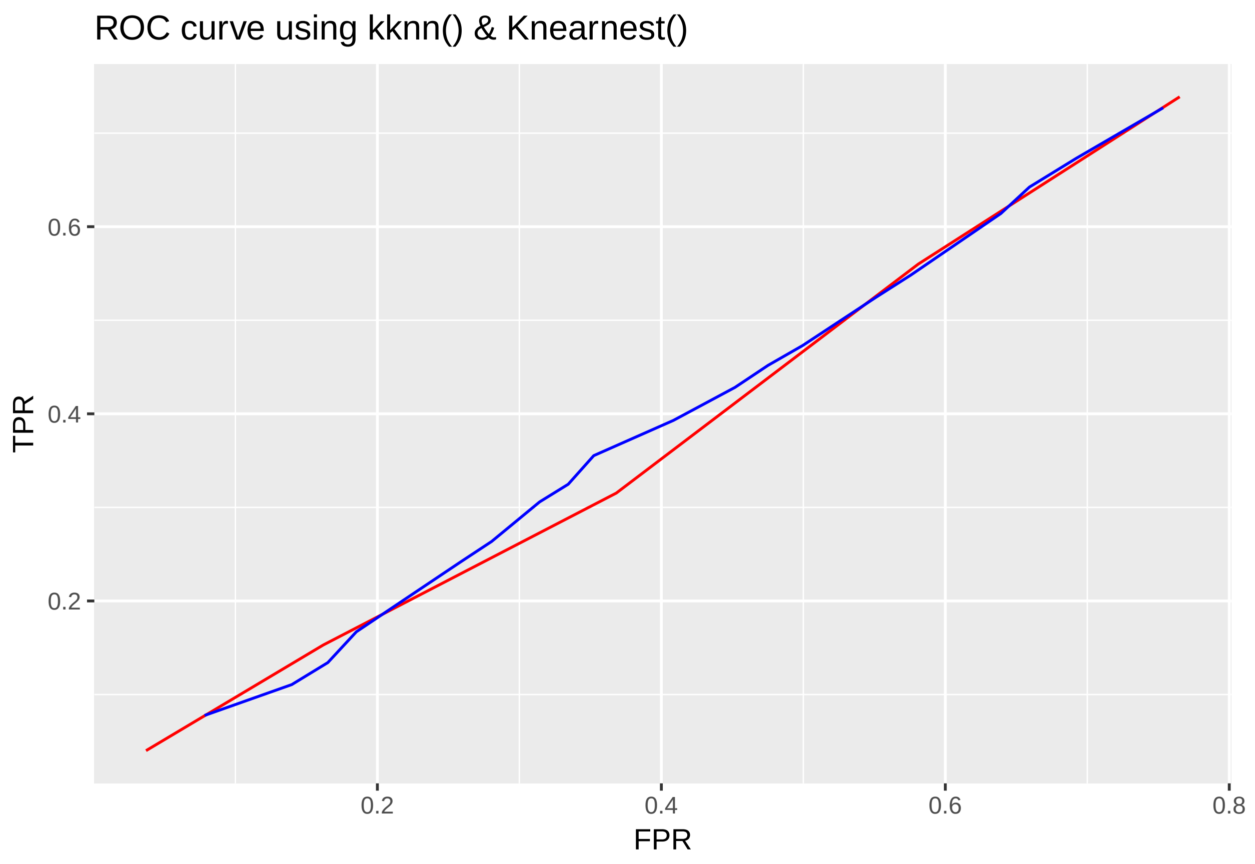
Machine Learning - Lab 1

Fahad Hameed

November 27, 2017

Assignment 1 Spam Classification with Nearest Neighbours



In the plot x-axis is FPR(False Positive Result) and y-axis is TPR(True Positive Result). The red line is the knearest classifier and the blue line is the built in knn. It is observed that the curve produced by the knearest is little better then the curve produced by knn.

APPENDIX

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

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

# Dividing Data into Test & Train (1)
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
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])

  # Calculating X-hat (2.1)
  X_hat = X/matrix(sqrt(rowSums(X^2)), nrow=n1, ncol=p-1)

  # Calculating Y-hat (2.2)
  Y_hat = Y/matrix(sqrt(rowSums(Y^2)), nrow = n2 , ncol = p - 1)

  # Matrix C (2.3)
  C <- X_hat %*% t(Y_hat)

  # Distance Matrix (2.4)
  D <- 1 - C

  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,]
    Prob[i] <- sum(N_i[,"spam"]) / k   # Knearest (Lecture 2b Slide 16)
  }
  return(Prob) #return proabilities
}

# Classifying (3)
probalities <- knearest(train,5, test)
probalities <- ifelse(probalities > 0.5, 1,0)
confusion_matrix_1 <- table(spam = test[,ncol(data)] , predicted_val = probalities)
miss_classfication_1 <- 1-sum(diag(confusion_matrix_1))/sum(confusion_matrix_1)
```

Misclassification rate obtained from calculated knearst by using k=5 is 0.3175182

```
miss_classfication_1
```

```
## [1] 0.3175182
```

The Confusion Matrix for knearest at k=5 is as follow

```
confusion_matrix_1
```

```
##     predicted_val
## spam   0   1
##    0 695 242
##    1 193 240
```

```
# Repeating with k=1 (4)
probalities <- knearest(train,1, test)
probalities <- ifelse(probalities > 0.5, 1,0)
confusion_matrix_2 <- table(spam = test[,ncol(data)] , predicted_val = probalities)
miss_classfication_2 <- 1-sum(diag(confusion_matrix_2))/sum(confusion_matrix_2)
```

At k=1 the Misclassification rate obtained is 0.3474453 with the same probability threshold means it will only take the closest training obeservation into consideration.

```
miss_classfication_2
```

```
## [1] 0.3474453
```

The Confusion Matrix for knearest at k=1 is as follow

```
confusion_matrix_2
```

```
##     predicted_val
## spam   0   1
##    0 639 298
##    1 178 255
```

Misclassification rate has increased from 0.3175182 to 0.3474453 as we change the value of k from 5 to 1 respectively.

```
# Using default knn (5)
knn <- kknn(Spam ~. , train, test , k = 5 )
probalities <- knn$fitted.values
probalities <- ifelse(probalities > 0.5, 1,0)
confusion_matrix_3 <- table(spam = test[,ncol(data)] , predicted_val = probalities)
miss_classfication_3 <- 1-sum(diag(confusion_matrix_3))/sum(confusion_matrix_3)
```

The Misclassification rate by using kknn package at k=5 is 0.3459854

```
miss_classfication_3
```

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

The Confusion Matrix for knearest at k=5 is as follow

```
confusion_matrix_3
```

```
##     predicted_val
## spam   0   1
##    0 640 297
##    1 177 256
```

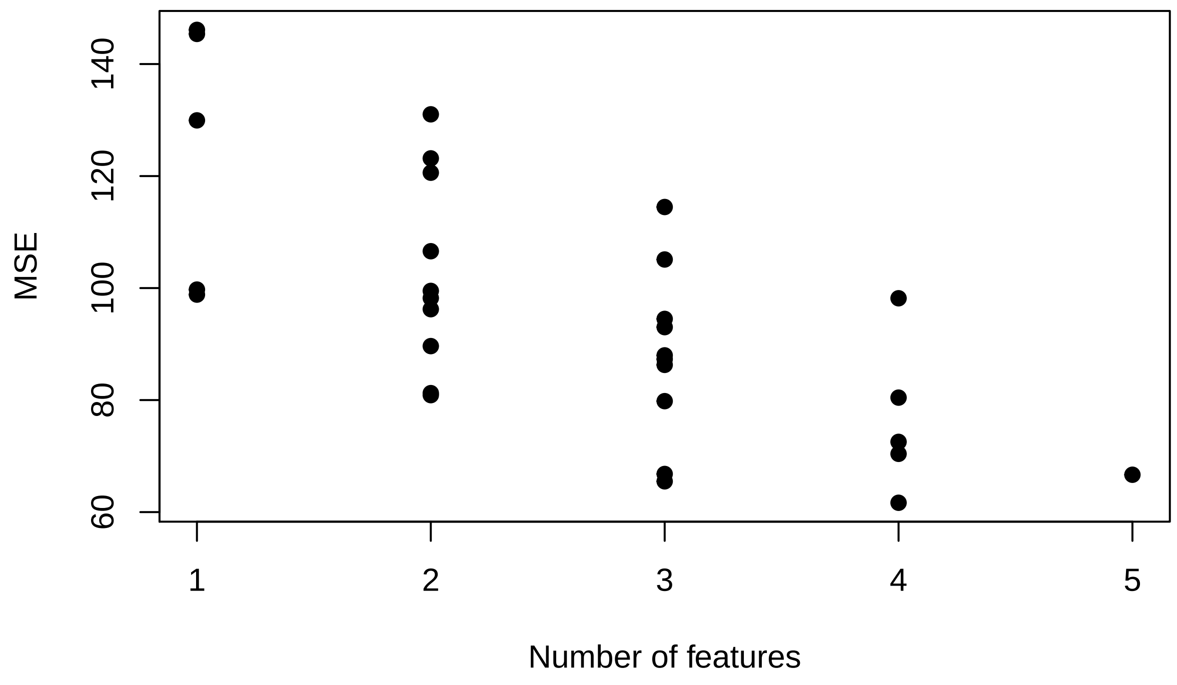
```
knn <- kknn(Spam ~. , train, test , k = 1 ) #standard kknn method at K = 5
probalities <- knn$fitted.values
probalities <- ifelse(probalities > 0.5, 1,0)
confusion_matrix <- table(spam = test[,ncol(data)] , predicted_val = probalities)
miss_classfication_4 <- 1-sum(diag(confusion_matrix))/sum(confusion_matrix)
```

Misclassification rate obtained from knearst with k=5 is 0.3175182 and at k=1 is 0.3474453 Whereas the misclassification rate by using kknn package at k=5 is 0.3459854. So the Misclassification rate at k=5 increased from 0.3175182 to 0.3459854 the rate change is 0.0284672. But the rate of Knn at k=5 is almost equal to knearest k=1 with very small difference.

In this step testing the two classifiers when we use different probability thresholds. We test thresholds from 0.05 to 0.95 with steps of 0.01.

```r
# 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

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)

# ploting the graph
X<-  as.data.frame(roc_curve_knearest)
Y <- as.data.frame(roc_curve_kkn_p)
ggplot() +
  geom_line(data = X, aes(x = X$FPR, y = X$TPR), color = "red") +
  geom_line(data = Y, aes(x = Y$FPR, y = Y$TPR), color = "blue")+
  ggtitle("ROC curve using kknn() & Knearest()")+xlab("FPR") +ylab("TPR")

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
```

As from the results it is seen that f1,f3,f4 and f5 represented as 1 0 1 1 1 are the optimal subset of features the MSE error score values for the selected subset of features is lower than any other subset of features. The prediction made by the selected features greatly describes the target. The plot between MSE and the Number of features shows as the number of features increases in the model the Mean square error MSE decreases. When increasing the number of features in the model the amount of data required to generalize the model increase exponentially. For this dataset it is Agriculture, Education, Catholic, Infant, Mortality. In general we can also observe that the more features that are selected the lower the error rate becomes.

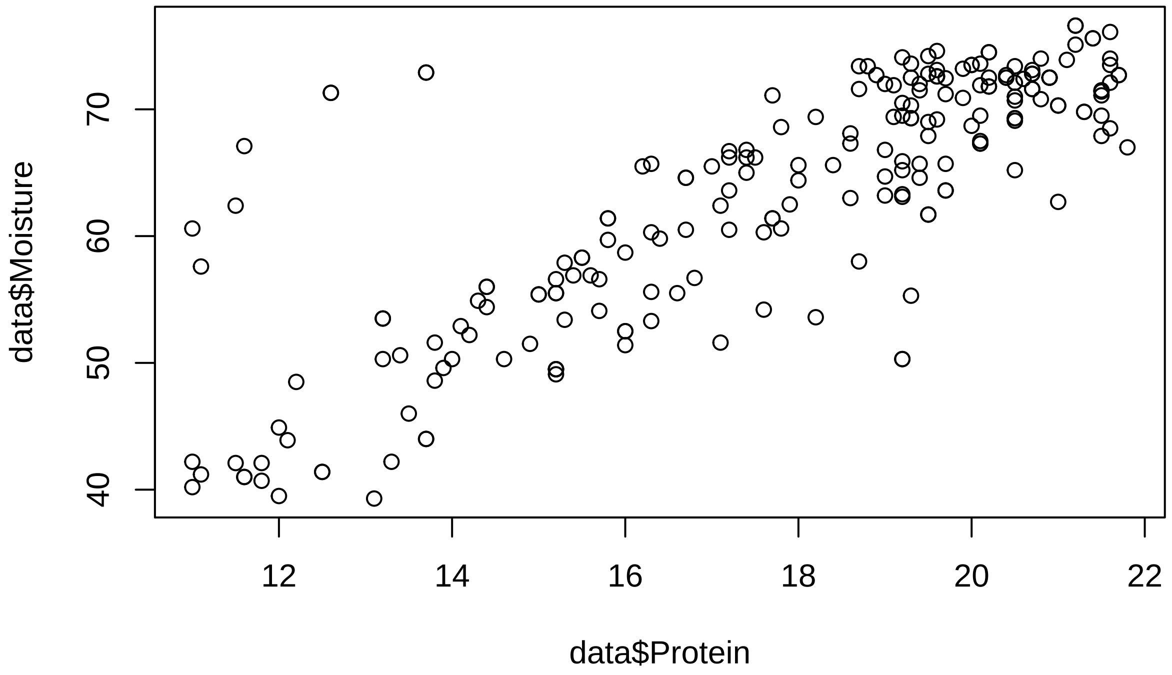
APPENDIX

```
mylin=function(X,Y, Xpred){
Xpred1=cbind(1,Xpred)
X= cbind(1,X)
beta <- solve(t(X) %*% X) %*% (t(X) %*% Y) # Lecture 1d slide(7)
Res=Xpred1 %*% beta
return(Res)
}
```

```r
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
seq_1 <- seq(1, n, sF)
seq_2 <- seq(0, n, sF)
ind_ <- which(model == 1)
for (k in 1:Nfolds){
i <- seq_1[k]
j <- seq_2[k+1]
# Selecting n/kfold indices
folds_ <- ind[i:j]
Xtest <- X1[folds_,ind_]
Xtrain <- X1[-folds_,ind_]
Ytrain <- Y1[-folds_]
Yp <- Y1[folds_]
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, type = "p", xlab = "Number of features",
ylab = "MSE", pch=19,cex=1)
#MISSING: plot MSE against number of features
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

Part 1



As in the plot of Protiens and moisture points are scattered and are away from the linear regression line so the data do not seems to be Linear.

Part 2

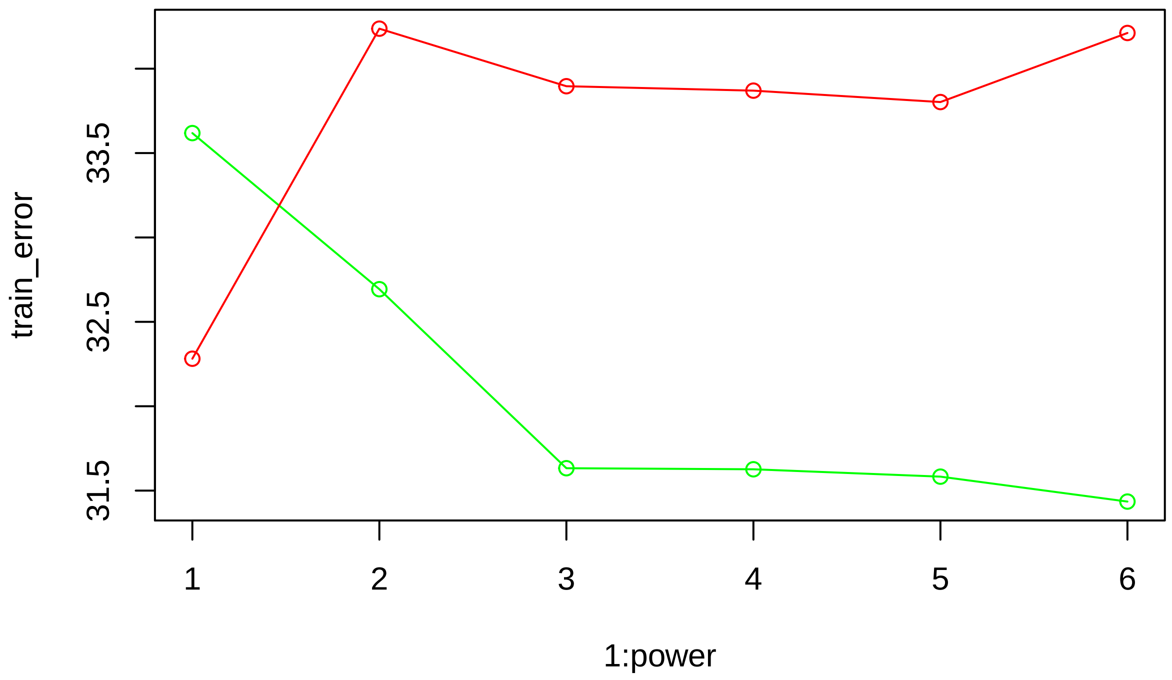
We approximate the model as w0 + w1xˆ1 + w2xˆ2 . . .

Moisture ~ $N(w_o + w_1x^1 + w_2x^2 + w_3x^3 + w_4x^4 + w_5x^5 + w_6x^6, \sigma^2)$ 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 Model is normally distributed Mean Square Error MSE criterion is the most appropiate to use when fitting the model to training data as MSE is the best unbiased estimator minimizing the sample variance giving MSE value nearer to zero which are the best.

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

Part 3

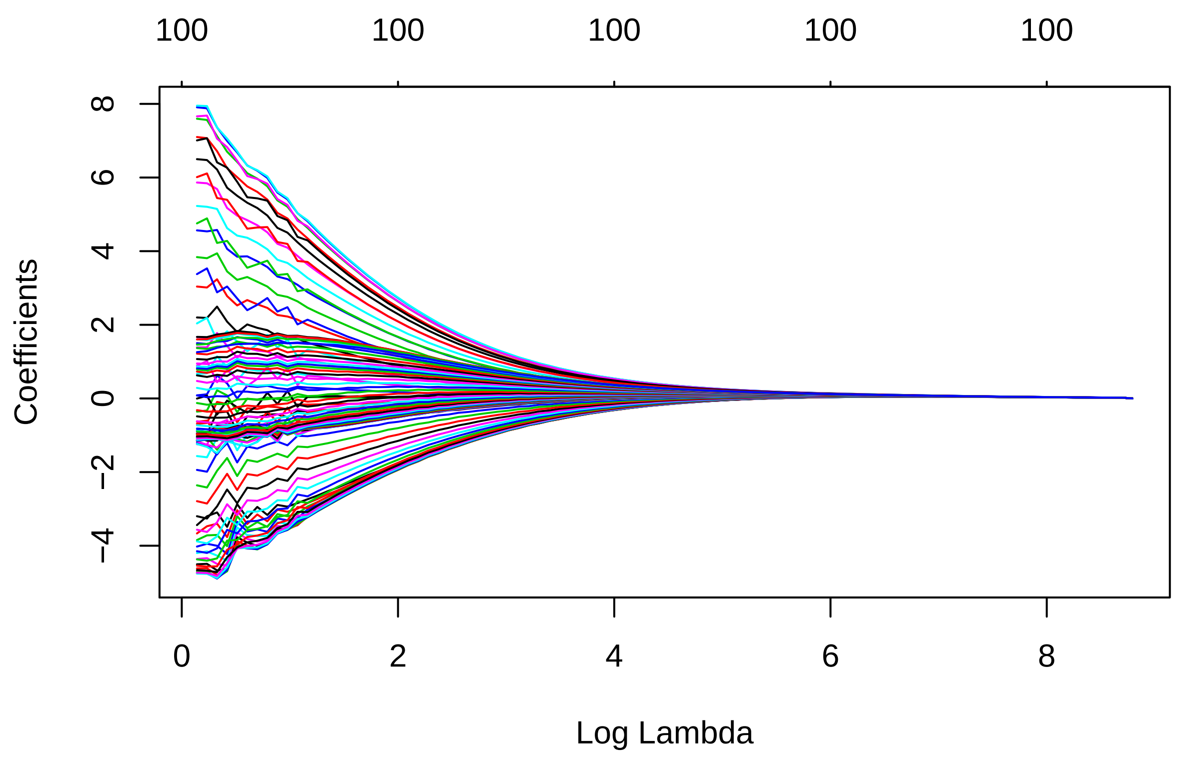


Observation of the plot indicates that as we increase the polynomial level the functiuon becomes more fitted for the training data and results in increasingly worse fit for the test data. As the model with Less MSE value is best so in that case model with ploynomial degree 1 of training data has less 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.

Part 4

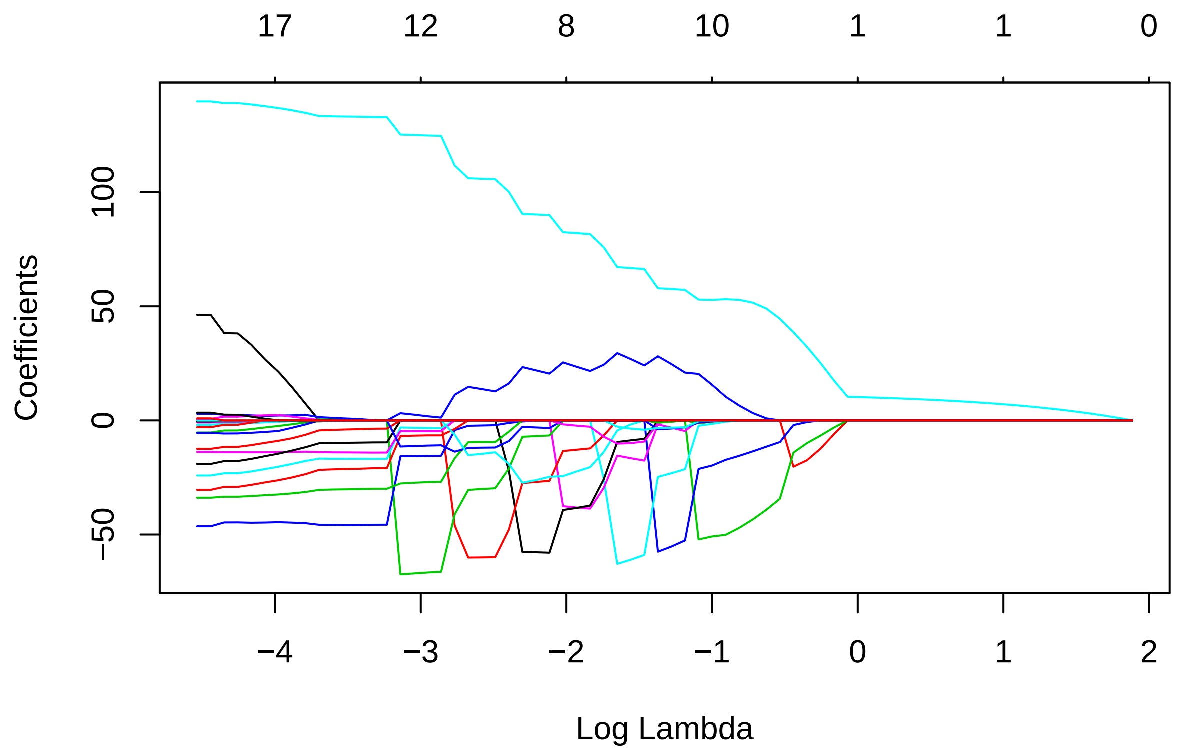
64 variables/columns have been selected for the final model which can be shown by anova component of stepwise function

Part 5



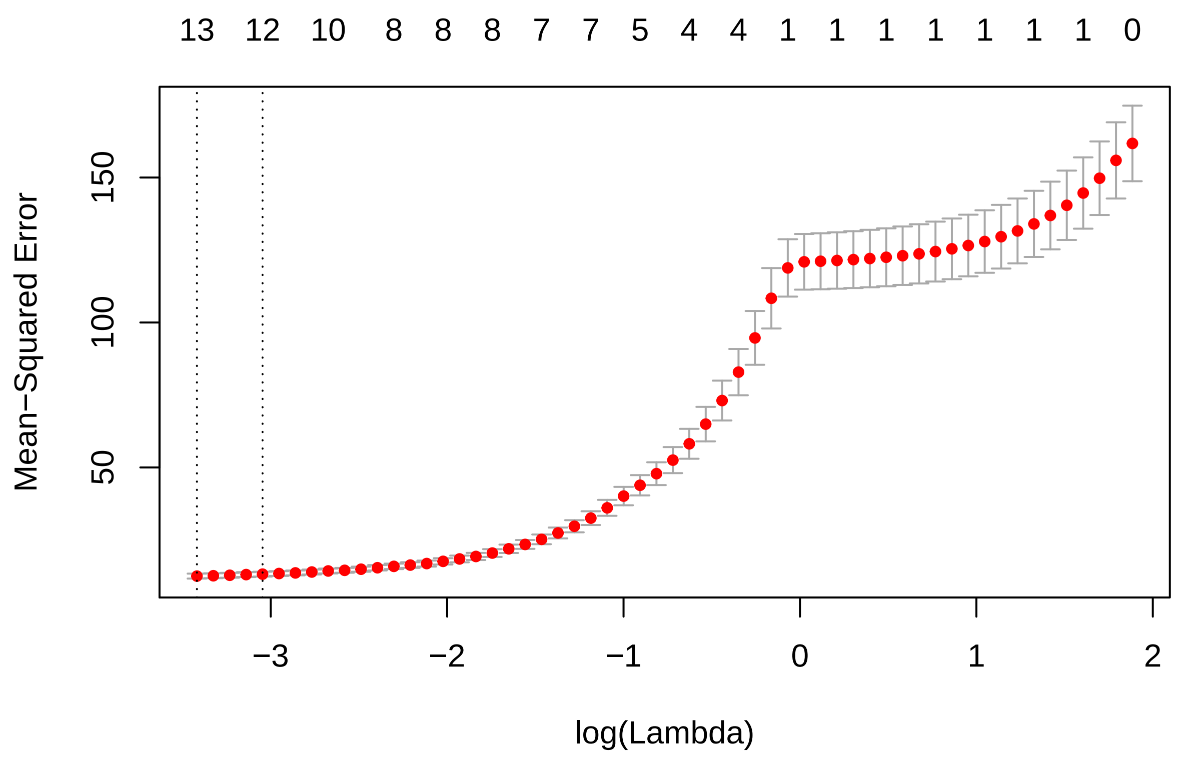
As shown by the results with increase in log lambda, the model coefficients converges to 0 that is for higher values of lambda, the model coefficient shrinks. We can observe the number of coefficients stay high over all the lambdas but the penalty factor causes the size of coefficients to decrease.

Part 6



In the comparison of Lasso plot with the Ridge Regression in ridge regression all coefficients are selected whereas the Lasso do not select all coefficients. When comparing Lasso with the ridge regression, in the ridge regression all model coefficients are selected while in the Lasso plot not all model coefficients are selected. In Ridge regression the model do not converges to zero at lambda equal to zero and in Lasso plot the model coefficient are near to zero.

Part 7



The graph it can be observed as the MSE increases exponentially upto log of 0. After this the error rate increases exponentially again. We can also observe variance increases with the lambda.

Part 8

In comparision with step aic model the cross validation using lasso and k-fold resulted in much less coefficients. The selection of variable in step 4 is based on the AIC value selecting 64 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(MASS)
library(readxl)
library(Matrix)
library(glmnet)

# 4.1

data = read_excel("tecator.xlsx")
plot(data$Protein,data$Moisture)
```

```r
# 4.3

set.seed(12345)
n = nrow(data)
train_indexes = sample(1:n,floor(n*0.5))
train_data = data[train_indexes,]
test_data = data[-train_indexes,]
power = 6
train_error = matrix(0,power,1)
test_error = matrix(0,power,1)

for(i in 1:power) {
  model = lm(Moisture ~ poly(Protein,i), data=train_data)
  train_predictions = predict(model,train_data)
  test_predictions = predict(model,test_data)

  train_error[i,] = mean((train_data$Moisture - train_predictions)^2)
  test_error[i,] = mean((test_data$Moisture - test_predictions)^2)
}

ylim = c(min(rbind(train_error,test_error)),max(rbind(train_error,test_error)))
plot(1:power,train_error, col="Green", ylim=ylim)
lines(1:power,train_error, col="Green")
points(1:power,test_error,col="Red")
lines(1:power,test_error, col="Red")

# 4.4
model = lm(Fat ~ . - Protein - Moisture - Sample, data=data)
steps = stepAIC(model,direction="both", trace=FALSE)
coeff_aics = steps$coefficients
n_coeff_aics = length(coeff_aics)
print(n_coeff_aics)

# 4.5
data_y = data$Fat

data = data[,-which(colnames(data) == "Sample")]
data = data[,-which(colnames(data) == "Fat")]
data = data[,-which(colnames(data) == "Protein")]
data = as.matrix(data[,-which(colnames(data) == "Moisture")])

ridge = glmnet(x=data, y=data_y, alpha=0)
plot(ridge, xvar="lambda")

# 4.6
lasso = glmnet(x=data, y=data_y, alpha=1)
plot(lasso, xvar="lambda")

# 4.7

lasso_cv = cv.glmnet(data,data_y, alpha=1)
lasso_cv_lambda = min(lasso_cv$lambda)
n_lass_cv = sum(coef(lasso_cv) != 0)
```

```r
print(lasso_cv_lambda)
print(n_lass_cv)
plot(lasso_cv)
```