FE590. Assignment #2.

2021-10-23

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:Muhammet Furkan Isik

CWID:10472193

Date:10/21/2021

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.

```r
CWID = 10472193 #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.
```

Question 1

You are part of a research team of a FinTech company dedicated to forecast the price direction of public US companies. Using the attached dataset, select the company assigned to you according to your Stevens ID in the spreadsheet “assignments”.

You must build a forecasting model of your assigned company’s next day’s excess return (stock return – risk free rate; ExRet) and its direction using the years 2014-2018 to train and 2019 to test your model.

The dataset is based on the Fama French 3 factor model:
$R_{it}-R_{ft}=\alpha_{it}+\beta_1(RM_t-R_{ft})+\beta_2 SMB_t+\beta_3 HML_t+\epsilon_{it}$

where: Rit=total return of a stock or portfolio i at time t Rft=risk free rate of return at time t RMt=total market portfolio return at time t Rit−Rft=expected excess return RMt−Rft=excess return on the market portfolio (index) SMBt=size premium (small minus big) HMLt=value premium (high minus low) β1,2,3=factor coefficients

The dataset includes the following variables or columns: Ticker: identifier of the stock Permno: permanent number of the stock Date: date of observation Alpha: Alpha of stock i b_mkt: Beta on mktrft b_smb: Beta on SMBt b_hml: Beta on HMLt ivol: Idiosyncratic Volatility: Variance of residuals or the part that cannot be explained by the Fama French model tvol: Total Volatility: Variance of stock returns Exret: Excess Return from Risk Model: stock return (price_t/price_(t-1)-1) – risk free rate

For this exercise you do not need to know the details of the Fama French model. However, further details and data can be obtained at:
http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/f-f_factors.html
https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

1. Create a csv file with the information of the company assigned to you according to your Stevens ID in the tab “assignments” of the provided database. Read this csv file in R. Sort the variables by date in a new dataset. List the names of the variables in the dataset.

```
# Importing raw data

library(readxl)
row_data <- read_excel("stocks100_2014_19.xlsx")

# Subsetting only MDT Ticker
data= row_data[row_data$TICKER== "MDT", ]

# Ordering the dataset  by date
data= data[order(data$DATE), ]
data= data[,c(2,1,3,4,5,6,7,8,9,10)]



names(data)

##  [1] "DATE"   "PERMNO" "alpha"  "b_mkt"  "b_smb"  "b_hml"  "ivol"   "tvol"
##  [9] "exret"  "TICKER"

head(data)

## # A tibble: 6 x 10
##       DATE PERMNO  alpha b_mkt  b_smb   b_hml    ivol    tvol    exret
TICKER
##      <dbl>  <dbl>  <dbl> <dbl>  <dbl>   <dbl>   <dbl>   <dbl>    <dbl>
<chr>
## 1 20140102  60097 0.0004 0.928 -0.234 -0.0669 0.00775 0.00995  0.00500 MDT
## 2 20140103  60097 0.0004 0.928 -0.220 -0.0661 0.00778 0.00998  0.0198  MDT
## 3 20140106   60097 0.0005 0.924 -0.242 -0.0454 0.00785 0.0100   0.0171  MDT
## 4 20140107   60097 0.0005 0.929 -0.237 -0.0346 0.00781 0.0100   0.00496 MDT
## 5 20140108   60097 0.0005 0.929 -0.238 -0.0425 0.00787 0.0101   0.0163  MDT
## 6 20140109   60097 0.0004 0.928 -0.248  0.0061 0.00802 0.0102  -0.0242  MDT
```

2. Generate a new variable exret1 which is the excess return of the next day and exret_sq which is squared of exret. As the variables PERMNO, DATE and TICKER will be unimportant, remove those fields from your data frame.

```
# Removing PERMNO, DATE and TICKER

data= subset(data, select= - c(DATE, PERMNO, TICKER) )
head(data)

## # A tibble: 6 x 7
##    alpha b_mkt  b_smb   b_hml    ivol    tvol    exret
##    <dbl> <dbl>  <dbl>   <dbl>   <dbl>   <dbl>    <dbl>
## 1 0.0004 0.928 -0.234 -0.0669 0.00775 0.00995  0.00500
## 2 0.0004 0.928 -0.220 -0.0661 0.00778 0.00998  0.0198
## 3 0.0005 0.924 -0.242 -0.0454 0.00785 0.0100   0.0171
## 4 0.0005 0.929 -0.237 -0.0346 0.00781 0.0100   0.00496
## 5 0.0005 0.929 -0.238 -0.0425 0.00787 0.0101   0.0163
## 6 0.0004 0.928 -0.248  0.0061 0.00802 0.0102  -0.0242

# Creating exret1 variable
e1= data[2: nrow(data), "exret"]

# Creating exret_sq variable
e2= e1^2


# Merging e1 and e2
e_1_2= cbind(e1,e2)
colnames(e_1_2)= c("exret1", "exret_sq")


# Adding them into the dataframe
data= data[1:nrow(data)-1, ]


data= cbind(data, e_1_2)

head(data)

##   alpha  b_mkt   b_smb   b_hml     ivol     tvol     exret    exret1
## 1 4e-04 0.9282 -0.2337 -0.0669 0.007747 0.009945  0.005005  0.019830
## 2 4e-04 0.9276 -0.2201 -0.0661 0.007783 0.009977  0.019830  0.017146
## 3 5e-04 0.9239 -0.2423 -0.0454 0.007853 0.010012  0.017146  0.004955
## 4 5e-04 0.9287 -0.2368 -0.0346 0.007810 0.010007  0.004955  0.016325
## 5 5e-04 0.9289 -0.2375 -0.0425 0.007874 0.010052  0.016325 -0.024235
## 6 4e-04 0.9279 -0.2480  0.0061 0.008023 0.010178 -0.024235  0.008848
##       exret_sq
## 1 3.932289e-04
## 2 2.939853e-04
## 3 2.455203e-05
## 4 2.665056e-04
## 5 5.873352e-04
## 6 7.828710e-05
```

3. What is the range of each quantitative variable? Answer this question using the range() function with the sapply() function e.g., sapply(cars, range). Print a simple table of the ranges of the variables. The rows should correspond to the variables. The first column should be the lowest value of the corresponding variable, and the second column should be the maximum value of the variable. The columns should be suitably labeled.

```
# Table for range data using sapply
range_data= sapply(data,range)

# transpose of data
range_data= t(range_data)

# Changing colnames
colnames(range_data)= c("Min", "Max")
#range_data

# Converting it  into a dataframe
as.data.frame(range_data)

##                  Min         Max
## alpha    -9.0000e-04 0.000800000
## b_mkt     5.6470e-01 1.113300000
## b_smb    -4.5510e-01 0.035000000
## b_hml    -6.7520e-01 0.086900000
## ivol      7.6140e-03 0.010500000
## tvol      9.3570e-03 0.013803000
## exret    -8.5106e-02 0.056594000
## exret1   -8.5106e-02 0.056594000
## exret_sq  2.5000e-11 0.007243031
```

4. What is the mean and standard deviation of each variable? Create a simple table of the means and standard deviations.

```
# Mean and Sd for each variable
mean_data= sapply(data, mean)
```

```
sd_data= sapply(data,sd)
# Taking transpose
t(mean_data)

##            alpha     b_mkt      b_smb      b_hml        ivol       tvol
## [1,] 0.000111332 0.8506675 -0.1876306 -0.3166404 0.009031587 0.01162728
##             exret       exret1     exret_sq
## [1,] 0.0001735394 0.0001723764 8.471263e-05

t(sd_data)

##             alpha     b_mkt     b_smb     b_hml         ivol        tvol
## [1,] 0.0003497974 0.1378713 0.1056678 0.1819759 0.0006319122 0.001058756
##            exret      exret1     exret_sq
## [1,] 0.009205883 0.009205383 0.0003089153

# Creating a dataframe
table= cbind(mean_data,sd_data)
as.data.frame(table)

##              mean_data      sd_data
## alpha     1.113320e-04 0.0003497974
## b_mkt     8.506675e-01 0.1378712588
## b_smb    -1.876306e-01 0.1056677970
## b_hml    -3.166404e-01 0.1819759045
## ivol      9.031587e-03 0.0006319122
## tvol      1.162728e-02 0.0010587562
## exret     1.735394e-04 0.0092058829
## exret1    1.723764e-04 0.0092053830
## exret_sq  8.471263e-05 0.0003089153
```

5. Split your data into a 70% training set and a 30% testing set. Using the regsubsets function in the leaps library, regress next day excess return on the remaining variables only using the training sample.

```
# Splitting the dataset

# Creating training dataset 70%
train= 0.7* nrow(data)
train=round(train)


data_train= data[1: train,  ]
tail(data_train)

##        alpha  b_mkt   b_smb   b_hml     ivol     tvol     exret    exret1
## 1051 -6e-04 0.8325 -0.1219 -0.2766 0.008830 0.010394  0.006470  0.008209
## 1052 -6e-04 0.8318 -0.1072 -0.2843 0.008847 0.010409  0.008209  0.008676
## 1053 -6e-04 0.8373 -0.1151 -0.2751 0.008844 0.010423  0.008676 -0.002445
## 1054 -6e-04 0.8342 -0.1087 -0.2742 0.008837 0.010438 -0.002445  0.002830
## 1055 -6e-04 0.8336 -0.1056 -0.2750 0.008840 0.010437  0.002830  0.007661
## 1056 -6e-04 0.8277 -0.1021 -0.2746 0.008854 0.010437  0.007661 -0.010955
##          exret_sq
## 1051 6.738768e-05
## 1052 7.527298e-05
## 1053 5.978025e-06
## 1054 8.008900e-06
## 1055 5.869092e-05
## 1056 1.200120e-04
```

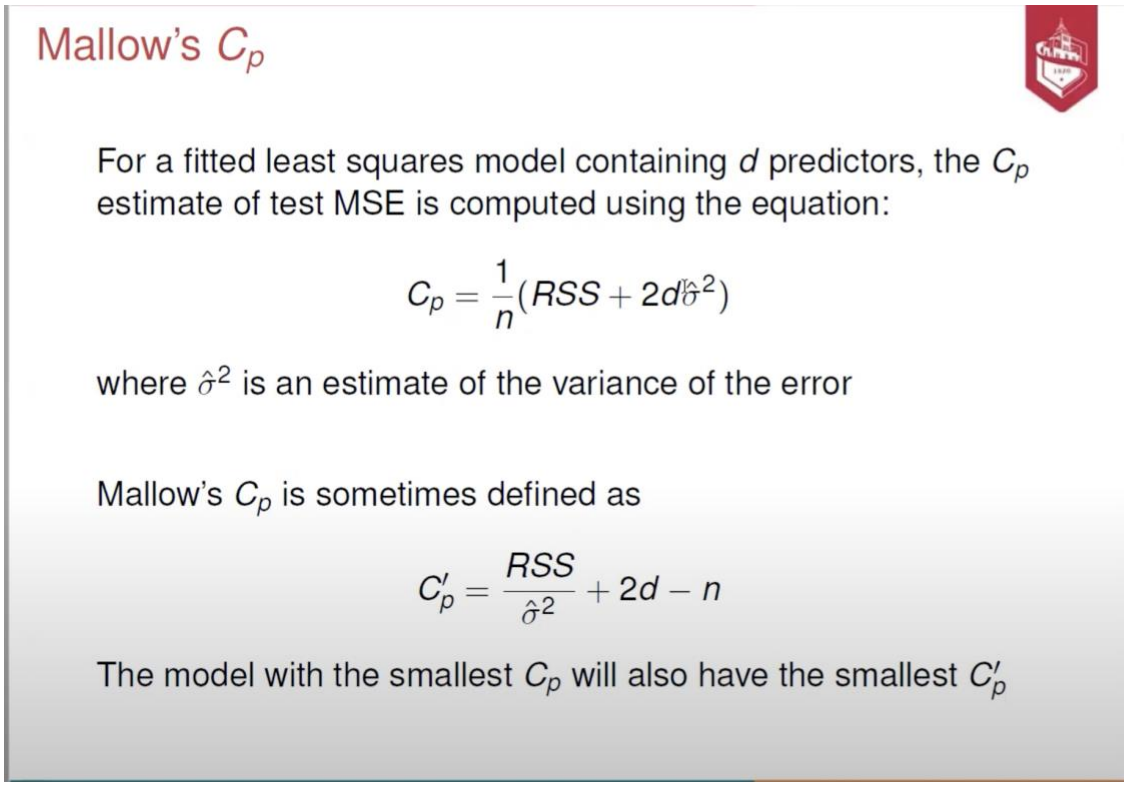
```
# Creating test dataset 30%
test= train+1
data_test= data[test: nrow(data),]

tail(data_test)
```

```
##        alpha  b_mkt   b_smb   b_hml     ivol     tvol     exret    exret1
## 1504 1e-04 0.6291 -0.4540 -0.3131 0.009103 0.011031 -0.003217  0.000634
## 1505 1e-04 0.6280 -0.4545 -0.3124 0.009102 0.010987  0.000634  0.000433
## 1506 2e-04 0.6043 -0.4185 -0.3261 0.009063 0.010791  0.000433 -0.010935
## 1507 2e-04 0.6032 -0.4074 -0.3288 0.009090 0.010608 -0.010935  0.005024
## 1508 2e-04 0.5978 -0.3991 -0.3294 0.009079 0.010568  0.005024  0.002287
## 1509 2e-04 0.5967 -0.3972 -0.3279 0.009080 0.010566  0.002287  0.003250
##          exret_sq
## 1504 4.019560e-07
## 1505 1.874890e-07
## 1506 1.195742e-04
## 1507 2.524058e-05
## 1508 5.230369e-06
## 1509 1.056250e-05
```

a. Print a table showing what variables would be selected using best subset selection for all predictors of the training set. Determine the optimal model using Mallows' Cp and output the model, including its coefficients.

```
# Best Subset Selection using training dataset
library(leaps)
ex.sub= regsubsets(exret~., data=data_train, nvmax=10)
t(summary(ex.sub)$which)
```

```
##                 1     2     3     4     5     6     7    8
## (Intercept)  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE TRUE
## alpha       FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE TRUE
## b_mkt       FALSE FALSE FALSE FALSE FALSE FALSE  TRUE TRUE
## b_smb       FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE TRUE
## b_hml       FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE
## ivol        FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE TRUE
## tvol        FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE TRUE
## exret1       TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE TRUE
## exret_sq    FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE TRUE
```



Mallow's C_p

For a fitted least squares model containing d predictors, the C_p estimate of test MSE is computed using the equation:

$$C_p = \frac{1}{n}(RSS + 2d\hat{\sigma}^2)$$

where $\hat{\sigma}^2$ is an estimate of the variance of the error

Mallow's C_p is sometimes defined as

$$C'_p = \frac{RSS}{\hat{\sigma}^2} + 2d - n$$

The model with the smallest C_p will also have the smallest C'_p

Mallow's Cp

- According to the Mallows' Cp Model 3 should be selected, which gives the minimum value

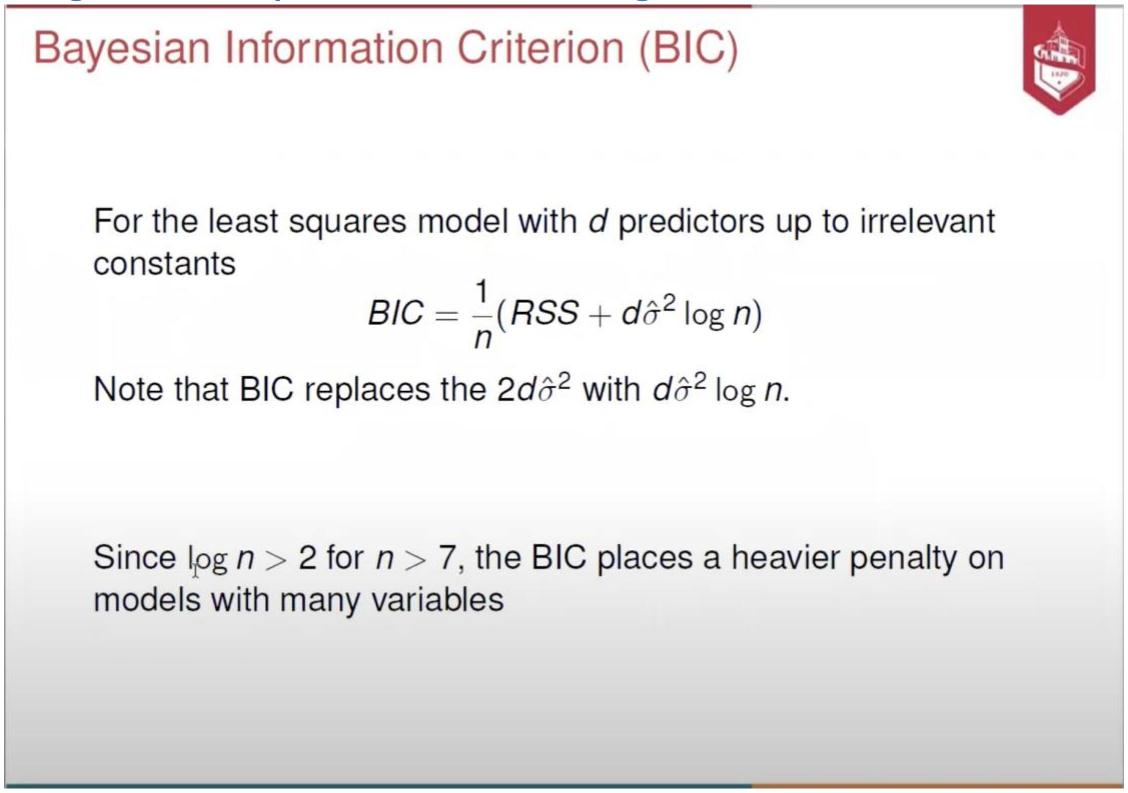
```
## According to the Mallows' Cp Model 3 should be selected
# Then our model is exret~ beta0 + b1*alpha + b2*exret1 + b3* exret_sq

# Using Mallows' Cp to determine the best fitted model
cp= summary(ex.sub)$cp
cp

## [1] 6.291117 1.274103 1.072380 2.899079 4.630677 6.199525 7.001561
9.000000

# Choosing the minimum Mallow's value
which.min(cp)

## [1] 3

#bic
#bic=summary(ex.sub)$bic
```

- Then our model is exret~ beta0 + b1* alpha + b2* exret1 + b3* exret_sq
- exret= -8.569053e-05 * beta0 + 2.072522e+00* alpha -9.578236e-02 * exret1 - 1.411347e+00 * exret_sq

```
# Then our model is exret~ beta0 + b1*alpha + b2*exret1 + b3* exret_sq
# exret= -8.569053e-05 * beta0 +  2.072522e+00* alpha -9.578236e-02 *exret1
```

```
-1.411347e+00 * exret_sq

cp_model= lm(exret~ alpha+exret1+ exret_sq, data=data_train)
# summary(cp_model)$coefficients
coef(cp_model)

##   (Intercept)         alpha        exret1      exret_sq
## -8.569053e-05  2.072522e+00 -9.578236e-02 -1.411347e+00
```

b. Print a table showing what variables would be selected using forward subset selection for all predictors of the training set. Determine the optimal model using BIC and output the model, including its coefficients.



Bayesian Information Criterion (BIC)

For the least squares model with d predictors up to irrelevant constants

$$BIC = \frac{1}{n}(RSS + d\hat{\sigma}^2 \log n)$$

Note that BIC replaces the $2d\hat{\sigma}^2$ with $d\hat{\sigma}^2 \log n$.

Since $\log n > 2$ for $n > 7$, the BIC places a heavier penalty on models with many variables

Bayesian Information Criteria

```
# Used regsubset function and specified forward method to implement forward
step wise implementation
for.sub= regsubsets(exret~., data=data_train, nvmax=10,method = "forward")
t(summary(for.sub)$which)

##                 1     2     3     4     5     6     7    8
## (Intercept)  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE TRUE
## alpha       FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE TRUE
## b_mkt       FALSE FALSE FALSE FALSE FALSE FALSE  TRUE TRUE
## b_smb       FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE TRUE
## b_hml       FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE
## ivol        FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE TRUE
## tvol         FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE TRUE
## exret1        TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE TRUE
## exret_sq     FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE TRUE
```

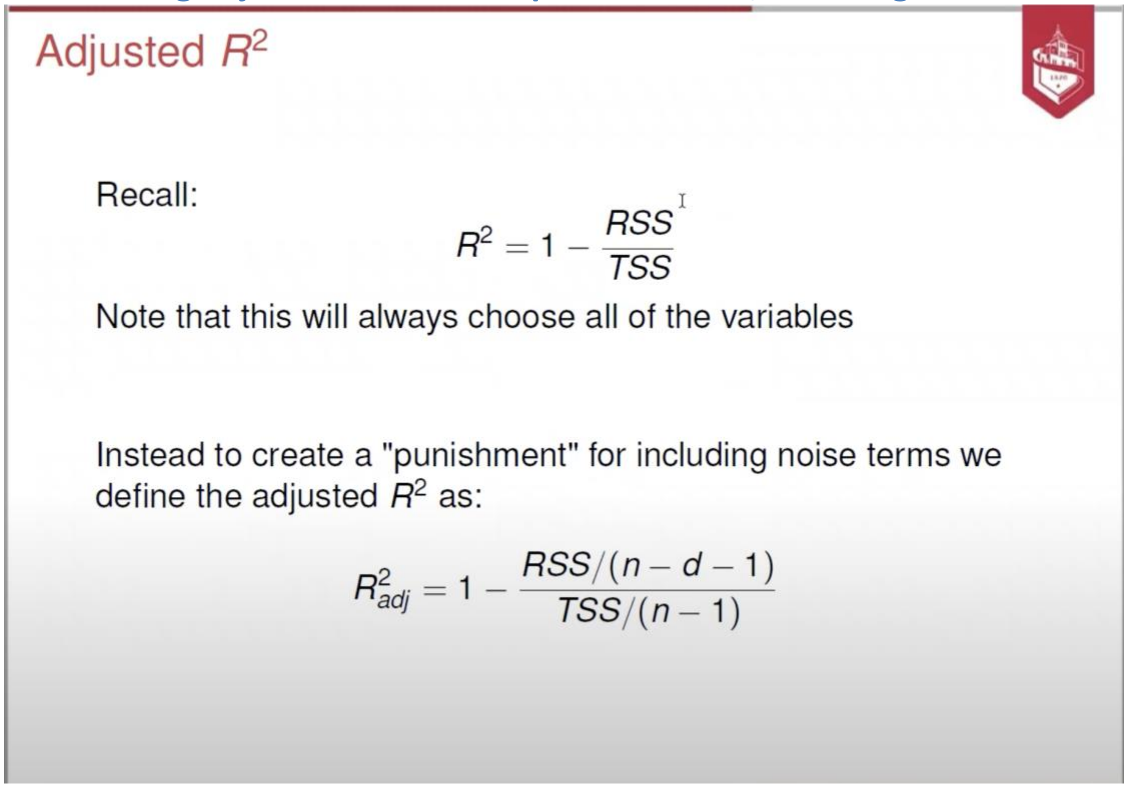
- Used BIC to come up with the model, according to BIC model with two predictors alpha1 and exret1 and constant term

```
# Used BIC to come up with the model, according to BIC model with two
predictors alpha1 and exret1 and constant term
bic= summary(for.sub)$bic
which.min(bic)

## [1] 2
```

- coefficients (Intercept) : -0.0002114784 alpha: 2.1030051398 exret1: -0.0868634902

```
# Model summary and coefficients   (Intercept) : -0.0002114784  alpha:
2.1030051398  exret1: -0.0868634902
bic_model= lm(exret~alpha+exret1,data=data_train)
summary(bic_model)

##
## Call:
## lm(formula = exret ~ alpha + exret1, data = data_train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.082243 -0.004553 -0.000072  0.004535  0.043282
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0002115  0.0003031  -0.698  0.48544
## alpha        2.1030051  0.7932461   2.651  0.00814 **
## exret1      -0.0868635  0.0305820  -2.840  0.00459 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.009207 on 1053 degrees of freedom
## Multiple R-squared:  0.01435,    Adjusted R-squared:  0.01248
## F-statistic: 7.666 on 2 and 1053 DF,  p-value: 0.0004951

coef(bic_model)

##   (Intercept)         alpha        exret1
## -0.0002114784  2.1030051398 -0.0868634902
```

c. Print a table showing what variables would be selected using backward subset selection for all predictors of the training set. Determine the optimal model using adjusted R^2 and output the model, including its coefficients.



Adjusted R^2

Recall:

$$R^2 = 1 - \frac{RSS}{TSS}$$

Note that this will always choose all of the variables

Instead to create a "punishment" for including noise terms we define the adjusted R^2 as:

$$R^2_{adj} = 1 - \frac{RSS/(n-d-1)}{TSS/(n-1)}$$

Adjusted R^2

```
# Used regsubset function and specified backward method to implement step
wise implementation
bac.sub= regsubsets(exret~., data= data_train, nvmax=10, method= "backward")
t(summary(bac.sub)$which)
```

```
##                 1     2     3     4     5     6     7    8
## (Intercept)  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE TRUE
## alpha       FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE TRUE
## b_mkt       FALSE FALSE FALSE FALSE FALSE FALSE  TRUE TRUE
## b_smb       FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE TRUE
## b_hml       FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE
## ivol        FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE TRUE
## tvol        FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE TRUE
## exret1       TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE TRUE
## exret_sq    FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE TRUE
```

- Determined the model by looking at the maximum r-squared value across different models which is model 8 using all variables

```
## Determining the model by looking at the maximum r-squared value
summary(bac.sub)$rsq
```

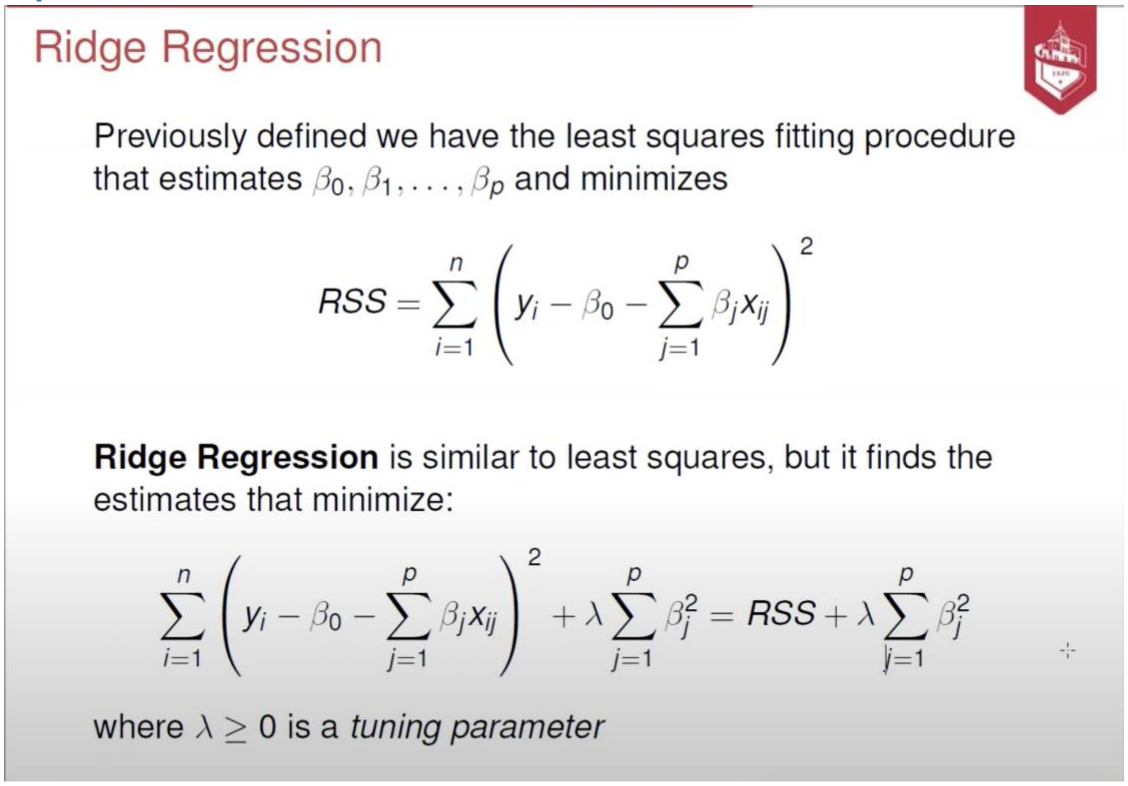
```
## [1] 0.007772254 0.014351234 0.016415515 0.016577997 0.016829645
0.017233883
## [7] 0.018357064 0.018358528

which.max(summary(bac.sub)$rsq)

## [1] 8
```

- Then our best model according to R-squared value is using all the variables and coefficient as follows

" (Intercept) alpha b_mkt b_smb b_hml ivol tvol exret1 -3.571634e-03 2.504058e+00 4.890125e-03 5.988364e-03 -9.127282e-05 8.152261e-01 -6.265090e-01 -9.563937e-02 exret_sq -1.297810e+00 "

```
# Then our best model according to R-squared value is using all the variables
and coefficient as follows

"
(Intercept)          alpha          b_mkt          b_smb          b_hml
ivol          tvol          exret1
-3.571634e-03  2.504058e+00  4.890125e-03  5.988364e-03 -9.127282e-05
8.152261e-01 -6.265090e-01 -9.563937e-02
     exret_sq
-1.297810e+00
"

## [1] "\n(Intercept)          alpha          b_mkt          b_smb          b_hml
ivol          tvol          exret1 \n-3.571634e-03  2.504058e+00  4.890125e-03
5.988364e-03 -9.127282e-05  8.152261e-01 -6.265090e-01 -9.563937e-02 \n
exret_sq \n-1.297810e+00 \n"

rsq.model= lm(exret~., data=data_train)
#summary(rsq.model)
coef(rsq.model)

##   (Intercept)         alpha         b_mkt         b_smb         b_hml
## -3.571634e-03  2.504058e+00  4.890125e-03  5.988364e-03 -9.127282e-05
##          ivol          tvol        exret1      exret_sq
##  8.152261e-01 -6.265090e-01 -9.563937e-02 -1.297810e+00
```

6.a. Using the training sample, fit a Ridge regression model with all the variables to forecast excess return. Create a graph with the diferent values of lambda and the coefficients. Using the cv.glmnet function in the glmnet library, fit a Ridge regression model with a 10-fold cross-validation to choose the tuning parameter lambda. Print the value of the coefficients. Using the best lambda and the test sample, predict next day excess return and calculate the mean squared error.



Ridge Regression

Previously defined we have the least squares fitting procedure that estimates $\beta_0, \beta_1, \ldots, \beta_p$ and minimizes

$$RSS = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2$$

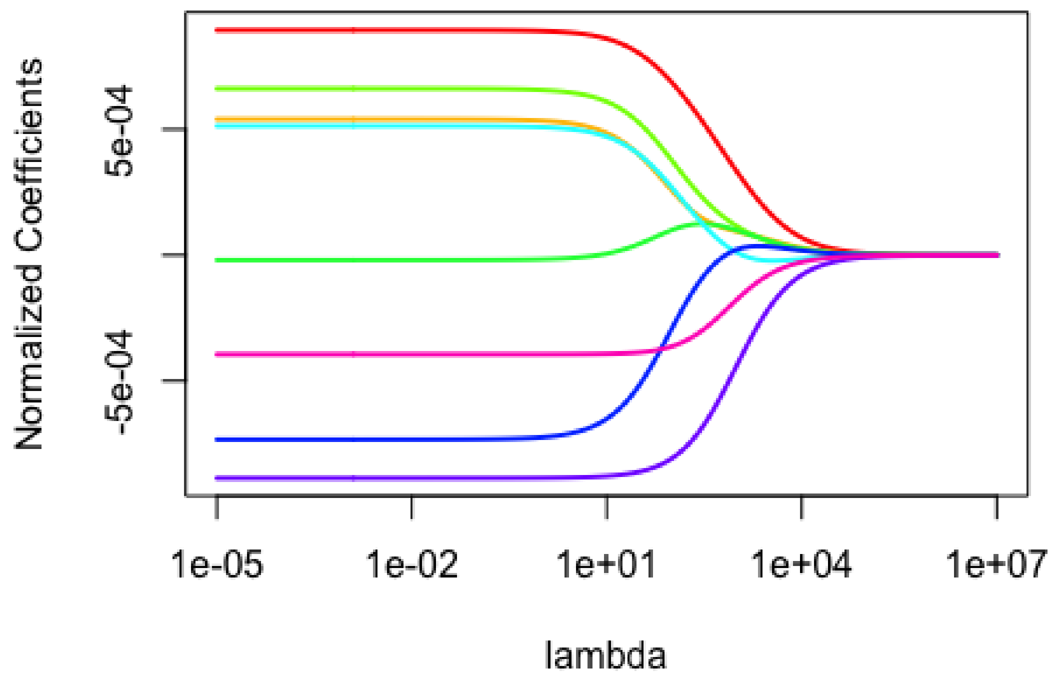
Ridge Regression is similar to least squares, but it finds the estimates that minimize:

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j^2$$

where $\lambda \geq 0$ is a *tuning parameter*

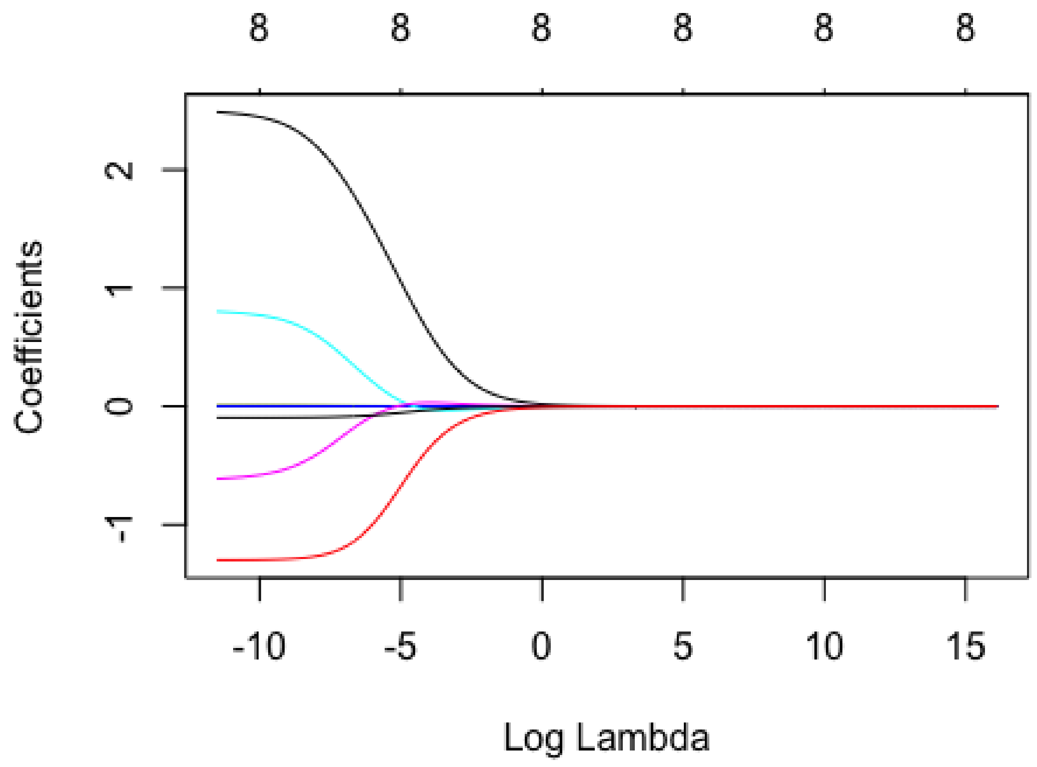
Ridge Regression

```r
# Fitting a ridge regression model using training dataset
library(MASS)
grid= 10^seq(7,-5, by= -.1)
a= lm.ridge(exret~., data=data_train, lambda = grid)


# Creating a graph with the diferent values of lambda and the coefficients
leg.col= rainbow(dim(a$coef)[1])
matplot(grid,t(a$coef), type= "l",lty=1,lwd=2, log = "x",col=leg.col,
xlab="lambda", ylab="Normalized Coefficients")
```

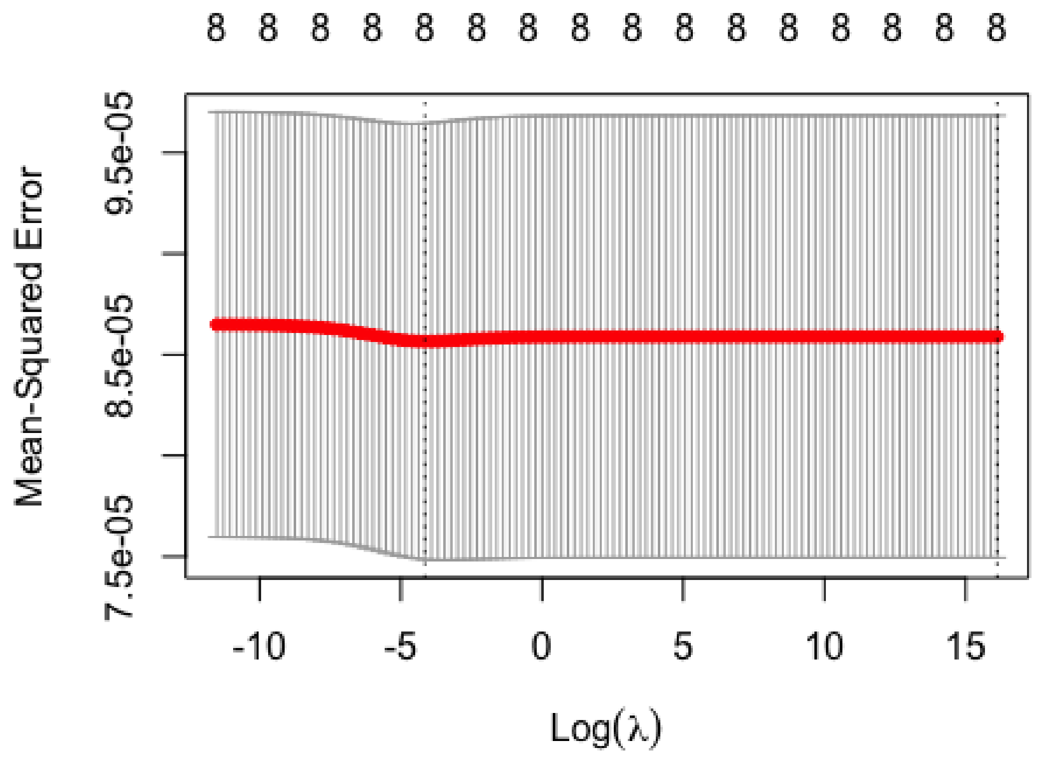


- Found the Tuning parameter which is -4.60517
- Calculated mean squared error which is 8.489139e-05

```
library(glmnet)
x= model.matrix(exret~., data=data_train)[,-1]
y= data_train$exret
ridge.mod= glmnet(x,y,alpha = 0,lambda=grid)


# 10-fold cross-validation to choose the tuning parameter lambda
cv.out= cv.glmnet(x,y,alpha=0,lambda=grid)
bestlam= cv.out$lambda.min

# Finding the Tuning parameter which is  -4.60517
log(bestlam)

## [1] -4.144653

plot(cv.out$glmnet.fit,"lambda")
```



```r
plot(cv.out)
```

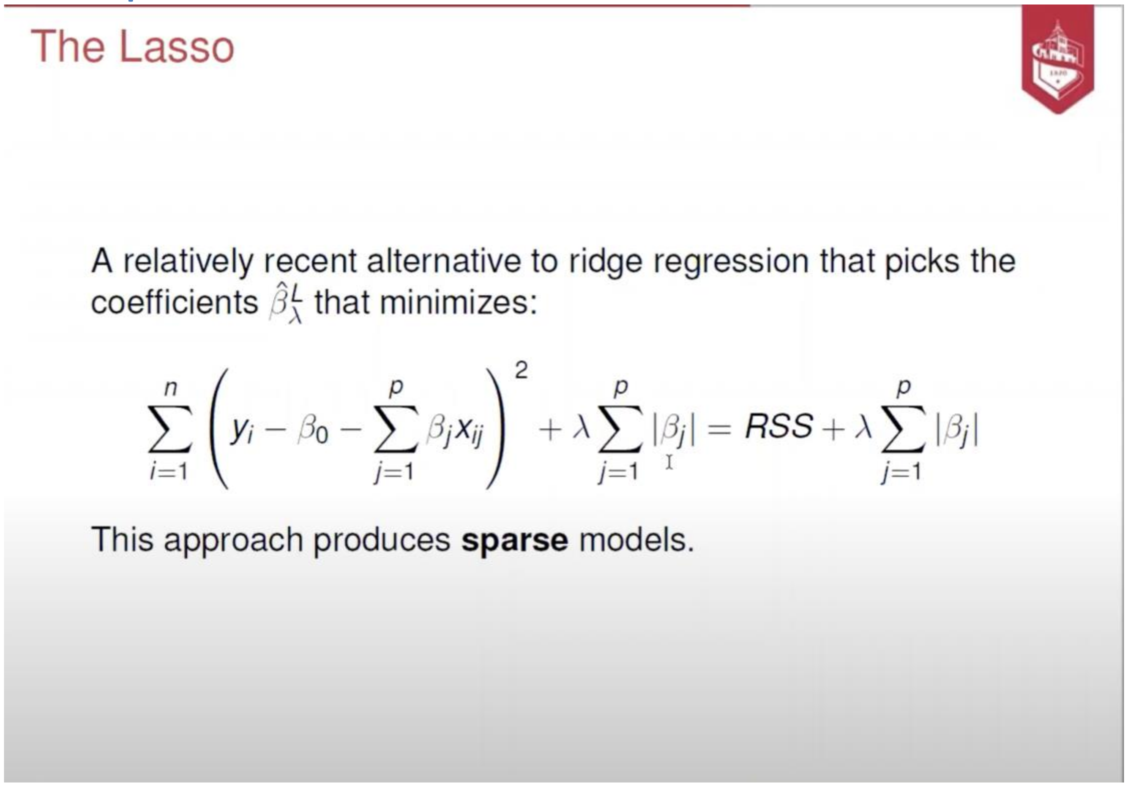


```
# Predicting values using ridge regression
bestrid.pred= predict(ridge.mod, s= bestlam,newx = x)


# Calculating mean squared error which is 8.489139e-05
mse_rid= mean((bestrid.pred- y)^2)
mse_rid

## [1] 8.489139e-05
```

- Predicted Coefficients

```
# Prediciting Coefficients
ridgecoef= glmnet(x,y,alpha=0,lambda = grid)
predict(ridgecoef,type="coefficients", s= bestlam)

## 9 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) -0.0005307564
## alpha        0.6784633576
## b_mkt        0.0007076546
## b_smb        0.0007150663
## b_hml        0.0003646288
## ivol        -0.0225943656
## tvol          0.0305012556
## exret1       -0.0335433538
## exret_sq     -0.3926163992
```

b. Fit a Lasso regression model with a 10-fold cross-validation to choose the tuning parameter lambda. Print the value of the coefficients. Using the best lambda and the test sample, predict next day excess return and calculate the mean squared error.



The Lasso

A relatively recent alternative to ridge regression that picks the coefficients $\hat{\beta}^L_\lambda$ that minimizes:

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = RSS + \lambda\sum_{j=1}^{p}|\beta_j|$$

This approach produces **sparse** models.

Lasso Regression

- Calculated tuning parameter which is -8.289306
- Calculated mse value which is 8.457083e-05
- Predicted Coefficients

```
# Fitting a lasso model
lasso.mod= glmnet(x,y,alpha=1,lambda=grid)

# 10-fold cross-validation to choose the tuning parameter lambda
cv.out=cv.glmnet(x,y,alpha=1, lambda=grid)

#plot(cv.out)
bestlam=cv.out$lambda.min
log(bestlam)

## [1] -8.289306
```

```
# predicting excess return
bestlas.pred= predict(lasso.mod, s=bestlam, newx=x)



# Prediciting Coefficients
lassocoef= glmnet(x,y,alpha=1,lambda = grid)
predict(lassocoef,type="coefficients", s= bestlam)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept) -8.263723e-05
## alpha        1.401691e+00
## b_mkt        .
## b_smb        .
## b_hml        .
## ivol         .
## tvol         .
## exret1      -6.274945e-02
## exret_sq    -4.088129e-01
```

```
# mse using lasso, ,mse value is 8.457083e-05
mean((bestlas.pred-y)^2)
```

```
## [1] 8.457083e-05
```

c. Compare and discuss the results of Lasso and Ridge regression indicating what approach you will choose and why.

- Both Lasso and Ridge Regression gives very small Mean Squared Error, respectively 8.457083e-05, 8.471404e-05.
- Lasso uses -8.059048 as tuning parameter, whereas ridge uses -4.60517 as tuning parameters
- It's wise to say that lasso is better in this case, since it shrinks the coeffiecients exactly to zero which gives the model a good interpretation and makes it more simple

```
# Both Lasso and Ridge Regression gives very small Mean Squared Error,
respectively 8.457083e-05, 8.471404e-05.
# Lasso uses  -8.059048 as tuning parameter, whereas  ridge uses -4.60517 as
tuning parameters

# it's wise to say that  lasso is better in this case,  since it shrinks to
coeffiecients exactly to zero which gives the model a good interpretation and
makes it more simple
```

Question 2

Create another field “Direction” in this data frame that looks to the direction of the excess return of the next period (exret1), this direction should be listed as a factor, not a number. After “Direction” is created, exret1 should not be included in the dataset.

1. Using the training set, run LDA to forecast “Direction” on the training set. Predict with the test sample. Calculate the confusion matrix and accuracy.

```r
# Adding Direction to the data, and assigning value 1 if exret1 >= 0 or 0 if
exret<= 0

data$Direction= 0

for (i in seq(1: nrow(data) ) ) {

if (data$exret1[i]>=0){


  data$Direction[i]=1


} else {


  data$Direction[i]=0

}

}


# Setting direction as  factor
data$Direction= as.factor(data$Direction)

table(data$Direction)

##
##   0   1
## 734 775

# Removing exret1 after Direction variable is created
data= subset(data, select= -exret1)
```

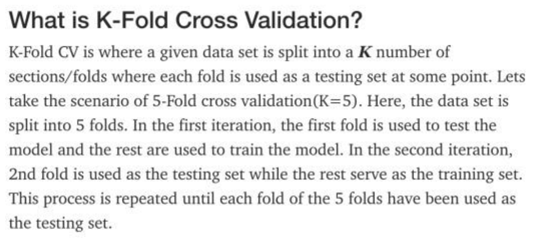
```
#head(data)


# Updating test and training data as well
data_train= data[1: train,  ]
data_test= data[(train+1): nrow(data),  ]

head(data_test)

##        alpha  b_mkt   b_smb   b_hml     ivol     tvol     exret
exret_sq
## 1057 -6e-04 0.8333 -0.1090 -0.2647 0.008877 0.010471 -0.010955 2.152089e-
06
## 1058 -6e-04 0.8318 -0.1118 -0.2631 0.008876 0.010457 -0.001467 1.032662e-
04
## 1059 -6e-04 0.8279 -0.1073 -0.2591 0.008843 0.010413 -0.010162 2.033476e-
06
## 1060 -6e-04 0.8303 -0.1156 -0.2565 0.008835 0.010448 -0.001426 2.724840e-
05
## 1061 -6e-04 0.8275 -0.1149 -0.2650 0.008821 0.010431 -0.005220 1.823290e-
05
## 1062 -6e-04 0.8341 -0.0948 -0.2560 0.008818 0.010429  0.004270 5.712100e-
06
##       Direction
## 1057         0
## 1058         0
## 1059         0
## 1060         0
## 1061         1
## 1062         1
```

- Created the confusion matrix
- Calculated accuracy which is 0.5055188

```
# Fitting LDA for Direction on training dataset
lda.fit= lda(Direction~.,data=data_train )
#lda.fit


# Predicting the direction using train data
lda.pred_train= predict(lda.fit, data_train)
lda.class_train= lda.pred_train$class

# Confusing matrix using train Data
#table(lda.class_train, data_train$Direction)
# Accuracy using training data
train_accur= mean(lda.class_train== data_train$Direction)
```

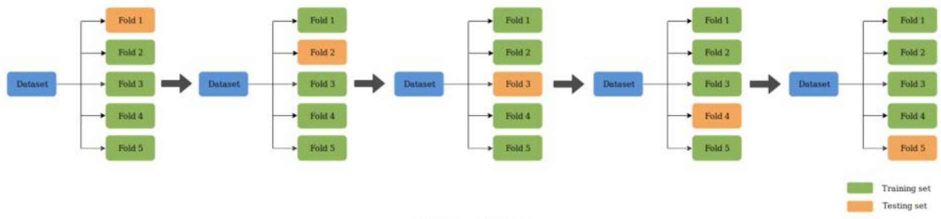
```
# Predicting the direction with test sample
lda.pred= predict(lda.fit,data_test)
lda.class= lda.pred$class

# Creating the Confusion Matrix
table(lda.class,data_test$Direction)

##
## lda.class   0   1
##         0 133 148
##         1  76  96

# Calculating accuracy
test_accur= mean(lda.class== data_test$Direction)

#(133+ 96) / (96+133+148 +76)


## Model has higher accuracy using training data in comparison to test data(
0.5055188 )

Accurcy_table= list("Train Accuracy "= train_accur, "Test
Accuracy"=test_accur )

data.frame(Accurcy_table)

##   Train.Accuracy. Test.Accuracy
## 1       0.5397727     0.5055188
```

2. Create code to determine the estimate of the expected test error of your model to forecast “Direction” using K=5 cross validation. Do this by actually splitting the complete dataset into five pieces (can use function cut) and give the average of the test error, not just by using a command from a package.



What is K-Fold Cross Validation?

K-Fold CV is where a given data set is split into a *K* number of sections/folds where each fold is used as a testing set at some point. Lets take the scenario of 5-Fold cross validation(K=5). Here, the data set is split into 5 folds. In the first iteration, the first fold is used to test the model and the rest are used to train the model. In the second iteration, 2nd fold is used as the testing set while the rest serve as the training set. This process is repeated until each fold of the 5 folds have been used as the testing set.



5-Fold Cross Validation

K Fold Cross Validation

- Data split into 5 equal pieces

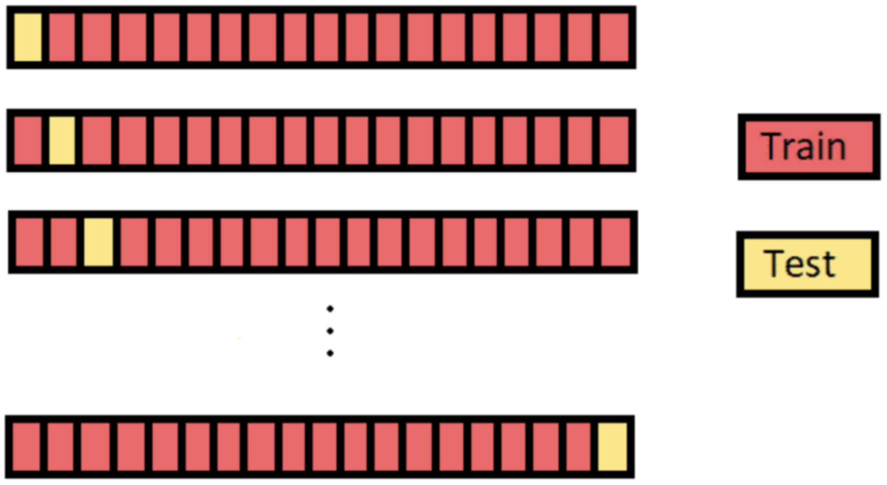
```
# K Fold Cross Validation, used boot library


# Splitting data into 5 equal pieces
data_1= data[(1: (nrow(data)/5)), ]
#tail(data_1)
data_2= data[(nrow(data_1)+1):  (2*(nrow(data)/5)) , ]
#tail(data_2)
data_3= data[(2*(nrow(data)/5)+1):  (3*(nrow(data)/5)) , ]
#tail(data_3)

data_4= data[(3*(nrow(data)/5)):  (4*(nrow(data)/5)) , ]
#tail(data_4)

data_5= data[(4*(nrow(data)/5)):nrow(data) , ]
tail(data_5)

##      alpha  b_mkt   b_smb   b_hml     ivol     tvol     exret     exret_sq
## 1503 1e-04 0.6321 -0.4551 -0.3169 0.009103 0.011077 -0.006939 1.034909e-05
## 1504 1e-04 0.6291 -0.4540 -0.3131 0.009103 0.011031 -0.003217 4.019560e-07
## 1505 1e-04 0.6280 -0.4545 -0.3124 0.009102 0.010987  0.000634 1.874890e-07
## 1506 2e-04 0.6043 -0.4185 -0.3261 0.009063 0.010791  0.000433 1.195742e-04
## 1507 2e-04 0.6032 -0.4074 -0.3288 0.009090 0.010608 -0.010935 2.524058e-05
## 1508 2e-04 0.5978 -0.3991 -0.3294 0.009079 0.010568  0.005024 5.230369e-06
##      Direction
## 1503         0
## 1504         1
## 1505         1
## 1506         0
## 1507         1
## 1508         1
```

- The average of the test error is 0.505309 using using k fold cross validation where k=5

```
# data_1 is the training set

lda.data_1= lda(Direction~.,  data= rbind(data_2,data_3,data_4,data_5) )
data_1.pred= predict(lda.data_1, newdata = data_1)
err_1= mean(data_1.pred$class!=data_1$Direction)

# data_2 is the test set

lda.data_2= lda(Direction~.,  data= rbind(data_1,data_3,data_4,data_5) )
data_2.pred= predict(lda.data_2, newdata = data_2)
err_2= mean(data_2.pred$class!=data_2$Direction)



# data_3 is the test set

lda.data_3= lda(Direction~.,  data= rbind(data_1,data_2,data_4,data_5) )
data_3.pred= predict(lda.data_3, newdata = data_3)
err_3= mean(data_3.pred$class!=data_3$Direction)


# data_4 is the test set

lda.data_4= lda(Direction~.,  data= rbind(data_1,data_2,data_3,data_5) )
data_4.pred= predict(lda.data_4, newdata = data_4)
err_4= mean(data_4.pred$class!=data_4$Direction)


# data_5 is the test set

lda.data_5= lda(Direction~.,  data= rbind(data_1,data_2,data_3,data_4) )
data_5.pred= predict(lda.data_5, newdata = data_5)
err_5= mean(data_5.pred$class!=data_5$Direction)



err_list= cbind(err_1,err_2,err_3,err_4,err_5)
```

```r
data.frame(err_list)

##       err_1     err_2     err_3 err_4     err_5
## 1 0.5149502 0.4768212 0.5016611   0.5 0.5331126

# The average of the test error 0.505309
mean(err_list)

## [1] 0.505309
```

3. Determine the LOOCV estimate of the expected test error of your model to forecast “Direction” using the complete dataset. How do your answers to each part of this question compare? Do you see any noticable differences between your answers? Why do you think that is?



Leave One Out Cross validation. This is another cross validation

K Fold Cross Validation

Mean error is by using LOOCV is 0.4910537 whereas mean error using K-fold is 0.505309 they are so close to each other

Obviously LOOCV takes more time than K-Fold to come up with MSE, since the number of folds equals to the number of observation -1

LOOCV can be subject to high variance or overfitting, as we are feeding the model almost all the training data to learn and just a single observation to evaluate.

K-Fold reduces the variance shown by LOOCV and introduces some bias by holding out a substantially large validation set.

```
err_cont=c()
#data
for (i in 1:nrow(data)) {
  train <- data[-i,]
  test <- data[i,]

  lda.fit= lda(Direction~.,  data= train)


  lda.pred= predict(lda.fit, test)

  lda.class=lda.pred$class


  err= mean( lda.class!= test$Direction )

  err_cont= c(err_cont,err)


}

mean(err_cont)

## [1] 0.4910537

# Mean error is by using LOOCV is 0.4910537 whereas mean error using K-fold
is 0.505309 they are so close to each other

# Obviously LOOCV takes more time than K-Fold to come up with MSE
# LOOCV can be subject to high variance or overfitting

# K-Fold reduces the variance shown by LOOCV and introduces some bias by
holding out a substantially large validation set.
```

Question 3

This question should be answered using the Weekly data set, which is part of the ISLR package. This data contains 1,089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

1. What does the data represent?

- Data represents weekly return with lag values till 5 and Volume, and Direction
- Min, Median and Quartiles values of the data represented in below

```
# Data represents weekly return with lag values till 5 and Volume, and
Direction
# Min, Median and Quartiles values of the data represented in below
library(ISLR)
head(Weekly)

##   Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today Direction
## 1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
## 2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
## 3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514        Up
## 4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712        Up
## 5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178        Up
## 6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372      Down

summary(Weekly)

##       Year           Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
##       Lag4               Lag5              Volume            Today
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540
##  Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260
##  Direction
##  Down:484
##  Up  :605
```

```
##
##
##
##
```

2. Use the full data set to perform a logistic regression with Direction as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

- According to summary results, Yes some of the variables statistically significant.
- Intercept and Lag value 2 values are significant, it could be also said that Lag 1(p-value 0.1181) is somewhat significant

```
# According to summary results, Yes some of the variables statistically
significant.
# Intercept and Lag value 2 values are significant, it could be also said
that Lag 1 is somewhat significant

log.fit= glm(Direction~ Lag1+ Lag2 +Lag3 +Lag4 +Lag5+Volume, data=Weekly,
family = binomial)
summary(log.fit)

##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
```

```
##
## Number of Fisher Scoring iterations: 4

coef(log.fit)

## (Intercept)        Lag1        Lag2        Lag3        Lag4        Lag5
##  0.26686414 -0.04126894  0.05844168 -0.01606114 -0.02779021 -0.01447206
##      Volume
## -0.02274153

# Predicting prob with given logistic regression model. type = "response"
option tells R to output probabilities of the form P(Y = 1|X

log.probs=predict(log.fit,type="response")

log.probs[1:10]

##         1         2         3         4         5         6         7
8
## 0.6086249 0.6010314 0.5875699 0.4816416 0.6169013 0.5684190 0.5786097
0.5151972
##         9        10
## 0.5715200 0.5554287
```

3. Fit a logistic regression model using a training data period from 1990 to 2008, using the predictors from the previous problem that you determined were statistically significant. Test your model on the held out data (that is, the data from 2009 and 2010) and express its accuracy.

- Accuracy is calculated which is 0.5769231

```
# Training data creation
Weekly_train= Weekly[Weekly["Year"] < 2009,]
#tail(Weekly_train)

# Test data creation
Weekly_test= Weekly[Weekly["Year"] >= 2009,]
#head(Weekly_test)
# Fitting a logistic regression with predictors found significant in the
model above, and used only  tranining data
log.fit_train= glm(Direction~Lag2+Lag1, data=Weekly_train, family=binomial)

summary(log.fit_train)

##
## Call:
## glm(formula = Direction ~ Lag2 + Lag1, family = binomial, data =
Weekly_train)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6149  -1.2565   0.9989   1.0875   1.5330
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.21109    0.06456   3.269  0.00108 **
## Lag2         0.05384    0.02905   1.854  0.06379 .
## Lag1        -0.05421    0.02886  -1.878  0.06034 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
## Residual deviance: 1347.0  on 982  degrees of freedom
## AIC: 1353
##
## Number of Fisher Scoring iterations: 4

# Predicting values using training data

weekly_prob_train= predict(log.fit_train,type="response")

glm_pred_train= rep(0,985)
glm_pred_train[weekly_prob_train>0.5]=1

# Assigning 0 and 1 for the Direction Down and Up
levels(Weekly_train$Direction)= c(0,1)

# Creating Confusion Matrix for the training data
#table(glm_pred_train,Weekly_train$Direction)

# Calculating the accuracy
#mean(glm_pred_train== Weekly_train$Direction)



# Testing the model on testing data set
weekly_prob= predict(log.fit_train,newdata = Weekly_test, type= "response")
glm_pred= rep(0,104)
glm_pred[weekly_prob>0.5]=1



# Confusion Matrix
table(glm_pred,Weekly_test$Direction )

##
## glm_pred Down Up
##         0    7  8
##         1   36 53

# Changing the level for Direction, 0 for Down, 1 for Up
levels(Weekly_test$Direction)= c(0,1)

# Confusion Matrix modified with only 0 and 1s
table(glm_pred,Weekly_test$Direction )

##
## glm_pred  0  1
##         0  7  8
##         1 36 53

# Calculating the Accuracy
## Accuracy is 0.5769231
mean(glm_pred== Weekly_test$Direction)

## [1] 0.5769231
```

4. Repeat Part 3 using LDA.

- Calculated Accuracy which is 0.5769231

```
library(MASS)

# Fitting a Model Using Linear Discriminant Analysis
lda.fit= lda(Direction~Lag2+Lag1, data= Weekly_train)
lda.predict= predict(lda.fit, Weekly_test)

lda.class= lda.predict$class



# Creating a confusin matrix
table(lda.class,Weekly_test$Direction )

##
## lda.class  0  1
##          0  7  8
##          1 36 53

# Calculating the Accuracy
## Accuracy is 0.5769231
mean(lda.class== Weekly_test$Direction)

## [1] 0.5769231
```

5. Repeat Part 3 using QDA.

- Calculated Accuracy which is 0.5576923

```
# Fitting a model using Quadratic Discriminant Analysis
qda.fit= qda(Direction~Lag2+Lag1, data= Weekly_train)
```

```
# Prediction Direction with QDA
qda.predict= predict(qda.fit,Weekly_test)
qda.class= qda.predict$class

# Creating a confusin matrix
table(qda.class, Weekly_test$Direction)

##
## qda.class  0  1
##          0  7 10
##          1 36 51

# Calculating the Accuracy
## Accuracy is 0.5576923
mean(qda.class== Weekly_test$Direction)

## [1] 0.5576923
```

6. Repeat Part 3 using KNN with K = 1, 2, 3.

- K=1,Calculated the Accuracy which is 0.4807692
- K=2,Calculated the Accuracy which is 0.4615385
- K=3, Calculated the Accuracy which is 0.5192308

```
library(class)

## Fitting a model using K- Nearest Neighbor using k=1

attach(Weekly_train)
train.X= cbind(Lag1,Lag2)
train.Direction= Weekly_train$Direction
attach(Weekly_test)
test.X= cbind(Lag1,Lag2)



knn.pred= knn(train=train.X,test=test.X, cl=train.Direction, k=1)

# Calculating the Accuracy
# Accuracy is 0.4807692
mean(knn.pred==Weekly_test$Direction)

## [1] 0.4807692

## Fitting a model using K- Nearest Neighbor using k=2
# Calculating the Accuracy
# Accuracy is 0.4615385
```

```
knn.pred= knn(train=train.X,test=test.X, cl=train.Direction, k=2)
mean(knn.pred==Weekly_test$Direction)

## [1] 0.5480769

## Fitting a model using K- Nearest Neighbor using k=3
knn.pred= knn(train=train.X,test=test.X, cl=train.Direction, k=3)
# Calculating the Accuracy
# Accuracy is 0.5192308
mean(knn.pred==Weekly_test$Direction)

## [1] 0.5192308
```

7. Which of these methods in Parts 3, 4, 5, and 6 appears to provide the best results on this data?

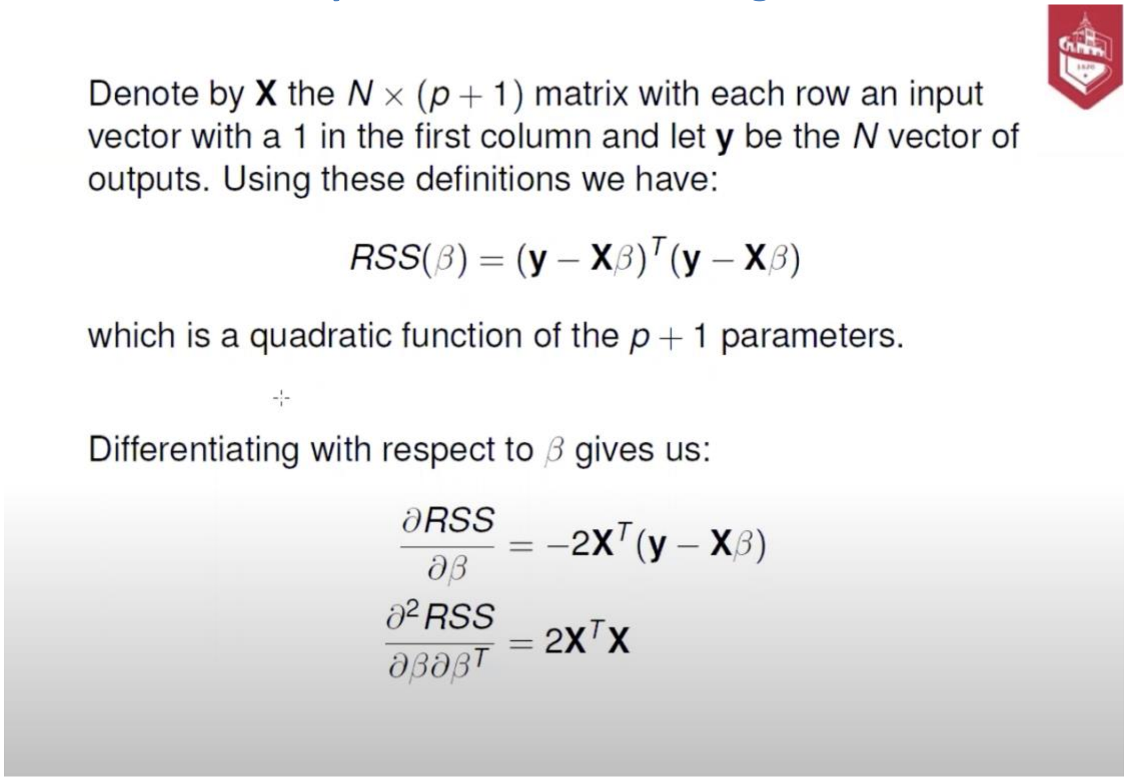
- Logistic Regression Model gives Accuracy= 0.5769231
- Linear Discriminant Analysis also gives Accuracy= 0.5769231
- Quadratic Discriminant Analysis gives Accuracy= 0.5576923
- KNN with K=1 gives Accuracy=0.4807692
- KNN with K=2 gives Accuracy=0.4615385
- KNN with K=3 gives Accuracy=0.5192308

- Among all of these different techniques, Logistic Regression and Linear Discriminant Analysis gave the highest Accuracy

```
# Logistic Regression Model gives Accuracy=  0.5769231
# Linear Discriminant Analysis also gives Accuracy= 0.5769231
# Quadratic Discriminant Analysis gives Accuracy= 0.5576923
# KNN with K=1 gives Accuracy=0.4807692
# KNN with K=2 gives Accuracy=0.4615385
# KNN with K=3 gives Accuracy=0.5192308

## Among all of these different techniques, Logistic Regression and Linear
Discriminant Analysis gave the highest Accuracy
```

Question 4

Write a function that works in R to gives you the parameters from a linear regression on a data set of n predictors. You can assume all the predictors and the prediction is numeric. Include in the output the standard error of your variables. You cannot use the lm command in this function or any of the other built in regression models.



Denote by $\mathbf{X}$ the $N \times (p+1)$ matrix with each row an input vector with a 1 in the first column and let $\mathbf{y}$ be the N vector of outputs. Using these definitions we have:

$$RSS(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)$$

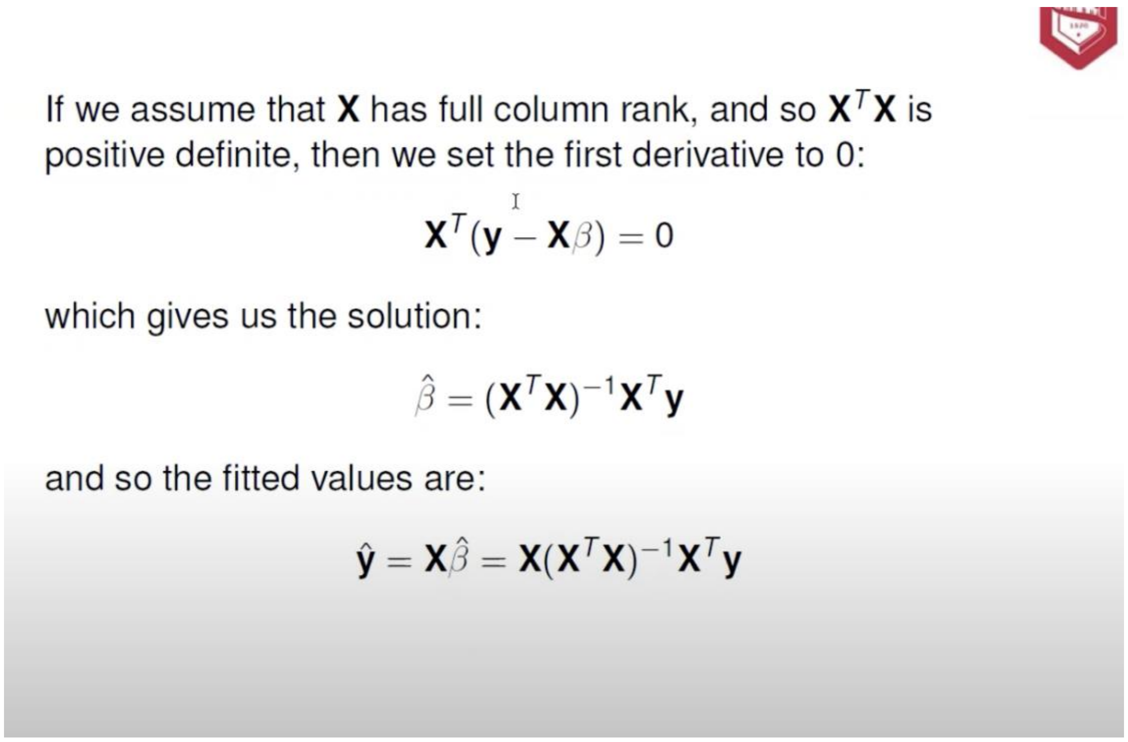
which is a quadratic function of the $p+1$ parameters.

Differentiating with respect to β gives us:

$$\frac{\partial RSS}{\partial \beta} = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta)$$

$$\frac{\partial^2 RSS}{\partial \beta \partial \beta^T} = 2\mathbf{X}^T\mathbf{X}$$

Least Square Method-1



If we assume that $\mathbf{X}$ has full column rank, and so $\mathbf{X}^T\mathbf{X}$ is positive definite, then we set the first derivative to 0:

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) = 0$$

which gives us the solution:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

and so the fitted values are:

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Least Square Method-2

$$X'X = \begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix}$$

Least Square Method-3

$$b = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{p-1} \end{bmatrix} = (X'X)^{-1}X'Y$$

Least Square Method-4

- Created a linear regression function, note column number should be specified to select the dependent variable

```r
# Creating a linear regression function, note column number should be
specified to select the  dependent variable

regress= function(dataset,col_num){

  Y= as.matrix(dataset[col_num])
  X= as.matrix (dataset[-col_num])

  # Adding 1s to the Matrix
  X = cbind(rep(1,nrow(dataset)), X  )
  a= t(X) %*% X
  b=  t(X) %*% Y
  a_inv= solve(a)
  beta= a_inv %*% b


  colnames(beta)= "coefficients"

  N=nrow(X)
  p= ncol(X)
  df= N-p-1

 # Calculating Standard Error of the coefficients
  sig_sq=   1/(N-p)      *      sum((Y- X%*%beta ) ^2 )
  SE= diag(sig_sq *a_inv)
  SE= sqrt(SE)

  # Calcualting t-value

  result= cbind(beta, SE)

  result= data.frame(result)
  result$t_value= result$coefficients/result$SE

  # Calculating p- value

  #2*pt(q=12.5785546,df= 5,lower.tail = F)

  #sapply(result$t_value, pt)

  return(result)


}
```

- Testing the function on a dataset
- Both lm function and own built function gives the same result

```r
library(readr)
# Testing the regression function using data set from the URL below

data_set <-
read_delim("https://online.stat.psu.edu/stat462/sites/onlinecourses.science.p
su.edu.stat462/files/data/soapsuds/index.txt",    "\t", escape_double =
FALSE, trim_ws = TRUE)
```

```r
# Adding another variable to the dataset to make it multivariable data set
data_set["soap_sq"] = data_set$soap^2
#head(data_set)


# Using lm function to create a regression model
lm(suds~soap+soap_sq,data=data_set)
```

```
##
## Call:
## lm(formula = suds ~ soap + soap_sq, data = data_set)
##
## Coefficients:
## (Intercept)         soap      soap_sq
##     -34.714       21.548       -1.095
```

```r
# Using regress own built function to create a regression model, both gives
the same result
regress(data_set,col_num=2)
```

```
##         coefficients         SE   t_value
##           -34.714286 23.9175170 -1.451417
## soap       21.547619  8.9013553  2.420712
## soap_sq    -1.095238  0.8067178 -1.357647
```