Machine Learning Lab 1

Phillip Hölscher

22 11 2018

Assignment 1. Spam classification with nearest neighbors

1. Split the data into train and test set.

2. Logistic regression to classify train and test set

and classify by: $\hat{Y}$ = if *if* p(Y = 1|X) 0.5, *otherwise* $\hat{Y} = 0$

Training set

Create the model:

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

Create the prediction (train set):

Create the classification : $\hat{Y}$ = if *if* p(Y = 1|X) 0.5, *otherwise* $\hat{Y} = 0$

Create a confusion matrics (train set):

Test set

Confusion matrices for *train* and *test* set:

Absolute values

```
confusion_matrix_2_train
```

```
##
## glm.pred_train   0   1
##              0 803  81
##              1 142 344
```

```
confusion_matrix_2_test
```

```
##
## glm.pred_test   0   1
##             0 791  97
##             1 146 336
```

Misclassification rate:

Train

```
## [1] 0.16
```

Test

```
## [1] 0.18
```

Analyse the result of Exercise 2:

We can observe a positive classification of 1147 of 1370 objects. This is a correct classification rate of 84% for the train case. For the test case do we have 1127 of 1370 object correct classification. This creates a correct classification rate of 82%. Both results are close to each other. But a misclassification rate of 16% (train) and 18% (test) is not good at all for a spam classification.

3. Logistic regression to classify train and test set

and classify by:

$\hat{Y}$ = if *if* p(Y = 1|X) 0.9, *otherwise* $\hat{Y} = 0$

Confusion matrices for *train* and *test* set:

Train

```
##
## glm.pred_train_3   0   1
##                0 944 419
##                1   1   6
```

Test

```
##
## glm.pred_test_3   0   1
##               0 936 427
##               1   1   6
```

Misclassification rate:

Train

```
## [1] 0.3065693
```

Test

```
## [1] 0.3124088
```

Analyse the result of Exercise 3:

We can observe that the number of Type 1 errors has increased to 31%. The Type 2 Error, the prediction of a negative classification, even if a spam email was present, hardly occurs in the classification. Since an email is classified as spam by a probability higher than 90%, a lot of emails which are actually spam email does not get classified as spam. This can be seen for the Type 1 error in the train and test case. But the type 2 error, got reduces a lot to just 1 in both cases. This means just 1 email, which is actually a spam email, does not get classified as spam. We can´t say this is a good email spam classification, a lot of emails, which are not spam, got classified as spam.

4. K-Nearest Neighbors mit K = 30

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

```
##
##       0   1
##   0 672 187
##   1 265 246
```

Misclassification rate: Train

```
## [1] 0.17
```

```
## [1] 0.33
```

Analyse the result of Exercise 4:

The misclassification rate for the train set is very simular to the in set 2(16% and 17%), but the test misclassification rate is higher than in step 2 (15%).

5. K-Nearest Neighbors mit K = 1

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

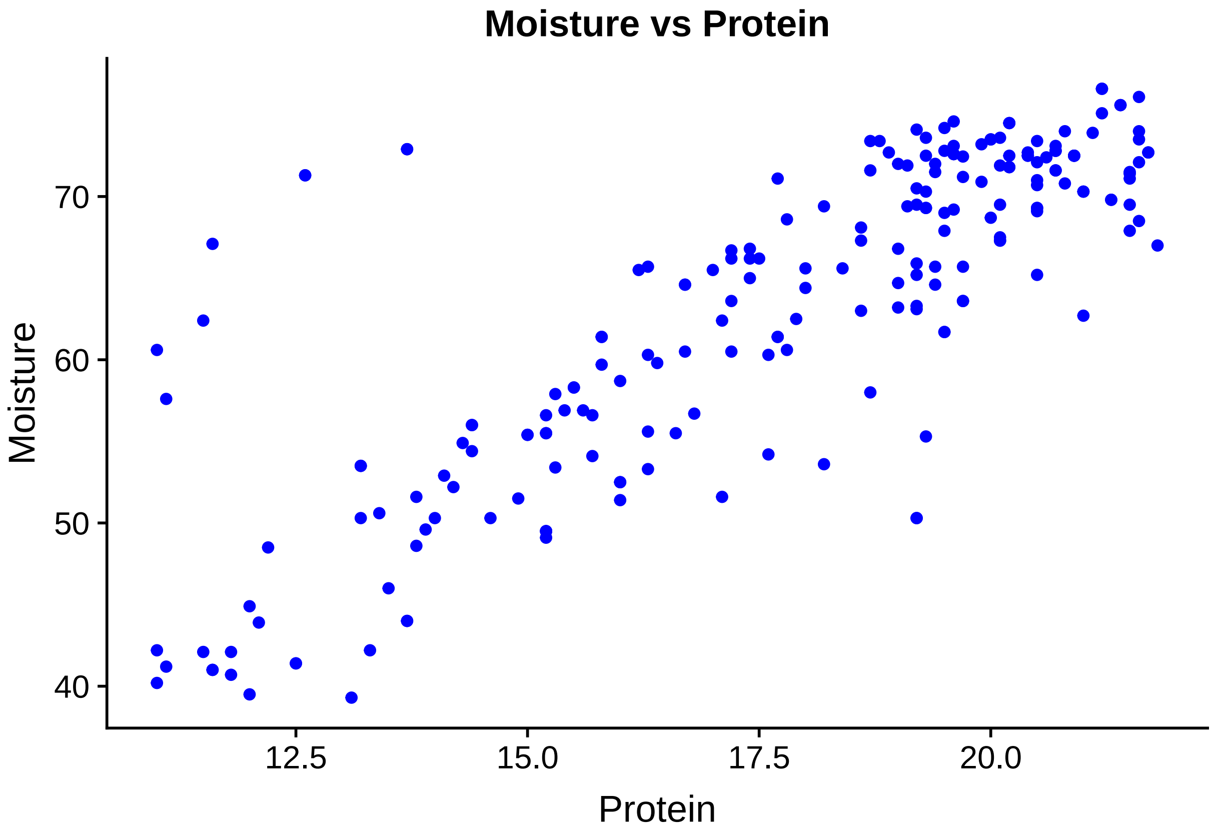
```
##
##       0   1
##   0 640 177
##   1 297 256
```

Analyse the result of Exercise 5:

Reducing k to the value 1 increases the complexity of the model. This means that a perfect result is achieved during training, but a very bad result in the test.

Assignment 4. Linear regression and regularization

1. Import the data and plot Moisture vs Protein



Basically one can say that the data can be described well by a linear model, even if outlier can be recognized. It can be seen that there is a strong positive correlation between the two variables.

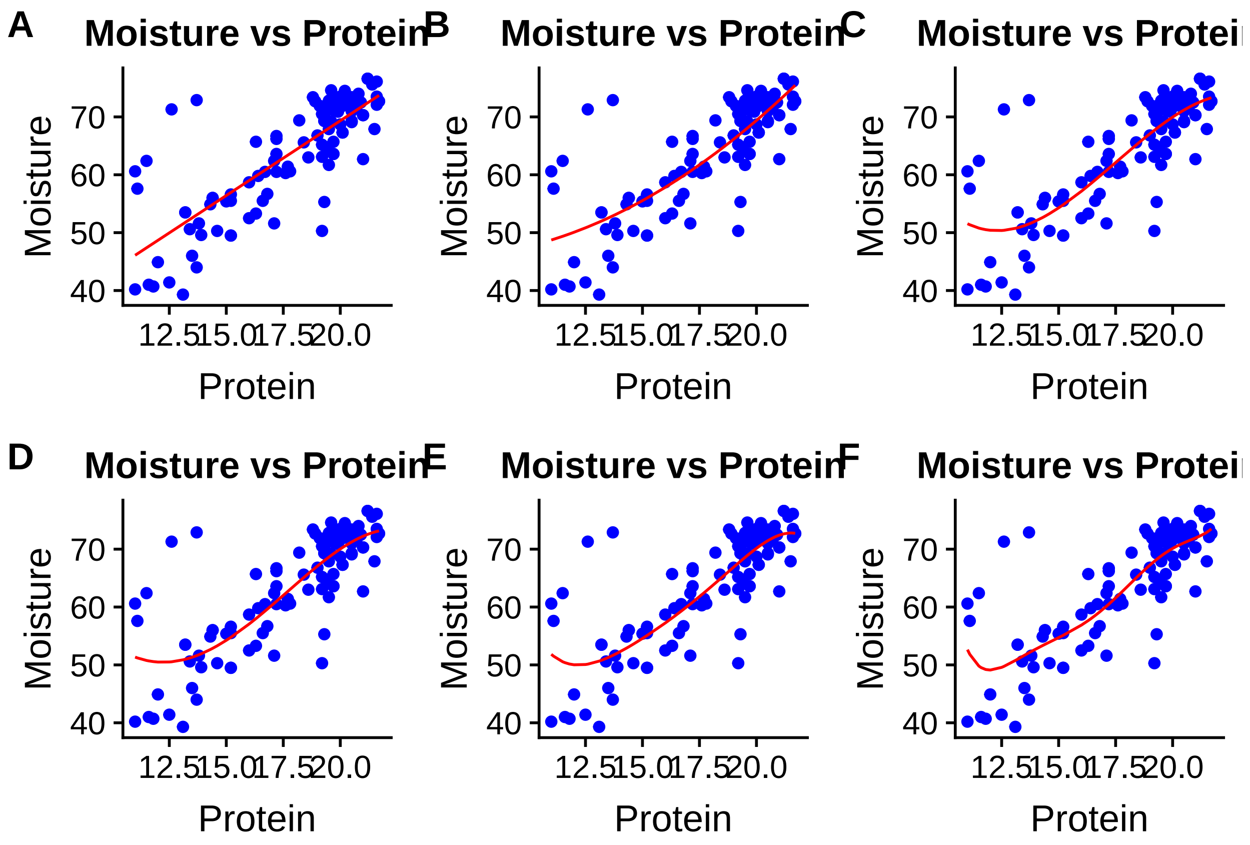
2. Probabilistic model which describes M

Theorie question - Lecture 1d $M_i\tilde{}\ N(prot_0 + prot_1x^1 + ... + prot_ix^i, \sigma^2)$ or $p(M|x, prot) = N(prot_0 + prot_1x^1 + ... + prot_ix^i, \sigma^2)$

3. Divide data set into train and test set, fit models and plot MSE of the models

Plot all model predictions:

- A: $M_1 = prot_1x^1$
- B: $M_2 = prot_1x^1 + prot_2x^2$
- C: $M_3 = prot_1x^1 + prot_2x^2 + prot_3x^3$
- D: $M_4 = prot_1x^1 + prot_2x^2 + prot_3x^3 + prot_4x^4$
- E: $M_5 = prot_1x^1 + prot_2x^2 + prot_3x^3 + prot_4x^4 + prot_5x^5$
- F: $M_6 = prot_1x^1 + prot_2x^2 + prot_3x^3 + prot_4x^4 + prot_5x^5 + prot_6x^6$



It can be seen that the curve of the graph becomes more curved with audible complexity. The model adapts better to the training data and thus receives a strong curvature at the beginning due to the previously mentioned outlier.

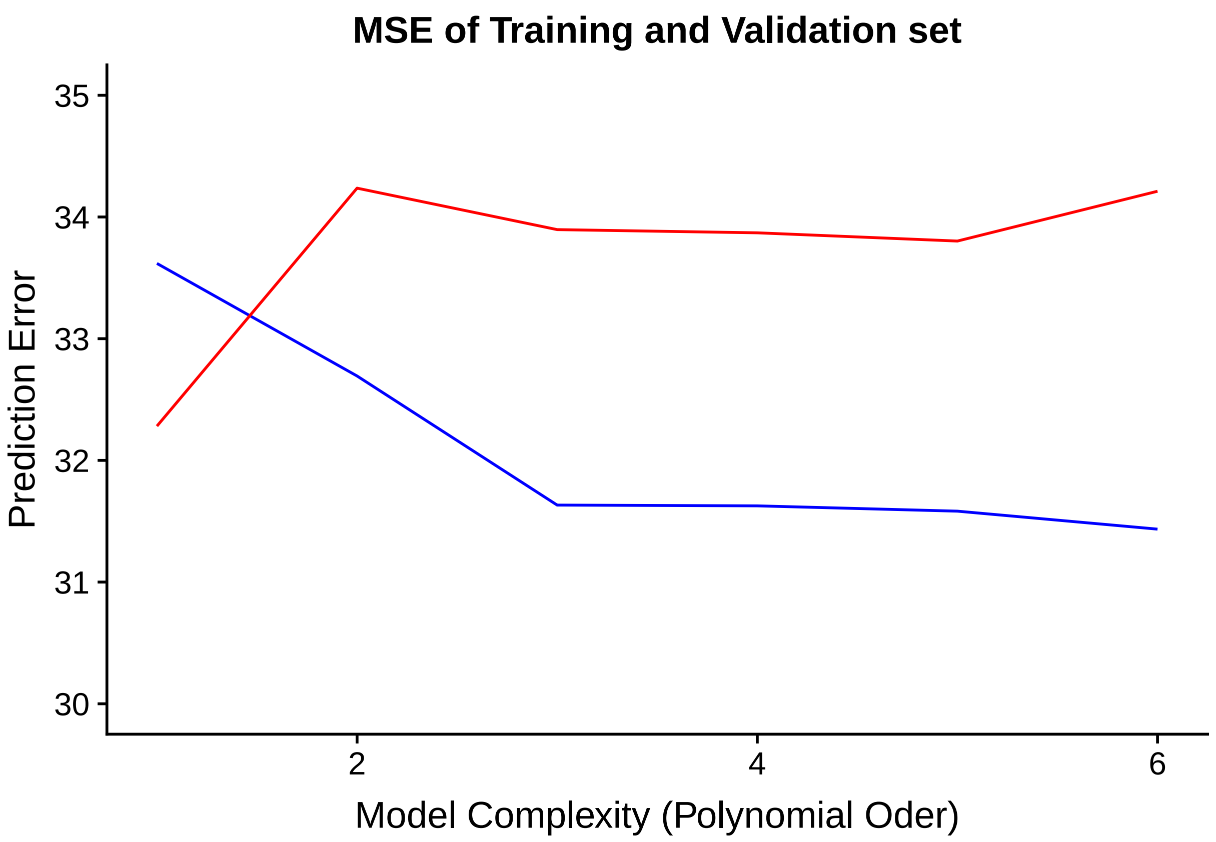
MSE of the training and validation data:

```
mse_data
```

```
##   mse_train_pred mse_validation_pred
## 1       33.61836            32.28154
## 2       32.69342            34.23708
## 3       31.63266            33.89615
## 4       31.62641            33.86992
## 5       31.58273            33.80234
## 6       31.43513            34.21152
```

Plot training and validation MSE:

- Blue: Training
- Red: Validation



We see here that as model complexity increases, the error calculated on the training set continues to decrease, where as the error on the validation set increases past the optimal polynomial order N=1. After the two graphs cross each other, they are further and further apart with higher model complexity.

In the context of the picture “MSE of Training and Validation set” and the “Bias, Variance and Model Complexity”, it can be said that the best model is also the simplest one. As already mentioned in task 4.1, the course of the data is well described by a linear relationship. It was also mentioned that the model adapts better to the training data, as can be seen in Model A to Model F. This overfitted the model, which is why the MSE of the training data decreases with further increase in complexity.

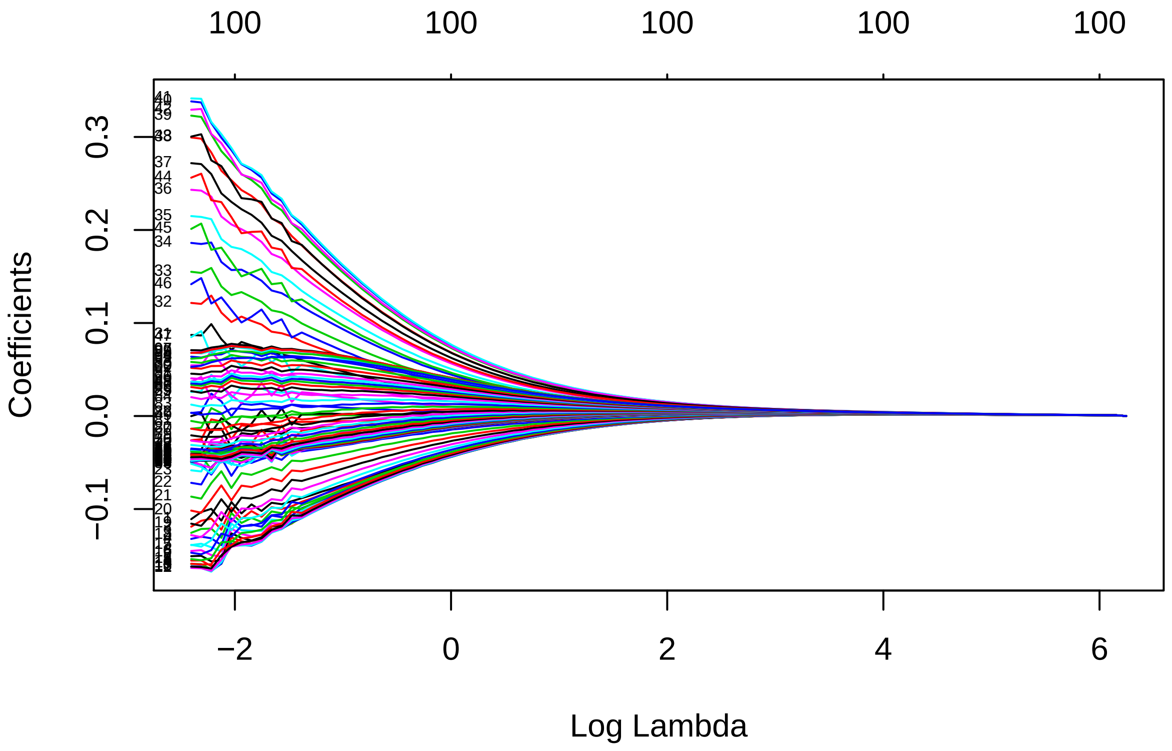
4. Feature selection

It depends on the selection method how many variables are selected.

- Forward selection: 100
- Backward selection: 63
- Selection both ways: 63

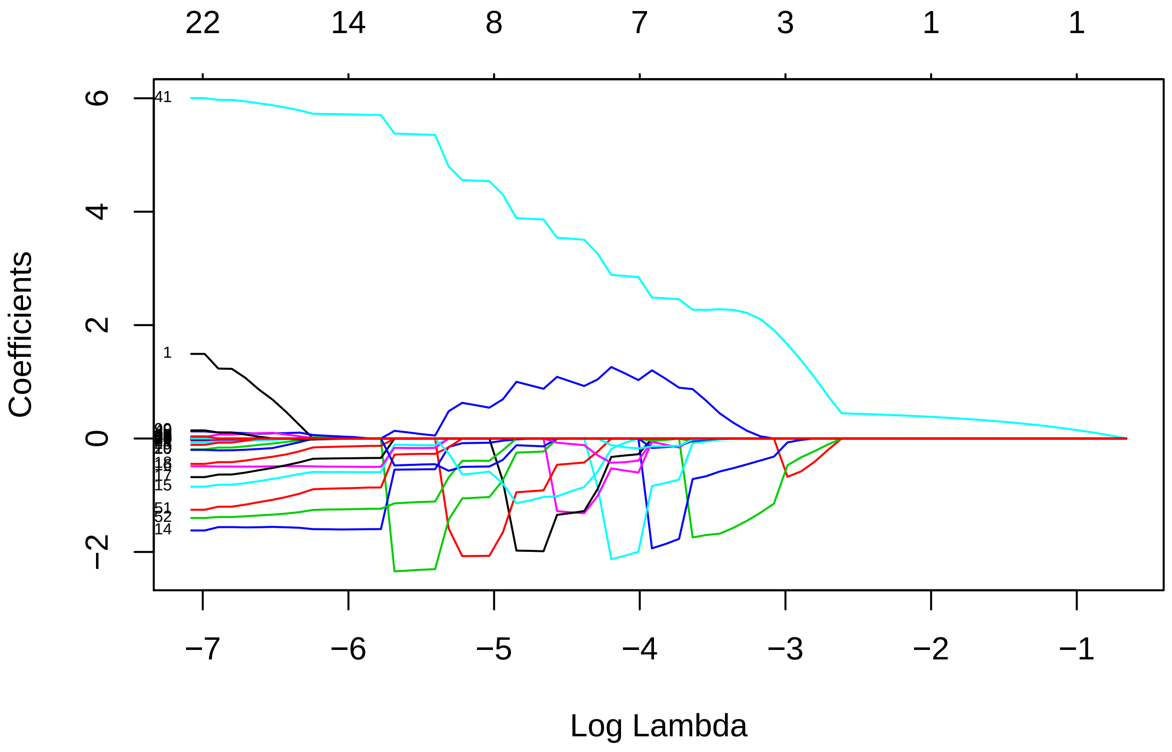
Forward selection does not remove a variable for the model. However, for both Backward and Both, 37 variables are removed and 63 are retained.

5. Ride regression:



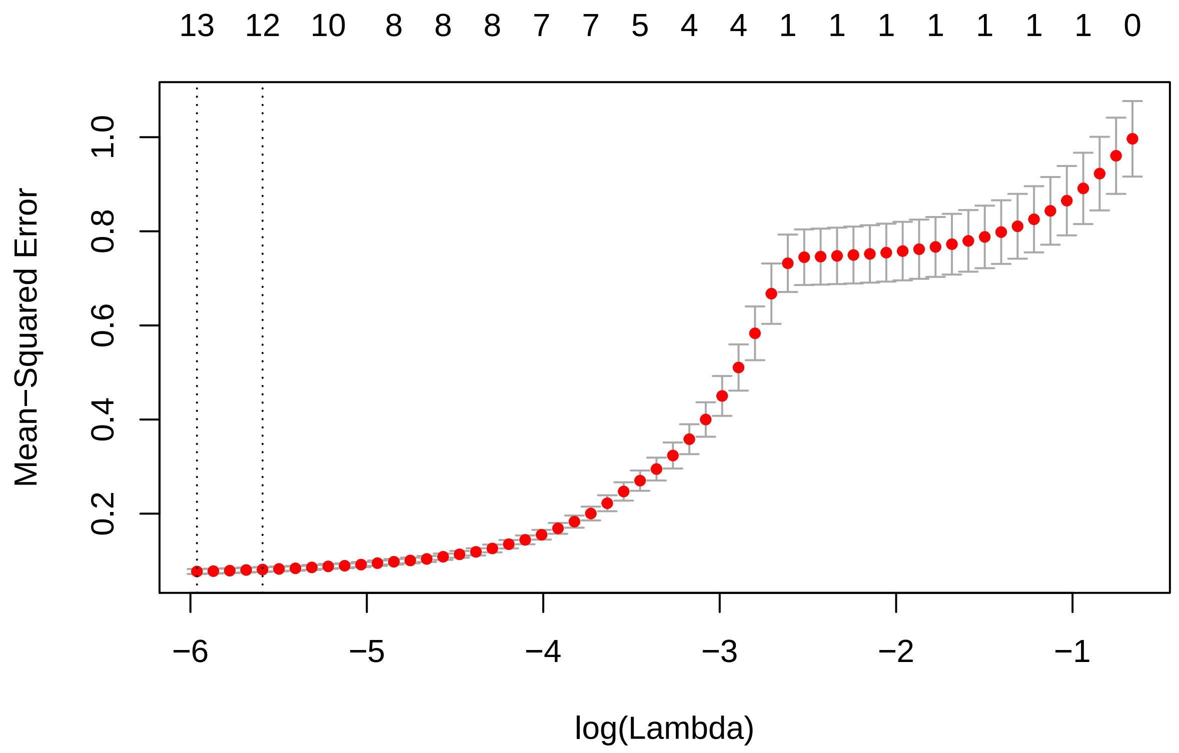
With increasing lambda the coefficients goes towards 0.

6. LASSO regression:



This visualization shows a similar result as in the previous task. With increasing lambda, the coefficents runs against 0.

7. Cross-Validation:



This visualization shows two vertical lines. The first line represents your minimum value of lambda and the second line represents the best. Min lambda = 0.002571916 Best lambda = 0.00373141 It can be seen that the second line runs to 12. This means that 12 variables correspond to the best model.

8. Compare the results for variable selection of a linear model via stepAIC and the cross-validation of the LASSO model.

With ridge all the variables almost equally fast go down to zero. Whereas for Lasso the constraints make some of the go down to zero much faster than the other once. This makes it more in line with step that selects some of the best variables.

```r
knitr::opts_chunk$set(echo = TRUE)
library(readxl)
library(ggplot2)
library(cowplot)
library(hydroGOF)
library(glmnet)
# set working directory
setwd("~/Google Drive/Uni/LiU/Statistics and Machine Learning/Autumn18T2/Machine Learning/Labs")

# read data file
#install.packages("readxl") # it´s an xlsx file - need the readxl package for this
library(readxl)
data <- read_excel("spambase.xlsx")
# this is the given code
n=dim(data)[1]
set.seed(12345)
# sample - sample takes a sample of the specified size from the elements of x
# floor - floor takes a single numeric argument x and returns a numeric
# vector containing the largest integers not greater than the corresponding elements of x.
id=sample(1:n, floor(n*0.5))
train=data[id,]
test=data[-id,]
# Tutorial Logistic Regression with R - Data Camp ####
# https://www.datacamp.com/community/tutorials/logistic-regression-R

# glm() function, which is generally used to fit generalized linear models

# Step1: Explore dataset
names(data)
head(data)
summary(data)
nrow(data)

# Step2: Check if data is missing
missing(data)


# Step3: Visulizing the data
# Skip this part


# Step4: Building Logistic Regression Model
# train set ####

# ------ fit the model ------ #
glm_fit <- glm(Spam ~ .,
                 data = train,
                 family = binomial) # binomial because we deal with logistic regression

# get an overview
#summary(glm_fit)
#head(glm_fit)
 # ------ create prediction train set ------ #
```

```r
glm.probs_train <- predict(glm_fit,
                     type = "response") # remove the dependent var
# type  - for logistic regression - response, this will give use the probability
# listed in a singel vector

# glm.probs  - returns the predicted probabilites that it´s a spam
# head(glm.probs_train)
# classify as spam or not spam based on probability 0.5
glm.pred_train <- ifelse(glm.probs_train > 0.5, 1, 0)
#head(glm.pred_train)
# table uses the cross-classifying factors to build a contingency table
# of the counts at each combination of factor levels.
confusion_matrix_2_train <- table(glm.pred_train, train$Spam)
# ------ create prediction train set ------ #
glm.probs_test <- predict(glm_fit,
                     type = "response",
                     newdata = test[-49])
glm.pred_test <- ifelse(glm.probs_test > 0.5, 1, 0)
confusion_matrix_2_test <- table(glm.pred_test, test$Spam)
confusion_matrix_2_train
confusion_matrix_2_test
### Analyse the result of Exercise 2:
correct_classification_train <- confusion_matrix_2_train[1,1] + confusion_matrix_2_train[2,2]
wrong_classification_train <- sum(confusion_matrix_2_train) - correct_classification_train
### Analyse the result of Exercise 2:
correct_classification_test <- confusion_matrix_2_test[1,1] + confusion_matrix_2_test[2,2]
wrong_classification_test <- sum(confusion_matrix_2_test) - correct_classification_test
missclassificationrate_train <- round(wrong_classification_train/sum(confusion_matrix_2_train),2)
missclassificationrate_train
missclassificationrate_test <- round(wrong_classification_test/sum(confusion_matrix_2_test),2)
missclassificationrate_test
# 3. Exercise - A1
# use the same fit model from 1.2
# clussify as spam if probability > 0.9
# train
glm.pred_train_3 <- ifelse(glm.probs_train > 0.9, 1, 0)

# test
glm.pred_test_3 <- ifelse(glm.probs_test > 0.9, 1, 0)

# create confusion_matrix
confusion_matrix_3_train <- table(glm.pred_train_3, train$Spam)
confusion_matrix_3_test <- table(glm.pred_test_3, test$Spam)
confusion_matrix_3_train
confusion_matrix_3_test
missclassificationrate_train_3 <-
(confusion_matrix_3_train[1,2] + confusion_matrix_3_train[2,1])/ sum(confusion_matrix_3_train)
missclassificationrate_train_3
missclassificationrate_test_3 <- (confusion_matrix_3_test[1,2] + confusion_matrix_3_test[2,1])/ sum(con
missclassificationrate_test_3
#kknn
#install.packages("kknn")
library(kknn)
```

```
# training data
kknn_train <- kknn(formula = as.factor(Spam)~.,  train = train, test = train, k = 30)

# test data
kknn_test <- kknn(formula = as.factor(Spam)~., train = train, test = test, k = 30)
# Confusion matrices
# train
cm_train_4 <- table(kknn_train$fitted.values, train$Spam)
cm_train_4
# test
cm_test_4 <- table(kknn_test$fitted.values, test$Spam)
cm_test_4
missclassificationrate_train_4 <-
round((cm_train_4[1,2] + cm_train_4[2,1])/ sum(cm_train_4),2)
missclassificationrate_train_4
missclassificationrate_test_4 <- round((cm_test_4[1,2] + cm_test_4[2,1])/ sum(cm_test_4),2)
missclassificationrate_test_4
# training data
kknn_train_1 <- kknn(formula = as.factor(Spam)~., train = train, test = train, k = 1)
table(kknn_train_1$fitted.values, train$Spam)
# test data
kknn_test_1 <- kknn(formula = as.factor(Spam)~., train = train, test = test, k = 1)
table(kknn_test_1$fitted.values, test$Spam)
# Import the data
tecatolr_data <- read_excel("tecator.xlsx")
ggplot(tecatolr_data) +
  geom_point(aes(x = tecatolr_data$Protein, y = tecatolr_data$Moisture),
             color = "blue") +
  ggtitle("Moisture vs Protein") +
  xlab("Protein") +
  ylab("Moisture")
n=dim(tecatolr_data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
training_tecatolr=tecatolr_data[id,]
validation_tecatolr=tecatolr_data[-id,]

# fit model to data
Assign4_fit1 <- lm(Moisture ~ Protein,
                  data = training_tecatolr)
Assign4_fit2 <- lm(Moisture ~ Protein + I(Protein^2),
                  data = training_tecatolr)
Assign4_fit3 <- lm(Moisture ~ Protein + I(Protein^2) + I(Protein^3),
                  data = training_tecatolr)
Assign4_fit4 <- lm(Moisture ~ Protein + I(Protein^2) + I(Protein^3) + I(Protein^4),
                  data = training_tecatolr)
Assign4_fit5 <- lm(Moisture ~ Protein + I(Protein^2) + I(Protein^3) + I(Protein^4) + I(Protein^5),
                  data = training_tecatolr)
Assign4_fit6 <- lm(Moisture ~ Protein + I(Protein^2) +I(Protein^3) + I(Protein^4) + I(Protein^5) + I(Pr
                  data = training_tecatolr)
# pred training values
Assign4_pred_training1 <- predict(Assign4_fit1)
Assign4_pred_training2 <- predict(Assign4_fit2)
```

```r
Assign4_pred_training3 <- predict(Assign4_fit3)
Assign4_pred_training4 <- predict(Assign4_fit4)
Assign4_pred_training5 <- predict(Assign4_fit5)
Assign4_pred_training6 <- predict(Assign4_fit6)

# create pred training vector
Assign4_pred_training <-c(Assign4_pred_training1,Assign4_pred_training2,Assign4_pred_training3,Assign4_
# pred training values
Assign4_pred_validation1 <- predict(Assign4_fit1, newdata = validation_tecatolr)
Assign4_pred_validation2 <- predict(Assign4_fit2, newdata = validation_tecatolr)
Assign4_pred_validation3 <- predict(Assign4_fit3, newdata = validation_tecatolr)
Assign4_pred_validation4 <- predict(Assign4_fit4, newdata = validation_tecatolr)
Assign4_pred_validation5 <- predict(Assign4_fit5, newdata = validation_tecatolr)
Assign4_pred_validation6 <- predict(Assign4_fit6, newdata = validation_tecatolr)

# create pred training vector
Assign4_pred_validation <-c(Assign4_pred_validation1,Assign4_pred_validation2,Assign4_pred_validation3,
M1_plot <-ggplot(training_tecatolr) +
  geom_point(aes(x = Protein, y = Moisture),
             color = "blue") +
  ggtitle("Moisture vs Protein") +
  xlab("Protein") +
  ylab("Moisture") +
  geom_line(aes(x = Protein, y = Assign4_pred_training1),
            color = "red",
            linetype = 1)
M2_plot <- ggplot(training_tecatolr) +
  geom_point(aes(x = Protein, y = Moisture),
             color = "blue") +
  ggtitle("Moisture vs Protein") +
  xlab("Protein") +
  ylab("Moisture") +
  geom_line(aes(x = Protein, y = Assign4_pred_training2),
            color = "red",
            linetype = 1)
M3_plot <- ggplot(training_tecatolr) +
  geom_point(aes(x = Protein, y = Moisture),
             color = "blue") +
  ggtitle("Moisture vs Protein") +
  xlab("Protein") +
  ylab("Moisture") +
  geom_line(aes(x = Protein, y = Assign4_pred_training3),
            color = "red",
            linetype = 1)
M4_plot <- ggplot(training_tecatolr) +
  geom_point(aes(x = Protein, y = Moisture),
             color = "blue") +
  ggtitle("Moisture vs Protein") +
  xlab("Protein") +
  ylab("Moisture") +
  geom_line(aes(x = Protein, y = Assign4_pred_training4),
            color = "red",
            linetype = 1)
```

```r
M5_plot <- ggplot(training_tecatolr) +
  geom_point(aes(x = Protein, y = Moisture),
             color = "blue") +
  ggtitle("Moisture vs Protein") +
  xlab("Protein") +
  ylab("Moisture") +
  geom_line(aes(x = Protein, y = Assign4_pred_training5),
            color = "red",
            linetype = 1)

M6_plot <- ggplot(training_tecatolr) +
  geom_point(aes(x = Protein, y = Moisture),
             color = "blue") +
  ggtitle("Moisture vs Protein") +
  xlab("Protein") +
  ylab("Moisture") +
  geom_line(aes(x = Protein, y = Assign4_pred_training6),
            color = "red",
            linetype = 1)

# put the plot together
# install.packages("cowplot")
plot_grid(M1_plot, M2_plot, M3_plot, M4_plot, M5_plot, M6_plot, labels = "AUTO")

# install.packages("hydroGOF")

mse_train_pred1 <- mse(training_tecatolr$Moisture,Assign4_pred_training1)
mse_train_pred2 <- mse(training_tecatolr$Moisture,Assign4_pred_training2)
mse_train_pred3 <- mse(training_tecatolr$Moisture,Assign4_pred_training3)
mse_train_pred4 <- mse(training_tecatolr$Moisture,Assign4_pred_training4)
mse_train_pred5 <- mse(training_tecatolr$Moisture,Assign4_pred_training5)
mse_train_pred6 <- mse(training_tecatolr$Moisture,Assign4_pred_training6)

mse_train_pred <- c(mse_train_pred1,mse_train_pred2,mse_train_pred3,mse_train_pred4,mse_train_pred5,mse_


mse_validation_pred1 <- mse(validation_tecatolr$Moisture,Assign4_pred_validation1)
mse_validation_pred2 <- mse(validation_tecatolr$Moisture,Assign4_pred_validation2)
mse_validation_pred3 <- mse(validation_tecatolr$Moisture,Assign4_pred_validation3)
mse_validation_pred4 <- mse(validation_tecatolr$Moisture,Assign4_pred_validation4)
mse_validation_pred5 <- mse(validation_tecatolr$Moisture,Assign4_pred_validation5)
mse_validation_pred6 <- mse(validation_tecatolr$Moisture,Assign4_pred_validation6)

mse_validation_pred <- c(mse_validation_pred1, mse_validation_pred2, mse_validation_pred3, mse_validati
mse_data <- data.frame(mse_train_pred,mse_validation_pred)
mse_data

labes <- c("Training" = "blue", "Validation" = "red")

ggplot() +
  ggtitle("MSE of Training and Validation set") +
  xlab("Model Complexity (Polynomial Oder)") +
  ylab("Prediction Error") +
  ylim(30,35) +
  # theme(legend.position = "bottom") +
  # guides(color = c("red", "blue")) +
  # scale_fill_manual(name="Graphs",values= c("red", "blue")) +
  geom_line(aes(x = c(1:length(mse_train_pred)) , y = mse_train_pred),color = "blue") +
  geom_line(aes(x = c(1:length(mse_validation_pred)) , y = mse_validation_pred),
            color = "red") +
  labs(colour = c("blue", "red"))

#install.packages("MASS")
library(MASS)
# just use the data fat and channel1-channel100
tecatolr_data_4.4 <- tecatolr_data[,2:102]


# fit the model
fit_4 <- lm(formula = Fat~. ,
            data = tecatolr_data_4.4)
# summary(fit_4)

# fecture selection with stepAIC
# backwards
step_back <- stepAIC(fit_4, direction = "backward")
#summary(step_back)

#forwards
step_for <- stepAIC(fit_4, direction = "forward")
#summary(step_for)

step_both <- stepAIC(fit_4, direction = "both")
#summary(step_both)

# compare the AIC´s
min(step_back$anova$AIC)
min(step_for$anova$AIC)
min(step_both$anova$AIC)

# how many variables were selected
step_back$rank -1  # -1 because of the dependent variable
step_for$rank -1
step_both$rank -1
#install.packages("glmnet")
response <- scale(tecatolr_data[,102])
colnames(response) <- NULL
covariates <- scale(tecatolr_data[,2:101])
colnames(covariates) <- NULL

# create the ridge regression
ridge_reg <- glmnet(x = as.matrix(covariates), y = response , alpha=0 ,family="gaussian")
# create plot
plot(ridge_reg, xvar="lambda", label=TRUE)
# create the LASSO regression
lasso_reg <- glmnet(x = as.matrix(covariates), y = response , alpha=1 ,family="gaussian")
```

```r
# create plot
plot(lasso_reg, xvar="lambda", label=TRUE)
# cross validation
lasso_reg_cv <- cv.glmnet(x = as.matrix(covariates), y = response , alpha=1 ,family="gaussian")

# lasso_reg_cv$lambda.min
#coef(lasso_reg_cv, s="lambda.min")


#min value of lambda
lambda_min <- lasso_reg_cv$lambda.min
#best value of lambda
lambda_1se <- lasso_reg_cv$lambda.1se
#regression coefficients
#coef(lasso_reg_cv,s=lambda_1se)

# plot
plot(lasso_reg_cv)
```