Computer lab 1 block 1

Biswas Kumar (bisku859)

12/31/2019

Assignment 1. Spam classification with nearest neighbors

The data file spambase.xlsx contains information about the frequency of various words, characters etc for a total of 2740 e-mails. Furthermore, these e-mails have been manually classified as spams (spam = 1) or regular e-mails (spam = 0). # 1. Import the data into R and divide it into training and test sets (50%/50%) by using the given code:

Solution : As per instruction, only the first half of data need to be considered and treated as new data set.

```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-uniform
## 'Rounding' sampler used
```

2. Use logistic regression (functions glm(), predict()) to classify the training and test data following by the classification principle :

$\hat{x} = 1$ if p(Y=1|X)>0.5, otherwise $\hat{x}$ =0 and report the confusion matrices (use table()) and the misclassification rates for training and test data. Analyse the obtained results.

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## confusion Matrix Train    Misclassification rate  0.1766423
```

```
##
##     FALSE TRUE
##   0   438   31
##   1    90  126
```

```
## confusion Matrix Test     Misclassification rate  0.2131387
```

```
##
##     FALSE TRUE
##   0   430   43
##   1   103  109
```

As evident from the output above, the misclassification rate was 17.6% with the training data while the same has been increased at 21.3 % with the test data. Its therefore training are marinally better classified than testing data, which should have been in ideal circumstances.

3 Use logistic regression to classify the test data by the classification principle

$\hat{x} = 1$ if p(Y=1|X)>0.8, otherwise $\hat{x}$ nd report the confusion matrices (use table()) and the misclassification rates for training and test data. Compare the results. What effect did the new rule have?

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## confusion Matrix Train    Misclassification rate  0.189781
```

```
##
##     FALSE TRUE
##   0   450   19
##   1   111  105
```

```
## confusion Matrix Test     Misclassification rate  0.2321168
```

```
##
##     FALSE TRUE
##   0   447   26
##   1   133   79
```

On comparision term, it is understood that we have raised the thresholding from 50% to 80 % . This increase in thresholding has resulted in higher misclassification rate both in terms of prediction on train and test data.

It signifies that the thresholding @ 50% was better than thresholding @ 80% for the given data sets and predictive model.

4. Use standard classifier kknn() with K=30 from package kknn, report the the misclassification rates for the training and test data and compare the results with step 2.

```
##
## confusion Matrix Train    Misclassification rate  0.4583942
```

```
##    fitted
##       0   1
##   0 303 166
##   1 148  68
```

```
##
## confusion Matrix Test     Misclassification rate  0.3416058
```

```
##    fitted
##       0   1
##   0 345 128
##   1 106 106
```

The misclassification of the the train and training data are too high (45.8% training & 34.1 % testing data sets)using kknn model (k=30) when compared with model in step 2 (17.6 % & 21.3 % respectively).

It is therefore, kknn is not a good model for the data sets.

5.Repeat step 4 for K=1 and compare the results with step 4. What effect does the decrease of K lead to and why?

```
##
## confusion Matrix Train    Misclassification rate  0.4729927

##    fitted
##       0   1
##   0 275 194
##   1 130  86

##
## confusion Matrix Train    Misclassification rate  0.3912409

##    fitted
##       0   1
##   0 305 168
##   1 100 112
```

At k=1, we have misclasification rate at 47.2% & 39.1% , which is higher when compraed with the results of K=30 in step 4 (45.8% & 34.1 % respectively). This illustrates that K=1 has even worse misclassification when compared with k=30.

Assignment 3. Feature selection by cross-validation in a linear model.

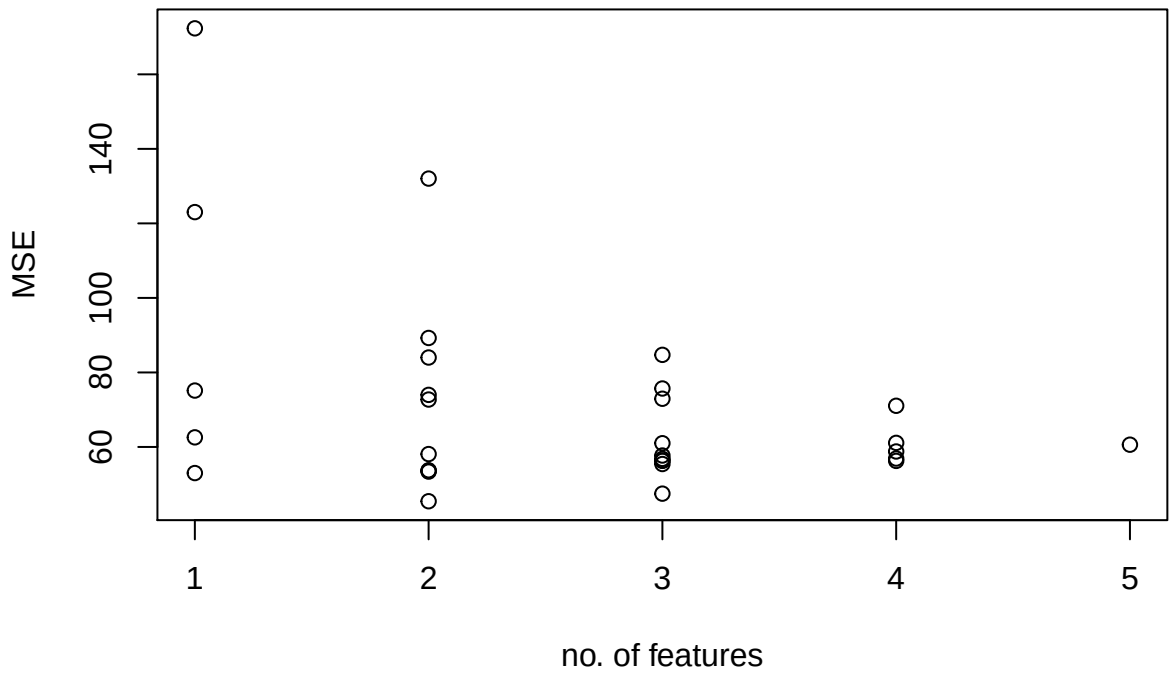
1. Implement an R function that performs feature selection (best subset selection) in linear regression by using k-fold cross-validation without using any specialized function like lm() (use only basic R functions). Your function should depend on:

• * matrix containing X measurements • * vector containing Y measurements • *folds: number of folds in the cross-validation. You may assume in your code that matrix X has 5 columns. The function should plot the CV scores computed for various feature subsets against the number of features, and it should also return the optimal subset of features and the corresponding cross-validation (CV) score. Before splitting into folds, the data should be permuted, and the seed 12345 should be used for that purpose.

2. Test your function on data set swiss available in the standard R repository:

• *ertility should be Y* • ll other variables should be X • *folds should be 5 Report the resulting plot and interpret it. Report the optimal subset of features and comment whether it is reasonable that these specific features have largest impact on the target.

```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-uniform
## 'Rounding' sampler used
```




```
## $CV
## [1] 45.43318
##
## $Features
## [1] 0 1 0 1 0
```

Looking at the plot, we can say that least MSE value is obtained when number of features are 2 and CV score is 45.43.

The optimal feature subset is given by (0 1 0 1 0) which can be expressed by below regression formula:

$$Fertility = \beta_0 + \beta_1 Examination + \beta_2 Catholic$$

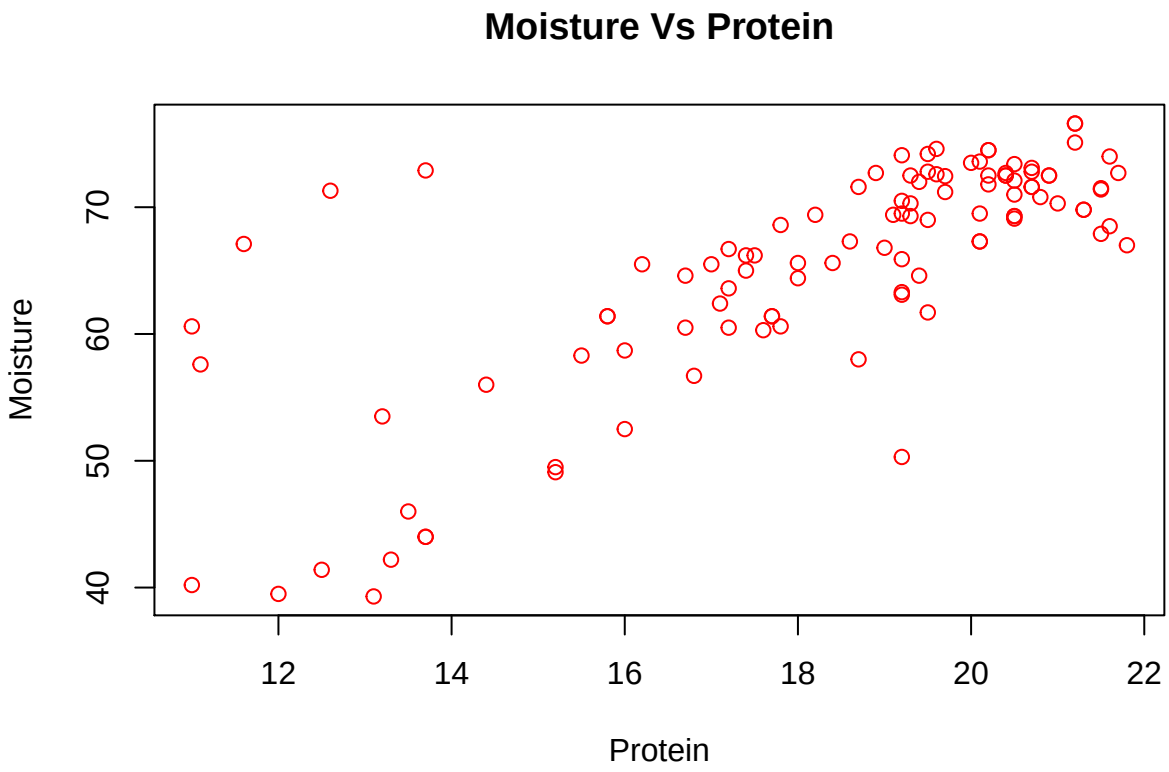
It means that the model highlights that Examination & catholic/religion is a major feature for fertility. In my opinion, "Examination" can not be a major feature when other feature like "Infant.Mortality" is not considered. It does not look like that these highlighted features shall have any major impact on the target.

Assignment 4. Linear regression and regularization

The Excel file tecator.xlsx contains the results of study aimed to investigate whether a near infrared absorbance spectrum can be used to predict the fat content of samples of meat. For each meat sample the data consists of a 100 channel spectrum of absorbance records and the levels of moisture (water), fat and protein. The absorbance is -log10 of the transmittance measured by the spectrometer. The moisture, fat and protein are determined by analytic chemistry.

1.

Import data to R and create a plot of Moisture versus Protein. Do you think that these data are described well by a linear model?



Looking at the plot, data can be described by the liner model with some outliars as few data points are far away and differs significantly from observations.

2.

Consider model Mi in which Moisture is normally distributed, and the expected Moisture is a polynomial function of Protein including the polynomial terms up to power i (i.e M1 is a linear model, M2 is a quadratic model and so on). Report a probabilistic model that describes i. Why is it appropriate to use MSE criterion when fitting this model to a training data?

Response :

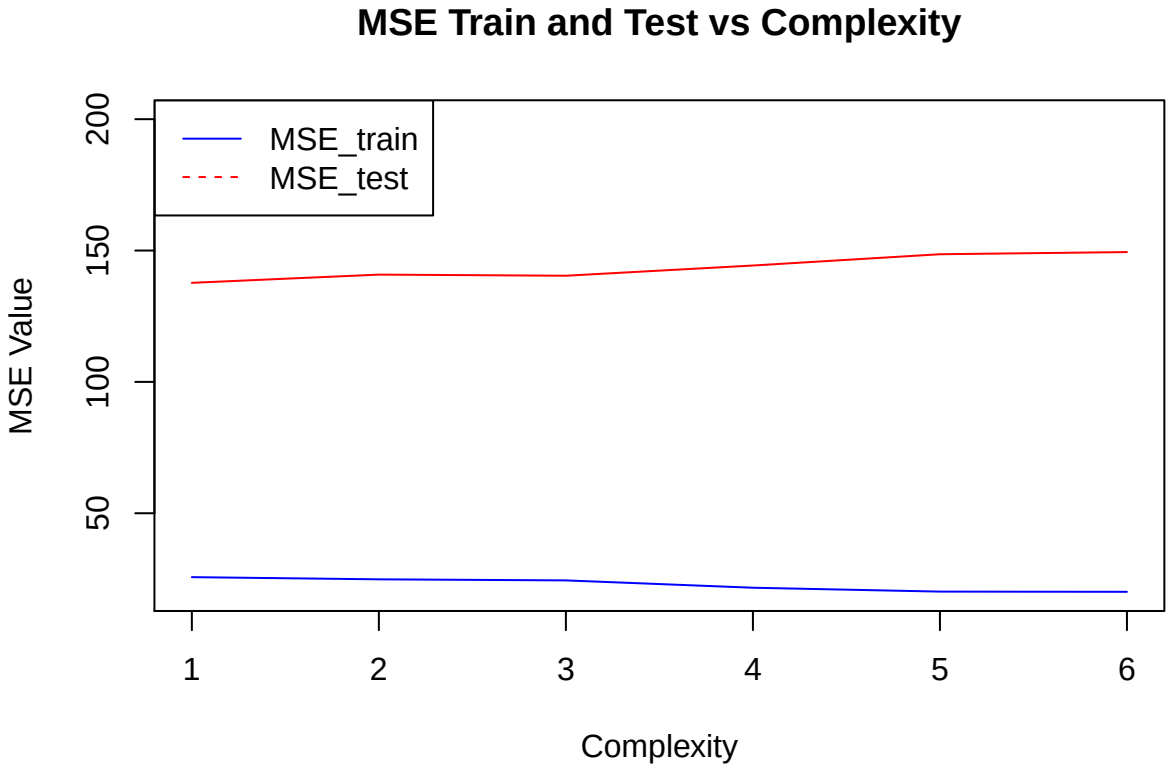
The linear model in which moisture is normally distributed is given below :

$$Moisture_i = \beta_0 + \beta_1 Protein^1 + \beta_2 Protein^2 + \beta_3 Protein^3 + \beta_4 Protein^4 + \beta_5 Protein^5 + \beta_6 Protein^6$$

MSE criteria (Mean Square Error) for a polynomial equation or model is ideal to judge weather the model is overfit or underfit . It is appropriate criteria as ploynomial models tends to overfit the training data set and hence MSE evaluation criteria gives us the much required insight.

3.

Divide the data into training and validation sets(50%/50%) and fit models. For each model, record the training and the validation MSE and present a plot showing how training and validation MSE depend on i (write some R code to make this plot). Which model is best according to the plot? How do the MSE values change and why? Interpret this picture in terms of bias-variance tradeoff. Use the entire data set in the following computations:



Looking at the plot, we observe that the training data has less MSE value and it keeps on declining with increase in complexity. It represents the ideal secenario as polynomial equaltion overfit the traing data with increasing order of complexity. However, the same is not true with test data. The increase in complexity can also increase with test data , as the model was too much overfitted with the training data. In other way, we can say that overfitted model with high complexity gives rise to high variance and low bias.

On observation , we see that the optimum complexity is 3 ,when both train and test MSE value has the minimum difference between them.

4.

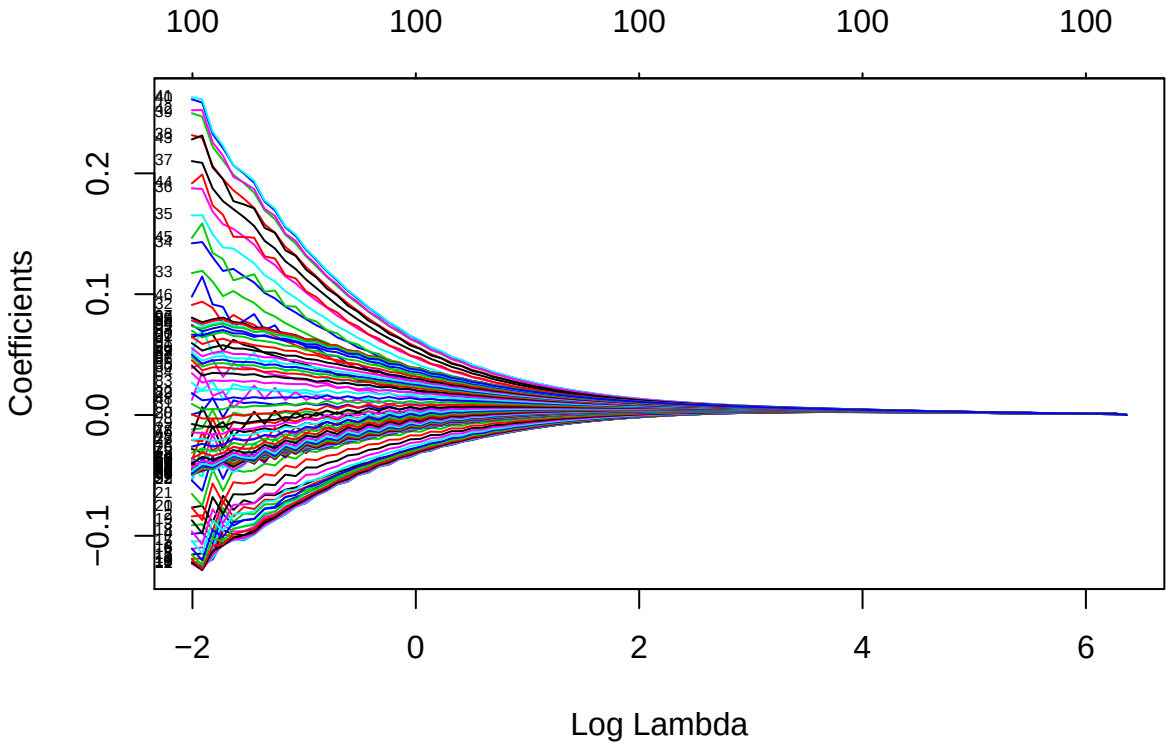
Perform variable selection of a linear model in which Fat is response and Channel1-Channel100 are predictors by using stepAIC. Comment on how many variables were selected.

```
## Overall, 95 variables were selected
```

As we know that one of the selected variable is intercept, hence the selected variable should be 94.

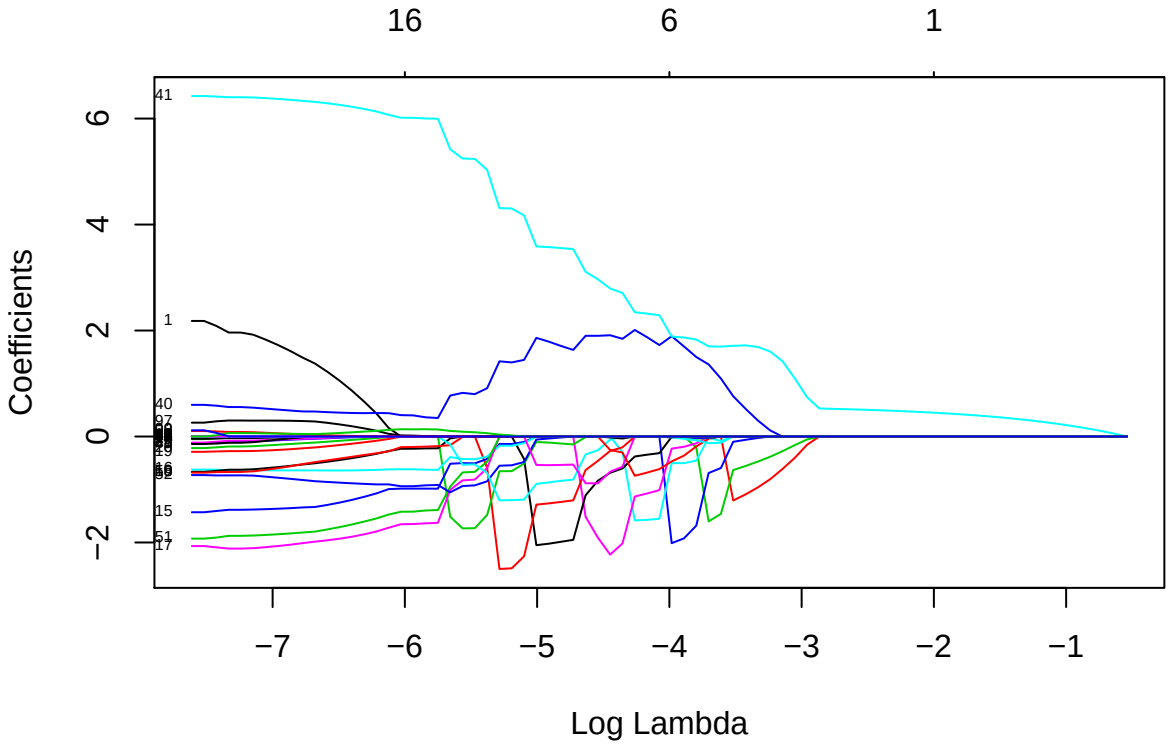
5.

Fit a Ridge regression model with the same predictor and response variables. Present a plot showing how model coefficients depend on the log of the penalty factor lambda and report how the coefficients change with lambda



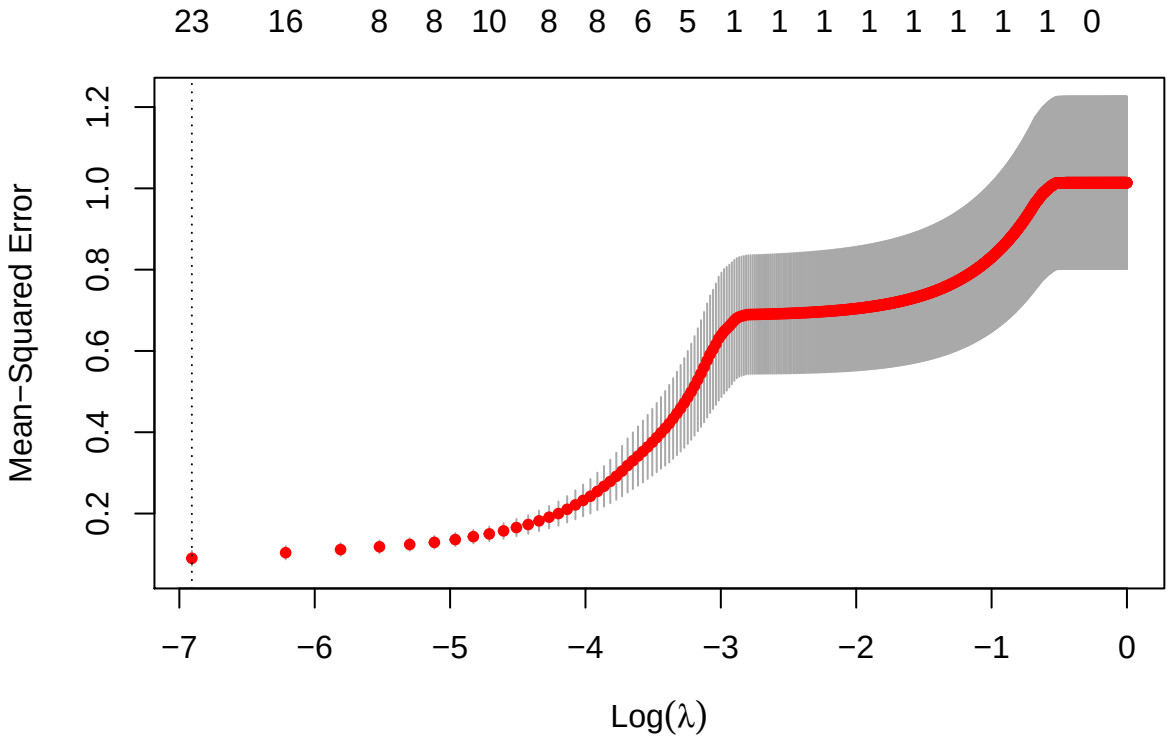
Looking at the plot, we can say that with increase of lambda, the coefficient decreases dramatically. It is interesting to note that these coefficient does not goes to zero because penalty factor stay associated with the parameters.In short, Ridge does not eliminate the parametrs. These fit are better when we have all important parameters in the data set.

6. Repeat step 5 but fit LASSO instead of the Ridge regression and compare the plots from steps 5 and 6. Conclusions?



Lasso fit drops the parametrs that it does not find significant and goes to zero when compared with ridge fit. Lasso fit is better when we have some irrelevant features in data set which can be omitted to give better predictive model.

7. Use cross-validation to find the optimal LASSO model (make sure that case lambda=0 is also considered by the procedure) , report the optimal lambda and how many variables were chosen by the model and make conclusions. Present also a plot showing the dependence of the CV score and comment how the CV score changes with lambda.




```
##
## Call:  cv.glmnet(x = as.matrix(covariates), y = response, lambda = seq(0,      1, 0.001), alpha = 1,
##
## Measure: Mean-Squared Error
##
##     Lambda Measure      SE Nonzero
## min  0.000 0.08284 0.02061     100
## 1se  0.001 0.08991 0.01825      23

## the optimal lambda = 0
```

It can be observed from the plot that MSE value rises with increase in lambda.

8. Compare the results from steps 4 and 7.

The CV optimum lasso model signifies that no variable has been dropped from the model. On the other hand, step 4 shows that there were 5 variables (i.e 100- 95), that were dropped from the linear model.

Although, the result of step 7 seems better than 4, it is wise to go ahead with step AIC model (with ony 5 variables elimination)as Lasso has high chance to overfit the model.

Code Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(readxl)
library(kknn)
library(MASS)
library(glmnet)
library(dvmisc)
library(ggplot2)
data=read_excel("spambase.xlsx")
d<-dim(data)
new_data<-data[1:(d[1]/2),] # first half of data
#dim(new_data) # 1370 rows & 49 columns
data<-new_data # saving new_data as data again
# from the lecture slide
n=dim(data)[1]
RNGversion('3.5.1') # advised to be used alongwith set seed
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,] # train data
test=data[-id,] # test data
#lecture slide of lm function
#model on  training the model
fit=glm((formula = Spam~.) , data = train,family ="binomial") # binomial family for spam data
fitted= predict(fit,newdata = train)
#thresholding 50%, and preparing table for confusion matrix
table=table(train$Spam, fitted > 0.5) # table function used , and fitted >0.5 for spam data
TP=table[2,2];TN=table[1,1];FN=table[2,1];FP=table[1,2]
#print confusion table and misclassification rate
cat("\nconfusion Matrix Train ","\t Misclassification rate ",(FP + FN)/sum(table))
table

#model on  test data
fitted= predict(fit,newdata = test)
#thresholding 50%, and preparing table for confusion matrix
table=table(test$Spam, fitted > 0.5) # table function used , and fitted >0.5 for spam data
TP=table[2,2];TN=table[1,1];FN=table[2,1];FP=table[1,2]
#print confusion table and misclassification rate
cat("confusion Matrix Test ","\t Misclassification rate ",(FP + FN)/sum(table))
table

#the new thresholding limit has been set at 80%
#On training data
fit=glm((formula = Spam~.) , data = train,family ="binomial") # binomial family for spam data
fitted= predict(fit,newdata = train)
#thresholding 80%, and preparing table for confusion matrix
table=table(train$Spam, fitted > 0.8) # table function used , and fitted >0.5 for spam data
TP=table[2,2];TN=table[1,1];FN=table[2,1];FP=table[1,2]
```

```r
#print confusion table and misclassification rate
cat("\nconfusion Matrix Train ","\t Misclassification rate ",(FP + FN)/sum(table))
table


# on test data
fitted= predict(fit,newdata = test)
#thresholding 80%, and preparing table for confusion matrix
table=table(test$Spam, fitted > 0.8) # table function used , and fitted >0.5 for spam data
TP=table[2,2];TN=table[1,1];FN=table[2,1];FP=table[1,2]
#print confusion table and misclassification rate
cat("confusion Matrix Test ","\t Misclassification rate ",(FP + FN)/sum(table))
table

#k=30

fit<-kknn(formula=as.factor(Spam)~.,train,test,k=30)
fitted=predict(fit,newdata = train)
# preparing table for confusion matrix
table=table(train$Spam, fitted) # table function used , and fitted >0.5 for spam data
TP=table[2,2];TN=table[1,1];FN=table[2,1];FP=table[1,2]
#print confusion table and misclassification rate
cat("\nconfusion Matrix Train ","\t Misclassification rate ",(FP + FN)/sum(table))
table

#On test data
fitted=predict(fit,newdata = test)
# preparing table for confusion matrix
table=table(test$Spam, fitted) # table function used , and fitted >0.5 for spam data
TP=table[2,2];TN=table[1,1];FN=table[2,1];FP=table[1,2]
#print confusion table and misclassification rate
cat("\nconfusion Matrix Test ","\t Misclassification rate ",(FP + FN)/sum(table))
table

#k=30

fit<-kknn(formula=as.factor(Spam)~.,train,test,k=1)
fitted=predict(fit,newdata = train)
# preparing table for confusion matrix
table=table(train$Spam, fitted) # table function used , and fitted >0.5 for spam data
TP=table[2,2];TN=table[1,1];FN=table[2,1];FP=table[1,2]
#print confusion table and misclassification rate
cat("\nconfusion Matrix Train ","\t Misclassification rate ",(FP + FN)/sum(table))
table

#On test data
fitted=predict(fit,newdata = test)
# preparing table for confusion matrix
table=table(test$Spam, fitted) # table function used , and fitted >0.5 for spam data
TP=table[2,2];TN=table[1,1];FN=table[2,1];FP=table[1,2]
#print confusion table and misclassification rate
cat("\nconfusion Matrix Train ","\t Misclassification rate ",(FP + FN)/sum(table))
table
```

```r
#using help submitted under assignment document for skeleton
data=swiss # 47 rows
data<-data[1:25,] # first half of data ( have taken first 25 rows)
RNGversion('3.5.1') # advised to be used alongwith set seed
set.seed(12345)
mylin=function(X,Y, Xpred){
  Xpred1=cbind(1,Xpred)
  X1=cbind(1,X)
  beta=solve(t(X1)%*%X1)%*%t(X1)%*%Y # formula for linear regression
  Res=Xpred1%*%beta # as per the skeleton
  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

            for (k in 1:Nfolds){

              # divided into 5 parts for the 25 rows and using the each one segment for testing one by
               ind_col=which(model==1)
               ind_row<-c()
               if(k==1){ind_row<-1:5}
               else if(k==2){ind_row<-6:10}
               else if(k==3){ind_row<-11:15}
               else if(k==4){ind_row<-16:20}
               else if(k==5){ind_row<-21:25}
               # Ypred, Yp
               Xpred=X1[ind_row,ind_col]
               X=X1[-ind_row,ind_col]
               Y=Y1[-ind_row]
               Ypred= mylin(X,Y, Xpred)
               Yp=Y1[ind_row]
```

```
            SSE=SSE+sum((Ypred-Yp)^2)
          }
          curr=curr+1
          MSE[curr]=SSE/n
          Nfeat[curr]=sum(model)
          Features[[curr]]=model

        }

  # plot MSE against number of features
  plot(Nfeat,MSE,xlab="no. of features")
 i=which.min(MSE)
  return(list(CV=MSE[i], Features=Features[[i]]))
}

myCV(as.matrix(data[,2:6]), data[[1]],5) # data for first 25 values of swiss

data=read_excel("tecator.xlsx")
d<-dim(data)
new_data<-data[1:(d[1]/2),] # first half of data as per  instruction
data<-new_data # saving new_data as data again
plot(x=data$Protein,y=data$Moisture,type ='p',xlab ="Protein",ylab = "Moisture",main="Moisture Vs Protei

# as per the lab slides, dividing data sets in 50% :50%.
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
test=data[-id,]

# GLM function training data
M1=glm(formula =Moisture~Protein,data=train)
M2=glm(formula =Moisture~Protein+I(Protein^2),data =train)
M3=glm(formula =Moisture~Protein+I(Protein^2)+I(Protein^3),data =train)
M4=glm(formula =Moisture~Protein+I(Protein^2)+I(Protein^3)+I(Protein^4),data =train)
M5=glm(formula =Moisture~Protein+I(Protein^2)+I(Protein^3)+I(Protein^4)+I(Protein^5),data =train)
M6=glm(formula =Moisture~Protein+I(Protein^2)+I(Protein^3)+I(Protein^4)+I(Protein^5)+I(Protein^6),data =

D1=predict(M1,data=train)
D2=predict(M2,data=train)
D3=predict(M3,data=train)
D4=predict(M4,data=train)
D5=predict(M5,data=train)
D6=predict(M6,data=train)

#Prediction on testing data
T1=predict(M1,data=test)
T2=predict(M2,data=test)
T3=predict(M3,data=test)
T4=predict(M4,data=test)
T5=predict(M5,data=test)
T6=predict(M6,data=test)
```

```r
#MSE calculation
 MSE_D1= sum((D1-train$Moisture)^2)/length(train$Moisture)
 MSE_D2= sum((D2-train$Moisture)^2)/length(train$Moisture)
 MSE_D3= sum((D3-train$Moisture)^2)/length(train$Moisture)
 MSE_D4= sum((D4-train$Moisture)^2)/length(train$Moisture)
 MSE_D5= sum((D5-train$Moisture)^2)/length(train$Moisture)
 MSE_D6= sum((D6-train$Moisture)^2)/length(train$Moisture)

 #combining MSE training data sets
 MSE_train<-c(MSE_D1,MSE_D2,MSE_D3,MSE_D4,MSE_D5,MSE_D6)
 MSE_T1= sum((T1-test$Moisture)^2)/length(test$Moisture)
 MSE_T2= sum((T2-test$Moisture)^2)/length(test$Moisture)
 MSE_T3= sum((T3-test$Moisture)^2)/length(test$Moisture)
 MSE_T4= sum((T4-test$Moisture)^2)/length(test$Moisture)
 MSE_T5= sum((T5-test$Moisture)^2)/length(test$Moisture)
 MSE_T6= sum((T6-test$Moisture)^2)/length(test$Moisture)
 #combining MSE testing data sets
 MSE_test=c(MSE_T1,MSE_T2,MSE_T3,MSE_T4,MSE_T5,MSE_T6)
 complexity<-1:6
#plotting
plot(x=complexity,y=MSE_train,type ="l",col="blue",ylab="MSE Value",xlab="Complexity",ylim=c(20,200),ma
lines(x=complexity,y=MSE_test,col="red",type = "l")
legend("topleft", c("MSE_train", "MSE_test"),col = c("blue", "red"), lty = c(1, 2))

#In reference to lecture slide
data=read_excel("tecator.xlsx")
d<-dim(data)
new_data=-data[1:(d[1]/2),] # first half of data as per  instruction
data<-new_data
train_data= data[-c(1, 103, 104)] # rejecting the data not required for training
table <- cbind(new_data,data$Fat)
fit <- lm(Fat ~ . , data=train_data) # fat is the response
step <- stepAIC(fit, direction ="both",trace=F)
variables_list <- step$coefficients
cat("Overall,", length(variables_list), "variables were selected \n")

covariates <- scale(train_data[,1:100])
response <- scale(train_data$Fat)
fit <- glmnet(covariates, response, alpha = 0, family = "gaussian")
plot(fit, xvar="lambda", label=TRUE)

fit_lasso<- glmnet(covariates, response, alpha = 1, family = "gaussian") # use of glmnet with alpha =1
plot(fit_lasso, xvar="lambda", label=T)
# as refererred in lecture slide ,include 0
lambda_updated<- append(fit_lasso$lambda,0)
model=cv.glmnet(as.matrix(covariates), response, alpha=1,family="gaussian",lambda=seq(0,1,0.001))
plot(model)
model
cat("the optimal lambda =", model$lambda.min )
```