9.649e-05   Median : 0.001774
##  Mean   : 0.0006498   Mean   : 2.026e-04   Mean   : 0.000720
##  3rd Qu.: 0.0119678   3rd Qu.: 7.939e-03   3rd Qu.: 0.011448
##  Max.   : 0.0759916   Max.   : 9.865e-02   Max.   : 0.083025
##    NIFTYINFRA             NIFTY50
```

```
##  Min.   :-0.1283560   Min.   :-0.1390375
##  1st Qu.:-0.0058349   1st Qu.:-0.0044786
##  Median : 0.0010635   Median : 0.0009872
##  Mean   : 0.0004893   Mean   : 0.0005650
##  3rd Qu.: 0.0074436   3rd Qu.: 0.0065395
##  Max.   : 0.0692837   Max.   : 0.0840029
```

```
print(head(all_data))
```

```
##      NIFTYAUTO    NIFTYBANK  NIFTYENERGY    NIFTYFMCG       NIFTYIT
## 1 -0.002856040  0.003666141  0.011900597  0.007398800 -0.0004631126
## 2  0.003284816 -0.008049789 -0.008104810  0.003401966  0.0129105808
## 3  0.020304943  0.012494973  0.015212045  0.006439617 -0.0092693616
## 4 -0.001644863  0.008139144 -0.003518646 -0.009700202 -0.0282460168
## 5 -0.001126162  0.001239376 -0.006939979  0.005741354  0.0086711509
## 6  0.012625380  0.029280057  0.008943978  0.008091158  0.0022575971
##     NIFTYMEDIA    NIFTYMETAL  NIFTYPHARMA  NIFTYREALTY   NIFTYINFRA
## 1  0.017678136  0.0035792075  0.000702940  0.021960101  0.003246697
## 2  0.001668339 -0.0012715497 -0.001323877  0.011220314  0.007709579
## 3  0.016568910  0.0314922345  0.014830571 -0.008403411  0.014406291
## 4 -0.011319652 -0.0001429567  0.001428541  0.008403411 -0.003819046
## 5  0.005295983 -0.0014485888 -0.014353865 -0.002793298 -0.005342525
## 6  0.003496669  0.0142511785  0.004180739  0.004187026  0.006986146
##         NIFTY50
## 1 -0.0002136393
## 2  0.0101189496
## 3 -0.0036324930
## 4 -0.0009405426
## 5  0.0063602167
## 6  0.0110444004
```

```
print(tail(all_data))
```

```
##         NIFTYAUTO    NIFTYBANK   NIFTYENERGY    NIFTYFMCG      NIFTYIT
## 1305  0.021763031  0.001042604  0.0006207789 -0.028637310 -0.030221742
## 1306  0.022048137  0.008896808  0.0087465365  0.010269181  0.011540114
## 1307 -0.005642300 -0.010380628  0.0116189516  0.007476725  0.013529583
## 1308 -0.010175689  0.010858472 -0.0051240405 -0.006081990 -0.005685896
## 1309  0.027578453 -0.009213922 -0.0253394291 -0.016227388 -0.021278426
## 1310 -0.005620954  0.002092679  0.0262030460  0.018640065  0.003021200
##        NIFTYMEDIA   NIFTYMETAL NIFTYPHARMA  NIFTYREALTY   NIFTYINFRA
## 1305 -0.019401434 -0.010728200 -0.01430654  0.005467238 -0.001478472
## 1306 -0.004578057 -0.003334834  0.01016817  0.011175773  0.012938076
## 1307 -0.001196784  0.002148359  0.01169144 -0.013961446  0.013173826
## 1308 -0.002550660 -0.019914807 -0.01836046 -0.038328453 -0.007819952
## 1309 -0.021891597 -0.028960594 -0.02033078  0.035087322 -0.020764212
## 1310  0.018830950  0.012712927  0.01102070 -0.003027078  0.020427425
##           NIFTY50
## 1305 -0.003122329
## 1306 -0.017432251
## 1307 -0.012598204
## 1308  0.010435581
## 1309  0.014831219
## 1310 -0.012767590
```

```r
cor(all_data)
```

```
##                NIFTYAUTO    NIFTYBANK NIFTYENERGY    NIFTYFMCG      NIFTYIT
## NIFTYAUTO    1.000000000  0.013053191 -0.01720921 -0.002467853 -0.032422885
## NIFTYBANK    0.013053191  1.000000000  0.04156784 -0.003863362  0.015858859
## NIFTYENERGY -0.017209214  0.041567840  1.00000000  0.536650985  0.410538725
## NIFTYFMCG   -0.002467853 -0.003863362  0.53665099  1.000000000  0.425499774
## NIFTYIT     -0.032422885  0.015858859  0.41053873  0.425499774  1.000000000
## NIFTYMEDIA   0.041903274  0.066493971  0.50340690  0.408584643  0.340621481
## NIFTYMETAL  -0.006461267  0.043728200  0.64713273  0.480481059  0.400836864
## NIFTYPHARMA  0.039781834  0.042777035  0.44859986  0.460422522  0.384765477
## NIFTYREALTY  0.636385598  0.022411924  0.03523078  0.022585676 -0.009832449
## NIFTYINFRA   0.007394314  0.046458027  0.79619138  0.631232904  0.477482286
## NIFTY50      0.029449131  0.103901643 -0.02734410 -0.065045757 -0.083790018
##             NIFTYMEDIA   NIFTYMETAL NIFTYPHARMA  NIFTYREALTY   NIFTYINFRA
## NIFTYAUTO   0.04190327 -0.006461267  0.03978183  0.636385598  0.007394314
## NIFTYBANK   0.06649397  0.043728200  0.04277704  0.022411924  0.046458027
## NIFTYENERGY 0.50340690  0.647132735  0.44859986  0.035230783  0.796191379
## NIFTYFMCG   0.40858464  0.480481059  0.46042252  0.022585676  0.631232904
## NIFTYIT     0.34062148  0.400836864  0.38476548 -0.009832449  0.477482286
## NIFTYMEDIA  1.00000000  0.528148707  0.39365972  0.102090101  0.586123004
## NIFTYMETAL  0.52814871  1.000000000  0.48195467  0.045386677  0.729273927
## NIFTYPHARMA 0.39365972  0.481954672  1.00000000  0.047330239  0.530858332
## NIFTYREALTY 0.10209010  0.045386677  0.04733024  1.000000000  0.053602287
## NIFTYINFRA  0.58612300  0.729273927  0.53085833  0.053602287  1.000000000
## NIFTY50     0.01028938 -0.011741631 -0.03287721  0.074950590 -0.025338539
##                 NIFTY50
## NIFTYAUTO    0.02944913
## NIFTYBANK    0.10390164
## NIFTYENERGY -0.02734410
## NIFTYFMCG   -0.06504576
## NIFTYIT     -0.08379002
## NIFTYMEDIA   0.01028938
## NIFTYMETAL  -0.01174163
## NIFTYPHARMA -0.03287721
## NIFTYREALTY  0.07495059
## NIFTYINFRA  -0.02533854
## NIFTY50      1.00000000
```

Categorizing Nifty as Bullish for positive return on Nifty and Bearish for negative return.

```r
clas_data <- all_data
NIFTY50_der <- rep(1,nrow(clas_data))
NIFTY50_der[which(clas_data$NIFTY50>=0)] = "Bullish"
NIFTY50_der[which(clas_data$NIFTY50<0)] ="Bearish"
NIFTY50_der <- as.factor(NIFTY50_der)
clas_data$NIFTY50 <- NIFTY50_der
clas_train <- sample(nrow(clas_data),floor(nrow(clas_data)/2))
clas_trainset <- clas_data[clas_train,]
clas_testset <- clas_data[-clas_train,]
```

```
set.seed(1)
train <- sample(nrow(all_data),floor(nrow(all_data)/2))
trainset <- all_data[train,]
testset <- all_data[-train,]
```

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.

To conduct regression on the quantitative variable, we use 4 method to do it. 1. Simple Linear Regression

2. Multiple Linear Regression with subset selection and lasso/ridge modification.

3,Support Vector Regression

4,Random Forest Regression:

##Simple Regression

Being in the Indian market, I know first hand that NIFTYBANK and NIFTY 50 are often traded as a pair, so lets fit a simple model and see if it works out

```
model1 <- lm(NIFTY50 ~NIFTYBANK, data = trainset)
summary(model1)
```

```
##
## Call:
## lm(formula = NIFTY50 ~ NIFTYBANK, data = trainset)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.130878 -0.005014  0.000556  0.005999  0.070823
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.551e-05  4.712e-04  -0.033    0.974
## NIFTYBANK    1.329e-01  2.719e-02   4.890 1.27e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01206 on 653 degrees of freedom
## Multiple R-squared:  0.03532,    Adjusted R-squared:  0.03384
## F-statistic: 23.91 on 1 and 653 DF,  p-value: 1.273e-06
```

The explanatory variable NIFTYBANK is significant and the Adj. R-Squared is 0.03384 respectively.Now, let us test this model using the test dataset.

```r
p1 <- predict(model1, newdata=testset)
e1 <- mean((testset$NIFTY50 - p1)^2)
e1
```

```
## [1] 0.0001293893
```

So, the Mean Square Error of the model1 is 0.0001293893, which seems very less.So, the model might be a good model.

Multiple Linear Regression

Now we perform the multiple linear regression. First of all, we do the subset selection to determine which variable is more likely to be included in the model.

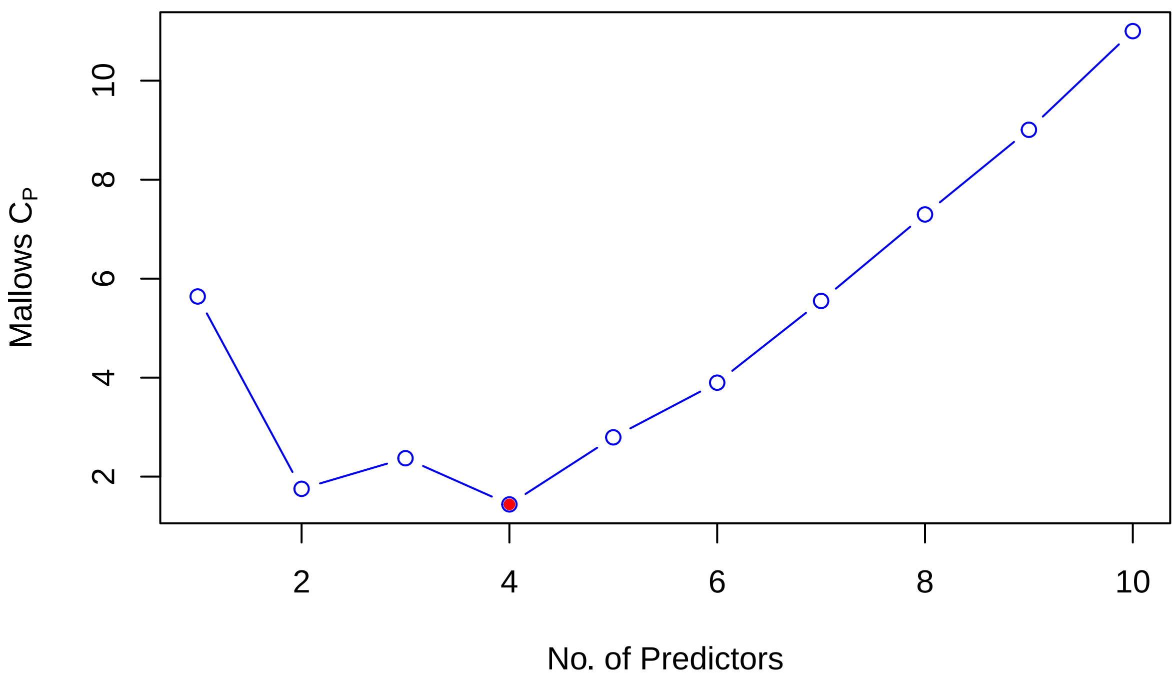
```r
subsets=regsubsets(NIFTY50~.,data=trainset,method="exhaustive",nvmax=30)
summary(subsets)
```

```
## Subset selection object
## Call: regsubsets.formula(NIFTY50 ~ ., data = trainset, method = "exhaustive",
##     nvmax = 30)
## 10 Variables  (and intercept)
##             Forced in Forced out
## NIFTYAUTO       FALSE      FALSE
## NIFTYBANK       FALSE      FALSE
## NIFTYENERGY     FALSE      FALSE
## NIFTYFMCG       FALSE      FALSE
## NIFTYIT         FALSE      FALSE
## NIFTYMEDIA      FALSE      FALSE
## NIFTYMETAL      FALSE      FALSE
## NIFTYPHARMA     FALSE      FALSE
## NIFTYREALTY     FALSE      FALSE
## NIFTYINFRA      FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           NIFTYAUTO NIFTYBANK NIFTYENERGY NIFTYFMCG NIFTYIT NIFTYMEDIA
## 1  ( 1 )  " "       "*"       " "         " "       " "     " "
## 2  ( 1 )  " "       "*"       " "         " "       "*"     " "
## 3  ( 1 )  "*"       "*"       " "         " "       "*"     " "
## 4  ( 1 )  "*"       "*"       " "         " "       "*"     " "
## 5  ( 1 )  "*"       "*"       " "         "*"       "*"     " "
## 6  ( 1 )  "*"       "*"       " "         "*"       "*"     " "
## 7  ( 1 )  "*"       "*"       "*"         "*"       "*"     " "
## 8  ( 1 )  "*"       "*"       "*"         "*"       "*"     " "
## 9  ( 1 )  "*"       "*"       "*"         "*"       "*"     " "
## 10  ( 1 ) "*"       "*"       "*"         "*"       "*"     "*"
##           NIFTYMETAL NIFTYPHARMA NIFTYREALTY NIFTYINFRA
## 1  ( 1 )  " "        " "         " "         " "
## 2  ( 1 )  " "        " "         " "         " "
## 3  ( 1 )  " "        " "         " "         " "
## 4  ( 1 )  " "        " "         "*"         " "
## 5  ( 1 )  " "        " "         "*"         " "
## 6  ( 1 )  "*"        " "         "*"         " "
```

```
## 7  ( 1 )  "*"        " "         "*"         " "
## 8  ( 1 )  "*"        "*"         "*"         " "
## 9  ( 1 )  "*"        "*"         "*"         "*"
## 10  ( 1 ) "*"        "*"         "*"         "*"
```

In order to determin the number of variables, we plot several indicators against the number of variables to find the best one.Finding the best model with the help of Mallow Cp.

```
c <- summary(subsets)$cp
plot(c ,type='b',xlab="No. of Predictors",ylab=expression("Mallows C"[P]),
col="blue")
points(which.min(c),c[which.min(c)],pch=20,
col="red")
```



The plots suggest here we select 4 variables. The 4 varaibles contain first 3 indices NIFTYBANK NIFTYIT NIFTYFMCG NIFTYREALTY. After regression, we give the summary of the model and the test MSE.

```
fit1 = lm(NIFTY50~NIFTYIT+NIFTYBANK+NIFTYFMCG+NIFTYREALTY,data = trainset)
summary((fit1))
```

```
##
## Call:
## lm(formula = NIFTY50 ~ NIFTYIT + NIFTYBANK + NIFTYFMCG + NIFTYREALTY,
##     data = trainset)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.126316 -0.005074  0.000643  0.006046  0.074170
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.386e-05  4.715e-04   0.114   0.9091
```

```
## NIFTYIT     -6.554e-02  3.703e-02  -1.770   0.0772 .
## NIFTYBANK    1.342e-01  2.721e-02   4.930 1.04e-06 ***
## NIFTYFMCG   -3.493e-02  4.746e-02  -0.736   0.4619
## NIFTYREALTY  1.778e-02  2.541e-02   0.700   0.4844
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01202 on 650 degrees of freedom
## Multiple R-squared:  0.04538,    Adjusted R-squared:  0.03951
## F-statistic: 7.725 on 4 and 650 DF,  p-value: 4.382e-06
```

```
prediction1 = predict(fit1,testset)
mse1 = mean((prediction1-testset$NIFTY50)^2)
mse1
```

```
## [1] 0.0001277241
```

As, we see in the summary of model2 the variable NIFTY FMCG and NIFTY REALTY are not significant. So we don't take that variable as part of our model.

```
fit2 = lm(NIFTY50~NIFTYIT+NIFTYBANK,data = trainset)
summary((fit2))
```

```
##
## Call:
## lm(formula = NIFTY50 ~ NIFTYIT + NIFTYBANK, data = trainset)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.126085 -0.004993  0.000588  0.005970  0.072812
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0000712  0.0004708   0.151   0.8798
## NIFTYIT     -0.0789289  0.0325011  -2.429   0.0154 *
## NIFTYBANK    0.1359216  0.0271160   5.013 6.93e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01201 on 652 degrees of freedom
## Multiple R-squared:  0.04397,    Adjusted R-squared:  0.04103
## F-statistic: 14.99 on 2 and 652 DF,  p-value: 4.306e-07
```

```
prediction2 = predict(fit2,testset)
mse2 = mean((prediction2-testset$NIFTY50)^2)
mse2
```

```
## [1] 0.0001287693
```

So, two variables are significant. The Adj. R-squared value of the model2 is 0.04103 which is greater than model1. So, the Mean Square Error of the model2 is 0.0001287693, which seems very less but is more than model1's MSE.

Support Vector Regression

Support vector regression is a generalization of the support vector machine to the regression problem. It is a fast and accurate way of interpolating data sets. It is useful when you have an expensive function you want to approximate over a known domain. It learns quickly and is systematically improvable. For building a model using SVR we use same variables obtained earlier based on Mallow Cp.

```
set.seed(1)
model3 <- svm(NIFTY50~NIFTYIT+NIFTYBANK+NIFTYFMCG+NIFTYREALTY,data = trainset,
                  type = "eps-regression")
```

Now, let us test this model using the test dataset.

```
set.seed(1)
p3 <- predict(model3, newdata=testset)
e3 <- mean((testset$NIFTY50 - p3)^2)
e3
```

```
## [1] 0.000128034
```

So, the Mean Square Error of the model2 is 0.000128034, which seems very less.So, the model might be a good model as well.

##(4) Random Forest Regression:

```
set.seed(1)
model4 <-randomForest(x = trainset[,c("NIFTYBANK","NIFTYIT",
                                      "NIFTYFMCG","NIFTYREALTY")],
                       y = trainset$NIFTY50, ntree = 501)
```

Now, let us test this model using the test dataset.

```
p4 <- predict(model4, newdata=testset)
e4 <- mean((testset$NIFTY50 - p4)^2)
e4
```

```
## [1] 0.0001312273
```

So, the Mean Square Error of the model2 is 0.0001296344, which seems very less. So, the model might be a good model as well.

```
 Comparison table of MSE of all four models:
```

```
data.frame("MSE"=c("model1"=e1,"model2"=mse2,"model3"=e3,"model4"=e4))
```

```
##                  MSE
## model1 0.0001293893
## model2 0.0001287693
## model3 0.0001280340
## model4 0.0001312273
```

Based on Mean Square Error (MSE) we can say that model 3 i.e. Support Vector Regression is the best model.

#Question 3:

Do the same approach as in question 2, but this time for a qualitative variable. #(1) Logistic Regression Model:

```
set.seed(1)
logfit=glm(NIFTY50~.,data=clas_trainset,family=binomial)
result1 = predict(logfit,clas_testset,type = "response")
test_predict1 = rep(1,nrow(clas_testset))
test_predict1[which(result1>=0.5)] <- "Bullish"
test_predict1[which(result1<0.5)] <- "Bearish"
res_table1 = table(test_predict1,clas_testset$NIFTY50)
acc1 = (res_table1[1,1]+res_table1[2,2])/sum(res_table1)
print(res_table1)
```

```
##
## test_predict1 Bearish Bullish
##       Bearish      63      77
##       Bullish     235     280
```

```
print(acc1)
```

```
## [1] 0.5236641
```

The accuracy of this regression on the test set is 0.5236641. Note that we include all the variable into the model. ##LDA

```
set.seed(1)
ldafit=lda(NIFTY50~.,data=clas_trainset)
result2 = predict(ldafit,clas_testset,type = "response")$class
res_table2 = table(result2,clas_testset$NIFTY50)
acc21 = (res_table2[1,1]+res_table2[2,2])/sum(res_table2)
print(res_table2)
```

```
##
## result2   Bearish Bullish
##   Bearish      63      76
##   Bullish     235     281
```

```
print(acc21)
```

```
## [1] 0.5251908
```

Using the LDA method we can get the accuracy of 0.5251908 ###QDA

```
set.seed(1)
qdafit = qda(NIFTY50~.,data=clas_trainset)
result3 = predict(qdafit,clas_testset,type = "response")$class
res_table3 = table(result3,clas_testset$NIFTY50)
acc23 = (res_table3[1,1]+res_table3[2,2])/sum(res_table3)
print(res_table3)
```

```
##
## result3   Bearish Bullish
##   Bearish      76      99
##   Bullish     222     258
```

```
print(acc23)
```

```
## [1] 0.5099237
```

Using the QDA method we can get the accuracy of 0.5099237 on the test set which is not better than the LDA method. # SVM Linear

```
set.seed(1)
tune.out = tune(svm,NIFTY50~.,data = clas_trainset,kernel="linear",
ranges = list(cost=c(0.001,0.01,0.1,1.5,10,100)))

bestmod41 = tune.out$best.model
result41 = predict(bestmod41,clas_testset)
res_table61 = table(result41,clas_testset$NIFTY50)
acc41 = (res_table61[1,1]+res_table61[2,2])/sum(res_table61)
print(acc41)
```

```
## [1] 0.5450382
```

```
conclusion2<- data.frame(c(acc1,acc21,acc23,acc41),
                         row.names=c("logistic regression","LDA","QDA","SVM"))
colnames(conclusion2) = "test accuracy"
print(conclusion2)
```

```
##                     test accuracy
## logistic regression     0.5236641
## LDA                     0.5251908
## QDA                     0.5099237
## SVM                     0.5450382
```

So the SVM will give us the highest test accuracy result. #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.

```
REQ<- read_csv("~/Desktop/RCODED2022/REQ.csv")
```

```
## Rows: 1259 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (6): SPY, QQQ, SPYr, QQQr, CSPY, CQQQ
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
SPYframe <- data.frame(REQ$SPYr,REQ$CSPY)
colnames(SPYframe)<-c("SPYreturn","BCSPY")
head(SPYframe)
```

```
##   SPYreturn BCSPY
## 1 -0.005557     0
## 2 -0.001966     0
## 3  0.011490     1
## 4 -0.003305     0
## 5 -0.012995     0
## 6 -0.003543     0
```

```r
QQQframe<- data.frame(REQ$QQQr,REQ$CQQQ)
colnames(QQQframe)<-c("QQQreturn","BCQQQ")
head(QQQframe)
```

```
##   QQQreturn BCQQQ
## 1 -0.009101     0
## 2 -0.004850     0
## 3  0.015359     1
## 4 -0.002023     0
## 5 -0.016320     0
## 6 -0.006340     0
```

##(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
set.seed(personal)
tune.out1 = tune(svm, SPYframe$BCSPY~SPYframe$SPYreturn,
                 data = SPYframe, kernel = "linear", ranges =
list(cost = c(0.01, 0.1, 1, 5, 10, 100)))
summary(tune.out1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  0.01
##
## - best performance: 0.1693962
##
## - Detailed performance results:
##    cost     error  dispersion
## 1 1e-02 0.1693962 0.001058822
## 2 1e-01 0.1749708 0.001144377
## 3 1e+00 0.1759687 0.001285223
## 4 5e+00 0.1760355 0.001349819
## 5 1e+01 0.1761104 0.001434044
## 6 1e+02 0.1761018 0.001442704
```

```r
tune.out2 = tune(svm, QQQframe$BCQQQ~QQQframe$QQQreturn,
                 data = QQQframe, kernel = "linear", ranges =
list(cost = c(0.01, 0.1, 1, 5, 10, 100)))
summary(tune.out2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  0.01
##
## - best performance: 0.1478152
##
## - Detailed performance results:
##    cost     error   dispersion
## 1 1e-02 0.1478152 0.0009830953
## 2 1e-01 0.1486310 0.0008955800
## 3 1e+00 0.1490398 0.0009017884
## 4 5e+00 0.1490667 0.0009655054
## 5 1e+01 0.1490626 0.0009637477
## 6 1e+02 0.1490883 0.0009213236
```

For both, when the cost equals to 0.01, the error reaches the lowest level. ##(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.

```r
set.seed(personal)
tune.out3 = tune(svm,SPYframe$BCSPY~SPYframe$SPYreturn,
                 data = SPYframe,kernel="radial",
                 ranges = list(cost=c(0.01,0.1,1,5,10,100),
                              gamma = c(0.01,0.1,1,3,5,10,100)))
summary(tune.out3)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##   100   100
##
## - best performance: 0.005008964
##
## - Detailed performance results:
##     cost gamma       error   dispersion
## 1  1e-02 1e-02 0.345750659 0.0023966703
## 2  1e-01 1e-02 0.167312402 0.0037966397
## 3  1e+00 1e-02 0.137559161 0.0009596839
```

```
## 4  5e+00 1e-02 0.128816517 0.0011033477
## 5  1e+01 1e-02 0.126077632 0.0004902363
## 6  1e+02 1e-02 0.120663278 0.0043903619
## 7  1e-02 1e-01 0.142297395 0.0027965481
## 8  1e-01 1e-01 0.099661854 0.0013040600
## 9  1e+00 1e-01 0.085817029 0.0013681214
## 10 5e+00 1e-01 0.079989790 0.0009877474
## 11 1e+01 1e-01 0.078240657 0.0008102863
## 12 1e+02 1e-01 0.073447044 0.0007718530
## 13 1e-02 1e+00 0.067681519 0.0007193267
## 14 1e-01 1e+00 0.049948866 0.0009418961
## 15 1e+00 1e+00 0.041012008 0.0006361107
## 16 5e+00 1e+00 0.037726469 0.0007180818
## 17 1e+01 1e+00 0.036413803 0.0006519407
## 18 1e+02 1e+00 0.033054679 0.0006212738
## 19 1e-02 3e+00 0.051006480 0.0007308201
## 20 1e-01 3e+00 0.032675725 0.0005072321
## 21 1e+00 3e+00 0.025857558 0.0003839591
## 22 5e+00 3e+00 0.023623015 0.0003781256
## 23 1e+01 3e+00 0.022833538 0.0003199719
## 24 1e+02 3e+00 0.020809540 0.0002741126
## 25 1e-02 5e+00 0.047155667 0.0007060461
## 26 1e-01 5e+00 0.027404288 0.0003850045
## 27 1e+00 5e+00 0.020948380 0.0002514057
## 28 5e+00 5e+00 0.019148429 0.0001972522
## 29 1e+01 5e+00 0.018534595 0.0002207073
## 30 1e+02 5e+00 0.016761304 0.0001973641
## 31 1e-02 1e+01 0.044665519 0.0007672344
## 32 1e-01 1e+01 0.022079013 0.0002158962
## 33 1e+00 1e+01 0.015875357 0.0001684078
## 34 5e+00 1e+01 0.014465641 0.0001593954
## 35 1e+01 1e+01 0.013983791 0.0001476736
## 36 1e+02 1e+01 0.012537386 0.0001751275
## 37 1e-02 1e+02 0.073817630 0.0027226253
## 38 1e-01 1e+02 0.014124308 0.0002591596
## 39 1e+00 1e+02 0.006366841 0.0001830009
## 40 5e+00 1e+02 0.005715285 0.0002062882
## 41 1e+01 1e+02 0.005542651 0.0002035158
## 42 1e+02 1e+02 0.005008964 0.0002046146
```

```r
set.seed(personal)
tune.out4 = tune(svm,QQQframe$BCQQQ~QQQframe$QQQreturn,
                 data = QQQframe,kernel="radial",
                 ranges = list(cost=c(0.01,0.1,1,5,10,100),
                               gamma = c(0.01,0.1,1,3,5,10,100)))
summary(tune.out4)
```

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

```
##   100   100
##
## - best performance: 0.005285562
##
## - Detailed performance results:
##      cost gamma       error   dispersion
## 1  1e-02 1e-02 0.329740526 0.0022846344
## 2  1e-01 1e-02 0.150699346 0.0031754233
## 3  1e+00 1e-02 0.129613383 0.0017273093
## 4  5e+00 1e-02 0.122393049 0.0012530173
## 5  1e+01 1e-02 0.120126190 0.0020159398
## 6  1e+02 1e-02 0.117145827 0.0128070431
## 7  1e-02 1e-01 0.129450143 0.0028834620
## 8  1e-01 1e-01 0.092334289 0.0012843726
## 9  1e+00 1e-01 0.079468350 0.0013711910
## 10 5e+00 1e-01 0.073832078 0.0011516191
## 11 1e+01 1e-01 0.071590407 0.0010419562
## 12 1e+02 1e-01 0.067152271 0.0015619599
## 13 1e-02 1e+00 0.059329532 0.0008029815
## 14 1e-01 1e+00 0.042650795 0.0007216310
## 15 1e+00 1e+00 0.035588700 0.0006395986
## 16 5e+00 1e+00 0.033042917 0.0004565887
## 17 1e+01 1e+00 0.032041138 0.0005788605
## 18 1e+02 1e+00 0.029195086 0.0004166782
## 19 1e-02 3e+00 0.045392374 0.0004521641
## 20 1e-01 3e+00 0.028844024 0.0003655889
## 21 1e+00 3e+00 0.023680331 0.0003079802
## 22 5e+00 3e+00 0.021770143 0.0004538592
## 23 1e+01 3e+00 0.021097900 0.0003637161
## 24 1e+02 3e+00 0.019212941 0.0003602333
## 25 1e-02 5e+00 0.043136095 0.0003366920
## 26 1e-01 5e+00 0.024766164 0.0003262609
## 27 1e+00 5e+00 0.019464059 0.0003278279
## 28 5e+00 5e+00 0.017655879 0.0003711749
## 29 1e+01 5e+00 0.016986330 0.0003462198
## 30 1e+02 5e+00 0.015149458 0.0002126324
## 31 1e-02 1e+01 0.043622115 0.0006327908
## 32 1e-01 1e+01 0.019934999 0.0003749515
## 33 1e+00 1e+01 0.014328083 0.0001889223
## 34 5e+00 1e+01 0.012875924 0.0001573577
## 35 1e+01 1e+01 0.012480140 0.0001205935
## 36 1e+02 1e+01 0.011257241 0.0001268442
## 37 1e-02 1e+02 0.087133603 0.0033951981
## 38 1e-01 1e+02 0.013660468 0.0002877039
## 39 1e+00 1e+02 0.006448777 0.0002417370
## 40 5e+00 1e+02 0.005990262 0.0002472911
## 41 1e+01 1e+02 0.005751193 0.0002424939
## 42 1e+02 1e+02 0.005285562 0.0002411005
```

For radial kernel,for both, when cost=100 and gamma = 100, the error will get the lowest level.

```
set.seed(personal)
tune.out4 = tune(svm,SPYframe$BCSPY~SPYframe$SPYreturn, data = SPYframe,kernel="polynomial",ranges = li
summary(tune.out4)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##   0.1      2
##
## - best performance: 0.3857429
##
## - Detailed performance results:
##   cost degree     error   dispersion
## 1 0.01      2 0.3857961 0.0007325168
## 2 0.10      2 0.3857429 0.0006799359
## 3 0.01      3 0.3907846 0.0002992808
## 4 0.10      3 0.3908073 0.0002907618
```

```
set.seed(personal)
tune.out5 = tune(svm,QQQframe$BCQQQ~QQQframe$QQQreturn, data = QQQframe,kernel="polynomial",ranges = lis
summary(tune.out5)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##   0.1      2
##
## - best performance: 0.3660504
##
## - Detailed performance results:
##    cost degree     error  dispersion
## 1  0.01      2 0.3664143 0.002991097
## 2  0.10      2 0.3660504 0.002831960
## 3 10.00      2 0.3660532 0.002815637
## 4  0.01      3 0.3761033 0.003292045
## 5  0.10      3 0.3761029 0.003290817
## 6 10.00      3 0.3668214 0.029793451
```

When using the polynomial basis kernel, for both the cv error reaches its lowest level for cost equals to 0.1 and dgree equals to 2.