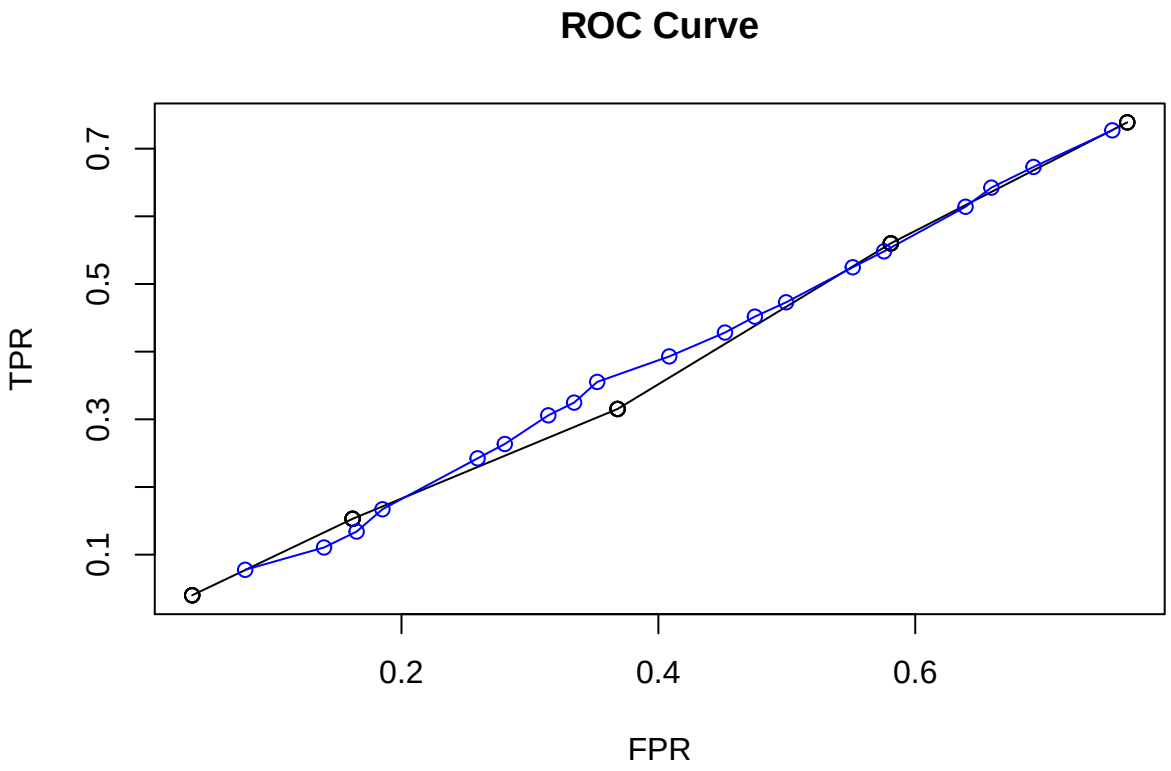
Group8_lab1

Group8

11/16/2017

Assignment 1 Spam Classification with Nearest Neighbours



APPENDIX

```
library(ggplot2)
library(kknn)
library(readxl)

data <- read_excel("spambase.xlsx")
data <- as.data.frame(data) # coonvert into data frame

n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,] #train is data
test=data[-id,] #test data is newData

knearest=function(data,k,newdata)
{
  n1=dim(data)[1]
```

```
  n2=dim(newdata)[1]
  p=dim(data)[2]
  Prob=numeric(n2)
  X = as.matrix(data[,-p])
  Y = as.matrix(newdata[-p]) # change xn to Yn

  X_hat = X/matrix(sqrt(rowSums(X^2)), nrow=n1, ncol=p-1)
  Y_hat = Y/matrix(sqrt(rowSums(Y^2)), nrow = n2 , ncol = p - 1)
  C <- X_hat %*% t(Y_hat)
  D <- 1 - C #distacne matrix calculate

  for (i in 1:n2 )
  {
    Ni <- as.data.frame(cbind(value=D[,i],spam=data[,p]))
    Ni <- Ni[order(Ni$value),]
    N_i <- Ni[1:k,] # get k values
    Prob[i] <- sum(N_i[,"spam"]) / k
  }
  return(Prob) #return proabilities
}


# Assignment 1 part 3
probalities <- knearest(train,5, test)

probalities <- ifelse(probalities > 0.5, 1,0)
conf_mat <- table(spam = train[,ncol(data)] , predicted_val = probalities)

miss_classfication_step3 <- (sum(conf_mat[ upper.tri(conf_mat) ]) +
                        sum(conf_mat[ lower.tri(conf_mat) ])) / sum(conf_mat) #missclassfication

# Assignnment 1 part 4
probalities <- knearest(train,1, test) #repeat step 3 for K = 1 which is

probalities <- ifelse(probalities > 0.5, 1,0)
conf_mat <- table(spam = train[,ncol(data)] , predicted_val = probalities)

miss_classfication_step4 <- (sum(conf_mat[ upper.tri(conf_mat) ]) +
                        sum(conf_mat[ lower.tri(conf_mat) ])) / sum(conf_mat) #missclassfication

# Assignment 1 part 5
knn <- kknn(Spam ~. , train, test , k = 5 ) #standard kknn method at  K = 5
probalities <- knn$fitted.values
probalities <- ifelse(probalities > 0.5, 1,0)

conf_mat <- table(spam = train[,ncol(data)] , predicted_val = probalities)

miss_classfication_step5 <- (sum(conf_mat[ upper.tri(conf_mat) ]) +
                        sum(conf_mat[ lower.tri(conf_mat) ])) / sum(conf_mat) #missclassfication
```

```
miss_classfication_step3
```

```
## [1] 0.4664234
```

```
miss_classfication_step5
```

```
## [1] 0.470073
```

The mis-classification rate obtained from calculated kkn is 0.3883212 and that by using kknn package is 0.3459854 which is almost similar

```
knn <- kknn(Spam ~. , train, test , k = 1 )  #standard kknn method at  K = 5
probalities <- knn$fitted.values
probalities <- ifelse(probalities > 0.5, 1,0)

conf_mat <- table(spam = train[,ncol(data)] , predicted_val = probalities)
miss_classfication <- (sum(conf_mat[ upper.tri(conf_mat) ]) +
                         sum(conf_mat[ lower.tri(conf_mat) ])) / sum(conf_mat) #missclassfication
```

```
miss_classfication_step4
```

```
## [1] 0.4788321
```

```
miss_classfication
```

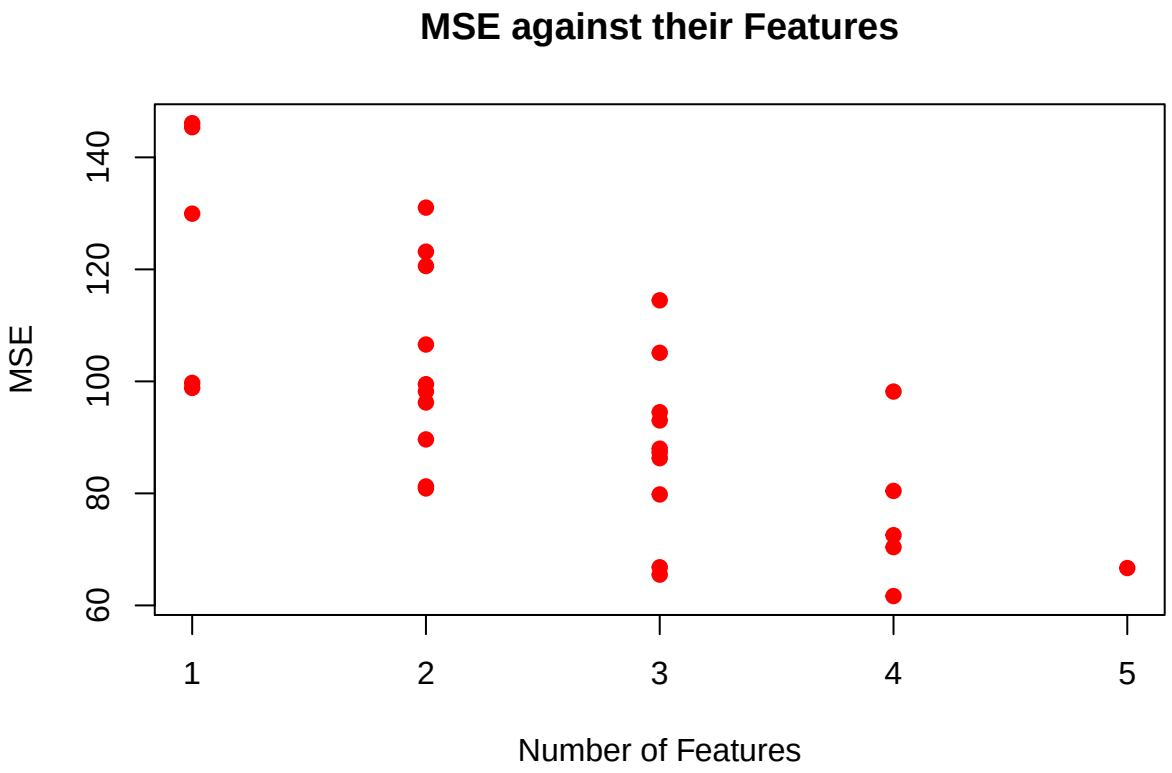
```
## [1] 0.470073
```

The mis-classification rate obtained from calculated kkn is 0.4788321 and that by using kknn package is 0.470073 which is almost similar

```
# Assignment 1 part 6
pi_values <- seq(from = 0.05, to= 0.95 , by=0.05)
Y <- train[,ncol(data)]
kkn <-  kknn(Spam ~., train , test , k = 5) #built in Knn for k = 5
knearest_p <- knearest(train, 5 , test) #knearst k = 5
kkn_p <- kkn$fitted.values # knn proabilties


###### in order to manually implement sensitivity and specificity

ROC <- function(Y, Yfit, p){
  m=length(p)
  TPR=numeric(m)
  FPR=numeric(m)
  for(i in 1:m)
  {
    t <- table(Y,Yfit>p[i])

    TPR[i] <-  t[2,2]/sum(t[2,])
    FPR[i] <-  t[1,2]/sum(t[1,])
  }
  return (list(TPR=TPR,FPR=FPR))
}

roc_curve_knearest <- ROC(Y, knearest_p , pi_values)
roc_curve_kkn_p <- ROC(Y, kkn_p , pi_values)

#plot graoh
plot <- plot(x = roc_curve_knearest$FPR,y = roc_curve_knearest$TPR,type = "l", main = "ROC Curve", xlab
lines(x= roc_curve_kkn_p$FPR,y = roc_curve_kkn_p$TPR, col = "blue")
points(x = roc_curve_knearest$FPR,y = roc_curve_knearest$TPR)
```

```
points(x= roc_curve_kkn_p$FPR,y = roc_curve_kkn_p$TPR, col = "blue")

sensitivity_kn <- 1 - roc_curve_knearest$FPR
sensitivity_knn <- 1 - roc_curve_kkn_p$FPR
```

Assignment 3 Feature Selection By Cross Validation




```
## $CV
## [1] 61.66766
##
## $Features
## [1] 1 0 1 1 1
```

The optimal subset of features are f1,f3,f4 and f5 i-e 10111, these specific features have the largest impact on target because the MSE values for the selected subset of features is lower than any other subset of features which implies that the prediction madeby the selected features greatly describes the target.

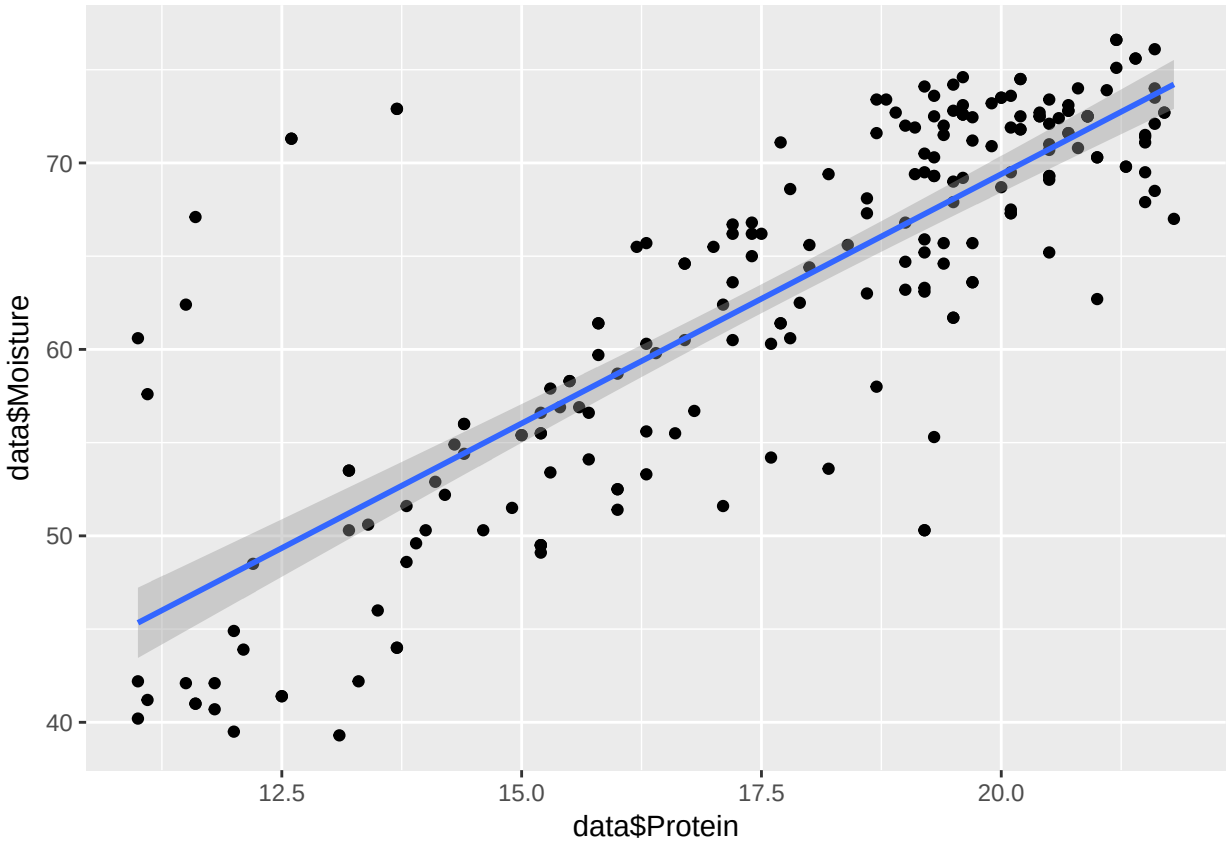
APPENDIX

```
#linear regression return response
mylin=function(X,Y, Xpred){
  Xpred1=cbind(1,Xpred)
  X=cbind(1,X)
  beta <- solve(t(X) %*% X) %*% t(X) %*% Y
  Res = Xpred1%*%beta
  return(Res)
}
```

```r
#my cv function
myCV=function(X,Y,Nfolds){
  n=length(Y)
  p=ncol(X)
  set.seed(12345)
  ind=sample(n,n)
  X1=X[ind,]
  Y1=Y[ind]
  sF=floor(n/Nfolds)
  MSE=numeric(2^p-1)
  Nfeat=numeric(2^p-1)
  Features=list()
  curr=0

  #we assume 5 features.
  for (f1 in 0:1)
    for (f2 in 0:1)
      for(f3 in 0:1)
        for(f4 in 0:1)
          for(f5 in 0:1){
            model= c(f1,f2,f3,f4,f5)
            if (sum(model)==0) next()
            SSE=0

            # generating sequence
            lower_index_seq <- seq(1,n,sF)
            upper_index_seq <- seq(0,n,sF)

            current_selected_feature <- which(model == 1)
            X2<- X1[,current_selected_feature,drop=F] #apply k fold

            for (k in 1:Nfolds)
            {
              i <- lower_index_seq[k]
              j <- upper_index_seq[k+1]

              k_fold_ind <- ind[i:j] # calculating indexes

              Xpred <- X2[k_fold_ind,]
              Xt <- X2[-k_fold_ind,]

              Yp <- Y1[k_fold_ind]
              Yt <- Y1[-k_fold_ind]

              Ypred <- mylin(Xt,Yt, Xpred)
              SSE=SSE+sum((Ypred-Yp)^2)
            }
            curr=curr+1
            MSE[curr]=SSE/n
            Nfeat[curr]=sum(model)
            Features[[curr]]=model
```

```
        }

    plot(x=Nfeat,y=MSE,  main = "MSE against their features",
         xlab = "Number of features", ylab ="MSE" ,
         col = "red", pch=19, cex=1)
    i=which.min(MSE)
  return(list(CV=MSE[i], Features=Features[[i]]))
}

myCV(as.matrix(swiss[,2:6]), swiss[[1]], 5)
```

Assignment 4 Linear Regression & Regulaization



From the linear regression line,it is evident that linear fitting is not good for the given data as data points spread widely and many points greatly deviates from the regression line.

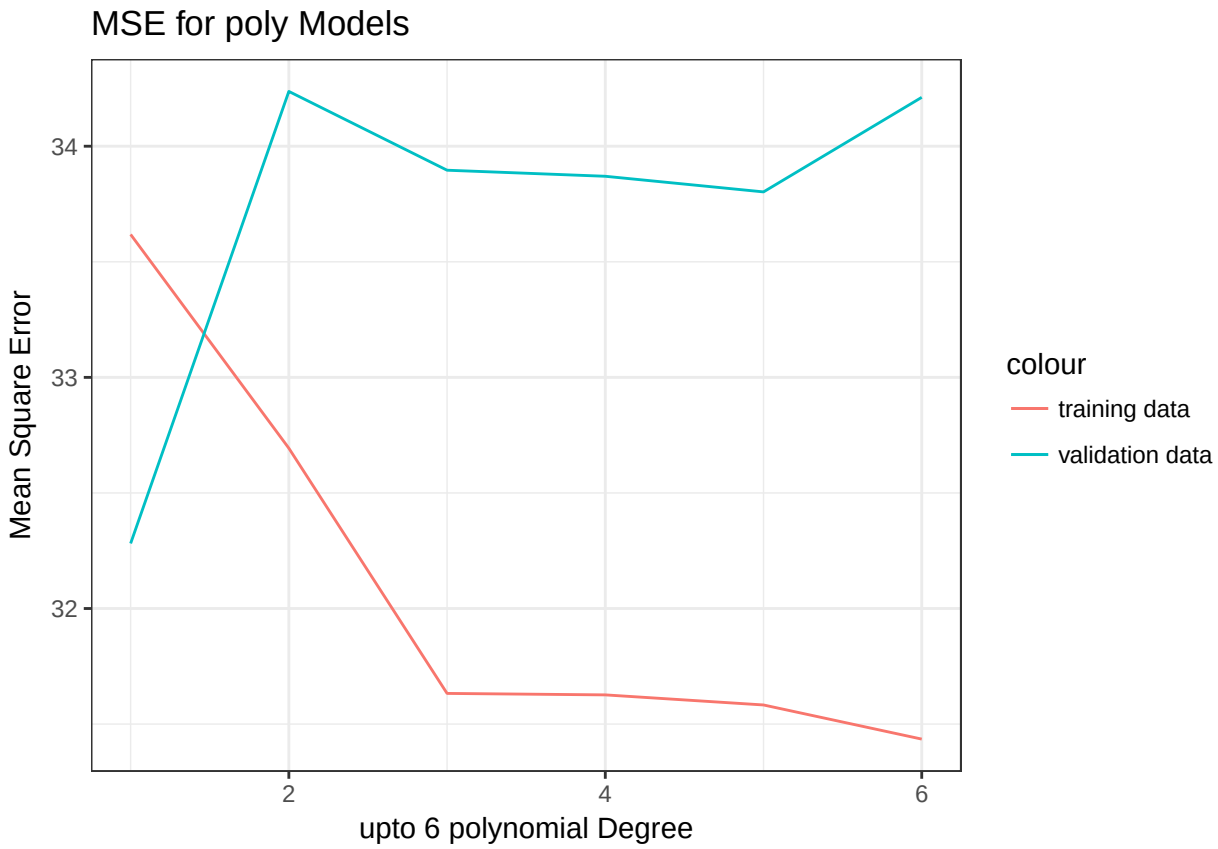
Assignment 4 part 2

Moisture $\sim N(w_o + w_1x^1 + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5 + w_6x^6, \sigma^2)$ or $M = w_o + w_1x^1 + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5 + w_6x^6 + \epsilon$ where $\epsilon \sim N(0, \sigma^2)$ and x = Protein

Since linear regression overfits data for higher polynomial. In such case ridge regression would be an appropriate probabilistic model to implement.

For Normal distribution, MSE criterion is the best unbiased estimator which is equivalent to the sample variance thus giving the best value of mean square error that is close to zero.

Assignment 4 part 3

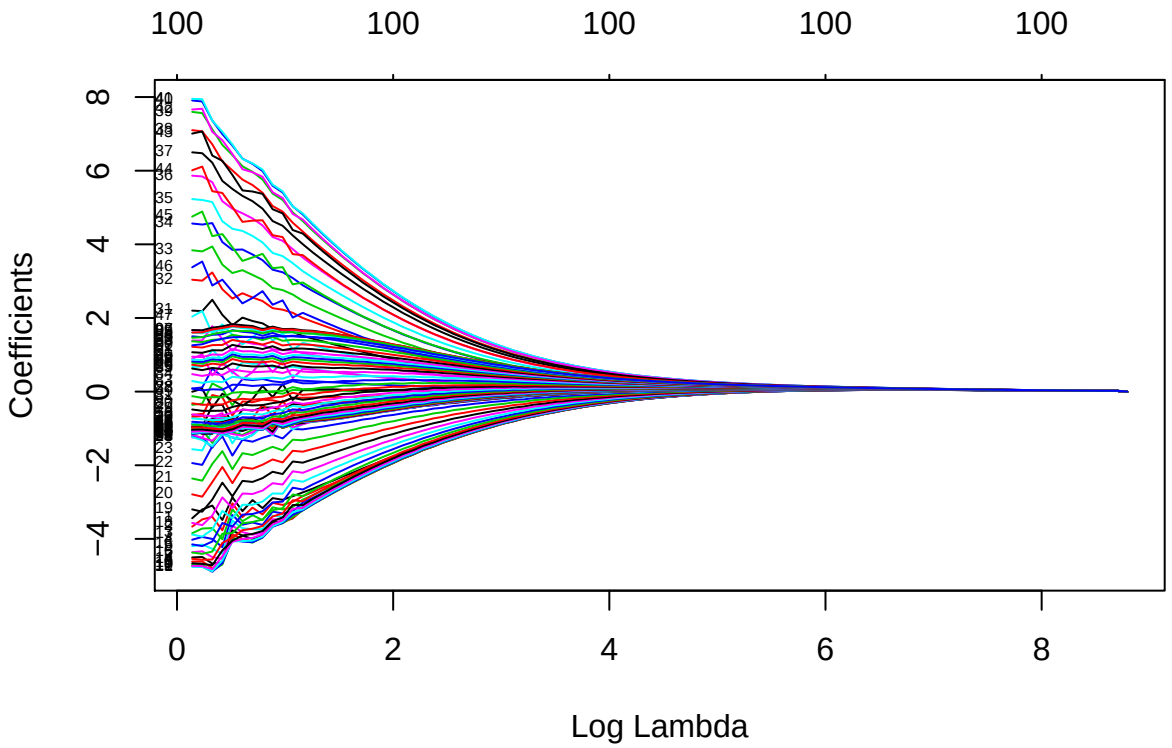


Accourding to the plot X-axis show the model Number with respect to its complexity from 1..6, model with Less MSE value is best so in that case model with ploynomial degree 6 of training data has less MSE value so this model is good. Moreover it has less variance but due to the higher order of polynomial,it has high complexity that is the bias factor is more.

Assignment 4 part 4

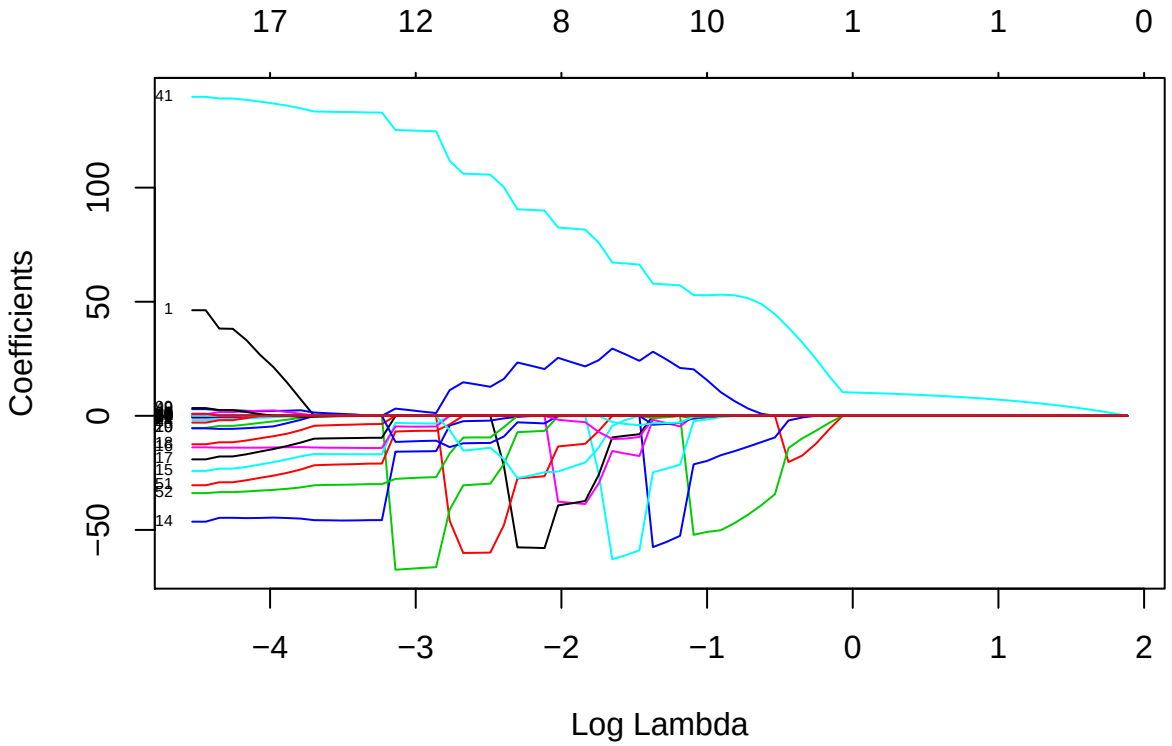
38 variables have been selected for the final model which can be shown by anova component of stepwise function

Assignment 4 part 5



With increase in log lambda, the model coefficients converges to 0 that is for higher values of lambda, the model coefficient shrinks

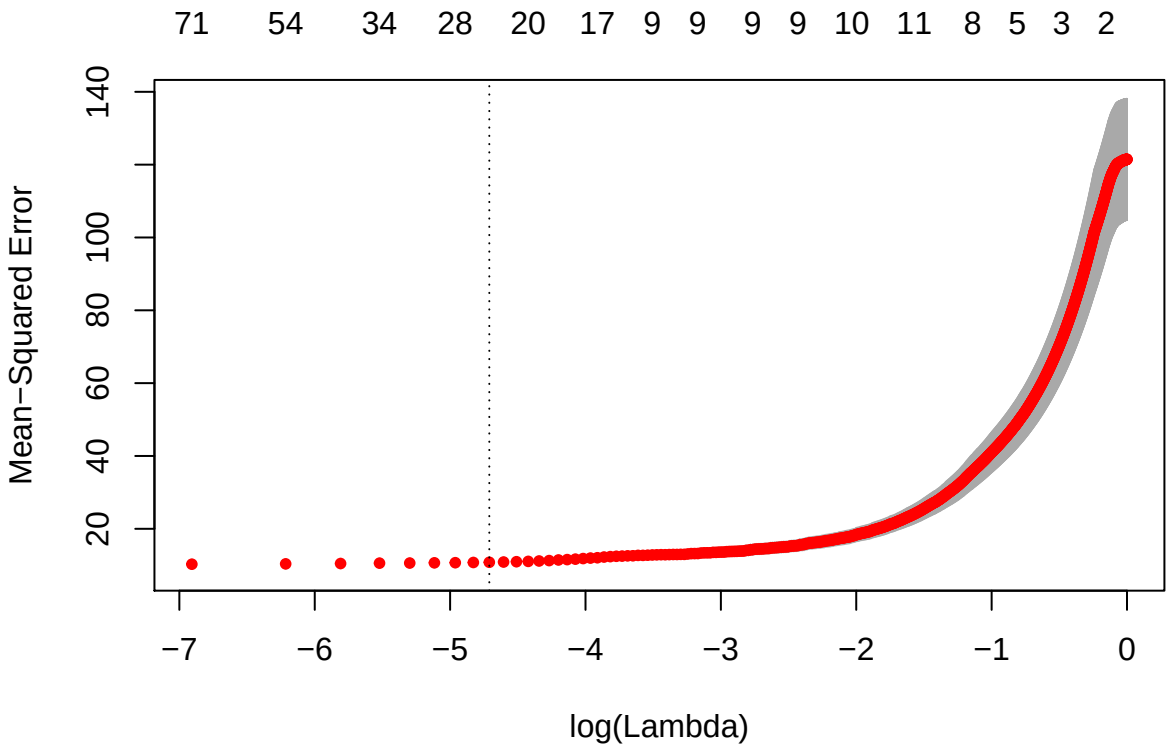
Assignment 4 part 6



for $\lambda < 0$, the model coefficients do not converges fast, but at $\lambda = 0$, as lambda increase, the number of parameters are driven to 0 except for 1 higher coefficient which decays slowly when λ becomes greater than 0

When the lasso plot is compared to the one for ridge regression, the selection of model coefficients is evident that is for ridge regression all coefficiennts are selected while Lasso does not select all coefficients. Moreover for ridge regression, the coefficients does not converges to 0 for lambda = 0 as compared to the plot of Lasso

Assignment 4 part 7



$\lambda = 0$ is the optimal lambda value. The best model is indicated by the dotted line in plot. For $\lambda = 0$ all variables have been chosen since when $\lambda = 0$, there is no shrinkage.

It can be shown from the plot that as lambda increases, the mean squared error increases. For the highest value of lambda that is 1, the mean squared error becomes more than 100

Assignment 4 part 8

The selection of variable in step 4 is based on the AIC value selecting only 38 variables for the best model, giving least error while in step 7, the selection of variable is dependent on the value of lambda. As lambda increases, the shrinkage is increased as a result error is increased. For lambda = 0, the error is least, and there is no shrinkage as a result all variables are chosen.

APPENDIX

```
library(readxl)
library(ggplot2)

#qestion 4.1
data <- read_excel("tecator.xlsx") #load a data
data <- as.data.frame(data) # coonvert into data frame

Moisture_plot <- ggplot(data, aes( x=data$Protein, y=data$Moisture) ) +
  geom_point() + geom_smooth(method=lm)

create_model_and_plot <- function(training, validation)
```

```r
{
  result <- list()
  training_mse <- c()
  validation_mse <- c()
  for (i in 1:6)
  {
    #for traning data
    model <- lm(Moisture ~ poly(Protein, degree = i, raw = TRUE) , data = training)
    y_hat <- predict(model, training)

    training_mse[i] <- mean((training$Moisture - y_hat)^2 )
    # validation data fiting
    y_hat <- predict(model,  validation)
    validation_mse[i] <- mean((validation$Moisture - y_hat)^2)
  }
  ind <- seq(from=1,to=6,by=1)
  result <- data.frame(training=training_mse, validation=validation_mse, x=ind)
}

n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,] #train is data
validation=data[-id,] #test data is newData
result<- create_model_and_plot(train, validation)


MSE_plot <- ggplot()+ geom_line(aes(x=result$x,y=result$training, color ="training data" )) +
  geom_line(aes(x =result$x, y = result$validation, color = "validation data"))+
  ggtitle("MSE for poly Models")+xlab("upto 6 polynomial Degree") +ylab("Mean Square Error")

# Assignment 4 part 4

library(MASS)
new_data <- data[2:102]
fit <- lm(Fat ~ . , data = new_data)
coef(fit)
step <- stepAIC(fit, direction = "both")
step$anova
summary(step)


# Assignment 4 part 5

library(glmnet)
response <- new_data$Fat
predictors <- as.matrix(new_data[1:100])
model10 <- glmnet(predictors ,response, alpha = 0, family = "gaussian"  )
plot(model10, xvar="lambda", label=TRUE)

# Assignment 4 (6)

model10 <- glmnet(predictors ,response, alpha = 1, family = "gaussian"  )
```

```r
plot(model10, xvar="lambda", label=TRUE)

# Assignment 4 (7)

set.seed(12345)
lambda.seq <- seq(0,1,0.001)
model=cv.glmnet(predictors, response, alpha=1,family="gaussian",lambda= lambda.seq)
model$lambda.min
coef(model, s="lambda.min")
model$lambda.1se
coef(model, s="lambda.1se")
plot(model)
```