Lab 1 Machine Learning

Aman Kumar Nayak

11/22/2019

Assignment 1

Ques 1.1:

Imported dataset spambase.xlsx and divided same into test train on random bases.

Part 1.2

```
## Confusion Matrix for Train data with Y= 1 if p(Y=|X) > 0.5 else 0

##    con1Train
##       0   1
##   0 803 142
##   1  81 344

## Misclassification Rate for Training Data is  0.1627737

## Confusion Matrix for test data with Y= 1 if p(Y=|X) > 0.5 else 0

##    condition1
##       0   1
##   0 791 146
##   1  97 336

## Misclassification Rate for Testing Data is  0.1773723
```

On analysizing misclassification rate (for Y= 1 if p(Y=|X) > 0.5 else 0) we could see that, missclassification rate for training data is less which is obvious as training and testing is done using same data but when comparing both misclassification rate comparitive difference is 1.46% only which is not that much.

Part 1.3

```
## Confusion Matrix for Train data with Y= 1 if p(Y=|X) > 0.8 else 0

##    con2Train
##       0   1
##   0 941   4
##   1 335  90

## Misclassification Rate for Training Data is  0.2474453

## Confusion Matrix for test data with Y= 1 if p(Y=|X) > 0.8 else 0

##    condition2
##       0   1
##   0 926  11
##   1 367  66
```

```
## Misclassification Rate for Testing Data is  0.2759124
```

With the change classification principle i.e. Y= 1 if p(Y=|X) > 0.8 else 0, the misclassification rate for training data is higher as compared to the classification principle P(Y=1|X) > 0.5 in both the cases of training and testing data.

Effect of increased threshold

Even though misclassification rate is high but it can be seen that significantly less mails have been classified as spam for this new classification principle as the threshold for classification is high which help in false spam classification of non-spam email.

Part 1.4

```
## Confusion Matrix for Train data in case of KNN at K=30

##
##       0   1
##   0 807 138
##   1  98 327

## At KNN with K=30 Misclassification Rate for Training Data is  0.1722628

## Confusion Matrix for Testing data in case of KNN at K=30

##
##       0   1
##   0 594 351
##   1 265 160

## At KNN with K=30 Misclassification Rate for Testing Data is  0.449635
```

As compared to the logistic regression models, the misclassification rate for the K- nearest neighbours model with k = 30 is bit high for training data while it is historically high for testing data.

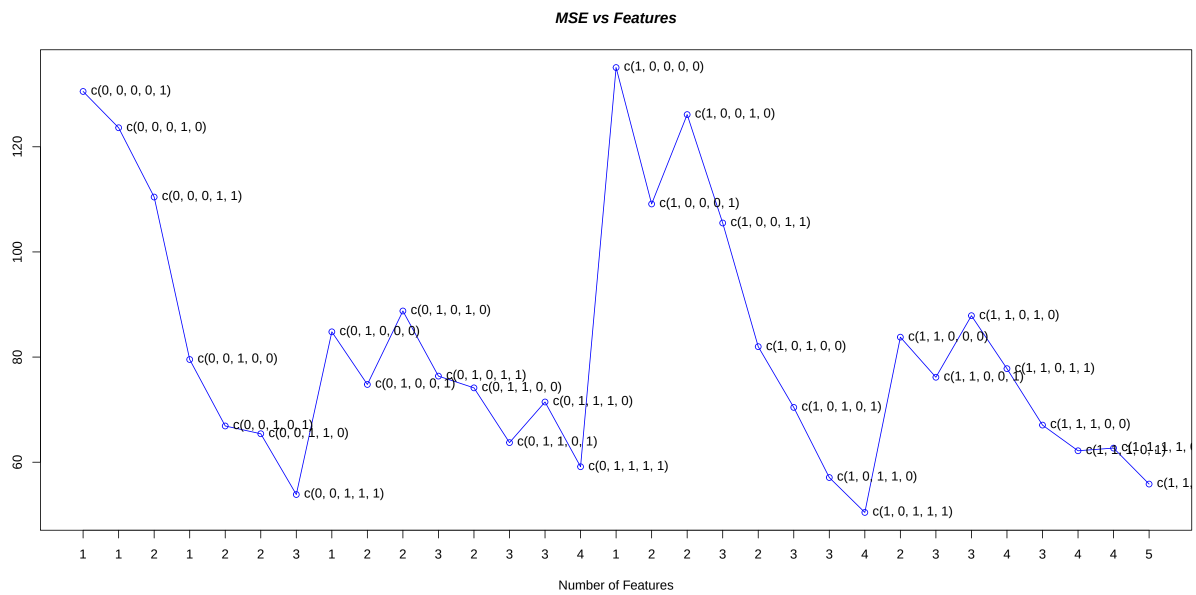
Part 1.5

```
## Confusion Matrix for Training data in case of KNN at K=1

##
##       0   1
##   0 945   0
##   1   0 425

## At KNN with K=1 Misclassification Rate for Training Data is  0

## Confusion Matrix for Testing data in case of KNN at K=1

##
##       0   1
##   0 559 386
##   1 258 167
```

At KNN with K=1 Misclassification Rate for Testing Data is 0.470073

When only one neighbour i.e. K = 1 is set we could see in case of training data misclassification is 0 as it is refereing to itself which is valid as train and prediction is done on same dataset while in case of test it is with 47% error rate which make model extremly poor as it is just referring to one closest neighbour.

Assignment 3



Cross Validation value is : 50.44948

Selected Features are : 1 0 1 1 1

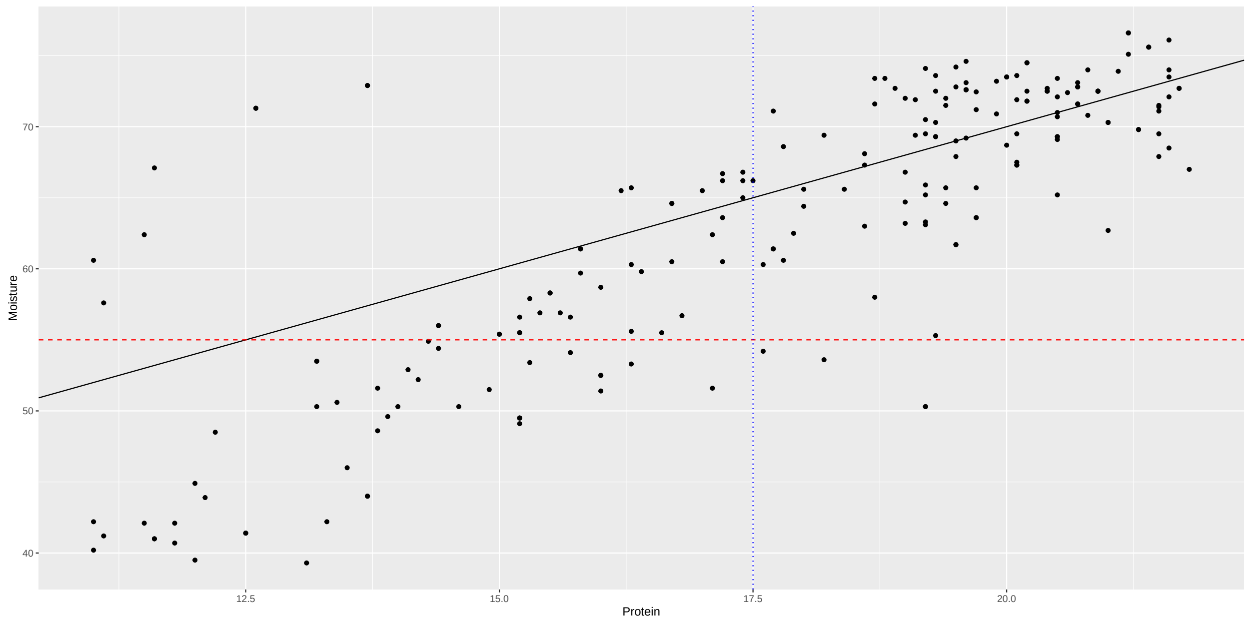
As per plot we can see that we are getting lowest value of **Cross Validation as MSE is 50.44948** for **feature group 1 0 1 1 1** so all idependent features except Examination have impact on the model.

When looking at ignored feature i.e. Examination which is measure of percentage draftees receiving highest mark on army examination which only cover specific set of population thus do not impact population as whole thus ignoring it will not impact model performace.

Thus selected features namely Agriculture, Education, Catholic and Infant.Mortality only have highest impact.

Assignment 4

Part 4.1




```
## Correlation between Moisture and Protein is  0.8145212
```

While trying to fit various straight lines over the distributed data it can be seen that non are able to cover spread of points in the data which will make induceses biasness as multiple points are always over and under the predicted curve and thus **simple linear model will not be ideal for this kind of data.** While we can see both Features have high correlation thus they can be modelled with polynomial model here.

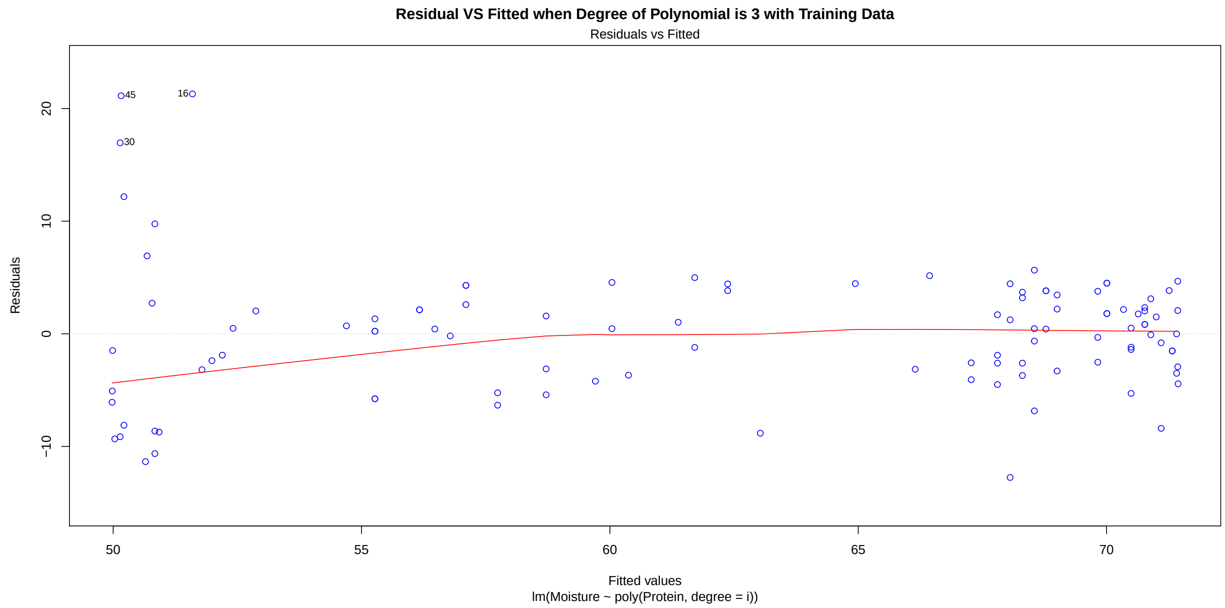
Part 4.2

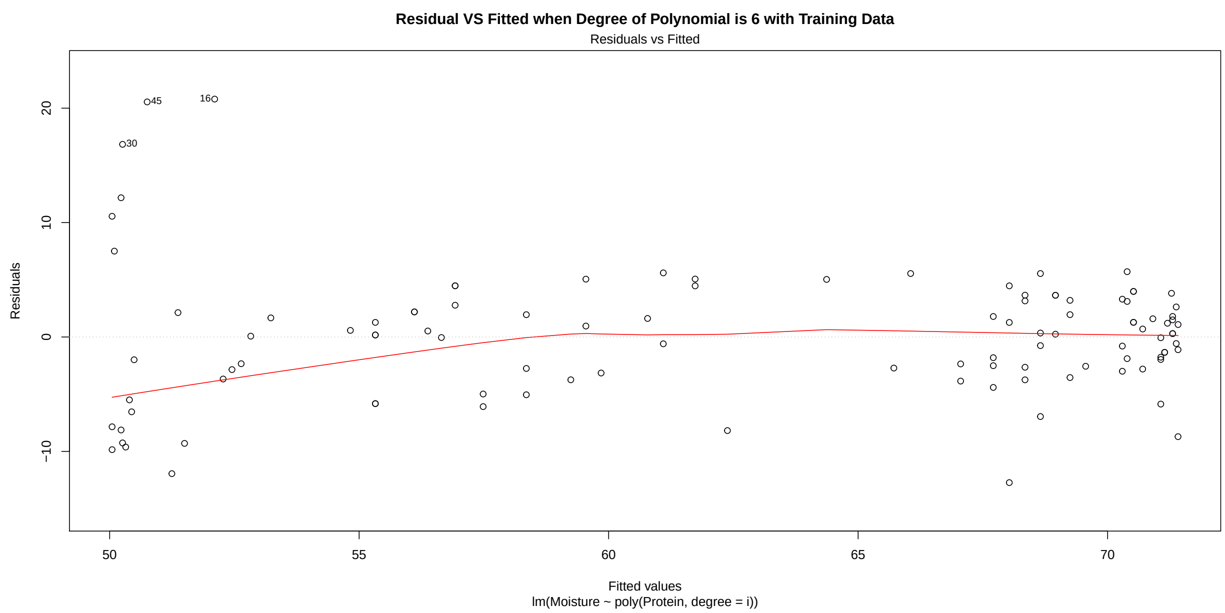
Below is asked probabilistic model which describe independent variable in Polynomial terms

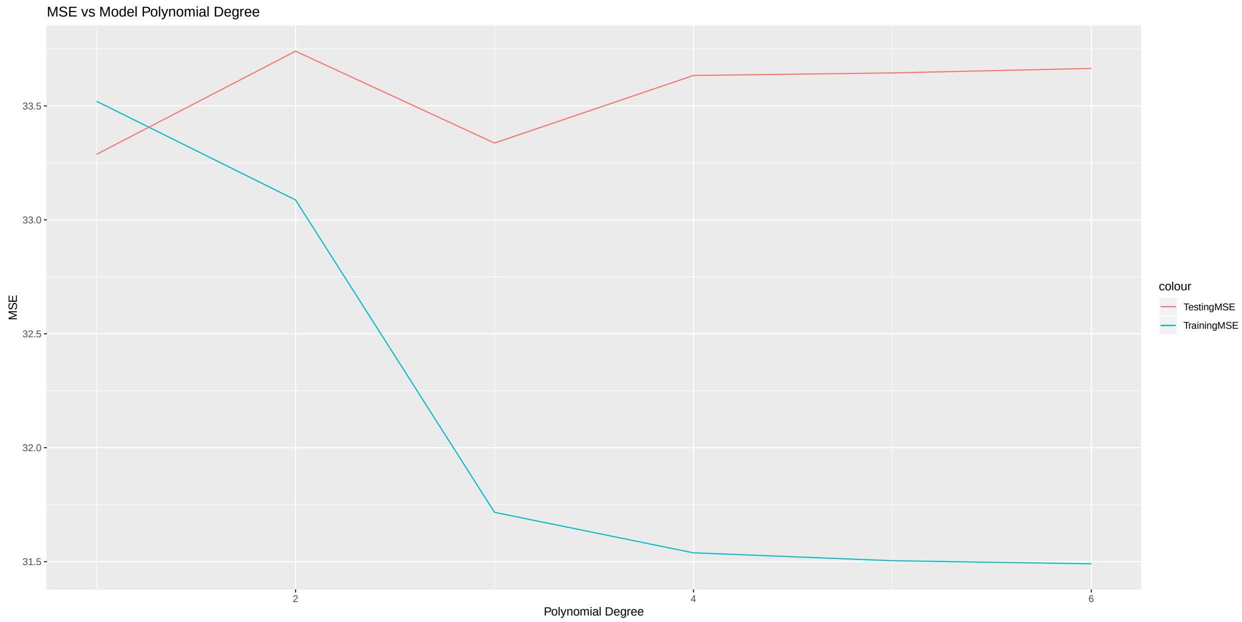
$Moisture \sim N(\beta 0 + \beta 1 * Protein^1 + \ldots.. + \beta i * Protein^i + \text{“}\sigma\text{”})$

MSE is critical criteria which provide us great details about how well model is performing with training data, if MSE is very low for training data but on contrast it is high for testing data it suggests that model is overfitted.

Part 4.3


Residual VS Fitted when Degree of Polynomial is 3 with Training Data
Residuals vs Fitted
Residuals
Fitted values
lm(Moisture ~ poly(Protein, degree = i))
45
16
30
20
10
0
−10
50
55
60
65
70


Residual VS Fitted when Degree of Polynomial is 6 with Training Data
Residuals vs Fitted
Residuals
Fitted values
lm(Moisture ~ poly(Protein, degree = i))
45
16
30
20
10
0
−10
50
55
60
65
70



In context to plot of "MSE VS Polynomial Degree"

We can see initialy when Polynomial Degree is 1, model is extemely simple as it try to fit straight line and thus have high bias and high variance which result in high value of MSE. As degree of polynomial increases model is more curved and thus we can see it is trying to reduce bias and more curved model, with decreassing value of MSE is obtaines in terms of training part thus with higher degree of polynomial model is trying to fit more perfectly with training data while it can be seen that apart from model where degree of polynomial is 3, MSE for test data is increasing thus model can be said to be in overfitting stage.

Thus Model with degree of polynomial 3 is best one for given data.

Part 4.4

```
## Summary of stepAIC with 63 selected features
```

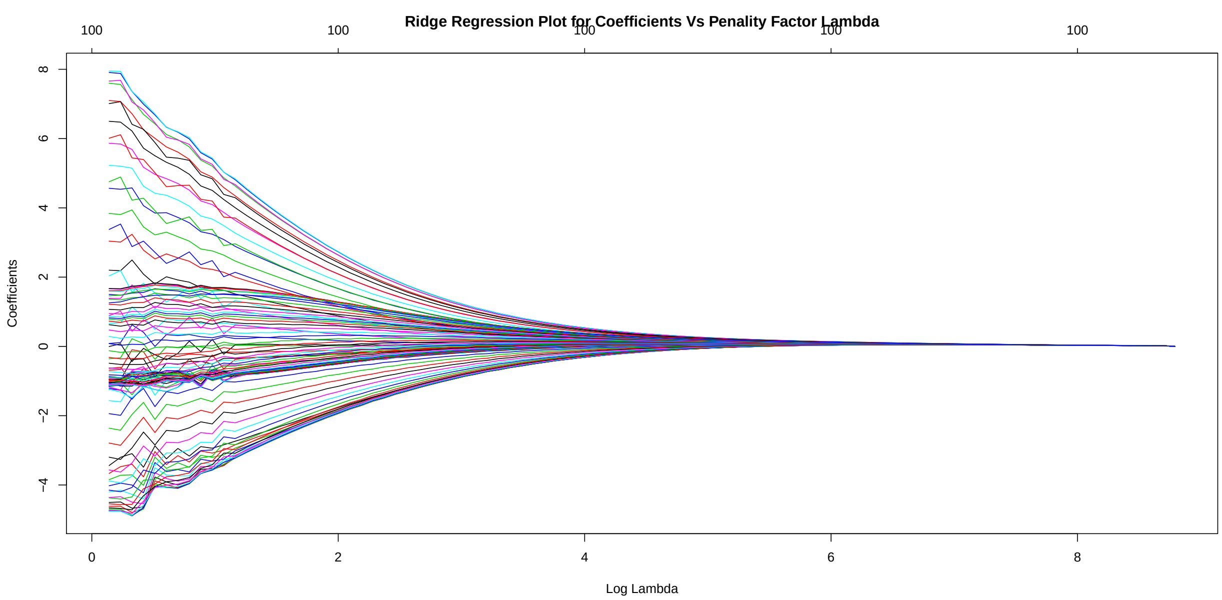
```
##
## Call:
## lm(formula = Fat ~ Channel1 + Channel2 + Channel4 + Channel5 +
##     Channel7 + Channel8 + Channel11 + Channel12 + Channel13 +
##     Channel14 + Channel15 + Channel17 + Channel19 + Channel20 +
##     Channel22 + Channel24 + Channel25 + Channel26 + Channel28 +
##     Channel29 + Channel30 + Channel32 + Channel34 + Channel36 +
##     Channel37 + Channel39 + Channel40 + Channel41 + Channel42 +
##     Channel45 + Channel46 + Channel47 + Channel48 + Channel50 +
##     Channel51 + Channel52 + Channel54 + Channel55 + Channel56 +
##     Channel59 + Channel60 + Channel61 + Channel63 + Channel64 +
##     Channel65 + Channel67 + Channel68 + Channel69 + Channel71 +
##     Channel73 + Channel74 + Channel78 + Channel79 + Channel80 +
##     Channel81 + Channel84 + Channel85 + Channel87 + Channel88 +
##     Channel92 + Channel94 + Channel98 + Channel99, data = tecator)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.82961 -0.57129 -0.00696  0.58152  2.86375
##
```

```
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)      7.093      1.453   4.882 2.64e-06 ***
## Channel1     10559.894   2333.430   4.525 1.21e-05 ***
## Channel2    -12636.967   3467.995  -3.644 0.000369 ***
## Channel4      8489.323   4637.993   1.830 0.069164 .  
## Channel5    -10408.967   4771.350  -2.182 0.030689 *  
## Channel7     -5376.018   3851.782  -1.396 0.164847    
## Channel8      7215.595   4246.489   1.699 0.091342 .  
## Channel11    -9505.520   5721.115  -1.661 0.098692 .  
## Channel12    37240.918  12290.648   3.030 0.002878 ** 
## Channel13   -41564.547  15892.375  -2.615 0.009817 ** 
## Channel14    34938.179  13290.454   2.629 0.009454 ** 
## Channel15   -23761.451   6584.006  -3.609 0.000417 ***
## Channel17     4296.572   3189.730   1.347 0.179998    
## Channel19    14279.808   5017.407   2.846 0.005042 ** 
## Channel20   -23855.616   5153.161  -4.629 7.85e-06 ***
## Channel22    18444.906   3381.683   5.454 1.97e-07 ***
## Channel24   -20138.426   4946.417  -4.071 7.52e-05 ***
## Channel25    18137.432   5374.094   3.375 0.000938 ***
## Channel26    -7670.318   3859.006  -1.988 0.048660 *  
## Channel28    20079.898   4991.631   4.023 9.06e-05 ***
## Channel29   -36351.014   7655.223  -4.749 4.72e-06 ***
## Channel30    18071.276   5863.802   3.082 0.002446 ** 
## Channel32     3838.013   2722.862   1.410 0.160729    
## Channel34    -9242.884   2225.926  -4.152 5.48e-05 ***
## Channel36     8070.938   3317.588   2.433 0.016152 *  
## Channel37    -9045.588   3536.621  -2.558 0.011522 *  
## Channel39    18664.454   5986.730   3.118 0.002183 ** 
## Channel40   -20069.709  10701.902  -1.875 0.062677 .  
## Channel41    22257.776  11122.533   2.001 0.047169 *  
## Channel42   -21760.853   5833.811  -3.730 0.000270 ***
## Channel45    18145.804   2985.416   6.078 9.50e-09 ***
## Channel46    -8225.696   3715.367  -2.214 0.028330 *  
## Channel47    -4986.549   2558.694  -1.949 0.053165 .  
## Channel48     2876.075   2014.985   1.427 0.155546    
## Channel50   -13009.410   4535.797  -2.868 0.004720 ** 
## Channel51    29251.161   6554.297   4.463 1.57e-05 ***
## Channel52   -26833.976   4389.473  -6.113 7.97e-09 ***
## Channel54    30954.862   4392.339   7.047 6.06e-11 ***
## Channel55   -35183.287   5646.314  -6.231 4.39e-09 ***
## Channel56    14912.986   2810.889   5.305 3.93e-07 ***
## Channel59    -8030.278   1887.431  -4.255 3.66e-05 ***
## Channel60    13071.416   2629.374   4.971 1.79e-06 ***
## Channel61    -7850.189   2246.864  -3.494 0.000625 ***
## Channel63    15059.275   3231.692   4.660 6.90e-06 ***
## Channel64   -19909.466   4727.696  -4.211 4.35e-05 ***
## Channel65     4190.184   3486.766   1.202 0.231346    
## Channel67    13850.508   3909.121   3.543 0.000526 ***
## Channel68   -25873.365   5304.223  -4.878 2.69e-06 ***
## Channel69    18362.385   3331.483   5.512 1.50e-07 ***
## Channel71    -9223.910   1558.752  -5.917 2.11e-08 ***
## Channel73    12456.498   2386.255   5.220 5.82e-07 ***
## Channel74    -5624.411   1933.590  -2.909 0.004177 ** 
```

```
## Channel78    -7927.105   2176.860  -3.642 0.000372 ***
## Channel79    15473.188   3812.200   4.059 7.89e-05 ***
## Channel80   -22391.895   4490.714  -4.986 1.67e-06 ***
## Channel81    13852.453   3105.934   4.460 1.59e-05 ***
## Channel84   -11442.630   3457.064  -3.310 0.001167 **
## Channel85    20228.671   4081.863   4.956 1.91e-06 ***
## Channel87   -15938.315   4102.273  -3.885 0.000153 ***
## Channel88     5647.072   3236.286   1.745 0.083033 .
## Channel92     6595.995   1864.595   3.537 0.000537 ***
## Channel94    -5497.846   1847.113  -2.976 0.003397 **
## Channel98    -8728.596   2489.314  -3.506 0.000598 ***
## Channel99     8554.587   1898.010   4.507 1.31e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.107 on 151 degrees of freedom
## Multiple R-squared:  0.9947, Adjusted R-squared:  0.9925
## F-statistic: 447.9 on 63 and 151 DF,  p-value: < 2.2e-16
```

While looking at summary, we can see that *63* features as they return lowest value of stepAIC which convey that only selected feature have highest impact on model prediction performance.

Part 4.5

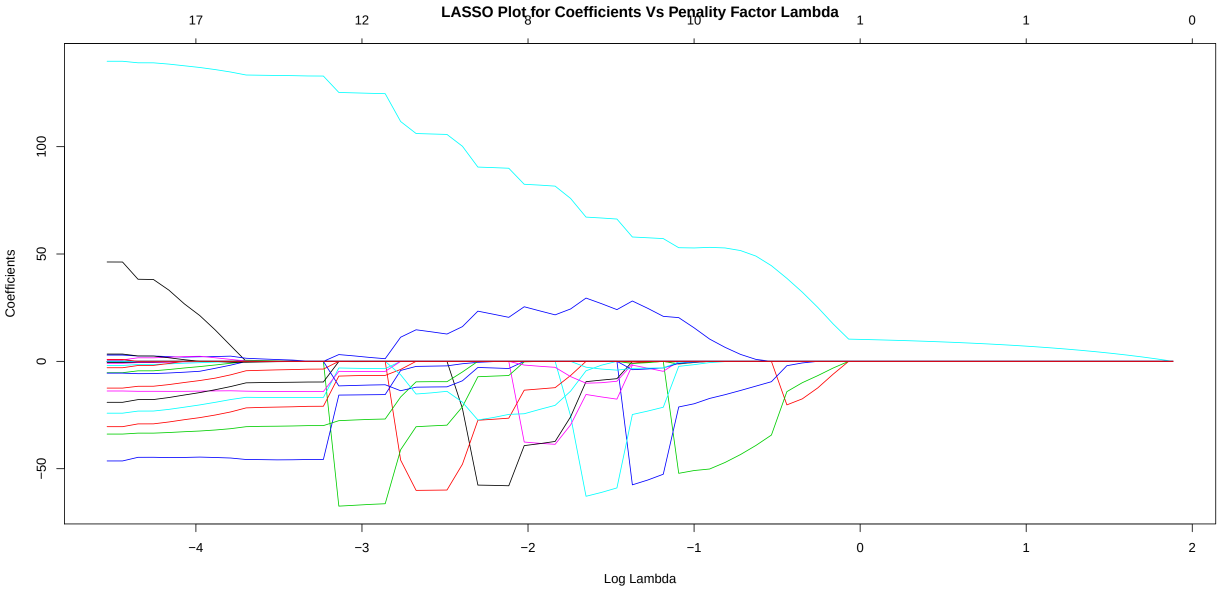


Lambda (λ) here is penality factor and objective is to make fit small by reducing residual sum of square plus add adding shrinkage penality. *Shrinkage penality is the lamda times the sum of squares of the coefficients.*

So as it can be seen that coefficients which are large are shrinking more with higher value of lamdba.

In regde regression number of variable remain same in final model.

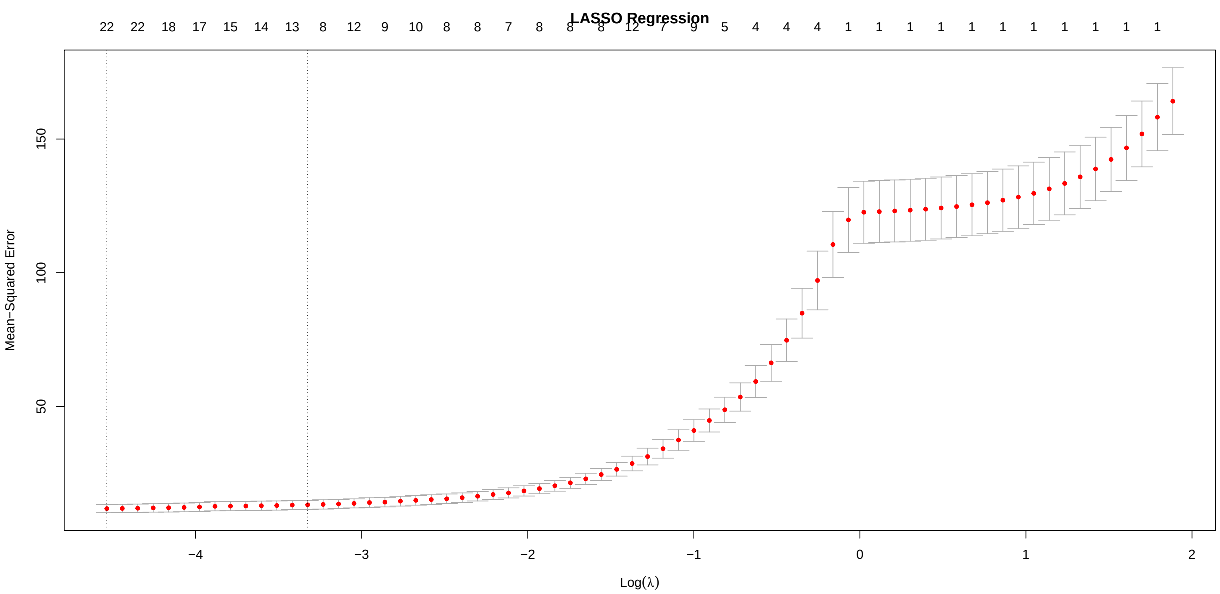
Part 4.6



Unlike Ridge Regression in Lasso Penality is sum of the absolute values of coefficients. Now here is LASSO it shrinks coefficients estimates towards to zero and it can even set variable effect to zero when higher enough value of lambda is available while it is not possible with ridge.

Thus LASSO apart from providing coefficient shrinking it also do feature selection based on value of lambda.

Part 4.7




```
## LASSO CV Model details are as below :

##
## Call:  cv.glmnet(x = as.matrix(reponse), y = as.matrix(tecator[, 102]),      family = "gaussian", al
##
## Measure: Mean-Squared Error
```

```
##
##      Lambda Measure    SE Nonzero
## min 0.01073   11.74 1.533      22
## 1se 0.03596   13.15 1.689      11
```

```
## Coefficients Values
```

```
##                        1
## (Intercept)  29.35933413
## Channel1      0.00000000
## Channel2      0.00000000
## Channel3      0.00000000
## Channel4      0.00000000
## Channel5      0.00000000
## Channel6      0.00000000
## Channel7      0.00000000
## Channel8      0.00000000
## Channel9      0.00000000
## Channel10     0.00000000
## Channel11     0.00000000
## Channel12     0.00000000
## Channel13     0.00000000
## Channel14   -45.72319212
## Channel15   -16.85432193
## Channel16   -14.10032123
## Channel17    -9.63550962
## Channel18    -3.64227816
## Channel19     0.00000000
## Channel20     0.00000000
## Channel21     0.00000000
## Channel22     0.00000000
## Channel23     0.00000000
## Channel24     0.00000000
## Channel25     0.00000000
## Channel26     0.00000000
## Channel27     0.00000000
## Channel28     0.00000000
## Channel29     0.00000000
## Channel30     0.00000000
## Channel31     0.00000000
## Channel32     0.00000000
## Channel33     0.00000000
## Channel34     0.00000000
## Channel35     0.00000000
## Channel36     0.00000000
## Channel37     0.00000000
## Channel38     0.00000000
## Channel39     0.03014549
## Channel40     0.12186923
## Channel41   132.92713434
## Channel42     0.00000000
## Channel43     0.00000000
## Channel44     0.00000000
## Channel45     0.00000000
```

```
## Channel46     0.00000000
## Channel47     0.00000000
## Channel48     0.00000000
## Channel49     0.00000000
## Channel50    -0.04063339
## Channel51   -20.95282664
## Channel52   -29.93640288
## Channel53     0.00000000
## Channel54     0.00000000
## Channel55     0.00000000
## Channel56     0.00000000
## Channel57     0.00000000
## Channel58     0.00000000
## Channel59     0.00000000
## Channel60     0.00000000
## Channel61     0.00000000
## Channel62     0.00000000
## Channel63     0.00000000
## Channel64     0.00000000
## Channel65     0.00000000
## Channel66     0.00000000
## Channel67     0.00000000
## Channel68     0.00000000
## Channel69     0.00000000
## Channel70     0.00000000
## Channel71     0.00000000
## Channel72     0.00000000
## Channel73     0.00000000
## Channel74     0.00000000
## Channel75     0.00000000
## Channel76     0.00000000
## Channel77     0.00000000
## Channel78     0.00000000
## Channel79     0.00000000
## Channel80     0.00000000
## Channel81     0.00000000
## Channel82     0.00000000
## Channel83     0.00000000
## Channel84     0.00000000
## Channel85     0.00000000
## Channel86     0.00000000
## Channel87     0.00000000
## Channel88     0.00000000
## Channel89     0.00000000
## Channel90     0.00000000
## Channel91     0.00000000
## Channel92     0.00000000
## Channel93     0.00000000
## Channel94     0.00000000
## Channel95     0.00000000
## Channel96     0.00000000
## Channel97     0.00000000
## Channel98     0.00000000
## Channel99     0.00000000
```

```
## Channel100    0.00000000
```

From above we can say that for lowest wale of MSE i.e. 11.74 we have 22 features selected here.

But looking at plot of MSE and Log lambda along wíth coefficient values for the model we can say that optimal lambda is Lambda.1se with value of 13.15 , and selected features are 11.

Part 4.8

Comparision of StepAIC and Lasso Cross-Validation result.

Number of selected coefficients in case of stepAIC were 63 variables while number was significantly reduced to just 11 in case of LASSO because of introduction of additional penality factor lambda which significantly reduce shrinkage here.

Appendix

```r
library(knitr)
knitr::opts_chunk$set(echo = TRUE,fig.height = 8,fig.width =16 , tidy = TRUE )

library(dplyr)
library(magrittr)
library(kknn)
library(readxl)
library(ggplot2)
library(MASS)
library(tidyverse)
library(broom)
library(glmnet)
#Import Data Into R
#import XML file in data frame
#location of Excel file in the system
path = "G:/MS Machine Learning/Term/Term2/ML/ML Assignment/1/spambase_CSV.csv"
#replace location as per file location

#loading CSV Data File into dataframe for analysis
df = data.frame(read.csv(file = path))

#braking df into part for traning and testing
n = dim(df)[1]
suppressWarnings(RNGversion("3.5.9"))
set.seed(12345)
id = sample(1:n , floor(n*0.5))

#braking df into test and train dataset
train = df[id ,]
test = df[-id,]


#Logistic Regression on Train Data Set
logReg = glm(Spam ~. , data = train , family = "binomial")

probabilites = logReg %>% predict(test , type = "response")
probabilitesTrain = logReg %>% predict(train , type = "response")

#cat("Confusion Matrix for Test data with Y= 1 if p(Y=|X) > 0.5 else 0")
```

```r
condition1 = ifelse(probabilites > 0.5, 1, 0)
con1 = table(test$Spam , condition1)
#con1
miscal1 = 1 - (sum(diag(con1)) / sum(con1))

#cat("Confusion Matrix for Test data with Y= 1 if p(Y=|X) > 0.5 else 0")
#miscal1


con1Train = ifelse(probabilitesTrain > 0.5, 1, 0)
con2 = table(train$Spam , con1Train)
#con2
miscal2 = 1 - (sum(diag(con2)) / sum(con2))

#cat("Confusion Matrix for Train data with Y= 1 if p(Y=|X) > 0.5 else 0")
#miscal2


cat("Confusion Matrix for Train data with Y= 1 if p(Y=|X) > 0.5 else 0" , "\n")
cat("\n")
con2
cat("\n")
cat("Misclassification Rate for Training Data is " , miscal2 , "\n")

cat("\n")

cat("Confusion Matrix for test data with Y= 1 if p(Y=|X) > 0.5 else 0" , "\n")
cat("\n")
con1
cat("\n")
cat("Misclassification Rate for Testing Data is " , miscal1 , "\n")


#cat("Confusion Matrix for Test data with Y= 1 if p(Y=|X) > 0.8 else 0")
condition2 = ifelse(probabilites > 0.8, 1, 0)
con3 = table(test$Spam , condition2)
#con3
miscal3 = 1 - (sum(diag(con3)) / sum(con3))
#miscal3

#cat("Confusion Matrix for Train data with Y= 1 if p(Y=|X) > 0.8 else 0")
con2Train = ifelse(probabilitesTrain > 0.8, 1, 0)
con4 = table(train$Spam , con2Train)
#con4
miscal4 = 1 - (sum(diag(con4)) / sum(con4))
#miscal4


cat("Confusion Matrix for Train data with Y= 1 if p(Y=|X) > 0.8 else 0" , "\n")
cat("\n")
con4
cat("\n")
cat("Misclassification Rate for Training Data is " , miscal4 , "\n")
```

```r
cat("\n")

cat("Confusion Matrix for test data with Y= 1 if p(Y=|X) > 0.8 else 0" , "\n")
cat("\n")
con3
cat("\n")
cat("Misclassification Rate for Testing Data is " , miscal3 , "\n")


knn30Train = kknn(as.factor(Spam) ~. , train , train , k = 30)
T1 =  table(train$Spam , knn30Train$fitted.values)
Knn_miscal1 = 1 - (sum(diag(T1)) / sum(T1))
#miscal1

knn30Test = kknn(as.factor(Spam) ~. , train , test , k = 30)
T2 =  table(train$Spam , knn30Test$fitted.values)
Knn_miscal2 = 1 - (sum(diag(T2)) / sum(T2))
#miscal2



cat("\n")
cat("Confusion Matrix for Train data in case of KNN at K=30" , "\n")

T1
cat("\n")

cat("At KNN with K=30 Misclassification Rate for Training Data is " , Knn_miscal1 , "\n")
cat("\n")



cat("Confusion Matrix for Testing data in case of KNN at K=30" , "\n")

T2
cat("\n")

cat("At KNN with K=30 Misclassification Rate for Testing Data is " , Knn_miscal2 , "\n")


#@ KNN = 1
knn1Train = kknn(as.factor(Spam) ~. , train , train , k = 1)
T3 =  table(train$Spam , knn1Train$fitted.values)
miscal3 = 1 - (sum(diag(T3)) / sum(T3))
#miscal3

knn1Test = kknn(as.factor(Spam) ~. , train , test , k = 1)
T4 =  table(train$Spam , knn1Test$fitted.values)
miscal4 = 1 - (sum(diag(T4)) / sum(T4))
#miscal4


cat("\n")
```

```r
cat("Confusion Matrix for Training data in case of KNN at K=1" , "\n")

T3
cat("\n")

cat("At KNN with K=1 Misclassification Rate for Training Data is " , miscal3 , "\n")

cat("\n")
cat("Confusion Matrix for Testing data in case of KNN at K=1" , "\n")

T4
cat("\n")

cat("At KNN with K=1 Misclassification Rate for Testing Data is " , miscal4 , "\n")

linReg = function(trainY,trainX , testY, testX){
  modelMatrix    = as.matrix(cbind(1,trainX))
  y              = as.matrix(trainY)
  #Computing beta for test parameter
  beta           = solve(t(modelMatrix)%*%modelMatrix)%*%t(modelMatrix)%*%y

  #Computing Y pred and residual of same
  predMatrix     = as.matrix(cbind(1,testX))
  fittedValues   = predMatrix%*%beta
  residual       = as.matrix(testY) - fittedValues
  return(residual)
}#linReg = function(){


myCV=function(X,Y,Nfolds){
  n=length(Y)
  p=ncol(X)
  suppressWarnings(RNGversion("3.5.9"))
  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()
  Features_axis <- c(0,0,0,0,0)
  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

            Xpred = data.frame(X1[,which(model == 1)])
            Y1 = data.frame(Y1)
            knode_s = 1
            knode_e = sF
            for (k in 1:Nfolds)
            {
              trainX = Xpred[-(knode_s:knode_e),]
              testX  = Xpred[knode_s:knode_e, ]
              trainY = Y1[-(knode_s:knode_e),]
              testY  = Y1[knode_s:knode_e,]


              residual = linReg(trainY,trainX ,testY, testX)

              SSE=SSE+sum((residual)^2)

                if(knode_e <= n)
                {
                  knode_s = (sF * k) +1
                  knode_e = sF * (k + 1)
                }else if(n%%Nfolds != 0)
                {
                  knode_s = knode_e +1
                  knode_e = n

                  trainX = Xpred[-(knode_s:knode_e),]
                  testX  = Xpred[knode_s:knode_e, ]
                  trainY = Y1[-(knode_s:knode_e),]
                  testY  = Y1[knode_s:knode_e,]


                  residual = linReg(trainY,trainX ,testY, testX)

                  SSE=SSE+sum((residual)^2)
                }
            }#for (k in 1:Nfolds)

            curr=curr+1
            MSE[curr]=SSE/n
            Nfeat[curr]=sum(model)
            Features[[curr]]=model
            Features_axis[curr] <- sum(model) #calculating total number of features used
          }#for(f5 in 0:1)

MSEplot = plot(MSE, type="o", col="blue", xaxt = "n" ,ann=FALSE)
            title(main="MSE vs Features", xlab = "Number of Features",
            text(MSE , labels = Features , pos = 4),
            font.main=4)
            axis(1,at=1:31, labels = Features_axis)
```

```r
  #MSEplot = plot(MSE)

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

op = myCV(swiss[,2:6] , swiss[,1] , 5)

cat("\n")

cat("Cross Validation value is : " , op$CV , "\n")

cat("\n")

cat("Selected Features are :" , op$Features)

cat("\n")


tecator <- read_excel("G:/MS Machine Learning/Term/Term2/ML/ML Assignment/1/tecator.xlsx")

ggplot(tecator) + geom_point(aes(x = Protein, y = Moisture), color = "black") +
geom_abline(intercept = 30 , slope = 2) +
geom_hline(yintercept = 55 , linetype = "dashed" , col = "red")  +
geom_vline(xintercept = 17.5 , linetype = "dotted" , col = "blue")


corela = cor(tecator$Protein , tecator$Moisture)

cat("Correlation between Moisture and Protein is ", corela)
#
# linReg = lm(formula = Moisture~Protein , data = tecator)
# summ = summary(linReg)
# cat("\n")
# cat("R Square value for Linear Regression model for this data is " , summ$r.squared , "\n")
# cat("\n")
#
# abline(linReg , lwd = 3 , col = "red")
# plot(linReg , which = c(1,1))

# Though R Square value is coming as 0.6634449 but while looking at Residual V/S Fitted Values we can m


#Divide data in train and test
suppressWarnings(RNGversion("3.5.1"))
set.seed(123456)

n = dim(tecator[,1])
i = sample(1:n , floor(n*.5))
trainData = tecator[i,]
testData  = tecator[-i,]
#train data
```

```r
trainMSE  = numeric(length = 6)
testMSE =  numeric(length = 6)
for (i in 1:6)
{
 polyLM = lm(Moisture~poly(Protein , degree = i) , data = trainData)
 trainMSE[i] = mean((polyLM$residuals)^2)

 pred = predict(polyLM , newdata = testData)
 testMSE[i] = mean((pred - testData$Moisture)^2)

 if(i == 3)
 {
   plot(polyLM , which = c(1,1) , col = "blue" ,
        main = "Residual VS Fitted when Degree of Polynomial is 3 with Training Data")
 }

  if(i == 6)
 {
   plot(polyLM , which = c(1,1) , col = "black",
        main = "Residual VS Fitted when Degree of Polynomial is 6 with Training Data")
 }
}

# plot(trainMSE , xlab = "Polynomianl Degree" , ylab = "MSE" , type = "l",
#      col = "red" )
# points(testMSE , col = "blue" , type = "l" ,  ylim = 20:40)

MSE = data.frame("TrainingMSE" = trainMSE , "TestingMSE" = testMSE)

ggplot(data = MSE)+
geom_line(aes(x = 1:6 , y = trainMSE , color = "TrainingMSE"))+
geom_line(aes(x = 1:6 , y = testMSE , color = "TestingMSE"))+
ylab("MSE") + xlab("Polynomial Degree") + ggtitle("MSE vs Model Polynomial Degree")




#stepAIC based model selection

lmFat = lm(Fat~.-Protein-Moisture-Sample , data = tecator)

stepAICModel = stepAIC(lmFat)



cat("Summary of stepAIC with 63 selected features")
summary(stepAICModel)

#Ridge Regression

# predictor = tecator$Fat
 reponse = as.matrix(tecator[,-c(1,102,103,104)])
#alpha is set to 0 as ridge penality and 1 for LASSO
```

```r
rrModel = glmnet(x = as.matrix(reponse) , y = as.matrix(tecator[,102]) ,
                 family = "gaussian" , alpha = 0)


plot(rrModel , xvar = "lambda" ,
     main = "Ridge Regression Plot for Coefficients Vs Penality Factor Lambda")


lassoModel = glmnet(x = as.matrix(reponse) , y = as.matrix(tecator[,102]) ,
                 family = "gaussian" , alpha = 1)

plot(lassoModel , xvar = "lambda" ,
     main = "LASSO Plot for Coefficients Vs Penality Factor Lambda")


cvModel = cv.glmnet(x = as.matrix(reponse) , y = as.matrix(tecator[,102]) ,
                 family = "gaussian" , alpha = 1)

plot(cvModel , xvar = "lambda" , main = "LASSO Regression")


# optimalLambda = cvModel$lambda.1se
# optimalMSE = cvModel$lambda
# optimalLambda
#
#
# lassoCoefficents = coef(cvModel , s = "lambda.1se")
# numberOfCoefficients = length(which(lassoCoefficents != 0))
cat("LASSO CV Model details are as below : \n")
cvModel

# cat("below is list of selected coefficients with at Lambda = min\n")
#
# lassoCoefficents = coef(cvModel , s = "lambda.min")
# lassoCoefficents = lassoCoefficents[which(lassoCoefficents != 0)]
# lassoCoefficents

cat("Coefficients Values")
as.matrix(coef(cvModel))
```