Oskar Hidén- oskhi872 Samuel Persson- sampe028 Oscar Moberg- oscmo462

Group B15

All Group members individually created all assignments. After everyone in the group was done with all three exercises. The group had two meetings comparing results and discussing the questions in the lab. The results and code for each assignment is created by:
Assignment 1: Samuel Persson
Assignment 2: Oscar Moberg
Assignment 4: Oskar Hidéen

Assignment 1 (Samuel Persson)

Step 1. See code.
Step 2.

		Prediction	
Test. Threshold 0.5		0	1
Actual	0	808	143
	1	92	327
		Prediction	
Train. Threshold 0.5		0	1
Actual	0	804	127
	1	93	346

Misclassification train: 0.1605839
Misclassification test: 0.1715328

Training data misclassification is lower than the test because the model is more fitted to training data than test.

Step 3.

		Prediction	
Test. Threshold 0.8		0	1
Actual	0	931	20
	1	314	105
		Prediction	
Train. Threshold 0.8		0	1
Actual	0	921	10
	1	333	106

Misclassification train: 0.250365
Misclassification test: 0.2437956

The new rule made the prediction of more emails as non-spam. This made the misclassification increase because even though some emails had a higher probability of being spam they were judged as non-spam. The test misclassification rate is also lower because of the threshold not being at the “probability threshold” of 0.5.

Step 4.
Misclassification train: 0.1671533
Misclassification test: 0.3131387

The misclassification of train was approximately the same as in step 2 while test was higher. This is probably because we find weights regarding importance of different coefficients in step 2 while in step 4 we just look at distance.

Step 5.
Misclassification train: 0
Misclassification test: 0.3591241

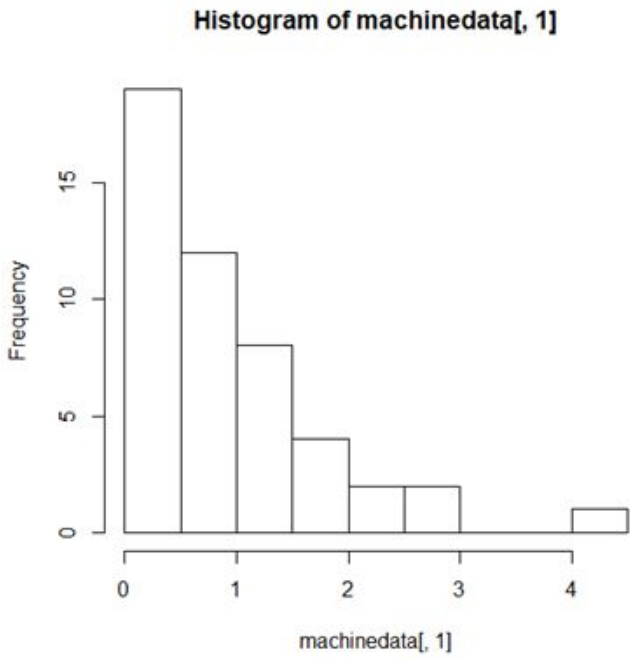
The misclassification rate of train went to 0 because we only choose the one point that is closest, which is itself. The Misclassification rate for test data went up a bit, probably because in this case, more data to compare with gives a better prediction

Assignment 2 (Oscar Moberg)

Step 2

In order to determine the distribution type of the data you can print a histogram of it and then analyse the result. Having a look at the histogram tells us that the data follows a exponential distribution which we also can tell by looking at the formula given to us:

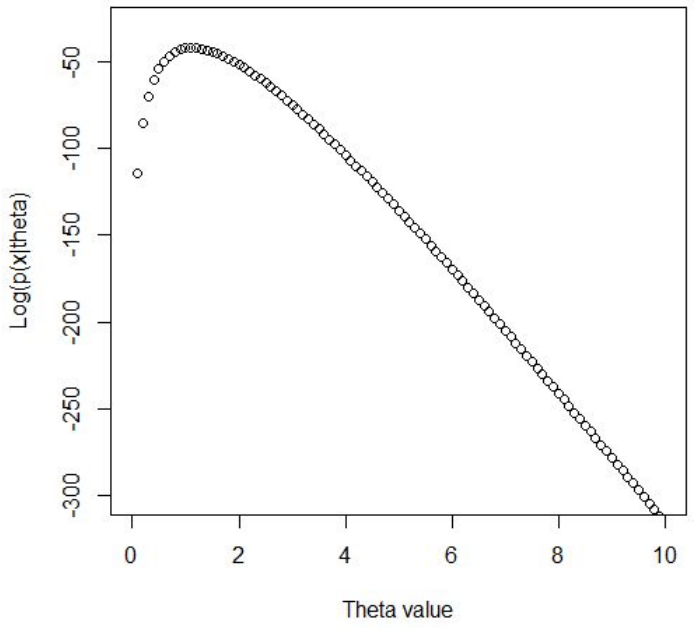
$$p(x|\theta) = \theta e^{-\theta x}$$



Function for computing log(p(x|theta)) for a given theta and data vector x:

```
loglikely = function(theta, data){
  probability = 0
  data = data$Length
  for (data_value in data) {
    probability = probability + log(theta*exp(-theta*data_value))
  }
  return(probability)
}
```

Plot showing the dependence of log-likelihood on theta:



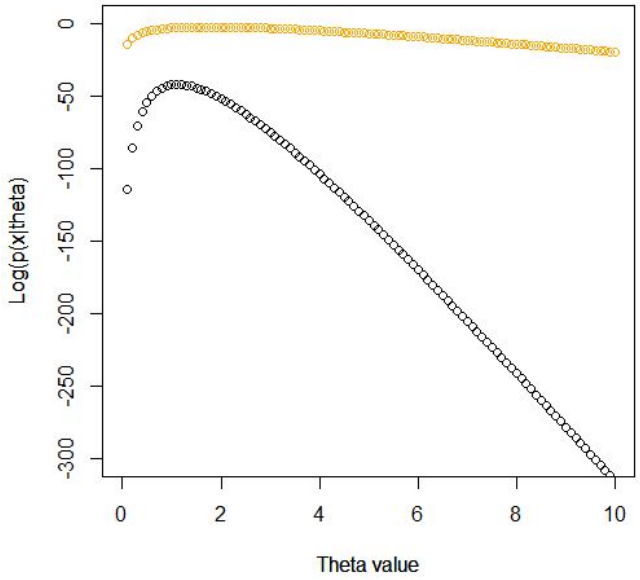
According to the plot the maximum likelihood value of theta = 1.1

Step 3

Repeat step 2 but use only the 6 first observations of the data:

Plot showing both likelihood curves:

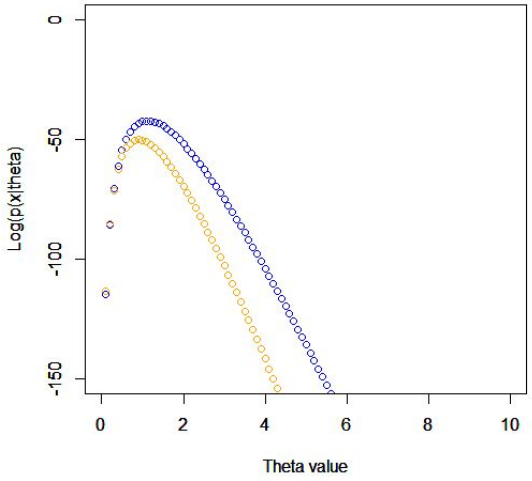
- The black plot represents our old result in step 2 of this assignment
- The orange plot is the new result when only using the first 6 observations.



From the plot we can see that the orange curve (6 observations) have a higher log-likelihood than using all the observations. This is due to having only 6 observations (compared to 48) it's not as complex as using all 48 observations, however 6 observations do not give us as much information and the curve representing 48 observations probably gives us a better fit for a given theta.

Step 4

What kind of measure is computed using l(theta) = log(p(x|theta)p(theta))?:

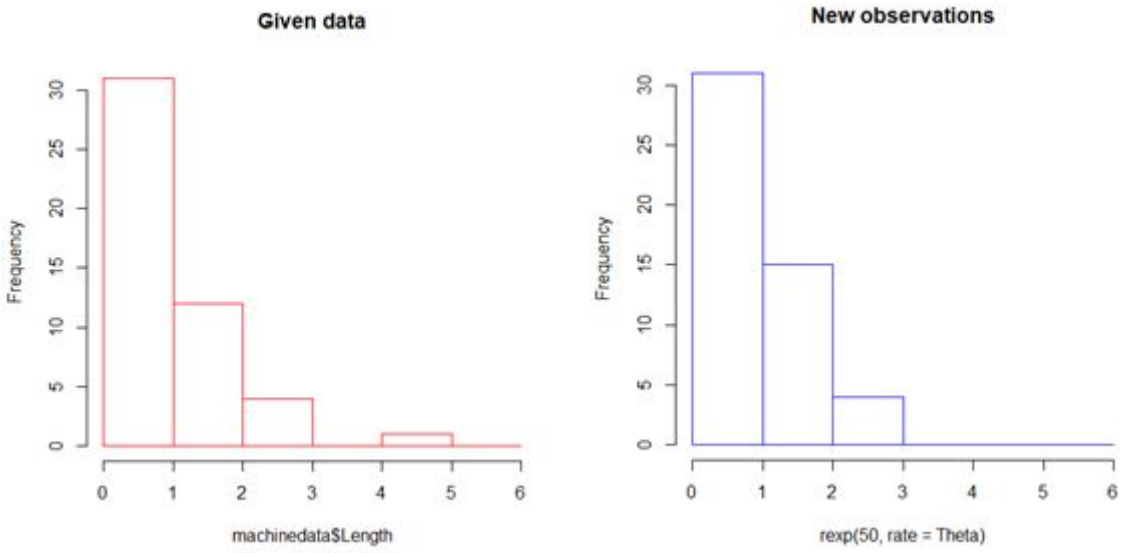


The blue curve is the log-likelihood for a given theta and the orange curve is representing the l(theta).

The measure we got is proportionally equal to the probability of what value theta is given a certain set of data.

The optimal theta was found to be 0.9 using this formula.

Step 5

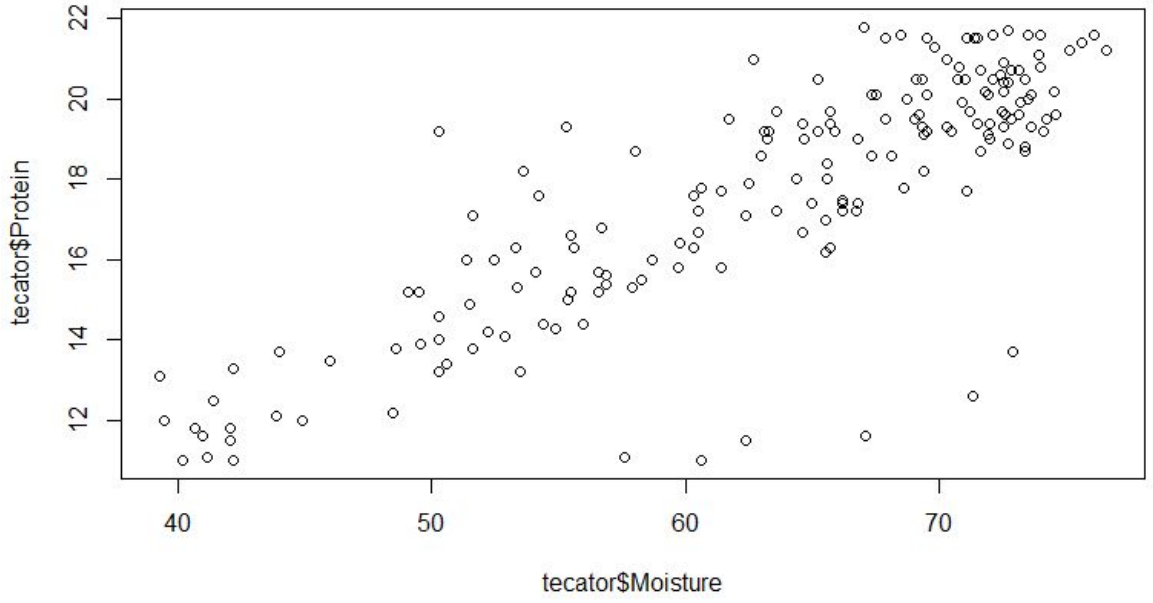


Conclusions: The red histogram shows the provided data and the blue is our newly generated data. Since we used the Theta value found to be the best suitable for the data provided to us when generating this new data, the new data highly resembles the old data. Thus we can see that the theta value we found actually was a good value in order to describe the given data.

Assignment 4 (Oskar Hidén)

Step 1

Plot of Moisture vs Protein:



Seems to be a linear dependence, which is why a linear module could be good. Some outliers can be found that deviates from the rest of the data.

Step 2

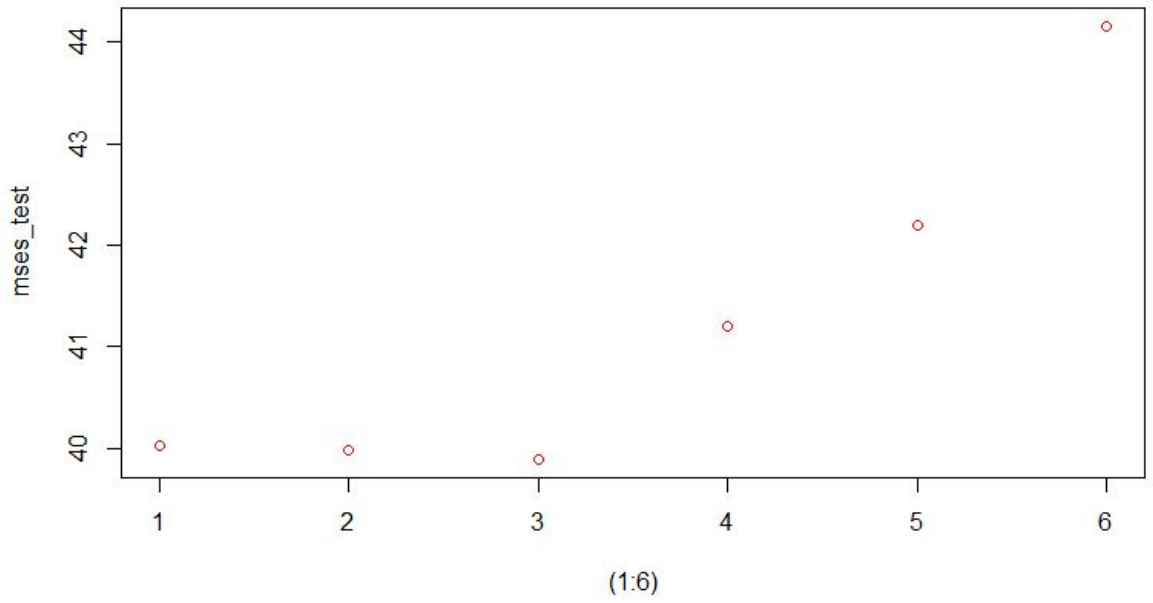
The models is described as:

$$M_i = \sum_{k=1}^{i} w_k * (Protein)^k + \varepsilon$$

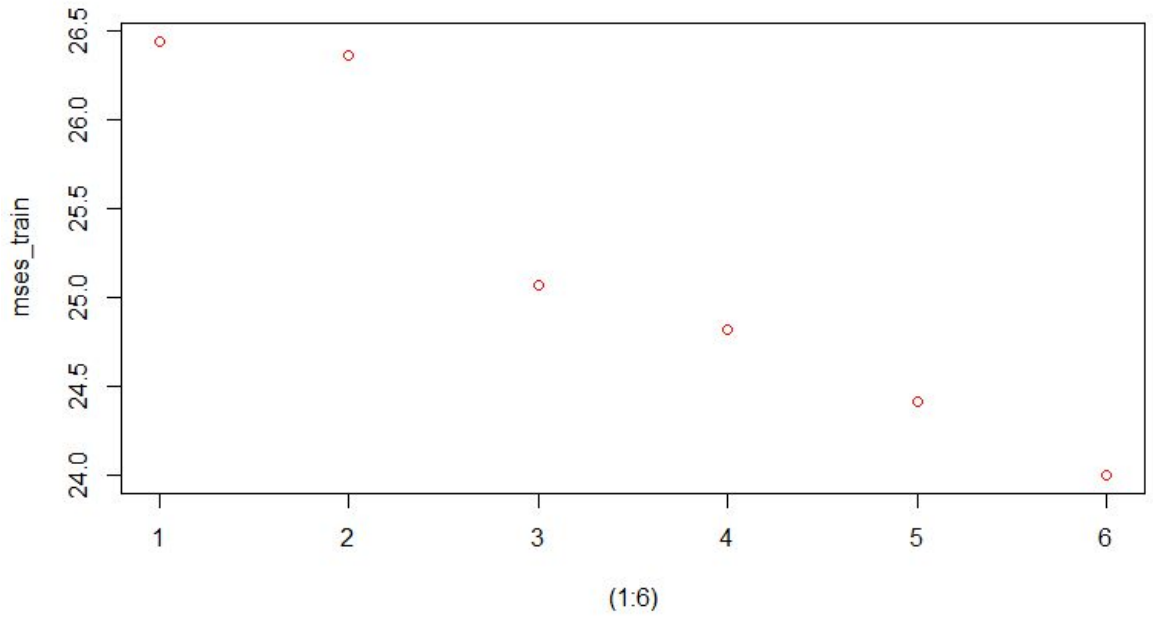
Mean Square Error criterion is appropriate to use to fit this model to training data since: Maximum likelihood method, to find the best model, is equivalent to minimizing MSE, given that predictions is normally distributed.

Step 3

MSE-plot of test data:



MSE-plot of training data:



According to the plot, i = 3 has the lowest MSE for test data. Therefore it is considered to be the best model. MSE values change since we fit the training data better with higher i. Therefore, MSE for training data is decreased for higher values of i. But, MSE for training data is decreased to i = 3 and then increased for higher values of i. In the plot, models with low values of i has high bias, and low bias for high values of i. When bias is decreased variance is increased in the models.

Step 4

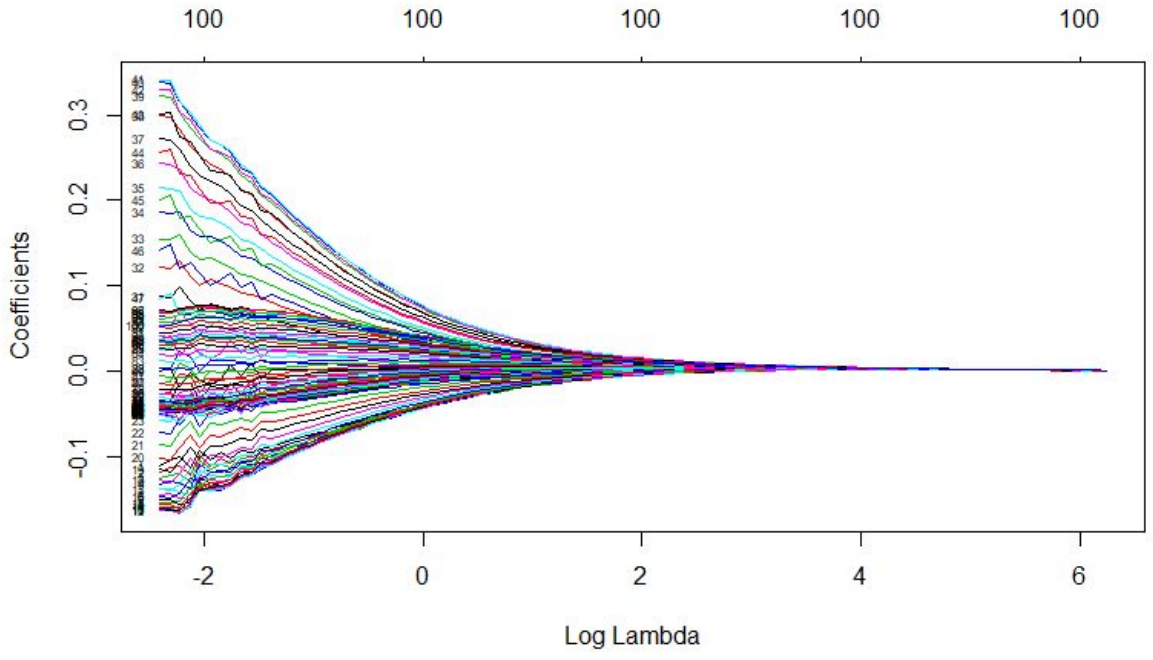
```
Final Model:
Fat ~ Channel1 + Channel2 + Channel4 + Channel5 + Channel7 +
    Channel8 + Channel11 + Channel12 + Channel13 + Channel14 +
    Channel15 + Channel17 + Channel19 + Channel20 + Channel22 +
    Channel24 + Channel25 + Channel26 + Channel28 + Channel29 +
    Channel30 + Channel32 + Channel34 + Channel36 + Channel37 +
    Channel39 + Channel40 + Channel41 + Channel42 + Channel45 +
    Channel46 + Channel47 + Channel48 + Channel50 + Channel51 +
    Channel52 + Channel54 + Channel55 + Channel56 + Channel59 +
    Channel60 + Channel61 + Channel63 + Channel64 + Channel65 +
    Channel67 + Channel68 + Channel69 + Channel71 + Channel73 +
    Channel74 + Channel78 + Channel79 + Channel80 + Channel81 +
    Channel84 + Channel85 + Channel87 + Channel88 + Channel92 +
    Channel94 + Channel98 + Channel99
```

63 variables were selected.

Step 5

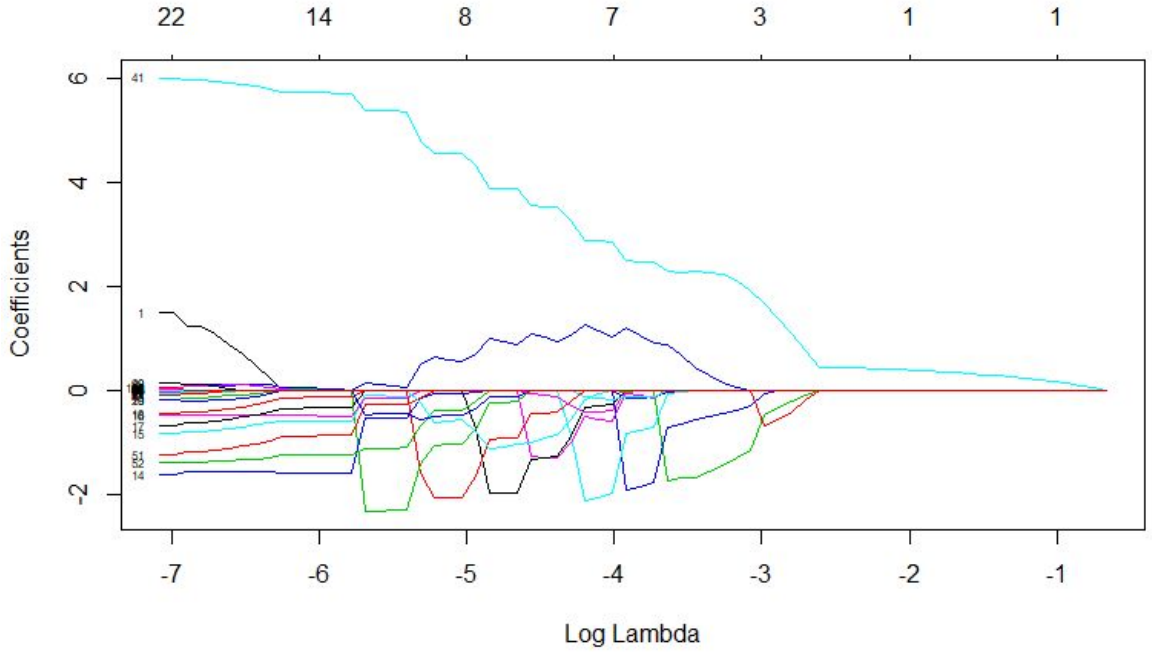
In the following steps scale() is used to center data when the matrix is created.

Plot of coefficients dependencies for ridge-regression:



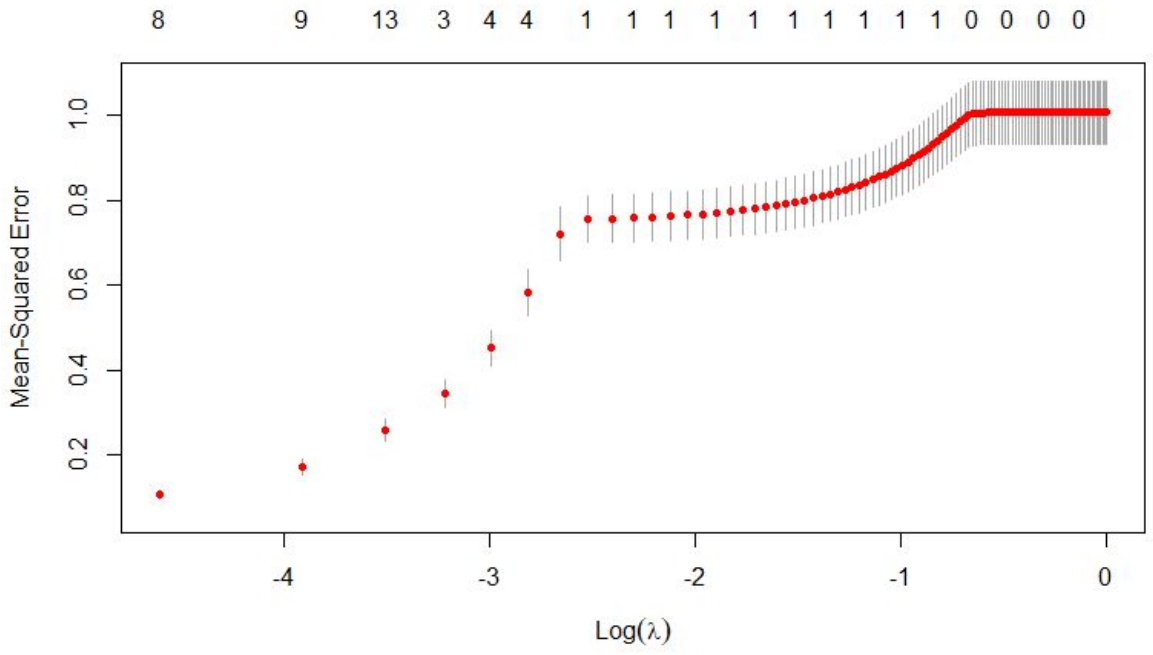
Step 6

Plot of coefficients dependencies for lasso-regression:



Lasso is setting many variable weights as 0 which makes the model dependent on fewer variables. Ridge use all variables but some of them has small weights.

Step 7



Lambda optimum is found at lambda = 0, the model then becomes a ordinary least square regression model, therefore the model will use all variables. MSE is increased for higher lambda values.

Step 8

StepAIC(step 4) use 63 while Lasso-regression(step 7) use all variables. Some of the variable weights in the lasso-model are quite low. And since stepAIC is discrete it is likely that these variables will not be used in the stepAIC model.

Appendix

Assignment 1 (Samuel Persson)

```
#----Assignment1----
missclass = function(X,X1){
  n=length(X)
  return(1-sum(diag(table(X,X1)))/n)
}

# Question 1
data =read.csv2("spambase.csv")
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
test=data[-id,]

#Question 2
fit = glm(formula=Spam~ ., data=train, family=binomial())

probabilityPredictTrain2 = predict(fit, newdata=train, type='response')
probabilityPredictTest2 = predict(fit, newdata=test, type='response')

discretePredictTrain2 = ifelse(probabilityPredictTrain2>0.5,"1","0")
discretePredictTest2 = ifelse(probabilityPredictTest2>0.5,"1","0")

confusionMatrixTrain2 = table(train [,"Spam"], discretePredictTrain2)
confusionMatrixTest2 = table(test[,"Spam"],discretePredictTest2)

missClassTrain2 = missclass(train[,"Spam"], discretePredictTrain2)
missClassTest2 = missclass(test[,"Spam"], discretePredictTest2)

print(confusionMatrixTest2)
print(confusionMatrixTrain2)
print(missClassTrain2)
print(missClassTest2)

#Question3

discretePredictTest3 = ifelse(probabilityPredictTest2>0.8, "1", "0")
discretePredictTrain3 = ifelse(probabilityPredictTrain2>0.8, "1", "0")
```

```
confusionMatrixTrain3 = table(train [,"Spam"], discretePredictTrain3)
confusionMatrixTest3 = table(test[,"Spam"],discretePredictTest3)

missClassTrain3 = missclass(train[,"Spam"], discretePredictTrain3)
missClassTest3 = missclass(test[,"Spam"], discretePredictTest3)

print(confusionMatrixTest3)
print(confusionMatrixTrain3)

print(missClassTrain3)
print(missClassTest3)

#Question4
libary(kknn)
kknnModelTrain4 = kknn(formula=as.factor(Spam)~ ., train=train, test=train, k=30)
kknnModelTest4 = kknn(formula=as.factor(Spam)~., train=train, test=test, k=30)

missClassTrain4 = missclass(train[,"Spam"], kknnModelTrain4$fitted.values)
missClassTest4 = missclass(test[,"Spam"], kknnModelTest4$fitted.values)

print(missClassTrain4)
print(missClassTest4)

#Question5

kknnModelTrain5 = kknn(formula=as.factor(Spam)~ ., train=train, test=train, k=1)
kknnModelTest5 = kknn(formula=as.factor(Spam)~., train=train, test=test, k=1)

missClassTrain5 = missclass(train[,"Spam"], kknnModelTrain5$fitted.values)
missClassTest5 = missclass(test[,"Spam"], kknnModelTest5$fitted.values)

print(missClassTrain5)
print(missClassTest5)
```

Assignment 2 (Oscar Moberg)

```
# ----- STEP 1 -----
machinedata = read.csv2("machinesdata.csv", header = TRUE)

#----- Plotting the data to determine the most suitable distribution -----
hist(machinedata[,1])

# ----- STEP 2 ------
```

```
plotvector = changeTheta(machinedata)
maxtheta = (which.max(plotvector))*0.1 - 0.1
plot(seq(from=0, to=10, by = 0.1), (plotvector), xlab = "Theta value", ylab = "Log(p(x|theta)",
xlim = c(0,10), ylim = c(-300, 0))

par(new = TRUE)

# ----- STEP 3 -----

kik = as.data.frame(machinedata[1:6,])
colnames(kik) = c("Length")
plotvector2 = changeTheta(kik)
maxtheta2 = (which.max(plotvector2))*0.1 - 0.1

plot(seq(from=0, to=10, by = 0.1), (plotvector2), col = "Orange", xlab = "Theta value", ylab =
"Log(p(x|theta)", xlim = c(0,10), ylim = c(-300, 0), axes = FALSE)

# ----- STEP 4 -----

plotvector = changeTheta(machinedata)
maxtheta = (which.max(plotvector))*0.1 - 0.1

plot(seq(from=0, to=10, by = 0.1), (plotvector), xlab = "Theta value", ylab = "Log(p(x|theta)",
col = "Blue", xlim = c(0,10), ylim = c(-150, 0))
axis(2, ylim = c(-300, 0), col = "blue")
par(new = TRUE)
lambda = 10

plot_lambda = changeTheta_lambda(machinedata, lambda)
optimaltheta = (which.max(plot_lambda))*0.1 - 0.1

plot(seq(from=0, to=10, by = 0.1), (plot_lambda), col = "Orange", xlab = "Theta value", ylab =
"Log(p(x|theta)", xlim = c(0,10), ylim = c(-150, 0))

# ----- STEP5 -----

Theta = 1.1
hist(rexp(50,rate=Theta),border = "blue", xlim=c(0,6), breaks=c(0,1,2,3,4,5,6), main = "New
observations")
hist(machinedata$Length, border= "red", xlim=c(0,6), breaks=c(0,1,2,3,4,5,6), main = "Given
data")

par(mfrow=c(1,2))
hist(machinedata[,1])
hist(rexp(50, rate = Theta))
```

Assignment 4 (Oskar Hidén)

```
tecator <- read_csv2("C:/Users/oskar/OneDrive/Universitet/Linköping
Universitet/År4/Machine learning/Lab 1/tecator.csv")

#-------1---------
plot(tecator$Moisture, tecator$Protein)

#-------3---------
n=dim(tecator)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
t_train=tecator[id,]
t_test=tecator[-id,]

mse = function(x, x_pred){
  squared_error = (x-x_pred)^2
  mse = sum(squared_error)/length(x)
  return(mse)
}


model_i_1 = glm(Moisture~Protein, data=t_train)
model_i_2 = glm(Moisture~Protein + I(Protein^2), data=t_train)
model_i_3 = glm(Moisture~Protein + I(Protein^2) + I(Protein^3), data=t_train)
model_i_4 = glm(Moisture~Protein + I(Protein^2) + I(Protein^3) + I(Protein^4), data=t_train)
model_i_5 = glm(Moisture~Protein + I(Protein^2) + I(Protein^3) + I(Protein^4) + I(Protein^5),
data=t_train)
model_i_6 = glm(Moisture~Protein + I(Protein^2) + I(Protein^3) + I(Protein^4) + I(Protein^5)
+ I(Protein^6), data=t_train)

#predictions for test data
pred_i_1 = predict(model_i_1, newdata = t_test)
pred_i_2 = predict(model_i_2, newdata = t_test)
pred_i_3 = predict(model_i_3, newdata = t_test)
pred_i_4 = predict(model_i_4, newdata = t_test)
pred_i_5 = predict(model_i_5, newdata = t_test)
pred_i_6 = predict(model_i_6, newdata = t_test)

#predictions for train data
pred_i_1_train = predict(model_i_1, newdata = t_train)
pred_i_2_train = predict(model_i_2, newdata = t_train)
pred_i_3_train = predict(model_i_3, newdata = t_train)
pred_i_4_train = predict(model_i_4, newdata = t_train)
```

```
pred_i_5_train = predict(model_i_5, newdata = t_train)
pred_i_6_train = predict(model_i_6, newdata = t_train)

#mse for test data
mse_test_1 = mse(t_test$Moisture, pred_i_1)
mse_test_2 = mse(t_test$Moisture, pred_i_2)
mse_test_3 = mse(t_test$Moisture, pred_i_3)
mse_test_4 = mse(t_test$Moisture, pred_i_4)
mse_test_5 = mse(t_test$Moisture, pred_i_5)
mse_test_6 = mse(t_test$Moisture, pred_i_6)

#mse for train data
mse_test_1_train = mse(t_train$Moisture, pred_i_1_train)
mse_test_2_train = mse(t_train$Moisture, pred_i_2_train)
mse_test_3_train = mse(t_train$Moisture, pred_i_3_train)
mse_test_4_train = mse(t_train$Moisture, pred_i_4_train)
mse_test_5_train = mse(t_train$Moisture, pred_i_5_train)
mse_test_6_train = mse(t_train$Moisture, pred_i_6_train)

print(mse_test_1)
print(mse_test_2)
print(mse_test_3)
print(mse_test_4)
print(mse_test_5)
print(mse_test_6)

print(mse_test_1_train)
print(mse_test_2_train)
print(mse_test_3_train)
print(mse_test_4_train)
print(mse_test_5_train)
print(mse_test_6_train)

mses_test = c(mse_test_1, mse_test_2, mse_test_3, mse_test_4, mse_test_5, mse_test_6)
mses_train = c(mse_test_1_train, mse_test_2_train, mse_test_3_train, mse_test_4_train,
mse_test_5_train, mse_test_6_train)
which.min(mses_test)
which.min(mses_train)

plot((1:6), mses_test, col="red")
plot((1:6), mses_train, col="red")

# Both in one plot
plot((1:6), mses_test, col="blue", ylim=c(23,45) )
par(new=TRUE)
plot((1:6), mses_train, col="red", ylim=c(23,45)  )
```

```
#4
sub_of_tecator = subset(tecator, select = - c(Protein, Moisture, Sample))

n=dim(sub_of_tecator)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
t_train_sub=sub_of_tecator[id,]
t_test_sub=sub_of_tecator[-id,]

library(MASS)
linear_model_fat = glm(Fat~., data=sub_of_tecator)
step = stepAIC(linear_model_fat, direction = "both")

step$anova
summary(step)

covariates = scale(subset(sub_of_tecator, select = -Fat))
response = scale(subset(sub_of_tecator, select = Fat))

#-------5---------
ridge = glmnet(as.matrix(covariates), response, alpha=0, family= "gaussian" )
plot(ridge, xvar="lambda", label=TRUE)

#-------6---------
lasso = glmnet(as.matrix(covariates), response, alpha=1, family= "gaussian")
plot(lasso, xvar = "lambda", label=TRUE) #higher lambda, lower vairance, higher variance.

#-------7---------
lasso_mse = cv.glmnet(covariates, response, gamma=1, lambda = seq(from = 0, to=1,
by=0.01))
plot(lasso_mse)
```