Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering
IDP Institut für Datenanalyse und Prozessdesign

HS 2016

Statistisches Data Mining (StDM)

Woche 6

Aufgabe 1 Lab

Read and do the excersises of chapter 4.6.1, 4.6.2, and 4.6.5 in ILSR

Aufgabe 2 Logistic Regression (based on an excerice in ISLR)

In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set. You might want to take a look at chapter 4.6.2 in ISLR before you start the excerise.

a) Create a binary variable, mpg01, that contains `TRUE` if mpg contains a value above its median, and `FALSE` if mpg contains a value below its median. Note you may find it helpful to use the data.frame() function to create a single data set containing both mpg01 and the other Auto variables. Attention (for our German and Swiss students and the lecturer!) mpg is millage per gallon, the larger the less fuel the car needs.

```
library(ISLR)
attach(Auto)
```

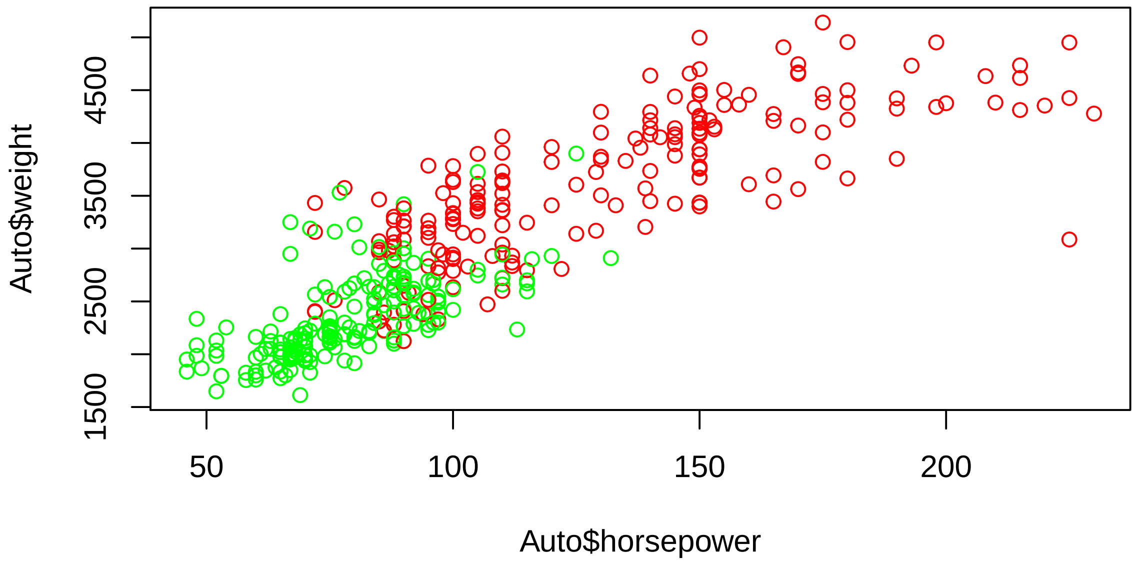
```
## The following objects are masked from Auto (pos = 3):
##
##     acceleration, cylinders, displacement, horsepower, mpg, name,
##     origin, weight, year

## The following objects are masked from Auto (pos = 4):
##
##     acceleration, cylinders, displacement, horsepower, mpg, name,
##     origin, weight, year

## The following objects are masked from Auto (pos = 7):
##
##     acceleration, cylinders, displacement, horsepower, mpg, name,
##     origin, weight, year

## The following object is masked from package:ggplot2:
##
##     mpg
```

```
med = median(Auto$mpg)
Auto$mpg01 = Auto$mpg > med
```

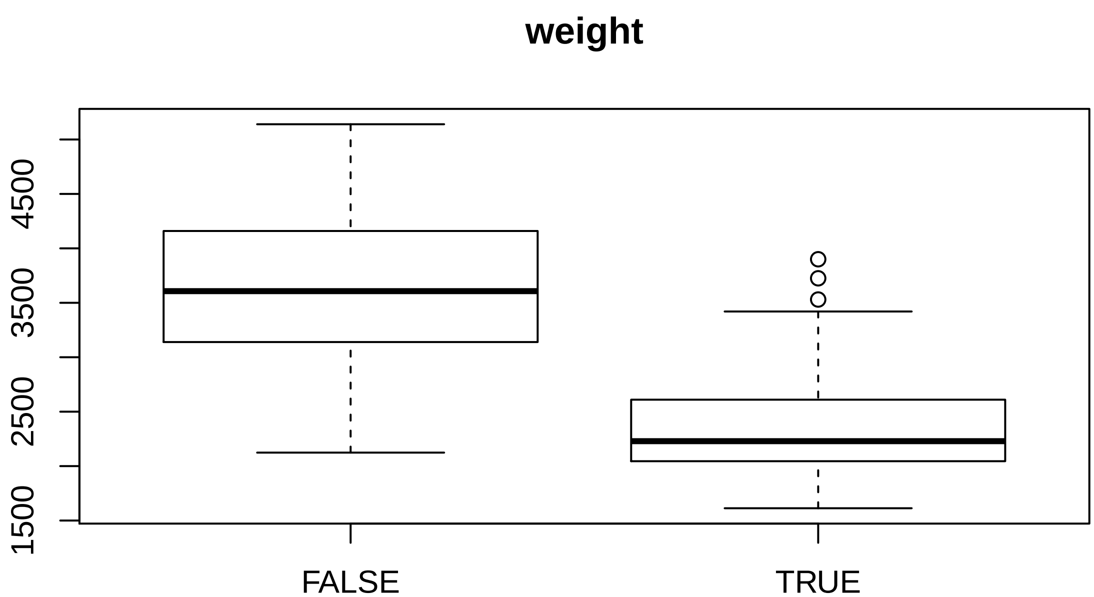
b) Explore the data graphically in order to investigate the association between `mpg01` and the other features. Which of the other features seem most likely to be useful in predicting mpg01?

Scatterplots, tables and boxplots may be useful tools to answer this question. Describe your findings.

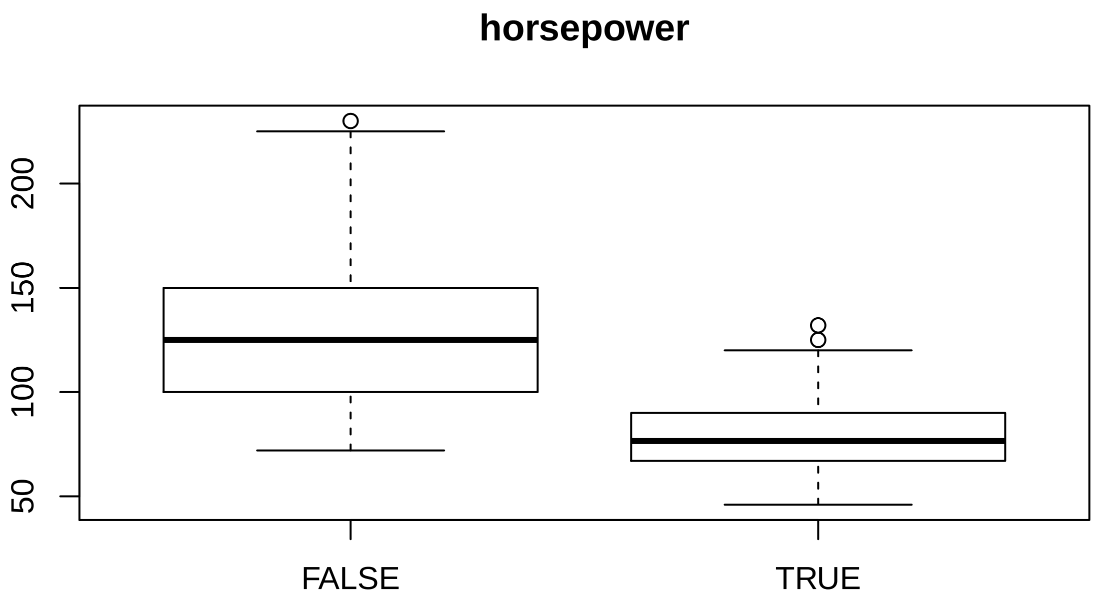
```
col = ifelse(Auto$mpg01, 'green', 'red')
plot(Auto$horsepower, Auto$weight, col=col)
```



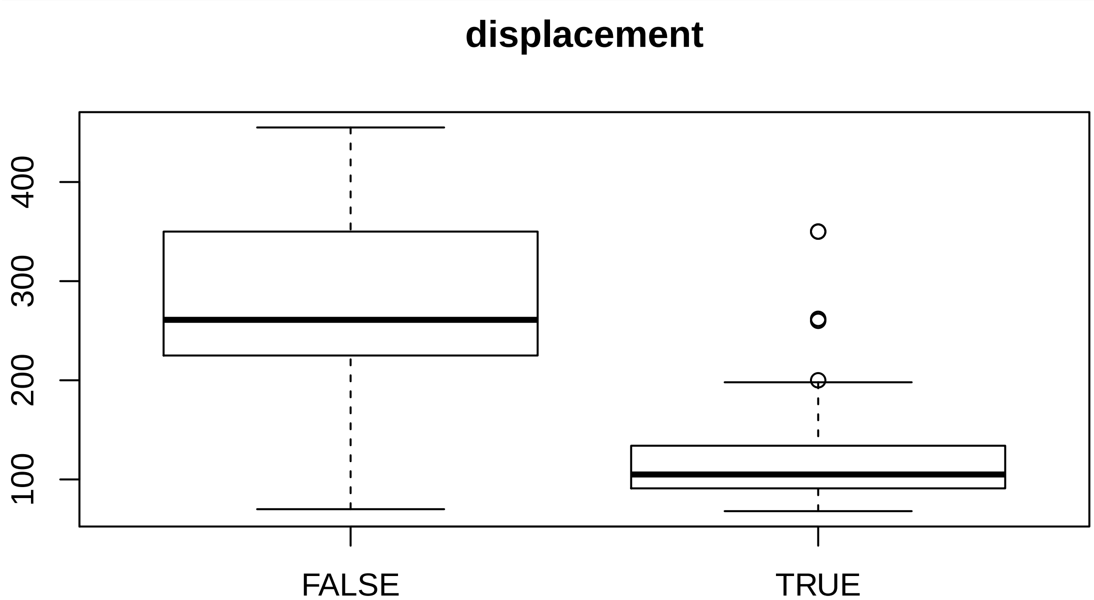

```
boxplot(Auto$weight ~ Auto$mpg01, main='weight')
```



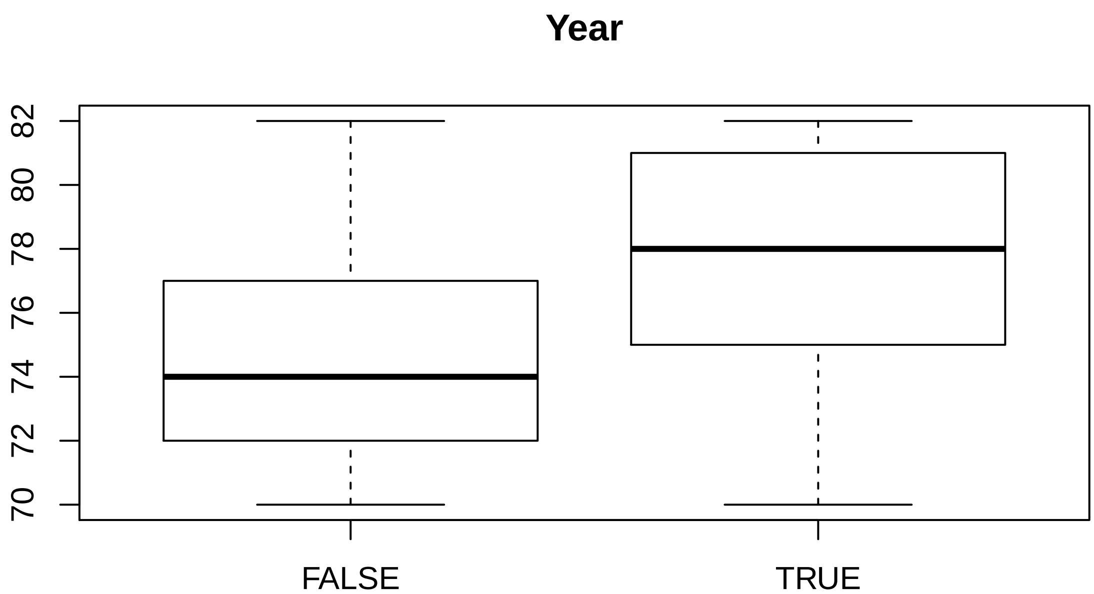

```
boxplot(Auto$horsepower ~ Auto$mpg01, main='horsepower')
```




```r
boxplot(Auto$displacement ~ Auto$mpg01, main='displacement')
```




```r
boxplot(Auto$year ~ Auto$mpg01, main='Year') # longer in the later years
```




```
table(Auto$origin , Auto$mpg01) #1 = USA needs gas
```

```
##
##     FALSE TRUE
##   1   173   72
##   2    14   54
##   3     9   70
```

```
table(Auto$cylinders , Auto$mpg01) #The more cylinders the less millage
```

```
##
##     FALSE TRUE
##   3     3    1
##   4    20  179
##   5     1    2
##   6    72   11
##   8   100    3
```

c) Randomly split the data into a training set (80 %) and a test set (20 %). One way is to draw random number from 1 to `nrow(Auto)` using `sample`. Note that `Auto[-idx,]` takes all but the indices.

```
set.seed(1)
idx_train = sample(nrow(Auto), 0.8 * nrow(Auto))
auto_train = Auto[idx_train,]
auto_test = Auto[-idx_train,]
nrow(auto_test)
```

```
## [1] 79
```

```
nrow(auto_train)
```

```
## [1] 313
```

```
nrow(Auto)
```

```
## [1] 392
```

d) Perform logistic regression on the training data in order to predict mpg01 using all variables except name and mpg. What is the error on the training set and on the test set of the model obtained?

```r
model.all = glm(mpg01 ~ cylinders + displacement + horsepower + weight + acceleration + year + ori
sum((predict(model.all, auto_train) > 0.5) == auto_train$mpg01) / nrow(auto_train)
```

```
## [1] 0.9073482
```

```r
sum((predict(model.all, auto_test) > 0.5) == auto_test$mpg01) / nrow(auto_test)
```

```
## [1] 0.8860759
```

e) Repeat d) using a knn with $k = 1, 5$ classifier. Produce two matrices with the features 'cylinders','displacement', 'horsepower', 'weight', 'acceleration', 'year' one from the training set and one from the test set. See

```r
library(class)
vars = c('cylinders','displacement', 'horsepower', 'weight', 'acceleration', 'year')
X_train = as.matrix(auto_train[, vars])
X_test = as.matrix(auto_test[, vars])
res = knn(train = X_train, test = X_test, cl=auto_train$mpg01, k = 1)
sum(res == auto_test$mpg01) / nrow(auto_test)
```

```
## [1] 0.8481013
```

```r
res = knn(train = X_train, test = X_test, cl=auto_train$mpg01, k = 1)
sum(res == auto_test$mpg01) / nrow(auto_test)
```

```
## [1] 0.8481013
```

```r
res = knn(train = X_train, test = X_test, cl=auto_train$mpg01, k = 5)
sum(res == auto_test$mpg01) / nrow(auto_test)
```

```
## [1] 0.8607595
```

f) Do a clever feature engineering. What preprocessing would you do for `knn`? Redo the analysis in e).

```r
 res = knn(train = scale(X_train), test = scale(X_test), cl=auto_train$mpg01, k = 5)
 sum(res == auto_test$mpg01) / nrow(auto_test)
```

```
## [1] 0.9493671
```

g) What is problematic with the feature `origin` in logistic regression and what could you do to solve it.

```r
# The feature origin is a categorical variable coded as integers. One should do factor before.
 model.matrix(model.all)[1:2,]
```

```
##     (Intercept) cylinders displacement horsepower weight acceleration year
## 106           1         8          360        170   4654         13.0   73
## 148           1         4           90         75   2108         15.5   74
##     origin
## 106      1
## 148      2
```

```r
 auto_train$origin = as.factor(auto_train$origin)
 auto_test$origin = as.factor(auto_test$origin)
 model.all = glm(mpg01 ~ cylinders + displacement + horsepower + weight + acceleration + year + or
 model.matrix(model.all)[1:2,]
```

```
##     (Intercept) cylinders displacement horsepower weight acceleration year
## 106           1         8          360        170   4654         13.0   73
## 148           1         4           90         75   2108         15.5   74
##     origin2 origin3
## 106       0       0
## 148       1       0
```

```
sum((predict(model.all, auto_test) > 0.5) == auto_test$mpg01) / nrow(auto_test)
```

```
## [1] 0.9493671
```

Aufgabe 3 Nochmals Krabben (Decission Surface)

a) Laden Sie den Krabbendatenset crabs des packages MASS ein. Führen Sie mit den numerischen Variablen des Crabs Datensatz eine PCA durch und verwenden PC1 und PC2, als neue Variablen für einen logistischen Klassifier. Bestimmen Sie die Konfusionmatrix auf dem Trainingsset (insample).

```
library(MASS)
pca <- prcomp(crabs[,4:8])
df = data.frame(PC1 = pca$x[,1], PC2 = pca$x[,2], sex = crabs$sex) #glm braucht einen Dataframe
fit = glm(sex ~ PC1 + PC2, data = df, family = binomial)
res.insample = predict(fit, df, type='response')
table(ifelse(res.insample > 0.5, 1, 0) ,crabs$sex)
```

```
##
##      F  M
##   0 90 15
##   1 10 85
```

b) Zeichen Sie, wie im letzten Arbeitsblatt den Scatterplot von $X_1 = PC1$ und $X_2 = PC2$ und färben die Krabben nach Ihrem Geschlecht. Zeichen Sie die 'Decission Boundary', d.h. die Punkte für die gilt $p(X_1, X_2) = p(Y = 1|X_1, X_2) = 0.5$ in den Plot ein. Verwenden Sie dazu die Formel für das odds-ratio:

$$\ln\left(\frac{p(X_1, X_2)}{1 - p(X_1, X_2)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

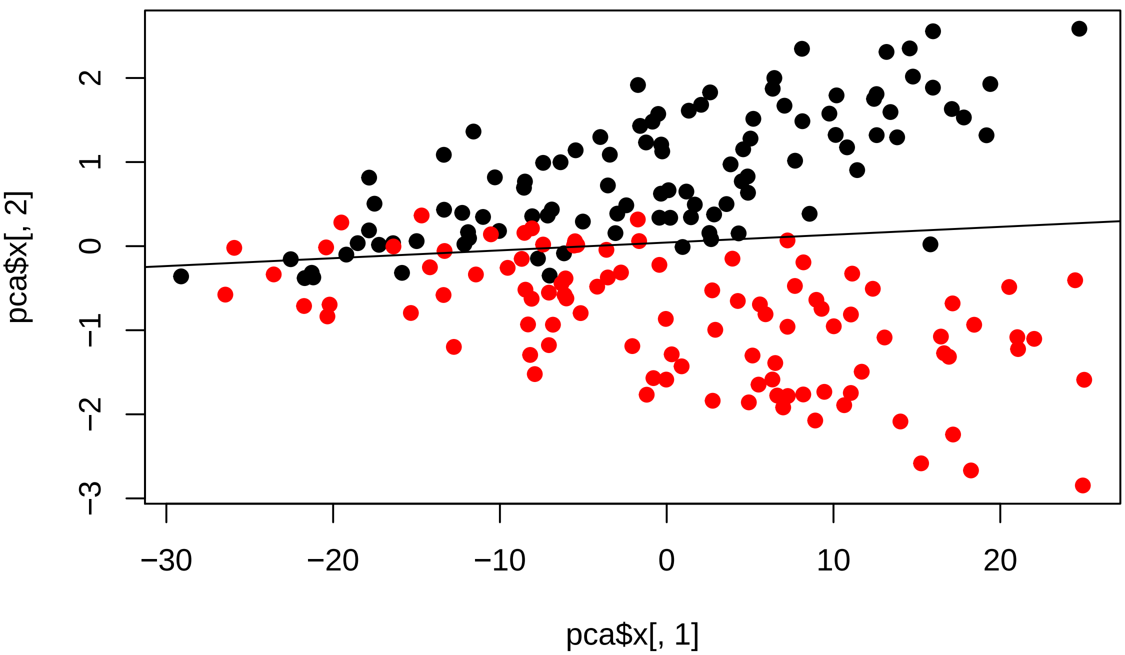
.
Wir setzten p=0.5, mit log(1)=0 folgt:,

$$0 = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

Nach X_2 auflösen:

$$X_2 = -\frac{\beta_0}{\beta_2} - \frac{\beta_1}{\beta_2} X_1$$

```
plot(pca$x[,1],pca$x[,2],pch=19,col=crabs$sex)
slope <- -coef(fit)[2]/(coef(fit)[3])
intercept <- -coef(fit)[1]/(coef(fit)[3])
abline(a = intercept, b=slope)
```



pca$x[, 2]
pca$x[, 1]
−3
−2
−1
0
1
2
−30
−20
−10
0
10
20