Introduction to Machine Learning - Lab 2

Gustav Sternelöv

Thursday, November 05, 2015

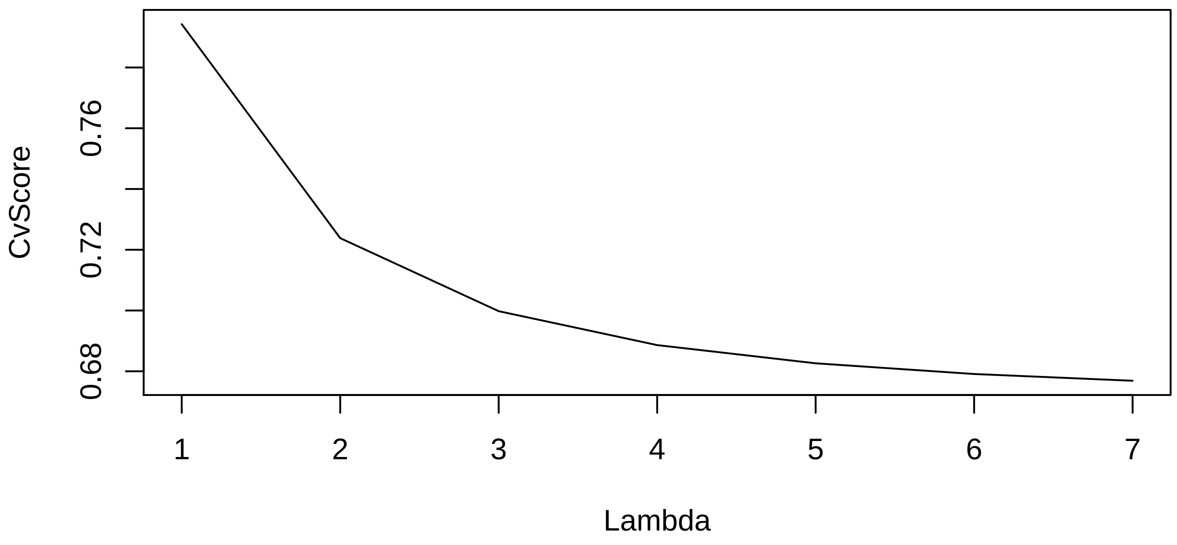
Assignment 1

1.1

The function *ridgereg_nfoldCV* which performs ridge regression by using n-fold cross validation is implemented at the first step of assignment 1. The R-code used to create the function can be seen in the appendix at the end of the report.

1.2

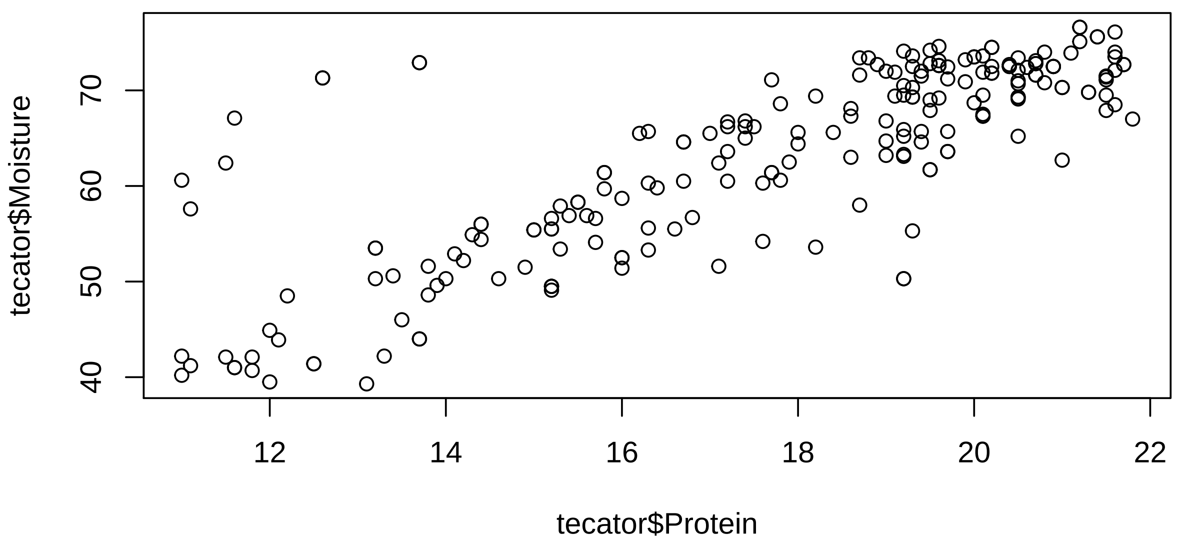
The function implemented in 1.1 is tested with the longley data set. The variables has been centered, but not scaled. Number of folds is set to 10 and the value for lambda goes from 1 to 7, by 1. The given values for the CV score is shown in the following table below and it can be seen that the CV score decreases for higher values of lambda. The conclusion from this result is that the best model is given with lambda set to 7.



Assignment 2

In the second assignment the data material called *tecator* is used. This file contains information from a study where the fat content in samples of meat was investigated. Fat is the response variable and the predicitor variables in the data, observed for every meat in the sample, are 100 channel spectrum of absorbance records and the levels of moisture and protein.

2.1



To use a linear model to describe the levels of moisture with the levels of protein may be an appopriate approach by the look of the plot.

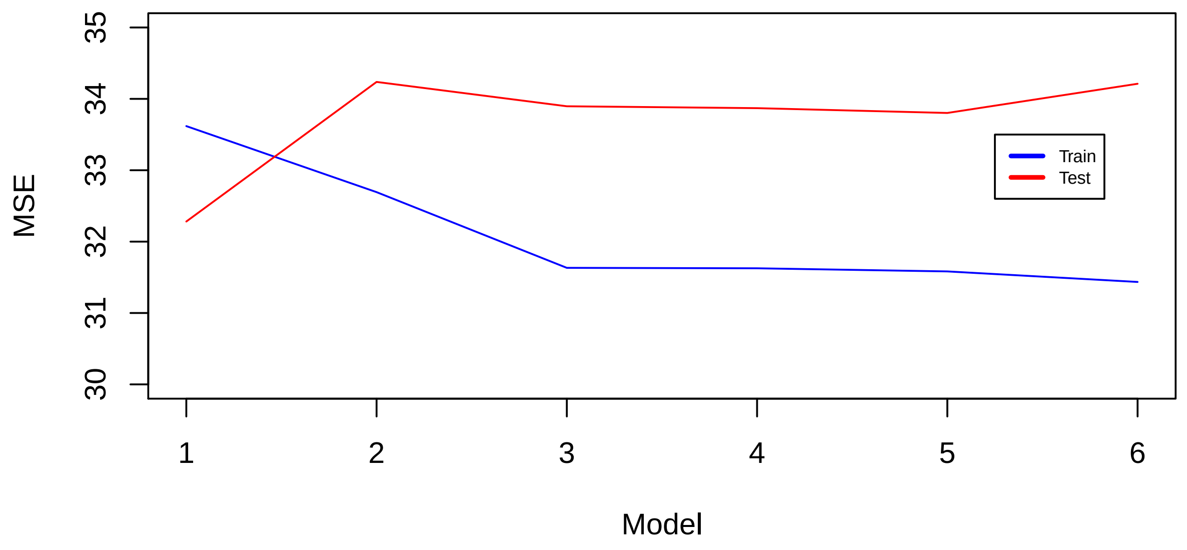
2.2

The probabilistic model for M_i where M_i is a model that has Moisture as a polynomial function of protein is shown below. The model includes polynomial terms up to power i where a model up to power 1 is equal to a linear model, up to power 2 is a quadratic model and so on.

$$Y \sim N(w_0 + \sum w_i x^i, \sigma^2)$$

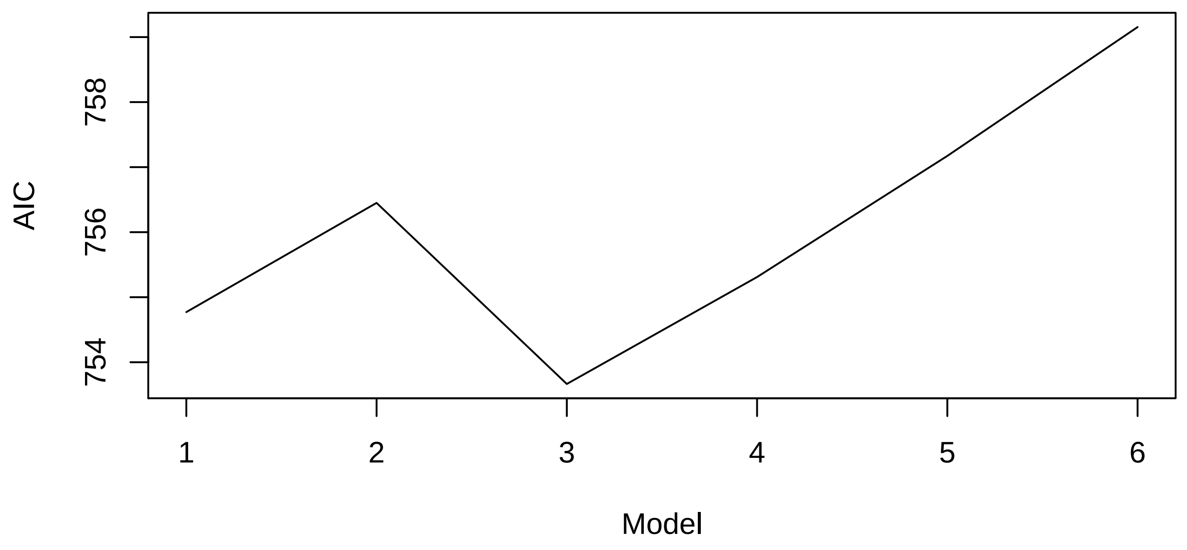
2.3

For exercise 2.3 the data set is divided into a training set and a validation set where each set contains 50 percent of the observations.
For models with polynomials of higher power the MSE for test data in general increases, and for training data it decreases. The best model for training data in terms of MSE is the, probably overfitted, model with polynomial terms up to a power of 6. This can be compared to the the test data which has the lowest MSE for a simple linear model. The best model is therefore model 1, a conclusion that suits well with the comments made about the relationship between moisture and protein in 2.1. In terms of the bias-variance trade-off an interpretation of the plot is that model 6 gives the lowest bias and highest variance and that model 1 gives the opposite result.



2.4

Now the whole data set is used to construct the same models. AIC values is computed to evaluate the models and decide which model that is the best one. The criterion used for comparing the models is that the AIC value should be as low as possible. With that criterion in mind is it concluded that the model with polynomial terms up to a power of 3 is the preferred model.

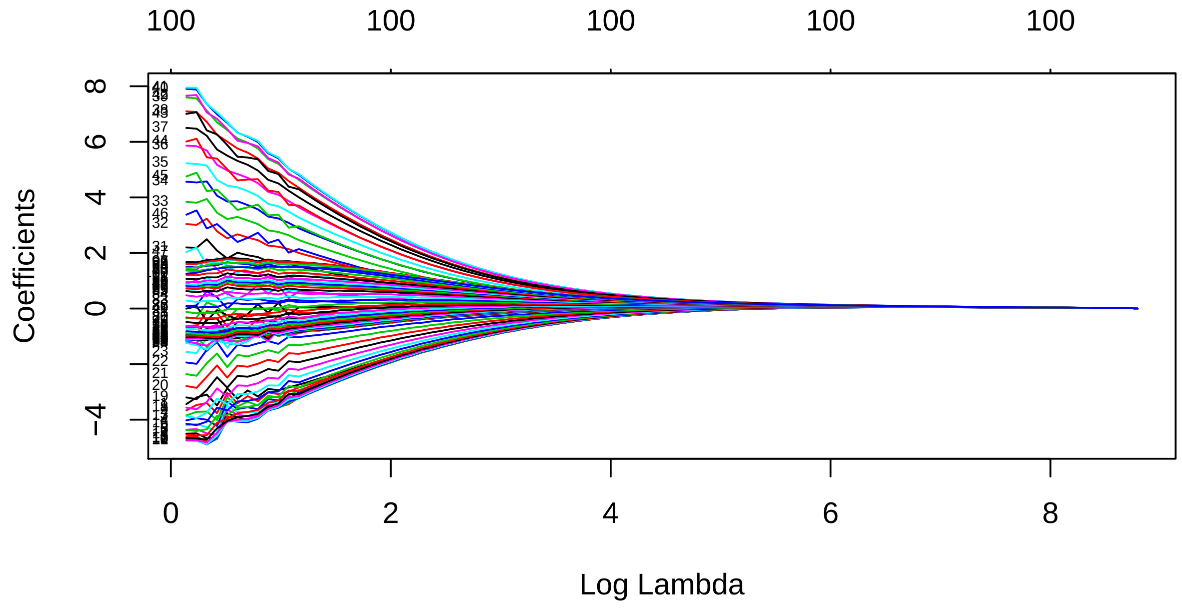


2.5

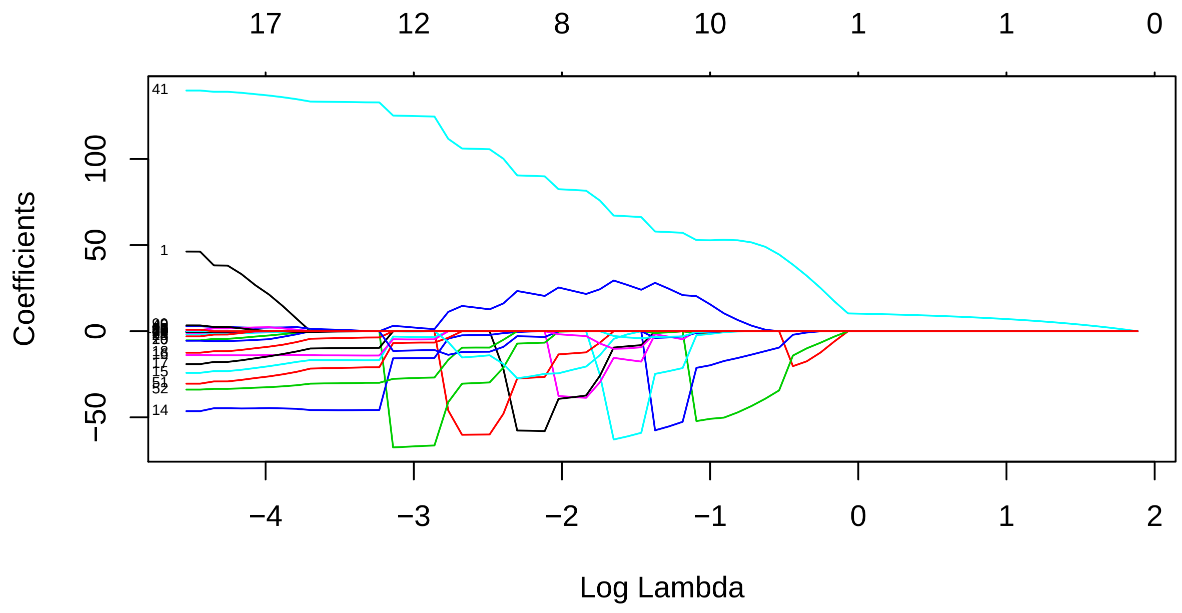
Another area where the AIC criterion can be useful is in the variable selection. Here the function *stepAIC* starts with a linear model that has *fat* as response variable and *channel1-channel100* as predictors. After running the variable selection procedure it is concluded that 63 of the predictors are selected.

2.6 - 2.7

With the same response and predictor variables as in 2.5 a ridge regression model is fitted. The difference between a ridge regression model and a linear model is the use of a penalty factor λ in the former model. The value of λ affects the coefficients in such a way that they shrinks.
How the values of the coefficients in the ridge regression model depends on the log of λ is illustrated by plotting these values.



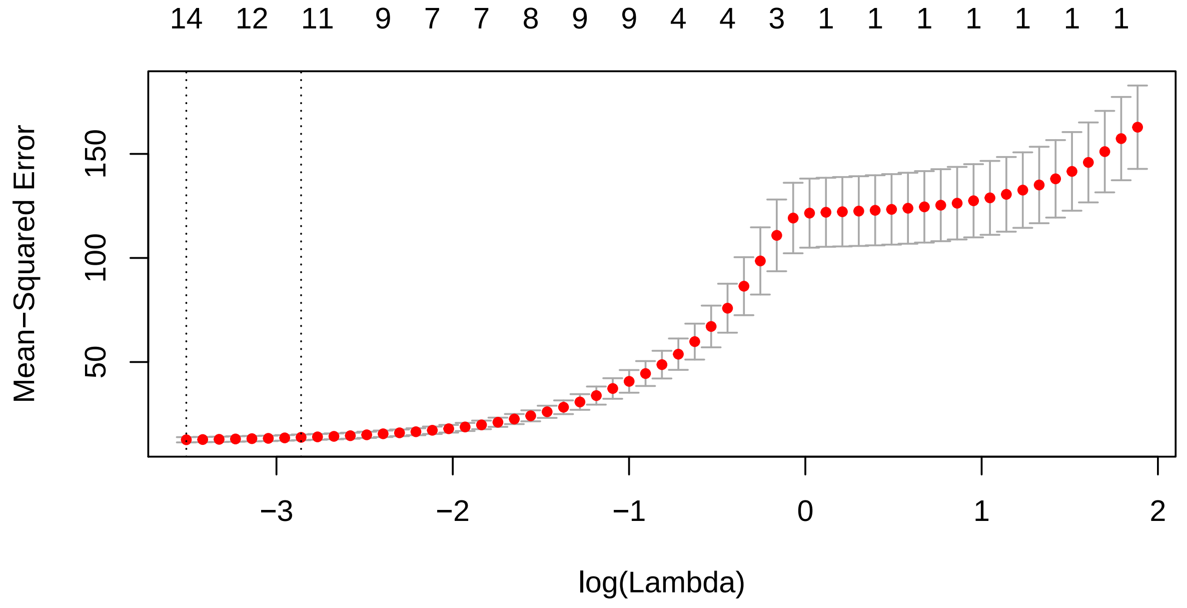
When fitting a LASSO model instead the dependence between the coefficients and log of λ changes. The difference between a Ridge regression model and a LASSO model is that LASSO shrinks the dimensionality by setting some features to zero. In practice LASSO works a bit like a variable selection procedure.



When comparing the plot for the ridge regression and for LASSO the characteristics for the respective methods becomes clear. For the ridge model the coefficients shrinks with higher values of lambda and slowly approaches zero. For the LASSO models coefficients it is harder to se a clear pattern in the dependence between the coefficients and log of λ. The majority of the coefficients quickly equals zero, some has a sort of wiggly curve and some of the coefficients decreases slowly for higher values of log λ.

2.8

To find a optimal LASSO model a possible approach is to use cross-validation. This results in a model with 0.03 as the optimal value of lambda and 14 variables.
How the CV score depend on the log of λ can be investigated with the following plot.



For higher values of the log λ the CV-score increases. The optimal λ was suggested to be approximately 0.03, but as can be seen in the graph it is not that clear which value of λ that is optimal. The difference between the CV scores for log λ values around -3 is very small, whilst the number of variables in the model differs. This might cause some uncertainty over the reliability for the chosen value for the penalty factor.

2.9

The variable selection done with *stepAIC* and the selection done with the cross-validated LASSO model choose a significantly different amount of variables. With *stepAIC* a model with 63 variables were obtained and from the cross-validated LASSO computations a model with 14 variables.

Appendix - R-code

```
## ---- echo=FALSE, eval=TRUE-----------------------------------------------
library(MASS)

# Reads in data set Longley
data(longley)
data <- longley

longley.x <- data.matrix(longley[, 1:6])
longley.y <- longley[, "Employed"]

ridgereg_nfoldCV <- function(x, y, lambda, nfolds){
  # Create the folds and initialize CV vector
  n <- length(y)
  seques <- floor(n/nfolds)
  reps <- n%%nfolds
  groups <- rep(seq(1,nfolds, 1), seques)
```

```r
  end_values <- rep(nfolds, reps)
  folds <- c(groups, end_values)
  folds <- sort(folds)

  #x <- cbind(rep(1, nrow(x)), x)
  CV <- 0
  for (i in 1:nfolds){
    testIndexes <- which(folds==i,arr.ind=TRUE)
    testData_x <- x[testIndexes, ]
    testData_y <- y[testIndexes]
    trainData_x <- x[-testIndexes, ]
    trainData_y <- y[-testIndexes]

    # find the mean value for columns in train, use that mean to scale the values
    # in test. For example, if mean in column 1 in train is 3, then use that value
    # to scale column 1 in test data.
    if (class(testData_x) == "numeric"){
      testData_x <- data.frame(t(testData_x))
    }else{
      testData_x <- data.frame(testData_x)
    }
    mean_x <- 0
    center_test <- matrix(ncol=ncol(testData_x), nrow=nrow(testData_x))
    for (j in 1:ncol(trainData_x)){
      mean_x[j] <- mean(trainData_x[,j])
      center_test[,j] <- testData_x[,j] - mean_x[j]
    }
    testData_y <- testData_y - mean(trainData_y)
    trainData_x <- scale(trainData_x, center=TRUE, scale=FALSE)
    trainData_y <- scale(trainData_y, center=TRUE, scale=FALSE)
    testData_x <- as.matrix(center_test)


    # Perform ridge regression on train data
    x_t <- t(trainData_x)
    I <- diag(ncol(trainData_x))
    BetaRidge <- solve(x_t %*% trainData_x + lambda * I) %*% x_t %*% trainData_y
    # Test regression on test data and compare with true values for test data
    y_hat_test <- testData_x %*% BetaRidge
    # Calculates CV
    CV[i] <- sum((testData_y - y_hat_test)^2)
  }
  CV_score <- (1/nfolds) * sum(CV)
  return(CV_score)
}
CvScore <- 0
for (i in 1:7){
  CvScore[i] <- ridgereg_nfoldCV(longley.x, longley.y, lambda=i, nfolds=10)
}
plot(CvScore, type="l", xlab="Lambda")

## ---- echo=FALSE---------------------------------------------------------
tecator <- read.csv("tecator.csv", sep=";", header = TRUE)
```

```
plot(tecator$Protein, tecator$Moisture)

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

# Extract data sets for moisture and protein data from train and test
Protein_train <- train$Protein
Moisture_train <- train$Moisture
Protein_test <- test$Protein
Moisture_test <- test$Moisture

poly_modelTrain <- list()
MSE_train <- 0
MSE_test <- 0

for (i in 1:6){
poly_modelTrain[[i]] <-  lm(Moisture_train ~ poly(Protein_train, i, raw=TRUE))
MSE_train[i] <- 1/length(Moisture_train) * sum(poly_modelTrain[[i]]$residuals^2)
y_hat_test <- predict(lm(Moisture_train ~ poly(Protein_train, i)), data.frame(Protein_train=Protein_tes
MSE_test[i] <- 1/length(Moisture_test)  * sum((y_hat_test-Moisture_test)^2)
}


plot(1:6, MSE_train, type="l", ylim=c(30,35), col="blue", ylab="MSE", xlab="Model")
lines(1:6, MSE_test, type="l", col="red")
legend(5.25,33.5,c("Train","Test"), lty=c(1,1),
  lwd=c(2.5,2.5),col=c("blue","red"),  cex=0.6)

## ---- echo=FALSE-------------------------------------------------------
AIC <- 0
for (i in 1:6){
  poly_modelTrain[[i]] <-  lm(tecator$Moisture ~ poly(tecator$Protein, i, raw=TRUE))
  n <- length(tecator$Moisture)
  RSS <- sum(poly_modelTrain[[i]]$residuals^2)
  AIC[i] <- 2*(i+1) + n * log(RSS/n)
}
plot(AIC, type="l", xlab="Model")

## ---- echo=FALSE, message=FALSE, warning=FALSE---------------------------
library(glmnet)
vars <- data.frame(tecator[,2:102])
# Create matrix with x variables
mat_vars <- as.matrix(vars[,1:100])
# y variable
mat_y <- as.matrix(vars[,101])

ridge_mod <- glmnet(mat_vars, mat_y, alpha=0, family = "gaussian")
plot(ridge_mod, xvar="lambda",label=TRUE)
```

```
## ---- echo=FALSE---------------------------------------------------------
lasso_mod <- glmnet(mat_vars, mat_y, alpha=1, family = "gaussian")
plot(lasso_mod, xvar="lambda",label=TRUE)

## ---- echo=FALSE---------------------------------------------------------
set.seed(12345)
lasso_cv <- cv.glmnet(mat_vars, mat_y, alpha=1, family = "gaussian")
plot(lasso_cv)

## ----code=readLines(knitr::purl('C:/Users/Gustav/Documents/Machine-Learning/Lab 2/Lab2.Rmd', documenta
##
```