Oskar Hidén oskhi827
2019-11-24

Assignment 1

Step 2

Test data. Threshold = 0.5		**Prediction**	
		0	1
Actual	0	808	143
	1	92	327

Misclassification rate = 0.1715

Training data. Threshold = 0.5		**Prediction**	
		0	1
Actual	0	804	127
	1	93	346

Misclassification rate = 0.1606

The misclassification rate on training data is lower. Which make sense because the model is created from this data.

Step 3

Test data. Threshold = 0.8		**Prediction**	
		0	1
Actual	0	913	20
	1	314	105

Misclassification rate = 0.2438

Training data. Threshold = 0.8		**Prediction**	
		0	1
Actual	0	921	10
	1	333	106

Oskar Hidén oskhi827
2019-11-24

Misclassification rate = 0.2504

The new rule made the model classify more emails as not spam. Which made the misclassification rate increase.

Step 4

kknn with K=30
Misclassification rate for test= 0.3131
Misclassification rate for training= 0.1672

Results from step two had a lower misclassification rate for both test and training data. But misclassification rate for training were quite close.

Step 5

kknn with K=1
Misclassification rate for test = 0.3591
Misclassification rate for training = 0

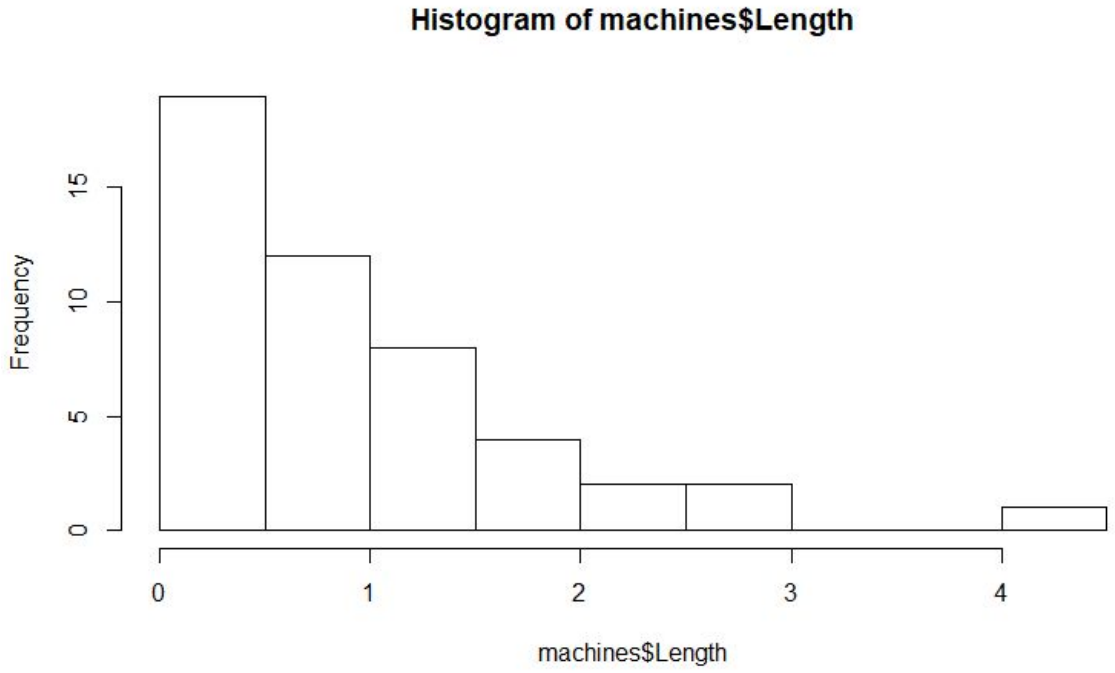
Misclassification rate for test data is increased. When we decrease K we only look at the number one nearest neighbor. That gives us a bigger error for test data. However the misclassification for training data is zero, since it is comparing with itself.

Oskar Hidén oskhi827
2019-11-24

Assignment 2

Step 2

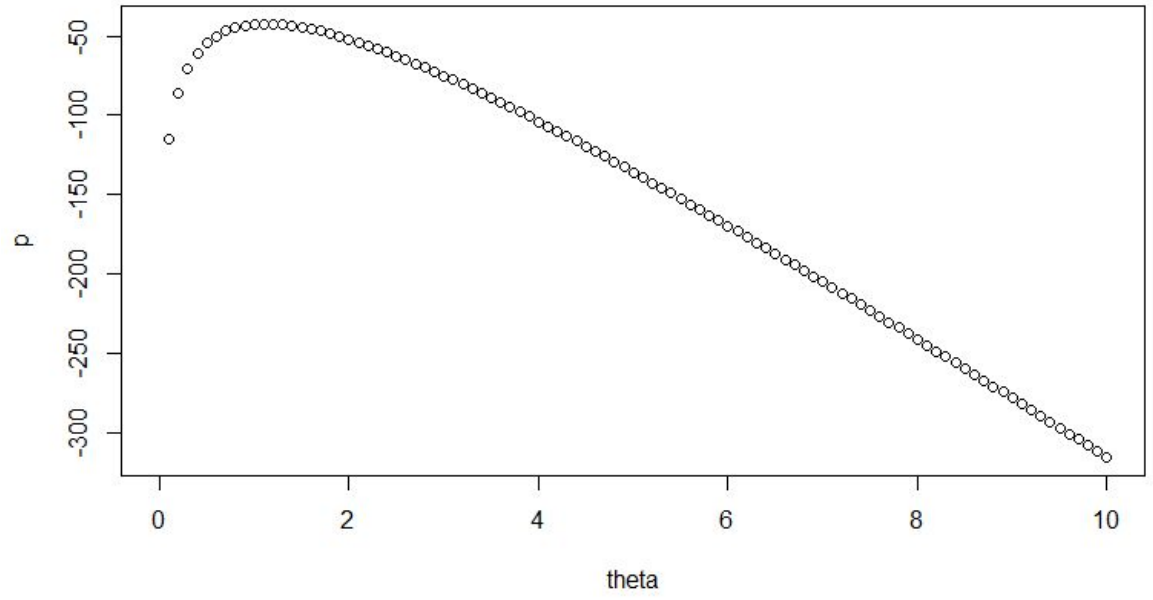
The lifetime of machines follows an Exponential distribution. We can see that by looking at following histogram and formula:



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

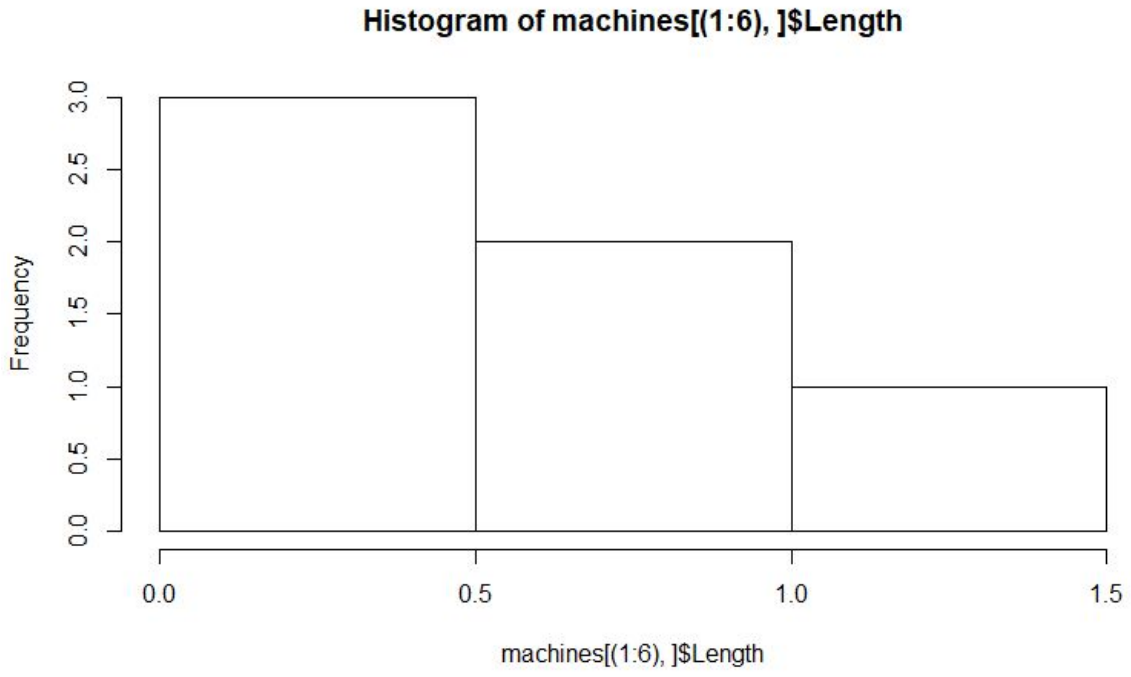
Oskar Hidén oskhi827
2019-11-24

Plot of curve showing the dependence log-likelihood on theta with entire dataset:



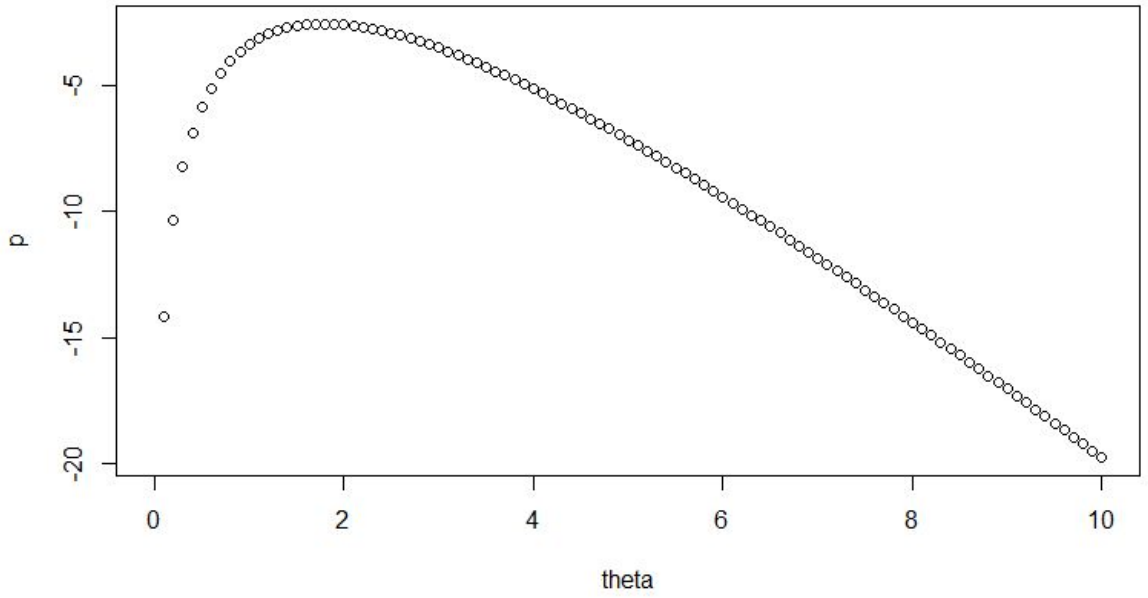
Maximum-likelihood with theta = 1.1

Step 3



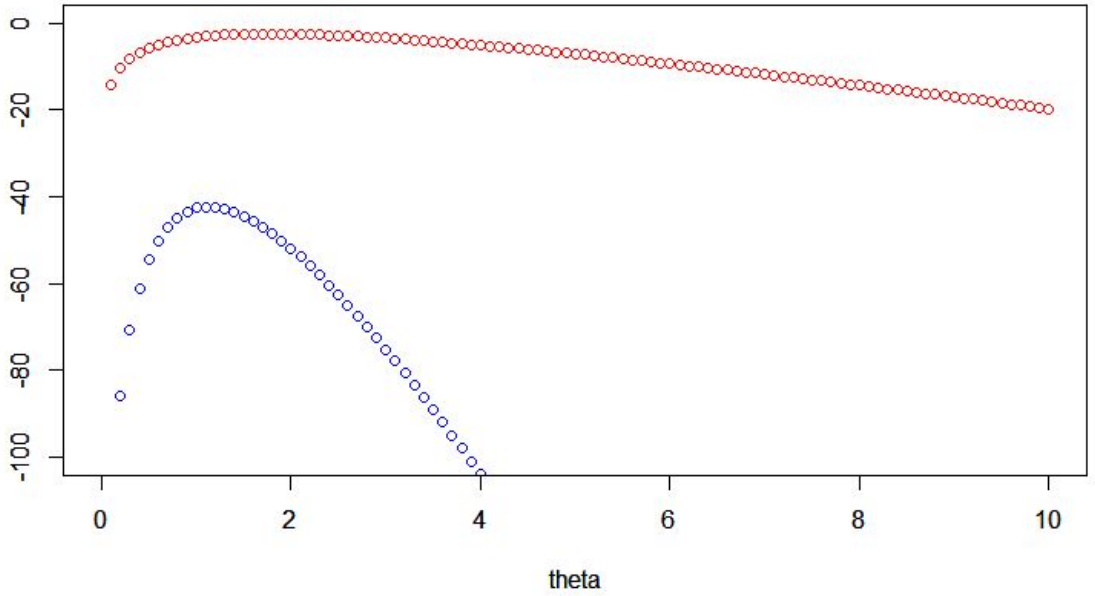
Oskar Hidén oskhi827
2019-11-24

Plot of curve showing the dependence log-likelihood on theta with six first rows as dataset:



Maximum-likelihood with theta = 1.8

Plot with both log-likelihood curves:



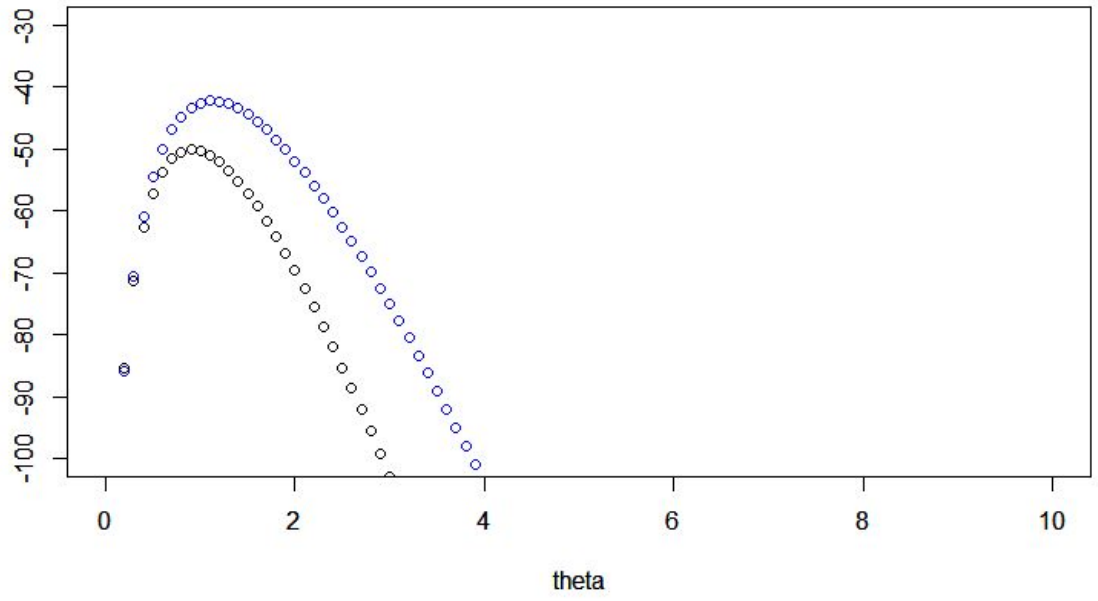
The reliability of the maximum likelihood in each case is different. The log-likelihood for first six has a higher log-likelihood. Which is due to fewer values in its dataset, which is used to calculate the log-likelihood. But the blue curve, which has a lot more observation is probably a better estimation of our model.

Oskar Hidén oskhi827
2019-11-24

Step 4

The measure computed by this function is: Given x which theta has the highest likelihood.

Plot showing *L(theta) of theta* as black and *log-likelihood of theta*(from step2) as blue:



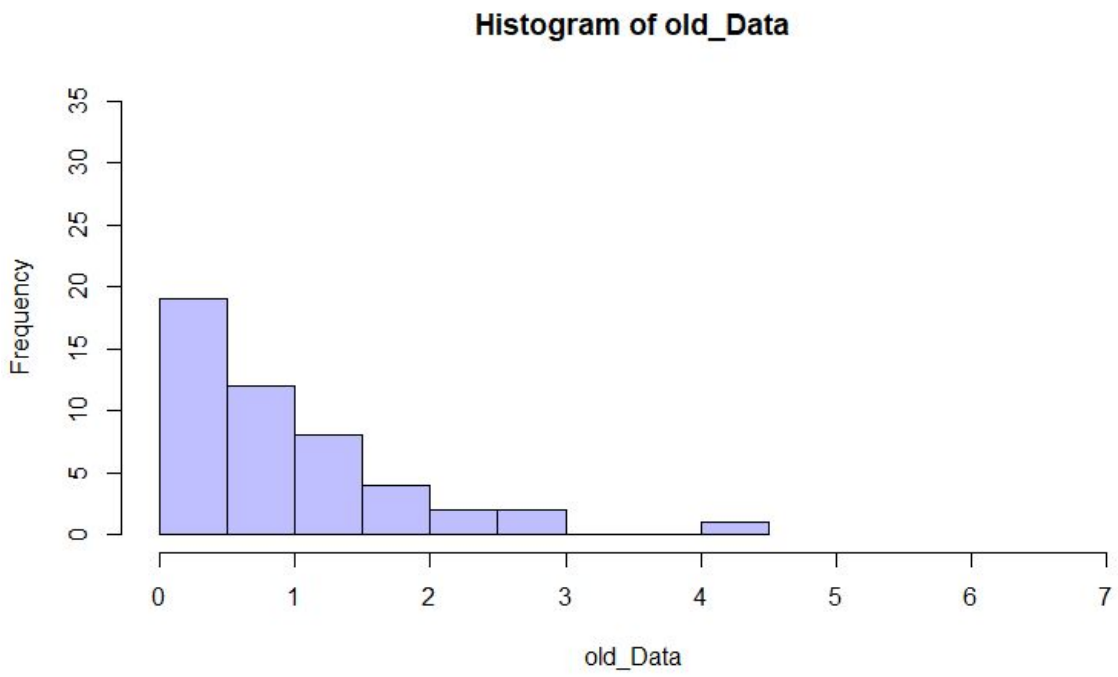
Optimal theta = 0.9

The optimal theta using this method is slightly lower than previous step, where theta = 1.1. The plots look similar and both the theta and log-likelihood is around the same size.

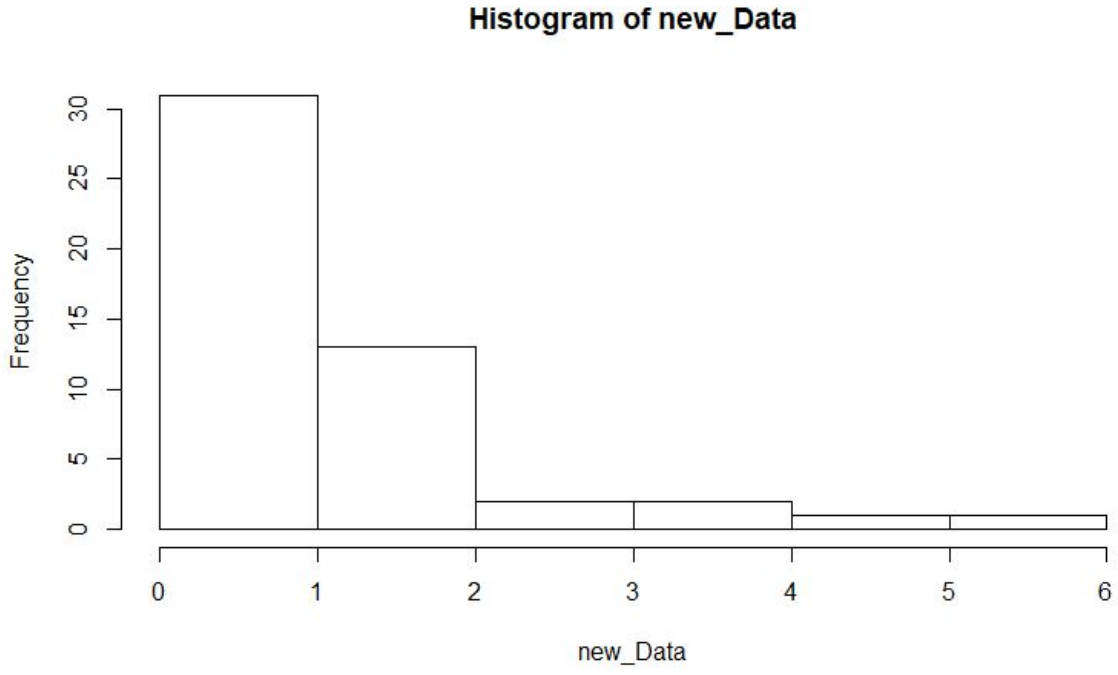
Oskar Hidén oskhi827
2019-11-24

Step 5

Histogram of old data:



Histogram of new data:



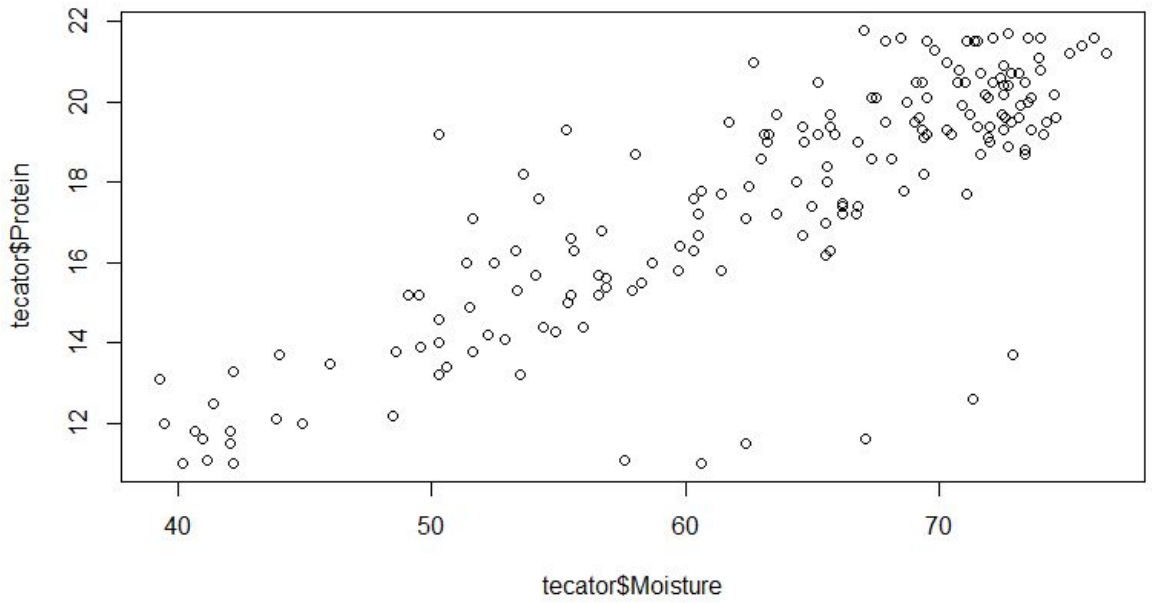
New and old data seems to follow the same exponential distribution. That makes sense, because the new data is generated with the lambda that best describes the distribution of the old data.

Oskar Hidén oskhi827
2019-11-24

Assignment 4

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:

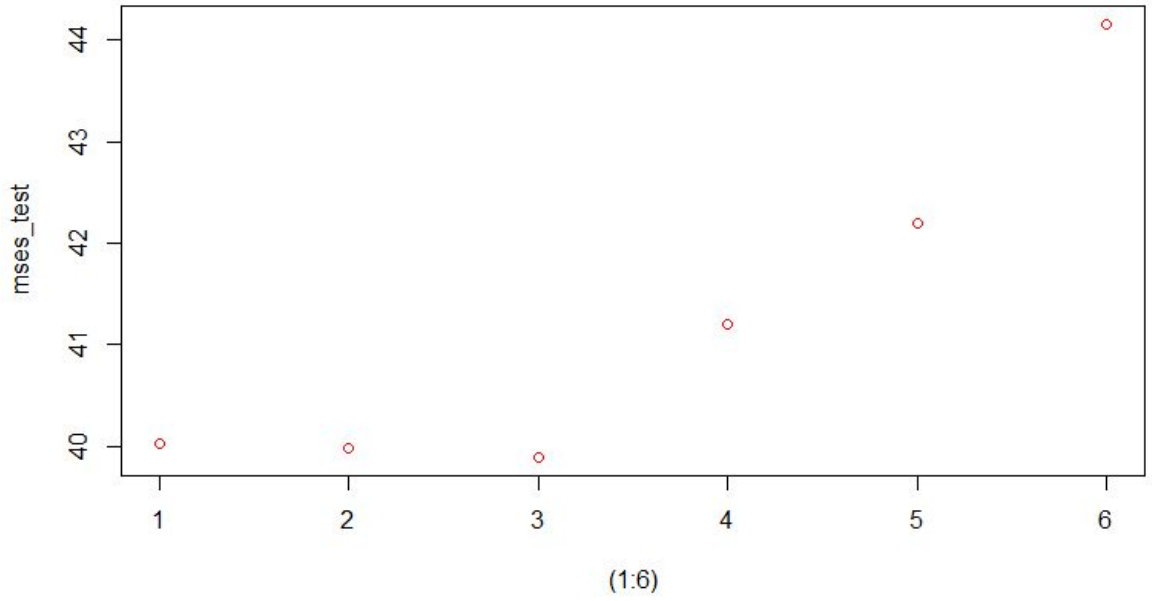
$$M_i = \sum_{k=1}^{i} w_k * (Protein)^k + \varepsilon$$

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.

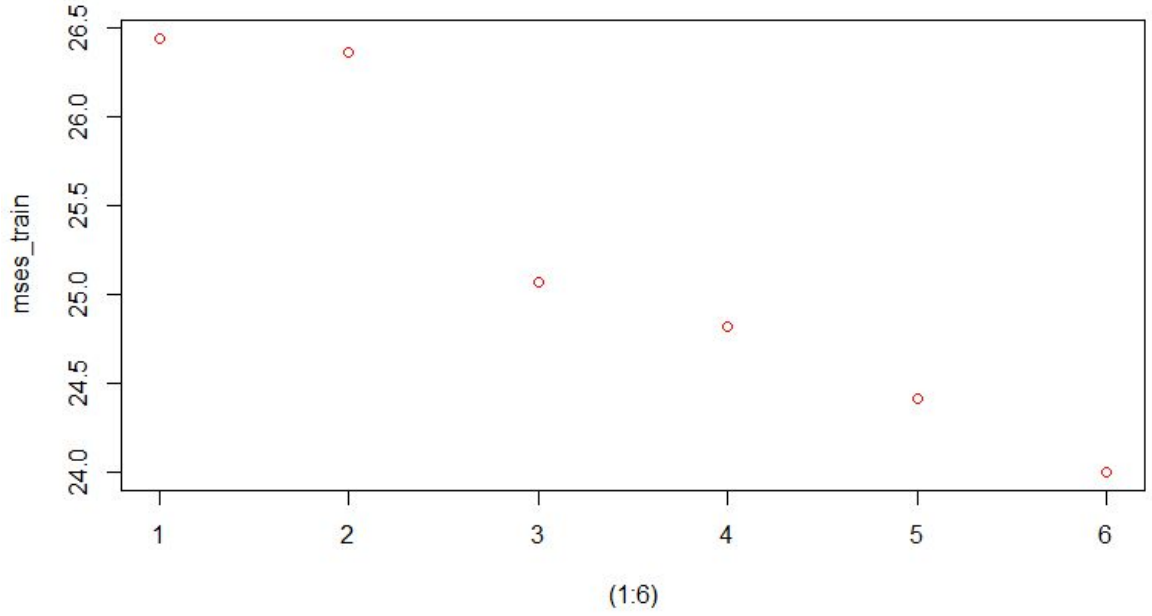
Oskar Hidén oskhi827
2019-11-24

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.

Oskar Hidén oskhi827
2019-11-24

Step 4

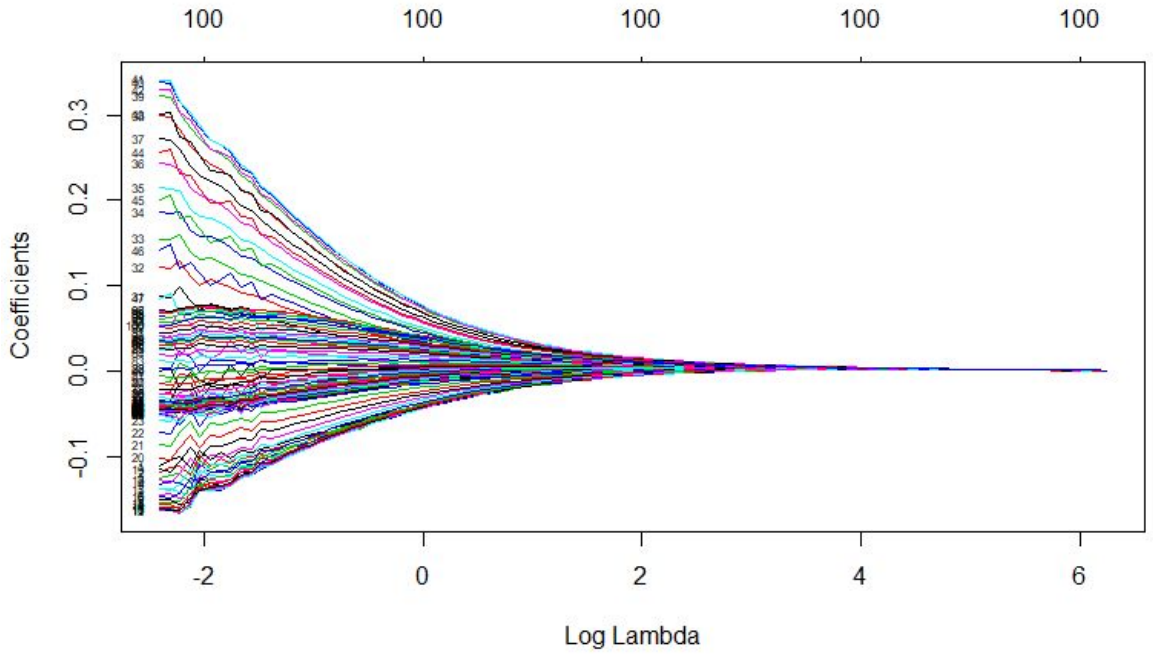
```
Final Model:
Fat ~ Channel1 + Channel2 + Channel4 + Channel5 + Channel7 +
    Channel8 + Channel11 + Channel12 + Channel13 + Channel14 +
    Channel15 + Channel17 + Channel19 + Channel20 + Channel22 +
    Channel24 + Channel25 + Channel26 + Channel28 + Channel29 +
    Channel30 + Channel32 + Channel34 + 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.

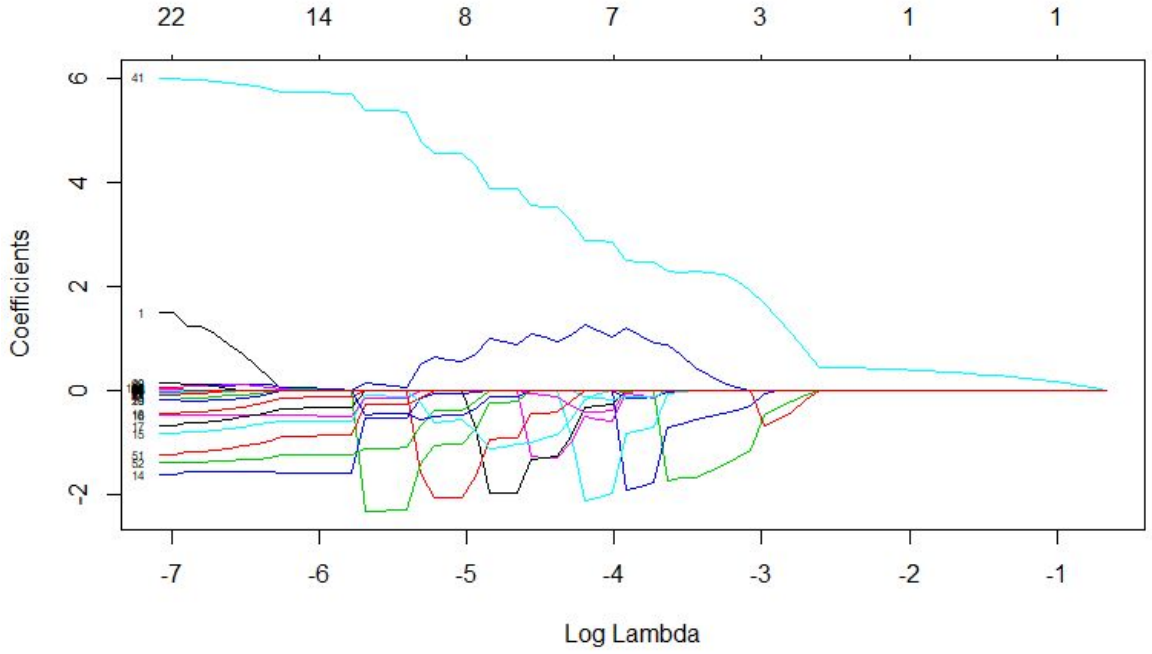
Plot of coefficients dependencies for ridge-regression:



Oskar Hidén oskhi827
2019-11-24

Step 6

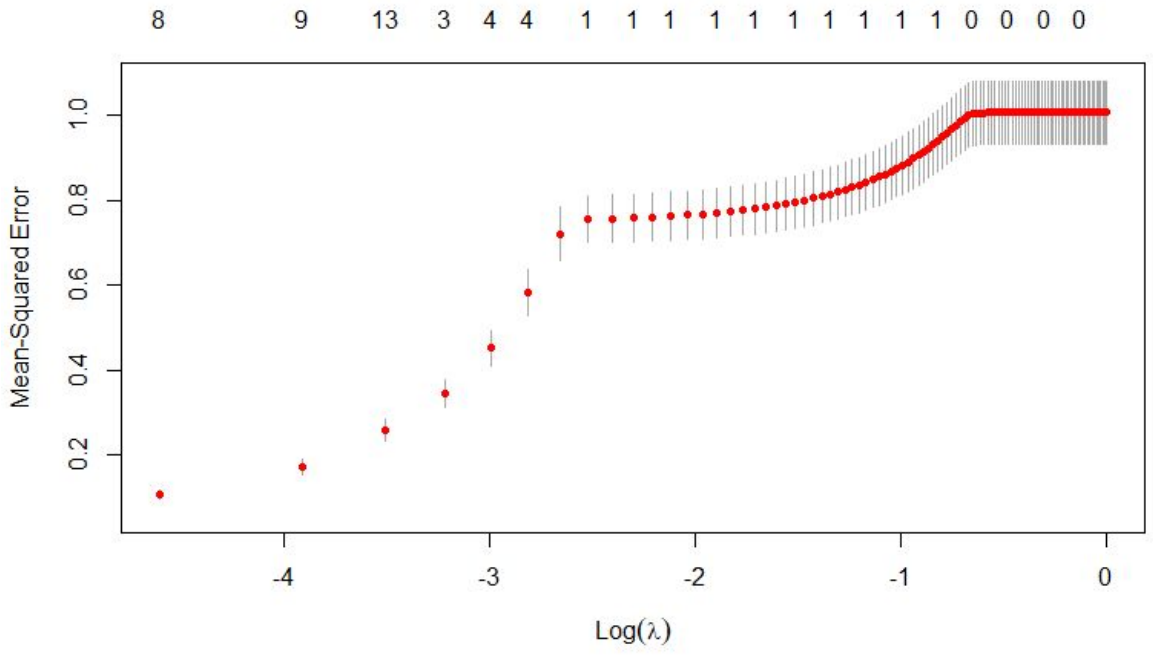
Plot of coefficients dependencies for lasso-regression:



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.

Oskar Hidén oskhi827
2019-11-24

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.

Oskar Hidén oskhi827
2019-11-24

Appendix

Assignment 1

```
library(readr)
library(glmnet)
data <- read_csv2("C:/Users/oskar/OneDrive/Universitet/Linköping Universitet/År4/Machine learning/Lab 1/spambase.csv")

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

predModel = glm(Spam~., data=train, family = binomial)

missclass=function(X, Xfit){
  n=length(X)
  print(table(X, Xfit))
  return (1-sum(diag(table(X, Xfit)))/n)
}

#Pred 1
prediction_test = predict(predModel, test, type="response")
prediction2_test = prediction_test
prediction_test[prediction_test<=0.5]=0
prediction_test[prediction_test>0.5]=1
plot(prediction_test)
missClassResult_test = missclass(test[[49]], prediction_test)
print(missClassResult_test)

prediction_train = predict(predModel, train, type="response")
prediction2_train = prediction_train
prediction_train[prediction_train<=0.5]=0
prediction_train[prediction_train>0.5]=1
plot(prediction_train)
missClassResult_train = missclass(train[[49]], prediction_train)
print(missClassResult_train)

#Pred 2
prediction2_train[prediction2_train<=0.8]=0
prediction2_train[prediction2_train>0.8]=1
```

```
prediction2_test[prediction2_test<=0.8]=0
prediction2_test[prediction2_test>0.8]=1
#plot(prediction2)

missClassResult_8_train = missclass(train[[49]], prediction2_train)
missClassResult_8_test = missclass(test[[49]], prediction2_test)
print(missClassResult_8_train)
print(missClassResult_8_test) #New rule makes more missclassifications.


#kknn
library(kknn)

knnn= kknn(as.factor(Spam)~., train, test, k=30)
missClassResultk30 = missclass(test[[49]], knnn$fitted.values)
knnn= kknn(as.factor(Spam)~., train, train, k=30)
missClassResultk30_train = missclass(train[[49]], knnn$fitted.values)

knnn1= kknn(as.factor(Spam)~., train, test, k=1)
missClassResultk1 = missclass(test[[49]], knnn1$fitted.values)
knnn1= kknn(as.factor(Spam)~., train, train, k=1)
missClassResultk1_train = missclass(train[[49]], knnn1$fitted.values)

print(missClassResultk30)
print(missClassResultk30_train)

print(missClassResultk1) #not as good
print(missClassResultk1_train)
```

Assignment 2

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

hist(machines$Length)#exponential

#log-likelihood
log_likelihood = function(theta, data){
  p=0
  data = data$Length
  for(i in data){
    p = p + log( theta*exp(-theta*i))
  }
  return (p)
```

```
}

#find my Theta
findTheta = function(data){
  p = 1:100
  i=1
  for(theta in seq(from=0, to=10, by=0.1)){
    p[i]= log_likelihood(theta, data)
    i = i+1
  }
  theta = seq(from=0, to=10, by=0.1)
  plot(theta, p)
  return(p)
}

logPTotal = findTheta(machines)
logPSix = findTheta(machines[(1:6), ])

thetaTotal = seq(from=0, to=10, by=0.1)[which.max(logPTotal)]
thetaSixFirst = seq(from=0, to=10, by=0.1)[which.max(logPSix)]

print(thetaTotal)
print(thetaSixFirst)
theta = seq(from=0, to=10, by=0.1)
plot(theta, logPTotal, col="blue", ylim=c(-100,0))
par(new=TRUE)
plot(theta, logPSix, col="red", ylim=c(-100,0))

findThetaBayesian = function(lambda, data){
  p = 1:100
  i=1
  for(theta in seq(from=0, to=10, by=0.1)){
    log_like_lambda = log(lambda)+(-theta*lambda)
    p[i]= log_like_lambda + log_likelihood(theta, data)
    i = i+1
  }
  theta = seq(from=0, to=10, by=0.1)
  plot(theta, p)
  return(p)
}

theta = seq(from=0, to=10, by=0.1)
LogsThetaBay = findThetaBayesian(10, machines)
thetaMaxBay = seq(from=0, to=10, by=0.1)[which.max(LogsThetaBay)]

plot(theta, logPTotal, col="blue", ylim=c(-100,-30))
```

```
par(new=TRUE)
plot(theta, LogsThetaBay, ylim=c(-100,-30))
print(thetaMaxBay)


#5
set.seed(12345)
new_Data = rexp(50, rate=thetaTotal)
old_Data = machines$Length

#plot hist in one diagram
p1 <- hist(old_Data)
p2 <- hist(new_Data)
plot( p1, col=rgb(0,0,1,1/4), xlim=c(0,7), ylim=c(0,35))  # first histogram
plot( p2, col=rgb(1,0,0,1/4), xlim=c(0,7),ylim=c(0,35), add=T) #second histogram
```

Assignment 4

```
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)
```

Oskar Hidén oskhi827
2019-11-24

```
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.
```

Oskar Hidén oskhi827
2019-11-24

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