Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering
IDP Institut für Datenanalyse
und Prozessdesign

HS 2016

Statistisches Data Mining (StDM)

Praktikum Woche 13

Aufgabe 1 Boosting

In dieser Aufgabe werden verschieden Klassifikationsmethoden auf simulierte Daten angewandt und die Performance verglichen.

a) Simulieren Sie 12'000 mal 10 Standardnormalverteilte Beobachtungen $X_1, X_2, ..., X_{10}$. Generieren Sie dann die Variable Y_i aufgrund der simulierten Daten folgendermassen:

$$Y_i = \begin{cases} 0 & \text{falls } \sum_{i=1}^{10} X_i^2 <= 9.34 \\ 1 & \text{sonst} \end{cases}$$

Verwenden Sie 2000 Beispiele zum Trainieren und 10000 zum Testen.

```
n<-12000;p<-10
set.seed(100)
x<-matrix(rnorm(n*p),ncol=p)
y<-as.factor(c(0,1)[as.numeric(apply(x^2,1,sum)>9.34)+1])

indtrain<-sample(1:n,2000,FALSE)
train<-data.frame(y=y[indtrain],x[indtrain,])
indtest<-setdiff(1:n,indtrain)
test<-data.frame(y=y[indtest],x[indtest,])
```

b) Erstellen Sie einen Klassifikationsbaum auf dem Trainings-Datensatz der nur aus dem Stumpf besteht. Schätzen Sie die Fehlerrate mithilfe des Test-Datensatzes ab. Hinweis:

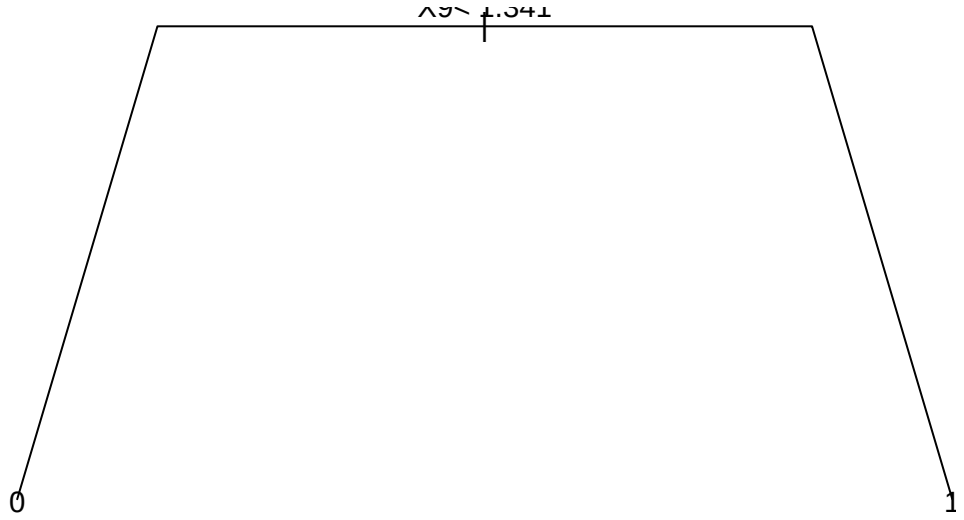
```
treeSingleStump<-rpart(y ~ .,data=train,
     control=list(maxdepth=1, cp=-1,minsplit=0,xval=0),method="class")
```

```
library(rpart)
library(mda)
```

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

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

```
treeSingleStump<-rpart(y~.,data=train,control=list(maxdepth=1,
                           cp=-1,minsplit=0,xval=0),method="class")
plot(treeSingleStump, branch=0.7, uniform=T)
text(treeSingleStump, use.n=TRUE, cex=0.9)
```




```
print(treeSingleStump)
```

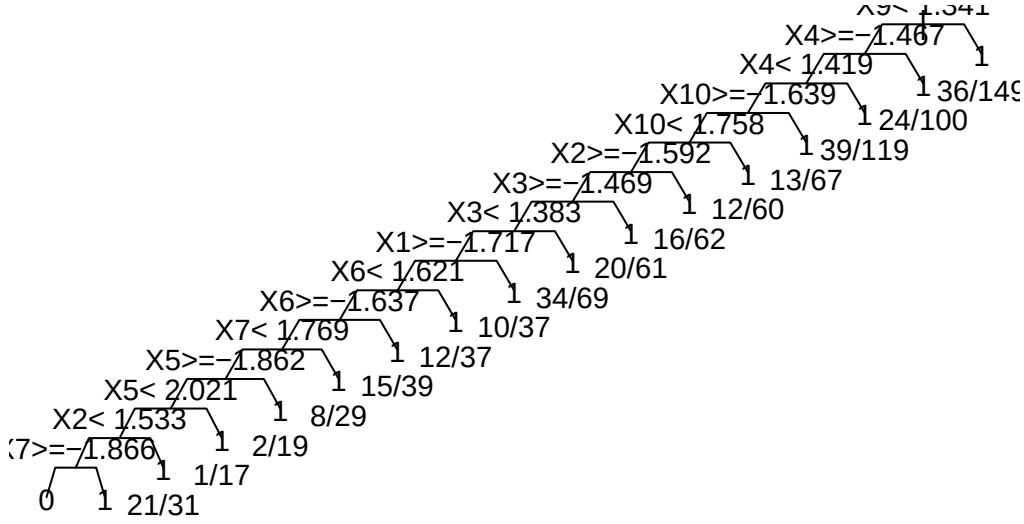
```
## n= 2000
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 2000 990 1 (0.4950000 0.5050000)
##   2) X9< 1.340778 1815 861 0 (0.5256198 0.4743802) *
##   3) X9>=1.340778 185  36 1 (0.1945946 0.8054054) *
```

```
confusion(true=test$y,  predict(treeSingleStump,newdata=test,type="class"))
```

```
##           true
## predicted    0    1
##         0 4794 4329
##         1  233  644
```

c) Erstellen Sie einen Klassifikationsbaum basierend auf den internen Stopp-Kriterien, berechnen Sie die Fehlerrate.

```
tree.default<-rpart(y~.,data=train,method="class")
plot(tree.default, branch=0.7, uniform=T)
text(tree.default, use.n=TRUE, cex=0.9)
```




```
  confusion(true=test$y,
        predict(tree.default,newdata=test,type="class"))
```

```
##          true
## predicted    0    1
##         0 3209  677
##         1 1818 4296
```

d) Wenden Sie einen Random Forest auf die Daten an.

```
  library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
  rf = randomForest(y ~ ., data=train)
  confusion(true=test$y,
        predict(rf,newdata=test,type="class"))
```

```
##          true
## predicted    0    1
##         0 3992  358
##         1 1035 4615
```

e) Wenden Sie Adaboost auf die Daten an. Verwenden Sie dazu das packet `gbm` und spielen Sie mit den Parametern `n.trees` und `interaction.depth` bis Sie einen Error kleiner als 0.12 bekommen. Hinweis: Sie müssen die Zielvariable wieder in einen numerischen Wert (0,1) umwandeln, verwenden Sie die distribution='adaboost'.

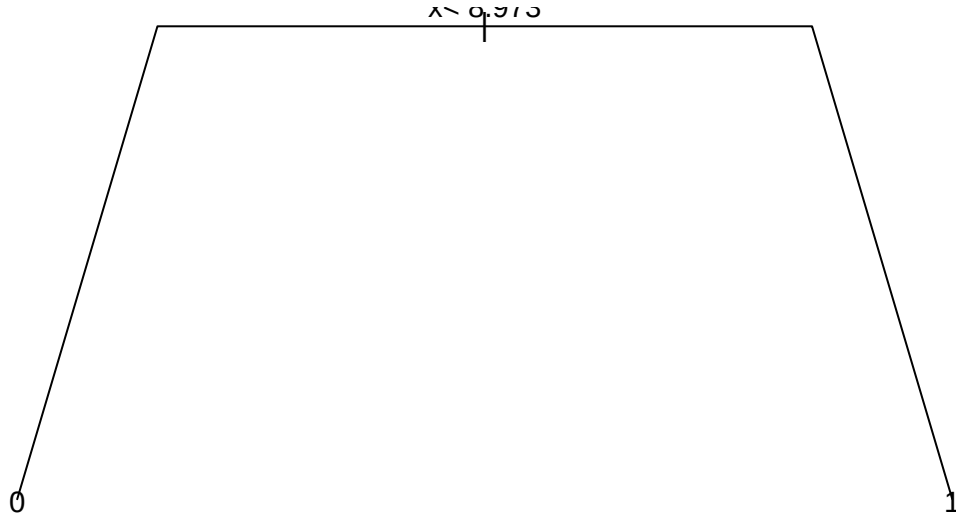
```
  library(gbm)
  train$y = as.integer(train$y) - 1
  test$y = as.integer(test$y) - 1
  gbt = gbm(y ~ ., data=train, n.trees = 20000, interaction.depth=2, distribution = "adaboost")
  pred = predict(gbt,newdata=test, n.trees = 20000, type = 'response')
  confusion(true=test$y, ifelse(pred > 0.5, 1, 0))
```

```
##          true
## predicted    0    1
##         0 4601  485
##         1  426 4488
```

f) Angenommen Sie kennen den Prozess der Datenerzeugung wie in a) beschrieben bis auf die Konstante 9.34. Überlegen Sie sich eine geschickte Transformation der Variablen $X_1, X_2 \dots X_10$ in eine neue Variable X und wenden Sie diese Transformation an. Verwenden Sie einen Klassifikationsbaum wie in c) und bestimmen Sie die Fehlerrate.

```
X = apply(train, MARGIN = 1, FUN=function(x) sum(x[2:10]^2))
train1 = data.frame(y=as.factor(train$y), x=X)
X = apply(test, MARGIN = 1, FUN=function(x) sum(x[2:10]^2))
test1 = data.frame(y=as.factor(test$y), x=X)

tree.default<-rpart(y~.,data=train1,method="class")
plot(tree.default, branch=0.7, uniform=T)
text(tree.default, use.n=TRUE, cex=0.9)
```




```
confusion(true=test1$y,predict(tree.default,newdata=test1,type="class"))
```

```
##           true
## predicted    0    1
##         0 4900  682
##         1  127 4291
```