Introduction to Machine Learning

Lab 2

Anton Persson, Emil Klasson Svensson, Mattias Karlsson, Rasmus Holm

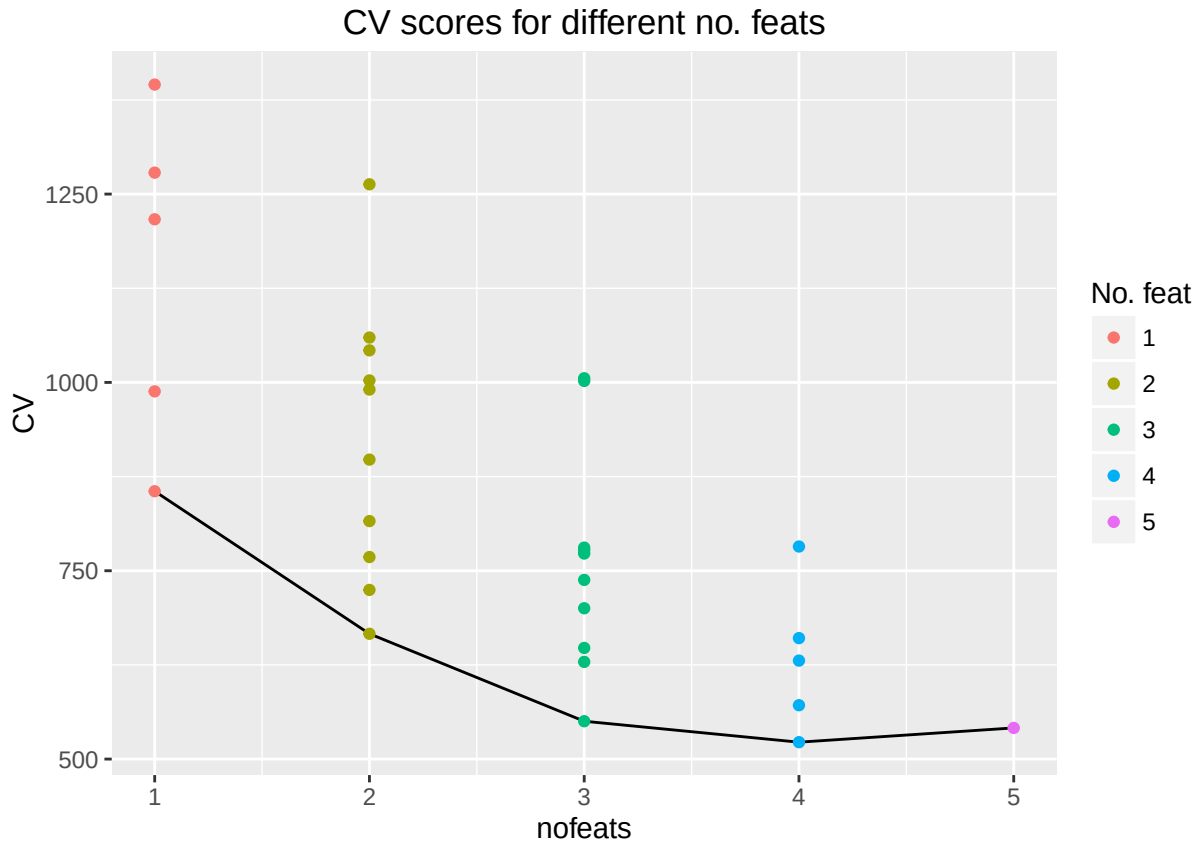
2016-11-15

Contents

Assignment 1

2

Plot the CV scores computed for various feature subsets against the number of features Report the resulting plot and interpret it. Report the optimal subset of features and comment whether it is reasonable that these specific features have largest impact on the target.




```
#>                    Model nofeats       CV
#> 12 Y ~ X1 + X3 + X4 + X5       4 522.4022
```

For the plot above we conclude that the model with all five features, one of the models with four features and one of the models with three features are the best options, and they are almost equally good. If you look closely, you can see that the lowest CV-score seems to belong to one of the four features models. The models with ony one explanatory variable are worse models than the other, which is valid for the models with two features as well. The plot shows how both the variation and the mean of the CV score decreases when the number of features increases.

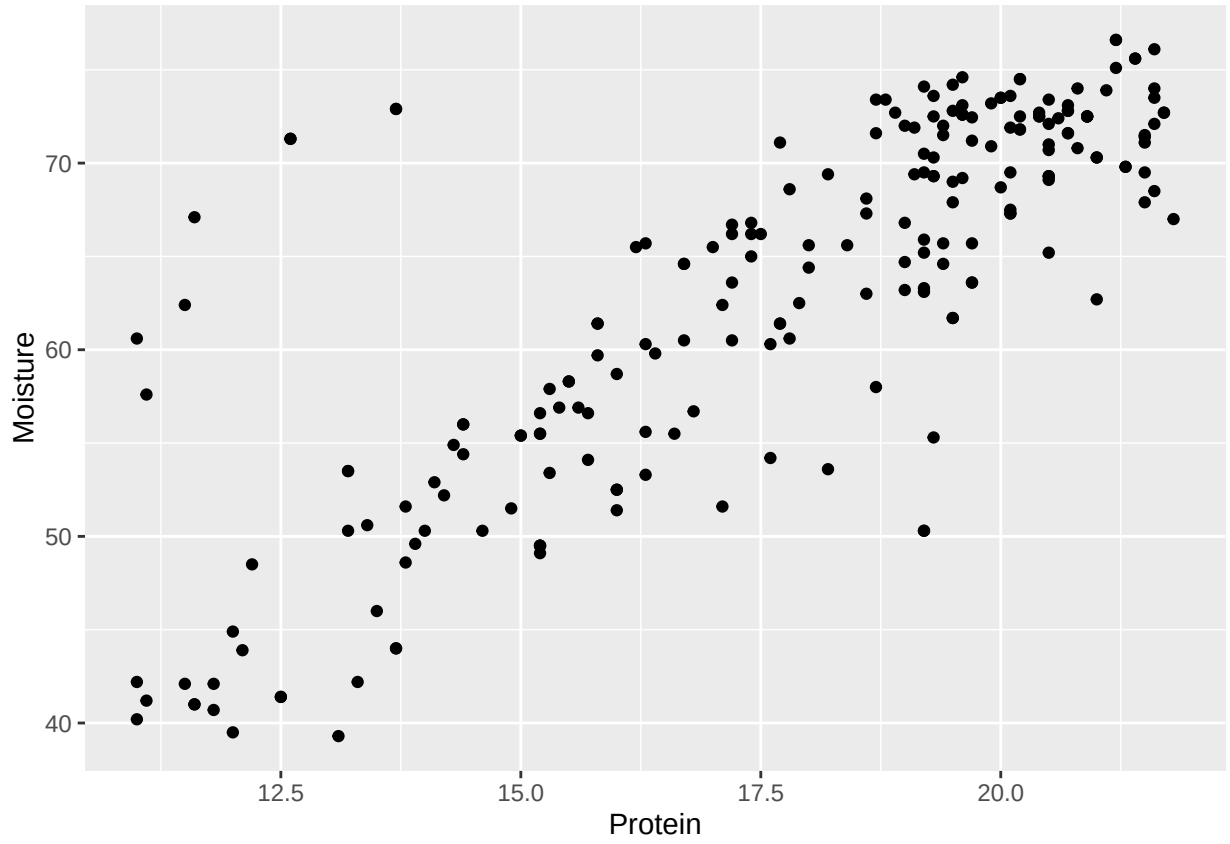
The best model is a model with four features, with the features X1, X3, X4 and X5. That is the variables *Agriculture, Education, Catholic* and *Infant Mortality*. The variable *Fertility* is the response variable. We think it is a reasonable result, even though we have a hard time describing why the share of males involved in the agriculture should impact the fertility. But probably in general the chosen variables serves as good indicators for well-being in the towns which could impact the fertility rate. The share of Catholics in the town could be explained by their belief that various methods of birth-control is some kind of a sin.

Assignment 2

1

Create a plot of Moisture versus Protein. Do you think that these data are described well by a linear model?

A linear model with Protein as a predictor would be a good fit to the variable Moisture. There are some outliers that could affect the fit in a negative way but in general we think that a linear model fit the data well.



2

Why is it appropriate to use MSE criterion when fitting this model to a training data? Report a probabilistic model that describes M_i

In this polynomial model we have that the target is $y = Moisture$, the response is $x = Protein$, and we assume that

$$y \sim \mathcal{N}(\mu, \sigma^2).$$

We know from the law of large numbers that the mean can be approximated by

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

and the mean squared error (MSE) of an estimator $\hat{\mu}$ is

$$\mathrm{MSE}(\hat{\mu}) = \mathrm{var}(\hat{\mu}) + [\mathrm{bias}(\hat{\mu})]^2 .$$

It turns out that

$$\mathrm{MSE}(\hat{\mu}) = \mathrm{E}\left[(\hat{\mu} - \mu)^2\right] = \frac{\sigma^2}{n}$$

is the best unbiased estimator for a Gaussian distribution, i.e. having the lowest MSE out of all unbiased estimators. That is the reason why it is reasonable to use the MSE to fit our model to the training data. The model will look like

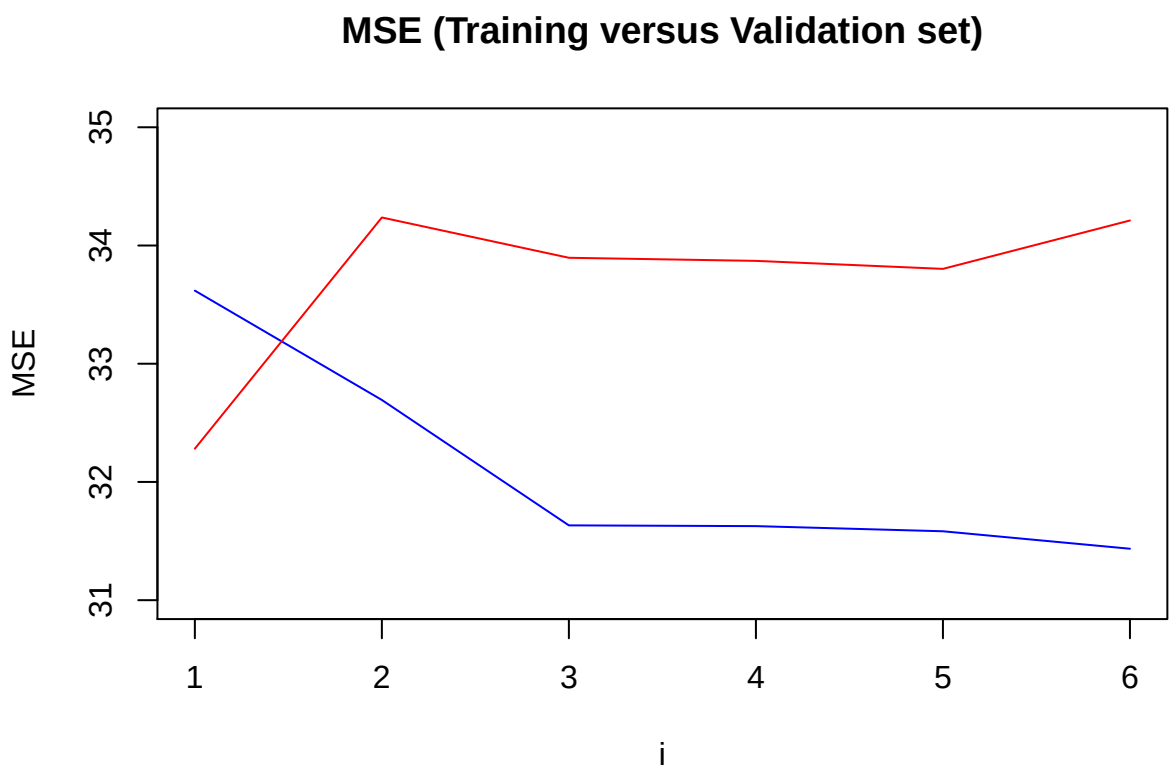
$$\mathrm{E}[y(x, w)] = \sum_{i=0}^{p} w_i \phi_i(x)$$

where $\phi_i(x) = x^i$.

3

Fit models $M_i, i = 1...6$. For each model, record the training and the validation MSE and present a plot showing how training and validation MSE depend on i.

The blue line represents the MSE for the training data and the red represents MSE for the validation set.



The plot above indicates that the linear model without higher degree of polynomials is the best. The model with a first degree polynomial has the lowest MSE for the test set. The MSE for all other powers of the polynom is worse for the test data. Each of the added polynomial terms increases the overfitting of the model. The MSE for the training dataset decreases as we add more terms, which is reasonable.

4

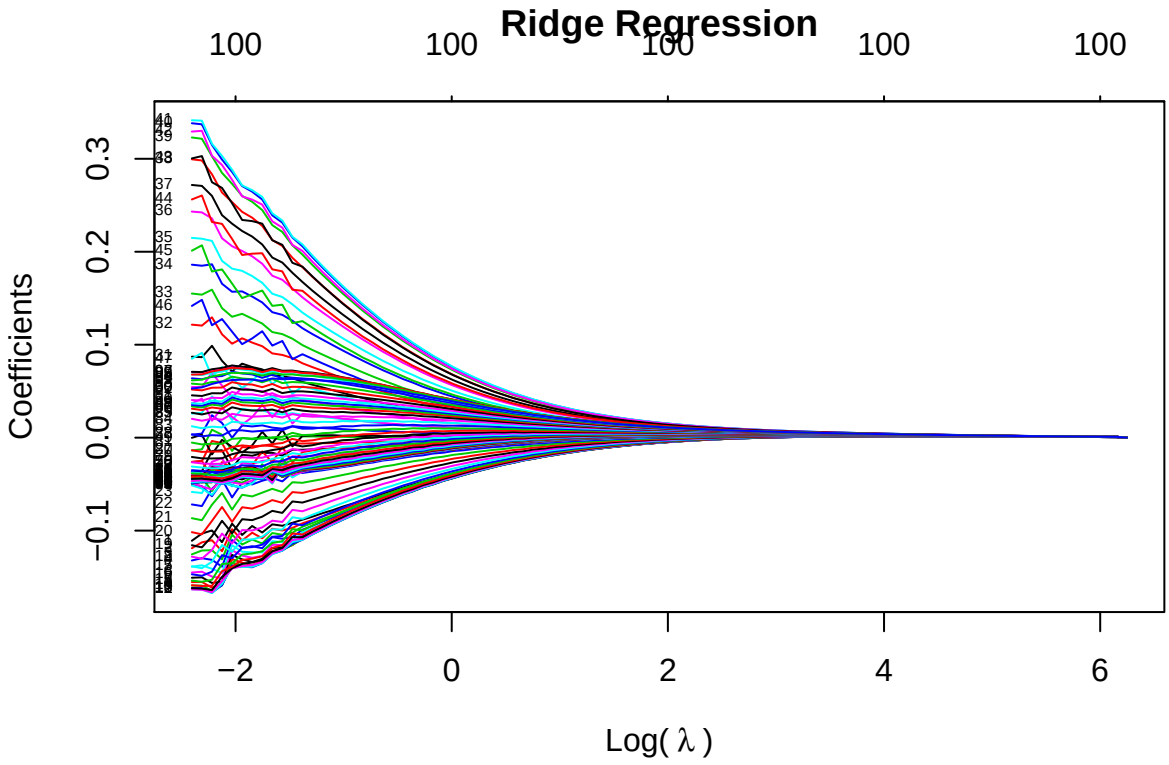
Comment on how many variables were selected.

By using the Akaike information criterion (AIC) we got that the optimal model contains 63 response variables excluding the intercept.

5

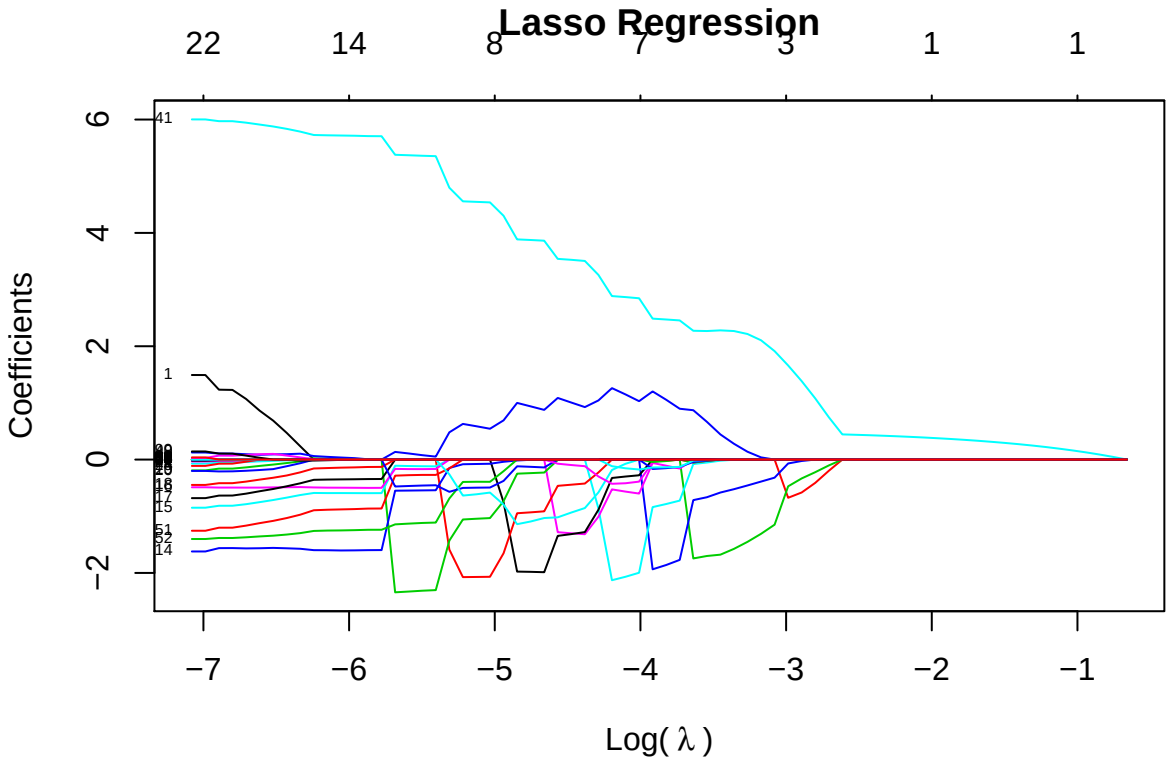
Present a plot showing how model coefficients depend on the log of the penalty factor λ and report how the coefficients change with λ

As λ increases the coefficients approaches zero. We can see that the coefficients decrease at different rates which indicate that those do not contribute as much to the model. We can also see that the number of features used are uniform for all lambdas, i.e. 100 in this case, which is to be expected from the regularization term used by ridge regression.



6

In lasso regression, the λ changes the values of the coefficients differently from ridge regression. The coefficients can both increase and decrease as λ increases but coefficient values converges to exactly zero, i.e. feature selection, which is not true for ridge regression.

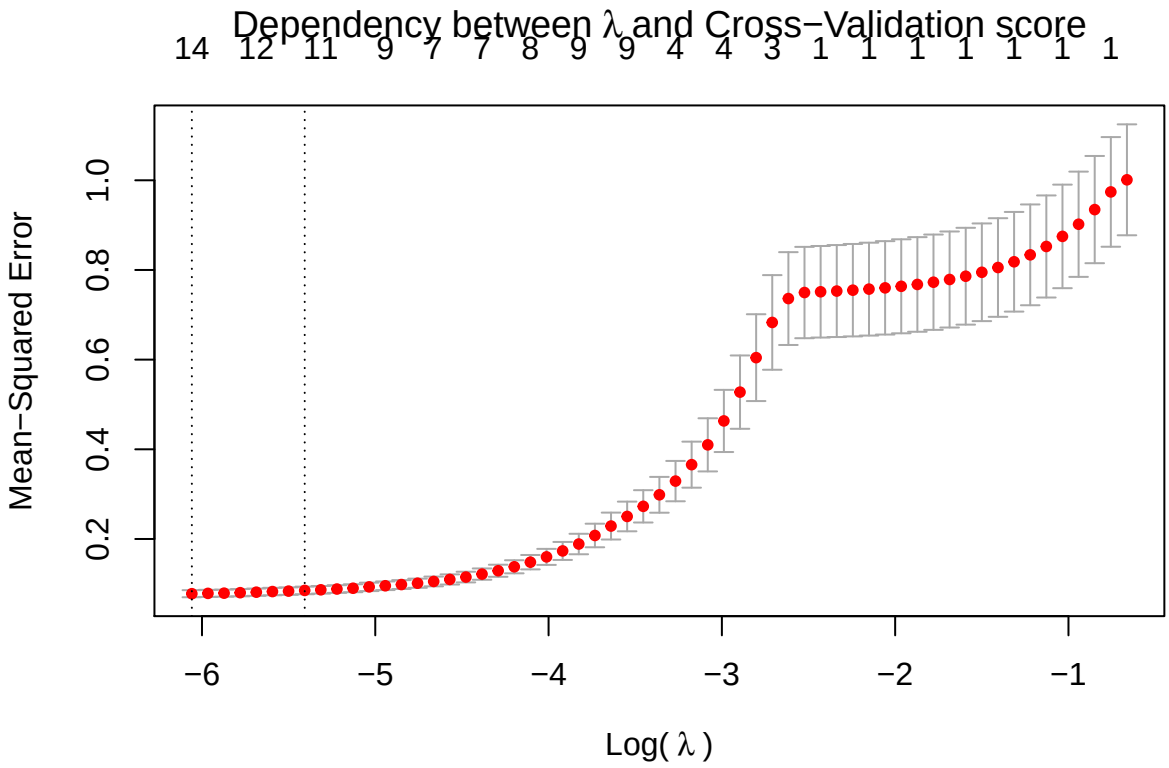


7

Use cross-validation to find the optimal LASSO model, report the optimal λ and how many variables were chosen by the model and make conclusions.

The optimal $\lambda = 0.002338$ for lasso regression resulted in 14 variables and was found by cross-validation using 10-folds. We can see that the lasso regularization term punishes complex models and since the lambda is relatively low in this case it seems that many of the variables are redundant.

As λ increases the CV-score increases exponentially up to the point where the model contains one variable. Then it restarts the CV-score increase at a different rate.



8

Compare the results from steps 4 and 7.

By using lasso regression, the optimal number of features are significantly less than found by AIC (14 compared to 63). This indicates that lasso regression penalize complex models more compared to AIC.

Appendix

Code for Assignment 1

```
cvLM <- function(Y, X, Nfolds){
  library(ggplot2)
  # DEFINING THE FUNCTION LINREG, FOR FITTING LINEAR MODELS

  linreg<-function(formula,data){
    formula <- formula(formula)
    des.mat <- model.matrix(formula , data) #Extracts the model matrix
    dep.var <- all.vars(formula)[1]         #Extracts the name of the y-variable
    dep.var <- as.matrix(data[dep.var])     #Extracts the data of the y-variable
                                            #and overwrites it with the data-colum

    #Calculating the beta coeffs. (X' %*% X)^-1 %*% X' %*% y
    beta.hat <- solve( t(des.mat) %*% des.mat )  %*% t(des.mat) %*% dep.var

    # Calculating the y-hat  , y_hat = X %*% beta_hat
    y.hat <- des.mat %*% beta.hat

    #Calculating the residuals e= y- y_hat
    res.err <- dep.var - y.hat

    l<-list( beta.hat = beta.hat, y.hat = y.hat, res.err = res.err)
    return(l)
  }


  #GENERATING ALL POSSIBLE PERMUTATIONS OF MODELS

  #Get the colnames for the X-variables
  q<-rep(paste0("X",c(1:5)))

  #Merge the data in to one data set and naming the columns
  myData <- cbind(Y,X)
  colnames(myData)<- c("Y",q)

  #Generating all possible combinations
  myComb<-sapply(c(1:5), FUN = combn, x = q )
  #Creating the vector that will hold all formulas
  myformula <- c(myComb[[1]])

  #Extracting the combinations of 2 and 3 X-variables and adding a + between them
  for (i in 2:3){
    for (j in 1:10){
      myformula[length(myformula)+1]<-(paste(myComb[[i]][,j],collapse = " + "))
    }
  }

  #Heres two rows that could replace the above for-loop
  #sapply(2:3, FUN = function(i) sapply(1:10, FUN = function(j)
  #paste(myComb[[i]][,j], collapse = " + " ) ) )
```

```
 #Extracting the combinations of 4 and 5 X-variables and adding a + between them
 #This is basicly a loop for the 4 combinations
 myformula <-c(myformula ,
               sapply(1:5, FUN =function(X)
                 paste(myComb[[4]][,X],collapse = " + " )
                 ),  paste(myComb[[5]],collapse = " + ")
               )

myformula<- paste("Y","~",myformula)

#### SPLITTING AND SUBSETING DATA IN TO K FOLDS

#calculatin no. rows
noobs<-dim(myData)[1]
K <- Nfolds

#Use sample to randomly draw the indexes of the dataset
#and reorder the data with them in a random manner
set.seed(12345)
myData<-myData[sample(noobs),]

#Create K equal indexes that are added to the data.
cut(1:noobs,breaks=K,labels = FALSE)

myData$index <- cut(1:noobs,breaks=K,labels = FALSE)


#init a counting vector "o" useed to loop in to the data.frame "linearModels"
#and a combination index "dataKombs" used for subseting the different datasets
#used for fitting models
o <- 1
linearModels<-data.frame(CV=1,model="text",nofeats=1,stringsAsFactors = FALSE)
dataKombs <- combn(1:K,K-1)
for (m in 1:length(myformula)){
for (l in (1:K)){

  #the data of the K-folds used for the model estimation
  data<-subset(myData, myData$index %in% dataKombs[,l] )


  #the fold that was left out in the model estimation
  predmatrix<-model.matrix(formula(myformula[m]),
            subset(myData, !(myData$index %in% dataKombs[,l] )))

  #Calculating the CV score for each model. sum((Y - Y(hat))^2)
  CV<-sum(
    (
      #this is the observed Y for the left out fold.
      subset(myData, !(myData$index %in% dataKombs[,l] ))[,1] -
       #predmatrix description above.
       predmatrix %*%
       #the estimated beta-hats.
```

```
        linreg(formula = myformula[m], data = data)$beta.hat
       )^2
     )

   #inserting the results in to the linearModels data.frame
   linearModels[o,] <- c(CV,myformula[m],ncol(predmatrix) - 1)
   o <- o + 1
    }
 }
 #reforming data to numeric again.
 linearModels[,1] <- as.numeric( linearModels[,1])
 linearModels[,3] <- as.numeric( linearModels[,3])

 #The mean for the different models, each model is estimadet K times
 plotdata<-suppressWarnings(aggregate(linearModels,by = list(linearModels$model),FUN= mean)[,-3])

 #renaming a column to ease plotting
 colnames(plotdata)[1] <- "Model"

 #plotting
 engr<-ggplot() +
   geom_line(data = aggregate(plotdata,list(plotdata$nofeats),FUN = min)[,c(3,4)],aes(x=nofeats,y=CV)) +
   geom_point(data = plotdata,aes(x=nofeats,y=CV,col = factor(nofeats)) ) +
       labs(title = "CV scores for different no. feats",color = "No. feat")


#displays the plot
plot(engr)

 #Returns the models with the lowest average CV-score
 return( plotdata[min(plotdata$CV) == plotdata$CV,c(1,3,2)])

}

cvLM(Y = swiss[,1],
     X = swiss[ , 2:ncol(swiss) ],
     Nfolds = 5 )
```

Code for Assignment 2

```
library(readxl)

tecator <- read_excel("../data/tecator.xlsx")

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

model <- function(train, test, power){
```

```
  result <- data.frame("ID" = 1:6,"Train" = NA, "Validation" = NA)

  for(i in power){
    # Formula

    fml <- Moisture ~ poly(x = Protein,degree = power[i],raw = TRUE)

    # Polynomial of i = power

    model <- lm(formula = fml, data = train)

    prediction <- predict(model, train)

    # MSE for training set

    MSE_train <- sum( (train$Moisture-prediction)^2 ) / nrow(train)

    # MSE for validation set

    prediction_validation <- predict(model, test)

    MSE_validation <- sum( (test$Moisture-prediction_validation)^2 ) / nrow(test)

    result$Validation[i] <- MSE_validation
    result$Train[i] <- MSE_train
  }
  return(result)
}

result <- model(train,test,c(1:6))

plot(x = result$ID, y = result$Train,
     main = "MSE (Training versus Validation set)", xlab ="i",
     ylab = "MSE",type = "l", col = "blue", ylim = c(31,35))
lines(x = result$ID, y = result$Validation, col = "red", type = "l")

trainAll=tecator[,c(-1,-103,-104)]

model <- lm(Fat ~.,data = trainAll)
modelAIC <- MASS::stepAIC(object = model,direction = 'both',trace=FALSE)
modelPara <- length(modelAIC$coefficients)-1

trainAllScale <- scale(trainAll)
modelRidge <- glmnet::glmnet(x = as.matrix(trainAllScale[,-101]),
                             y = trainAllScale[,101],
                             family = "gaussian",
                             alpha = 0)

plot(modelRidge, xvar = "lambda",
     main = "Ridge Regression", label = TRUE,
     xlab = bquote(plain("Log(")~lambda~plain(")")))

modelLasso <- glmnet::glmnet(x = as.matrix(trainAllScale[,-101]),
```

```
                              y = trainAllScale[,101],
                              family = "gaussian",
                              alpha = 1)

plot(modelLasso, xvar = "lambda",
     main = "Lasso Regression", label = TRUE,
     xlab = bquote(plain("Log(")~lambda~plain(")")))

set.seed(12345)
modelLassoCV <- glmnet::cv.glmnet(x = as.matrix(trainAllScale[,-101]),
                                  y = trainAllScale[,101],
                                  family = "gaussian",
                                  alpha = 1,standardize = FALSE)
nVar <- length(coef(modelLassoCV, s = 'lambda.min')@x)-1

plotMain <- bquote(plain("Dependency between") ~ lambda ~ plain("\nand Cross-Validation score"))

plot(modelLassoCV, main = plotMain,
     xlab = bquote(plain("Log(")~lambda~plain(")")))

modelAICPredict <- predict(modelAIC)

modelAICPredict1 <- ( sum( (trainAll[,101]-modelAICPredict)^2) ) /nrow(trainAll)

modelLassoCVPredict <- predict(modelLassoCV, newx = as.matrix(trainAllScale[,-101]), s="lambda.min")

modelLassoCVPredict1 <- ( sum( (trainAll[,101]-modelLassoCVPredict)^2 ) )/nrow(trainAll)

modelLassoCVPredict1 <- round(modelLassoCVPredict1,4)
```

Contributions

We divided the work into two parts and discussed/compiled the results in pairs. Then we all discussed our findings together as a whole group and checked that everyone had similar/understood the results.