TDDE01 - Computer lab 1 block 1

Viktor Blidh, Jonatan Baggman, Christian Rasmussen

November 2019

1 Introduction

This is a report for the first lab in TDDE01 - Machine learning. The report was written by Viktor Blidh (first two assignments) and Jonatan Baggman (assignment 4).

2 Assignments

The lab consisted of three different assignments which will be presented in the following sections.

2.1 Assignment 1

The purpose of this assignment was to classify emails either as spam or not spam using different regression models. A set of 2740 emails was given, half of which was used as training data while the other half was used as test data. Firstly logisitic regression was used on the training data, with the following classification principle:

$$\hat{Y} = 1 \text{ if } p(Y = 1 \,|X) > 0.5, \text{ otherwise } \hat{Y} = 0 \tag{1}$$

This resulted in a confusion matrix, which can be seen in table 1. This matrix shows the amount of *True Negatives*(894), *False Positives*(307), *False Negatives*(37) and *True Positives*(132) that the classification principle yielded. The accuracy of the prediction can be calculated by calculating the sum of *True Positives* and *True Negatives* and dividing it by all predictions. The

pred	0	1
0	894	307
1	37	132

Table 1: Confusion matrix for training data with classification 1

misclassification rate of the prediction is given by calculating $1 -$ accuracy. For the training data the misclassification rate was 25%.
Next, the logistic regression model was used on the testing data. The confusion matrix can be seen in 2. The misclassification rate for the testing data was 35%.

pred	0	1
0	836	365
1	115	54

Table 2: Confusion matrix for testing data with classification 1

These results shows that the model and classification principle is quite bad, since the misclassification rate was very high, even for the data it trained on.

The next sub assignment was to use the same model with another classification principle. The principle was:

$$\hat{Y} = 1 \text{ if } p(Y = 1 \,|X) > 0.8, \text{ otherwise } \hat{Y} = 0 \tag{2}$$

The confusion matrices for training and test data can be seen in table 3 and 4, respectively.

pred	0	1
0	925	467
1	6	2

Table 3: Confusion matrix for training data with classification 2

The result of using this classification principle is that almost all mails will be classified as not spam. This can be seen in the confusion matrices since there is a high amount of *True Negatives*, but also a high amount of *False Negatives*. Therefore, using this principle does not yield a desirable result.

pred	0	1
0	946	416
1	5	3

Table 4: Confusion matrix for testing data with classification 2

In the last part of the assignment the *K-Nearest-Neighbours* (knn) algorithm was used to classify the emails. Firstly the algorithm was used with the K-value 30. The confusion matrices for the training and testing data with this algorithm can be seen in tables 5 and 6 respectively. The misclassification rate for the training data was 17%, and for the training data it was 31%. This is slightly better than the previous classifications, but still not very good.

pred	0	1
0	779	152
1	77	362

Table 5: Confusion matrix for training data with knn with K=30

pred	0	1
0	702	249
1	180	239

Table 6: Confusion matrix for testing data with knn with K=30

Finally knn was used with a K-value of 1. This means that the output always will be assigned the class of its single closest neighbour, namely itself. The confusion matrices for testing and training data with K=1 can be seen in tables 7 and 8. The missclassification rate is 0% for training data, and 36% for test data.

pred	0	1
0	931	0
1	0	439

Table 7: Confusion matrix for training data with knn with K=1

pred	0	1
0	644	307
1	185	234

Table 8: Confusion matrix for testing data with knn with K=1

Using a K value of 1 leads to overfitting, since the model tries to fit into the training data completely. So while it gives a good prediction in training, it doesn't yield good results when applied on other data.

2.2 Assignment 2

In the second assignment different models was used to predict the lifetime of machines. Firstly the probability model of $\theta e^{-\theta x}$ was assumed. The distribution of x seems to be following a normal distribution. Figure 1 shows the dependence of log-likelyhood on the value of θ when the entire data was used for fitting. The maximum likelyhood was received with a θ-value of 1.13.

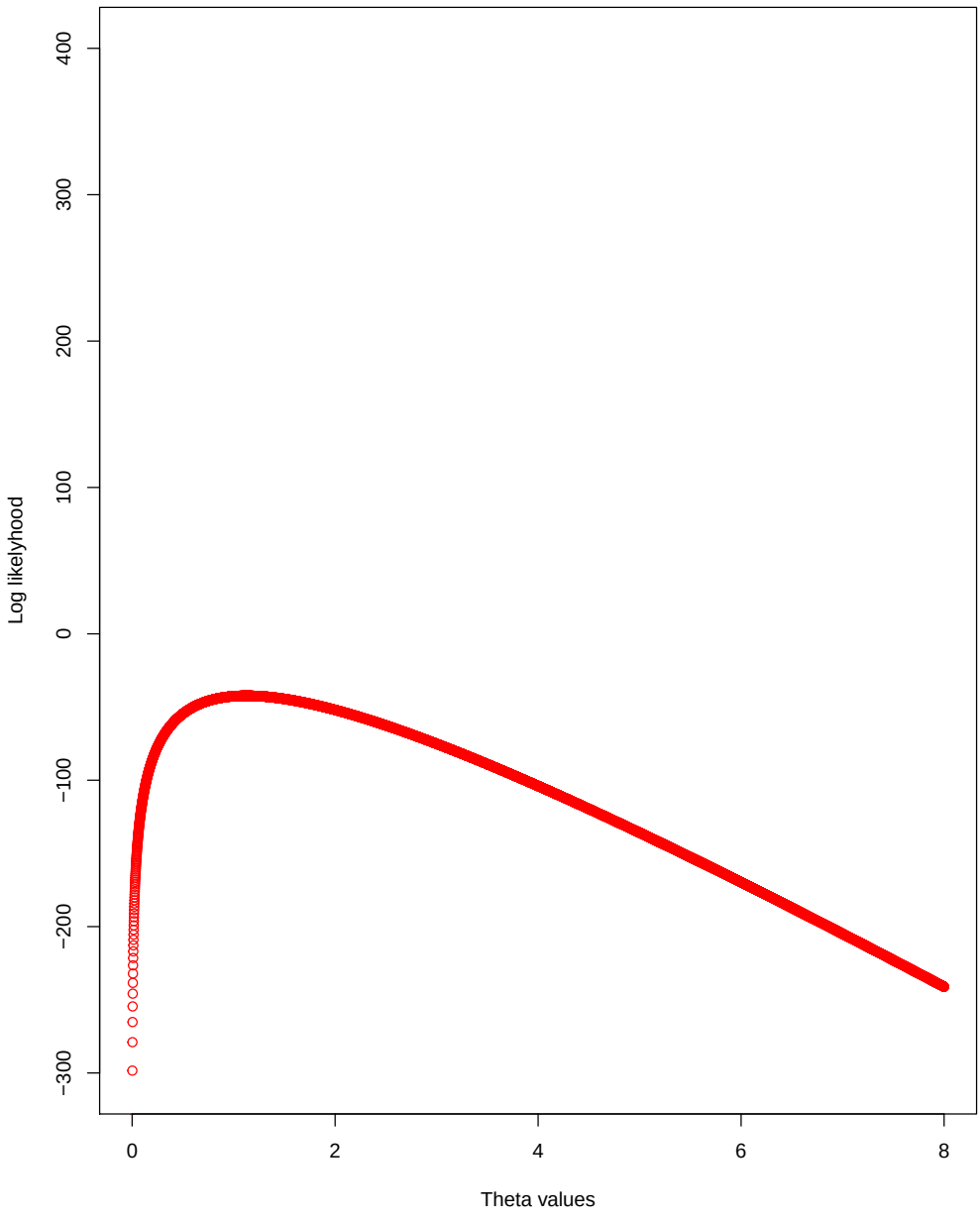

Figure 1: Dependence of log likelyhood on *theta* values

If only the six first entries of the machine data was used the model becomes less reliable, since less data was used to fit the model. Figure 2 also shows the log-likelyhood when six entries was used (blue line). As can be seen in the figure no maximum θ can be found, the graph keeps growing.

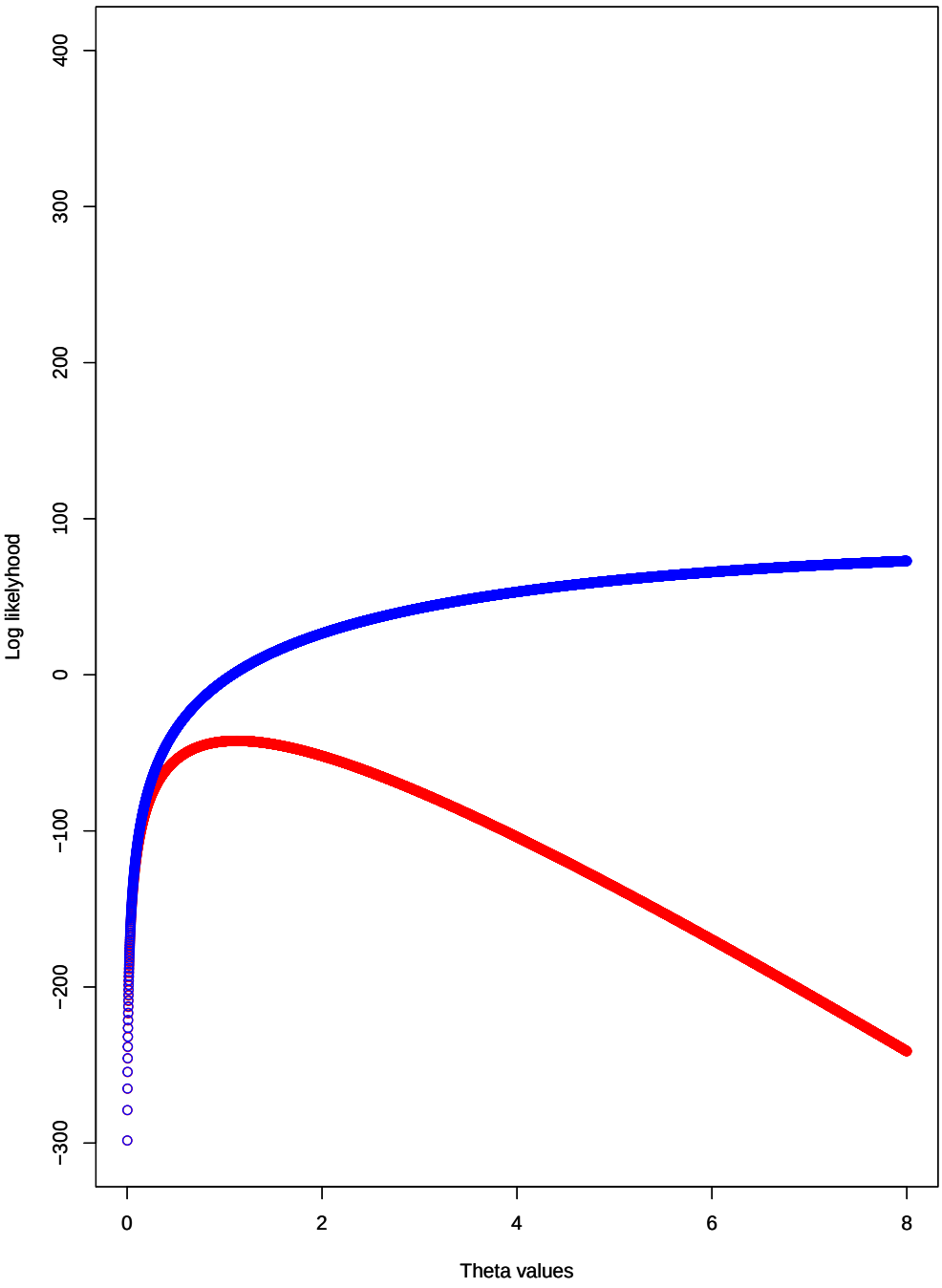

Figure 2: Dependence with full data-set as well as with 6 entries

In the next sub-assignment a Bayesian model was used instead. The Bayesian model also made use of a prior $p(\theta) = \lambda e^{-\lambda\theta}$ with a fixed λ of 10, The log-likelyhood is then given by $l(\theta) = log(p(x|\theta)p(\theta))$. This gives a posterior probability of the lifetime of a machine, since it also uses a prior to weight the probability. Figure 3 shows the log-likelyhood of the Bayesian model (yellow line) and the standard model (red line). In these plots the full data-set was used.

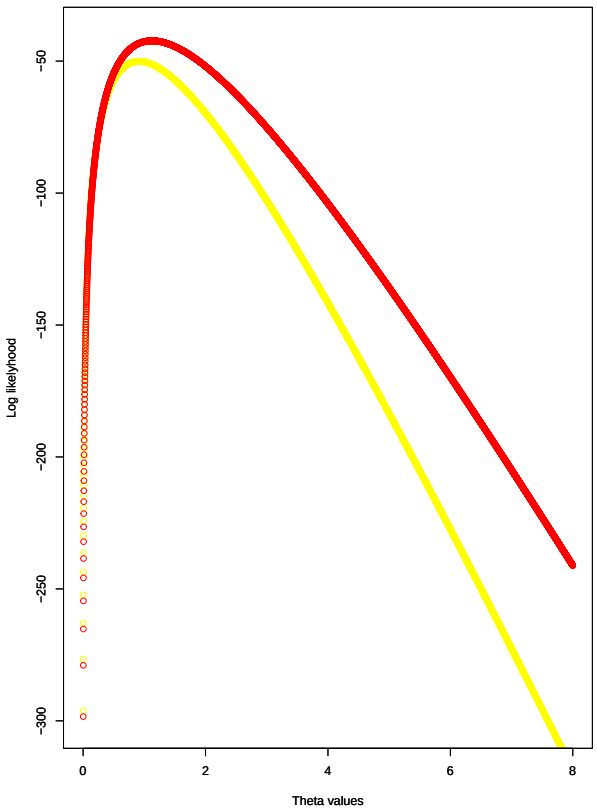

Figure 3: Log-likelyhood of Bayesian model and standard model

As can be seen in figure 3 the Bayesian model does not provide a better θ than the standard model. The optimal θ for the Bayesian model is approximately 0.91, but its value is less than that of the standard model.

Lastly the most optimal θ found ($\theta = 1.13$), was used to generate 50 new observations from $p(x|\theta) = \theta e^{-\theta x}$. Figures 4 and 5 shows histograms displaying the distribution of the original data and the generated data, respectively. The distribution of the generated data is quite similar to the original data, which implies that the optimal θ found is close to the actual optimum.

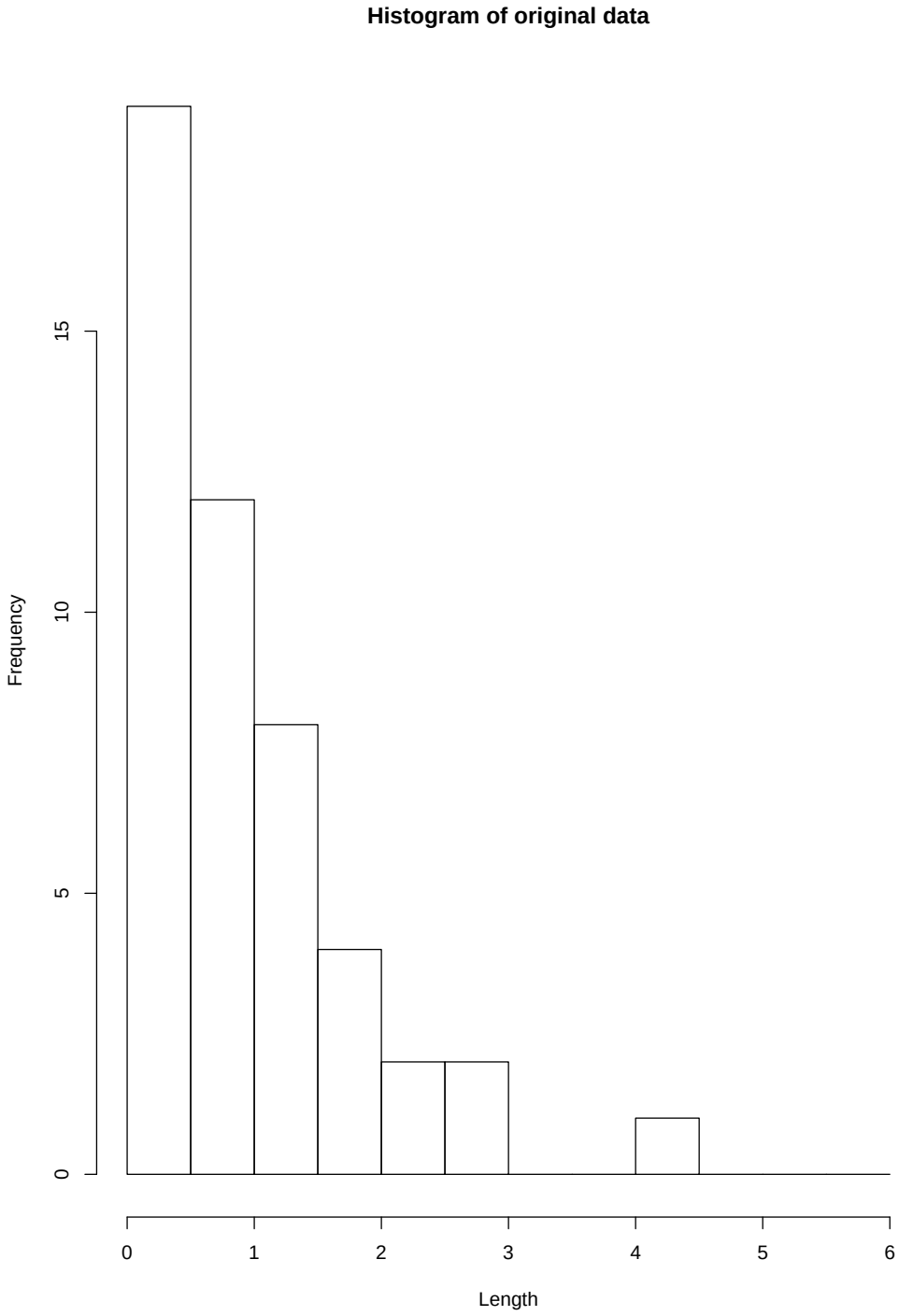

Figure 4: Histogram of distribution of original data

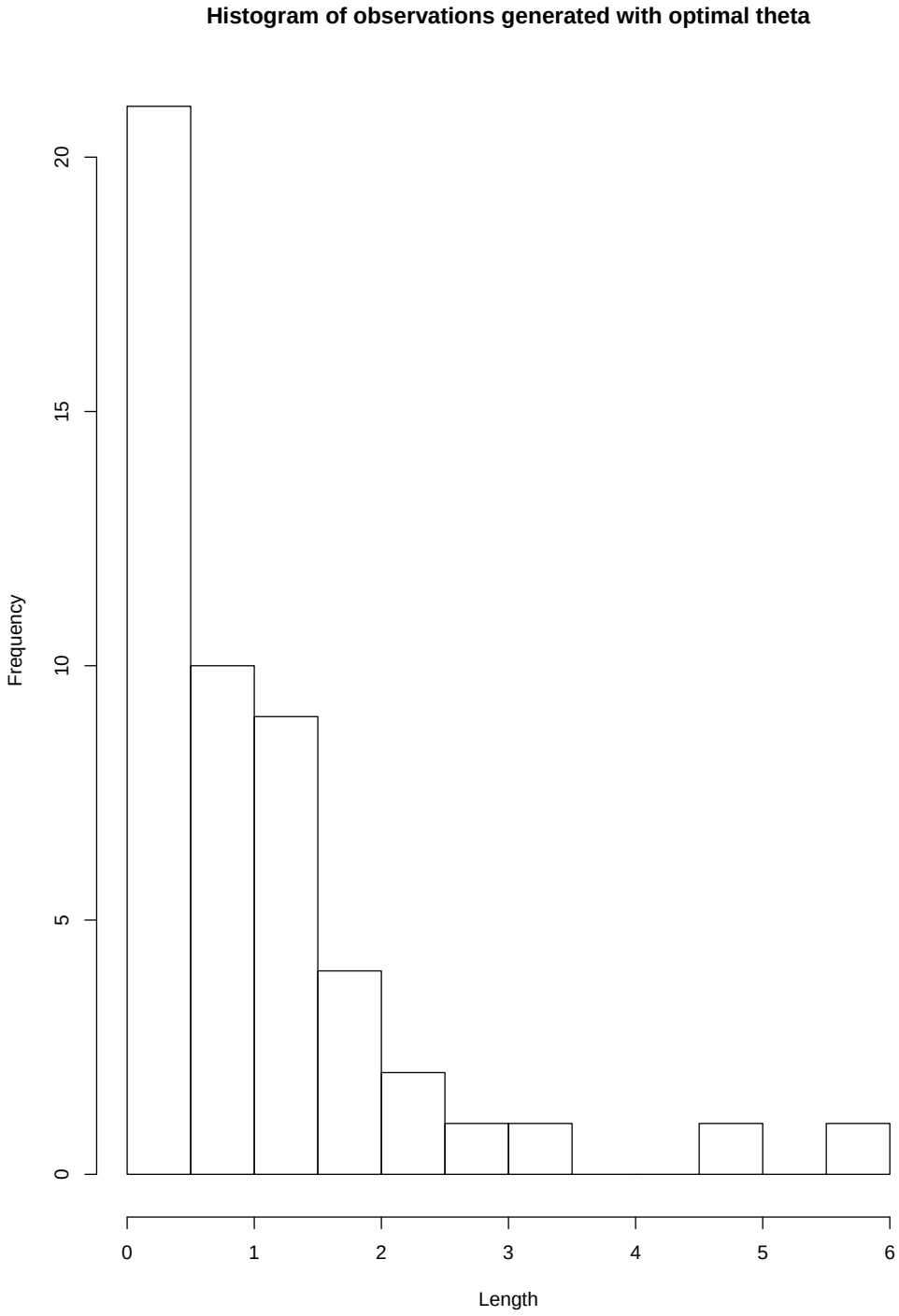

Figure 5: Histogram of distribution of generated data

2.3 Assignment 4

A data-set containing data of meat samples was analyzed. In figure 6 the moisture level is plotted against protein content.

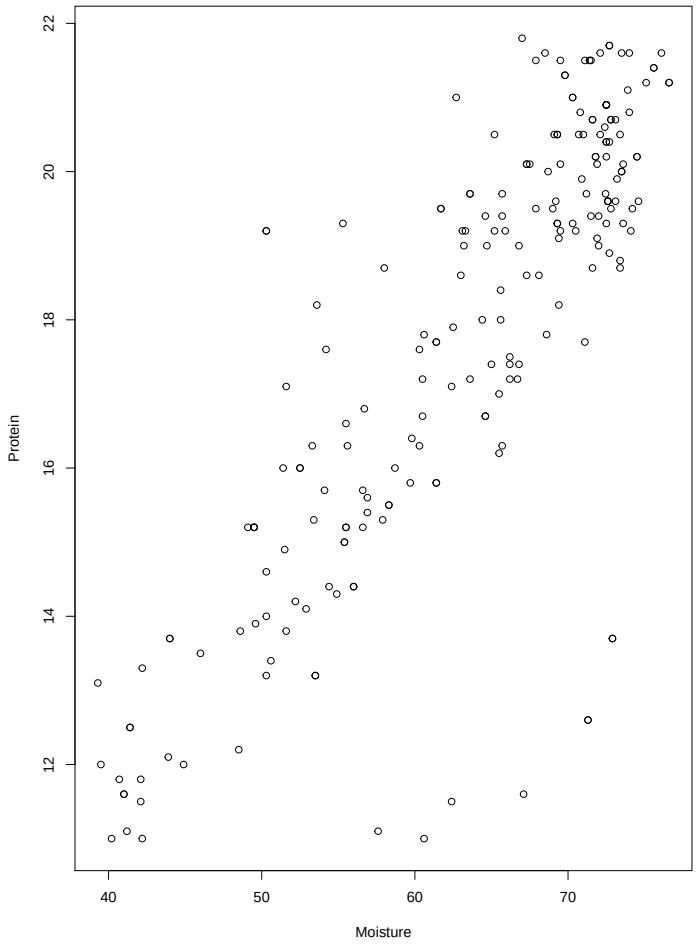

Figure 6: Moisture vs Protein

The graph shows that a linear model would explain the data reasonably well. M_i for a certain i is a model which predicts the moisture level based on

protein content. M_i is distributed according to equation (3).

$$M_i \sim N(\sum_{n=0}^{i} w^i protein^i, \sigma^2) \tag{3}$$

Mean square error (MSE) can be used as a cost function which is to minimize when training models like Mi. With this measurement, the errors of the predictions are squared and summed and then the sum is divided by the degrees of freedom of the model. with MSE, different models with different degrees of freedom can be compared to find an optimal one.

6 models were created using Mi with i values from 1 to 6. In figure 7, both training (red) and testing (blue) MSE can be seen.

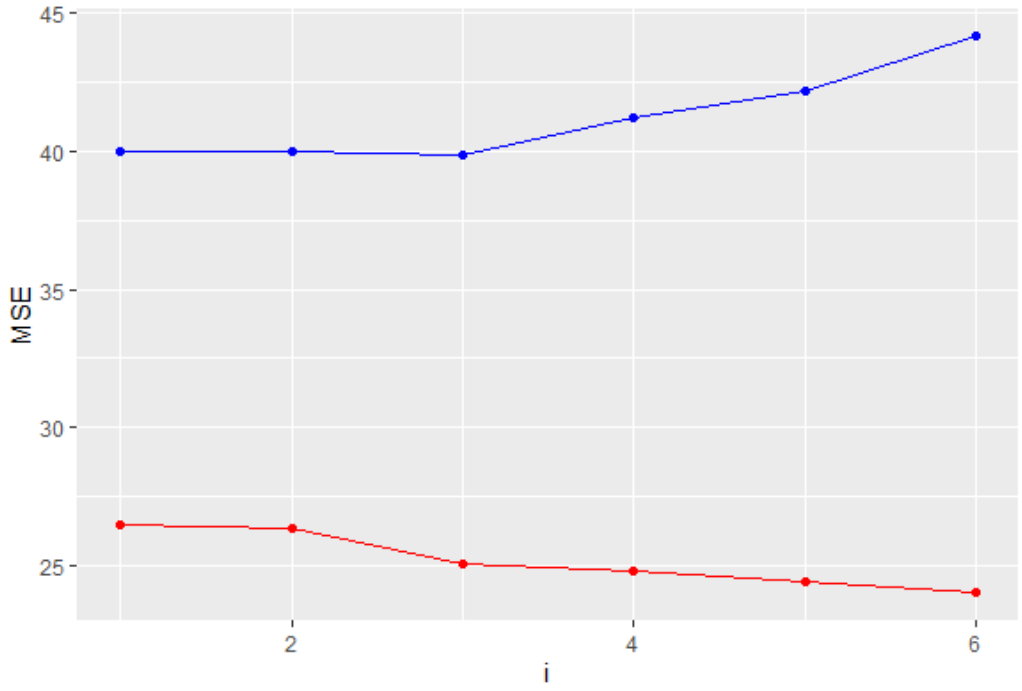

Figure 7: MSE for training and test data

According to the graph, M3 is the best model as it has the lowest MSE score for the testing data-set (blue). For training data, the MSE is lowered for increasing values of i while for testing data, it first decreases but then starts to increase. At first, for low i, the model has a bias and that is causing the

MSE. For high i instead, there is no bias resulting in the MSE score, instead, the variance is high as the complex model is overfitted to the training data. This is causing the increase in MSE when applying the models to testing data.

Then the fat content of the meat-data was analyzed. 100 channels of ir-data were used as potential variables for a model. Using stepAIC for variable selection 63 of the potential channels were selected as variables.

A ridge regression model was then used to fit the channels to the fat content. Figure 8 shows how the coefficients change with an increasing lambda. The graph shows how the coefficients are decreasing with an increasing lambda. Note that all coefficients decrease together towards zero.

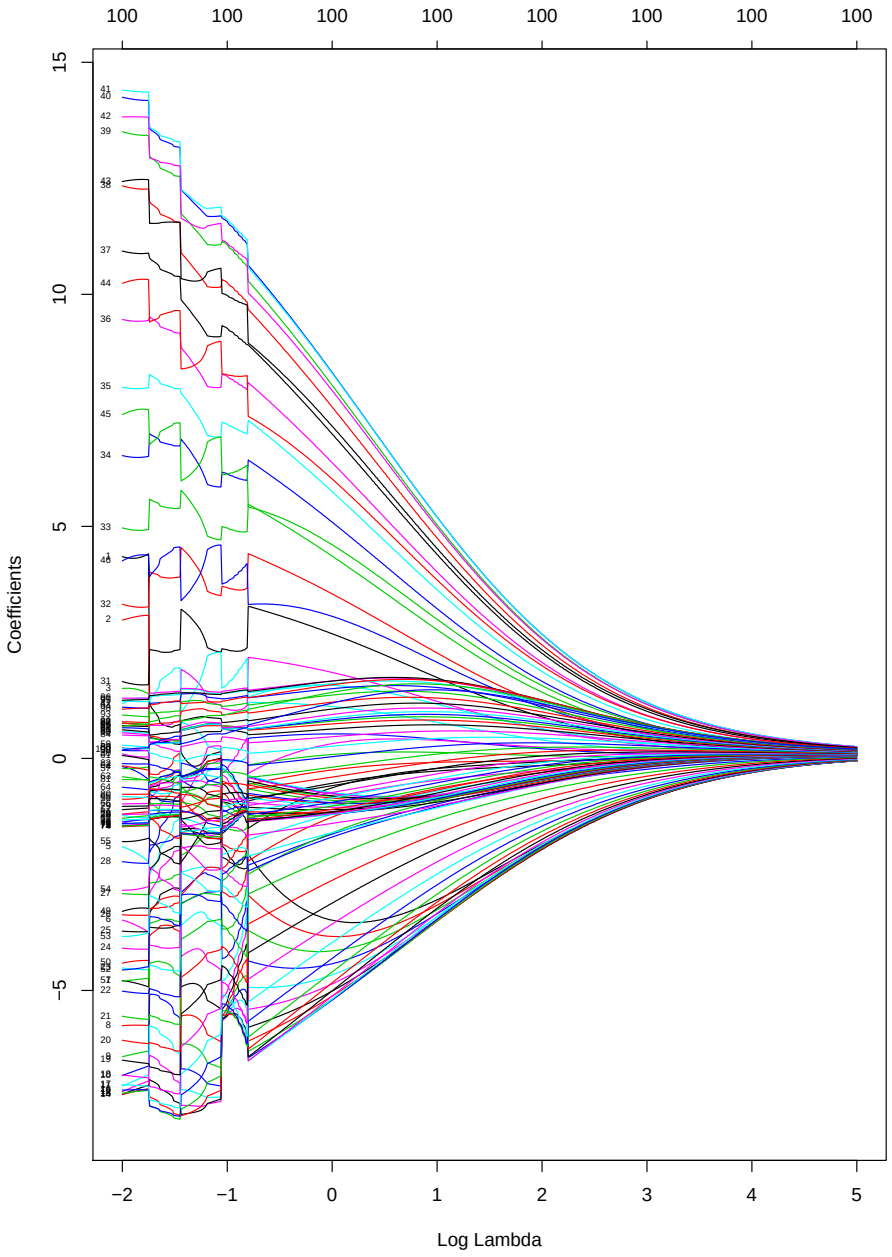

Figure 8: Ridge regression

The same thing was then done using LASSO instead of ridge regression. The

graph of how the coefficients change with lambda can be seen in figure 9. The graph shows how some coefficients at a time take the value 0 while a few keep their higher value. comparing this plot with the one for ridge regression shows how the two methods differ. Ridge decreases all coefficients, while LASSO keeps important ones high.

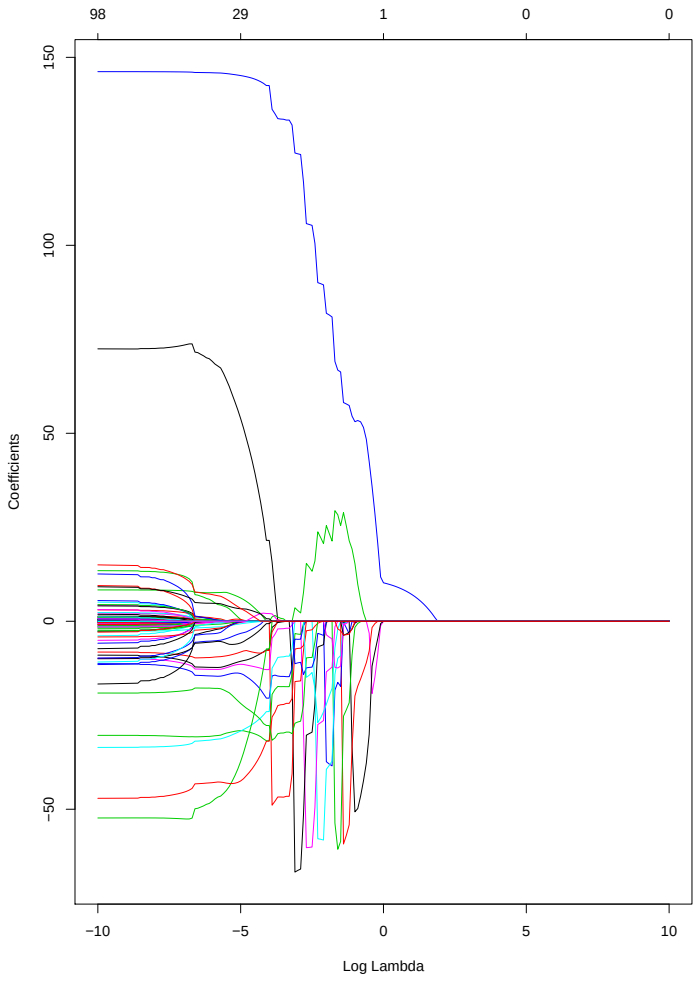

Figure 9: Ridge regression using LASSO

Cross-validation (CV) was then used to find the optimal LASSO model. In figure 10, the CV-score for each lambda can be seen.

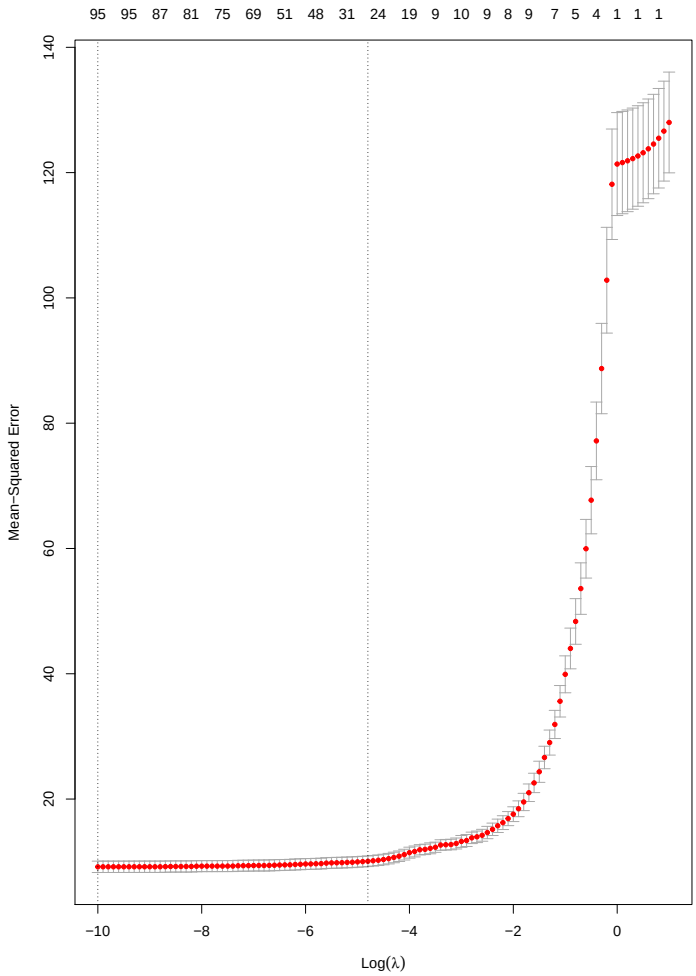

Figure 10: Cross validation for different lambdas

The lambda value with the lowest CV-score was 0, i.e log(lambda)=-inf. With this lambda, no penalty was used and as a result, all channels were used. The graph shows that the CV-score increases with an increasing lambda.

Comparing the results using stepAIC and cross-validation there is a dif-

ference in if any channels should be excluded or not. Using stepAIC, only 63 channels were selected while cross-validation did not exclude any. Therefore, it seems like stepAIC is a method more prone to reduce the number of variables compared to cross-validation.

Appendices

A Code appendix

A.1 Assignment 1

```
library("readxl")
library("kknn")
data <- read_excel("spambase.xlsx");

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

#2.
#Training data
g = glm(Spam~.,data=train)
pred = ifelse(predict(g)>0.5,1,0)
conf_matrix = table(pred, unlist(train[,49]))
mrate = 1 - (sum(diag(conf_matrix))/sum(conf_matrix))

#Test data
conf_matrix = table(pred, unlist(test[,49]))
mrate = 1 - (sum(diag(conf_matrix))/sum(conf_matrix))
conf_matrix

#3.
#Training data
g = glm(Spam~.,data=train)
pred = ifelse(predict(g)>0.8,1,0)

conf_matrix = table(pred, unlist(train[,49]))
mrate = 1 - (sum(diag(conf_matrix))/sum(conf_matrix))

# Test data
```

```
conf_matrix = table(pred, unlist(test[,49]))
mrate = 1 - (sum(diag(conf_matrix))/sum(conf_matrix))

#4
kknn.k30.test = kknn(formula=as.factor(Spam)~., train=train, test=test, k=30)
kknn.k30.train = kknn(formula=as.factor(Spam)~., train=train, test=train, k=30)
conf_matrix=table(train$Spam, kknn.k30.train$fitted.values)
missrate.k30.train = 1 - (sum(diag(conf_matrix))/sum(conf_matrix))
conf_matrix=table(test$Spam, kknn.k30.test$fitted.values)
missrate.k30.test = 1 - (sum(diag(conf_matrix))/sum(conf_matrix))

#5
kknn.k1.train = kknn(as.factor(Spam)~., train=train, test=train, k=1)
kknn.k1.test = kknn(as.factor(Spam)~., train=train, test=test, k=1)
conf_matrix=table(train$Spam, kknn.k1.train$fitted.values)
missrate.k1.train = 1 - (sum(diag(conf_matrix))/sum(conf_matrix))
conf_matrix=table(test$Spam, kknn.k1.test$fitted.values)
missrate.k1.test = 1 - (sum(diag(conf_matrix))/sum(conf_matrix))
```

A.2 Assignment 2

```
library("readxl")
set.seed(12345)
data = (read_excel("machines.xlsx"))

n = dim(data)[1]
plot(data)

theta.vector = seq(0,8,0.001)

calc_lhood = function(theta, x) {
  lhood = theta^n*exp(-theta*sum(x))
  log(lhood)
}

calc_bayesian_lhood = function(theta, x){
  p0 = 10*exp(-10*theta)
  lhood = theta^n*exp(-theta*sum(x))*p0
```

```
  log(lhood)
}


log.likelyhoods = calc_lhood(theta.vector, data$Length)
ind = which(log.likelyhoods==max(log.likelyhoods))
max = ind*0.001
plot(theta.vector, log.likelyhoods, col="Red",xlab="Theta values", ylab="Log likel
par(new=TRUE)
log.likelyhoods.6values = calc_lhood(theta.vector, data$Length[1:6])
ind = which(log.likelyhoods.6values==max(log.likelyhoods.6values))
max.6values = ind*0.001
plot(theta.vector, log.likelyhoods.6values, col="Blue", xlab="Theta values", ylab=

par(new=FALSE)

bay.likelyhoods = calc_bayesian_lhood(theta=theta.vector, x=data$Length)
plot(theta.vector, bay.likelyhoods, col="Yellow", xlab="Theta values", ylab = "Log

par(new=TRUE)
plot(theta.vector, log.likelyhoods, col="Red",xlab="Theta values", ylab="Log likel

bay.max = max(bay.likelyhoods) #0.912
std_max = max(log.likelyhoods)
bay.max_theta = theta.vector[which.max(bay.likelyhoods)] #0.912
std.max_theta = theta.vector[which.max(log.likelyhoods)] #1.126

new_obs = rexp(50, std.max_theta)

hist(new_obs, main = "Histogram of observations generated with optimal theta", xla
hist(as.numeric(data$Length), main="Histogram of original data", xlab = "Length",
```

A.3 Assignment 4

```
library("readxl")
library("MASS")
library("glmnet")
```

```
setwd("C:\\Users\\Victor\\Documents\\R Projects\\tdde01-labs\\LAB1")

data = read_excel("tecator.xlsx")

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

plot(data$Moisture, data$Protein, xlab="Moisture", ylab = "Protein")
#Quite linear

MSE.train = numeric(length = 6)
MSE.valid = numeric(length = 6)

for (i in 1:6){
  M = lm(Moisture~poly(Protein, i), data = train)
  s = summary(M)

  MSE.train[i] = mean(s$residuals^2)
  pred = predict(M, validation)
  MSE.valid[i] = mean((pred-validation$Moisture) ^2)
}

xlab = "Polynomial factor"
main="Mean square errors"
ylab = "MSE"
ylim = range(22:45)

plot(1:6, MSE.train, type="b",
      main = main,xlab = xlab,ylab=ylab,ylim=ylim,col = "Red"
     )
par(new=TRUE)
plot(1:6, MSE.valid,type="b", main=main, xlab = xlab, ylab=ylab, ylim=ylim, col="B
legend(2,1, c("Train", "Test"), col=c("Red","Blue"))

#4
```

```
fat = data$Fat
fatmodel = lm(fat~ . -(Sample + Protein + Moisture + Fat), data = data)
steps = stepAIC(fatmodel)
length(steps$coefficients) #64

#5
fat = data.frame(data)[,"Fat"]
coefficients = data.frame(data)[,2:101]
lambdas = exp(seq(5,-2, -0.01))
fit = glmnet(as.matrix(coefficients), fat, lambda=lambdas, alpha = 0)
summary(fit)
plot(fit, xvar="lambda",label=TRUE)

#6
lambdas = exp(seq(10, -10, -0.1))
lasso = glmnet(as.matrix(coefficients), fat, lambda=lambdas, alpha=1)
summary(lasso)
plot(lasso, xvar="lambda")

#7
cross_validation = cv.glmnet(as.matrix(coefficients), fat, lambda=exp(seq(-10, 1,0
min(cross_validation$lambda)
plot(cross_validation)
summary(cross_validation)
```