FA590. Assignment #3.

2021-11-19

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: Timothy Borin

CWID: 10454256

Date: 11/11/21

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.

```
CWID = 10454256 #Place here your Campus wide ID number, this will personalize
#your results, but still maintain the reproduceable 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.
```

1 point for every item of every question. Total = 22

Question 1

You have to build a predictive model for targeting offers to consumers, and conduct some model performance analytics on the result.

class: A factor with levels CH and MM indicating whether the customer purchased Citrus Hill or Minute Maid Orange Juice WeekofPurchase: Week of purchase StoreID: Store ID PriceCH: Price charged for CH PriceMM: Price charged for MM DiscCH: Discount offered for CH DiscMM: Discount offered for MM SpecialCH: Indicator of special on CH SpecialMM: Indicator of special on MM LoyalCH: Customer brand loyalty for CH SalePriceMM: Sale price for MM SalePriceCH: Sale price for CH PriceDiff: Sale price of MM less sale price of CH Store7: A factor with levels No and Yes indicating whether the sale is at Store 7 PctDiscMM: Percentage discount for MM PctDiscCH: Percentage discount for CH ListPriceDiff: List price of MM less list price of CH STORE: Which of 5 possible stores the sale occured at

We will use historical data on past customer responses (contained in the file marketing1.csv) in order to build a classification model to forecast the customers' decision to purchase Citrus Hill or Minute Maid.

(a) You must randomly split your data set using 70% and 30% of the observations for the training and test data set respectively.

```
df=read.csv("marketing1.csv") # Import csv
df=data.frame(df) # make dataframe with marketing data
df=na.omit(df) # get rid of NAs
class(df$class) #check class of var 'class'
```

```
## [1] "character"
```

```
df$class=as.factor(df$class) # change var 'class' to type 'factor'
df$Store7=as.factor(df$Store7)# change var 'Store7' to type 'factor'
class(df$class) #check to ensure var is now proper type
```

```
## [1] "factor"
```

```
class(df$Store7)
```

```
## [1] "factor"
```

```
dt=sort(sample(nrow(df), nrow(df)*.7)) #Sorting data into training and test
train<-df[dt,]
test<-df[-dt,]
```

(b) Fit a tree to the training data, with "class" as the response and the other variables as predictors. Use the summary() function to produce summary statistics about the tree, and describe the results obtained. What is the training error rate? How many terminal nodes does the tree have?

```
library(tree)
library(MASS)
library(randomForest)
#rm(tree.oj) #REMEMBER TO RM THE TREE.OJ SO IT RESETS BEFORE RUNNING FINAL TIME
tree.oj=tree(class~.,data=train) #build decision tree
summary(tree.oj)
```

```
##
## Classification tree:
## tree(formula = class ~ ., data = train)
## Variables actually used in tree construction:
## [1] "LoyalCH"     "SalePriceMM" "SpecialCH"   "PriceDiff"
## Number of terminal nodes:  9
## Residual mean deviance:  0.7546 = 558.4 / 740
## Misclassification error rate: 0.1696 = 127 / 749
```

```
#Training Error Rate:0.1696 or 16.96%
#Total No. of Terminal Nodes:9
```

(c) Type in the name of the tree object in order to get a detailed text output. Pick one of the terminal nodes, and interpret the information displayed.

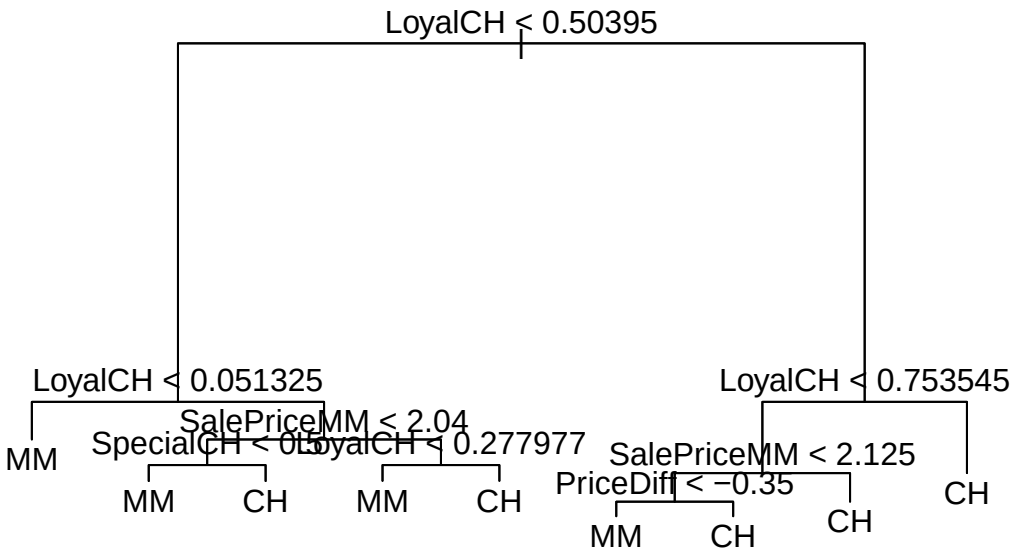
```
tree.oj
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 749 993.20 CH ( 0.62216 0.37784 )
##    2) LoyalCH < 0.50395 328 400.00 MM ( 0.29878 0.70122 )
##      4) LoyalCH < 0.051325 55  17.18 MM ( 0.03636 0.96364 ) *
##      5) LoyalCH > 0.051325 273 354.10 MM ( 0.35165 0.64835 )
##       10) SalePriceMM < 2.04 146 158.50 MM ( 0.23288 0.76712 )
##         20) SpecialCH < 0.5 118 107.40 MM ( 0.16949 0.83051 ) *
##         21) SpecialCH > 0.5 28  38.82 CH ( 0.50000 0.50000 ) *
##       11) SalePriceMM > 2.04 127 176.00 MM ( 0.48819 0.51181 )
##         22) LoyalCH < 0.277977 41  47.69 MM ( 0.26829 0.73171 ) *
##         23) LoyalCH > 0.277977 86 116.20 CH ( 0.59302 0.40698 ) *
##    3) LoyalCH > 0.50395 421 318.70 CH ( 0.87411 0.12589 )
##      6) LoyalCH < 0.753545 173 200.40 CH ( 0.73410 0.26590 )
##       12) SalePriceMM < 2.125 113 149.10 CH ( 0.62832 0.37168 )
##         24) PriceDiff < -0.35 16  17.99 MM ( 0.25000 0.75000 ) *
##         25) PriceDiff > -0.35 97 120.00 CH ( 0.69072 0.30928 ) *
##       13) SalePriceMM > 2.125 60  29.39 CH ( 0.93333 0.06667 ) *
##      7) LoyalCH > 0.753545 248  63.75 CH ( 0.97177 0.02823 ) *
```

```
#Outputs tree information
#Terminal Node : 9
#For those customers with brand loyalty to CH greater than 0.5, if CH is 0.38 cents cheaper or less, th
```

(d) Create a plot of the tree, and interpret the results.

```
#Plot and text run simultaneously produces simple visualization. rpart.plot is much more visually appea
plot(tree.oj)
text(tree.oj)
```




```
# I'm sure I dont have to tell you this but make sure to run both of the above commands simultaneously
#Brand loyalty is a big deciding factor out of this group of customers, followed by PriceDiff. If a cus
#likely to end up purchasing MM, but CH still made up the majority of all customers, regardless of bran
```

(e) Predict the response on the test data, and produce a confusion matrix comparing the test labels to the predicted test labels. What is the test error rate?

```
tree.oj.test=tree(class~.,data=test) #test tree
pred=predict(tree.oj.test,test,type="class") #Prediction formula
table(pred,test$class) #Confusion matrix
```

```
##
## pred  CH  MM
##   CH 169  18
##   MM  18 116
```

```
#Test Error Rate : 0.11215 or 11.22%
```

(f) Apply the cv.tree() function to the training set in order to determine the optimal tree size.

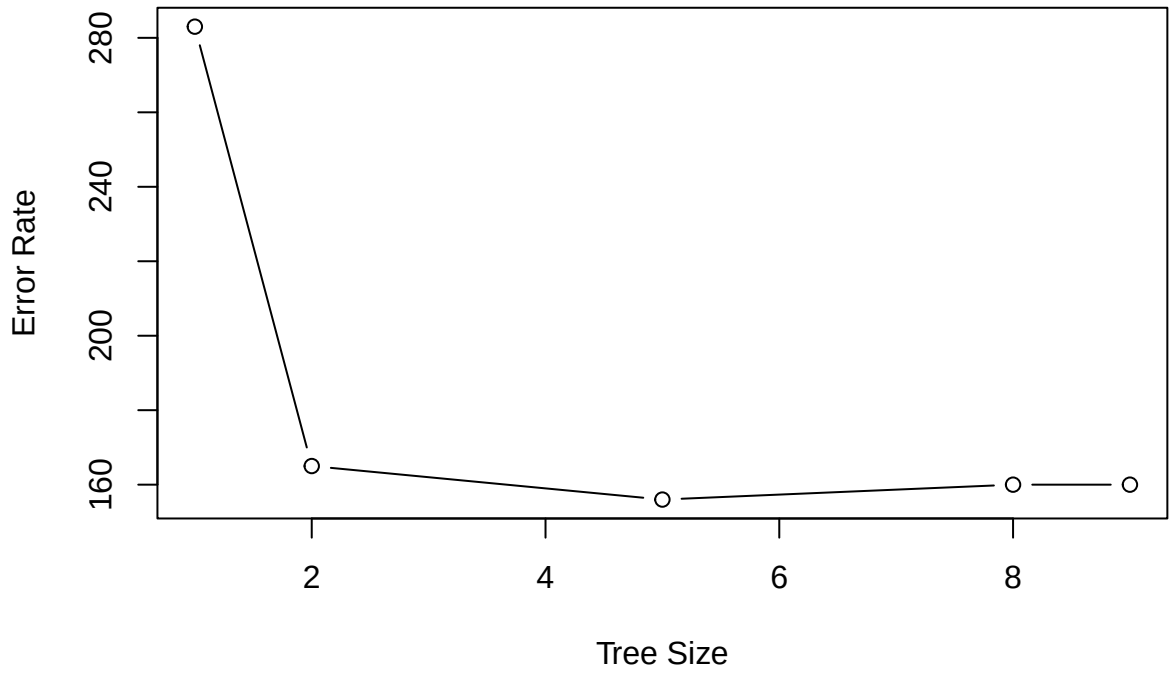
```
tree.oj.prune=cv.tree(tree.oj, FUN=prune.misclass, K=10) #cv.tree function
tree.oj.prune
```

```
## $size
## [1] 9 8 5 2 1
##
## $dev
## [1] 160 160 156 165 283
##
## $k
## [1]       -Inf   0.000000   2.666667   5.333333 132.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

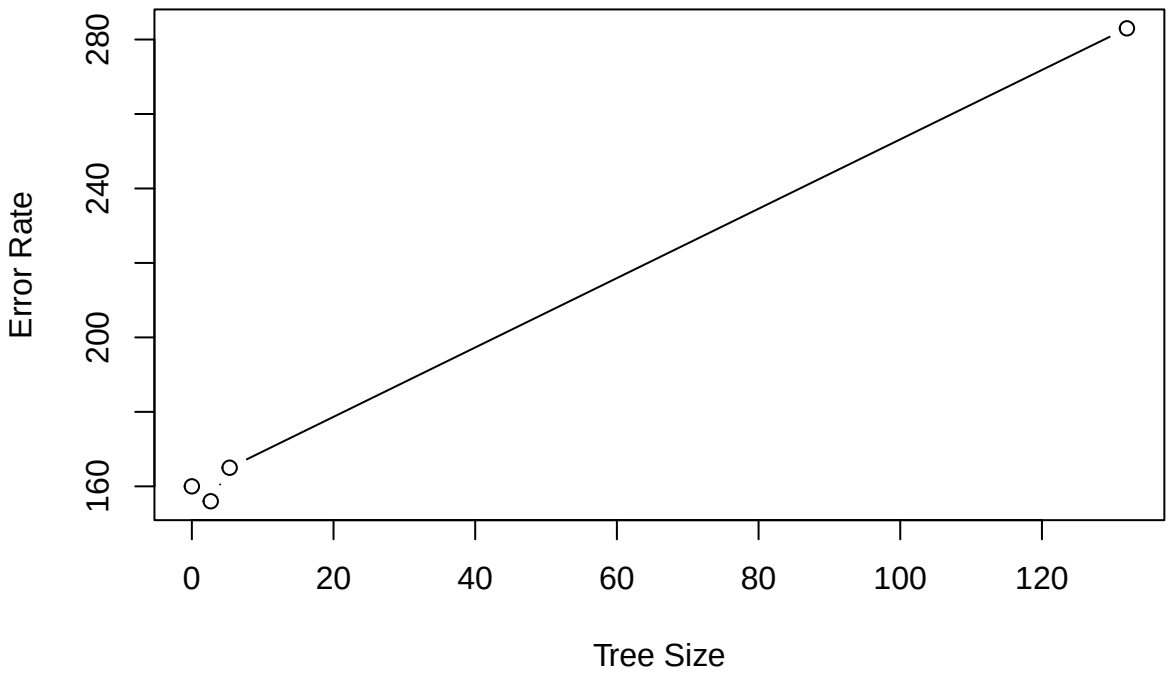
```
# Optimal Tree Size : 5 Terminal Nodes
```

(g) Produce a plot with tree size on the x-axis and cross-validated classification error rate on the y-axis.

```
plot(tree.oj.prune$size,tree.oj.prune$dev,xlab="Tree Size", ylab="Error Rate", type="b") #Plot for two
```




```
plot(tree.oj.prune$k,tree.oj.prune$dev,xlab="Tree Size", ylab="Error Rate", type="b")
```



(h) Which tree size corresponds to the lowest cross-validated classification error rate?

```
# The lowest Cross-Validated classification error rate belongs to tree of size 5
```

(i) Produce a pruned tree corresponding to the optimal tree size obtained using cross-validation. If cross-validation does not lead to a selection of a pruned tree, then create a pruned tree with five terminal nodes.

```
library(tree) #Necessary library implementation
prune.oj=prune.misclass(tree.oj, best=4) #Prune the training tree
summary(prune.oj) #Summary of pruned training tree
```

```
##
## Classification tree:
## snip.tree(tree = tree.oj, nodes = c(10L, 3L))
## Variables actually used in tree construction:
## [1] "LoyalCH"     "SalePriceMM"
## Number of terminal nodes:  5
## Residual mean deviance:  0.8848 = 658.3 / 744
## Misclassification error rate: 0.1802 = 135 / 749
```

```
prune.oj.test=prune.misclass(tree.oj.test, best=4) #Prune the test tree
summary(prune.oj.test) #Summary of pruned test tree
```

```
##
## Classification tree:
## snip.tree(tree = tree.oj.test, nodes = c(2L, 7L, 13L, 12L))
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  4
## Residual mean deviance:  0.7369 = 233.6 / 317
## Misclassification error rate: 0.1371 = 44 / 321
```

```
#Training Error : 0.1802 or 18.02%
#Test Error : 0.1371 or 13.71%
```

(j) Compare the training and test error rates between the pruned and unpruned trees. Which is higher?

```
# The pruned trees have a higher error rate than the unpruned tree.
```

Question 2

You have to predict the daily return of the bitcoin for the next period. The file BTCreturns.csv includes the following variables:

Daily return based on adjusted daily closing prices: Core ETFs that represent complete market: VTI: Vanguard Total Stock Market ETF return VXUS: Vanguard Total International Stock ETF return BND: Vanguard Total Bond Market ETF return BNDX: Vanguard Total International Bond ETF return

Investment style: VUG Vanguard Growth ETF return VTV Vanguard Value ETF return

Sectors: Technology (growth) and energy (value) QQQ Invesco Nasdaq return XLE Energy ETF return

Cryptocurrencies: ETH Ethereum return ETH_V Ethereum trading volume BTC Bitcoin return BTC_V Bitcoin trading volume

Additional market factors from 5 factors Fama French model: RM-Rf : market return minus risk free rate (market risk premium) SMB: Small Minus Big (firm size): difference of average return on 9 small and 9 big stock portfolios HML: High Minus Low (value): difference of average return on 2 value and 2 growth portfolios RMW (Robust Minus Weak): difference of average return on 2 robust and 2 weak operating profitability portfolios CMA (Conservative Minus Aggressive): difference of average return on 2 conservative and 2 aggressive investment portfolios

a) Generate a new variable BTC1 which is the BTC return of the next day. Make sure that you sort the dataset according to "date." After sorting, you can remove "date" from your data set. Split your data set into 70% and 30% training and testing datasets respectively.

```
BitC=read.csv("BTCreturns.csv") #Import Bitcoin csv file
BTC1=vector(length=1544) #BTC1 var creation
BitC=data.frame(BitC) #Ensuring BTC is a dataframe
BitC$Date=as.Date(BitC$Date, format = "%m/%d/%Y") #Converting Date from type 'chr' to type 'date'
BitC=BitC[order(BitC$Date),] #Sorting by date
for (i in 1:1544){ #For loop to assign BTC1 to next day's return
  BTC1[i]=BitC$BTC[i+1]
}
BTC1=as.data.frame(BTC1) #Making sure new var is data frame
BitC=cbind(BitC,BTC1) #Binding new var to dataframe
```

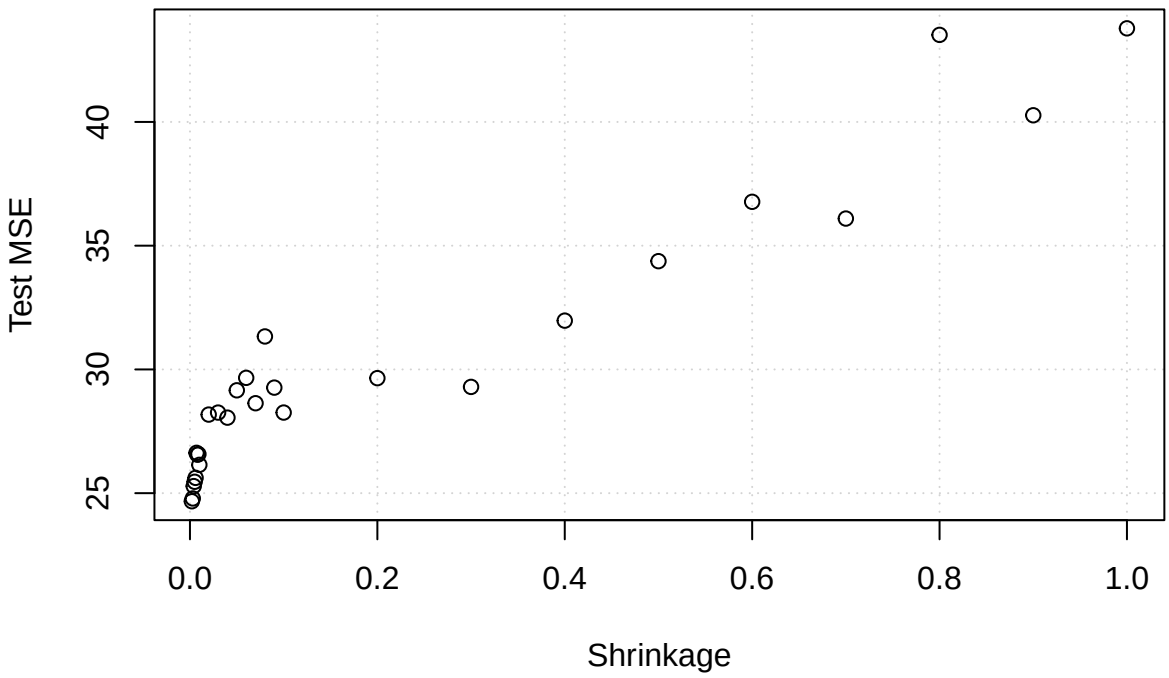
```
drops="Date" #Setting Drop var
BitC=BitC[,!(names(BitC) %in% drops)] #Dropping Dates from Dataframe
BitC=na.omit(BitC) #Ensuring no NAs exist
BitC=BitC[-c(1544),]# Removing last row with the NA, NA still shows up
#1,543 rows * 0.7 is 1080.1, so training set will go from 1-1080, and test from 1081-1543
train1<-BitC[1:1080,]
test1<-BitC[1081:1543,]
train1=as.data.frame(train1)
test1=as.data.frame(test1)
```

(b) Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter lambda. Hint: use the gbm package, the gbm() function with the option distribution="Gaussian" to apply boosting to a regression problem.

```
library(gbm)
lambdas=c(c(), seq(0.002, 0.01, by=0.001)) #Creating an array of different lambda values to test from
lambdas=c(lambdas, seq(0.02, 0.1, by=0.01))
lambdas=c(lambdas, seq(0.2,1, by=0.1))
lambda.length=length(lambdas) #Creating lambda length var for future use
train1.err=rep(NA, lambda.length) # Training and test arrays to store values in
test1.err=rep(NA, lambda.length)
for (i in 1:lambda.length){ # For loop : Runs GBM for value lambda from the array, predicts both test a
  train.boost=gbm(BTC1~.,data=train1,distribution="gaussian",n.trees=1000,shrinkage=lambdas[i]) # in an
  train1.pred=predict(train.boost, train1, n.trees=1000)
  test1.pred=predict(train.boost, test1, n.trees=1000)
  train1.err[i]=mean((train1$BTC1 - train1.pred)^2)
  test1.err[i]=mean((test1$BTC1 - test1.pred)^2)
}
```

(c) Produce a plot with different shrinkage values on the x-axis and the corresponding test set MSE on the y-axis.

```
plot(x=lambdas, y=test1.err, xlab = "Shrinkage", ylab= "Test MSE", grid(nx=NULL)) #Plot grid based on T
```



(d) Using the best shrinkage value, retrain your boosting model with the training dataset. What is the test set MSE for this approach?

```
lambdas[which.min(test1.err)] #Finding the optimal lambda value from the prev test set to set shrinkage
```

```
## [1] 0.002
```

```
train.boost1=gbm(BTC1~.,data=train1,distribution="gaussian",n.trees=1000,shrinkage=0.002) #Retraining d
train2.pred=predict(train.boost1, train1, n.trees=1000) #running the same training and test sets as ear
test2.pred=predict(train.boost1, test1, n.trees=1000)
train2.err=mean((train1$BTC1 - train2.pred)^2)
test2.err=mean((test1$BTC1 - test2.pred)^2)
test2.err #Finding the test set MSE
```

```
## [1] 24.65525
```

```
#Test Set SME of 24.65525
#Creating the array to store MSEs and type of model
vec1=c("Boosting", "Bagging", "randomForest", "SVM", "SVMNL", "GAM")
vec2=c(test2.err)
```

(e) Apply bagging to the training set. What is the test set MSE for this approach?

```
library(randomForest)
library(MASS)
library(rpart)
library(rattle)
bag.tree=randomForest(BTC1~.,data=train1, mtry=17, importance=TRUE) #17 mrty uses all but the indep. va
yhat.bag=predict(bag.tree,newdata=test1) #creating prediction value
rd.test=test1$BTC1 # setting test val for MSE calculation, this wont change for all the formulas so thi
var3=mean((yhat.bag-rd.test)^2) #MSE Calculation
vec2=c(vec2, var3) #MSE Storage
var3
```

```
## [1] 55.79212
```

```
# Test MSE is 55.79212
```

(f) Apply random forest to the training set. What is the test set MSE for this approach? Which variables appear to be the most important predictors in the random forest model?

```
randforest=randomForest(BTC1~., data=train1, mtry=5, importance=TRUE) #RandomForest calculation, mtry=5
yhat.rf=predict(randforest,newdata=test1) #MSE calculations
var4=mean((yhat.rf-rd.test)^2)
vec2=c(vec2,var4)
#Test MSE is 43.31796
```

(g) Apply support vector machine to the training set. What is the test set MSE for this approach?

```
library(e1071)
main=svm(BTC1~., data=train1, kernel="linear", cost=0.01, scale=T) #Linear kernel, cost was changed thr
yhat.svm=predict(main, newdata=test1)
var5=mean((yhat.svm-rd.test)^2)
vec2=c(vec2,var5)
# Test Set MSE for Multiple Costs
# Cost 0.01: 26.36608
# Cost 0.1 : 28.58219
# Cost 1   : 30.02192
# Cost 100 : 30.04326
# Cost 10000 : 49.47812
# The lower the cost, the better the test set MSE in this approach
```

(h) Apply support vector machine with a nonlinear kernel to the training set. What is the test set MSE for this approach?

```
main2=svm(BTC1~., data=train1, kernel="radial", gamma = 1, cost = 10, scale = FALSE) # Introducing nonl
yhat2.svm=predict(main2, newdata=test1)                                               # during running
var6=mean((yhat2.svm-rd.test)^2)
vec2=c(vec2, var6)
# Test cost 0.01 :24.49791
# Test cost 0.1  :24.49579
# Test cost 1    :24.49428
# Test cost 10   :24.49359
# Test cost 100  :24.49516
```

```
# Test cost 1000 :24.49516
# Test cost 10000:24.49516
# the best test set MSE is 24.49359 with cost value 10
```

(i) Perform subset selection (your choice on how) in order to identify a satisfactory model that uses just a subset of the predictors (if your approach suggests using all of the predictors, then follow your results and use them all).I suggest that you use the function stepAIC.

```
library(MASS)
testvar=lm(BTC1~., data=train1) #Getting the dataset prepped to run stepAIC to find the best subset
stepAIC(testvar, direction="both") #Finding the ideal subset
```

```
## Start:  AIC=3333.49
## BTC1 ~ VTI + VXUS + BND + BNDX + VUG + VTV + QQQ + XLE + RM.Rf +
##     SMB + HML + RMW + CMA + ETH + ETH_V + BTC + BTC_V
##
##         Df Sum of Sq   RSS    AIC
## - HML    1    0.0246 22878 3331.5
## - BND    1    0.1310 22878 3331.5
## - VTV    1    0.1432 22878 3331.5
## - BNDX   1    0.1778 22879 3331.5
## - RMW    1    0.6520 22879 3331.5
## - BTC_V  1    0.8177 22879 3331.5
## - QQQ    1    0.9769 22879 3331.5
## - VTI    1    2.1764 22881 3331.6
## - ETH    1    3.6631 22882 3331.7
## - VXUS   1    5.7542 22884 3331.8
## - VUG    1    6.9930 22885 3331.8
## - BTC    1    8.2581 22887 3331.9
## - ETH_V  1    8.4112 22887 3331.9
## - RM.Rf  1    8.8670 22887 3331.9
## - SMB    1   11.2595 22890 3332.0
## - XLE    1   19.6135 22898 3332.4
## - CMA    1   22.7772 22901 3332.6
## <none>               22878 3333.5
##
## Step:  AIC=3331.49
## BTC1 ~ VTI + VXUS + BND + BNDX + VUG + VTV + QQQ + XLE + RM.Rf +
##     SMB + RMW + CMA + ETH + ETH_V + BTC + BTC_V
##
##         Df Sum of Sq   RSS    AIC
## - BND    1    0.1115 22878 3329.5
## - VTV    1    0.1307 22878 3329.5
## - BNDX   1    0.1811 22879 3329.5
## - RMW    1    0.6896 22879 3329.5
## - BTC_V  1    0.8317 22879 3329.5
## - QQQ    1    0.9927 22879 3329.5
## - VTI    1    2.2054 22881 3329.6
## - ETH    1    3.6877 22882 3329.7
## - VXUS   1    5.9549 22884 3329.8
## - VUG    1    7.5946 22886 3329.8
## - BTC    1    8.2510 22887 3329.9
```

```
## - ETH_V  1    8.4719 22887 3329.9
## - RM.Rf  1    9.0543 22887 3329.9
## - SMB    1   11.2429 22890 3330.0
## - XLE    1   21.3580 22900 3330.5
## - CMA    1   22.8949 22901 3330.6
## <none>               22878 3331.5
## + HML    1    0.0246 22878 3333.5
##
## Step:  AIC=3329.49
## BTC1 ~ VTI + VXUS + BNDX + VUG + VTV + QQQ + XLE + RM.Rf + SMB +
##     RMW + CMA + ETH + ETH_V + BTC + BTC_V
##
##         Df Sum of Sq   RSS    AIC
## - BNDX   1    0.0771 22879 3327.5
## - VTV    1    0.1363 22879 3327.5
## - RMW    1    0.7015 22879 3327.5
## - BTC_V  1    0.8515 22879 3327.5
## - QQQ    1    0.9719 22879 3327.5
## - VTI    1    2.1417 22881 3327.6
## - ETH    1    3.6466 22882 3327.7
## - VXUS   1    6.1385 22885 3327.8
## - VUG    1    7.5023 22886 3327.8
## - BTC    1    8.2227 22887 3327.9
## - ETH_V  1    8.5415 22887 3327.9
## - RM.Rf  1    8.9491 22887 3327.9
## - SMB    1   11.2721 22890 3328.0
## - XLE    1   21.4100 22900 3328.5
## - CMA    1   22.7896 22901 3328.6
## <none>               22878 3329.5
## + BND    1    0.1115 22878 3331.5
## + HML    1    0.0051 22878 3331.5
##
## Step:  AIC=3327.5
## BTC1 ~ VTI + VXUS + VUG + VTV + QQQ + XLE + RM.Rf + SMB + RMW +
##     CMA + ETH + ETH_V + BTC + BTC_V
##
##         Df Sum of Sq   RSS    AIC
## - VTV    1    0.1469 22879 3325.5
## - RMW    1    0.6899 22879 3325.5
## - BTC_V  1    0.8577 22879 3325.5
## - QQQ    1    0.9907 22880 3325.5
## - VTI    1    2.1964 22881 3325.6
## - ETH    1    3.6344 22882 3325.7
## - VXUS   1    6.1615 22885 3325.8
## - VUG    1    7.8551 22886 3325.9
## - BTC    1    8.2462 22887 3325.9
## - ETH_V  1    8.5565 22887 3325.9
## - RM.Rf  1    9.6254 22888 3326.0
## - SMB    1   11.2123 22890 3326.0
## - XLE    1   21.5574 22900 3326.5
## - CMA    1   23.0097 22902 3326.6
## <none>               22879 3327.5
## + BNDX   1    0.0771 22878 3329.5
## + HML    1    0.0170 22879 3329.5
```

```
## + BND    1    0.0075 22879 3329.5
##
## Step:  AIC=3325.5
## BTC1 ~ VTI + VXUS + VUG + QQQ + XLE + RM.Rf + SMB + RMW + CMA +
##     ETH + ETH_V + BTC + BTC_V
##
##         Df Sum of Sq   RSS    AIC
## - RMW    1     0.628 22879 3323.5
## - BTC_V  1     0.848 22880 3323.5
## - QQQ    1     1.046 22880 3323.6
## - ETH    1     3.633 22882 3323.7
## - VTI    1     5.637 22884 3323.8
## - VXUS   1     6.087 22885 3323.8
## - BTC    1     8.425 22887 3323.9
## - ETH_V  1     8.527 22887 3323.9
## - RM.Rf  1    10.948 22890 3324.0
## - VUG    1    20.688 22899 3324.5
## - XLE    1    21.667 22900 3324.5
## - CMA    1    23.798 22902 3324.6
## - SMB    1    41.240 22920 3325.4
## <none>               22879 3325.5
## + VTV    1     0.147 22879 3327.5
## + BNDX   1     0.088 22879 3327.5
## + HML    1     0.007 22879 3327.5
## + BND    1     0.007 22879 3327.5
##
## Step:  AIC=3323.53
## BTC1 ~ VTI + VXUS + VUG + QQQ + XLE + RM.Rf + SMB + CMA + ETH +
##     ETH_V + BTC + BTC_V
##
##         Df Sum of Sq   RSS    AIC
## - BTC_V  1     0.876 22880 3321.6
## - QQQ    1     1.055 22880 3321.6
## - ETH    1     3.554 22883 3321.7
## - VTI    1     5.534 22885 3321.8
## - VXUS   1     6.030 22885 3321.8
## - BTC    1     8.439 22888 3321.9
## - ETH_V  1     8.595 22888 3321.9
## - RM.Rf  1    10.948 22890 3322.0
## - VUG    1    20.985 22900 3322.5
## - XLE    1    21.531 22901 3322.5
## - CMA    1    23.259 22903 3322.6
## - SMB    1    40.725 22920 3323.5
## <none>               22879 3323.5
## + RMW    1     0.628 22879 3325.5
## + VTV    1     0.085 22879 3325.5
## + BNDX   1     0.073 22879 3325.5
## + HML    1     0.029 22879 3325.5
## + BND    1     0.012 22879 3325.5
##
## Step:  AIC=3321.57
## BTC1 ~ VTI + VXUS + VUG + QQQ + XLE + RM.Rf + SMB + CMA + ETH +
##     ETH_V + BTC
##
```

```
##         Df Sum of Sq   RSS    AIC
## - QQQ    1     1.021 22881 3319.6
## - ETH    1     3.858 22884 3319.8
## - VTI    1     5.464 22886 3319.8
## - VXUS   1     6.068 22886 3319.9
## - BTC    1     9.199 22889 3320.0
## - RM.Rf  1    10.869 22891 3320.1
## - VUG    1    21.000 22901 3320.6
## - XLE    1    21.423 22902 3320.6
## - CMA    1    23.751 22904 3320.7
## - SMB    1    40.639 22921 3321.5
## <none>               22880 3321.6
## - ETH_V  1    67.415 22948 3322.8
## + BTC_V  1     0.876 22879 3323.5
## + RMW    1     0.657 22880 3323.5
## + BNDX   1     0.079 22880 3323.6
## + VTV    1     0.077 22880 3323.6
## + HML    1     0.044 22880 3323.6
## + BND    1     0.016 22880 3323.6
##
## Step:  AIC=3319.62
## BTC1 ~ VTI + VXUS + VUG + XLE + RM.Rf + SMB + CMA + ETH + ETH_V +
##     BTC
##
##         Df Sum of Sq   RSS    AIC
## - ETH    1     3.926 22885 3317.8
## - VXUS   1     6.554 22888 3317.9
## - VTI    1     7.373 22889 3318.0
## - BTC    1     9.106 22890 3318.1
## - RM.Rf  1    13.111 22894 3318.2
## - XLE    1    21.001 22902 3318.6
## - CMA    1    28.611 22910 3319.0
## - VUG    1    30.078 22911 3319.0
## - SMB    1    40.100 22921 3319.5
## <none>               22881 3319.6
## - ETH_V  1    66.968 22948 3320.8
## + QQQ    1     1.021 22880 3321.6
## + BTC_V  1     0.843 22880 3321.6
## + RMW    1     0.666 22881 3321.6
## + VTV    1     0.117 22881 3321.6
## + BNDX   1     0.100 22881 3321.6
## + HML    1     0.066 22881 3321.6
## + BND    1     0.006 22881 3321.6
##
## Step:  AIC=3317.81
## BTC1 ~ VTI + VXUS + VUG + XLE + RM.Rf + SMB + CMA + ETH_V + BTC
##
##         Df Sum of Sq   RSS    AIC
## - BTC    1     5.853 22891 3316.1
## - VXUS   1     6.436 22892 3316.1
## - VTI    1     7.480 22893 3316.2
## - RM.Rf  1    13.110 22898 3316.4
## - XLE    1    20.566 22906 3316.8
## - CMA    1    27.873 22913 3317.1
```

```
## - VUG    1    29.226 22914 3317.2
## - SMB    1    39.223 22924 3317.7
## <none>               22885 3317.8
## - ETH_V  1    67.022 22952 3319.0
## + ETH    1     3.926 22881 3319.6
## + BTC_V  1     1.142 22884 3319.8
## + QQQ    1     1.090 22884 3319.8
## + RMW    1     0.585 22885 3319.8
## + VTV    1     0.120 22885 3319.8
## + HML    1     0.112 22885 3319.8
## + BNDX   1     0.088 22885 3319.8
## + BND    1     0.001 22885 3319.8
##
## Step:  AIC=3316.08
## BTC1 ~ VTI + VXUS + VUG + XLE + RM.Rf + SMB + CMA + ETH_V
##
##         Df Sum of Sq   RSS    AIC
## - VXUS   1     6.221 22897 3314.4
## - VTI    1     6.955 22898 3314.4
## - RM.Rf  1    12.535 22904 3314.7
## - XLE    1    20.567 22912 3315.1
## - CMA    1    27.772 22919 3315.4
## - VUG    1    29.679 22921 3315.5
## - SMB    1    38.294 22929 3315.9
## <none>               22891 3316.1
## - ETH_V  1    68.411 22959 3317.3
## + BTC    1     5.853 22885 3317.8
## + BTC_V  1     1.655 22889 3318.0
## + QQQ    1     0.969 22890 3318.0
## + ETH    1     0.673 22890 3318.1
## + RMW    1     0.650 22890 3318.1
## + VTV    1     0.256 22891 3318.1
## + BNDX   1     0.125 22891 3318.1
## + HML    1     0.078 22891 3318.1
## + BND    1     0.000 22891 3318.1
##
## Step:  AIC=3314.38
## BTC1 ~ VTI + VUG + XLE + RM.Rf + SMB + CMA + ETH_V
##
##         Df Sum of Sq   RSS    AIC
## - VTI    1     5.525 22903 3312.6
## - RM.Rf  1    11.906 22909 3312.9
## - XLE    1    25.637 22923 3313.6
## - CMA    1    26.921 22924 3313.6
## - VUG    1    29.510 22927 3313.8
## - SMB    1    36.624 22934 3314.1
## <none>               22897 3314.4
## - ETH_V  1    68.574 22966 3315.6
## + VXUS   1     6.221 22891 3316.1
## + BTC    1     5.638 22892 3316.1
## + BTC_V  1     1.680 22896 3316.3
## + QQQ    1     1.431 22896 3316.3
## + ETH    1     0.659 22897 3316.3
## + RMW    1     0.593 22897 3316.3
```

```
## + HML    1     0.318 22897 3316.4
## + VTV    1     0.165 22897 3316.4
## + BNDX   1     0.157 22897 3316.4
## + BND    1     0.020 22897 3316.4
##
## Step:  AIC=3312.64
## BTC1 ~ VUG + XLE + RM.Rf + SMB + CMA + ETH_V
##
##         Df Sum of Sq   RSS    AIC
## - RM.Rf  1    13.915 22917 3311.3
## - XLE    1    26.364 22929 3311.9
## - CMA    1    33.338 22936 3312.2
## - VUG    1    33.466 22936 3312.2
## - SMB    1    41.554 22944 3312.6
## <none>               22903 3312.6
## - ETH_V  1    66.876 22970 3313.8
## + VTI    1     5.525 22897 3314.4
## + BTC    1     5.198 22898 3314.4
## + VXUS   1     4.791 22898 3314.4
## + VTV    1     3.757 22899 3314.5
## + QQQ    1     3.156 22900 3314.5
## + BTC_V  1     1.527 22901 3314.6
## + ETH    1     0.764 22902 3314.6
## + BNDX   1     0.509 22902 3314.6
## + RMW    1     0.495 22902 3314.6
## + HML    1     0.315 22902 3314.6
## + BND    1     0.068 22903 3314.6
##
## Step:  AIC=3311.29
## BTC1 ~ VUG + XLE + SMB + CMA + ETH_V
##
##         Df Sum of Sq   RSS    AIC
## - CMA    1    22.281 22939 3310.3
## <none>               22917 3311.3
## - XLE    1    44.767 22961 3311.4
## - VUG    1    53.984 22971 3311.8
## - ETH_V  1    65.228 22982 3312.4
## - SMB    1    67.391 22984 3312.5
## + RM.Rf  1    13.915 22903 3312.6
## + VXUS   1     8.945 22908 3312.9
## + VTV    1     8.794 22908 3312.9
## + HML    1     7.806 22909 3312.9
## + VTI    1     7.534 22909 3312.9
## + BTC    1     5.399 22911 3313.0
## + BNDX   1     2.342 22914 3313.2
## + QQQ    1     2.109 22915 3313.2
## + BND    1     2.051 22915 3313.2
## + BTC_V  1     1.625 22915 3313.2
## + RMW    1     1.019 22916 3313.2
## + ETH    1     0.560 22916 3313.3
##
## Step:  AIC=3310.34
## BTC1 ~ VUG + XLE + SMB + ETH_V
##
```

```
##         Df Sum of Sq   RSS    AIC
## - XLE    1    24.269 22963 3309.5
## - VUG    1    31.704 22971 3309.8
## <none>               22939 3310.3
## + CMA    1    22.281 22917 3311.3
## - ETH_V  1    63.672 23003 3311.3
## - SMB    1    68.976 23008 3311.6
## + QQQ    1     9.335 22930 3311.9
## + BTC    1     5.029 22934 3312.1
## + VXUS   1     4.977 22934 3312.1
## + RM.Rf  1     2.858 22936 3312.2
## + BNDX   1     2.674 22936 3312.2
## + BND    1     2.581 22936 3312.2
## + BTC_V  1     2.081 22937 3312.2
## + HML    1     1.337 22938 3312.3
## + VTV    1     0.464 22938 3312.3
## + ETH    1     0.454 22938 3312.3
## + VTI    1     0.301 22939 3312.3
## + RMW    1     0.084 22939 3312.3
##
## Step:  AIC=3309.48
## BTC1 ~ VUG + SMB + ETH_V
##
##         Df Sum of Sq   RSS    AIC
## - VUG    1    10.606 22974 3308.0
## <none>               22963 3309.5
## + XLE    1    24.269 22939 3310.3
## - ETH_V  1    61.180 23024 3310.4
## + RM.Rf  1    16.111 22947 3310.7
## + VXUS   1    15.289 22948 3310.8
## + HML    1    13.694 22950 3310.8
## + VTV    1     9.680 22954 3311.0
## + VTI    1     9.003 22954 3311.1
## + BNDX   1     5.550 22958 3311.2
## + BTC    1     5.257 22958 3311.2
## + BND    1     4.916 22958 3311.3
## + QQQ    1     2.385 22961 3311.4
## + CMA    1     1.783 22961 3311.4
## + BTC_V  1     1.596 22962 3311.4
## + RMW    1     1.197 22962 3311.4
## + ETH    1     0.266 22963 3311.5
## - SMB    1    85.195 23048 3311.5
##
## Step:  AIC=3307.98
## BTC1 ~ SMB + ETH_V
##
##         Df Sum of Sq   RSS    AIC
## <none>               22974 3308.0
## - ETH_V  1    60.204 23034 3308.8
## + HML    1    19.733 22954 3309.1
## + VUG    1    10.606 22963 3309.5
## + QQQ    1     8.035 22966 3309.6
## + VTI    1     6.034 22968 3309.7
## + BTC    1     5.478 22968 3309.7
```

```
## + RM.Rf  1     5.073 22969 3309.7
## + BNDX   1     4.285 22970 3309.8
## - SMB    1    81.613 23055 3309.8
## + XLE    1     3.171 22971 3309.8
## + BND    1     2.509 22971 3309.9
## + VTV    1     2.201 22972 3309.9
## + BTC_V  1     1.766 22972 3309.9
## + ETH    1     0.223 22974 3310.0
## + RMW    1     0.170 22974 3310.0
## + VXUS   1     0.113 22974 3310.0
## + CMA    1     0.046 22974 3310.0
```

```
##
## Call:
## lm(formula = BTC1 ~ SMB + ETH_V, data = train1)
##
## Coefficients:
## (Intercept)          SMB        ETH_V
##   4.778e-01   -5.297e-01   -7.829e-11
```

```
#Ideal Values : BTC1 ~ SMB, ETH_V
```

(j) Fit a GAM on the training data with this reduced dataset, using splines of each feature with 5 degrees of freedom. What is the test set MSE for this approach? What are the relevant nonlinear variables?

```
library(mgcv)
gamvar=gam(BTC1~ s(SMB, k=5)+s(ETH_V, k=5), data=train1) #GAM model with DoF set to 5 for the variables
summary(gamvar)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## BTC1 ~ s(SMB, k = 5) + s(ETH_V, k = 5)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.3115     0.1405   2.216   0.0269 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df     F p-value
## s(SMB)     1      1 3.826  0.0507 .
## s(ETH_V)   1      1 2.822  0.0933 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.00404   Deviance explained = 0.589%
## GCV = 21.391  Scale est. = 21.331    n = 1080
```

```
yhat.gam=predict(gamvar, newdata=test1) # MSE Prediction
var8=mean((yhat.gam-rd.test)^2)
vec2=c(vec2, var8)
#Test Set MSE : 27.53583
```

(k) Build a table to compare the test set MSE of your best model for:

- Boosting
- Bagging
- Random Forests
- Support vector machine
- Support vector machine with nonlinear kernel
- GAM

```
as.data.frame(vec1) # Setting the two vectors as DFs to cbind them
```

```
##           vec1
## 1     Boosting
## 2      Bagging
## 3 randomForest
## 4          SVM
## 5        SVMNL
## 6          GAM
```

```
as.data.frame(vec2)
```

```
##       vec2
## 1 24.65525
## 2 55.79212
## 3 43.31796
## 4 26.36608
## 5 24.49359
## 6 27.53584
```

```
MSEtable=cbind(vec1,vec2) #Creating the MSE table through cbind
MSEtable
```

```
##      vec1           vec2
## [1,] "Boosting"     "24.6552536047637"
## [2,] "Bagging"      "55.7921190501351"
## [3,] "randomForest" "43.3179563926409"
## [4,] "SVM"          "26.3660802527747"
## [5,] "SVMNL"        "24.4935888971672"
## [6,] "GAM"          "27.5358363054102"
```

(l) Discuss and explain your results of the previous table. Why do you think that some algorithms performed better than others? What explains the result of the best algorithm?

```
# Support Vector Machines with Nonlinear Kernel performed the best for MSE. Boosting performed well ove
```