Lab1

Andreas C Charitos-andch552

21 Nov 2018

Contents

Assignment 1

1

2

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

```
## [1] "The confusion matrix for train data is :"
```

```
##            True train values
## Predictions   0   1
##           0 803  81
##           1 142 344
```

```
## ======================================
## Accuracy on train data is:  0.8372263
## Precision on train data is: 0.908371
```

```
## [1] "The confusion matrix for test data is :"
```

```
##           True test values
## Prediction   0   1
##          0 791  97
##          1 146 336
```

```
## ======================================
## Accuracy on test data is:  0.8226277
## Precision on test data is: 0.8907658
```

```
## ======================================
## Missclasification error in train:  0.1627737
## Missclasification error in test: 0.1773723
```

From the confusion matrix regarding train data we can see that 803 obs classified correctly as not spam and 344 as spam also 81 obs classified as not spam when they where spam (Type I error-false alarm) and 142 obs classified as spam when they where not. On the other side,regarding test data we can see that 791 obs classified correctly as not spam and 336 as spam also 97 obs classified as not spam when they where spam (Type I error-false alarm) and 146 obs classified as spam when they where not. Finaly,we can observe the misclassification rates and accuracies which are (0.1628 , 0.837) and (0.177,0.823) respectfully for train and test data and are qiute similar.

3

```
## [1] "The confusion matrix for the train data is :"
##            True train values
## Predictions   0   1
##           0 944 419
##           1   1   6
```

```
## ====================================
## Accuracy on train data is:  0.6934307
```

```
## [1] "The confusion matrix for test data is :"
```

```
##            True train values
## Predictions   0   1
##           0 936 427
##           1   1   6
```

```
## ====================================
## Accuracy on test data is:  0.6875912
```

```
## ====================================
## Missclasification error in train:  0.3065693
## Missclasification error in test: 0.3124088
```

Again using the confusion matrices we can conclude that for train data 944 obs classified correctly as not spam and only 6 as spam also 419 obs classified as not spam when they where spam (Type I error-false alarm) and 1 obs classified as spam when it was not. On the other side,regarding test data we can see that 936 obs classified correctly as not spam and only 6 as spam also 427 obs classified as not spam when they where spam (Type I error-false alarm) and 1 obs classified as spam when it was not. Finaly,we can observe the misclassification rates and accuracies which are (0.306 , 0.69) and (0.312,0.68) respectfully for train and test data.

Comparing the results with question 2 we can see that when using a higher threshold more data clasiffied as not spam when they where spam actually resulting higher misclasification rate and lower accuracy than using a lower therhold.That is because the higher threshold is more likely to predict and obs as not spam.

4

```
##
## Call:
## kknn(formula = Spam ~ ., train = strain, test = strain, k = 30)
##
## Response: "nominal"
```

```
##                Real class
## Predicted class   0   1
##               0 807  98
##               1 138 327
```

```
##
## Call:
## kknn(formula = Spam ~ ., train = strain, test = stest, k = 30)
##
## Response: "nominal"
```

```
##                Real class
## Predicted class   0   1
##               0 672 187
##               1 265 246
```

```
## ====================================
## Missclasification error in train:  0.1722628
## Missclasification error in test: 0.329927
```

```
##
## =====================================
## Accuracy in train:  0.8277372
## Accuracy in test: 0.670073
```

Using KNN with 30 neightbours we can see that misclassification and test data is 0.1722 and 0.3299 repsectfully,Regarding the results for step 2 we have roughly the same accuracy and misclassification error in train data but we have higher misclasification error and lower accuracy on the test data.

5

```
##
## Call:
## kknn(formula = Spam ~ ., train = strain, test = strain, k = 1)
##
## Response: "nominal"
```

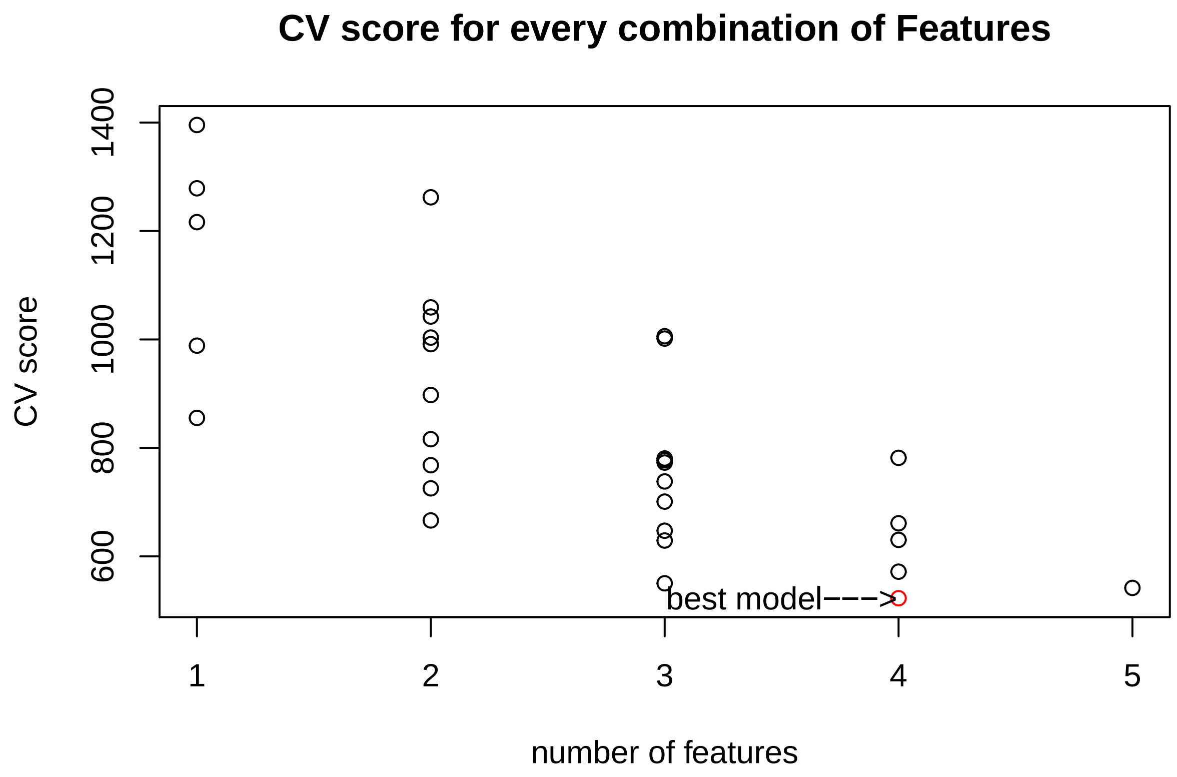
```
##                Real class
## Predicted class   0   1
##               0 945   0
##               1   0 425
```

```
##
## Call:
## kknn(formula = Spam ~ ., train = strain, test = stest, k = 1)
##
## Response: "nominal"
```

```
##                Real class
## Predicted class   0   1
##               0 640 177
##               1 297 256
```

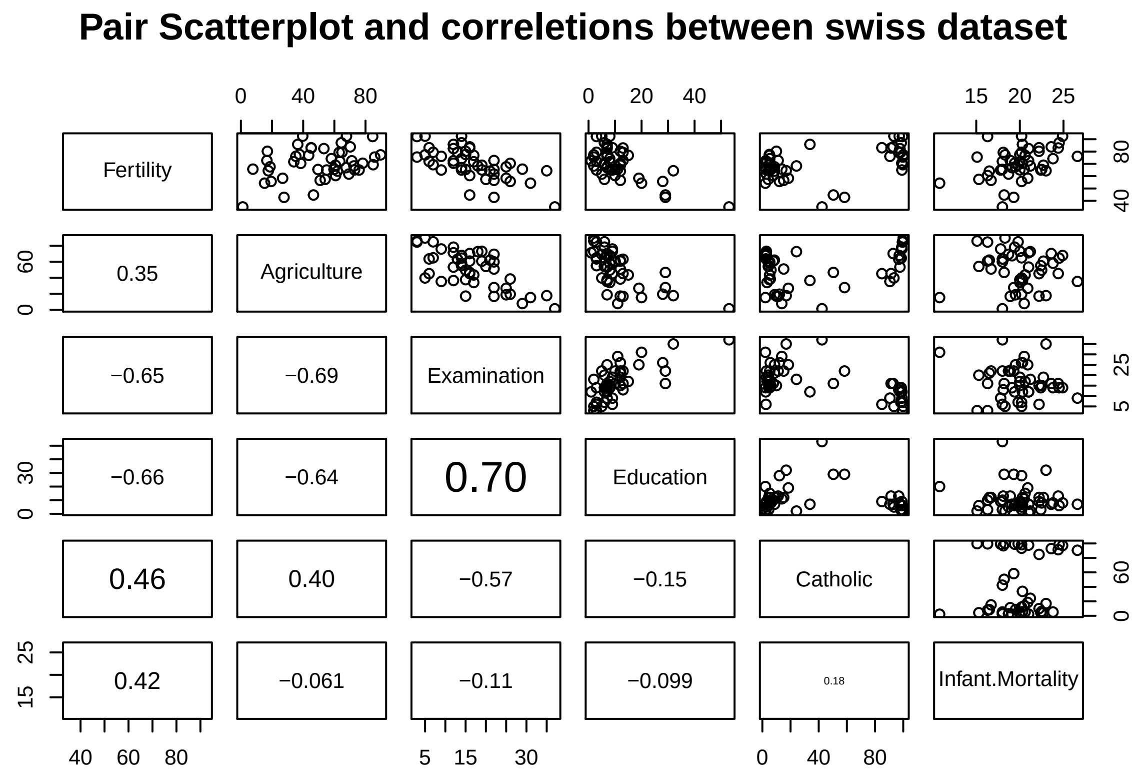
```
## =====================================
## Missclasification error in train:  0
## Missclasification error in test: 0.3459854
```

When decreasing K we can see that the misclassification error for the test data remains roughly the same but when comparing the misclasification errors for the train data the error is 0 for the K=1.This is beacause model is looking for only one closest neightbour and that tend to make model ovarfiting.We can also support this claim comparing the confusion matrices.Again for test data confusion matrices they are roughly the same but regarding train data the model with k=1 does a very high accurate predictions finding the true class for every observation in train data as we can see from confusion matrix there is 0 (TypeI) and (Type II) error.



```
## $`best combination`
## [1] "Agriculture"      "Education"        "Catholic"
## [4] "Infant.Mortality"
##
## $`best cv score`
## [1] 522.8431
```

We tested the linear model for every combination of the 5 independent features ("Agriculture","Examination","Education", "Catholic","Infant.Mortality") which is 31 diffrent models and evaluating each one with 5-fold Cross Validation we observe that the best combination of features predicting "Fertility" is ("Agriculture","Education", "Catholic","Infant.Mortality") and located as the red circle in plot.



Using the above plot of pair scatterplota we can obatain usefull information aboout the conection between Fertility and the features chosen by the best model.Starting from the connection of Fertility and Agriculture we can see that higher percentage of males involved in agriculture occupation tend to have higher fertility.The connection between Fertility and Education seems negative meaning that higher percentage of education is connected to lower fertility.Moving to the connection of Fertility and Catholic there seems to be 2 clusters that they might be reprecent the diffrence between catholic and protestant fertility.Finally,the connection between Fertility and Infant.Mortality is positive. In conclusion,the feature chosen as we can see from the scatterplots and the correlations between Fetility have a large impact on explaing Fertility.Moreover the fact that Examination was not chosen by the model might be because there is high correlation between Examination and Education and the effect both on the model is very small something we can suppport comparing the $R-Squared$ for our selected model and one with all the features which is (0.6707) and (0.671) respecfully.

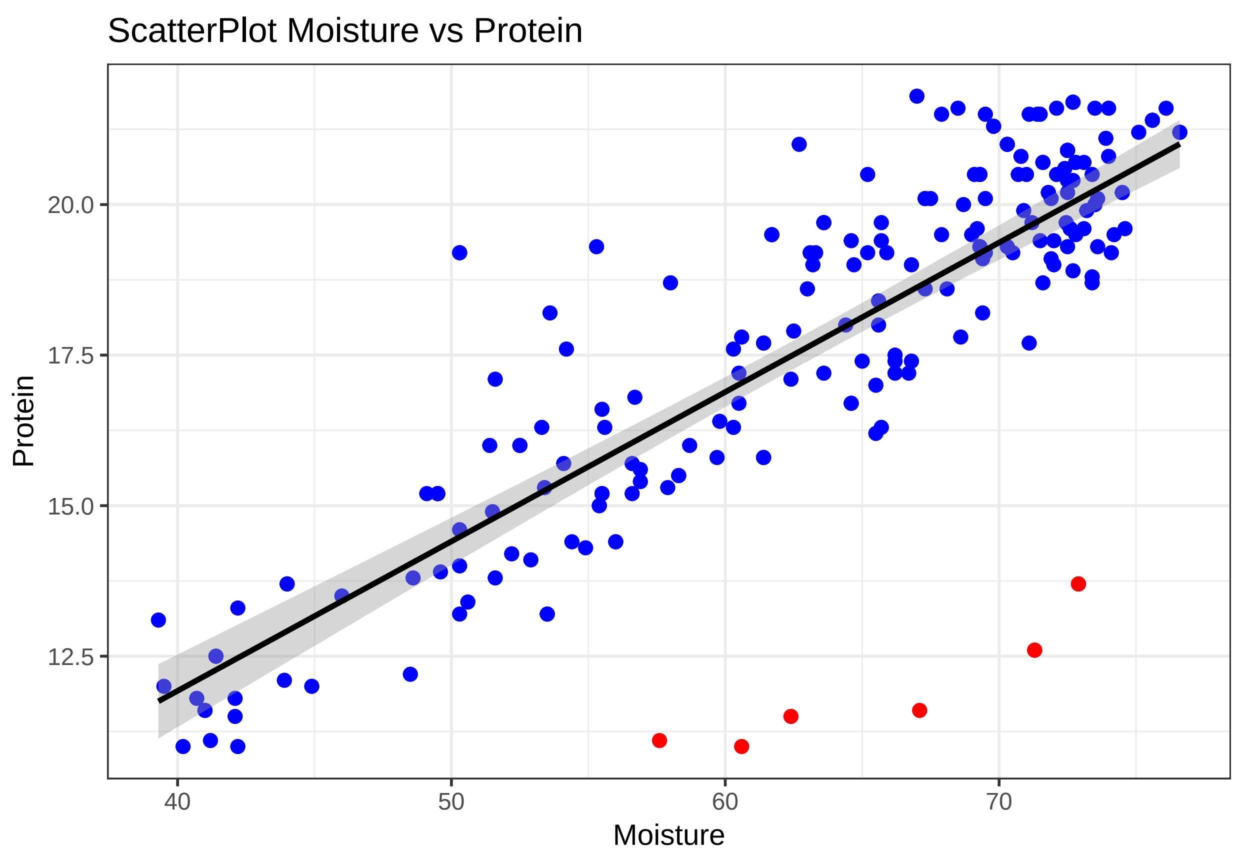
Assignment 4

1

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```



As we can observe for the scatterplot we can assume a linear model model will be a good fit for the data althougth there are some observations that are located in the lower left on the plot colored with red and having low Moisture and protein that we might consider them as outliers.

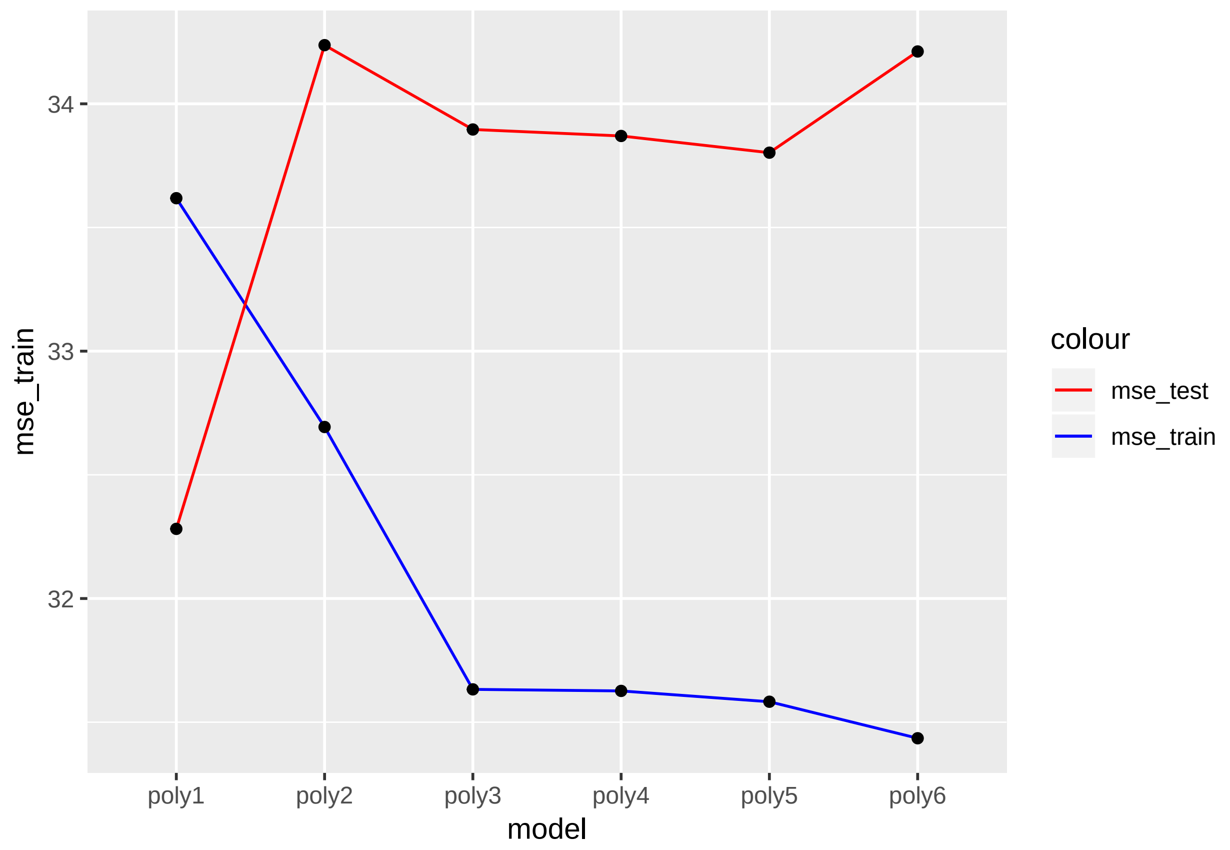
2

Lets denote Moisture as $\mathbf{y} = \left[\begin{array}{c} y_1, y_2, .., y_n \end{array}\right]$ and Protein as $\mathbf{x} = \left[\begin{array}{c} x_1, x_2, .., x_n \end{array}\right]$

The polynomial model up to power k can be expressed as : $y \sim N(X\beta + \epsilon, \sigma^2)$ where : β is a vector $\beta = \left[\begin{array}{c} \beta_1, \beta_2, .., \beta_n \end{array}\right]$ and X is a matrix $\mathbf{X} = \left[\begin{array}{c} 1, x_1, .., x_1^k \\ 1, x_2, ...x_2^k \\ .... \\ 1, x_n, ...x_n^k \end{array}\right]$

we allready know that maximizing likelihood is equivalent to minimaxing the negative log likelihood. $min - \log(\left(\frac{1}{\sqrt{(2\pi\sigma)^n}}\right) e^{-} \left(\frac{\sum_{i=1}^n (y_i - X_i\beta)^2}{\sqrt{(2\sigma^2)}}\right) = n \log \sqrt{(2\pi\sigma)} + \left(\frac{1}{2\sigma^2}\right) \sum_{i=1}^n (y_i - X_i\beta)^2 = = \left(\frac{1}{2\sigma^2}\right) \sum_{i=1}^n (y_i - \hat{y}_i)^2$ (1)
from 1 we can conlude that minimizing -log is equivalent to minimizing MSE.

3



According to the plot a model with a polynomial terms up to five seems to be more favourite because it has the lowest test mse compared with the other models combined with a very low train error.Furthermore looking at the plot we can see that as the complexity increases adding more polynomial terms the train error drops dramatically from the simple model to polynomial with 3rd power and then decreases slightly.But in the opposite the test error rises significally from the simple model to model with polynomial 2nd power and the slightly decreases until polynomial with power 5th and increases again for the polynomial with 6th power.Translating this plot in terms of bias-variance trade off we can say that as the complexity of model increases the train error is decreasing but test error starts decreasing and reaches a mininum point and then starts grows again.

4

```
## [1] "The number of coefficients with the intercept are :"
```

```
## [1] 64
```

```
## [1] "The vslues of coefficients are :"
```

```
##   (Intercept)       Channel1      Channel2       Channel4       Channel5
##      7.093133  10559.893784 -12636.966607   8489.323117 -10408.966948
##      Channel7       Channel8     Channel11      Channel12      Channel13
##  -5376.017738   7215.595409  -9505.520235  37240.918374 -41564.546571
##     Channel14      Channel15     Channel17      Channel19      Channel20
##  34938.179314 -23761.450875   4296.572462  14279.808102 -23855.616123
##     Channel22      Channel24     Channel25      Channel26      Channel28
##  18444.905722 -20138.426065  18137.431996  -7670.318234  20079.898191
##     Channel29      Channel30     Channel32      Channel34      Channel36
## -36351.013717  18071.275531   3838.013358  -9242.884498   8070.938452
##     Channel37     Channel39     Channel40     Channel41     Channel42
##  -9045.587624  18664.454171 -20069.708579  22257.776227 -21760.853228
##     Channel45     Channel46     Channel47     Channel48     Channel50
##  18145.803786  -8225.696060  -4986.549169   2876.074542 -13009.409717
##     Channel51     Channel52     Channel54     Channel55     Channel56
##  29251.160946 -26833.976402  30954.861519 -35183.287363  14912.986496
##     Channel59     Channel60     Channel61     Channel63     Channel64
##  -8030.277501  13071.415506  -7850.189324  15059.274961 -19909.466348
##     Channel65     Channel67     Channel68     Channel69     Channel71
##   4190.183533  13850.508143 -25873.365427  18362.384676  -9223.909939
##     Channel73     Channel74     Channel78     Channel79     Channel80
##  12456.497755  -5624.411385  -7927.104791  15473.187794 -22391.894812
##     Channel81     Channel84     Channel85     Channel87     Channel88
##  13852.452651 -11442.629734  20228.671387 -15938.315283   5647.072201
##     Channel92     Channel94     Channel98     Channel99
##   6595.995241  -5497.846381  -8728.596111   8554.587048
```

```
## [1] "The mean squared error is :"
```

```
## [1] 0.8598985
```

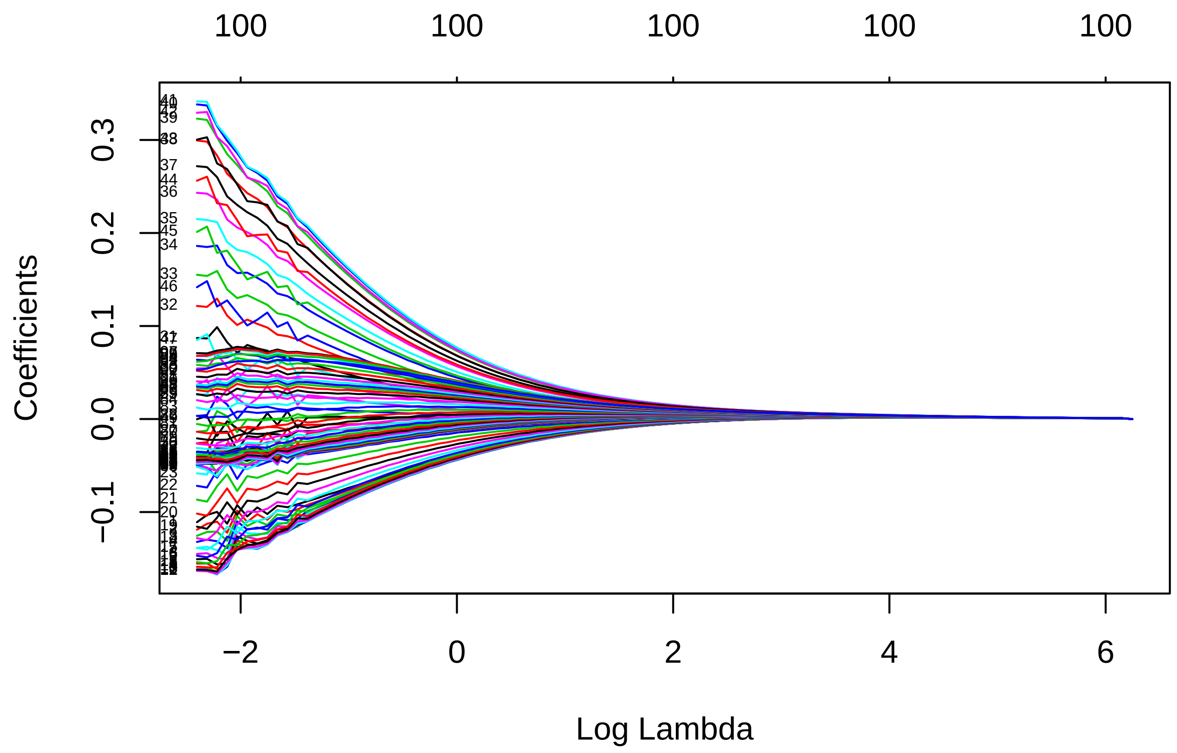
We observe that using stepAIC the best model returned is a model with 63 plus one for interept and the corresponding features listed above.

5

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

```
## Loading required package: foreach
```

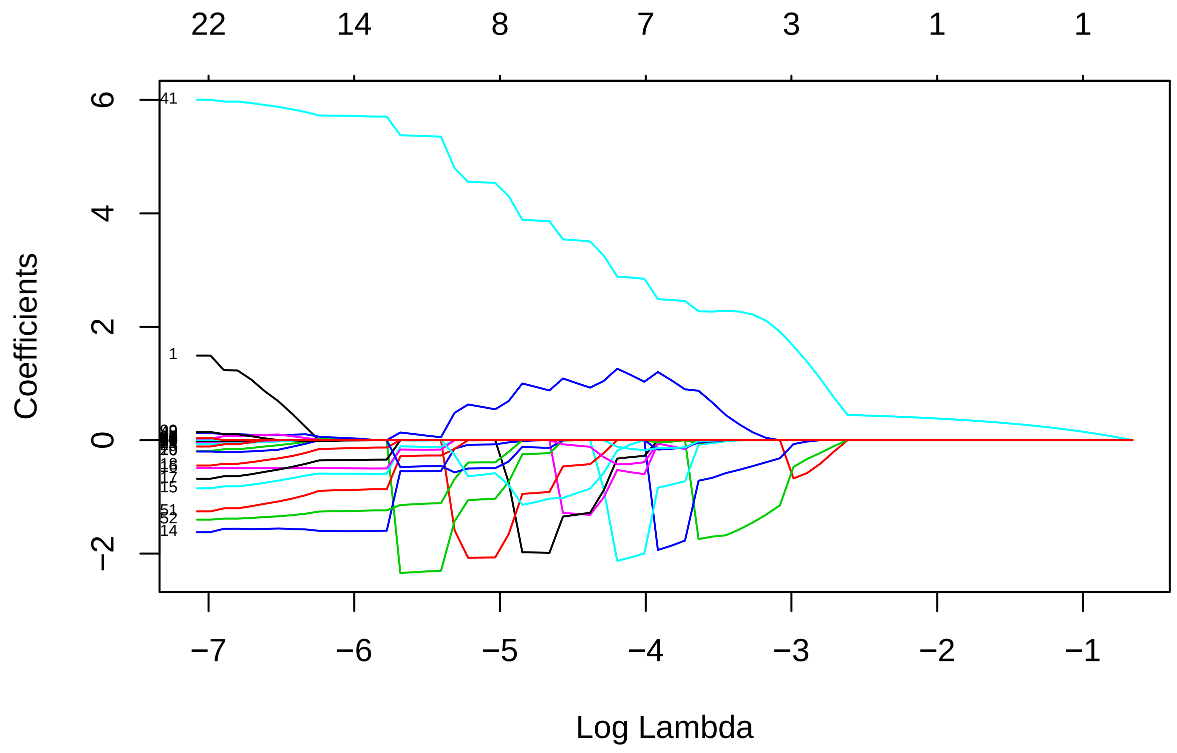
```
## Loaded glmnet 2.0-16
```



```
## [1] 0.09030492
```

From the graph of coefficients and log(lambda) we can see that the effect of ridge regression as lambda the increases is srinking the coefficients.Also we can notice that after a specific value of lambda the coefficients are all become zero.Ridge regression penalizes large coefficients by srinking and because we can't have zero out coefficients we either end up including all the coefficients in the model, or none of them.

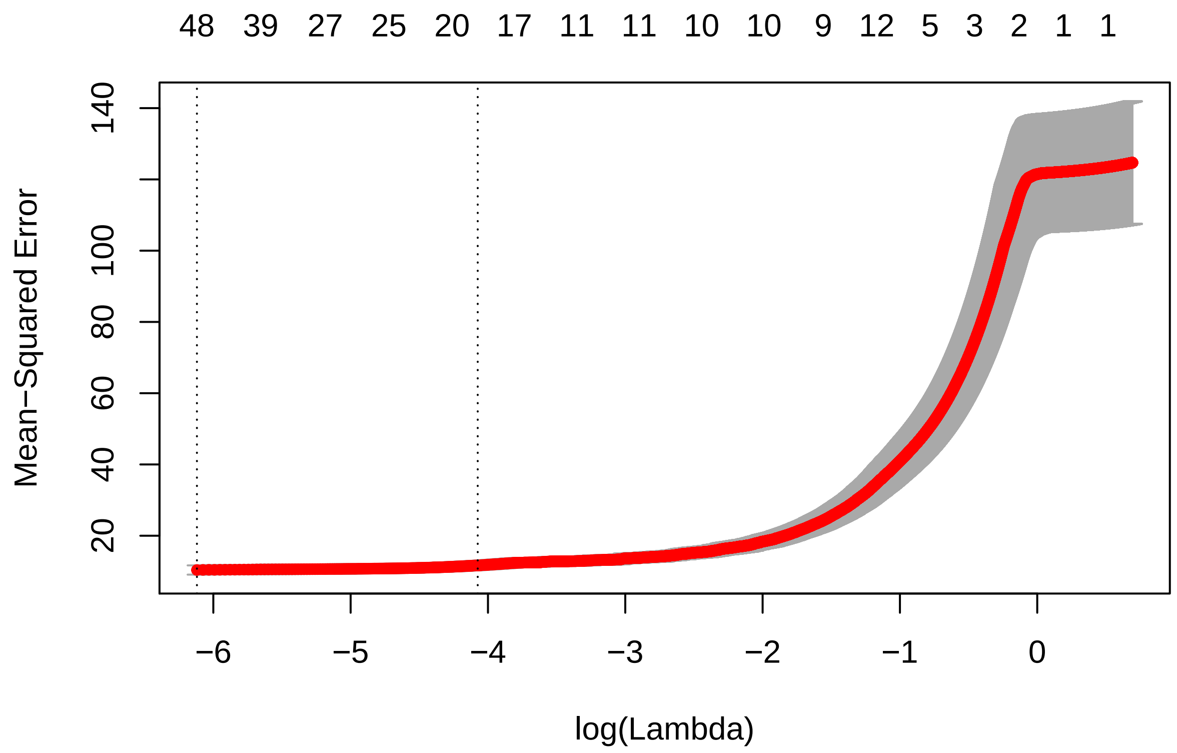
6



From the graph of coefficients and log(lambda) we can see that the effect of LASSO regression as lambda increases is penalizing coefficients and turn them to zero.In oppose to ridge regression in lasso for diffent lamda parameter we can have model where some coeeficients are 0 so lasso regression does both parameter shrinkage and variable selection automatically.

7

```
## Warning: from glmnet Fortran code (error code -19980); Number of nonzero
## coefficients along the path exceeds pmax=100 at 9980th lambda value;
## solutions for larger lambdas returned
```



```
## [1] "The minimun lambda obtained is:"

## [1] 0.0022

## [1] "The 1se lambda obtained is:"

## [1] 0.017

## [1] "The number of coefficients is :"

## [1] 20

## [1] "The minimum MSE score obtained is :"

## [1] 10.41708
```

The minimum lambda returned is 0.0022 which coresponds to the best model that may be too complex of slightly overfitted.The 1se lambda is 0.017 which returns the simplest model that has comparable error to the best model with lambda min.The number of the variables selected by the model are 20. Analyzing the plot showing the dependance of cv score with the penalty parameter lambda we can see that the MSE is growing rapidly as lambda increases and then after a value of lambda is reaches a plateu.

8

Comparing the results for the model used with stepAIC and LASSO model with cv we can observe that the LASSO coeffients are less compared with stepAIC.This is the result of LASSO regression which makes also feature selection and penalyzes large coefficients by turning them to 0.Moreover,comparing the MSE scores obained for both methods we can see that the MSE for the LASSO regression(10.41708) is higher compared to the one obtained with stepAIC(0.85) thus we can conclude that stepAIC offers a better model.

Appendix

```r
spam<-readxl::read_excel("spambase.xlsx")
#spam$Spam<-as.factor(spam$Spam)
#levels(spam$Spam)<-c("not spam","spam")
###Immporting the data and split to train and test
n=dim(spam)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=spam[id,]
test=spam[-id,]

model1<-glm(Spam~.,data=train,family = binomial(link = "logit"))
pred_train<-predict(model1,train,type="response")
pred_test<-predict(model1,test,type="response")
#broom::glance(model1) # use to see diagnostics of the model
fit.pred_train<-ifelse(pred_train>0.5,1,0)
fit.pred_test<-ifelse(pred_test>0.5,1,0)
#confusion matrix for train data
print("The confusion matrix for train data is :")
table(fit.pred_train,train$Spam,dnn = c("Predictions","True train values"))

train_accuracy<-(803+344)/(803+142+344+81)
train_precision<-(803)/(803+81)
cat("=====================================\nAccuracy on train data is: ",
    train_accuracy,"\nPrecision on train data is:",train_precision)

#confusionmatrix for test data
print("The confusion matrix for test data is :")
table(fit.pred_test,test$Spam,dnn = c("Prediction","True test values"))
test_accuracy<-(791+336)/(791+97+146+336)
test_precision<-791/(791+97)
cat("=====================================\nAccuracy on test data is: ", test_accuracy,
    "\nPrecision on test data is:",test_precision)

mis_error<-function(X,X1){
  n<-length(X)
  return(1-sum(diag(table(X,X1)))/n) #misclassification error function
}

miserror_train<-mis_error(fit.pred_train,train$Spam)
miserror_test<-mis_error(fit.pred_test,test$Spam)
cat("=====================================\nMissclasification error in train: ", miserror_train,
    "\nMissclasification error in test:",miserror_test)


fit.pred_train1<-ifelse(pred_train>0.9,1,0)
fit.pred_test1<-ifelse(pred_test>0.9,1,0)
#confusion matrix for train data
print("The confusion matrix for the train data is :")
table(fit.pred_train1,train$Spam,dnn = c("Predictions","True train values"))
#model accuracy on train data
ac1<-mean(train$Spam==fit.pred_train1)
cat("=====================================\nAccuracy on train data is: ", ac1)
```

```r
#confusionmatrix for test data
print("The confusion matrix for test data is :")
table(fit.pred_test1,test$Spam,dnn = c("Predictions","True train values"))
#model accuracy on test data
ac2<-mean(test$Spam==fit.pred_test1)
cat("=====================================\nAccuracy on test data is: ", ac2)

miserror_train1<-mis_error(fit.pred_train1,train$Spam)
miserror_test1<-mis_error(fit.pred_test1,test$Spam)
cat("=====================================\nMissclasification error in train: ", miserror_train1,
    "\nMissclasification error in test:",miserror_test1)

library(kknn)
s<-spam
#we convert to factor in order to use the model$fit otherwise doesn't work
s$Spam<-as.factor(s$Spam)
n=dim(s)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
strain=s[id,]
stest=s[-id,]


#first train the model with train,train and then make predictions
(knn<-kknn(Spam ~ ., strain,strain, k = 30))
table(knn$fit,strain$Spam,dnn=c("Predicted class","Real class"))
e1<-mis_error(knn$fit,strain$Spam)

#them train with train,test and make predictions
(knn1<-kknn(Spam ~ .,strain,stest, k = 30))
table(knn1$fit,stest$Spam,dnn=c("Predicted class","Real class"))
e2<-mis_error(knn1$fit,stest$Spam)

ac3<-mean(knn$fit==strain$Spam)
ac4<-mean(knn1$fit==stest$Spam)
cat("=====================================\nMissclasification error in train: ", e1,
    "\nMissclasification error in test:",e2)
cat("\n=====================================\nAccuracy in train: ", ac3,
    "\nAccuracy in test:",ac4)
(knn3<-kknn(Spam ~ .,strain,strain, k = 1))
table(knn3$fit,strain$Spam,dnn=c("Predicted class","Real class"))
e3<-mis_error(knn3$fit,strain$Spam)

(knn4<-kknn(Spam ~ .,strain,stest, k = 1))
table(knn4$fit,stest$Spam,dnn=c("Predicted class","Real class"))
e4<-mis_error(knn4$fit,stest$Spam)


cat("=====================================\nMissclasification error in train: ", e3,
    "\nMissclasification error in test:",e4)

feature_selection<-function(X,Y,N){
  n<-ncol(X)
  idx<-1:2^n-1
  t<-vector()
  mat<-sapply(idx, function(id){
    t<-cbind(t,as.integer(intToBits(id)))
    t})
  m<-mat[1:n,2:ncol(mat)]
  ######################################
  set.seed(12345)
  #X<-X[sample(nrow(X)),]
  #Y<-Y[sample(length(Y))]
  id<-sample(nrow(X))
  X<-X[id,]
  Y<-Y[id]
  #Create N equally size folds
  folds <- cut(seq(1,nrow(X)),breaks=N,labels=FALSE)
  d<-matrix(0,nrow=N,ncol=dim(m)[2])
  n_features<-rep(0,ncol(d))
  for (i in 1:ncol(m)){
    x<-X[which(m[,i]==1)]
    n_features[i]<-ncol(x)
    for(j in 1:N){

      testIndexes <- which(folds==j,arr.ind=TRUE)
      testX <- as.matrix(x[testIndexes, ])
      trainX <- as.matrix(x[-testIndexes, ])
      testy<-Y[testIndexes]
      trainy<-Y[-testIndexes]
      trainX<-cbind(1,trainX)
      testX<-cbind(1,testX)

      w<-round(as.vector(solve(t(trainX)%*%trainX)%*%t(trainX)%*%trainy),3)
      y_pred<-round(as.matrix(testX%*%w),3)
      sse<-sum((testy-y_pred)^2)
      d[j,i]<-sse
    }
  }
 d<-d
 s<-apply(d, MARGIN = 2, function(x) mean(x, na.rm=TRUE))
  bindex<-which(s==min(s))
  best_comb<-X[which(m[,bindex]==1)]

  plot(x=n_features,y=s,type="p",xlab="number of features",ylab="CV score",
       col=ifelse(s==s[bindex],"red","black"),main="CV score for every combination of Features")
  text(x=3.5,y=522.8431,labels=c("best model--->"))
  return(list("best combination"=colnames(best_comb),"best cv score"=s[bindex]))
}


Y<-swiss[,"Fertility"]
X<-swiss[!names(swiss)%in%c("Fertility")]
D<-feature_selection(X,Y,5)
D
```

```r
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <-cor(x, y)
    txt <- format(c(r, 0.123456789), digits = digits)[1]
    txt <- paste0(prefix, txt)
    if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor * r)
}

pairs(swiss,lower.panel = panel.cor,main="Pair Scatterplot and correletions between swiss dataset")
#import libaries and data
library(ggplot2)
library(plotly)
tecator<-readxl::read_excel("tecator.xlsx")

library(dplyr)
p<-ggplot(tecator,aes(x=Moisture,y=Protein))+
  geom_point(color = ifelse(tecator$Moisture>55&tecator$Moisture<78&tecator$Protein<14, "red", "blue")
             ,size = 2)+
  ggtitle("ScatterPlot Moisture vs Protein")+theme_bw()+geom_smooth(method = "lm",color="black")

p


#alternative code with plot

#plot(tecator$Moisture,tecator$Protein,col =
       #ifelse(tecator$Moisture>55&tecator$Moisture<78&tecator$Protein<14, "red", "blue"),pch=13)
#abline(lm(Protein ~ Moisture,data=tecator))



#split data to 50\50
n=dim(tecator)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
traindata=tecator[id,]
validata=tecator[-id,]

#create a matrix to store the iteration data
d<- matrix(0, ncol = 2, nrow = 6)
colnames(d)<-c("mse_train","mse_test")
rownames(d)<-c("poly1","poly2","poly3","poly4","poly5","poly6")

#for loop
for (i in 1:6){
  m<-lm(Moisture~poly(Protein,i),data=traindata)
  preds_valid<-predict(m,validata)
  preds_train<-predict(m,traindata)

  mse <- function(true,predicted) {
    return (mean((true-predicted)^2))
    }

  mse_train<-mse(traindata$Moisture,preds_train)
  mse_test<-mse(validata$Moisture,preds_valid)
  d[i,1]<-mse_train
  d[i,2]<-mse_test
}

dd<-as.data.frame(d)
dd$model<-rownames(dd)


pplot<- ggplot(data = dd, aes(x = model,group=1))+
  geom_line(aes(y=mse_train,color="mse_train"))+geom_point(aes(y=mse_train))+
  geom_line(aes(y=mse_test,color="mse_test"))+
  geom_point(aes(y=mse_test))+
  scale_color_manual(labels = c("mse_test", "mse_train"), values = c("red", "blue"))

#print he plot
pplot



#alternative code with the matrix d and plot comand

# plot(seq(1,6),d[,1],col="blue",pch=19,ylim=c(31,35))
# lines(seq(1,6),d[,1],col="blue")
# points(seq(1,6),d[,2],col="red",pch=19)
# lines(seq(1,6),d[,2],col="red")
# legend(4,33,legend=c("MSE train","MSE test"),col=c("blue","red"),lty=1)
#


###plot with plotly
# library(plotly)
# dd
# pp <- plot_ly(dd, x = ~model, y = ~mse_train, name = 'mse_train', type = 'scatter',mode="lines+marker
#   add_trace(x=~model,y = ~mse_test, name =" mse_test", mode = 'markers+lines')
# pp
###plot with ggplot
#p1<-ggplot(dd,aes(x=model,y=mse_train,group=1))+geom_point(color="blue")+geom_line()
#p1+geom_point(data=dd, aes(x=model,y=mse_test,color="red"))+geom_line(data=dd,aes(x=model,y=mse_test))

#====================Use the entire dataset for the rese=========================================
###4
library(MASS)
tecator_fat<- tecator[!names(tecator)%in%c("Sample","Protein","Moisture")]
mod<-lm(Fat~.,data= tecator_fat)
step<-stepAIC(mod,direction="both",trace=F)
#summary(step)
#the coefficients selected are
print("The number of coefficients with the intercept are :")
```

```r
length(step$coefficients) # 64 wirh the intercept
print("The vslues of coefficients are :")
step$coefficients
print("The mean squared error is :")
mean((step$fit-tecator_fat$Fat)^2) #mean square error

###5##Ridge
library(glmnet)
#glmnet takes x and y matrices not formula!
#we scale the covariates and responses
covariates<-scale(tecator_fat[,-101])
responses<-scale(tecator_fat[,101])
mod1<-glmnet(as.matrix(covariates),responses,family="gaussian",alpha=0)

plot(mod1,xvar="lambda",label=TRUE)
min_lambda=min(mod1$lambda)
min_lambda
###6##LASSO
mod2<-glmnet(as.matrix(covariates),responses,family="gaussian",alpha=1)
plot(mod2,xvar="lambda",label=TRUE)
###7##CV-LASSO###
set.seed(12345)
#we used the unscaled data here as we did with the stepAIC
mod3<-cv.glmnet(as.matrix(tecator_fat[,-101]),tecator_fat$Fat,alpha=1,
                family="gaussian",lambda=seq(0,2,0.0001),type.measure = "mse")
plot(mod3)
print("The minimun lambda obtained is:")
mod3$lambda.min
print("The 1se lambda obtained is:")
mod3$lambda.1se
print("The number of coefficients is :")
co<-coef(mod3,s="lambda.1se")
length(co@x) #co@p
print("The minimum MSE score obtained is :")
#min(mod3$cvm)
pr_lasso<-p <- predict(mod3, newx=as.matrix(tecator_fat[,-101]), s="lambda.1se")
mse_lasso <- mean((pr_lasso-tecator_fat$Fat)^2)
mse_lasso
```