Lab 1

Bjorn Hansen, Erik Anders, Ahmed Alhasan

11/23/2019

Assignment 1.Spam classification with nearest neighbors

First is to import the data and divide it into training and test sets. Below is shown the header of the data.

The glm() and predict() function is then used on a the sets shown above to predict the spam data. The confusion matrices for the test and trained data sets is shown below.

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

Below is presented the accuracy of the classifier and confusion matrices. As can be seen the when the clasifier accuracy drops for P > 0.8 as it seems to become to sensitive and marks even normal emails as spam.

Well, that always happens, setting the threshold is setting the border.

```
## [1] "Accuracy with test data with P > 0.5: 0.828467153284671"
```

```
##    Y hat
## Y     0   1
##   0 808 143
##   1  92 327
```

You are supposed to show this for training and test!
Also you're supposed to show the misclassification rate, not the accuracy.

```
## [1] "Accuracy with P > 0.8: 0.756204379562044"
```

```
##    Y hat
## Y     0   1
##   0 931  20
##   1 314 105
```

Below is presented a classifier trained on the same data as above but this time using K nearest neighbors. As can be seen below, the KNN classifier works considerbly worse for our data than the logistic regression classifier.

Same here, please show training and test.

```
## [1] "Accuracy with K = 30: 0.536496350364964"
```

```
##    Y hat
## Y     0   1
##   0 589 342
##   1 293 146
```

These numbers seem to be wrong, the accuracy should be higher.

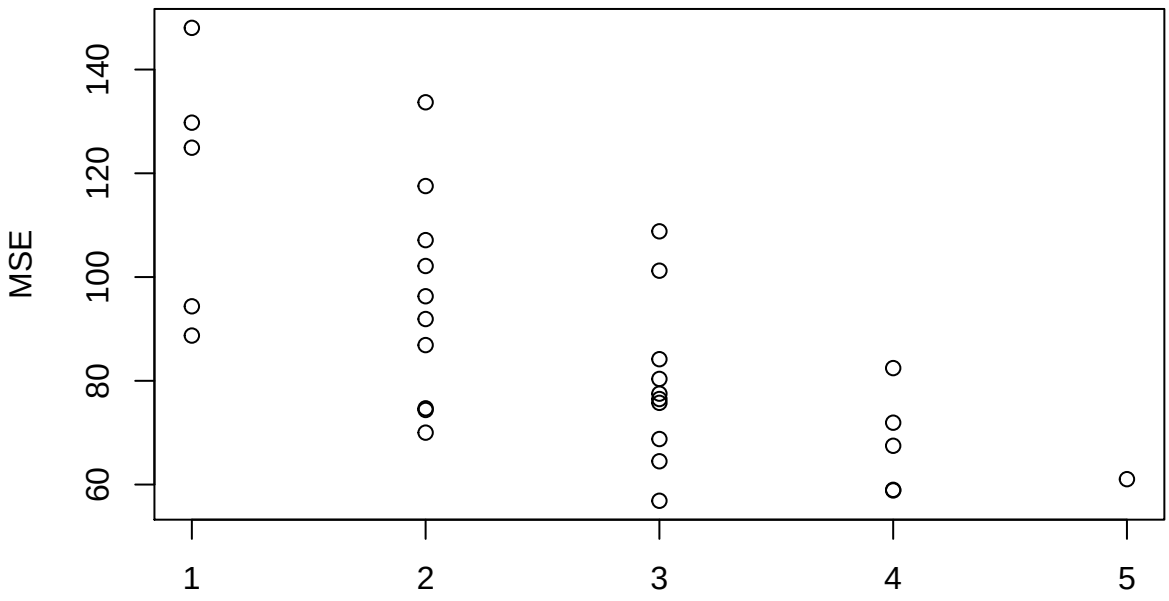
```
## [1] "Accuracy with K = 1: 0.532846715328467"
```

```
##    Y hat
## Y     0   1
##   0 560 371
##   1 269 170
```

You created the model using kknn(formula=Spam ~.,train = train,test = test,k = 30),
So you predict on the test set but then you use
kknn.fitted.results30 <- ifelse(results30$fitted.values > 0.5,1,0)
kknn.misClasificError30 <- mean(kknn.fitted.results30 != train$Spam)
which compares the prediction from test with the ones from train. So that is wrong.

What does K do then, how does in influence how your model generalises? How would you choose the optimal K?
What about overfitting? You did not answer the question what the decrease of K is doing.

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



Please provide a source for your beta estimates.
Slides, internet or derive it by hand.

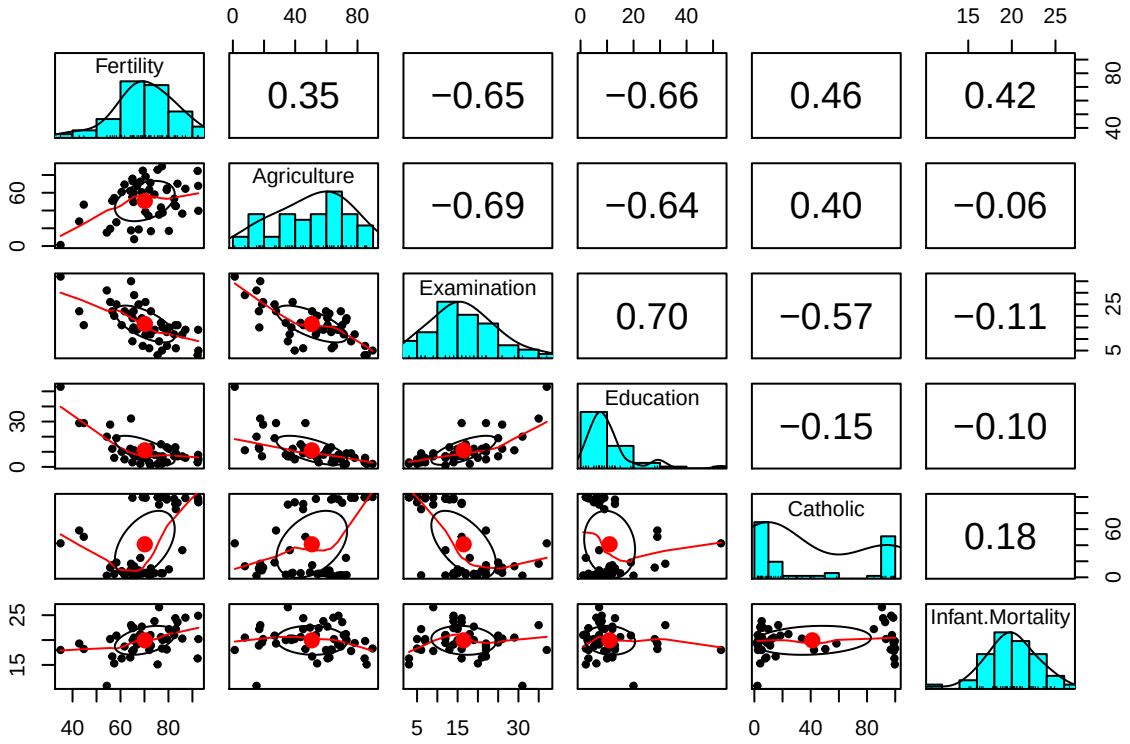
```
## $CV
## [1] 56.87859
##
## $Features
## [1] 0 0 1 1 1
```

Analysis:

• From the plot the cross-validation shows that accuracy of prediction is increasing with the number of features and the error at its minimum when 4 features are selected (Agriculture, Education, Catholic & Infant.Mortality), using all 5 features is a little worse but closely accurate, probably using more than 4 features start to show overfitting. That is contrary to your plot and values, where 3 features are selected.

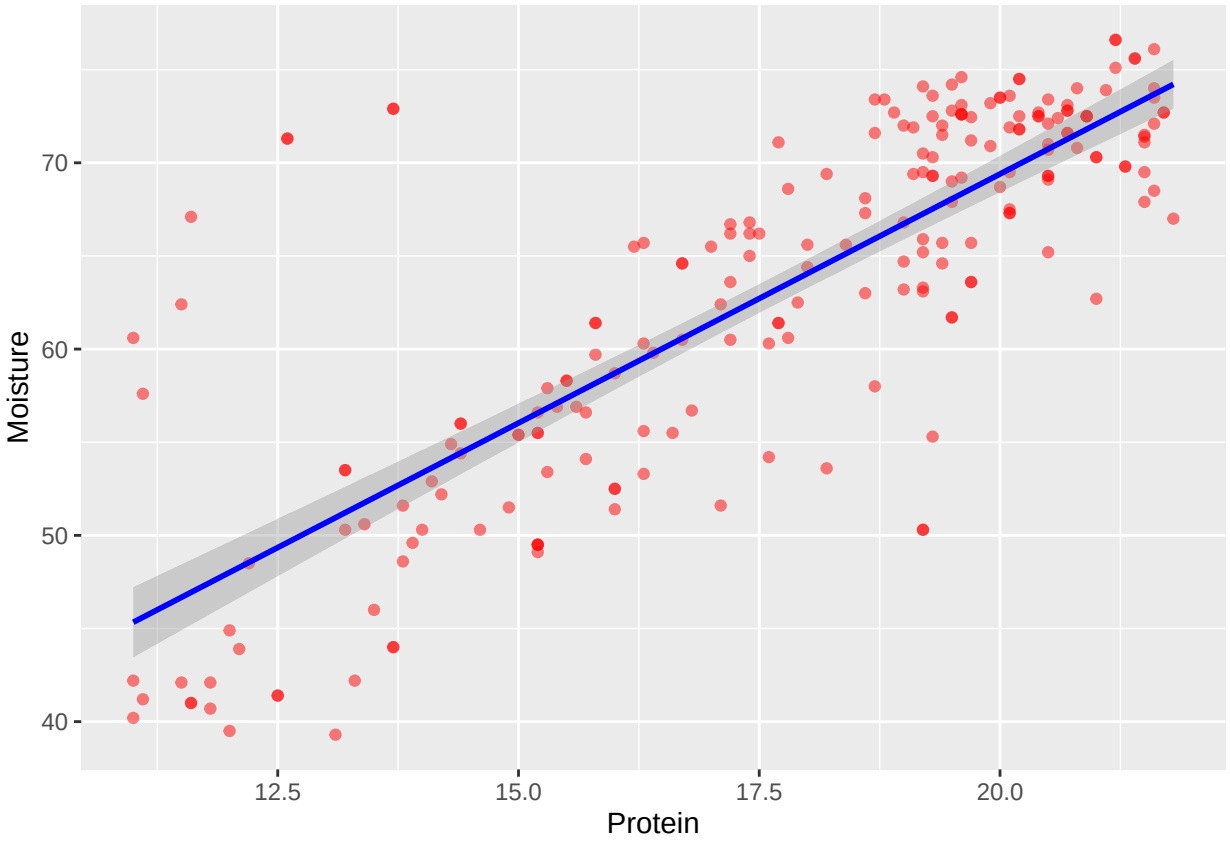
• In a different implementation inside our group we got the same selection of features but a slightly different value of the minimum MSE. It was 55.01898 instead of the 50.44948 calculated here. It is 56.87859

• To further analize the features of the data set we use the function pairs.panels from the package psych to plot an overview of the correlation between the features.



• However, from the single correlation between each of the features with the Fertility variable it is not clear why the cross-validation have selected these 4 features (Agriculture, Education, Catholic & Infant.Mortality) and dropped (Examination) as the best selection with least MSE, since Examination is more correlated (negative correlation) with Fertility than Agriculture or Infant.Mortality.

Assignment 4. Linear regression and regularization



Analysis:

- Yes, there is a clear linear relationship between Moisture and Protein.

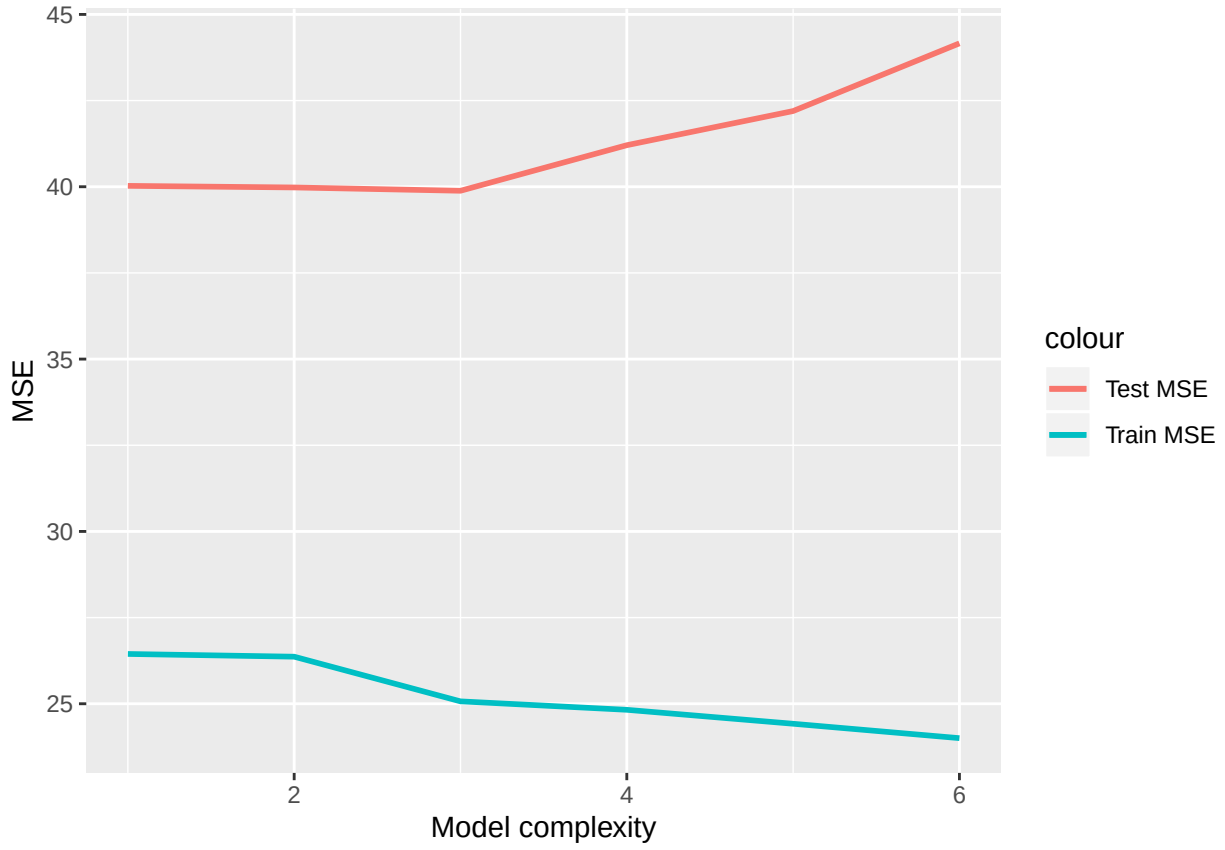
This is only partly correct, you didn't say anything about how the error is distributed. Also your model is Normal. You basically wrote down how your model would predict. This form is also not dependent on i (you used n, but I mean a general form)

- Probabilistic model : $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \varepsilon.$

- This model is a basis expansion of X_1 so all the variables are dependent on X_1, therefore it is appropriate to use MSE to eleminate these explanatory variables.

No. It is appropriate to use the MSE as it is the MLE of a Normal distribution. Here we assume that our error is normally distributed, that is why we take MSE.

```
##   train_MSE test_MSE
## 1     26.44    40.03
## 2     26.37    39.98
## 3     25.07    39.88
## 4     24.82    41.21
## 5     24.42    42.20
## 6     24.00    44.16
```



- The model with i = 3 is the best model because it have the least MSE when tested with test dataset.
  Most of the time, but I wouldn't necessarily say always.
- MSE values always decrease when increasing the complexity of the model and testing it with the training data set because overfitting and using train data as test data.
- When testing with the test data, the optimum value is reached when i = 3, after that it will be overfitted. Same as point 1).
- Because $MSE = Var(\hat{y}) + [Bias(\hat{y})]^2 + Var(\epsilon)$ and because that variance is inherently a nonnegative quantity, and squared bias is also nonnegative. Hence, we see that the expected test MSE can never lie below $Var(\epsilon)$, the irreducible error. Therefore, bias & variance are competetive meaning if it bias minimized variance will increase and vice versa. Good!

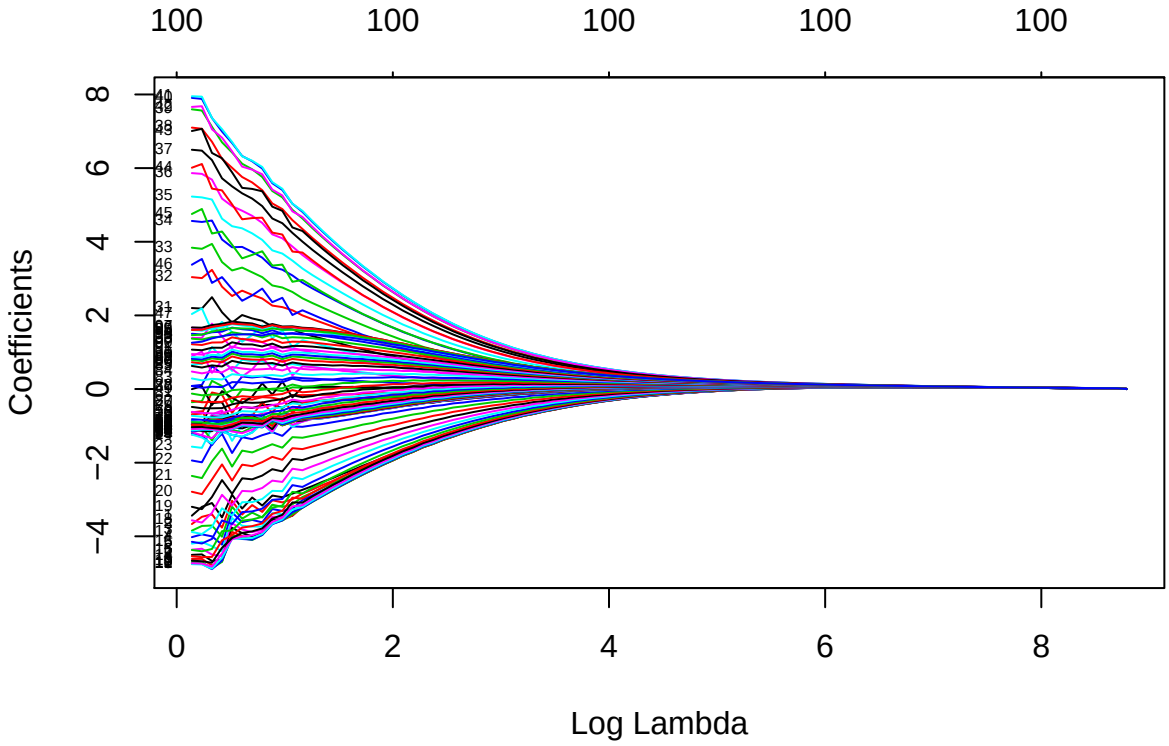
```
##                   [,1]
## Selected Channels   64
```

```
##                   [,1]
## Selected Channels   96
```

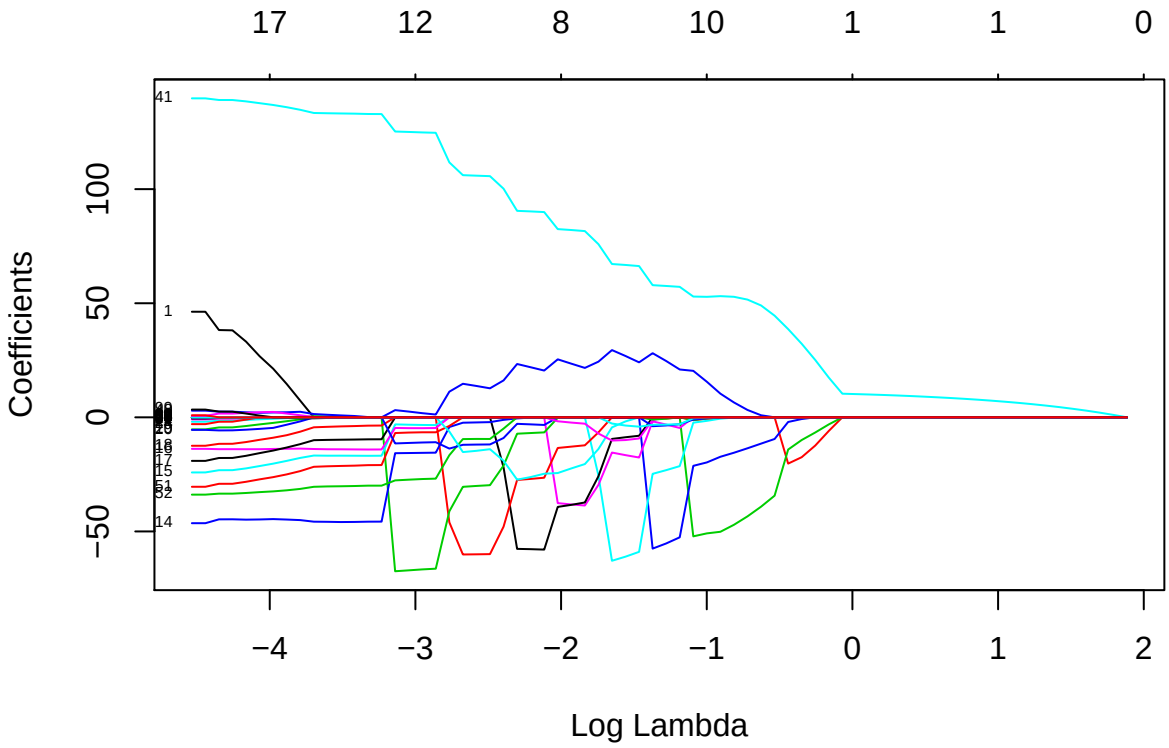
```
##       [,1]
## MSE 680.23
```

- When all data is used 64 channels where selected as the best model to predict Fat. stepAIC,

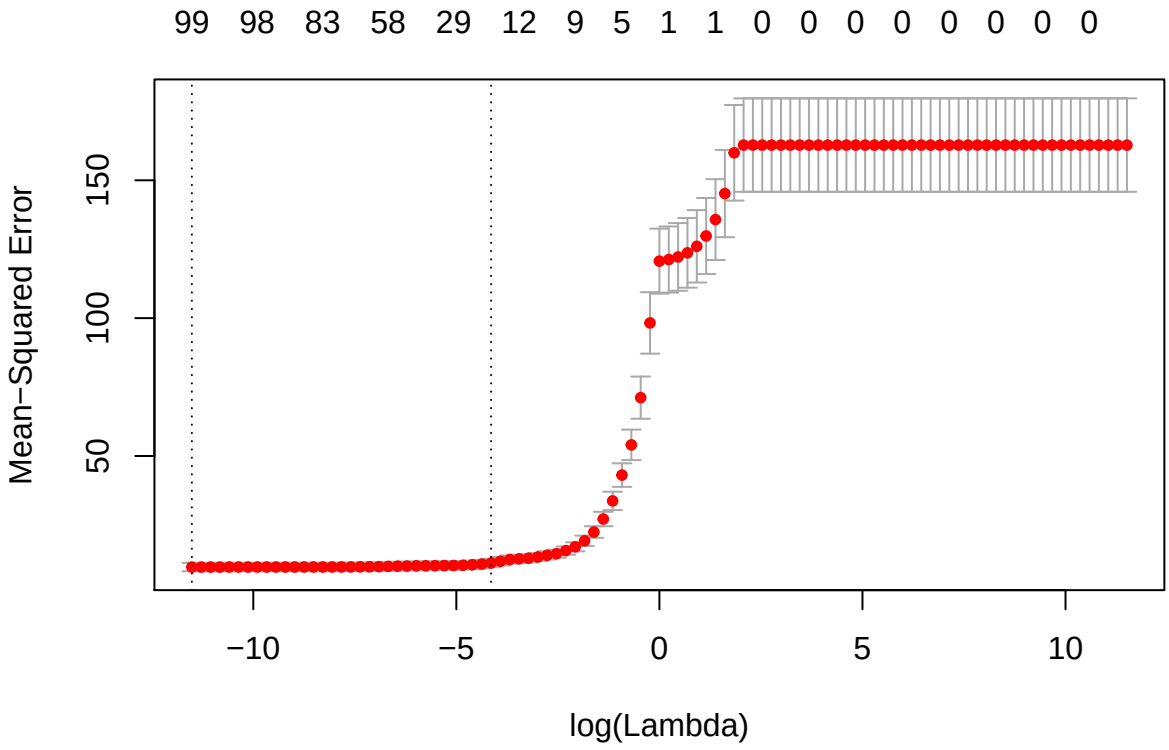
- A backward step wise selection (default) starts with the most complicated model (a model with all the 100 variables) and drops the least useful predictors
- The model with the minimum SSE is selected (the one with 64 channels in this case), however this might not be the ideal model because how stepAIC works it could drop a variable in the begining that could be part of the ideal model.
- When only training data is used 96 channels has been selected, and when this model is tested on the test dataset an MSE of 680.23 obtained.



- Lambda adds a penalty to the coefficients "excpet β_0", so the higher lambda is, the least significant variables will get close to 0.



- With Lasso, the least correlated variables will get eliminated completely (set to 0), until lambda reaches a value where all coefficients are eliminated except for the intercept. In this case, from the 100 variables (channels) we started with, channel# 41 was the last to get eliminated.




```
##                  [,1]
## Optimum Lambda 1e-05
```

```
##     [,1]
## MSE 9.71
```

Actually lambda = 0 is. You should've included it.

- 0.00001 is the optimal lambda that gives the minimum mean cross-validated error(for this particular C.V.).
- All the 100 variables have been selected with the optimal lambda, and as log lambdas(lambdas) grows larger the MSE also increases , however a close to minimum MSE can be acheived by only 20 variables.
- In stepAIC 64 variable where selected that could gave the minimum MSE, when these variables where used in a linear regression "without penalty" by using 2 datasets (1 for training and 1 for testing) a 680.23 MSE was obtained.
- When using 10 fold "default" C.V. on the lasso regression a much better MSE "9.71" was obtained, this is because C.V. utilize the data much more than stepAIC. However the actual comparision between stepAIC and C.V. is in the way they regularize the variables, using C.V. we can use a simpler model to obtain the obtimum result (20 variables from looking to the plot) while in stepAIC a much more complicated model where selected (64 variables)

APPENDIX

```r
##### Assignment 1

data = read.csv("Data/spambase.csv", header = TRUE)

n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
test=data[-id,]

model <- glm(Spam ~.,family=binomial(link='logit'),data=train)
#train data
fitted.results_train <- predict(model,newdata=train,type='response')
fitted.results_train <- ifelse(fitted.results_train > 0.5,1,0)
misClasificError_train <- mean(fitted.results_train != train$Spam)


#test data
fitted.results_test <- predict(model,newdata=test,type='response')
fitted.results_test <- ifelse(fitted.results_test > 0.5,1,0)
misClasificError_test <- mean(fitted.results_test != test$Spam)

#3. >0.8 with test data
fitted.results_0.8 <- predict(model,newdata=test,type='response')
fitted.results_0.8 <- ifelse(fitted.results_0.8 > 0.8,1,0)
misClasificError_0.8 <- mean(fitted.results_0.8 != test$Spam)

print(paste('Accuracy with test data with P > 0.5:',1-misClasificError_test))
table("Y"=test$Spam,"Y hat"=fitted.results_test)

print(paste('Accuracy with P > 0.8:',1-misClasificError_0.8))
table("Y"=test$Spam,"Y hat"=fitted.results_0.8)

library(kknn)




results30<-kknn(formula=Spam ~.,train = train,test = test,k = 30)
kknn.fitted.results30 <- ifelse(results30$fitted.values > 0.5,1,0)
kknn.misClasificError30 <- mean(kknn.fitted.results30 != train$Spam)

results1<-kknn(formula=Spam ~.,train = train,test = test,k = 1)
kknn.fitted.results1 <- ifelse(results1$fitted.values > 0.5,1,0)
kknn.misClasificError1 <- mean(kknn.fitted.results1 != train$Spam)

print(paste('Accuracy with K = 30:',1-kknn.misClasificError30))
table("Y"=train$Spam,"Y hat"=kknn.fitted.results30)
print(paste('Accuracy with K = 1:',1-kknn.misClasificError1))
table("Y"=train$Spam,"Y hat"=kknn.fitted.results1)
```

```
##### Assignment 3

#linear regression
mylin=function(X,Y, Xpred){
  Xpred1=cbind(1,Xpred)
  X=cbind(1,X)
  beta<-solve(t(X)%*%X)%*%t(X)%*%Y
  Res=Xpred1%*%beta
  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){
              iind<- ((k-1)*sF+1):(k*sF)
              modelb<-as.logical(model)
              Xm<-as.matrix(X1[,modelb])
              Xp<-as.matrix(Xm[-iind,])
              Xm<-as.matrix(Xm[iind,])
              Ytrain<-Y1[-iind]
              Yp<-Y1[iind]
              Ypred<-mylin(Xp,Ytrain, Xm)
              SSE=SSE+sum((Ypred-Yp)^2)
            }
            curr=curr+1
            MSE[curr]=SSE/n
            Nfeat[curr]=sum(model)
            Features[[curr]]=model
          }
  plot(Nfeat, MSE, xlab = "Number of features", ylab = "MSE")
  i=which.min(MSE)
  return(list(CV=MSE[i], Features=Features[[i]]))
}

data("swiss")
```

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

library(psych)
pairs.panels(swiss)



##### Assignment 4
library(ggplot2)
library(glmnet)
library(MASS)

tecator <- read.csv("Data/tecator.csv", header = TRUE)

ggplot(tecator, aes(x = Protein, y = Moisture)) +
  geom_point(alpha = 0.5, color = 'red') +
  geom_smooth(method = "lm", color = "blue")

n=dim(tecator)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=tecator[id,]
test=tecator[-id,]

M <- function(data,power) {
  MSE <- matrix(0,power,1)

  for (i in 1:power) {
    model <- lm(Moisture ~ poly(Protein,i), data = train)
    pred  <- predict(model, data)
    MSE[i,] <- mean((data$Moisture - pred)^2)
    }

  return(MSE)
}
train_MSE <- M(train,6)
test_MSE  <- M(test,6)


df = data.frame(train_MSE,test_MSE)

round(df,2)

ggplot(df) +
  geom_line(aes(x = 1:6, y = train_MSE, color = "train_MSE"), size = 1) +
  geom_line(aes(x = 1:6, y = test_MSE, color = "test_MSE"), size = 1) +
  ylab("MSE") +
  xlab("Model complexity")

#Using stepAIC on all data
channels_all_data   <- tecator[,2:101]
Fat_model_all_data  <- lm(tecator$Fat ~ ., channels_all_data)
best_fit_all_data   <- stepAIC(Fat_model_all_data, trace=FALSE)
```

```r
as.matrix(c("Selected Channels"= length(best_fit_all_data$coefficients)))

#Using stepAIC on training data only
channels_train_data   <- train[,2:101]
Fat_model_train_data  <- lm(train$Fat ~ ., channels_train_data)
best_fit_train_data   <- stepAIC(Fat_model_train_data, trace=FALSE)

as.matrix(c("Selected Channels"= length(best_fit_train_data$coefficients)))

#Testing stepAIC model on the test data
AIC_model  <- lm(formula(best_fit_train_data), channels_train_data)
pred  <- predict(AIC_model, test)
MSE   <- as.matrix(c("MSE" = mean((test$Fat - pred)^2)))
round(MSE,2)


ridge <- glmnet(x = as.matrix(tecator[,2:101]),
                y = as.matrix(tecator$Fat),
                alpha = 0,
                family = "gaussian")

plot(ridge, xvar="lambda", label=TRUE)

lasso <- glmnet(x = as.matrix(tecator[,2:101]),
                y = as.matrix(tecator$Fat),
                alpha = 1,
                family = "gaussian")

plot(lasso, xvar="lambda", label=TRUE)

set.seed(12345)
lambda_selection <- 10^seq(-5,5,0.1)
lasso_cv         <- cv.glmnet(x = as.matrix(tecator[,2:101]),
                       y = as.matrix(tecator$Fat),
                       alpha=1,
                       family="gaussian",
                       lambda = lambda_selection)

plot(lasso_cv)
as.matrix(c("Optimum Lambda" = lasso_cv$lambda.min))
round(as.matrix(c("MSE" = lasso_cv$cvm[which.min(lasso_cv$cvm)])),2)
```

References

https://www.r-bloggers.com/how-to-perform-a-logistic-regression-in-r/

https://stats.stackexchange.com/questions/306267/is-mse-decreasing-with-increasing-number-of-explanatory-variables

https://cran.r-project.org/web/packages/glmnet/vignettes/glmnet.pdf