Lab 1 Group B 12

Axel Holmberg (axeho681), Jonathan Reimertz (jonre639) and Wilhelm Hansson (wilha431)

Assignment 1

1.

Import the data into R and divide it into training and test sets (50%/50%)

```
data <- read.csv2("spambase.csv")

#Split data into training and test set.

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

2.

Use logistic regression (functions glm(), predict()) to classify the training and test data by the classification principle $\hat{Y} = 1$ *if* $(Y = 1|X) > 0.5$*, otherwise* $\hat{Y} = 0$ *and report the confusion matrices (use table()) and the misclassification rates for training and test data. Analyse the obtained results.*

```
#GLM model for data with family
#binomial --> only 0s and 1s
model <- glm(Spam ~ ., family=binomial, data=train)

predictModel= predict(model, newdata=test, type="response")

#Split up the model into spam and
#not spam
probability <- ifelse(predictModel > 0.5, "1", "0")

confMatrix <- table(probability, test[,"Spam"]) #Confusionmatrix from the model
#Diagonal of the misclassfication rate by dividing the
#diagonal from the confusionmatricx with the whole confusionmatrix
modelDiag <- diag(confMatrix)
missClMa1 = 1-(sum(modelDiag)/sum(confMatrix))

#Prints results
print("Confusion matrix 2:")
```

```
## [1] "Confusion matrix 2:"
```

```
print(confMatrix)
```

```
##
## probability   0   1
##           0 808  92
##           1 143 327
```

```
print("missclassification 2:")
```

```
## [1] "missclassification 2:"
```

```
print(missClMa1)
```

```
## [1] 0.1715328
```

Analyse the obtained results.

The confusion matrix gives us the misclassification rate. I would say that the misclassification rate is okay for its' aplication. It is all very dependent on the use case though.

3.

Use logistic regression to classify the test data by the classification principle $\hat{Y} = 1$ *if* $p(Y = 1|X) > 0.8$, *otherwise* $\hat{Y} = 0$ *and report the confusion matrices (use table()) and the misclassification rates for training and test data. Compare the results. What effect did the new rule have?*

```
#Split up the model into spam and not spam
probability2 <- ifelse(predictModel > 0.8, "1", "0")

confMatrix2 <- table(probability2, test[,"Spam"])

modelDiag2 <- diag(confMatrix2)

missClMa2 = 1-(sum(modelDiag2)/sum(confMatrix2))


print("Confusion matrix 3:")
```

```
## [1] "Confusion matrix 3:"
```

```
print(confMatrix2)
```

```
##
## probability2   0   1
##            0 931 314
##            1  20 105
```

```
print("missclassification 3:")
```

```
## [1] "missclassification 3:"
```

```
print(missClMa2)
```

```
## [1] 0.2437956
```

What effects did the new rule have?

The new rule made it so that the rate is a bit worse, so it should stay at the previous value.

4.

Use standard classifier kknn() with K=30 from package kknn, report the the misclassification rates for the training and test data and compare the results with step 2.

```r
#KKNN with K=30

kknn_K30 = kknn(Spam ~ ., train=train, test=test, k=30)
kknn_K30_pred = predict(kknn_K30)

#Split up the model into spam and not spam
kknn_K30_pred <- ifelse(kknn_K30_pred > 0.5, 1, 0)


confMa_K30 = table(kknn_K30_pred, test[,"Spam"])
misCl_K30 = 1-sum(diag(confMa_K30)/sum(confMa_K30))

print("Misclassification 4:")
```

```
## [1] "Misclassification 4:"
```

```r
print(misCl_K30)
```

```
## [1] 0.3131387
```

Compare the results with step 2.

The misclassification rate is even worse than in step 2. Probably because of the realtively small dataset, which fits a parametric method better. If one would have more data then the K-nearest neighbour could be better for this application, as that works better for non-parametric methods.

5.

Repeat step 4 for $K = 1$ and compare the results with step 4. What effect does the decrease of K lead to and why?

```r
#KKNN with K=1
kknn_K1 = kknn(Spam ~ ., train=train, test=test, k=1)
kknn_K1_pred = predict(kknn_K1)

#Split up the model into spam and not spam
kknn_K1_pred <- ifelse(kknn_K1_pred > 0.5, 1, 0)

confMa_K1 = table(kknn_K1_pred, test[,"Spam"])
misCl_K1 = 1-sum(diag(confMa_K1)/sum(confMa_K1))

print("missclassification 5:")
```

```
## [1] "missclassification 5:"
```

```r
print(misCl_K1)
```

```
## [1] 0.3591241
```

What effect does the decrease of K lead to and why?

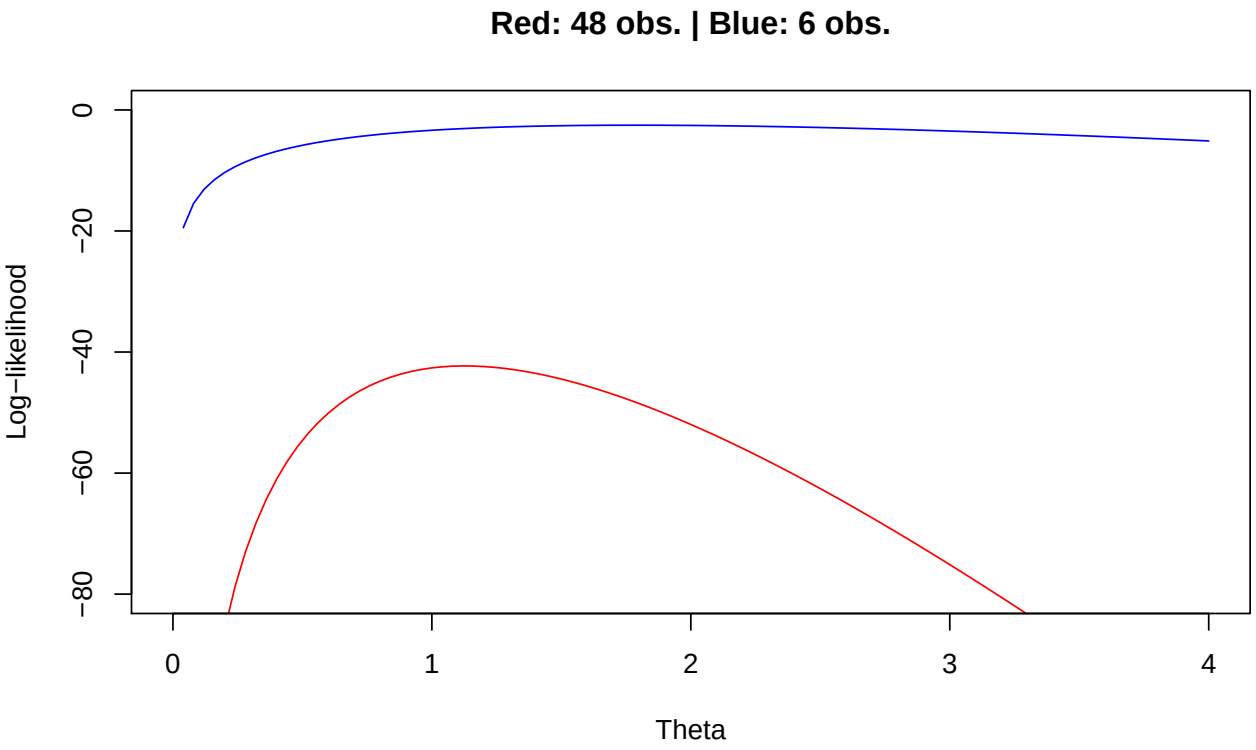
The decrease in K leads to a noisier prediction and is thereby worse. It can also lead to overfitting.

Assignment 2

This assignment included creating and evaluating probability models by using the log-likelihood function as well as a bayesian model to learn the expected lifetime of a certain machine. With the only input variable length.

1-3. Log-likelihood function

By using the log-likelihood function to find the maximum likelihood value of θ, two models were created; one by using all the given data (48 observations) and one by only using 6 observations.

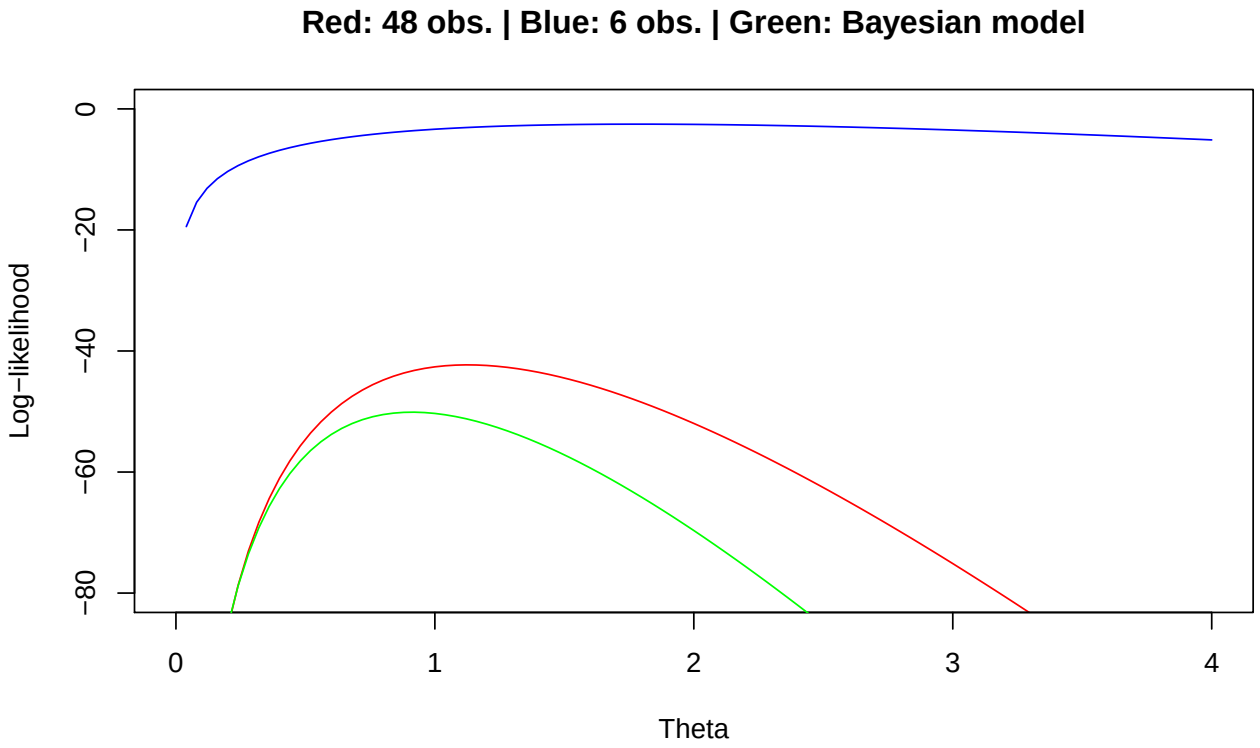


From the first model we can see that the maximum likelihood value is 1.13 compared to 1.79 in the second model. By compairing the plots of the two models we can also see that the first model is more steep and not so stable when changing θ and the second model is more stable so the likelihood-value does not change too much depending on θ.

4. Bayesian model

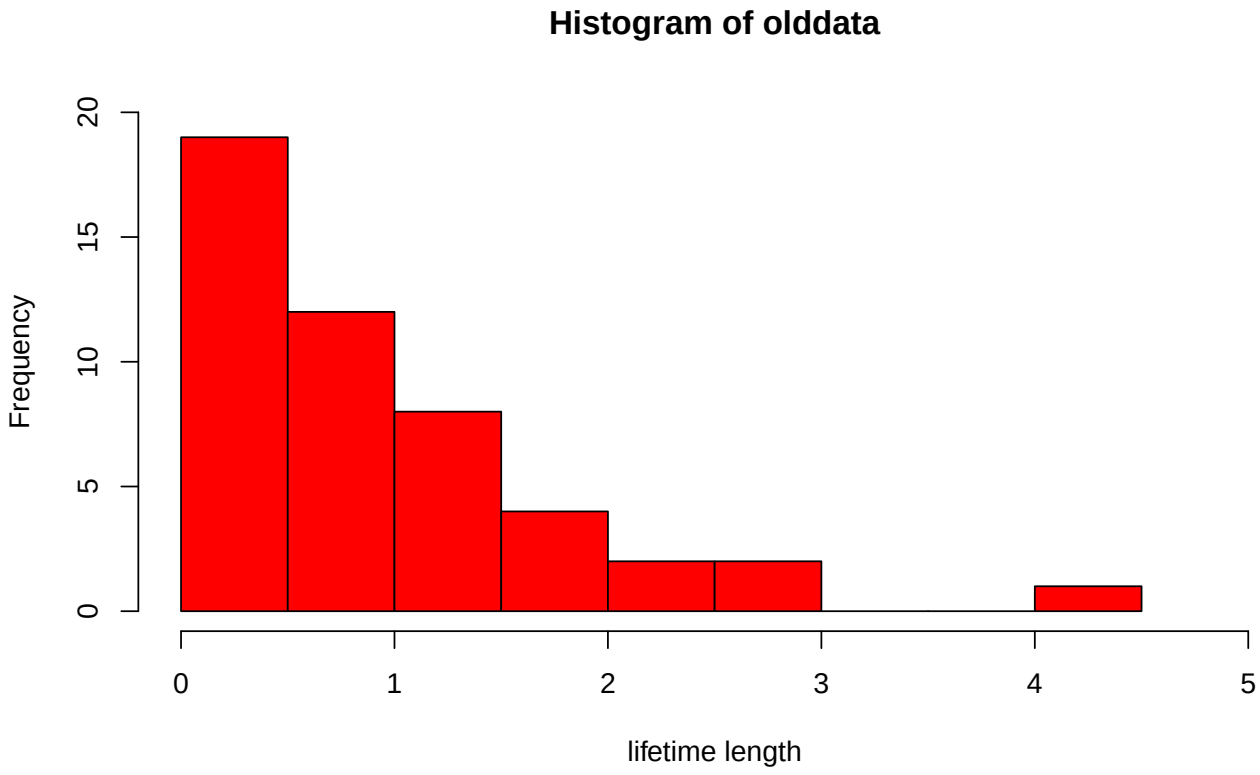
The used function, $l(\theta) = log(p(x|\theta) \cdot p(\theta))$, computes what could be interpreted as the log-likelihood of $p(x \cap \theta)$, as bayes theorem gives $p(x|\theta) = \frac{p(x \cap \theta)}{p(\theta)}$.

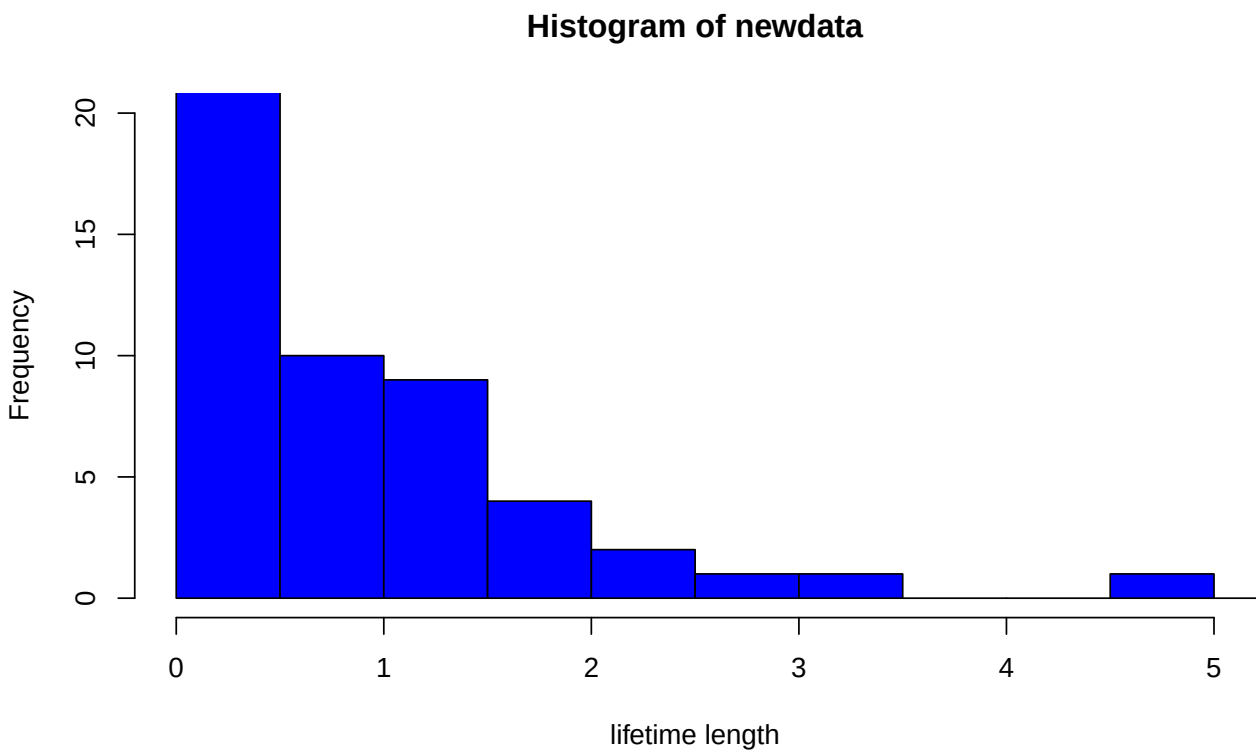
By comparing this to the result of step 2 (and step 3), it shows that the resulting plot is quite similar to the plot of step 2 with the difference of giving lower values and being more steep. The maximum likelihood value of θ is somewhat lower (0.91) using the bayesian model compared to the model of step 2 (1.13).



5. Assessment of θ-value from step 2

By generating 50 new observations by using the θ-value obtained from step 2 two histograms were made; one out of the old data and one of the new/generated data - as seen below.



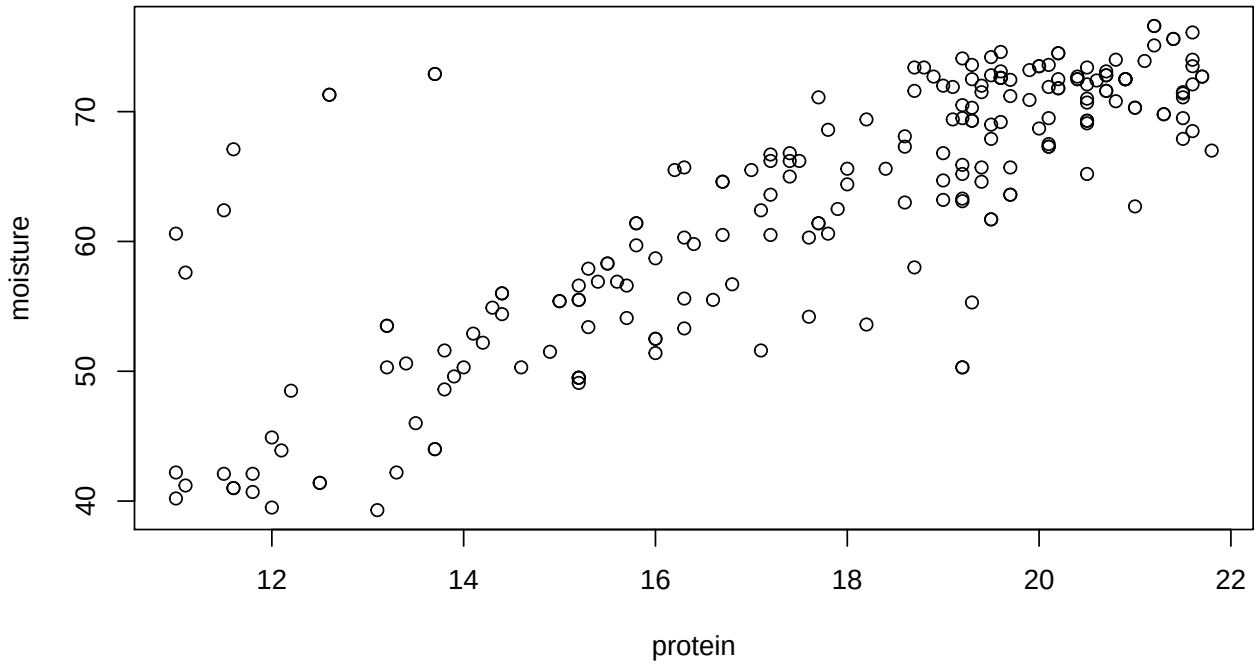


When comparing these we can see that the obtained θ-value of step 2 is a good estimation of the machines' expected lifetime.

Assignment 4. Linear regression and regularization

1. Import data to R and create a plot of Moisture versus Protein. Do you think that these data are described well by a linear model?

```
# Import of file
tecator <- read_excel("tecator.xlsx")

moisture <- tecator$Moisture
protein <- tecator$Protein

plot(protein, moisture)
```



Observation: The data seems to be described well by a linear model.

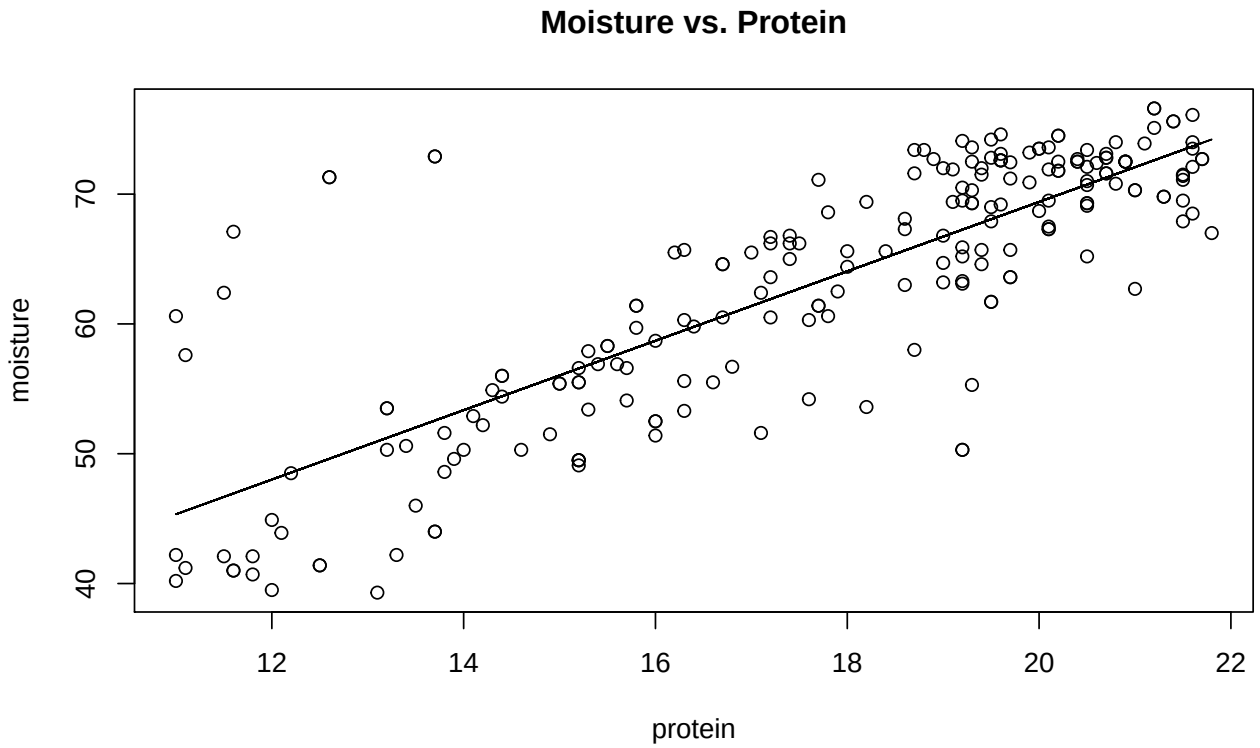
2. Consider model M_i in which Moisture is normally distributed, and the expected Moisture is a polynomial function of Protein including the polynomial terms up to power i (i.e M_1 is a linear model, M_2 is a quadratic model and so on). Report a probabilistic model that describes M_i. Why is it appropriate to use MSE criterion when fitting this model to a training data? *Answer:* The form of the models are as follows: $M_1 = \beta_0 + \beta_1 x + \epsilon$ $M_2 = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$ $[\ldots 6]$ where $\epsilon \sim N(0, \sigma^2)$ The MSE criterion minimizes the error in the prediction, and is therefore an appropriate model to use.

```
protein_and_moisture <- tecator[,103:104]

#creating model based on data
model1 <- lm(formula = moisture ~ protein, data=protein_and_moisture)
summary(model1)
```

```
##
## Call:
## lm(formula = moisture ~ protein, data = protein_and_moisture)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.9611  -3.1660   0.0255   2.5836  21.6858
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  15.9246     2.3405   6.804 1.01e-10 ***
## protein       2.6738     0.1305  20.491  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.758 on 213 degrees of freedom
## Multiple R-squared:  0.6634, Adjusted R-squared:  0.6619
## F-statistic: 419.9 on 1 and 213 DF,  p-value: < 2.2e-16
```

```
#making prediction based on model
predicted_values <- predict(model1, interval = "confidence")
```



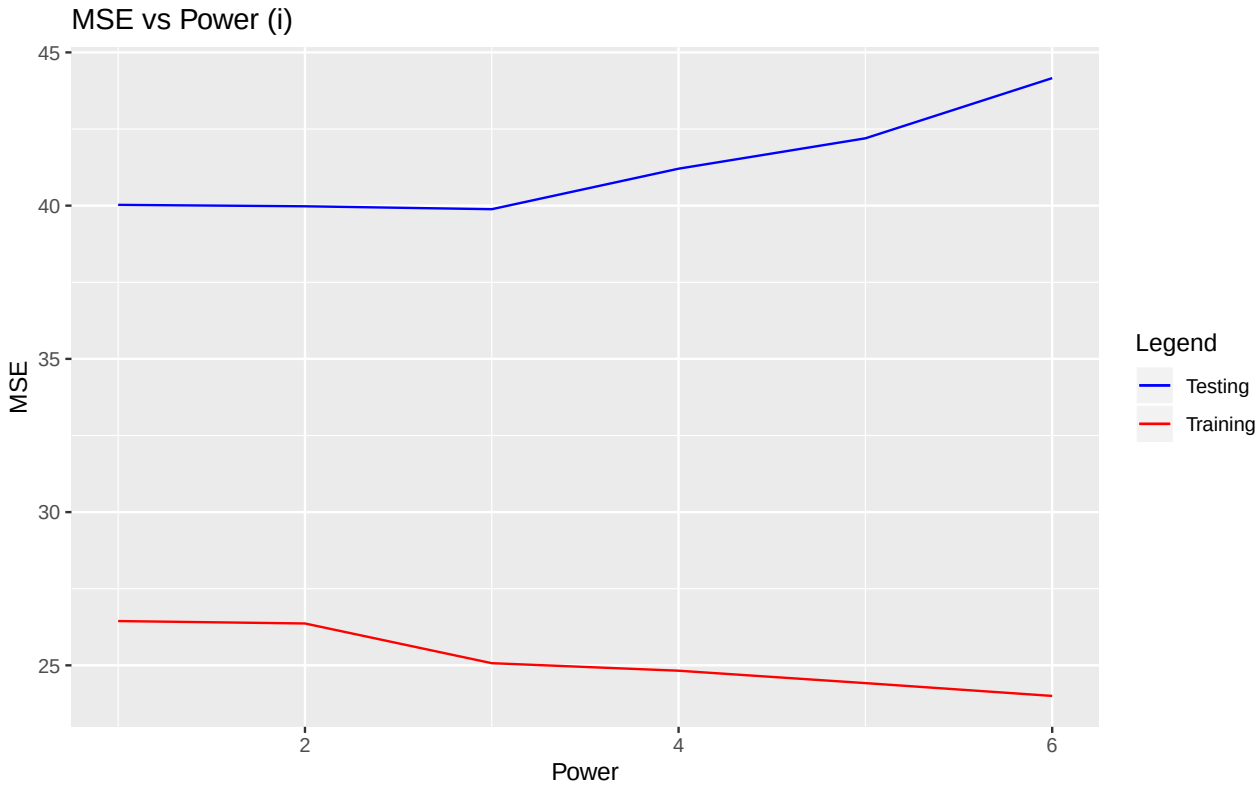
Graph: Points correspond to measured values and the line is the model prediction.

3. Divide the data into training and validation sets 50%/50% and 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. Which model is best according to the plot? How do the MSE values change and why? Interpret this picture in terms of bias-variance tradeoff.

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

# create models
M1 <- lm(formula = Moisture ~ poly(Protein,1), data=train)
M2 <- lm(formula = Moisture ~ poly(Protein,2), data=train)
M3 <- lm(formula = Moisture ~ poly(Protein,3), data=train)
```

```r
M4 <- lm(formula = Moisture ~ poly(Protein,4), data=train)
M5 <- lm(formula = Moisture ~ poly(Protein,5), data=train)
M6 <- lm(formula = Moisture ~ poly(Protein,6), data=train)

#predict based on test using model
M1.pred.test <- predict(M1,newdata=test)
M2.pred.test <- predict(M2,newdata=test)
M3.pred.test <- predict(M3,newdata=test)
M4.pred.test <- predict(M4,newdata=test)
M5.pred.test <- predict(M5,newdata=test)
M6.pred.test <- predict(M6,newdata=test)

#predict based on train using model
M1.pred.train <- predict(M1,newdata=train)
M2.pred.train <- predict(M2,newdata=train)
M3.pred.train <- predict(M3,newdata=train)
M4.pred.train <- predict(M4,newdata=train)
M5.pred.train <- predict(M5,newdata=train)
M6.pred.train <- predict(M6,newdata=train)

# calculate MSE
M1.pred.test.mse = mse(test$Moisture,M1.pred.test)
M2.pred.test.mse = mse(test$Moisture,M2.pred.test)
M3.pred.test.mse = mse(test$Moisture,M3.pred.test)
M4.pred.test.mse = mse(test$Moisture,M4.pred.test)
M5.pred.test.mse = mse(test$Moisture,M5.pred.test)
M6.pred.test.mse = mse(test$Moisture,M6.pred.test)
mse_test <- c(M1.pred.test.mse,
              M2.pred.test.mse,
              M3.pred.test.mse,
              M4.pred.test.mse,
              M5.pred.test.mse,
              M6.pred.test.mse)

M1.pred.train.mse = mse(train$Moisture,M1.pred.train)
M2.pred.train.mse = mse(train$Moisture,M2.pred.train)
M3.pred.train.mse = mse(train$Moisture,M3.pred.train)
M4.pred.train.mse = mse(train$Moisture,M4.pred.train)
M5.pred.train.mse = mse(train$Moisture,M5.pred.train)
M6.pred.train.mse = mse(train$Moisture,M6.pred.train)
mse_train <- c(M1.pred.train.mse,
               M2.pred.train.mse,
               M3.pred.train.mse,
               M4.pred.train.mse,
               M5.pred.train.mse,
               M6.pred.train.mse)

power_i <- c(1:6)
results.train = data.frame(Power_i= power_i, MSE1 = mse_train)
results.test = data.frame(Power_i= power_i, MSE2 = mse_test)

results = cbind(results.train, results.test$MSE2 )
colnames(results) = c("Power_i", "MSE1", "MSE2")
```

```
ggplot(results) +
  geom_line(aes(x = Power_i, y = MSE1, colour = "Training")) +
  geom_line(aes(x = Power_i, y = MSE2, colour = "Testing")) +
  labs(title="MSE vs Power (i)", y="MSE", x="Power", color = "Legend") +
  scale_color_manual(values = c("blue", "red"))
```



Answer: As can be observed in the graph, after $i = 3$ the increase in variance decreases the MSE of the training set. However, a large increase in MSE in the testing set is observed as a result of overfitting the model to the training set. Therefore the model $M_3 = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \epsilon$ is chosen.

The entire data set is used in the following computations.

4. Perform variable selection of a linear model in which Fat is response and *Channel1-Channel100* are predictors by using stepAIC. Comment on how many variables were selected.

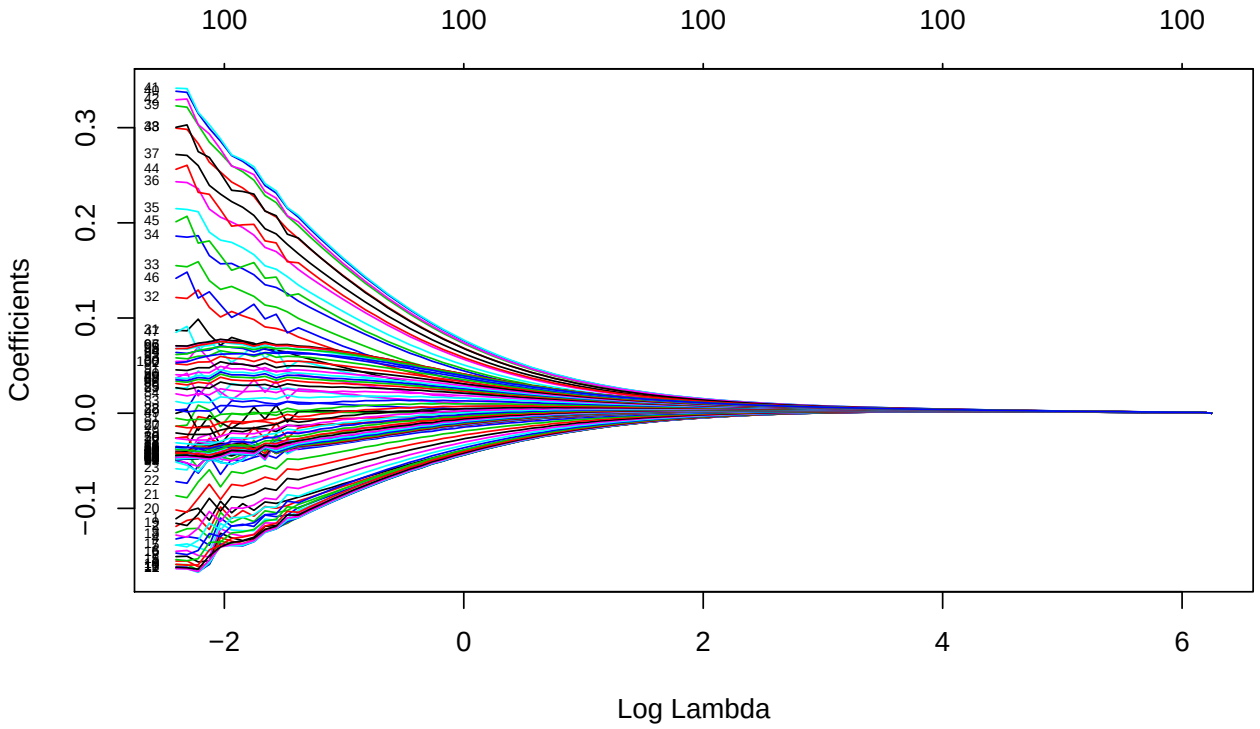
```
fat_model <- lm(tecator$Fat ~ ., data = tecator[,2:101])
```

```
fat_model_reduced <- stepAIC(fat_model, trace = FALSE, direction="both")
```

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

5. Fit a Ridge regression model with the same predictor and response variables. Present a plot showing how model coefficients depend on the log of the penalty factor λ and report how the coefficients change with λ.

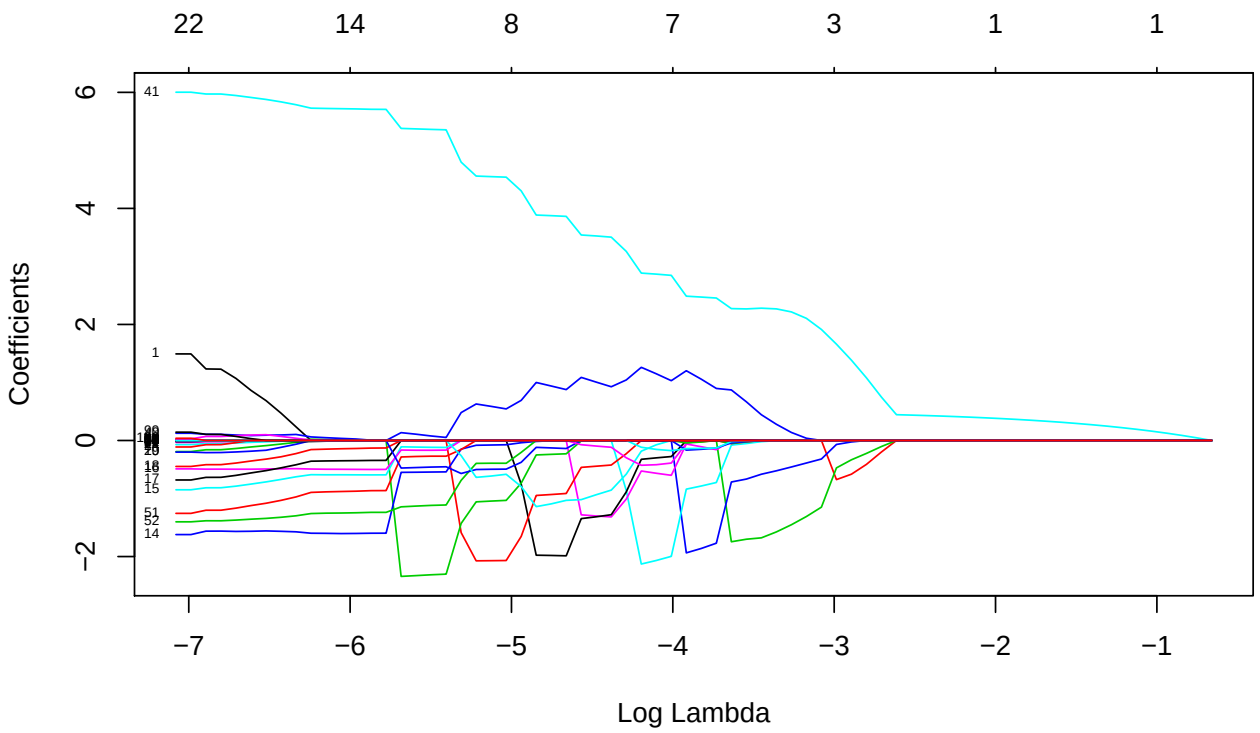
```
covariates=scale(tecator[,2:101])
response=scale(tecator[,102])
ridge_model=glmnet(as.matrix(covariates), response, alpha=0, family="gaussian")
plot(ridge_model, xvar="lambda", label=TRUE)
```



Observation: In the graph one can observe that the values of the coefficients decrease when the value of λ increases.

6. Repeat step 5 but fit LASSO instead of the Ridge regression and compare the plots from steps 5 and 6. Conclusions?

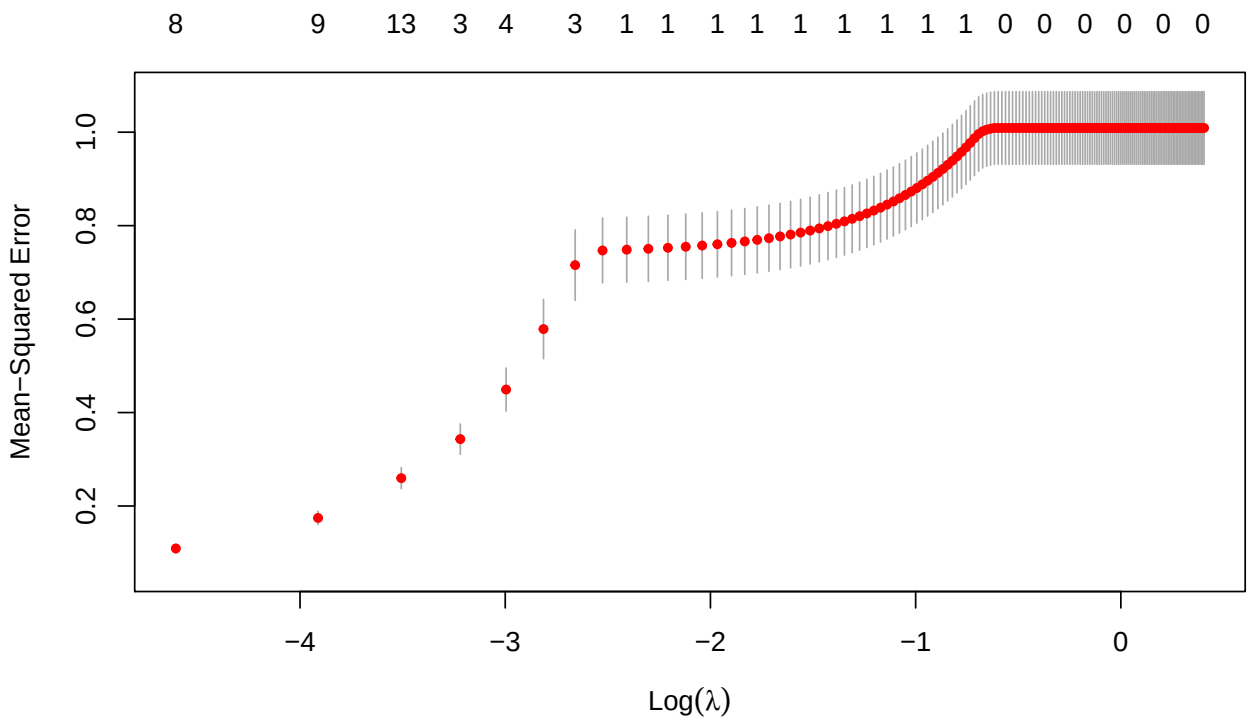
```
covariates=scale(tecator[,2:101])
response=scale(tecator[,102])
lasso_model=glmnet(as.matrix(covariates), response, alpha=1,family="gaussian")
plot(lasso_model, xvar="lambda", label=TRUE)
```



Conclusion: The coefficients decreases as λ increases which is also the case in step 5 . The major different observed is how this decreas happens. The difference is due to the difference in the penelty term where ridge has $\lambda * \Sigma x^2$ and LASSO $\lambda * \Sigma |x|$. TThis results in Ridge making the feutures really realy close to zero but never zero where instead LASSO actually sets unwanted feutures to actual zero.

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. Present also a plot showing the dependence of the CV score and comment how the CV score changes with λ.

```
lasso_model_cv = cv.glmnet(as.matrix(covariates),
                           response,
                           alpha=1,
                           family="gaussian",
                           lambda=seq(0,1.5,0.01))
plot(lasso_model_cv)
```




```
#coef(lasso_model_cv, s="lambda.min")
print(paste("Optimal lambda: ",lasso_model_cv$lambda.min))
```

```
## [1] "Optimal lambda:  0"
```

Comment: A $\lambda^* = 0$ means that all 100 feutures where selected byt the cross validation.

8. Compare the results from steps 4 and 7.

Answer: Selected features in step 4 was 63 and in step 7 it was 100. The conclusion of this is that according to cross validation with LASSO, all features are needed to minimize the MSE.