# Helpfile

Axel Holmberg (axeho681)

#Helpful info

**How to ensure that your model is not overfitting?** Keep the design of the model simple. Try to reduce the noise in the model by considering fewer variables and parameters. Cross-validation techniques such as K-folds cross validation help us keep overfitting under control. Regularization techniques such as LASSO help in avoiding overfitting by penalizing certain parameters if they are likely to cause overfitting.

## Code from labs with comments

```
library(kknn)
setwd("~/Programming/TDDE01/Lab 1")

#1.
#Import the data into R and divide it into training and test sets (50%/50%)
data <- read.csv2("spambase.csv")

#Split data into training and test set.

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

#2.
#Use logistic regression (functions glm(), predict()) to classify the training
#and test data by the classification principle
#Y_hat=1 if (Y=1|X) > 0.5, otherwise Y_hat=0
#and report the confusion matrices (use table()) and the misclassification rates for
#training and test data. Analyse the obtained results.

model <- glm(Spam ~ ., family=binomial, data=train) #GLM model for data with family binomial --> only 0

predictModel= predict(model, newdata=test, type="response")

probability <- ifelse(predictModel_test > 0.5, "1", "0") #Split up the model into spam and not spam

confMatrix <- table(probability, test[,"Spam"]) #Confusionmatrix from the model

modelDiag <- diag(confMatrix) #Diagonal of the confMa

#Missclassfication rate by dividing the diagonal from the confusionmatricx with the whole confusionmatr
missClMa1 = 1-(sum(modelDiag)/sum(confMatrix))
```

```r
#Same but for training data
predictModel_train = predict(model, newdata=train, type="response")

probability_train <- ifelse(predictModel_train > 0.5, "1", "0") #Split up the model into spam and not sp
confMatrix_train <- table(probability_train, train[,"Spam"]) #Confusionmatrix from the model
modelDiag_train <- diag(confMatrix_train) #Diagonal of the

missClMa1_train = 1-(sum(modelDiag_train)/sum(confMatrix_train)) #Missclassfication rate by dividing th

#Prints results
print("Confusion matrix 2 test:")
print(confMatrix)
print("missclassification 2 test:")
print(missClMa1)

print("Confusion matrix 2 train:")
print(confMatrix_train)
print("missclassification 2 train:")
print(missClMa1_train)
#3.
#Use logistic regression to classify the test data by the classification principle
#Y_hat=1 if p(Y=1|X) > 0.8, otherwise Y_hat=0
#and report the confusion matrices (use table()) and the misclassification rates for
#training and test data. Compare the results. What effect did the new rule have?
probability2 <- ifelse(predictModel > 0.8, "1", "0") #Split up the model into spam and not spam

confMatrix2 <- table(probability2, test[,"Spam"])

modelDiag2 <- diag(confMatrix2)

missClMa2 = 1-(sum(modelDiag2)/sum(confMatrix2))

#Same but with training data
probability2_train <- ifelse(predictModel_train > 0.8, "1", "0") #Split up the model into spam and not
confMatrix2_train <- table(probability2_train, train[,"Spam"]) #Confusionmatrix from the model
modelDiag2_train <- diag(confMatrix2_train) #Diagonal of the

missClMa2_train = 1-(sum(modelDiag2_train)/sum(confMatrix2_train)) #Missclassfication rate by dividing


print("Confusion matrix 3:")
print(confMatrix2)
print("missclassification 3:")
print(missClMa2)

print("Confusion matrix 3 train:")
print(confMatrix2_train)
print("missclassification 3 train:")
print(missClMa2_train)
#4.
#Use standard classifier kknn() with K=30 from package kknn, report the the
#misclassification rates for the training and test data and compare the results with step 2.
```

```r
#KKNN with K=30

kknn_K30 = kknn(Spam ~ ., train=train, test=test, k=30)
kknn_K30_pred = predict(kknn_K30)

kknn_K30_pred <- ifelse(kknn_K30_pred > 0.5, 1, 0) #Split up the model into spam and not spam


confMa_K30 = table(kknn_K30_pred, test$Spam)
misCl_K30 = 1-sum(diag(confMa_K30)/sum(confMa_K30))

#Training data
confMa_K30_train = table(kknn_K30_pred, train$Spam)
misCl_K30_train = 1-sum(diag(confMa_K30_train)/sum(confMa_K30_train))

print("Missclassification 4 testing:")
print(misCl_K30)

print("Missclassification 4 training:")
print(misCl_K30_train)


#5.
#Repeat step 4 for K=1 and compare the results with step 4. What effect does the decrease of K lead to

#KKNN with K=1
kknn_K1 = kknn(Spam ~ ., train=train, test=test, k=1)
kknn_K1_pred = predict(kknn_K1)

kknn_K1_pred <- ifelse(kknn_K1_pred > 0.5, 1, 0) #Split up the model into spam and not spam

confMa_K1 = table(kknn_K1_pred, test[,"Spam"])
misCl_K1 = 1-sum(diag(confMa_K1)/sum(confMa_K1))

#Training data
confMa_K1_train = table(kknn_K30_pred, train[,"Spam"])
misCl_K1_train = 1-sum(diag(confMa_K1_train)/sum(confMa_K1_train))

print("Missclassification 5 testing:")
print(misCl_K1)

print("Missclassification 5 training:")
print(misCl_K1_train)
```