Rolf Sievert
2018-11-26

Computer Lab 1

Assignment 1

Description:
A data file called *spambase* containing email information and a spam classification (either 0 or 1). The dataset was split into two sets, training and test data.

Part 2

One model was created, trained on train data, with the spam classification as response. A prediction was made on the test data and train data, with a result greater than 0.5 counted as spam and a lower value as not spam. See tables below showing result on prediction.

Prediction on test data	**False**	**True**
False	791	146
True	97	336
Misclassification rate: 0.1773723		

Prediction on train data	**False**	**True**
False	803	142
True	81	344
Misclassification rate: 0.1627737		

It is reasonable that the misclassification rate of the train data is lower, since the model was trained with the same data.

Part 3

These tables show the same prediction described previously, but instead of greater than 0.5, greater than 0.9 is interpreted as spam.

Prediction on test data	**False**	**True**
False	936	1
True	427	6
Misclassification rate: 0.3124088		

Rolf Sievert
2018-11-26

Prediction on train data	**False**	**True**
False	944	1
True	419	6
Misclassification rate: 0.3065693		

This new value made a worse performance of the classification, as seen on the misclassification rates. Just as before, classification on training data gives better result.

Part 4

Using the kknn method, these results were given (with values > 0.5 interpreted as spam and K=30):
Prediction on test data: **Misclassification rate:** 0.329927
Prediction on train data: **Misclassification rate:** 0.1722628

This result is worse than in *Part 2*, which is reasonable since a lot of data is taken into account when classifying emails (it is a general model).

Part 5

Same as part 4, but with K=1:
Prediction on test data: **Misclassification rate:** 0.3459854
Prediction on train data: **Misclassification rate:** 0

This results in a perfect classification for training data, but that's because the model is very overfitted. The result is not nearly as good for the test data, which is expected since the model is overfitted for training data.

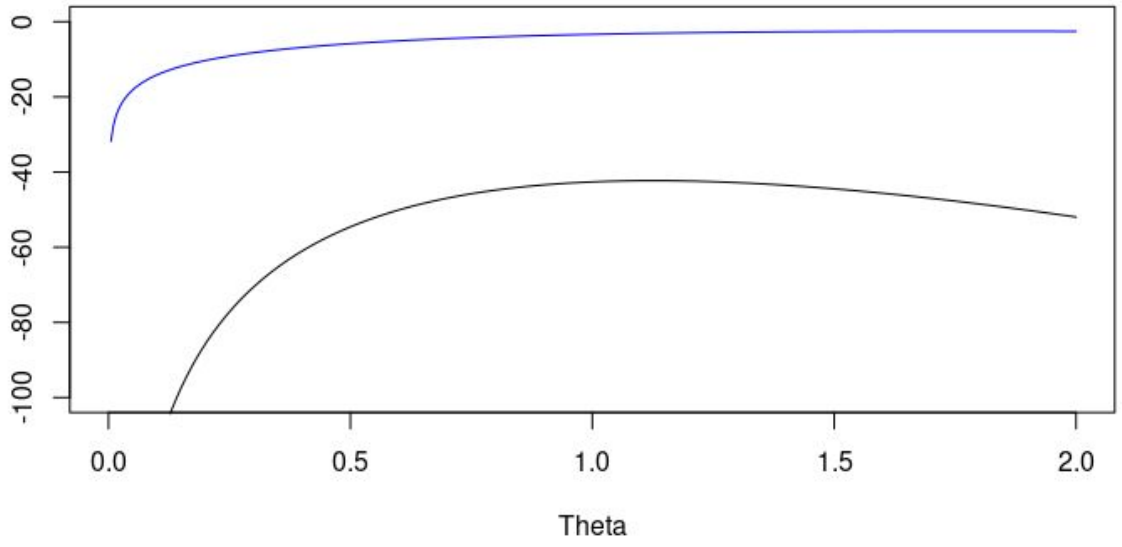
Rolf Sievert
2018-11-26

Assignment 2

A file called *machines* contains information about the lifetime of certain machines. It contains length information for all the machines.

Part 2 & 3

Using the probability model p(x|Theta) = Theta*exp(-Theta*x) where x is lifespan length. This is an exponential distribution of x. To calculate the maximum likelihood value of Theta, the log likelihood log(p(x|Theta)) was plotted, where x is a data vector, for a number of Thetas given by the sequence seq(from=0, to=2, by=0.005). This function (dark blue) is plotted in the graph below, together with a plot of the same function but with only the first 6 observations (from data set) instead of all.

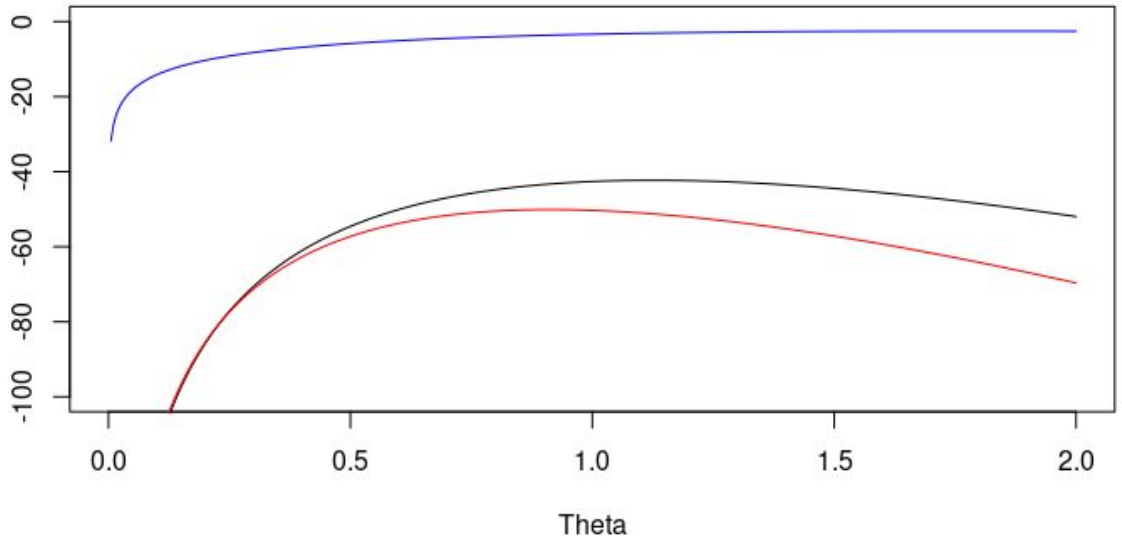


Max Theta for entire dataset: 1.125
Max Theta for first 6 observations: 1.785
The reliability increases when more data is available, it has a more obvious maximum (more pointy top in the graph).

Part 4

With a bayesian model p(x|Theta) = Theta*exp(-Theta*x) and a prior p(x) = 10*exp(-10*x), the function *l* was created, defined by *l*(Theta) = log(p(x|Theta)*p(x)), measuring posterior probability. The function *l* can be seen as the red line in the plot below.

Rolf Sievert
2018-11-26

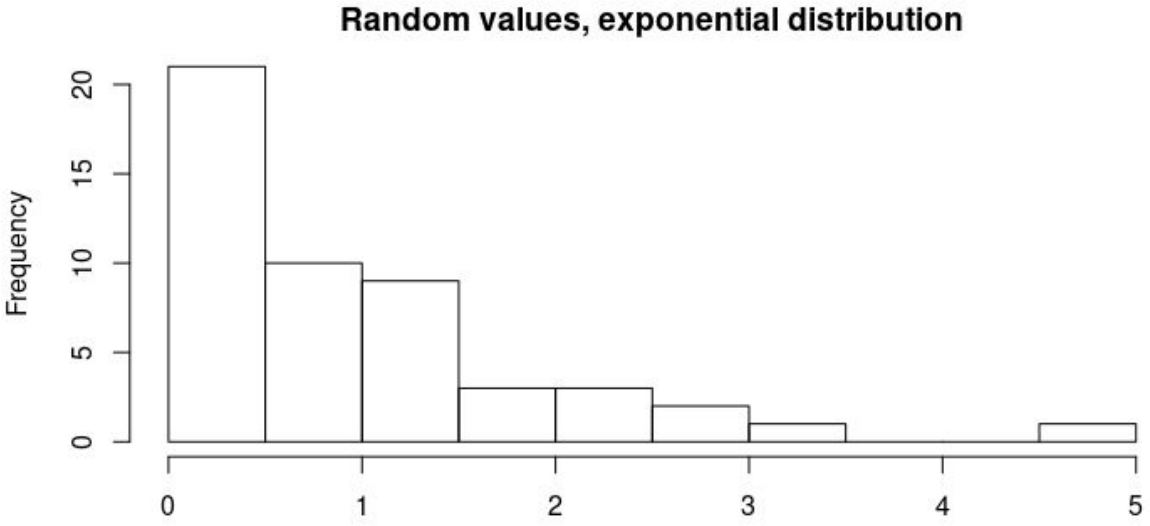


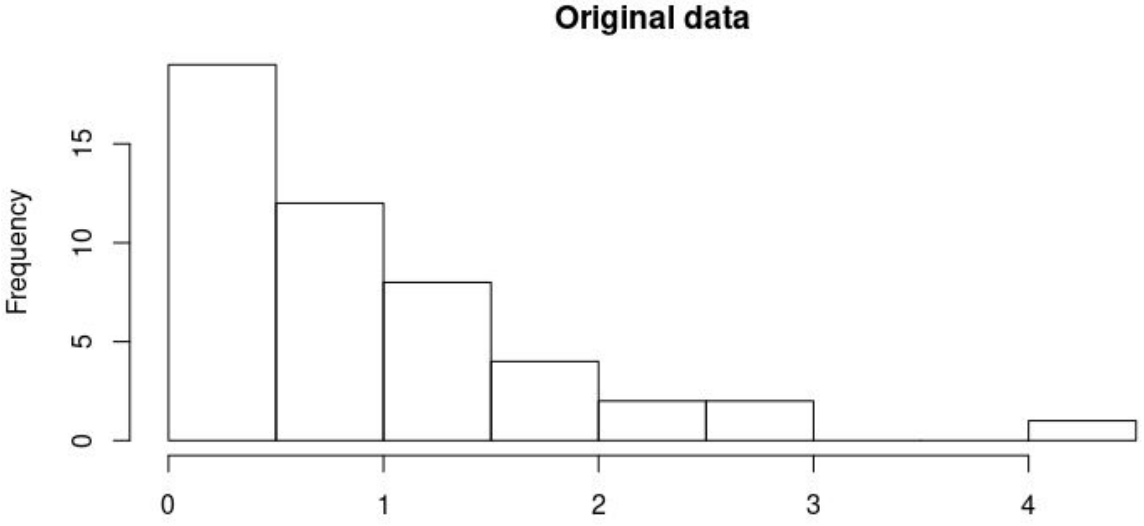
The optimal Theta for l was 0.91, which is lower than in *Part 2* and more well defined.

Part 5

Here are 50 new values generated randomly with exponential distribution and from this data set and the original data set, two histograms are created.

Rolf Sievert
2018-11-26





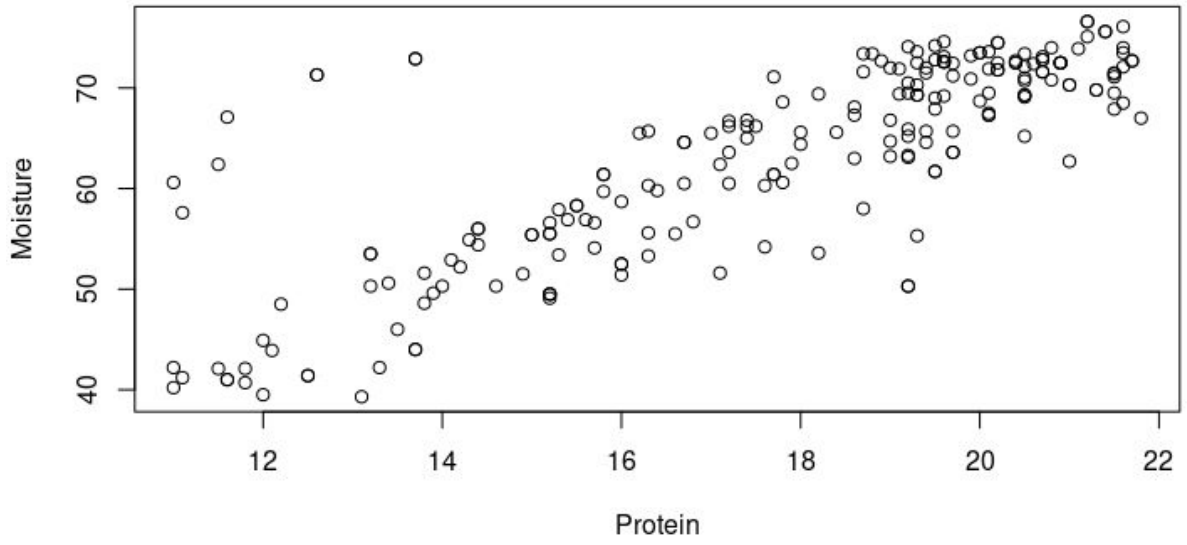
The randomly generated data and the original data looks very similar, and it can be assumed that the original data therefore is exponentially distributed.

Rolf Sievert
2018-11-26

Assignment 4

Part 1

This graph shows Moisture versus Protein, and as can be seen in the graph, a linear model can describe it (since the points are together shaped as a line).



Part 2

A probabilistic model that describes *Mi*:

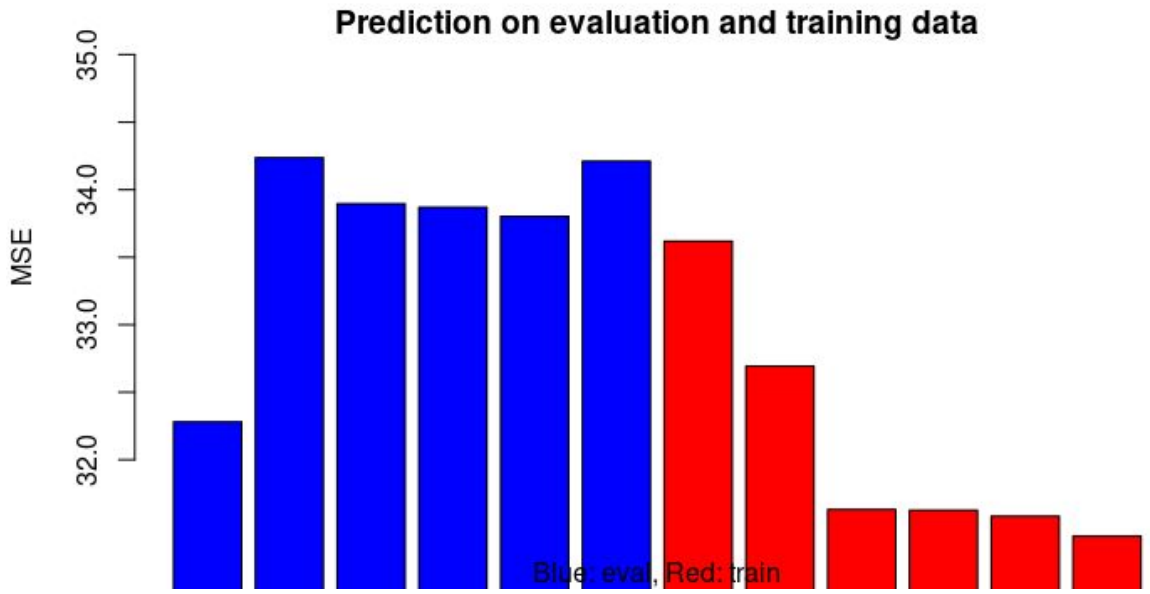
Mi = w0 + w1*x + w2*x^2 + … + w2*x^i + e, where e is error

The data is normally distributed and MSE is appropriate to use with normally distributed data.

Part 3

The data set was equally split into training and validation sets. Polynomial models were created with order 1 to 6. These were validated with both the training set and validation set. The MSE for each model on each validation set used in this case is shown in the graph below.

Rolf Sievert
2018-11-26



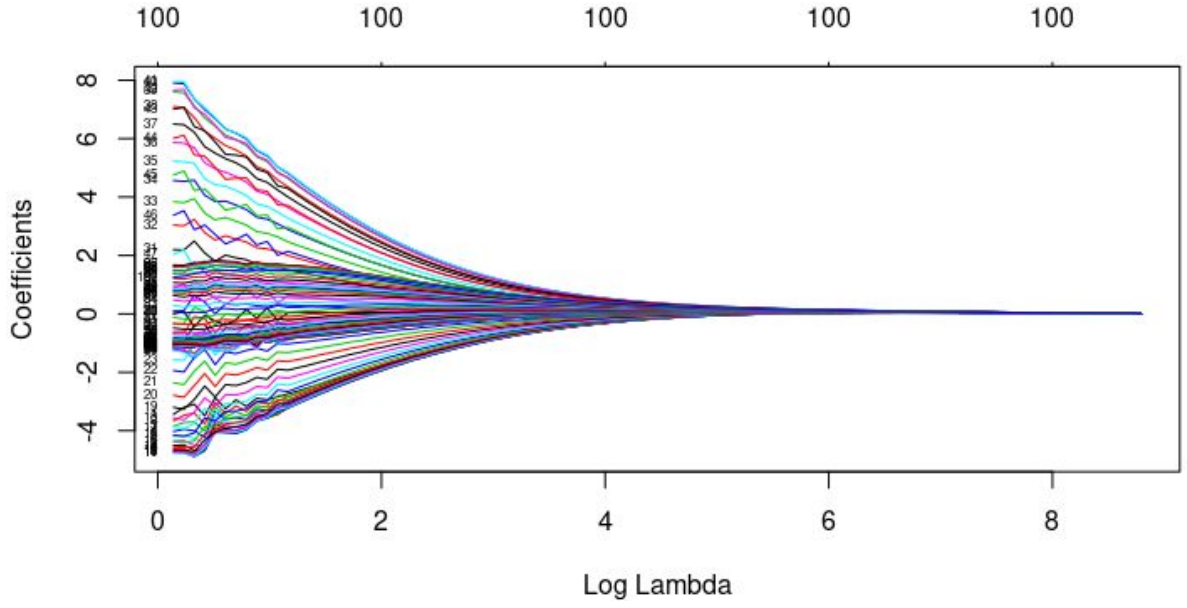
In the case of evaluation data, the linear model gives the best result, whereas the training data has the highest polynomial rank as best. There is high variance for high rank and high bias for low rank.

Part 4

By using stepAIC, the selection of variables was reduced to 63 (performed on the model in which Fat is response and Channel1-100 are predictors).

Part 5

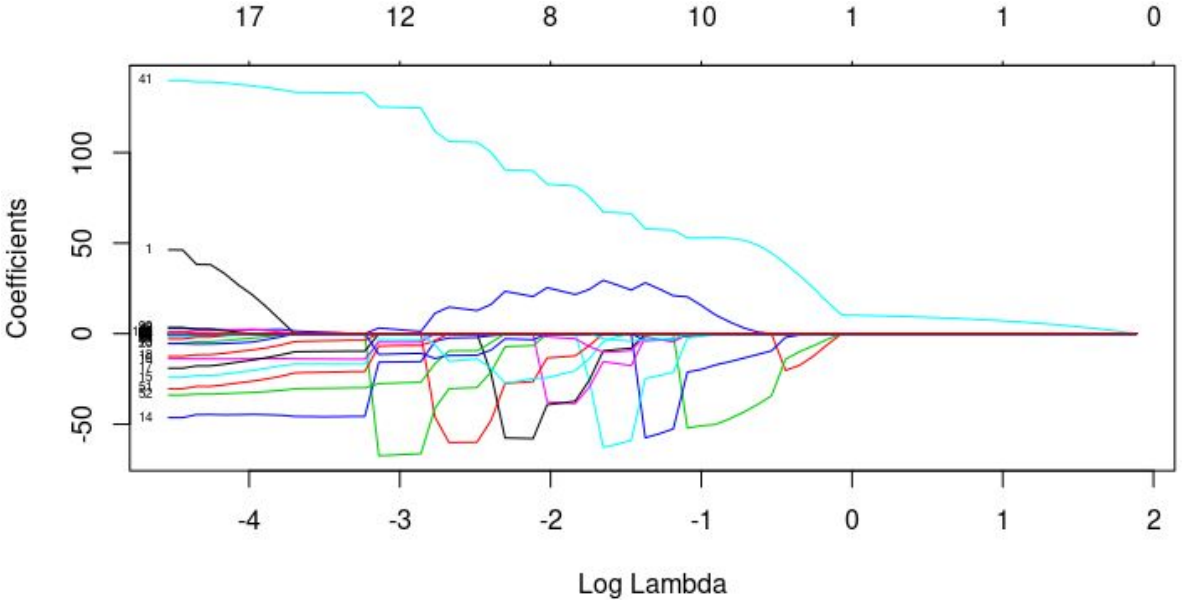
Using the same response and predictor variables as previously, a ridge regression was performed. Below is a graph showing how coefficients change depending on the penalty factor lambda.



Rolf Sievert
2018-11-26

Part 6

The below graph shows a plot were LASSO is used instead of Ridge regression.



Compared to *Part 5*, all values don't just decrease as lambda increases, they vary a lot more.

Part 7

Lambda=0 was selected as the optimal lambda for the optimal LASSO model when using cross-validation. The number of coefficients are the of course 100 (the whole data set), since none were reduced.

Part 8

Part 4 didn't give optimal result in this case, but it is more general than in *Part 7*. A more general model could be the preferred one as it is more simple and may make logical conclusions, whereas a complex model may use variables that doesn't have a logical connection to the result.

Rolf Sievert
2018-11-26

Appendix

Assignment 1

```
library("kknn")
# 1.1

# Read data
data = read.csv("spambase.csv")

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

# 1.2
# Create a General Linear Model
model = glm(formula = Spam~., data = train, family = 'binomial')

# Make prediction on test
predict_test = predict(model, test, 'response')
# Make prediction on train
predict_train = predict(model, train, 'response')
# Show result, match actual answers to predicted ones
table_test_0.5 = table(test$Spam, predict_test > 0.5)
table_train_0.5 = table(train$Spam, predict_train > 0.5)
cat("\nPrediction with GML model, test data. 0.5")
print(table_test_0.5)
cat("Misclassification rate: ")
cat((sum(table_test_0.5[2]) +
sum(table_test_0.5[3]))/sum(table_test_0.5[1:4]))
cat("\n\n");
cat("Prediction with GML model, train data. 0.5")
print(table_train_0.5)

cat("Misclassification rate:")
cat((sum(table_train_0.5[2]) +
sum(table_train_0.5[3]))/sum(table_train_0.5[1:4]))
cat("\n\n");

# 1.3
table_test_0.9 = table(test$Spam, predict_test > 0.9)
```

Rolf Sievert
2018-11-26

```
table_train_0.9 = table(train$Spam, predict_train > 0.9)
cat("Prediction with GML model, test data. 0.9")
print(table_test_0.9)
cat("Misclassification rate:")
cat((sum(table_test_0.9[2]) +
sum(table_test_0.9[3]))/sum(table_test_0.9[1:4]))
cat("\n\n");

cat("Prediction with GML model, train data. 0.9")
print(table_train_0.9)
cat("Misclassification rate:")
cat((sum(table_train_0.9[2]) +
sum(table_train_0.9[3]))/sum(table_train_0.9[1:4]))
cat("\n\n");

# 1.4
K = 30
kknn_test = kknn(formula = Spam~., train = train, test = test, k = K)
kknn_train = kknn(formula = Spam~., train = train, test = train, k = K)
table_kknn_test = table(test$Spam, kknn_test$fitted.values > 0.5)
table_kknn_train = table(train$Spam, kknn_train$fitted.values > 0.5)
cat("KKNN result using test data test for K=30")
print(table_kknn_test)
cat("Misclassification rate:")
cat((sum(table_kknn_test[2]) +
sum(table_kknn_test[3]))/sum(table_kknn_test[1:4]))
cat("\n\n");
cat("KKNN result using test data train for K=30")
print(table_kknn_train)
cat("Misclassification rate:")
cat((sum(table_kknn_train[2]) +
sum(table_kknn_train[3]))/sum(table_kknn_train[1:4]))
cat("\n\n");

# 1.5
K = 1
kknn_test = kknn(formula = Spam~., train = train, test = test, k = K)
kknn_train = kknn(formula = Spam~., train = train, test = train, k = K)
table_kknn_test = table(test$Spam, kknn_test$fitted.values > 0.5)
table_kknn_train = table(train$Spam, kknn_train$fitted.values > 0.5)
cat("KKNN result using test data test for K=1")
print(table_kknn_test)
cat("Misclassification rate:")
cat((sum(table_kknn_test[2]) +
sum(table_kknn_test[3]))/sum(table_kknn_test[1:4]))
cat("\n\n");
```

Rolf Sievert
2018-11-26

```
cat("KKNN result using test data train for K=1")
print(table_kknn_train)
cat("Misclassification rate:")
cat((sum(table_kknn_train[2]) +
sum(table_kknn_train[3]))/sum(table_kknn_train[1:4]))
```

Assignment 2

```
# 2.1
# Read data
data = read.csv("machines.csv")

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

# 2.2 Exponential distribution
pxt = function(x, theta){
    # Prod returns product of all arguments
    return (log(prod(theta*exp(-theta*x))))
}
thetas = seq(from=0, to=2, by=0.005)
theta_fn = sapply(thetas, function(theta) pxt(data$Length, theta))
plot(thetas, theta_fn, xlabel="X", ylabel="Y", type="p")
theta_max = thetas[which.max(theta_fn)]
cat("Max theta value: ")
cat(theta_max)
cat("\n\n")

# 2.3
short_data = data$Length[1:6]
short_theta_fn = sapply(thetas, function(theta) pxt(short_data, theta))
par(mar=c(4,4,2,1))
plot(thetas, theta_fn, xlab="Theta", ylab="", type="l", ylim=c(-100, 0))
lines(thetas, short_theta_fn, col="blue")
short_theta_max = thetas[which.max(short_theta_fn)]
cat("Max theta value of short dataset: ")
cat(short_theta_max)
cat("\n\n")

# 2.4
l = function(x, theta){
    return (log(pxt(x, theta)*pt(theta)))
```

```
}
theta_fn = sapply(thetas, function(theta) l(data$Length, theta))
l_max = thetas[which.max(theta_fn)]
lines(thetas, theta_fn, col="red")
cat("Max theta value: ")
cat(l_max)
cat("\n\n")

# 2.5
new_data = rexp(50, theta_max)
hist(new_data, main="Random values, exponential distribution", xlab="")
hist(data$Length, main="Original data", xlab="")
```

Assignment 4

```
# Read data
data = read.csv("tecator.csv")

# 4.1
plot(data$Protein, data$Moisture, xlab="Protein", ylab="Moisture", type =
"p")

# 4.3
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
eval=data[-id,]

mse_eval_list = c()
mse_train_list = c()
for (x in c(1, 2, 3, 4, 5, 6)){
    model=lm(formula = Moisture ~ poly(Protein, x, raw = TRUE), data =
train)
    # Eval data
    mse = mean((eval$Moisture - predict(model, eval))^2)
    mse_eval_list=c(mse_eval_list, mse)

    # Train data2
    mse = mean((train$Moisture - predict(model, train))^2)
    mse_train_list=c(mse_train_list, mse)

}
counts <- table(mse_eval_list, mse_train_list)
barplot(c(mse_eval_list, mse_train_list),
```

```
        main="Prediction on evaluation and training data",
        ylab="MSE",
        xlab="Blue: eval, Red: train",
        col=c("blue","blue","blue","blue","blue","blue","red", "red",
"red", "red", "red", "red"),
        ylim=c(32, 35))

# 4.4
channels = data[,2:102]
model = lm(formula = Fat~. , data = channels)
step_alg = stepAIC(model, direction="both", trace = FALSE)
cat("Length: ", length(coef(step_alg)) -1, "\nCoefficients: ",
coef(step_alg))

# 4.5
covariates=channels[,1:100]
response=channels[, 101]
model0=glmnet(as.matrix(covariates),
              response,
              alpha=0,
              family="gaussian")
plot(model0, xvar="lambda", label=TRUE)

# 4.6
# LASSO, alpha = 1
model1=glmnet(as.matrix(covariates),
              response,
              alpha=1,
              family="gaussian")
plot(model1, xvar="lambda", label=TRUE)

# 4.7
model=cv.glmnet(as.matrix(covariates),
                response,
                lambda = seq(0, 5, 0.005),
                alpha=1,
                family="gaussian")
cat("\n\n\nMin lambda: ", model$lambda.min)
plot(model)
cat("\nNumber of coefficients of lambda min: ")
coeffs = coef(model, s=model$lambda.min)
cat(sum (coeffs[2:101,] != 0))
```