# TDDE01 Machine Learning Lab 1 Group C05

Ashin Mathew (ashma352)

Abdul Habib (abdha804)

## Assignment 1

```
#1.1 Import data and divide it into train and test sets
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,]

#1.2 Logistic regression with threshold 0.5
model <- glm(Spam ~ ., family = "binomial", data = train)

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

predtrain <- predict(model,newdata=train,type ="response")
predtest <- predict(model,newdata = test, type = "response")

cm5train = table(train$Spam, predtrain > 0.5)
cm5test = table(test$Spam, predtest > 0.5)
mc5train = misclassification(cm5train)
mc5test = misclassification(cm5test)

cm5train

##
##     FALSE TRUE
##   0   803  142
##   1    81  344

cm5test

##
##     FALSE TRUE
##   0   791  146
##   1    97  336

print(paste("Misclassification Rate Training Data :", mc5train))

## [1] "Misclassification Rate Training Data : 0.162773722627737"

print(paste("Misclassification Rate Test Data :", mc5test))

## [1] "Misclassification Rate Test Data : 0.177372262773723"

#1.3 Logistic regression with threshold 0.8
cm8train = table(train$Spam, predtrain > 0.8)
```

```
cm8test = table(test$Spam, predtest > 0.8)
mc8train = misclassification(cm8train)
mc8test = misclassification(cm8test)

cm8train
```

```
##
##     FALSE TRUE
##   0   941    4
##   1   335   90
```

```
cm8test
```

```
##
##     FALSE TRUE
##   0   926   11
##   1   367   66
```

```
print(paste("Misclassification Rate Training Data :", mc8train))
```

```
## [1] "Misclassification Rate Training Data : 0.247445255474453"
```

```
print(paste("Misclassification Rate Test Data :", mc8test))
```

```
## [1] "Misclassification Rate Test Data : 0.275912408759124"
```

The misclassification rate is higher when 0.8 threshold is used. Number of emails classified as spam is less than the first classification which used a 0.5 threshold.

```
#1.4 KKNN with K = 30
k30train = kknn(as.factor(Spam)~., train, train, k = 30)
k30test = kknn(as.factor(Spam)~., train, test, k = 30)
cmk30train = table(train$Spam, k30train$fit)
cmk30test = table(train$Spam, k30test$fit)
mck30train = misclassification(cmk30train)
mck30test = misclassification(cmk30test)

cmk30train
```

```
##
##       0   1
##   0 807 138
##   1  98 327
```

```
cmk30test
```

```
##
##       0   1
##   0 594 351
##   1 265 160
```

```
print(paste("Misclassification Rate Training Data :", mck30train))
```

```
## [1] "Misclassification Rate Training Data : 0.172262773722628"
```

```
print(paste("Misclassification Rate Test Data :", mck30test))
```

```
## [1] "Misclassification Rate Test Data : 0.44963503649635"
```

The misclassification rate is higher than the logistic regression in the step 2

```
#1.5 KKNN with K = 1
k1train = kknn(as.factor(Spam)~., train, train, k = 1)
k1test = kknn(as.factor(Spam)~., train, test, k = 1)
cmk1train = table(train$Spam, k1train$fit)
cmk1test = table(train$Spam, k1test$fit)
mck1train = misclassification(cmk1train)
mck1test = misclassification(cmk1test)
```

```
cmk1train
```

```
##
##       0   1
##   0 945   0
##   1   0 425
```

```
cmk1test
```

```
##
##       0   1
##   0 559 386
##   1 258 167
```

```
print(paste("Misclassification Rate Training Data :", mck1train))
```

```
## [1] "Misclassification Rate Training Data : 0"
```

```
print(paste("Misclassification Rate Test Data :", mck1test))
```

```
## [1] "Misclassification Rate Test Data : 0.47007299270073"
```

The classifier has a misclassification rate of 0 on training data but the misclassification rate is higher. This is because every data point in the training data has only itself as nearest neighbor.

## Assignment 2

#2.1

```
library(readxl)

machines <- data.matrix(read_excel("/home/habib/Li.U/TDDE01 ML/labs/lab11/machines.xlsx"))
```

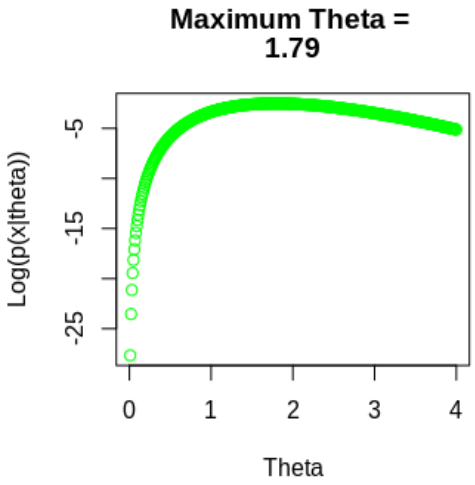
1. The example of life time machine where variable x is the length of lifetime machine and distributed exponentially.
2. The probability is computed with following model.

$$p(x|\theta) = \theta e^{-\theta x} \text{ for } \boldsymbol{x}\text{=}Length$$

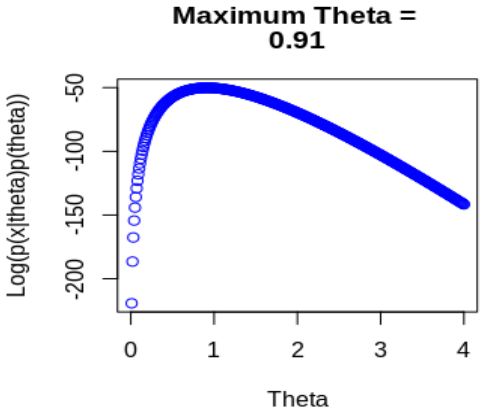
The curve shows the dependency over theta.

Entire data is used fitted and the sequence of theta is from 0 to 4 with 0.01 step size.

3. this curve is the shows the first six observation from theta.

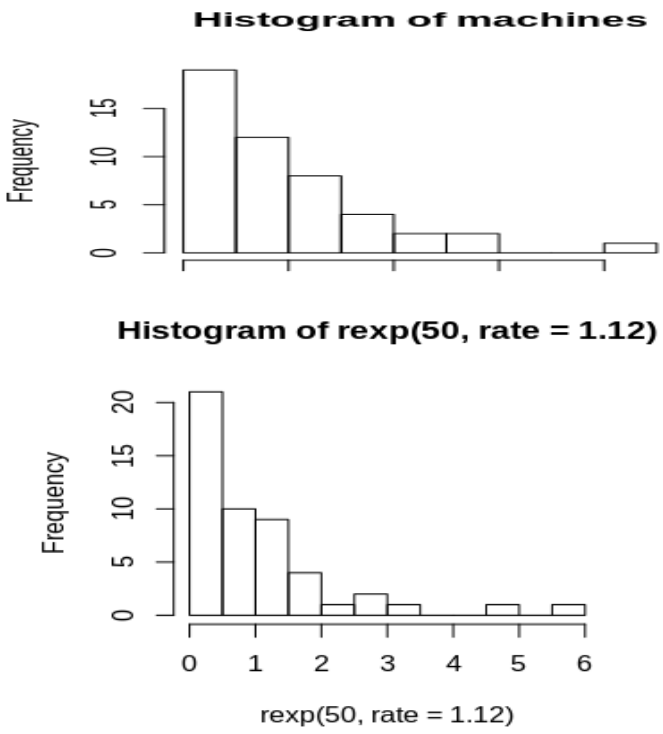



4. Bayesian probability plot, theta value is close to the theta got from log likelihood.



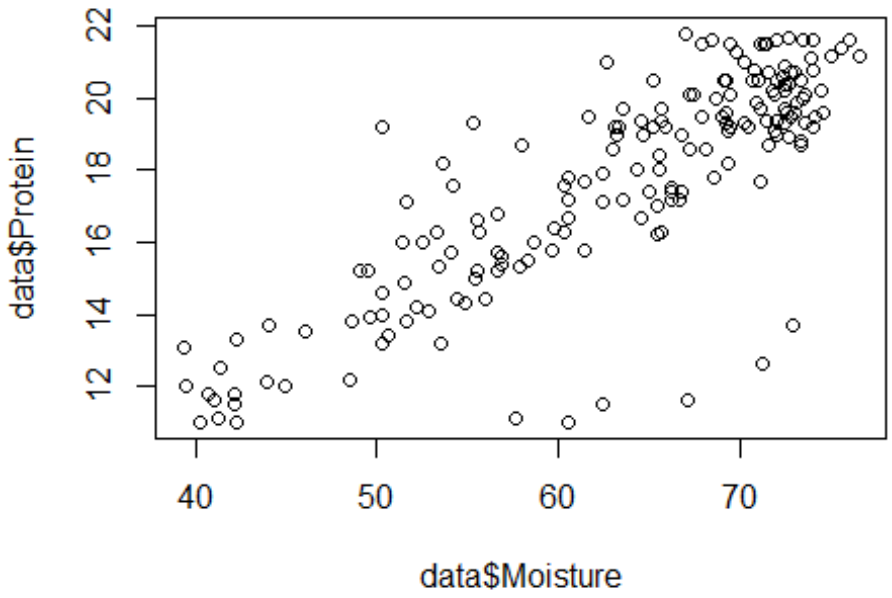
5. Histogram for the given data

Histogram generated for the data sets with theta 1.12




## Assignment 4

```
#4.1

data <- read_excel("tecator.xlsx")

plot(x=data$Moisture,y=data$Protein, type="p")
```



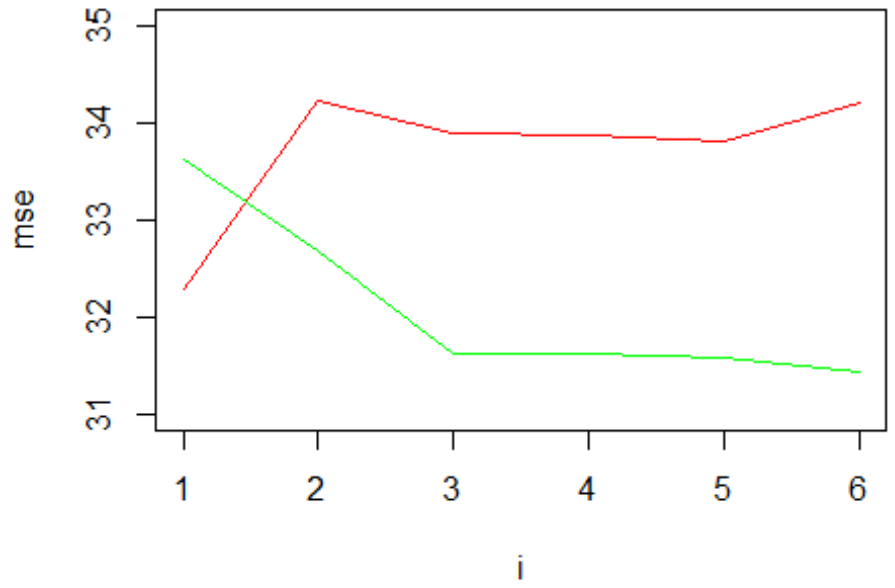

The data cam described well by a linear model

4.2

The model with smallest MSE fits the data better. We calculate the MSE of each model M1, M2,….,M6 to find which model has least MSE.

```
#4.3

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

m1=lm(Moisture~poly(Protein,1), data = train)
m2=lm(Moisture~poly(Protein,2), data = train)
m3=lm(Moisture~poly(Protein,3), data = train)
m4=lm(Moisture~poly(Protein,4), data = train)
m5=lm(Moisture~poly(Protein,5), data = train)
m6=lm(Moisture~poly(Protein,6), data = train)

m1pred = predict(m1, newdata = train)
m2pred = predict(m2, newdata = train)
m3pred = predict(m3, newdata = train)
m4pred = predict(m4, newdata = train)
m5pred = predict(m5, newdata = train)
m6pred = predict(m6, newdata = train)
```

```r
m1val = predict(m1, newdata = test)
m2val = predict(m2, newdata = test)
m3val = predict(m3, newdata = test)
m4val = predict(m4, newdata = test)
m5val = predict(m5, newdata = test)
m6val = predict(m6, newdata = test)

msetrain = numeric(6)
msetrain[1] = mse(m1pred,train$Moisture)
msetrain[2] = mse(m2pred,train$Moisture)
msetrain[3] = mse(m3pred,train$Moisture)
msetrain[4] = mse(m4pred,train$Moisture)
msetrain[5] = mse(m5pred,train$Moisture)
msetrain[6] = mse(m6pred,train$Moisture)

msetest = numeric(6)
msetest[1] = mse(m1val,test$Moisture)
msetest[2] = mse(m2val,test$Moisture)
msetest[3] = mse(m3val,test$Moisture)
msetest[4] = mse(m4val,test$Moisture)
msetest[5] = mse(m5val,test$Moisture)
msetest[6] = mse(m6val,test$Moisture)

plot(1:6, msetest, type="l",col="red", xlab="i", ylab="mse", ylim=c(31,35))
points(1:6, msetrain, type = "l", col="green")
```

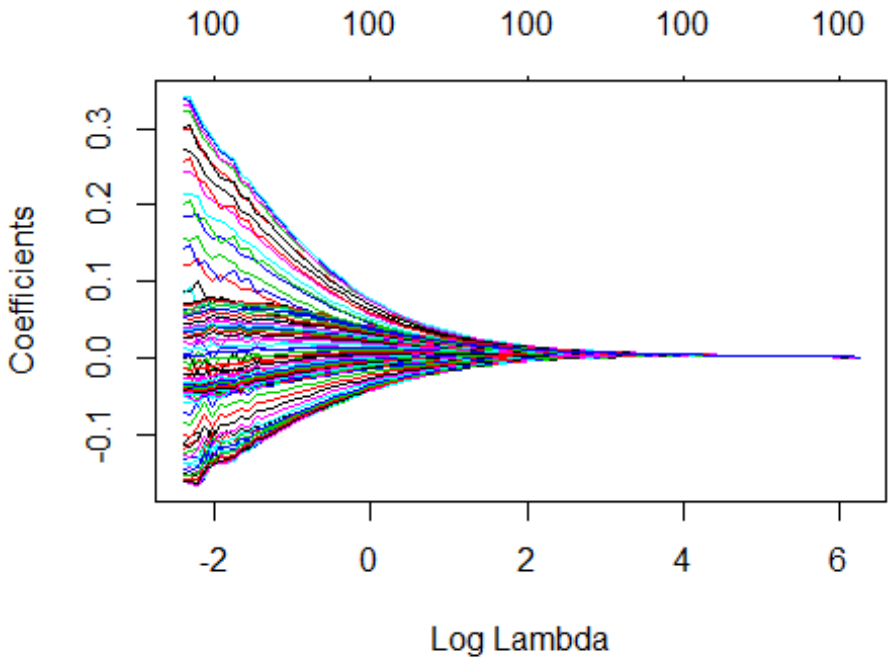


The plot shows MSE of both training data (green) and test data (red) for different regression models $M_i$ Where i={1,2,3,4,5,6}. For test data MSE is lowest when i = 1. For training data MSE decreases as i increases.

```
channels = data[,2:101]
model = lm(data$Fat~., data=channels)
saic = stepAIC(model, trace = 0)
```

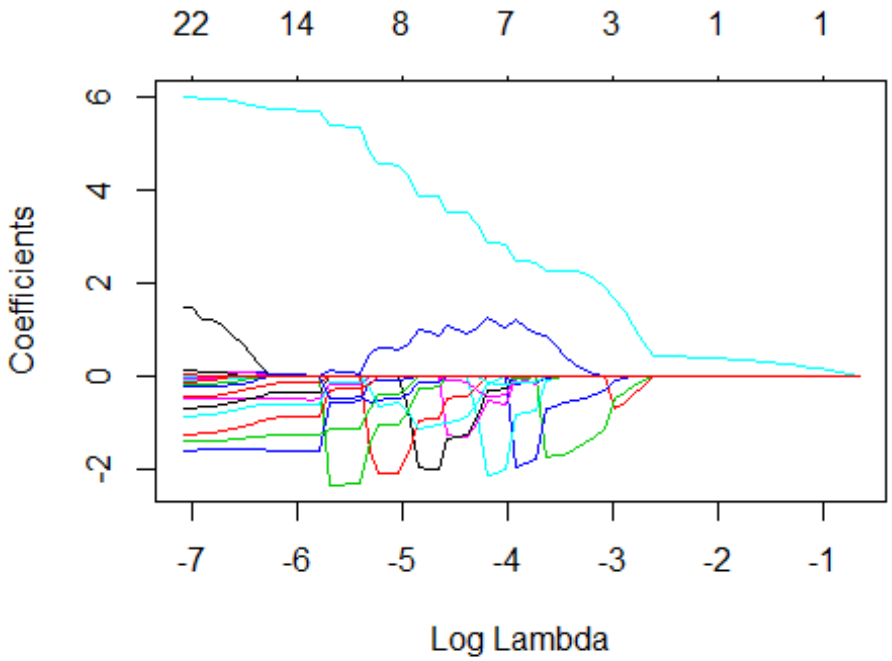
```
4.4 64 variables were selected
```

```
#4.5

schannels = scale(data[,2:101])
ridge = glmnet(x=data.matrix(schannels), y=scale(data$Fat), alpha = 0)
plot.glmnet(ridge, xvar = "lambda")
```



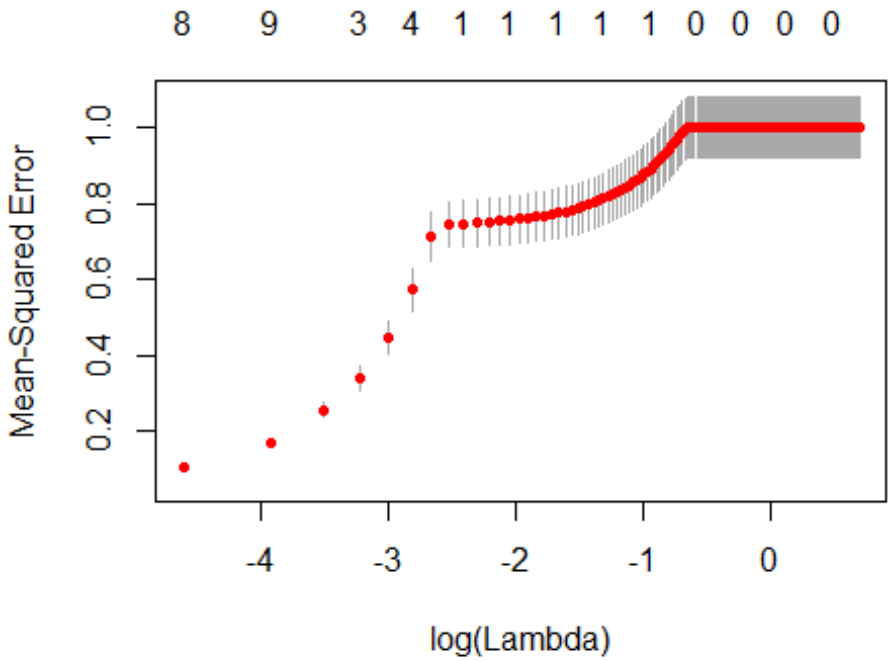

Each line in the plot shows a coefficient in the ride regression. We can observe that $\lambda$ tends to infinity while coefficients tends to 0.

```
#4.6

lasso = glmnet(x=data.matrix(schannels), y=scale(data$Fat), alpha = 1)
plot.glmnet(lasso, xvar = "lambda")
```




The plot shows that the coefficients of lasso regressions depends on the log of $\lambda$ . Comparing to step 5 both $\lambda$ and coefficients exhibits same trend but the coefficients tend to 0 faster in lasso.

```
#4.7

cvlasso = cv.glmnet(x=as.matrix(schannels), y=scale(data$Fat), alpha=1,lambda
= seq(0,2,0.01))
cvlasso$lambda.min

## [1] 0

plot(cvlasso)
```




The plot show the CV score decreases with $\lambda$. To minimize CV score $\lambda$ should be as low as possible. So $\lambda$ = 0 is the optimal value.

4.8

The stepAIC in 4.4 chooses less variables than the cross validation in assignment 4.7

**Code for Assignment 1**

```
setwd("D:/MScCS/SEM 3/TDDE01/Labs")
require(readxl)
require(kknn)

misclassification <- function(tab){
  misscal <- 1-sum(diag(tab))/sum(tab)
}

#1.1
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,]

#1.2
model <- glm(Spam ~ ., family = "binomial", data = train)
predtrain <- predict(model,newdata=train,type ="response")
predtest <- predict(model,newdata = test, type = "response")

cm5train = table(train$Spam, predtrain > 0.5)
cm5test = table(test$Spam, predtest > 0.5)
mc5train = misclassification(cm5train)
mc5test = misclassification(cm5test)

cm5train
cm5test
print(paste("Misclassification Rate Training Data :", mc5train))
print(paste("Misclassification Rate Test Data :", mc5test))
```

```
#1.3
cm8train = table(train$Spam, predtrain > 0.8)
cm8test = table(test$Spam, predtest > 0.8)
mc8train = misclassification(cm8train)
mc8test = misclassification(cm8test)

cm8train
cm8test
print(paste("Misclassification Rate Training Data :", mc8train))
print(paste("Misclassification Rate Test Data :", mc8test))

#1.4
k30train = kknn(as.factor(Spam)~., train, train, k = 30)
k30test = kknn(as.factor(Spam)~., train, test, k = 30)
cmk30train = table(train$Spam, k30train$fit)
cmk30test = table(train$Spam, k30test$fit)
mck30train = misclassification(cmk30train)
mck30test = misclassification(cmk30test)

cmk30train
cmk30test
print(paste("Misclassification Rate Training Data :", mck30train))
print(paste("Misclassification Rate Test Data :", mck30test))

#1.5
k1train = kknn(as.factor(Spam)~., train, train, k = 1)
k1test = kknn(as.factor(Spam)~., train, test, k = 1)
cmk1train = table(train$Spam, k1train$fit)
cmk1test = table(train$Spam, k1test$fit)
mck1train = misclassification(cmk1train)
```

```
mck1test = misclassification(cmk1test)

cmk1train
cmk1test
print(paste("Misclassification Rate Training Data :", mck1train))
print(paste("Misclassification Rate Test Data :", mck1test))
```

**Code for Assignment 2**

```
#2.1
library(readxl)
machines <- data.matrix(read_excel("/home/habib/Li.U/TDDE01 ML/labs/lab11/machines.xlsx"))

#2.2
log_likelihood <- function(theta, data){
  probModel = prod(theta * exp(-1 * theta * data))
  return(log(probModel))
}
theta = seq(from=0, to=4, by=0.01)
log1 = numeric(0)

for (i in 1:length(theta)) {
  log1[i] = log_likelihood(theta[i], machines)
}

maxTheta = theta[which.max(log1)]
plot(theta, log1,
     col = "red",
     xlab = "Theta", ylab = "Log(p(x|theta))",
     main = c("Maximum Theta = ", maxTheta))
```

```
#2.3 with 6 samples

theta = seq(from=0, to=4, by=0.01)
log2 = numeric(0)

for (i in 1:length(theta)) {
  log2[i] = log_likelihood(theta[i], machines[1:6])
}
maxTheta16 = theta[which.max(log2)]
plot(theta, log2,
     col = "green",
     xlab = "Theta", ylab = "Log(p(x|theta))",
     main = c("Maximum Theta = ", maxTheta16))
#2.4 baysian log likelihood
theta = seq(from=0, to=4, by=0.01)
log3 = numeric(0)

baysian_log_likelihood <-function(data, theta){
  probModel = prod(theta * exp(-1 * theta * data))
  p = 10*exp(-10 * theta)
  lo = log(probModel * p)
  return(lo)
}

for (i in 1:length(theta)) {
  log3[i] = baysian_log_likelihood(machines,theta[i] )
}
```

```
maxThetaBasyian = theta[which.max(log3)]
plot(theta, log3,
    col = "blue",
    xlab = "Theta", ylab = "Log(p(x|theta)p(theta))",
    main = c("Maximum Theta = ",maxThetaBasyian))

# 2.5 hist
hist(machines,15)
set.seed(12345)
hist(rexp(50, rate = 1.12), 15)
```

**Code for Assignment 4**

```
setwd("D:/MScCS/SEM 3/TDDE01/Labs")
require(readxl)
require(MASS)
require(glmnet)

mse = function(y,yhat){
  return(mean((yhat-y)^2))
}

data <- read_excel("tecator.xlsx")

plot(x=data$Moisture,y=data$Protein, type="p")

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

```
test=data[-id,]

m1=lm(Moisture~poly(Protein,1), data = train)
m2=lm(Moisture~poly(Protein,2), data = train)
m3=lm(Moisture~poly(Protein,3), data = train)
m4=lm(Moisture~poly(Protein,4), data = train)
m5=lm(Moisture~poly(Protein,5), data = train)
m6=lm(Moisture~poly(Protein,6), data = train)

m1pred = predict(m1, newdata = train)
m2pred = predict(m2, newdata = train)
m3pred = predict(m3, newdata = train)
m4pred = predict(m4, newdata = train)
m5pred = predict(m5, newdata = train)
m6pred = predict(m6, newdata = train)

m1val = predict(m1, newdata = test)
m2val = predict(m2, newdata = test)
m3val = predict(m3, newdata = test)
m4val = predict(m4, newdata = test)
m5val = predict(m5, newdata = test)
m6val = predict(m6, newdata = test)

msetrain = numeric(6)
msetrain[1] = mse(m1pred,train$Moisture)
msetrain[2] = mse(m2pred,train$Moisture)
msetrain[3] = mse(m3pred,train$Moisture)
msetrain[4] = mse(m4pred,train$Moisture)
msetrain[5] = mse(m5pred,train$Moisture)
msetrain[6] = mse(m6pred,train$Moisture)
```

```
msetest = numeric(6)
msetest[1] = mse(m1val,test$Moisture)
msetest[2] = mse(m2val,test$Moisture)
msetest[3] = mse(m3val,test$Moisture)
msetest[4] = mse(m4val,test$Moisture)
msetest[5] = mse(m5val,test$Moisture)
msetest[6] = mse(m6val,test$Moisture)

plot(1:6, msetest, type="l",col="red", xlab="i", ylab="mse", ylim=c(31,35))
points(1:6, msetrain, type = "l", col="green")

channels = data[,2:101]
model = lm(data$Fat~., data=channels)
saic = stepAIC(model, trace = 0)
saic$anova

schannels = scale(data[,2:101])
ridge = glmnet(x=data.matrix(schannels), y=scale(data$Fat), alpha = 0)
plot.glmnet(ridge, xvar = "lambda")

lasso = glmnet(x=data.matrix(schannels), y=scale(data$Fat), alpha = 1)
plot.glmnet(lasso, xvar = "lambda")

cvlasso = cv.glmnet(x=as.matrix(schannels), y=scale(data$Fat), alpha=1,lambda = seq(0,2,0.01))
cvlasso$lambda.min
plot(cvlasso)
coef(cvlasso, s="lambda.min")
```