MachineLearning_Lab01

Sreenand.S

24/11/2019

Assignment 1 :Spam classification with nearest neighbors

1.1

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

## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

Spambase datafile is imported and the code is divided into 2 chunks of testing and training data which are 50:50.

1.2

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

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

##
## y_pred   0   1
##      0 803  81
##      1 142 344

## [1] 0.1627737

##
## y_pred_test   0   1
##    Not Spam 791  97
##    Spam     146 336

## [1] 0.1773723
```

p(Y = 1|X)>0.5

The model is fitted with logistic regression for the training data and the misclassifier is observed to be 16.28% The model is fitted with logistic regression for the test data and the misclassifier is observed to be 17.90%. It is observed that the misclassifier for the test data is more than that of the training data.This is due to the fact that classification and test of data is done on the same data.

1.3

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

##
## y_pred2   0   1
##       0 941 335
##       1   4  90

## [1] 0.2474453
```

```
##
## y_pred_test2   0   1
##     Not Spam 926 367
##     Spam      11  66
```

```
## [1] 0.2759124
```

$p(Y = 1|X)>0.8$

The model is fitted with logistic regression and the misclassification for the trainig data is observed to be 24.74% The model is fitted with logistic regression and the misclassification for the test data is observed to be 27.59% . It is observed that the training data has lesser miscalculation rate as compared to when the model is fitted with the test data and it os collectively higher than when the probability of prediction is set to 0.5.

1.4

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

```
##
## y_pred_train_kknn   0   1
##          Not Spam 698  32
##          Spam     247 393
```

```
## [1] 0.2036496
```

```
##
## y_pred_test_kknn   0   1
##         Not Spam 568  81
##         Spam     369 352
```

```
## [1] 0.3284672
```

When K=30

When the KNN Classifier is used it is observed that the misclassification for the training data is 20.36%. When the KNN Classifier is used it is observed that the misclassification for the test data is 32.84% It is observed that the when tested in the training data it has a lower miscalculation rate as compared to test data when K=30 with KNN model for $p(Y = 1|X)>0.5$.There is a significant difference between the miscalculation rates of the train data and the test data.

From this it is observed that the Logistic is more efficient classifier than the KNN classifier as it has higher accuracy and KNN takes more time to fetch the results.

1.5

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

```
##
## y_pred_train_kknn   0   1
##          Not Spam 945   0
##          Spam       0 425
```

```
## [1] 0
```

```
##
## y_pred_test_kknn   0   1
##         Not Spam 599 149
##         Spam     338 284
```

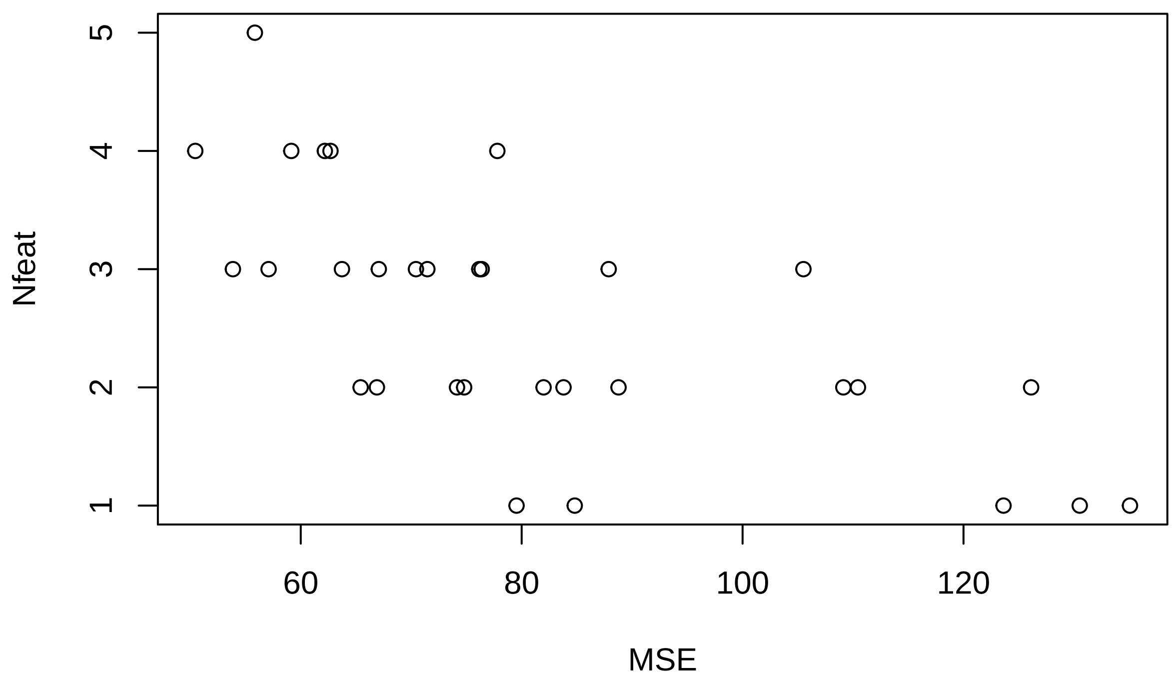
```
## [1] 0.3554745
```

When K=1

When the KNN Classifier is used it is observed that the misclassification for the training data is 0% and the Accuracy is 100%. When the KNN Classifier is used it is observed that the misclassification for the test data is 35%. It is observed that when k=1 there is no miscalculation for p(Y = 1|X)>0.5 for the train data but when used on test data it yields a higher misclassification rate than the logistic regression.Thus we can say that for a higher value of k it yields much better results as compared to lower values and when k=1 it classifies all itss nearest neighbours correctly which is 1.

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

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




```
## $CV
## [1] 50.44948
##
## $Features
## [1] 1 0 1 1 1
```

Here it is observed that the CV score is 50.44% and 1,3,4,5 are the best models that can be used. Therefore these features would have the largest impact.

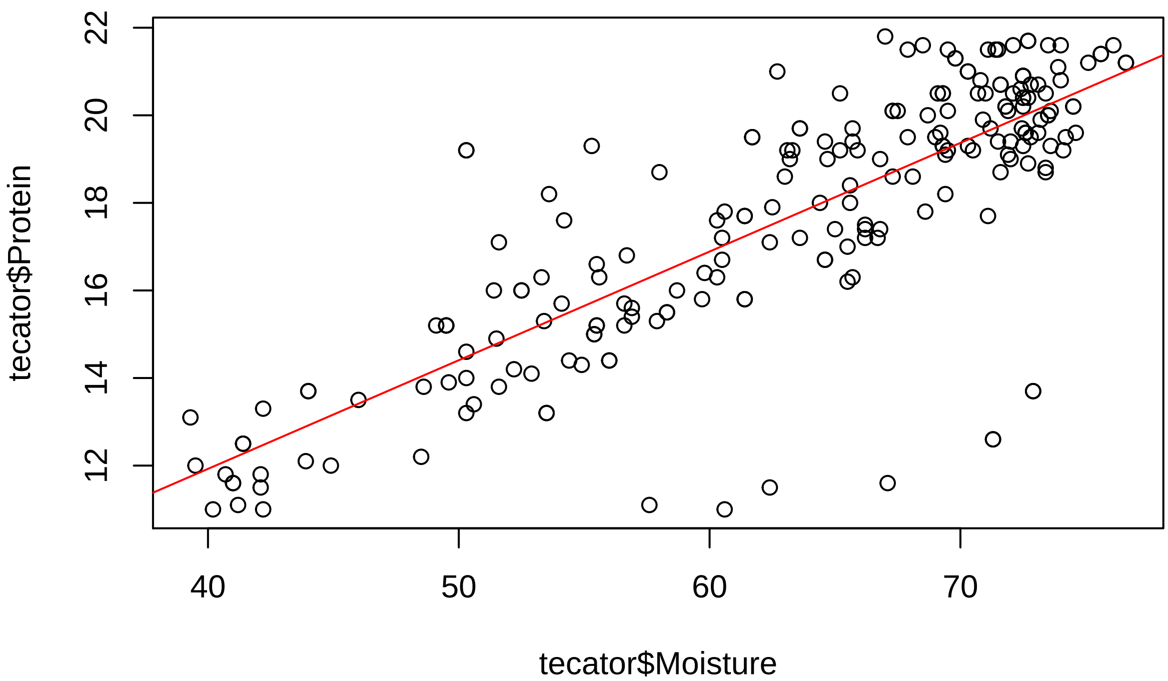
Assignment 4. Linear regression and regularization

4.1 Importing data and plotting moisture vs protein

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

```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-1
```




```
##
## Call:
## lm(formula = tecator$Protein ~ tecator$Moisture)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.0915 -0.7725  0.1228  0.9340  4.7192
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.99987    0.77465   2.582   0.0105 *
## tecator$Moisture  0.24813    0.01211  20.491   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.754 on 213 degrees of freedom
## Multiple R-squared:  0.6634, Adjusted R-squared:  0.6619
```

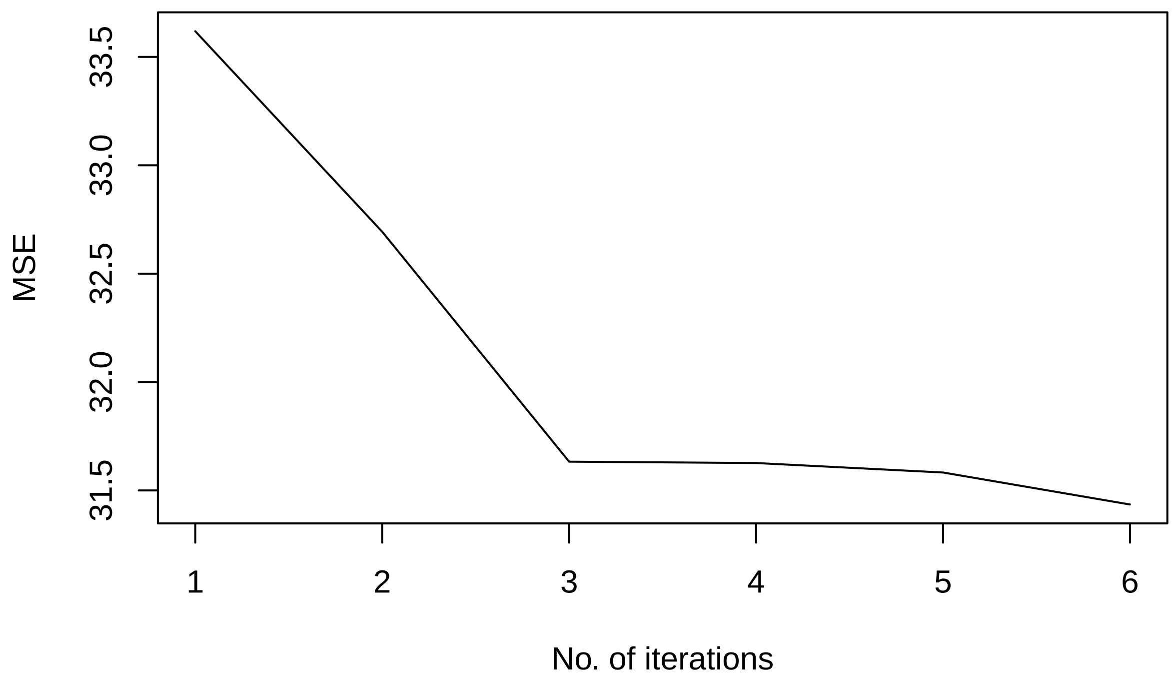
```
## F-statistic: 419.9 on 1 and 213 DF,  p-value: < 2.2e-16
```

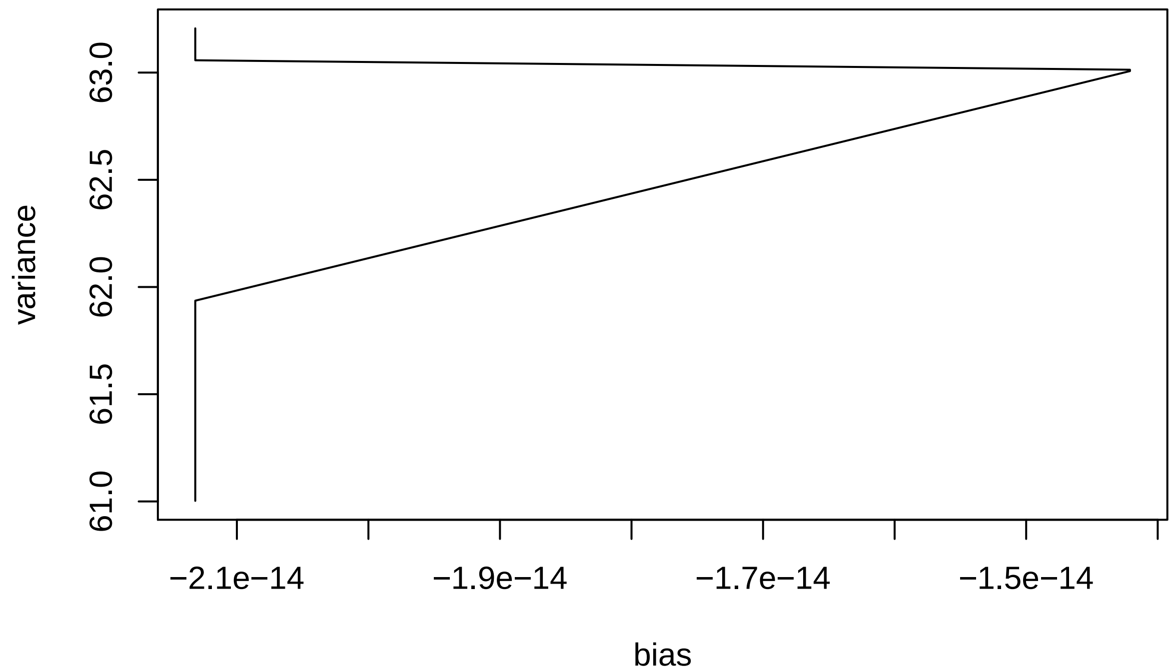
Formula:Y_hat = 1.99987 + 0.24813x

Here it is seen that the correlation is positive and hence this data can be described well by a linear model. Since the p-value is lesser than 0.05 which indicates that moisture is a significant parameter of protein.

4.2

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

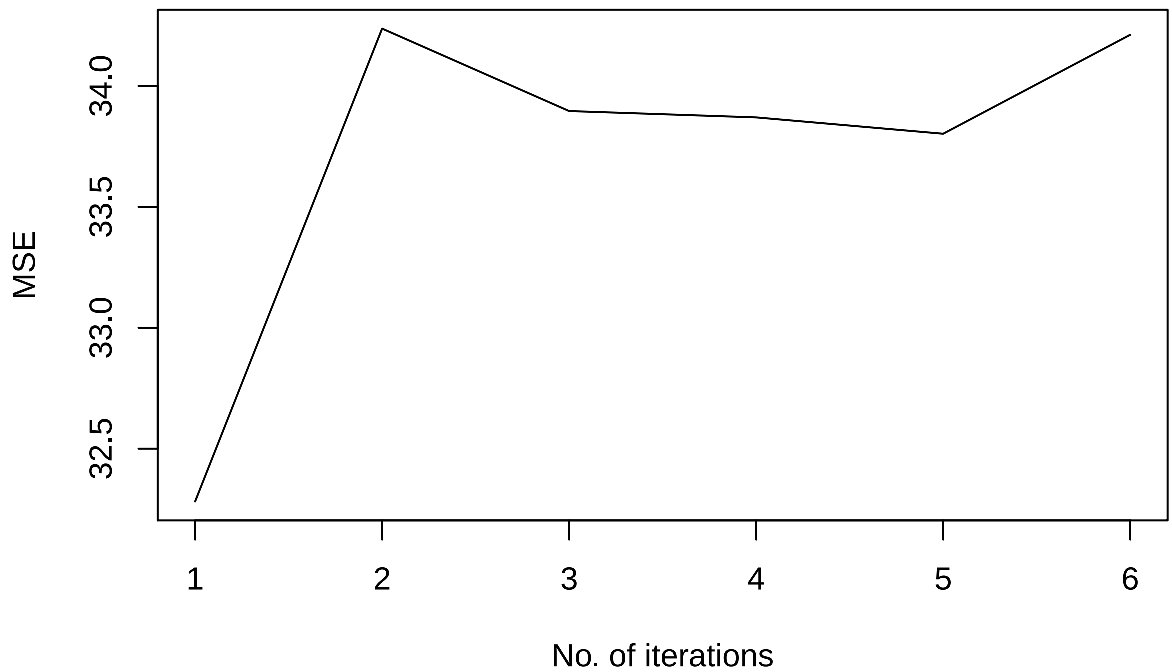


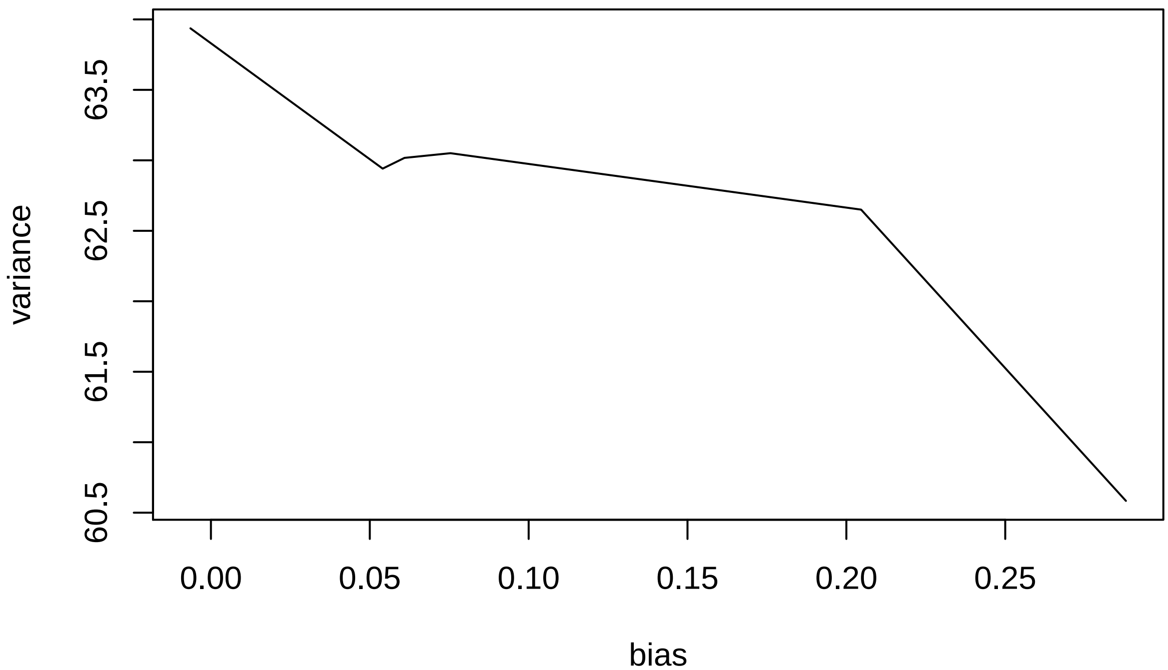


It is observed that as the degree of polynomial increases the MSE criterion decreases .The MSE criterion is the least when the degree of the polynomial is 6.This shows that MSE can be used as criterion when fitting these type of models.

4.3

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


MSE
34.0
33.5
33.0
32.5
1
2
3
4
5
6
No. of iterations



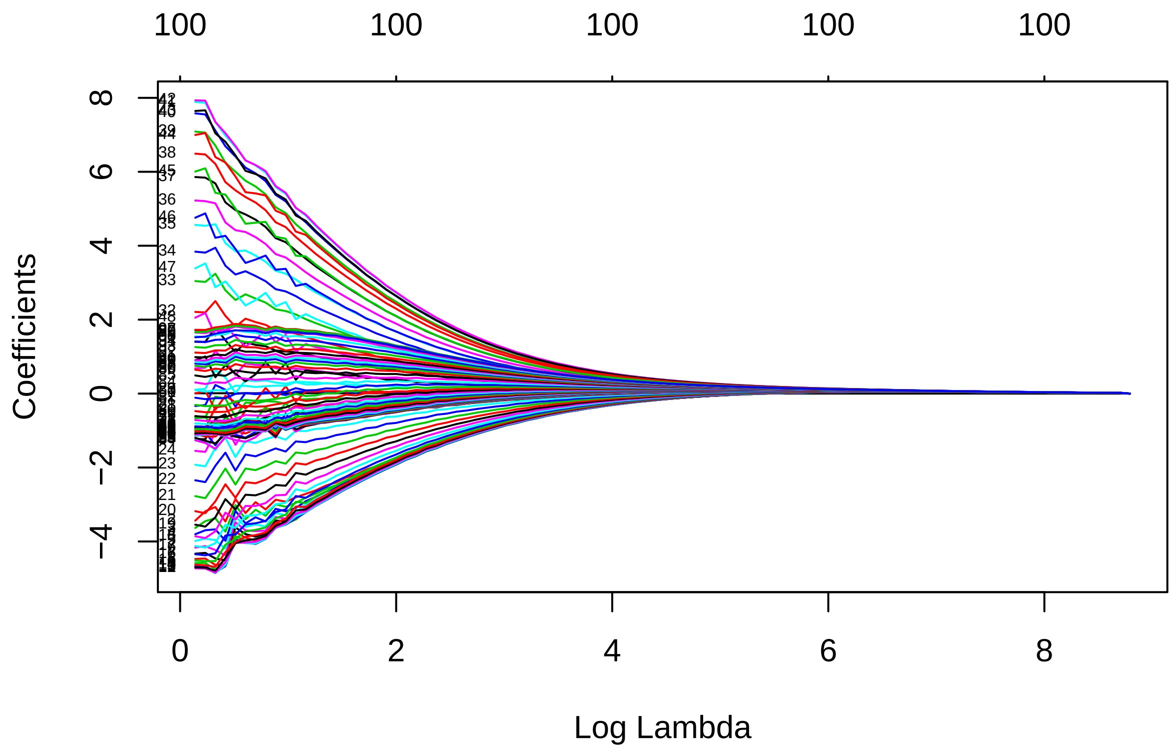
Here it is observe that the lowest value for the validation set is for the polynomial with degree 1.Therefore it is the best mode for the given problem.

4.4

63 parameters have been selected siginficant factor to have a influence on the response variable. After reaching 95 the AIC factor is constant for every trial.

4.5

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

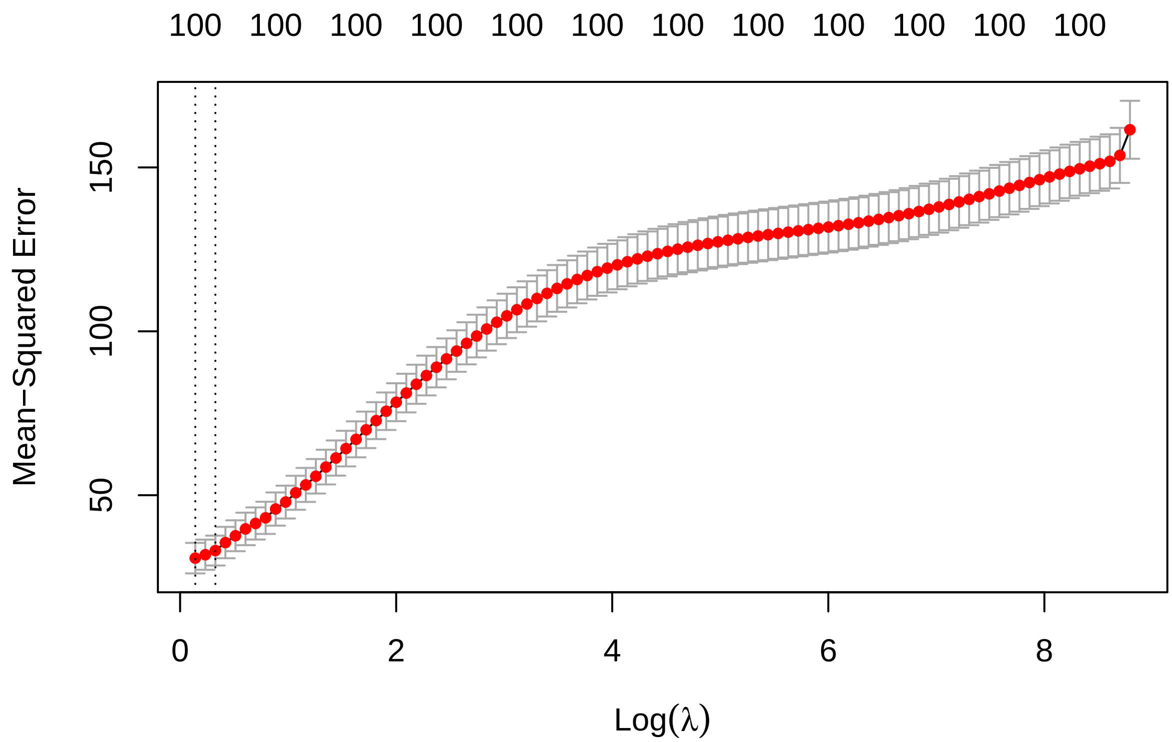



```
## [1] 1.150511

##
## Call:  glmnet(x = X, y = Y, alpha = 0)
##
##      Df    %Dev Lambda
## 1   100 0.00000 6584.0
## 2   100 0.06096 5999.0
## 3   100 0.06530 5466.0
## 4   100 0.06986 4980.0
## 5   100 0.07460 4538.0
## 6   100 0.07950 4135.0
## 7   100 0.08455 3767.0
## 8   100 0.08974 3433.0
## 9   100 0.09503 3128.0
## 10  100 0.10040 2850.0
## 11  100 0.10580 2597.0
## 12  100 0.11130 2366.0
## 13  100 0.11680 2156.0
## 14  100 0.12220 1964.0
## 15  100 0.12760 1790.0
## 16  100 0.13290 1631.0
## 17  100 0.13810 1486.0
## 18  100 0.14320 1354.0
## 19  100 0.14810 1234.0
## 20  100 0.15290 1124.0
## 21  100 0.15740 1024.0
```

```
## 22 100 0.16180  933.2
## 23 100 0.16600  850.3
## 24 100 0.16990  774.8
## 25 100 0.17370  705.9
## 26 100 0.17720  643.2
## 27 100 0.18060  586.1
## 28 100 0.18380  534.0
## 29 100 0.18680  486.6
## 30 100 0.18970  443.4
## 31 100 0.19250  404.0
## 32 100 0.19480  368.1
## 33 100 0.19740  335.4
## 34 100 0.20000  305.6
## 35 100 0.20250  278.4
## 36 100 0.20510  253.7
## 37 100 0.20760  231.2
## 38 100 0.21030  210.6
## 39 100 0.21300  191.9
## 40 100 0.21580  174.9
## 41 100 0.21880  159.3
## 42 100 0.22190  145.2
## 43 100 0.22520  132.3
## 44 100 0.22870  120.5
## 45 100 0.23250  109.8
## 46 100 0.23650  100.1
## 47 100 0.24080   91.2
## 48 100 0.24540   83.1
## 49 100 0.25030   75.7
## 50 100 0.25560   69.0
## 51 100 0.26120   62.8
## 52 100 0.26730   57.3
## 53 100 0.27390   52.2
## 54 100 0.28070   47.5
## 55 100 0.28820   43.3
## 56 100 0.29590   39.5
## 57 100 0.30440   36.0
## 58 100 0.31320   32.8
## 59 100 0.32290   29.9
## 60 100 0.33280   27.2
## 61 100 0.34370   24.8
## 62 100 0.35450   22.6
## 63 100 0.36650   20.6
## 64 100 0.37860   18.8
## 65 100 0.39170   17.1
## 66 100 0.40510   15.6
## 67 100 0.41920   14.2
## 68 100 0.43370   12.9
## 69 100 0.44870   11.8
## 70 100 0.46440   10.7
## 71 100 0.47960    9.8
## 72 100 0.49700    8.9
## 73 100 0.51270    8.1
## 74 100 0.53060    7.4
## 75 100 0.54700    6.7
```

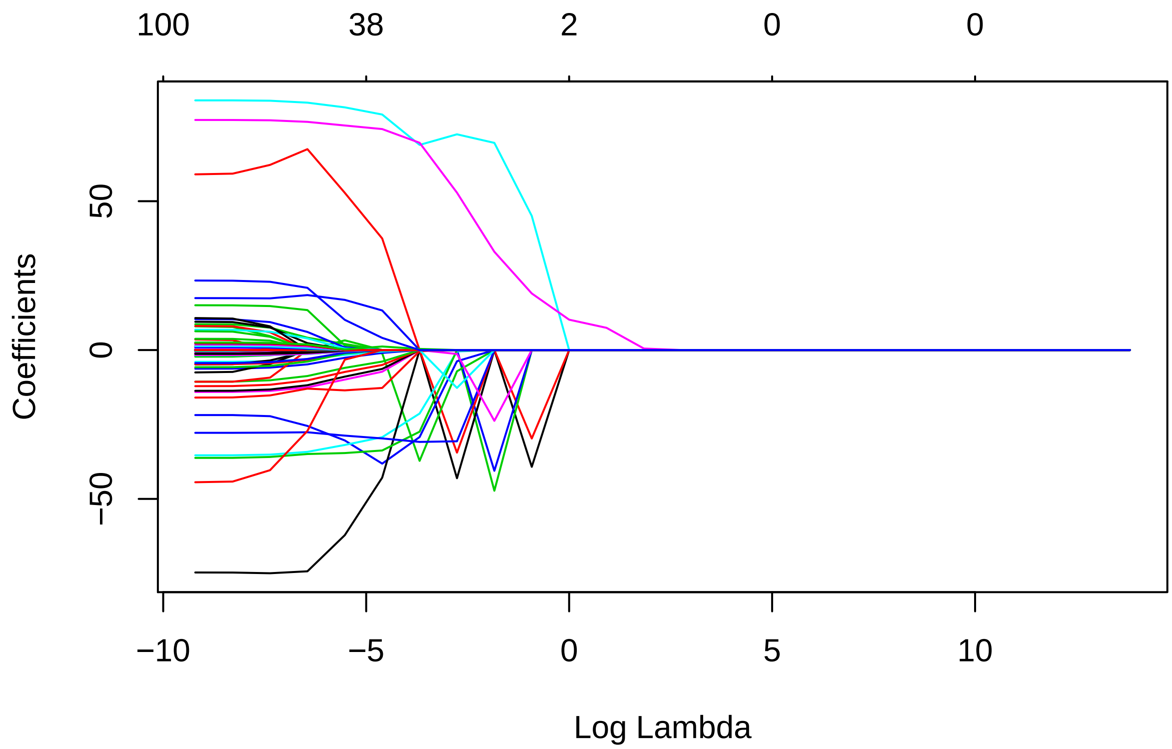
```
## 76 100 0.56460    6.1
## 77 100 0.58190    5.6
## 78 100 0.59940    5.1
## 79 100 0.61670    4.6
## 80 100 0.63400    4.2
## 81 100 0.65130    3.9
## 82 100 0.66760    3.5
## 83 100 0.68500    3.2
## 84 100 0.69580    2.9
## 85 100 0.71820    2.7
## 86 100 0.72570    2.4
## 87 100 0.74810    2.2
## 88 100 0.75500    2.0
## 89 100 0.75970    1.8
## 90 100 0.77140    1.7
## 91 100 0.78990    1.5
## 92 100 0.80520    1.4
## 93 100 0.81800    1.3
## 94 100 0.81770    1.2
```



As log(lambda)/lambda increases, the model coefficients appear to move towards zero but does not reach zero.

4.6

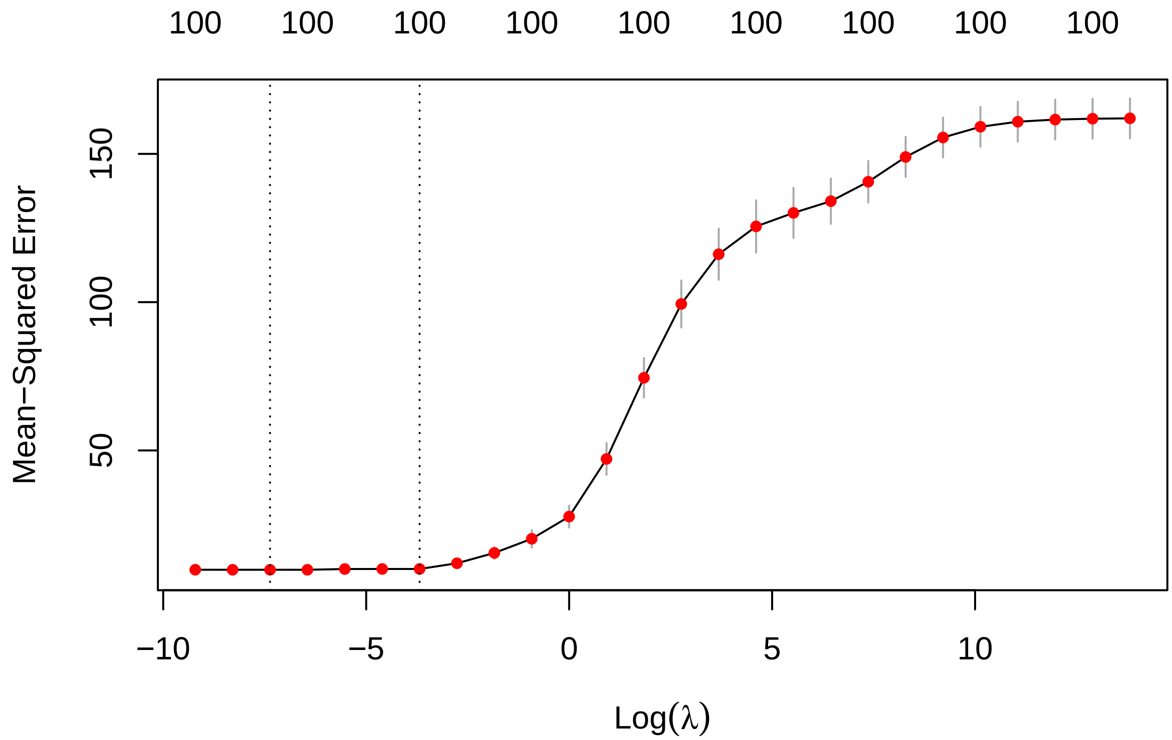
```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-
## uniform 'Rounding' sampler used
```




```
## [1] 0.0006309573

##
## Call:  glmnet(x = X, y = Y, lambda = lseq, alpha = 0)
##
##      Df    %Dev  Lambda
## 1   100 0.00048 1000000
## 2   100 0.00120  398100
## 3   100 0.00301  158500
## 4   100 0.00745   63100
## 5   100 0.01807   25120
## 6   100 0.04051   10000
## 7   100 0.08158    3981
## 8   100 0.13460    1585
## 9   100 0.17770     631
## 10  100 0.20470     251
## 11  100 0.23490     100
## 12  100 0.29530      40
## 13  100 0.39950      16
## 14  100 0.54940       6
## 15  100 0.72070       3
## 16  100 0.84150       1
## 17  100 0.88500       0
## 18  100 0.91370       0
## 19  100 0.93400       0
## 20  100 0.94480       0
## 21  100 0.94480       0
```

```
## 22 100 0.94660       0
## 23 100 0.94690       0
## 24 100 0.94700       0
## 25 100 0.94700       0
## 26 100 0.94700       0
## 27 100 0.94710       0
```

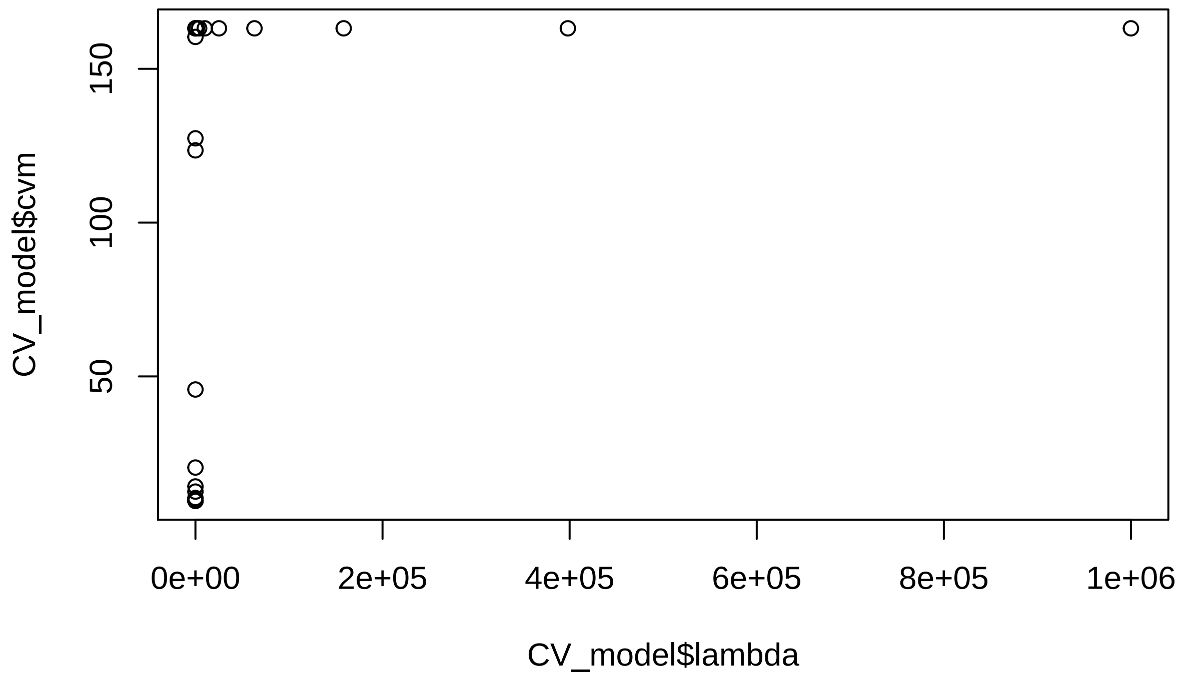


As log(lambda) increase the coefficeinets move towards zero.Also, here the MSE is directly proprtional to log(lambda) The lasso regression takes less parameters to form a model, this helps in reducing overfitting.Hence, lasso regression is good for this data.

4.7

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

```
## [1] 0
```




```
## 100 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## Sample     4.287484e-03
## Channel1  -1.001508e+02
## Channel2   3.295101e+01
## Channel3   3.140301e+01
## Channel4   2.763325e+01
## Channel5   2.401934e+01
## Channel6   2.165084e+01
## Channel7   1.967110e+01
## Channel8   1.907048e+01
## Channel9   1.418026e+01
## Channel10  1.117950e+01
## Channel11  9.000528e+00
## Channel12  4.765171e+00
## Channel13  2.222984e-01
## Channel14 -4.914474e+00
## Channel15 -1.005805e+01
## Channel16 -1.335023e+01
## Channel17 -1.900558e+01
## Channel18 -2.547316e+01
## Channel19 -2.747405e+01
## Channel20 -2.763953e+01
## Channel21 -2.109937e+01
## Channel22 -1.211532e+01
## Channel23 -3.566785e-01
```

```
## Channel24  1.284975e+01
## Channel25  2.100868e+01
## Channel26  2.010812e+01
## Channel27  1.055602e+01
## Channel28 -2.646660e+00
## Channel29 -1.039230e+01
## Channel30 -1.759401e+01
## Channel31 -2.258453e+01
## Channel32 -2.312947e+01
## Channel33 -1.860405e+01
## Channel34 -9.000484e+00
## Channel35 -8.731588e-01
## Channel36  4.438009e+00
## Channel37  7.222174e+00
## Channel38  1.349528e+01
## Channel39  1.689462e+01
## Channel40  2.251588e+01
## Channel41  3.320001e+01
## Channel42  4.659227e+01
## Channel43  5.327838e+01
## Channel44  4.406784e+01
## Channel45  2.151413e+01
## Channel46 -8.030382e+00
## Channel47 -3.501827e+01
## Channel48 -5.937246e+01
## Channel49 -5.626796e+01
## Channel50 -4.902151e+01
## Channel51 -4.547679e+01
## Channel52 -3.617988e+01
## Channel53 -1.790706e+01
## Channel54  6.462342e+00
## Channel55  2.893093e+01
## Channel56  4.503172e+01
## Channel57  4.953491e+01
## Channel58  4.152206e+01
## Channel59  2.634193e+01
## Channel60  2.242453e+01
## Channel61  1.089339e+01
## Channel62  4.655292e+00
## Channel63 -5.028978e+00
## Channel64 -8.476892e+00
## Channel65 -1.086221e+01
## Channel66 -9.690813e+00
## Channel67 -3.312843e+00
## Channel68 -1.380022e+00
## Channel69 -7.950456e+00
## Channel70 -1.493488e+01
## Channel71 -1.317936e+01
## Channel72 -7.924711e+00
## Channel73 -1.218866e+01
## Channel74 -1.756573e+01
## Channel75 -1.354140e+01
## Channel76 -2.980281e+00
## Channel77 -5.368919e+00
```

```
## Channel78 -7.594243e+00
## Channel79 -8.096247e+00
## Channel80 -6.096468e+00
## Channel81 -6.962912e+00
## Channel82 -5.310629e+00
## Channel83 -3.534425e+00
## Channel84 -1.270226e+00
## Channel85  2.677247e+00
## Channel86  4.393113e+00
## Channel87  5.432606e+00
## Channel88  3.651401e+00
## Channel89  1.630434e+00
## Channel90 -9.356368e-01
## Channel91 -2.484704e+00
## Channel92 -6.059932e-01
## Channel93  1.128427e+00
## Channel94  3.916530e+00
## Channel95  6.882021e+00
## Channel96  8.688672e+00
## Channel97  1.174128e+01
## Channel98  1.292105e+01
## Channel99  1.263547e+01
```

Since lambda is taken to be zero the model is now a linear regression which takes all the 100 values as significant parameters.

4.8

Step 4 variable selection is based on stepAIC which selects 63 variables and in step 7 all the 100 variables are selected.

Appendix

```
RNGversion("3.5.1")
knitr::opts_chunk$set(echo = TRUE)
library(readxl)
library(caTools)
library(class)
library(kknn)
spambase<-read_excel("spambase.xlsx")
spambase$Spam<-as.factor(spambase$Spam)
n=dim(spambase)[1]
RNGkind(sample.kind = "Rounding")
suppressWarnings(RNGversion("3.5.9"))
set.seed(12345)
id = sample(1:n,floor(n*0.5))
train = spambase[id,]
test = spambase[-id,]
RNGversion("3.5.1")
library(readxl)
library(caTools)
library(class)
library(kknn)
spambase<-read_excel("spambase.xlsx")
```

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

RNGversion("3.5.1")
#Fitting the model to logistic regression
classifier_train = glm(formula = Spam~.,family = binomial(link = 'logit'),data = train)
#Prediction for training data
prediction = predict(classifier_train,newdata = train, type = 'response')
prediction2 = predict(classifier_train,newdata = test, type = 'response')
#Classification of the predicted values(0.5)
y_pred = ifelse(prediction>0.5,1,0)
confusionMatrix_train =  table(y_pred,train$Spam)
confusionMatrix_train
#Misclassifier for training data(0.5)
misClasifier1 = 1 - (sum(diag(confusionMatrix_train))/sum(confusionMatrix_train))
misClasifier1
#Prediction for the testset
#prediction2 = predict(classifier_test,newx = as.matrix(test),type = 'response')
#for 0.5

y_pred_test = ifelse(prediction2>0.5,"Spam","Not Spam")
confusionMatrix_test = table(y_pred_test,test$Spam)
confusionMatrix_test
#misclasifier
misClasifier3 = 1 - (sum(diag(confusionMatrix_test))/sum(confusionMatrix_test))
misClasifier3
RNGversion("3.5.1")
#Classification of the predicted train values(0.8)
y_pred2 = ifelse(prediction>0.8,1,0)
confusionMatrix_train =  table(y_pred2,train$Spam)
confusionMatrix_train
#Miscalculation for training data (0.8)
misClasifier2 = 1 - (sum(diag(confusionMatrix_train))/sum(confusionMatrix_train))
misClasifier2
#Classification of the predicted test values(0.8)
y_pred_test2 = ifelse(prediction2>0.8,"Spam","Not Spam")
confusionMatrix_test = table(y_pred_test2,test$Spam)
confusionMatrix_test
#Miscalculation for test data (0.8)
misClasifier4 = 1 - (sum(diag(confusionMatrix_test))/sum(confusionMatrix_test))
misClasifier4
RNGversion("3.5.1")

#KNN(training data)
#Fittinging model to KNN
classifier_knn_train =kknn(formula = Spam~.,train,train,na.action = na.omit(),k=30,distance = 1 ,kernel
#PRediction for the trainignset using kknn
prediction4 = fitted(classifier_knn_train)
#Classification of the prediction
```

```r
y_pred_train_kknn = ifelse(prediction4>0.5,"Spam","Not Spam")
confusionMatrix_train_kknn1 = table(y_pred_train_kknn,train$Spam)
confusionMatrix_train_kknn1
#misClasifier
misClasifierkknn1 = 1 - (sum(diag(confusionMatrix_train_kknn1))/sum(confusionMatrix_train_kknn1))
misClasifierkknn1#KNN
#KNN(test data)
#Fittinging model to KNN
classifier_knn_test =kknn(formula = Spam~.,train,test,na.action = na.omit(),k=30,distance = 1 ,kernel =
#Prediction for the testset using knn
prediction5 = fitted(classifier_knn_test)
#Classification of the prediction
y_pred_test_kknn = ifelse(prediction5>0.5,"Spam","Not Spam")
confusionMatrix_test_kknn2 = table(y_pred_test_kknn,test$Spam)
confusionMatrix_test_kknn2
#misClasifier
misClasifierkknn2 = 1 - (sum(diag(confusionMatrix_test_kknn2))/sum(confusionMatrix_test_kknn2))
misClasifierkknn2
RNGversion("3.5.1")
#Fitting model to KNN
classifier_k1_knn_tr =kknn(formula = Spam~.,train,train,na.action = na.omit(),k=1,distance = 1 ,kernel =
#Prediction for the trainingset using kknn
prediction6 = fitted(classifier_k1_knn_tr)
#Classification of the prediction
y_pred_train_kknn = ifelse(prediction6>0.5,"Spam","Not Spam")
confusionMatrix_train_kknn3 = table(y_pred_train_kknn,train$Spam)
confusionMatrix_train_kknn3
#misClasifier
misClasifierkknn3 = 1 - (sum(diag(confusionMatrix_train_kknn3))/sum(confusionMatrix_train_kknn3))
misClasifierkknn3
#Fitting model to KNN
classifier_k1_knn_te =kknn(formula = Spam~.,train,test,na.action = na.omit(),k=1,distance = 1 ,kernel =
#Prediction for the testset using kknn
prediction7=fitted(classifier_k1_knn_te)
#Classification of the prediction
y_pred_test_kknn = ifelse(prediction7>0.5,"Spam","Not Spam")
confusionMatrix_test_kknn4 = table(y_pred_test_kknn,test$Spam)
confusionMatrix_test_kknn4
#misClasifier
misClasifierkknn4 = 1 - (sum(diag(confusionMatrix_test_kknn4))/sum(confusionMatrix_test_kknn4))
misClasifierkknn4
RNGversion("3.5.1")
#linear regression
mylin=function(X,Y, Xpred){
  Xpred1=cbind(1,Xpred)
  #MISSING: check formulas for linear regression and compute beta
  X=cbind(1,X)
  #beta
  beta= solve(t(X)%*%X)%*%t(X)%*%Y
  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

            for (k in 1:Nfolds){
              #MISSING: compute which indices should belong to current fold
              index1<-(k-1)*sf
              index2<-k*sf
              flag<-((index1)+1):index2
              #MISSING: implement cross-validation for model with features in "model" and iteration i.
              X_test<-X1[flag,which(model==1)]
              X_train<-X1[-flag,which(model==1)]
              Yp<-Y1[flag]
              Y_train<-Y1[-flag]
              Ypred<-mylin(X_train,Y_train,X_test)

              #MISSING: Get the predicted values for fold 'k', Ypred, and the original values for folf
              SSE=SSE+sum((Ypred-Yp)^2)
            }
            curr=curr+1
            MSE[curr]=SSE/n
            Nfeat[curr]=sum(model)
            Features[[curr]]=model

          }

  #MISSING: plot MSE against number of features
  plot(MSE,Nfeat)
```

```r
  i=which.min(MSE)
  return(list(CV=MSE[i], Features=Features[[i]]))
}

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

RNGversion("3.5.1")
library(readxl)
library(MASS)
library(glmnet)

tecator<-read_excel("tecator.xlsx")

#4.1
model<-lm(tecator$Protein~tecator$Moisture)
plot(tecator$Moisture,tecator$Protein)
abline(model,col = 'red')
summary(model)
RNGversion("3.5.1")
n=dim(tecator)[1]
set.seed(12345)
id = sample(1:n,floor(n*0.5))
train<-tecator[id,]
test<-tecator[-id,]
#ActualY<-test$Moisture
model<-list()
variance<-numeric(6)
Y<-numeric(6)
bias<-vector()
MSE<-vector()
for (i in 1:6) {
  model$i<-lm(Moisture~poly(Protein,i),data = train)
  Y<-predict(model$i,newdata = train,type = "response")
  bias[i]<-mean(Y)-mean(train$Moisture)
  variance[i]<-var(Y)
  MSE[i]<-mean((train$Moisture-Y)^2)
}
plot(x=c(1:6),y=MSE,xlab= "No. of iterations",ylab = "MSE",type = "l")
plot(bias,variance,type = "l")
RNGversion("3.5.1")
n=dim(tecator)[1]
set.seed(12345)
id = sample(1:n,floor(n*0.5))
train<-tecator[id,]
test<-tecator[-id,]
#ActualY<-test$Moisture
model<-list()
variance<-numeric(6)
Y<-numeric(6)
bias<-vector()
MSE<-vector()
for (i in 1:6) {
  model$i<-lm(Moisture~poly(Protein,i),data = train)
```

```
  Y<-predict(model$i,newdata = test,type = "response")
  bias[i]<-mean(Y)-mean(test$Moisture)
  variance[i]<-var(Y)
  MSE[i]<-mean((test$Moisture-Y)^2)
}
plot(x=c(1:6),y=MSE,xlab= "No. of iterations",ylab = "MSE",type = "l")
plot(bias,variance,type = "l")
RNGversion("3.5.1")
X<-tecator[,2:102]

model2<-lm(Fat~.,data = X)
fit<-stepAIC(model2,direction = "both")
fit$annova
RNGversion("3.5.1")
X <- data.matrix(tecator[,1:100])
Y <- data.matrix(tecator$Fat)
fit <- glmnet(X, Y, alpha = 0)
plot(fit,xvar = "lambda",label = TRUE)
ridge_cv <- cv.glmnet(X, Y, alpha = 0)
best_lambda <- ridge_cv$lambda.min
best_lambda
best_fit <- ridge_cv$glmnet.fit
ridge_cv$glmnet.fit
plot(ridge_cv,type = 'l')
RNGversion("3.5.1")
X <- data.matrix(tecator[,1:100])
Y <- data.matrix(tecator$Fat)
lseq <- 10^seq(-4,6, .4)
lseq[length(lseq)+1]=0
fit <- glmnet(X, Y, alpha = 1, lambda  = lseq)
plot(fit,xvar = "lambda",label = TRUE)
ridge_cv <- cv.glmnet(X, Y, alpha = 0, lambda = lseq)
best_lambda <- ridge_cv$lambda.min
best_lambda
best_fit <- ridge_cv$glmnet.fit
ridge_cv$glmnet.fit
plot(ridge_cv,type = 'l')
RNGversion("3.5.1")
CV_model <- cv.glmnet(X, Y, alpha = 1, lambda = lseq)
lambda_b <- CV_model$lambda.min
lambda_b
lasso <- glmnet(x=X, y=Y,alpha=1, lambda = lambda_b)
cv_score <- cbind(CV_model$lambda,CV_model$cvm)
min_cv <- cv_score[which(cv_score[,1] == CV_model$lambda.min),2]
optimal_features <- as.matrix(coef(lasso))
total_optimal_features <- length(which(optimal_features[,1] !=0))
plot(CV_model$lambda,CV_model$cvm)
lasso$beta

}
```