Lab1 Report TDDE01

Karol Wojtulewicz

Assignment 1: Spam classification with nearest neighbors

For this task we first import the data from an excel file spambase.xlsx into the RStudio using following function:

```
my_data = readxl::read_excel(path = "/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab1/Assig
```

Later as suggested in the lab description we divide the datafile into two chunks, one for training and one for testing:

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

For the logistic regression we use (as suggested in the lab description) "glm()" function and get the following:

```
fit = glm(Spam~.,family=binomial(link='logit'),data=train)
```

Now, after the model is trained, we can test it with the predict function and later see how it performed with the confusion matrix. In the Assignment 1.2 the goal was to see test the data with the classification principle, where p>0.5 ?? 1 : 0. After using this principle we can show the confusion matrix for the results as well as print the missclassification error and accuracy.

```
results.train = predict(fit,train, type = "response")
results = predict(fit,test, type = "response")
results.train05 = ifelse(results.train > 0.5,1,0) #Assignment 1.2a
results05 = ifelse(results > 0.5,1,0) #Assignment 1.2b
```

Assignment 1.2a confusion matrix:

```
##
##                   Pred not spam Pred spam
##   Actual not spam           803       142
##   Actual spam                81       344
```

```
## [1] "Missclassification error train 0.162773722627737"
```

```
## [1] "Accuracy train  0.837226277372263"
```

Assignment 1.2b confusion matrix:

```
##
##                   Pred not spam Pred spam
##   Actual not spam           791       146
##   Actual spam                97       336
```

```
## [1] "Missclassification error test 0.177372262773723"
```

```
## [1] "Accuracy test  0.822627737226277"
```

The obtained accuracy is around 80% for both train and test sets as well as the missclassification is lower for the predictions on the training data than test data. This is a very good result, because it is not overfitting to the traing data, but it is still highet than the results for predicting on the test data. The confusion matrix

gives a bit better insight into the results. Here we can see something interesting, the precision for the spam is not that high (around 70% for both train and test),

```
## [1] "Precission train:  0.707818930041152"
```

```
## [1] "Precission test:  0.697095435684647"
```

and tells us, that the model will classify some emails as a spam even if they're not. This could be a "deal breaker" for the classifier as we want that value to be as high in procentage as possible.

In the assignment 1.3 the task was to do the same thing as in assignment 1.2, but with the classification principle, where p>0.9 ?? 1 : 0. The result can be seen below.

```
results.train09 = ifelse(results.train > 0.9,1,0) #Assignment 1.2a
results09 = ifelse(results > 0.9,1,0) #Assignment 1.2b
```

Assignment 1.3a confusion matrix:

```
##
##                   Pred not spam Pred spam
##   Actual not spam           944         1
##   Actual spam               419         6
```

```
## [1] "Missclassification error train 0.306569343065693"
```

```
## [1] "Accuracy train  0.693430656934306"
```

Assignment 1.3a confusion matrix:

```
##
##                   Pred not spam Pred spam
##   Actual not spam           936         1
##   Actual spam               427         6
```

```
## [1] "Missclassification error test 0.312408759124088"
```

```
## [1] "Accuracy test  0.687591240875912"
```

In this case, the accuracy went down but even more interesting results can be observed in the confusion matrix. There can observe quite different story to the 1.2. The recall (true positives/(false negatives + true positives)) is extreamly low for both train and test datasets. This implies, that the model is likely to classify an actual spam as a normal email, which would be equal to flooding the mailbox with trash mail.

```
## [1] "Recall train:  0.0141176470588235"
```

```
## [1] "Recall test:  0.0138568129330254"
```

The last two subassignments we need to use a library "kknn" as suggested in the lab documentation. K-nn is a nonparametric model and uses the training data for classifying new data. First we use the "kknn" function to build the model for both train and test data.

```
modelknn.train30 <- kknn(Spam~., train, train, k=30)
modelknn.test30 <- kknn(Spam~., train, test, k=30)
result.train30 <- floor(fitted(modelknn.train30) + 0.5)
result.test30 <- floor(fitted(modelknn.test30) + 0.5)
misClasificError.kknn30.train = mean(result.train30 != train$Spam)
misClasificError.kknn30.test = mean(result.test30 != test$Spam)
```

Here, the objective was to print the missclassification rate and compare it with the results found in step 2. As we can see below, the result of predicting the train data is very simmilar to the one found in the assignment 1.2 for both the train and test datasets. However if we look at the results from testing the test dataset, we find, that the results here vary quite a lot. The misclassification rate here is almost double as high, which means poor classification quality.

```
## [1] "Misclassification K = 30, train, train 0.172262773722628"
```

```
## [1] "Misclassification K = 30, train, test 0.32992700729927"
```

Next objective was to create a KNN model with the k=1. It is worth noting that this low of a k value can cause noise and overfitting of the model and this is exactly what happens when we test the model on the training dataset.

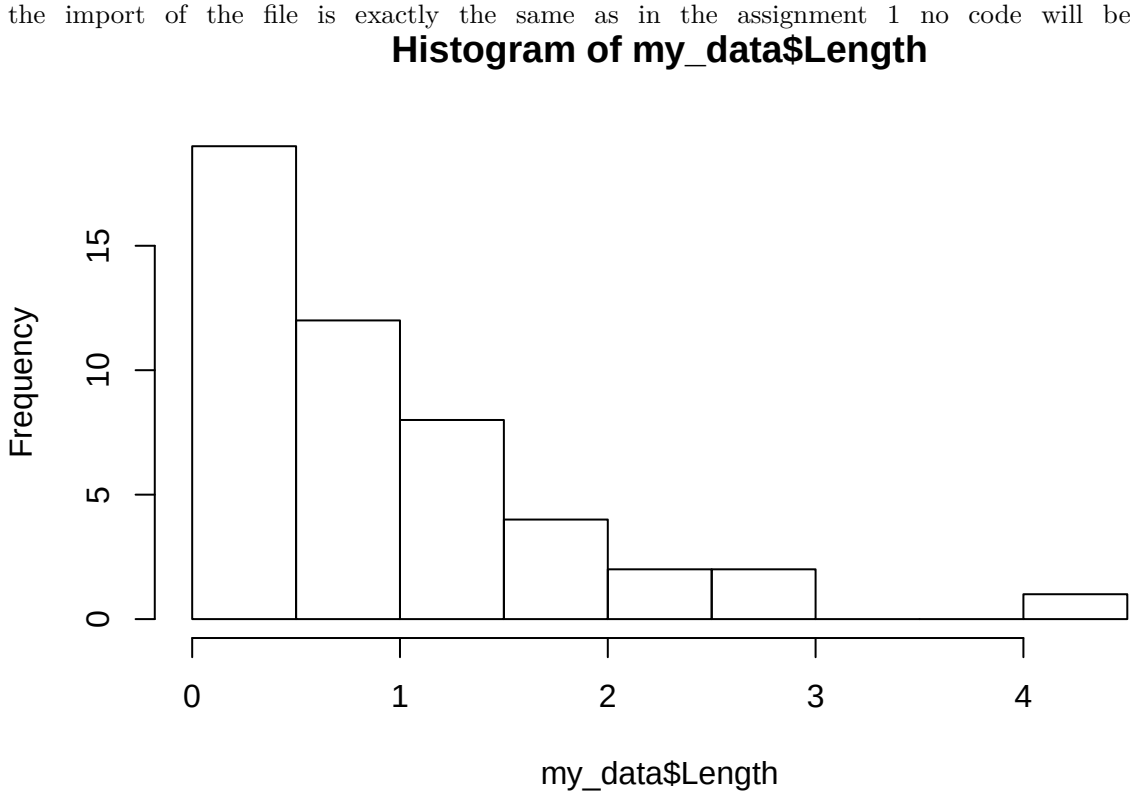
Here the results show misclassification rate of 0 for the training dataset, that show, that the model is overfitted for the dataset. This is later proven, by the very poor results of the testing on the test dataset, that yields even poorer misclassification rate, than with k=30 at around 35%.

```
## [1] "Misclassification K = 1, train, train 0"
```

```
## [1] "Misclassification K = 1, train, train 0.345985401459854"
```

Assignment 2: Inference about lifetime of machines

In this assignment we will use a datafile provided in the lab called machines.xlsx. Because the import of the file is exactly the same as in the assignment 1 no code will be provided.

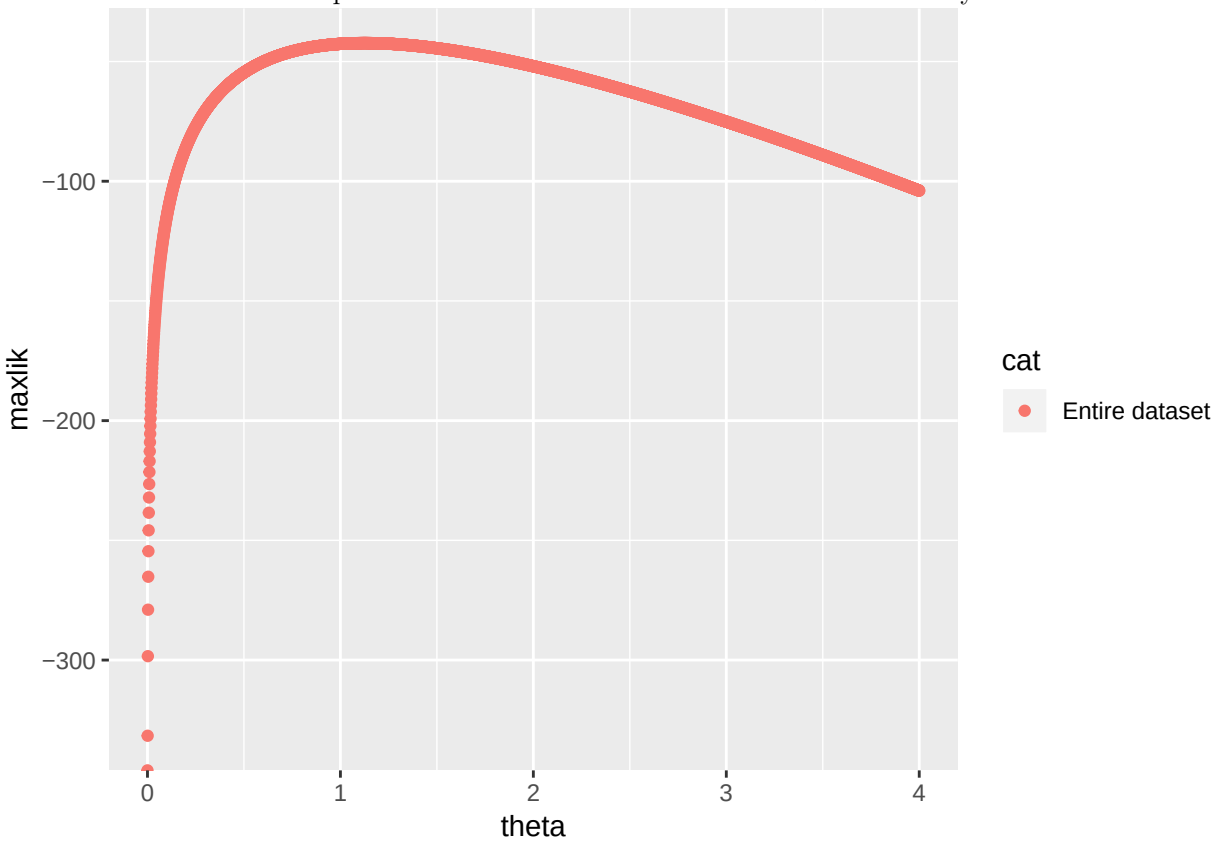


We start of with describing the distribution type of x ($\mathrm{p}(x \mid \theta) = \theta e^{-\theta x}$) as exponential distribution. Next task is to write a function that computes the log-likelihood $\log(\mathrm{p}(x \mid \theta))$. We start with defining the function with two input varialbes, one for x and one for theta. Later we create a return vector theta.return for returning the results. Lastly we do a for loop, that loops through all of the theta values to show how the theta varies.

```
logLikelihood = function(x, theta){

  theta.return = 1:length(theta)

  for(a in 1 : length(theta)){
    probVector = theta[a]*exp(-theta[a]*x)
    theta.return[a] = log(prod(probVector))
```

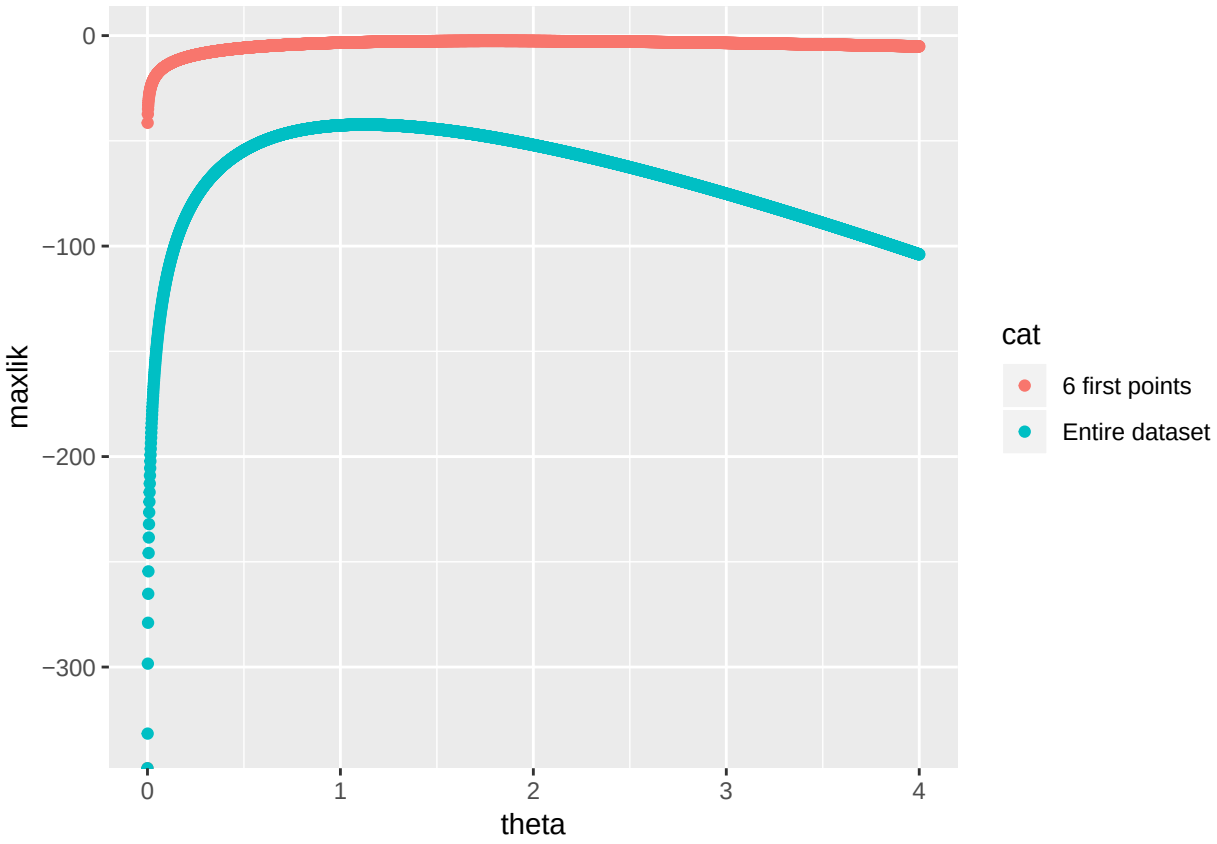
```
  }

  return(theta.return)
}
```

For the task of showing how the tha function is dependent on the theta value, we picked different values to start with, but ended up with 0<=theta<=4 which gave most interesting results. In the graph below we can see how the log likelihood varies with respect to theta. When theta is around 0, the likelyhood is below -300 and reaches it's peek around theta value of 1. For theta > 1 the likelyhood starts to decrease.



The maximum likelihood as read from the plot is between -50 and -40 for theta value around 1.

In the assignment 1.3 the steps from the assignment 1.2 are supposed to be repeated however with just 6 first observations from the dataset and to be plotted on to the graph found in 1.2. In the graph seen below we can see how the theta likelihood varies in respect to theta for just first 6 data samples. The curve for the first 6 points starts of at around the maximum likelihood from the entire data set at around -50 and -40 and after it reaches theta 1 it seem to converge torwards 0. We can read from the graph that the maximum likelihood is reached for theta around 1 just like for the entire dataset, however it does not decrease as rappidly as the previous graph.



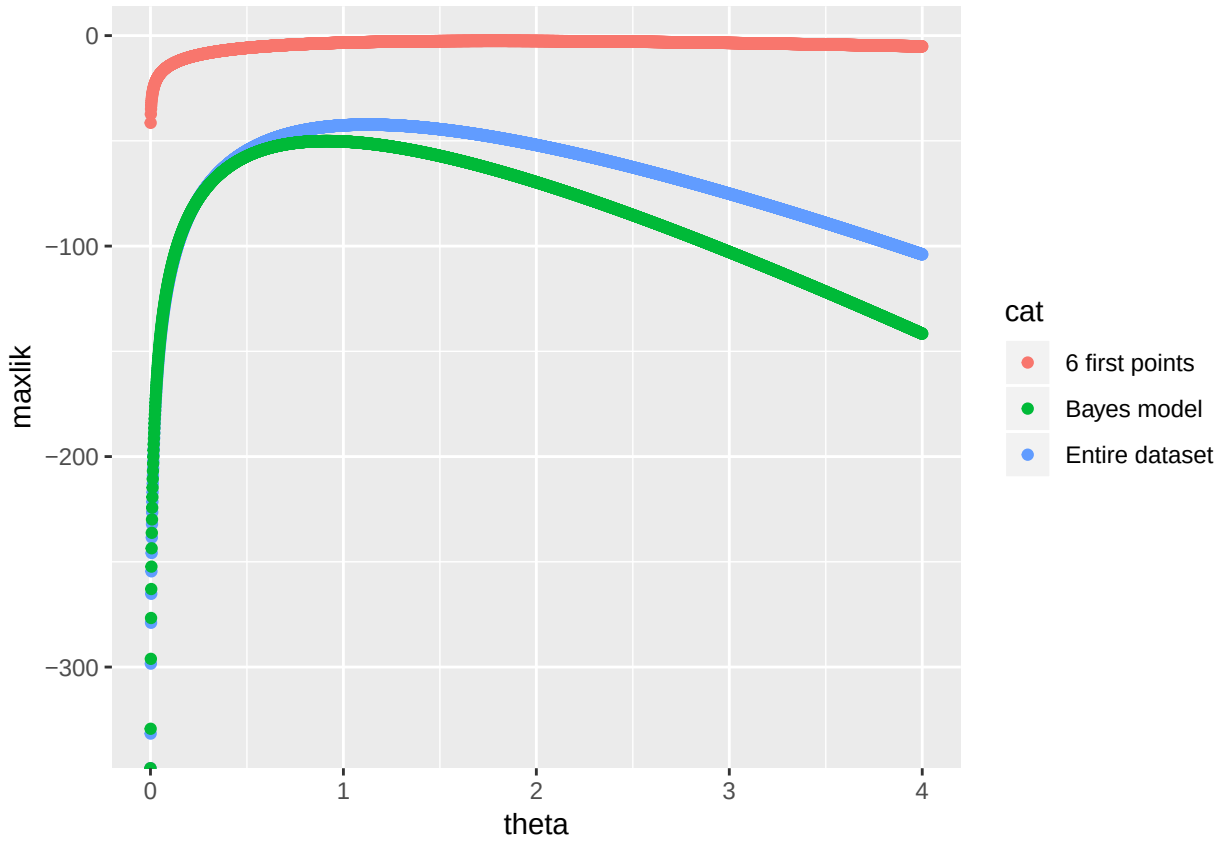
For the assignment 1.4 we have a Bayesian model with function $\mathrm{p}(x \mid \theta) = \theta e^{-\theta x}$ and a prior function $\mathrm{p}(\theta) = \lambda e^{-\theta\lambda}$ where $\lambda = 10$ and we want to show the dependence of function $l(\theta) = \log(\mathrm{p}(x \mid \theta)\mathrm{p}(\theta))$ (which will be the log of posterior) on θ and plot the function with the previous ones. In the graph below we can se how similar the bayesian model is to the model that uses the entire dataset. The baysian model decreases a bit faster than the first model as well as it has a slightly different maximum value. The maximum likelihood for the functions are following:

```
## Max likelihood for all datapoints:  -42.29453  with theta= 1.126
```

```
## Max likelihood for first 6 datapoints:  -2.521202  with theta= 1.786
```

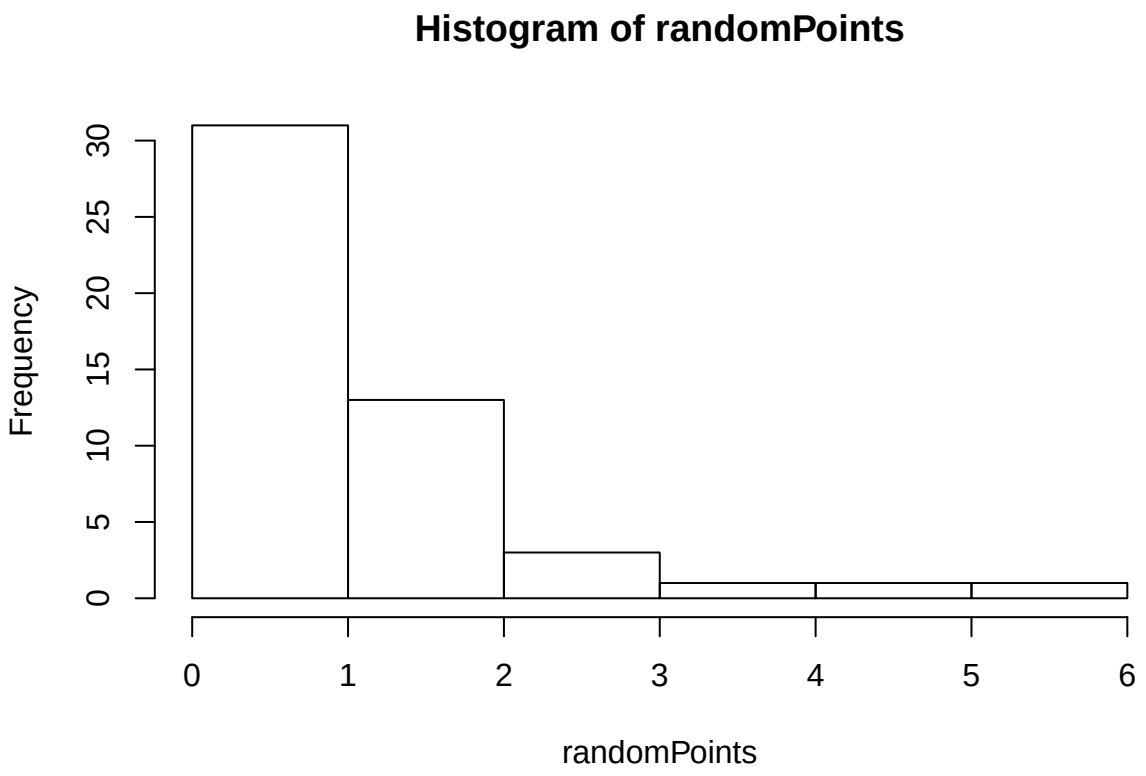
```
## Max likelihood for bayesian model:  -50.10891  with theta= 0.912
```

Summarizing, theta that gives maximum log liklihood varies between 0.912 to 1.786 for all models



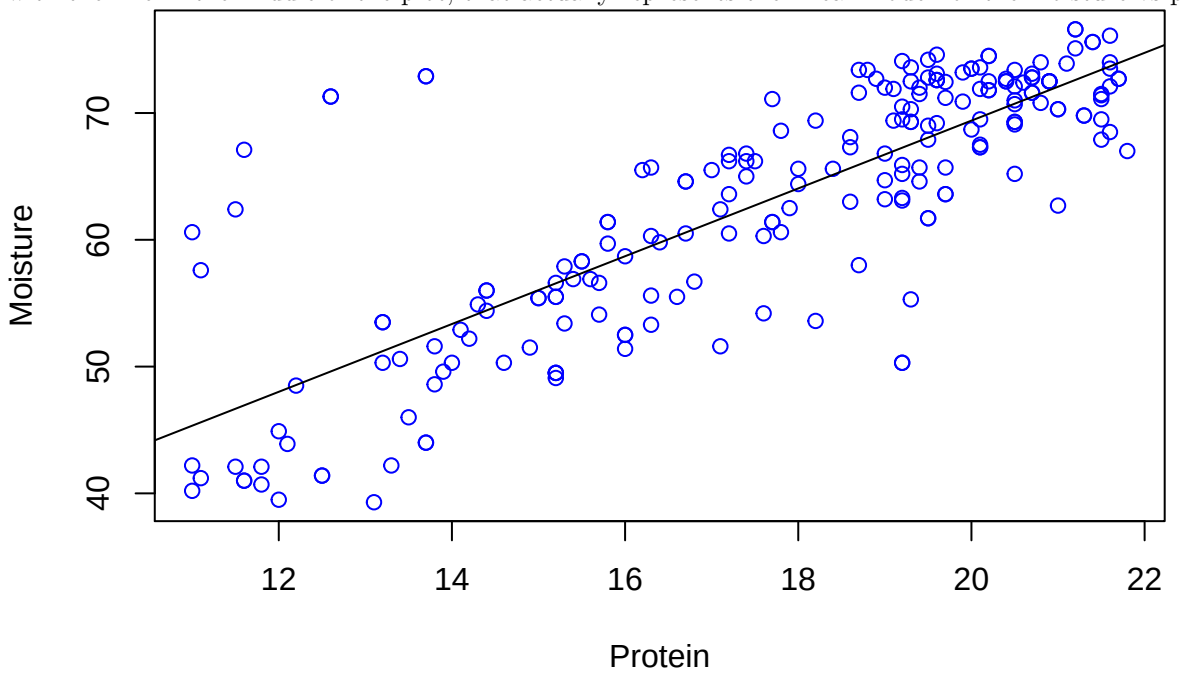
For the last subassignment 1.5 we need to create a histogram with the θ value from 1.2 using the standard numbers generator:

```
set.seed(12345)
randomPoints = rexp(50, rate=1.126)
```



Assignment 4: Linear regression and regularization

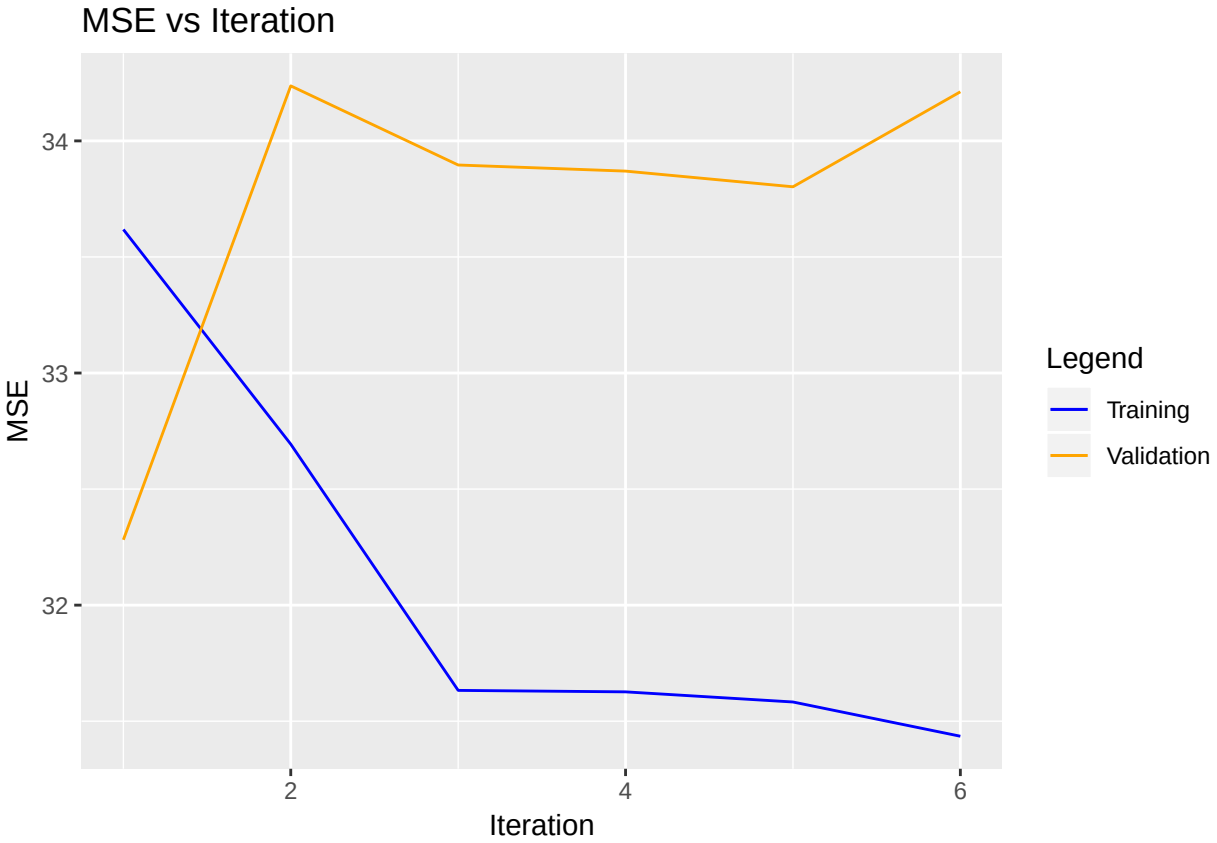
In this assignment the first task is to import the dataset "tecator.xlsx", create a plot Moisture vs Protein and answer the question if the data can be described well by a linear model. If we look at the plot below, most of the data points are linearly increasing. There are some anomalies found along the moisture axis, that diviate from the rest, but overall the data can be described well by a linear model. This can be seen with the line in the middle of the plot, that actually represents the linear model for the moisture vs protein.



If the we have a model M_i where the moisture is normaly distributed: $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ and the expected moisture is polynomial function of protein: $E = \beta_0 + x\beta_1 + x^2\beta_2 + ...x^n\beta_n + \epsilon$ then the model M_i can be described as: $M_i(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-E)^2}{2\sigma^2}}$

The probabilistic model that describes M_i is regression and it is appropriate to use Mean square error (MSE) criterion because it gives average squared difference between the estimated values and the value that is estimated (the error). Minimizing this error function gives a model that fits the data in a better manner, meaning, the line that approximates the function of the given dataset will be selected so that the MSE is as low as possible.

In the assignment 4.3 we need to train regression models M_i where $i = \{1, 2, 3, 4, 5, 6\}$ using different polynomials (i is the highest order of the polynomial). The first M_1 is a linear model, that is simply a linear regression line as seen in the first plot in the assignment 4. M_2 is a quadric model, M_3 is cubic etc. We are supposed to train these models with the training set, which is 50% of the original dataset and test the models using both the training data set as well as test data set separatelly and then plot how the MSE varies for the different models. The hypothesis for this is, that for the prediction made on the data set used for training (training dataset) will have MSE that is lowered with higher amount of the polynomial, that will eventually lead to overfitting of the model. This will cause poorer performance on the test data.



Seen in the figure above is the plot of MSE of the prediction on both training data and validation (test) data. As expected the MSE for the test on the training dataset is decreasing significantly with the higher order of the model, where for the test data the MSE decreases up to the order of 5 and then increases. This is the effect of overfitting, and as we can see in iteration 6, the MSE for the training data set is still lower than for the iteraion 5. We want the model, that minimizes the MSE but does not overfit. In this case the appropriate model will be of order $i = 5$.

In the assignment 4.4 we need to perform a variable selection where FAT form the dataset is the response and Channel1-100 are predictior. For this purpouse we use function *stepAIC* from the MASS library and print the amount of the selected variables. We will perform the step in the both directions which should yield

best results.

```
## [1] "Number of selected variables:  63"
```

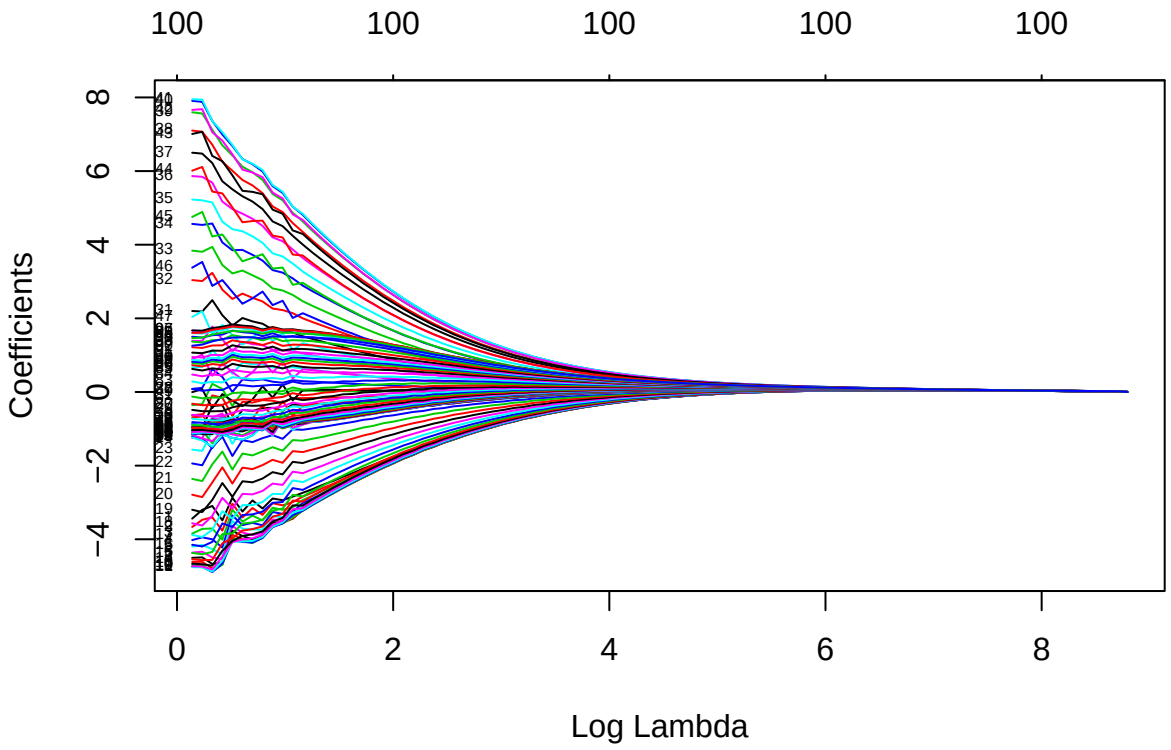
As seen above, the number of selected variables is 63.

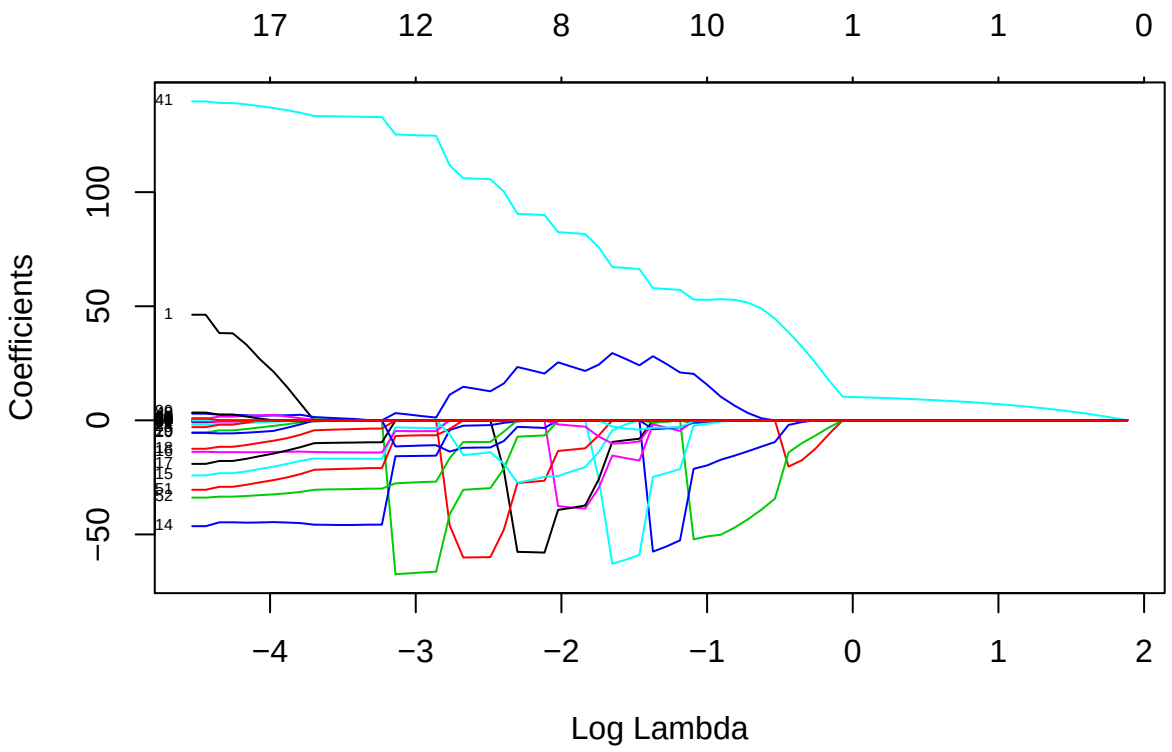
We will present the assignments 4.5 and 4.6 together because the Ridge regression and LASSO models are very similar and the difference is only the α variable, that is set to 0 in Ridge and 1 for LASSO.

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

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

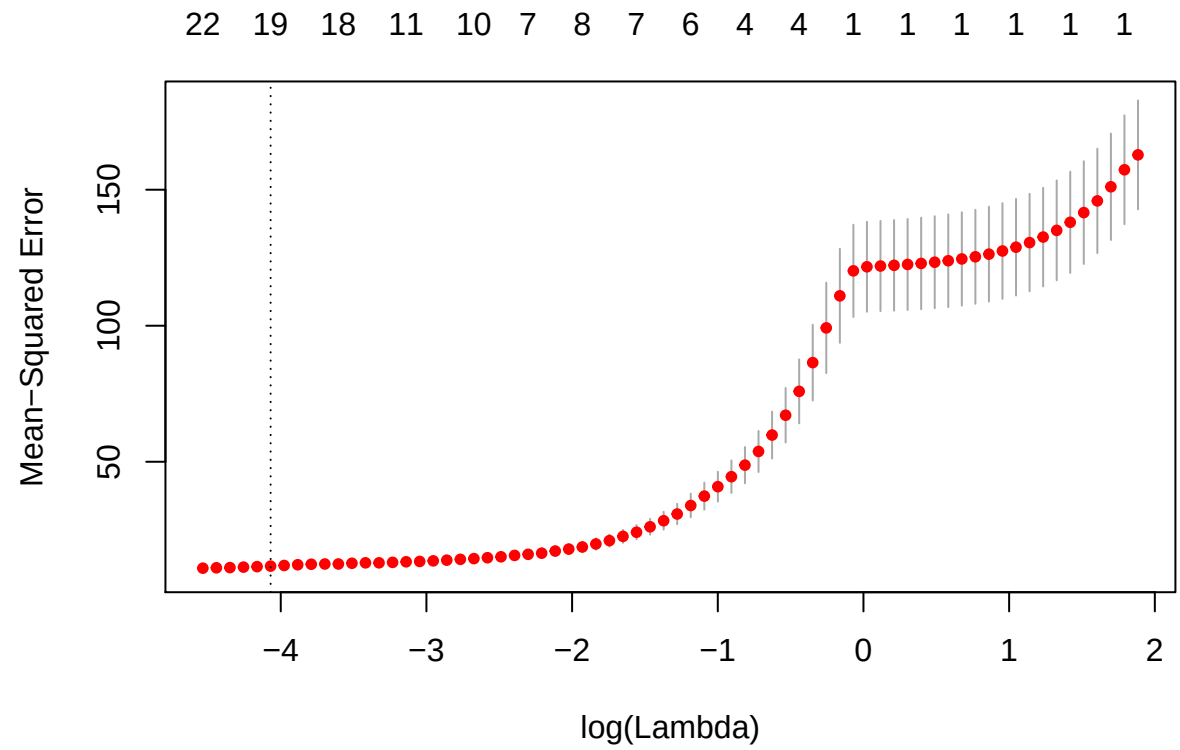
```
## Loaded glmnet 2.0-16
```





Seen are graphs for both Ridge regression and LASSO. As it can be observed all of the variables in Ridge converge torwards 0 very quickly and in very similar manner compared to LASSO, where the values converge slower torwards 0. The difference between the two is that the Ridge regression "squeezes" down the features so that they have very small impact on the model but are never 0, where the lasso actually make the unwanted features equal 0.

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



In the

assignment 4.7 we got do the cross validation to find the most optimal LASSO model from the previous assigment The result is plotted above. The printed value is the λ value that minimizes the MSE and it is equal to 0. This means that number of features chosen by the cross validation is 100 (if $\lambda = 0$ none of the features gets removed). Comparing it to the 4.4, where the number of features was 63 it is around 50% increase in feratures. This mean that all of the features might be nessesairy to minimize the MSE (at least according to the CV for LASSO).

Apendix

The code used in the assignments is provided here.

Assignment 1 code

```
library(kknn)
my_data = readxl::read_excel(path = "/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab1/Assig
n=dim(my_data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train = my_data[id,]
test = my_data[-id,]

fit = glm(Spam~.,family=binomial(link='logit'),data=train)

results.train = predict(fit,train, type = "response")
results = predict(fit,test, type = "response")
results.train05 = ifelse(results.train > 0.5,1,0)
#Assignment 1.2a
results05 = ifelse(results > 0.5,1,0) #Assignment 1.2b

#Assignment 1.2a
table(factor(train$Spam, labels=c("Actual not spam", "Actual spam")),
      factor(results.train05, labels=c("Pred not spam", "Pred spam")));

missclassification05.train = mean(results.train05 != train$Spam)
print(paste('Missclassification error train',missclassification05.train))
print(paste('Accuracy train ',1-missclassification05.train))

#Assignment 1.2b
table(factor(test$Spam, labels=c("Actual not spam", "Actual spam")),
      factor(results05, labels=c("Pred not spam", "Pred spam")));

missclassification05 = mean(results05 != test$Spam)
print(paste('Missclassification error test',missclassification05))
print(paste('Accuracy test ',1-missclassification05))

print(paste('Precission train: ',344/(142+344)))
print(paste('Precission test: ',336/(146+336)))

results.train09 = ifelse(results.train > 0.9,1,0) #Assignment 1.2a
results09 = ifelse(results > 0.9,1,0) #Assignment 1.2b

#Assignment 1.3a
```

```
table(factor(train$Spam, labels=c("Actual not spam", "Actual spam")),
      factor(results.train09, labels=c("Pred not spam", "Pred spam")));
missclassification09.train = mean(results.train09 != train$Spam)
print(paste('Missclassification error train',missclassification09.train))
print(paste('Accuracy train ',1-missclassification09.train))

#Assignment 1.3b
table(factor(test$Spam, labels=c("Actual not spam", "Actual spam")),
      factor(results09, labels=c("Pred not spam", "Pred spam")));
missclassification09 = mean(results09 != test$Spam)
print(paste('Missclassification error test',missclassification09))
print(paste('Accuracy test ',1-missclassification09))

print(paste('Recall train: ',6/(6+419)))
print(paste('Recall test: ',6/(6+427)))

modelknn.train30 <- kknn(Spam~., train, train, k=30)
modelknn.test30 <- kknn(Spam~., train, test, k=30)
result.train30 <- floor(fitted(modelknn.train30) + 0.5)
result.test30 <- floor(fitted(modelknn.test30) + 0.5)
misClasificError.kknn30.train = mean(result.train30 != train$Spam)
misClasificError.kknn30.test = mean(result.test30 != test$Spam)


print(paste("Misclassification K = 30, train, train", misClasificError.kknn30.train))
print(paste("Misclassification K = 30, train, test", misClasificError.kknn30.test))


modelknn.train1 <- kknn(Spam~., train, train, k=1)
modelknn.test1 <- kknn(Spam~., train, test, k=1)
result.train1 <- floor(fitted(modelknn.train1) + 0.5)
result.test1 <- floor(fitted(modelknn.test1) + 0.5)
misClasificError.kknn1.train = mean(result.train1 != train$Spam)
misClasificError.kknn1.test = mean(result.test1 != test$Spam)

print(paste("Misclassification K = 1, train, train",misClasificError.kknn1.train))
print(paste("Misclassification K = 1, train, train",misClasificError.kknn1.test))
```

Assignment 2 code

```
my_data = readxl::read_excel(path = "/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab1/Assig
hist(my_data$Length)

logLikelihood = function(x, theta){

  theta.return = 1:length(theta)

  for(a in 1 : length(theta)){
    probVector = theta[a]*exp(-theta[a]*x)
    theta.return[a] = log(prod(probVector))
  }
```

```
  return(theta.return)
}

library(ggplot2)
theta = seq(0, 4, by=0.001)
result1.vector = logLikelihood(my_data$Length, theta)
result1 = data.frame(theta = theta, maxlik= result1.vector)
result1$cat = "Entire dataset"
ggplot(result1, aes(theta, maxlik, colour = cat))+
  geom_point()
#print(max(result1))

result2.vector = logLikelihood(head(my_data,6), theta)

result2 = data.frame(theta= theta, maxlik= result2.vector)

result2$cat = "6 first points"

df = rbind(result2, result1 )

ggplot(df, aes(theta, maxlik, colour = cat))+
  geom_point()

  p.prior= function(lambda, theta){
  returnProb = lambda*exp(-theta*lambda)
  return(returnProb)
}

logLikelihoodlabda = function(theta,x, lambda){

  theta.return = 1:length(theta)

  for(a in 1 : length(theta)){
    probVector = theta[a]*exp(-theta[a]*x)
    theta.return[a] = log(prod(probVector)*p.prior(lambda, theta[a]))
  }

  return(theta.return)
}
result.bayes.vector = logLikelihoodlabda(theta,my_data$Length, 10)
result.bayes = data.frame(theta = theta, maxlik= result.bayes.vector)
result.bayes$cat = "Bayes model"

df = rbind(result2, result1, result.bayes )

cat("Max likelihood for all datapoints: ", as.numeric(max(result1.vector)), " with theta=",
    theta[as.numeric(which.max(result1.vector))])
cat("Max likelihood for first 6 datapoints: ", as.numeric(max(result2.vector)), " with theta=",
    theta[as.numeric(which.max(result2.vector))])
cat("Max likelihood for bayesian model: ", as.numeric(max(result.bayes.vector)), " with theta=",
    theta[as.numeric(which.max(result.bayes.vector))])
#print(paste("First 6 datapoints: "max(as.vector(result2)))
#print(paste("Bayesian model: "max(as.vector(result.bayes)))
```

```r
ggplot(df, aes(theta, maxlik, colour = cat))+
  geom_point()

set.seed(12345)
randomPoints = rexp(50, rate=1.126)

hist(randomPoints)
#x <- seq(0, 3, length.out=1000)
#dat <- data.frame(x=x, px=dexp(x, rate=1.126))
#lines(x,dat$px , col="blue", lwd=2)
```

Assignment 4 code

```r
library(ggplot2)
my_data4 = readxl::read_excel(path = "/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab1/teca
plot(my_data4$Protein, my_data4$Moisture, col = "blue")

linProtMois <- lm(my_data4$Moisture ~ my_data4$Protein)
abline(linProtMois)

#assignment 4.3

my_data4 = readxl::read_excel(path = "/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab1/teca

set.seed(12345)

dimension = dim(my_data4)[1]
set.seed(12345)
id = sample(1:dimension, floor(dimension*0.5))
train = my_data4[id,]
test = my_data4[-id,]

#models:
M1 = lm(Moisture ~ poly(Protein,1), data = train)
M2 = lm(Moisture ~ poly(Protein,2), data = train)
M3 = lm(Moisture ~ poly(Protein,3), data = train)
M4 = lm(Moisture ~ poly(Protein,4), data = train)
M5 = lm(Moisture ~ poly(Protein,5), data = train)
M6 = lm(Moisture ~ poly(Protein,6), data = train)

new <- data.frame(Protein = train$Protein)
M1.predicted.train = predict(M1, newdata = new)
M2.predicted.train = predict(M2, newdata = new) #interval = "confidence" ?
M3.predicted.train = predict(M3, newdata = new)
M4.predicted.train = predict(M4, newdata = new)
M5.predicted.train = predict(M5, newdata = new)
M6.predicted.train = predict(M6, newdata = new)

new.test <- data.frame(Protein = test$Protein)
M1.predicted.test = predict(M1, newdata = new.test)
M2.predicted.test = predict(M2, newdata = new.test) #interval = "confidence" ?
```

```
M3.predicted.test = predict(M3, newdata = new.test)
M4.predicted.test = predict(M4, newdata = new.test)
M5.predicted.test = predict(M5, newdata = new.test)
M6.predicted.test = predict(M6, newdata = new.test)

mse <- function(M, data){
  return(mean((data - M)^2))
}

print(length(test$Moisture))
print(paste("MSE 1 train",mse(M1.predicted.train, train$Moisture)))
print(paste("MSE 2 train",mse(M2.predicted.train, train$Moisture)))
print(paste("MSE 3 train",mse(M3.predicted.train, train$Moisture)))
print(paste("MSE 4 train",mse(M4.predicted.train, train$Moisture)))
print(paste("MSE 5 train",mse(M5.predicted.train, train$Moisture)))
print(paste("MSE 6 train",mse(M6.predicted.train, train$Moisture)))
print(paste("MSE 1 test",mse(M1.predicted.test, test$Moisture)))
print(paste("MSE 2 test",mse(M2.predicted.test, test$Moisture)))
print(paste("MSE 3 test",mse(M3.predicted.test, test$Moisture)))
print(paste("MSE 4 test",mse(M4.predicted.test, test$Moisture)))
print(paste("MSE 5 test",mse(M5.predicted.test, test$Moisture)))
print(paste("MSE 6 test",mse(M6.predicted.test, test$Moisture)))

MSE.1 = mse(M1.predicted.train, train$Moisture)
MSE.2 = mse(M2.predicted.train, train$Moisture)
MSE.3 = mse(M3.predicted.train, train$Moisture)
MSE.4 = mse(M4.predicted.train, train$Moisture)
MSE.5 = mse(M5.predicted.train, train$Moisture)
MSE.6 = mse(M6.predicted.train, train$Moisture)
MSE.1.v = mse(M1.predicted.test, test$Moisture)
MSE.2.v = mse(M2.predicted.test, test$Moisture)
MSE.3.v = mse(M3.predicted.test, test$Moisture)
MSE.4.v = mse(M4.predicted.test, test$Moisture)
MSE.5.v = mse(M5.predicted.test, test$Moisture)
MSE.6.v = mse(M6.predicted.test, test$Moisture)


iterations <- c(1,2,3,4,5,6)
results.training <- c(MSE.1,MSE.2,MSE.3,MSE.4,MSE.5,MSE.6)
results.validation <- c(MSE.1.v,MSE.2.v,MSE.3.v,MSE.4.v,MSE.5.v,MSE.6.v)

results.training.dataframe = data.frame(Iteration= iterations, MSE1 = results.training)
results.validation.dataframe = data.frame(Iteration= iterations, MSE2 = results.validation)

results.binded = cbind(results.training.dataframe, results.validation.dataframe$MSE2 )
colnames(results.binded) = c("Iteration", "MSE1", "MSE2")

ggplot(results.binded) +
  geom_line(aes(x = Iteration, y = MSE1, colour = "Training")) +
  geom_line(aes(x = Iteration, y = MSE2, colour = "Validation")) +
  labs(title="MSE vs Iteration", y="MSE", x="Iteration", color = "Legend") +
  scale_color_manual(values = c("blue", "orange"))
```

```r
#Assignment 4.4

my_data4 = readxl::read_excel(path = "/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab1/teca

library(MASS)

fat.vector = my_data4[,102]
sliced.data = my_data4[,2:101]

model.sliced = lm(fat.vector$Fat~. , data = sliced.data)

reduced = stepAIC(model.sliced, direction = c("both"), trace = FALSE)

print(paste("reduced: ", length(reduced$coefficients-1))) #-1 for the coefficient B_0


#Assignment 4.5,4.6
library(glmnet)

my_data4 = readxl::read_excel(path = "/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab1/teca

set.seed(12345)
sliced.data = my_data4[,2:101]

ridge =glmnet(as.matrix(sliced.data),my_data4$Fat, alpha=0, family = "gaussian")
lasso=glmnet(as.matrix(sliced.data),my_data4$Fat, alpha=1, family = "gaussian")

plot(ridge, xvar="lambda", label=T)
plot(lasso, xvar="lambda", label=T)

#Assignment 4.7

library(glmnet)

lasso=glmnet(as.matrix(sliced.data),my_data4$Fat, alpha=1, family = "gaussian")
my_data4 = readxl::read_excel(path = "/Users/karolwojtulewicz/Google\ Drive/skola/TDDE01/Labs/Lab1/teca
sliced.data = my_data4[,2:101]

model=cv.glmnet(as.matrix(sliced.data),my_data4$Fat, alpha=1,
family="gaussian", lambda = c(lasso$lambda,0))
print(model$lambda.min) #lambda that minimizes the error
plot(model)
#coef(model, s="lambda.min")
```