Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering
IDP Institut für Datenanalyse
und Prozessdesign

HS 2016

Statistisches Data Mining (StDM)

Woche 7

Aufgabe 1 Boston crime data (LDA vs logistische Regression)

Verwenden Sie den Boston Datensatz (boston aus package MASS) um vorherzusagen, ob ein Stadtteil (Zeile im Datensatz) eine grosse Kriminalitätsrate hat (grösser als der Median alle Stadtteile). Verschaffen Sie sich einen Überblick über die Daten und verwenden Sie logistische Regression sowie LDA. Teilen Sie das Datenset in ein Trainings und in ein Testset (je 50% der Daten)

a) Logistische Regression

```
require(MASS)
```

```
## Loading required package: MASS
```

```
## Warning: package 'MASS' was built under R version 3.2.2
```

```
attach(Boston)
crime01 = rep(0, length(crim))
crime01[crim>median(crim)] = 1
Boston = data.frame(Boston, crime01)

train = 1:(dim(Boston)[1]/2)
test = (dim(Boston)[1]/2+1):dim(Boston)[1]
Boston.train = Boston[train,]
Boston.test = Boston[test,]
crime01.test = crime01[test]

glm.fit = glm(crime01 ~. -crime01-crim, data=Boston.train, family=binomial)
```

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

```
glm.probs = predict(glm.fit, Boston.test, type="response")
glm.pred = rep(0, length(glm.probs))
glm.pred[glm.probs > 0.5] = 1
mean(glm.pred != crime01.test)
```

```
## [1] 0.1818182
```

b) LDA

```
lda.fit = lda(crime01~.-crime01-crim, data=Boston.train)
lda.pred = predict(lda.fit, Boston.test)
mean(lda.pred$class != crime01.test)
```

```
## [1] 0.1343874
```

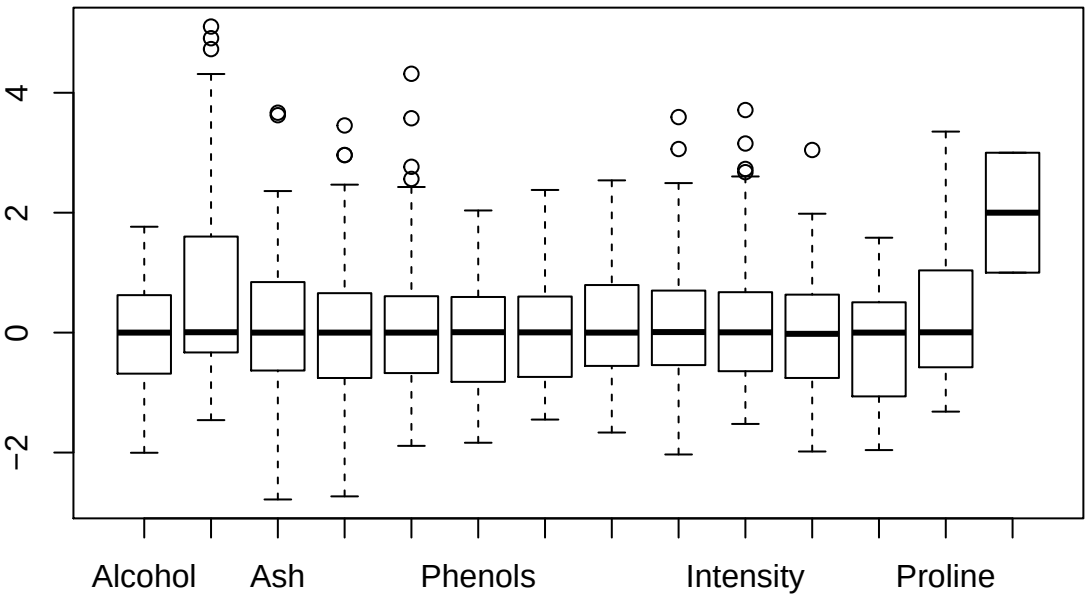
Aufgabe 2 Weinerkennung (Leave one out crossvalidation)

Der in dieser Aufgabe betrachtete Datensatz enthält die Resultate der chemischen Analyse von 178 italienischen Weinen. Es wurden die folgenden Grössen gemessen:

Alcohol:	Alkoholgehalt	**Malic:**	Apfelsäure
Ash:	Menge der Rückstände	**Alcalinity:**	Basizität von Rückständen
Magnesium:	Magnesiumgehalt	**Phenols:**	Totaler Phenolgehalt
Flavanoids:	Geschmack tragende Phenole	**Nonflavanoid:**	neutrale Phenole
Proanthocyanins:	Proanthocyanin	**Intensity:**	Farbintensität
Hue:	Farbton	**OD280:**	OD280/OD315
Proline:	Prolingehalt		

a) Laden Sie die Daten und fertigen einen einen Boxplot der Daten an. Was fällt auf?

```
wein <- read.csv(file.path(baseDir, "Weine3.dat"), sep="", header = TRUE)
wein$Sorte = as.factor(wein$Sorte)
boxplot(wein) #Die Daten scheinen irgendwie normiert zu sein.
```



b) Verwenden Sie LDA, um die Weinsorte vorherzusagen. Beginnen Sie mit einem naiven Ansatz bei dem Sie auf dem gesamten Datensatz trainieren und vorhersagen. Berechnen Sie die Konfusionsmatrix zum Beispiel mit dem Befehl `confusion` aus dem `mda` Packet.

```
require(mda) #For confusion
```

```
## Loading required package: mda
```

```
## Warning: package 'mda' was built under R version 3.2.5
```

```
## Loading required package: class
```

```
## Loaded mda 0.4-9
```

```
require(MASS)
wein.lda = lda(Sorte ~ ., data=wein)
wein.pred = predict(wein.lda, wein)
confusion(true=wein$Sorte, object=wein.pred$class)
```

```
##          true
## predicted  1  2  3
##         1 59  0  0
##         2  0 70  0
##         3  0  0 48
## attr(,"error")
## [1] 0
```

c) Schreiben Sie eine Leave-One-Out Kreuzvalidierung in dem Sie die i-te Zeile für das Training des LDA Klassifiers weglassen und dann die i-te Zeile vorhersagen. Erzeugen Sie sich so einen Vektor, der jeweils die vorhergesage Sorte enthält und bestimmen Sie so die Konfusionsmatrix.

Tipp: Die i-te Zeile können Sie mit `x[-i,]` weglassen.

```
preds = rep(0, nrow(wein))
for (i in 1:nrow(wein)) {
  wein.lda = lda(Sorte ~ ., data=wein[-i,])
  preds[i] = predict(wein.lda, wein[i,])$class
}
confusion(true=wein$Sorte, object=preds)
```

```
##          true
## predicted  1  2  3
##         1 59  1  0
##         2  0 68  0
##         3  0  1 48
## attr(,"error")
## [1] 0.01129944
```

d) Eine schnelle LOO-Kreuzvalidierung kann man durch die Option `CV=TRUE` im lda Befehl erreichen, schauen Sie sich die Hilfe an und berechnen Sie die Konfusionsmatrix analog zu c).

```
wein.lda.cv = lda(Sorte ~ ., data=wein, CV=TRUE)
confusion(true=wein$Sorte, object=wein.lda.cv$class)
```

```
##          true
## predicted  1  2  3
##         1 59  1  0
##         2  0 68  0
##         3  0  1 48
## attr(,"error")
## [1] 0.01129944
```

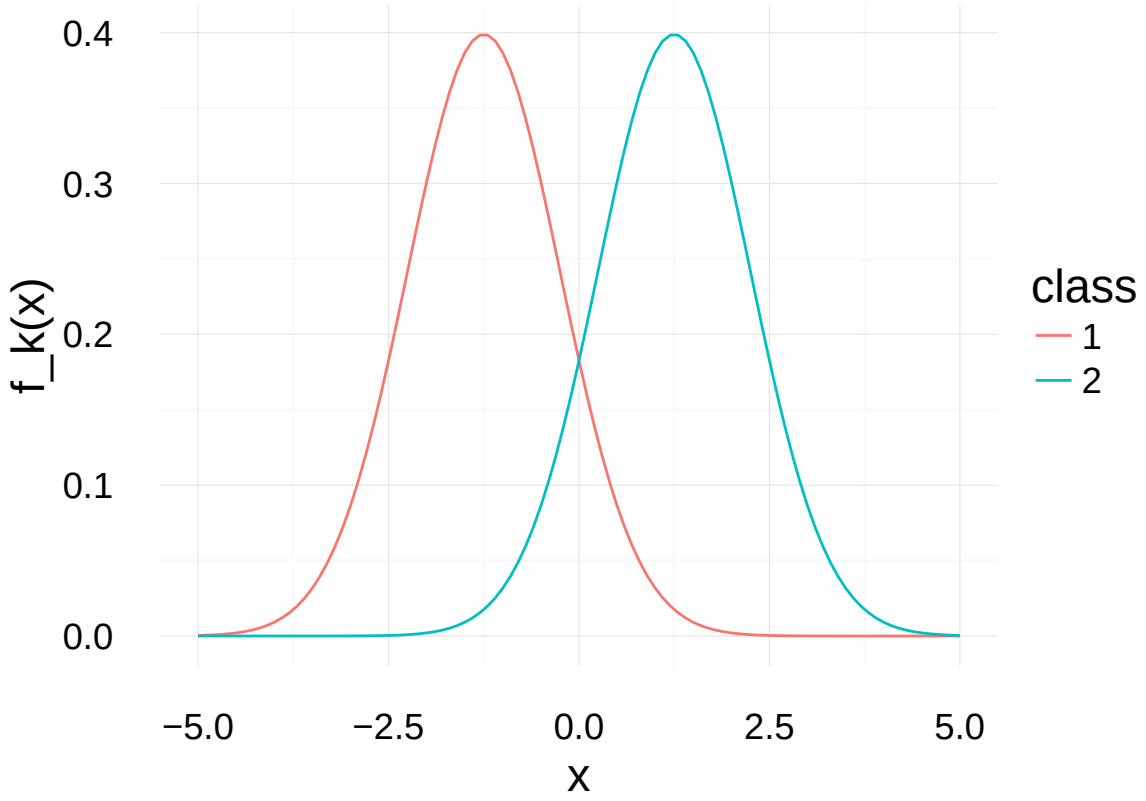
Aufgabe 3 Simulationsstudie (LDA im Vergleich zu Bayes)

Ziel dieser Aufgabe ist es festzustellen, wie nahe ein Klassifikator an der optimalen Lösung ist. Wie in der Vorlesung besprochen ist die LDA als Bayes Klassifikator optimal, falls man die Verteilungen kennt. Allerdings ist die Situtaion, bei der man die Verteilung kennt etwas artifiziel. Wir simulieren daher Daten aus einer bekannten Verteilung und versuchen evaluieren die Performance des LDA Klassifikators im Vergleich zum Bayes Klassifikator

Wir betrachten im folgenden 2 normalverteilte Likelihooddichten $f_k(x)$, für die Klasse $k = 0$ um $\mu_0 = -1.25$ verteilt, für die Klasse 1 um $\mu_0 = -1.25$ verteilt. Die Standartabweichung ist jedesmal 1.

a) Was ist die optimale 'Accuracy'? Fragen Sie sich zuerst wo ist hier die Grenze ist, bei der Sie x zur Klasse 1 bzw. zur Klasse 2 zuordnen. Tragen Sie Konfusionsmatrix die jeweiligen

Prozentzahlen ein.




```
  pnorm(0, -1.25, 1)
```

```
## [1] 0.8943502
```

b) Simulationsstudie. Ziehen Sie nun $N = 10$ Zufallszahlen aus der Klasse 1 und 2, wobei $\pi_1 = \pi_2 = 0.5$ ist. Sie können dazu die Funktion `makeData` verwenden (versuchen Sie den code zu verstehen). Lernen Sie einen LDA Klassifier mit diesen Daten und wenden ihn auf 100 Testdaten an, die Sie immer neu aus der gleichen Verteilung ziehen. Wiederholen Sie das Experiment 1000 mal. Wie ist die Verteilung der Accucary, was ist der Mittelwert? Vergleichen Sie Ihn mit a)

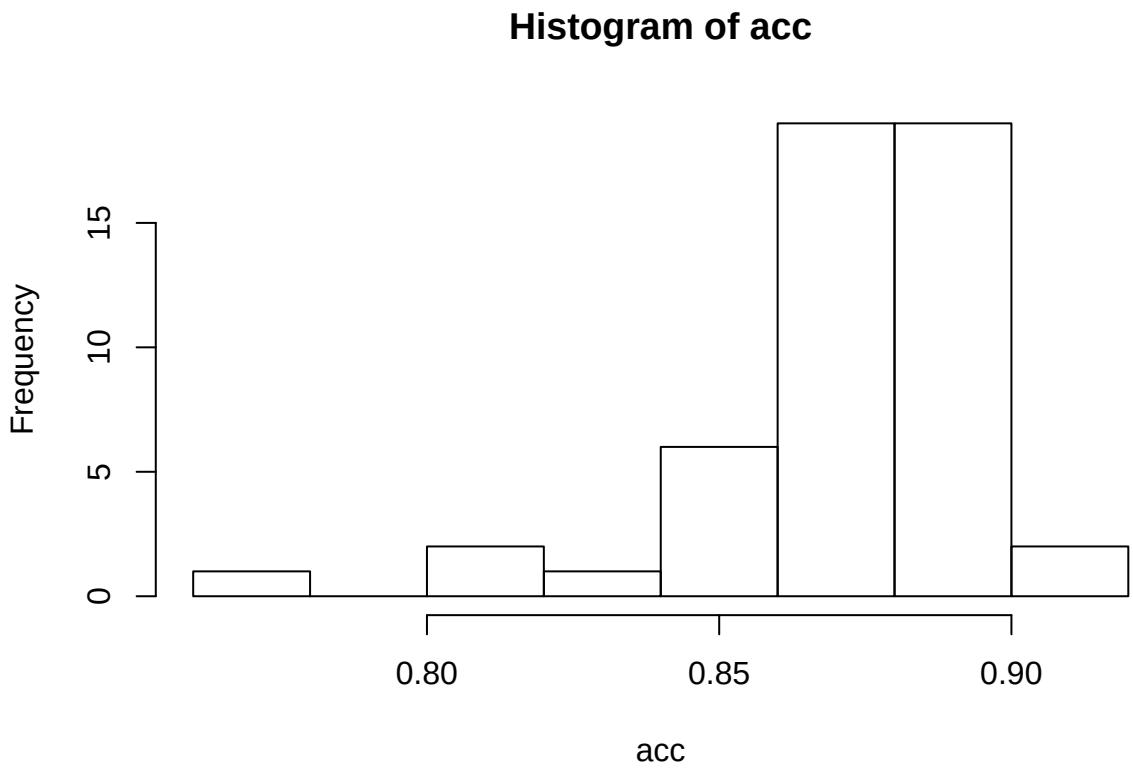
```
makeData = function(N=10) {
    z = rbinom(1,N,prob=0.5)
    #########
    # Ein kleiner Hack. Die lda Funktion muss Beispiele von beiden Klassen sehen können
    # Sollte man eigendlich in der Auswertungsfunktion berücksichtigen
    # Allerdings ist für grosses N der Fehler vernachlässgbar.
    if (z == 0) {
      z == 1
    }
    if (z == N) {
      z == N-1
    }
    y = c(rep(0, z), rep(1, N-z))
    x = c(rnorm(z,-1.25,1), rnorm(N-z,+1.25,1))
    return (data.frame(y = as.factor(y), x = x))
  }
```

```r
library(MASS)
checkAcc = function(df, method = 'LDA') {
  if (method == 'LDA') {
    #Es gibt nur von einer Klasse im diesen, sollte man alles zu dieser Klasse zuordnen.
    #Leider schaft das die LDA function in R nicht
    if ((sum(df$y == 0) == 0) | (sum(df$y==1) == 0)) {
      return (0.5)
    }
    fit=lda(y~x,data=df)
    df_test =  makeData(1000)
    pred = predict(fit, df_test) #Testing Error
    sum(pred$class == df_test$y) / 1000
  }
}

N = 10
acc = rep(NA, 50) #<-- Hier eigendlich 1000 mal
for (r in 1:length(acc)) {
  df = makeData(N)
  acc[r] = checkAcc(df)
}
mean(acc)
```

```
## [1] 0.87172
```

```r
hist(acc)
```



c) Wiederholen Sie Aufgabe b) und bestimmen die mittlere Accuracy für $N = 5, 10, 20, 50, 70, 100, 200$. Tragen Sie dies geeignet auf. Gegen welchen Wert strebt die mittlere Accuracy?

```
REPS = 50 #Höher für bessere Schätzung
N = c(5,10,20,50,70,100,200)
acc = matrix(nrow = length(N), ncol = REPS)
for (i in 1:length(N)) {
  for (r in 1:REPS) {
    df = makeData(N[i])
    acc[i, r] = checkAcc(df)
  }
}
rowMeans(acc)
```

```
## [1] 0.83166 0.87068 0.88632 0.88894 0.89294 0.89232 0.89406
```

```
plot(N, rowMeans(acc), ylim=c(0.75, 0.95))
abline(h=pnorm(0,-1.25))
```