Lab1_Report

Saman Zahid

11/22/2017

Note: The changes are highlighted by bolding and change in code is mentioned with "change here" comment (which is in assignment 1)

Assignment 1 Spam Classification with Nearest Neighbours

```
## [1] 0.3883212
```

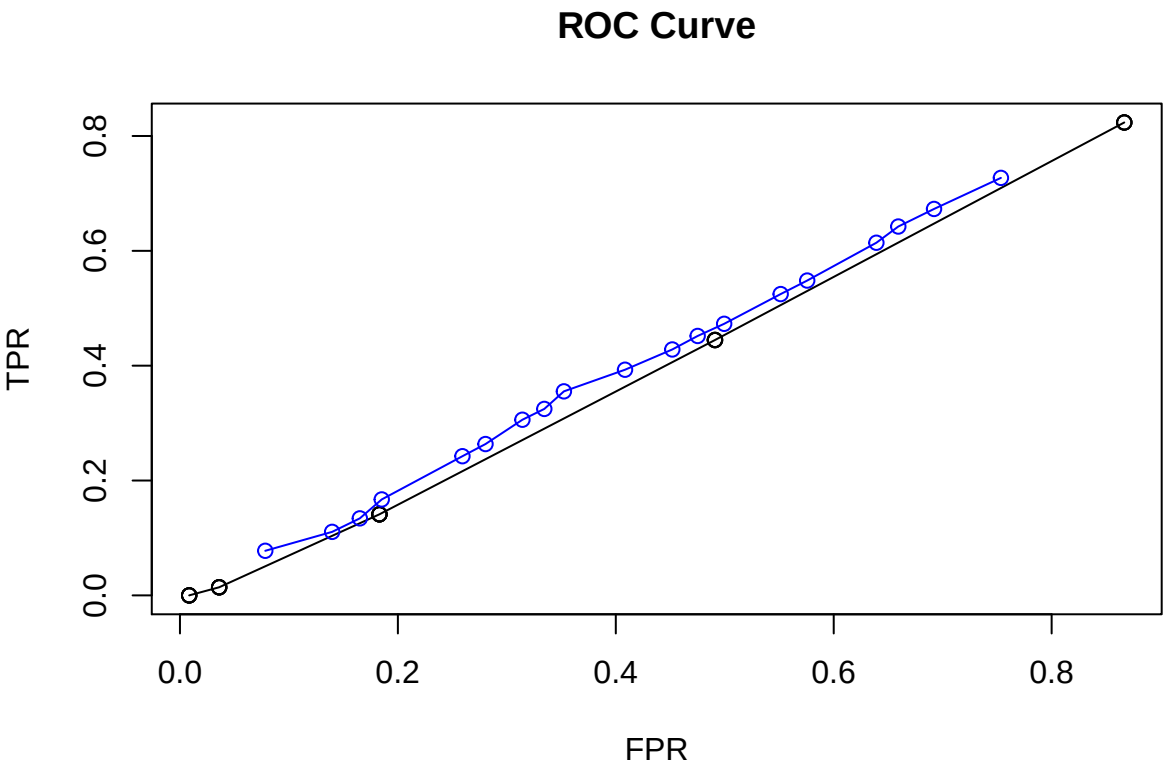
```
## [1] 0.3459854
```

The mis-classification rate for k=5 obtained from calculated kkn is 0.3883212 and that by using kknn package is 0.3459854. Since the value obtained from kknn package is less thus kknn is a better classifier

```
## [1] 0.429927
```

```
## [1] 0.3883212
```

The mis-classification rate for k=5 obtained from calculated kkn is 0.429927 and that by using kknn package is 0.3459854. Since the value obtained from kknn package is much less thus kknn is a better classifier



ROC curves for self implemented knearest method is almost similar to the one created using kknn package.Since area under the curve for the line created on the basis of kknn is slightly larger than that of knearest function, therefore it is a better classifier

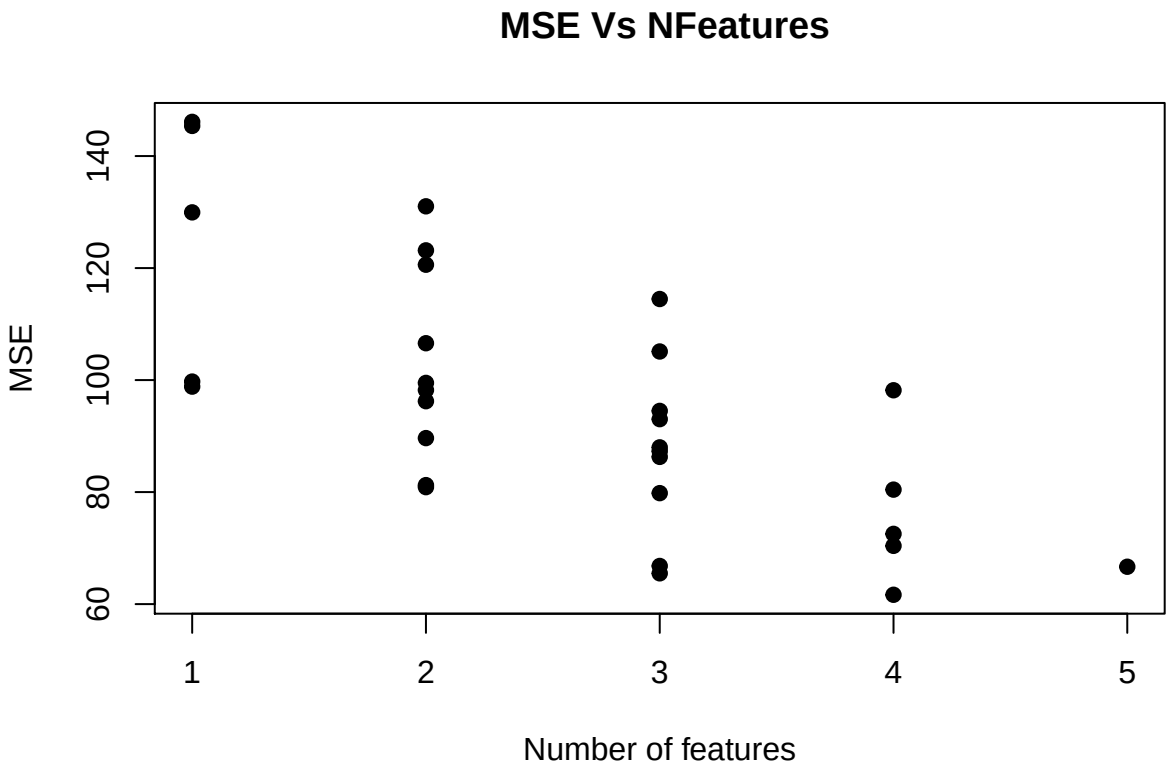
APPENDIX

```
n=dim(spambase)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=spambase[id,]
test=spambase[-id,]


data <- train
newdata <- test


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])

  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

  Ni = numeric(k)

  for (i in 1:n2 ){

    distMat = order(D[i,])
    Ni = data[distMat[1:k],"Spam"]

    Prob[i] <- sum(Ni)/k

  }

  return(Prob)
}

prob <- knearest(data,5,newdata)


col_num <- ncol(data)
c_mat <- table(newdata[,col_num],prob > 0.5) ## change here

# misclassification rate
```

```r
mis_class <- sum(c_mat[row(c_mat) != col(c_mat)])/ sum(c_mat)

# Assignnment 1 part 4

prob <- knearest(data,1,newdata)


col_num <- ncol(data)
c_mat <- table(newdata[,col_num],prob > 0.5) ## change here

# misclassification rate

mis_class <- sum(c_mat[row(c_mat) != col(c_mat)])/ sum(c_mat)

# Assignment 1 part 5
library(kknn)

kkn <- kknn(Spam~.,data,newdata,k=5)

probability <- kkn$fitted.values

Y_pred <- ifelse(probability > 0.5,1,0)

col_num <- ncol(data)

# Confusion matrix

c_mat <- table(newdata[,col_num],Y_pred)

# misclassification rate

mis_class <- sum(c_mat[row(c_mat) != col(c_mat)])/ sum(c_mat)

kkn <- kknn(Spam~.,data,newdata,k=1)

probability <- kkn$fitted.values

Y_pred <- ifelse(probability > 0.5,1,0)

col_num <- ncol(data)

# Confusion matrix

c_mat <- table(newdata[,col_num],Y_pred) ## change here

# misclassification rate

mis_class <- sum(c_mat[row(c_mat) != col(c_mat)])/ sum(c_mat)

# Assignment 1 part 6

kkn <- kknn(Spam~.,data,newdata,k=5)
```

```r
kkn_probability <- kkn$fitted.values

knearest_prob <- knearest(data,5,newdata)


pi_values <- seq(from = 0.05, to = 0.95, by= 0.05)

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

ROC=function(Y, Yfit, pi_value){
  m=length(pi_value)
  TPR=numeric(m)
  FPR=numeric(m)


  for(i in 1:m){

    t=table(Y,Yfit > pi_value[i])

    N_p <- sum(t[2,])
    N_n <- sum(t[1,])

    TPR[i]= t[2,2]/N_p
    FPR[i]= t[1,2]/N_n
  }
  return (list(TPR=TPR,FPR=FPR))

}

Y <- data[,ncol(data)]
knearest_ROC <- ROC(Y,knearest_prob,pi_values)

kknn_ROC <- ROC(Y,kkn_probability,pi_values)

plot <- plot(x = knearest_ROC$FPR,y = knearest_ROC$TPR,type = "l", main = "ROC Curve", xlab = "FPR", yla
lines(x= kknn_ROC$FPR,y = kknn_ROC$TPR, col = "blue")
points(x = knearest_ROC$FPR,y = knearest_ROC$TPR)
points(x= kknn_ROC$FPR,y = kknn_ROC$TPR, col = "blue")


sensitivity_kn <- 1 - knearest_ROC$FPR
sensitivity_knn <- 1 - kknn_ROC$FPR
```

Assignment 3 Feature Selection By Cross Validation




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

The optimal subset of features is f1,f3,f4 and f5 i-e 10111 because the risk factor for the selected feature is least. While selecting a feature we check for the minimum risk. The selected feature has the lowest the Mean Squared Error and the predicted or fitted value using regression for the selected features best interprets the target variable that is gives the predicted value closest to the target values thus it can be assumed that the selected features have the largest impact on target.

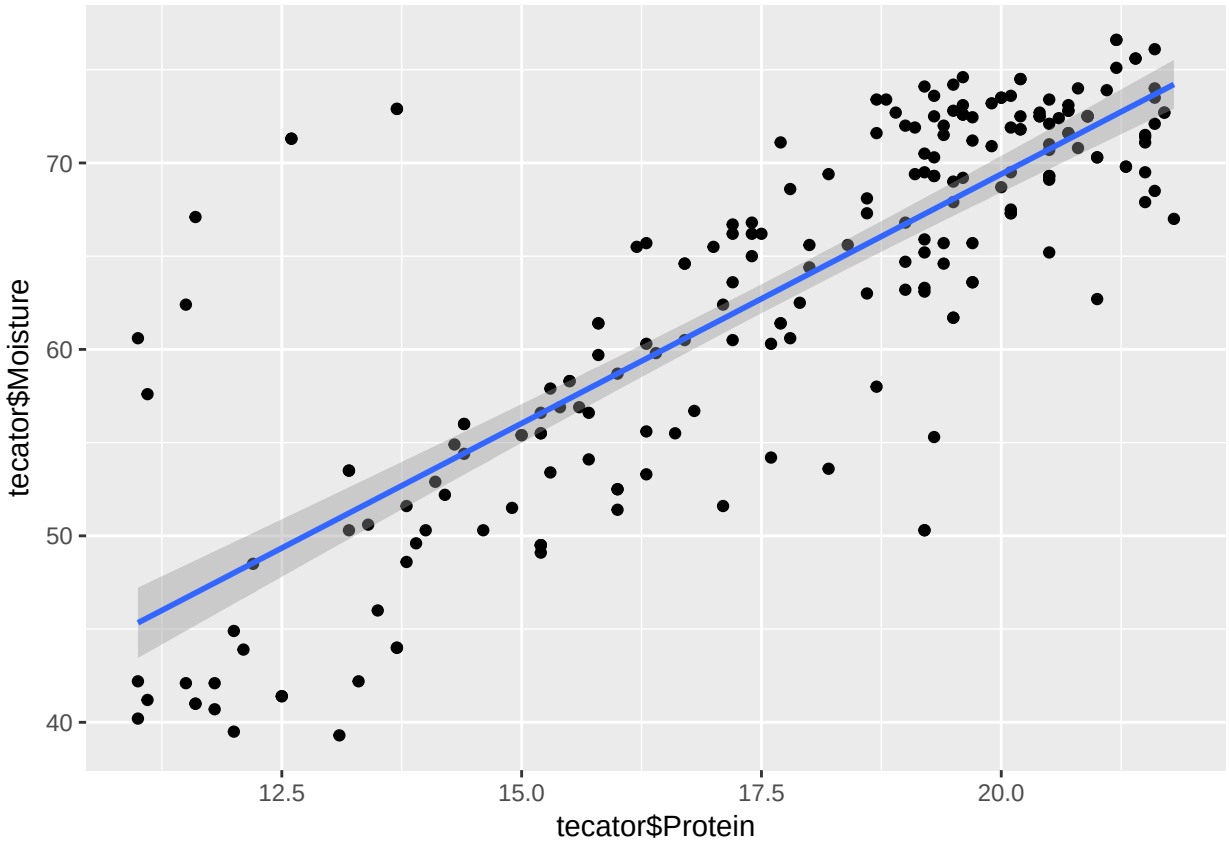
APPENDIX

```
#linear regression
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)
}
```

```
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

            #selecting model
            curr_model <- which(model == 1)

            # Generating sequences to select index later
            lower_seq1 <- seq(1, n, sF)
            upper_seq2 <- seq(0, n, sF)


            for (k in 1:Nfolds){

              i <- lower_seq1[k]
              j <- upper_seq2[k+1]

              # Selecting n/kfold indices
              ind_fold <- ind[i:j]

              Xtest <- X1[ind_fold,curr_model]
              Xtrain <- X1[-ind_fold,curr_model]

              Ytrain <- Y1[-ind_fold]
              Yp <- Y1[ind_fold]

              Ypred <- mylin(X = Xtrain,Y = Ytrain,Xpred = Xtest)
              SSE=SSE+sum((Ypred-Yp)^2)
            }
            curr=curr+1
            MSE[curr]=SSE/n
            Nfeat[curr]=sum(model)
            Features[[curr]]=model
```

```
        }

  plot(Nfeat,MSE, main = " MSE Vs NFeatures", type = "p", xlab = "Number of features",
       ylab = "MSE", 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

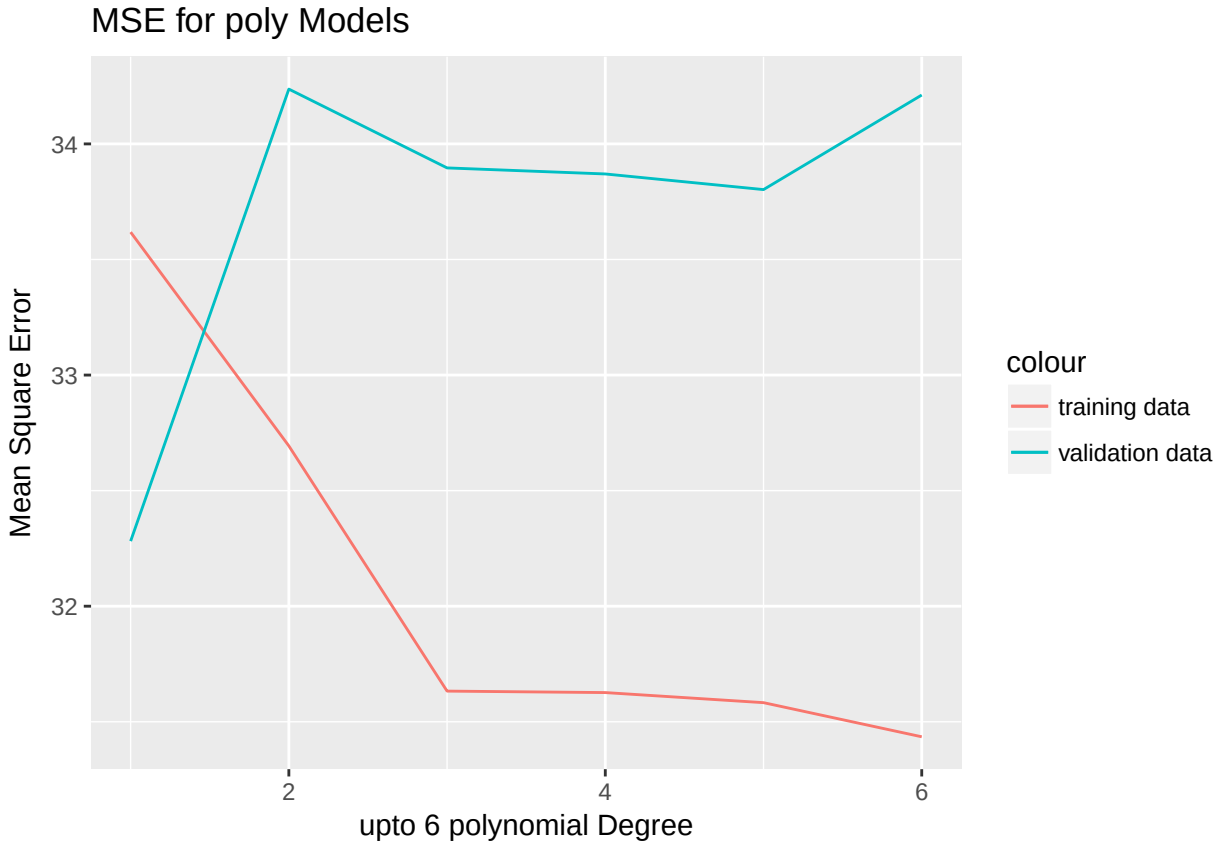
Moisture $\sim N(w_o + w_1x^1 + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5 + w_6x^6, \sigma^2)$ or $M = w_o + w_1x^1 + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5 + w_6x^6 + \epsilon$ where $\epsilon \sim N(0, \sigma^2)$ and x = Protein

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

```
## [1] 33.61836 32.69342 31.63266 31.62641 31.58273 31.43513
## [1] 32.28154 34.23708 33.89615 33.86992 33.80234 34.21152
```

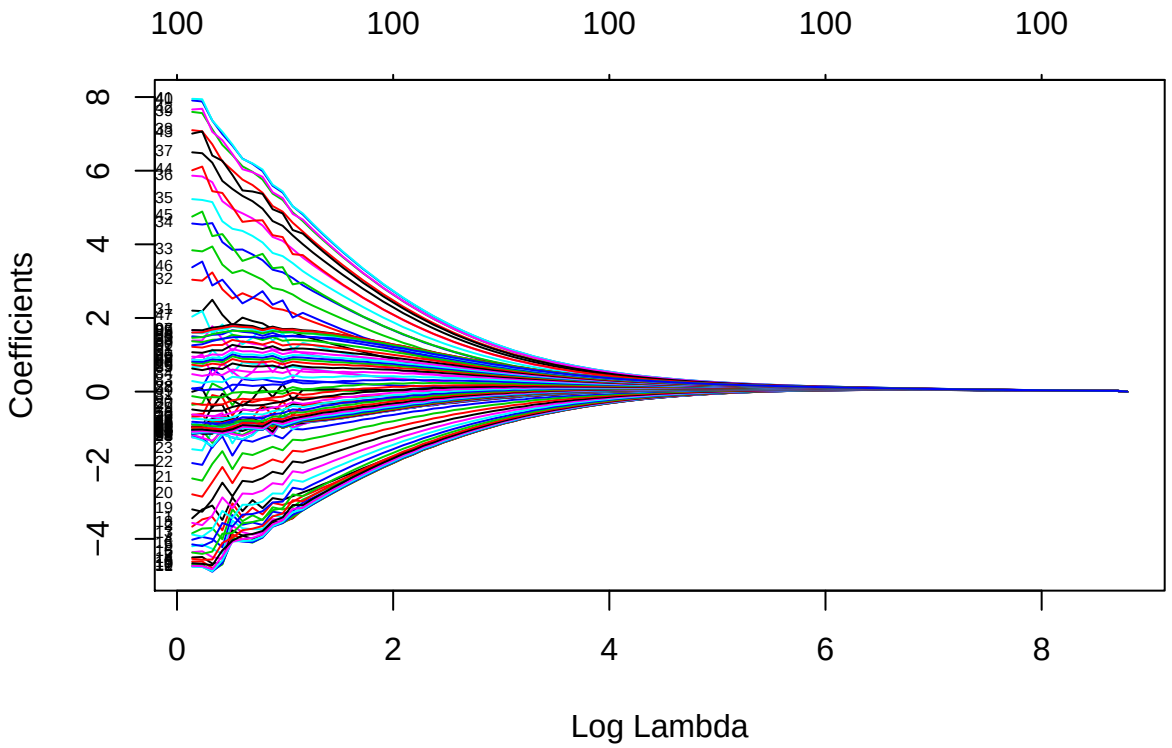


Bias variance tradeoff can be explained as change in variance for different set of data. Since for the unbiased estimator MSE is equal to variance, and it is evident from the graph that for training data the variance is decreasing (as expected) but for the validation data the variance is increasing. The lowest value for the validation set is for the polynomial of degree 1 therefore it is the best model.

Assignment 4 part 4

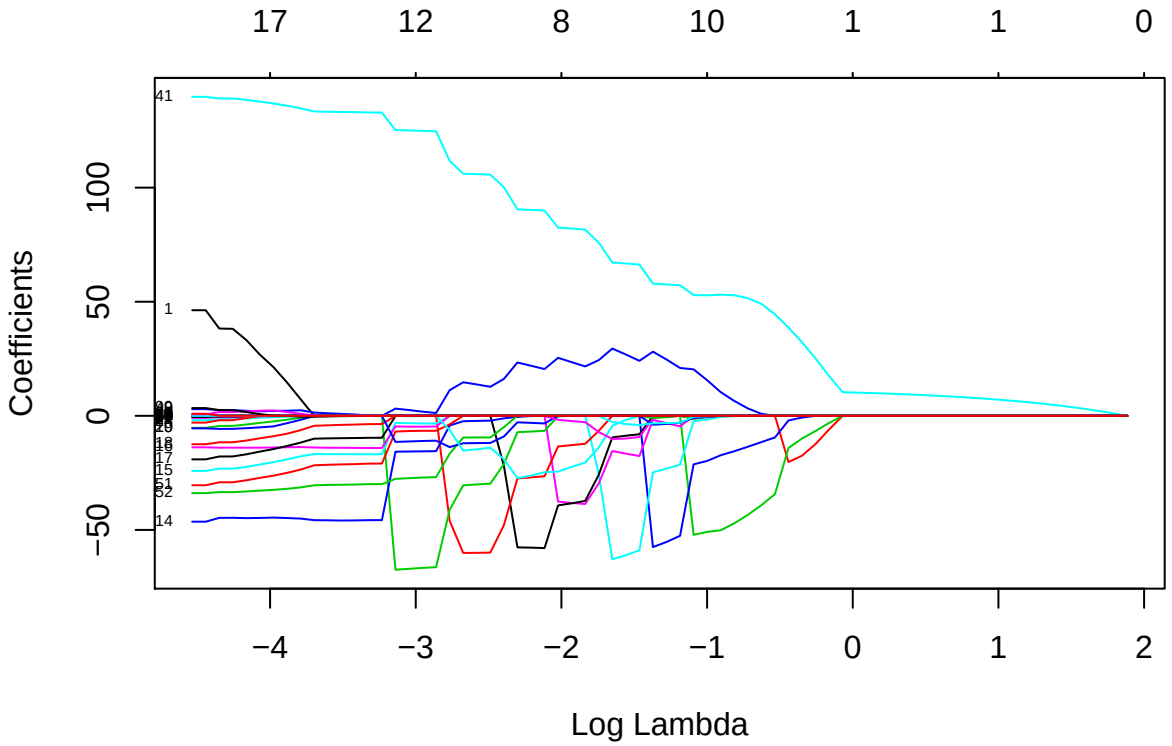
63 variables have been selected for the final model which can be shown by summary function

Assignment 4 part 5



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

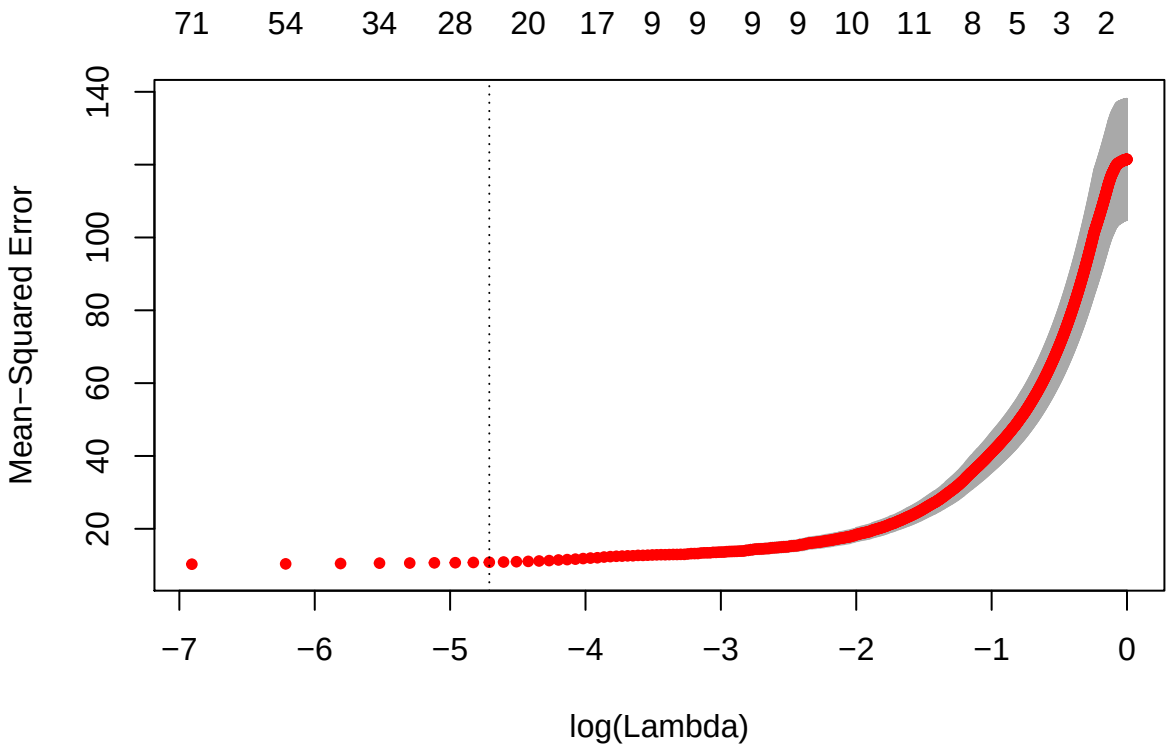
Assignment 4 part 6



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 63 variables while in step 7, the selection of variable is dependent on the value of lambda(the shrinkage coefficient) selecting all 100 variables.

APPENDIX

```
library(readxl)
library(ggplot2)
tecator <- read_excel("tecator.xlsx")

tecator <- as.data.frame(tecator)

ggplot(tecator,aes(tecator$Protein,tecator$Moisture)) + geom_point(aes(tecator$Protein,tecator$Moisture


## Assignment 4 (3)

n=dim(tecator)[1]
```

```r
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=tecator[id,]
validation=tecator[-id,]
SSE = 0

MSE_train <- c()
MSE_valid <- c()

for (i in 1:6)
{
  M_train <- lm(Moisture ~ poly(Protein, degree = i, raw = TRUE) , data = train)
  Y_pred_train = predict(M_train,train)
  Y_pred_valid = predict(M_train,validation)

  MSE_train[i] <- mean((train$Moisture - Y_pred_train)^2)
  MSE_valid[i] <- mean( (validation$Moisture  - Y_pred_valid)^2 )

}

x <- seq(1,6,1)
MSE_val <- data.frame(x=x, MSE_training = MSE_train, MSE_validation = MSE_valid)

MSE_train
MSE_valid

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

library(MASS)

Channeldata <- tecator[,2:102]
fit <- lm(Fat~., data = Channeldata)

step <- stepAIC(fit,direction = "both")

step$anova

summary(step)

# Assignment 4 part 5

library(glmnet)

Channels_cov <-tecator[,2:101]
response <- tecator[,102]

ridgefit <- glmnet(as.matrix(Channels_cov),response,alpha = 0,family = "gaussian")
# summary(ridgefit)
plot(ridgefit, xvar="lambda", label = "True")

# Assignment 4 (6)
```

```r
lassofit <- glmnet(as.matrix(Channels_cov),response,alpha = 1,family = "gaussian")
plot(lassofit, xvar="lambda", label = "True")

# Assignment 4 (7)

set.seed(12345)
lambda_seq <- seq(0,1,0.001)

model <- cv.glmnet(as.matrix(Channels_cov),response,alpha = 1,family = "gaussian",
                   lambda = lambda_seq)

model$lambda.min
model$glmnet.fit

# par(mfrow=c(1,1))
plot(model)

coef(model, s="lambda.min")
```