Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering
IDP Institut für Datenanalyse und Prozessdesign

HS 2016

Statistisches Data Mining (StDM)

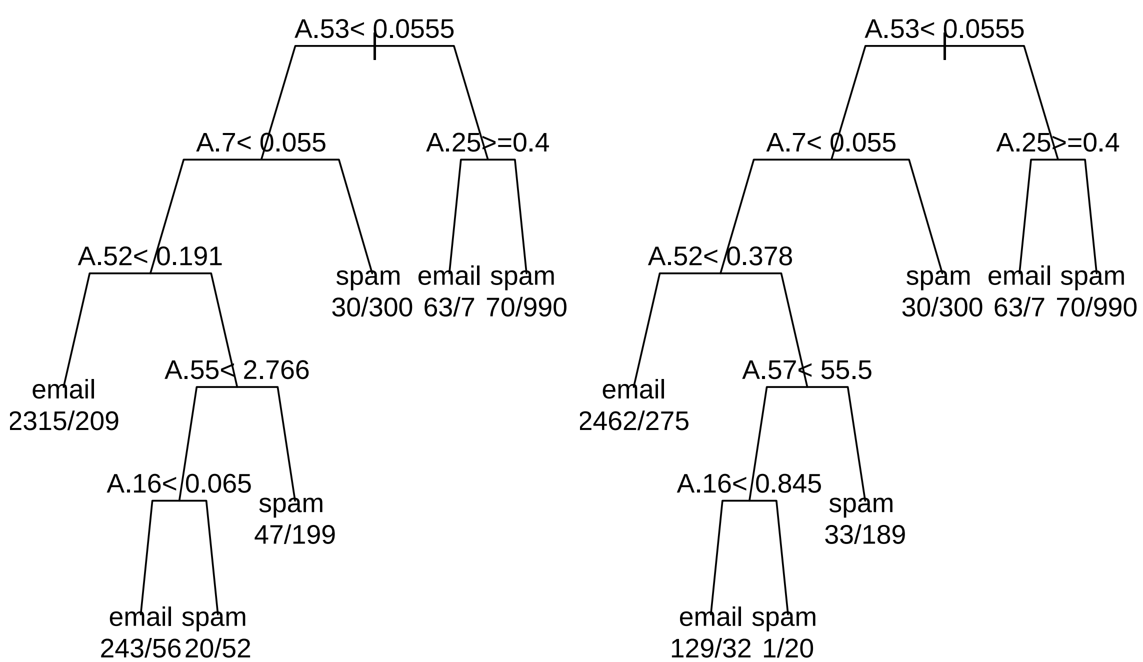
Praktikum Woche 12

Aufgabe 1 Random Forest

In this excercise, we try to detect spam given some features of the email. Source: A part of the exercise is from Dr. Markus Kalisch.

a) Have a look at the data set “spam“in the package “ElemStatLearn“. Fit a classification tree based on the gini and the information-criteria. Calculate the naive error rate.

```
library(ElemStatLearn)
data(spam)
library(rpart)
#erstelle Klassifikationsb?ume
spam.rpI <- rpart(spam ~ ., data=spam, method="class",
                  parms=list(split='information'))
spam.rpG <- rpart(spam ~ ., data=spam, method="class",
                  parms=list(split='gini'))
#stelle die Bäume grafisch dar
par(mfrow=c(1,2), mar=c(1,1,4,1), xpd=T)
plot(spam.rpI, branch=0.7, uniform=T); text(spam.rpI, use.n=TRUE, cex=0.9)
plot(spam.rpG, branch=0.7, uniform=T); text(spam.rpG, use.n=TRUE, cex=0.9)
```



```r
#Die Kriterien führen zu unterschiedlichen Resultaten

library(mda)
confusion(true=spam$spam,object=predict(spam.rpI,newdata=spam, type="class"))
```

```
##          true
## predicted email spam
##     email  2621  272
##     spam    167 1541
```

```r
confusion(true=spam$spam,object=predict(spam.rpG,newdata=spam, type="class"))
```

```
##          true
## predicted email spam
##     email  2654  314
##     spam    134 1499
```

b) Calculate the error rate based upon 10-fold cross-validation for gini method (default)

```r
library(caret)
test_folds = createFolds(spam$spam, 10) #Creating 10 folds
t.pr <- rep(NA, nrow(spam))
count = 1
for(test in test_folds){
  t.rp <- rpart(spam ~ ., data=spam[-test,],method="class")
  t.pr[test] <-predict(t.rp, newdata=spam[test,], type="class")
}
confusion(true=as.numeric(as.factor(spam$spam)), object=t.pr)
```

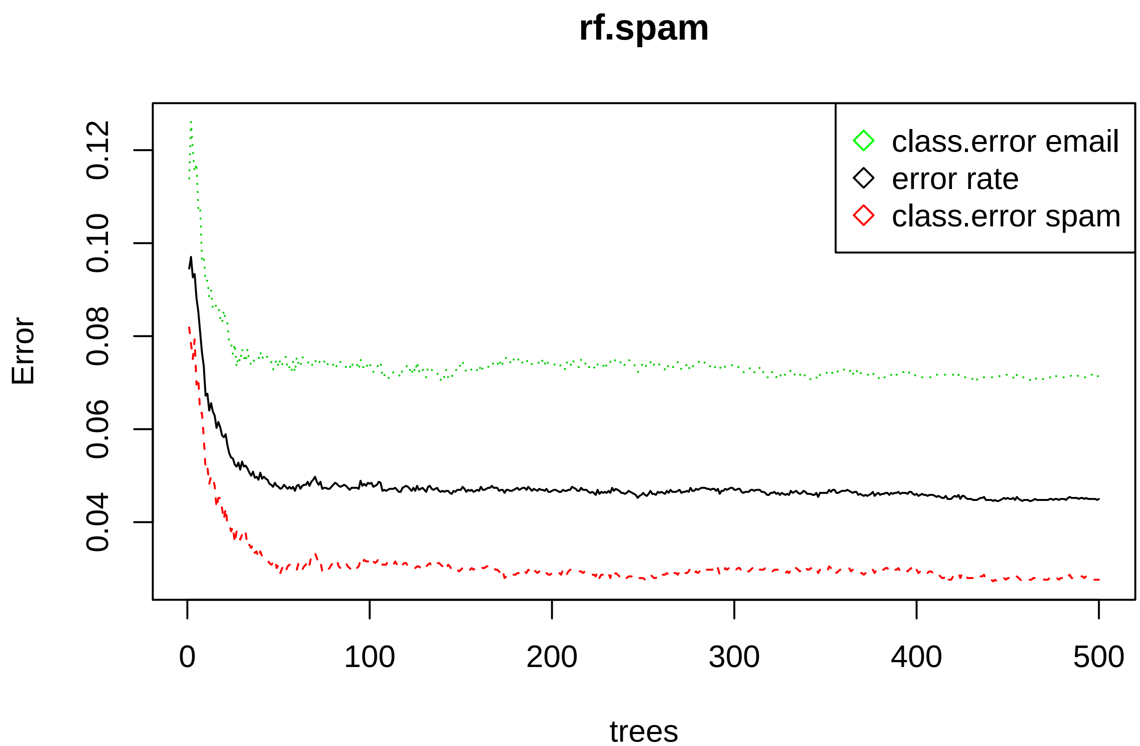
```
##          true
## predicted    1    2
##         1 2619  320
##         2  169 1493
```

c) Fit a random Forest with the default settings. (Use seed 123 in order to reproduce the solution). Be patient: this may take several seconds.

```r
library(randomForest)
set.seed(123)
rf.spam<-randomForest(spam~.,data=spam)
```

d) Plot the error rate vs. the number of fitted trees. How many trees are necessary? Refit the model with the chosen number of trees. How long does it take now? Have a look at the output. What error rate do you expect for new predictions (OOB error rate)? What is the error rate in the 'spam'-class?

```r
plot(rf.spam)
legend("topright",c("class.error email","error rate","class.error spam"),
    col=c("green","black","red"),pch=5)
```




```r
set.seed(123)
randomForest(spam~.,data=spam,importance=T,ntree=100) #See OOB estimates
```

```
##
## Call:
##  randomForest(formula = spam ~ ., data = spam, importance = T,      ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 4.52%
## Confusion matrix:
##       email spam class.error
## email  2710   78  0.02797704
## spam    130 1683  0.07170436
```

e) Suppose, we get a new email and want to predict the spam label. For simplicity, we refit the Random Forest on 2601 randomly chosen emails and save the remaining 2000 emails as test set. How does the OOB error compare with the error on the test set? (use ntree = 100, and set.seed = 123)

```r
set.seed(123)
fitSet<-sample(1:nrow(spam),2601)
set.seed(123)
rf3.spam<-randomForest(spam~.,data=spam[fitSet,],importance=T,ntree=100)
rf3.spam # OOB error rate:4.96%
```

```
##
## Call:
##  randomForest(formula = spam ~ ., data = spam[fitSet, ], importance = T,      ntree = 100)
##                Type of random forest: classification
```

```
##                        Number of trees: 100
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 4.96%
## Confusion matrix:
##       email spam class.error
## email  1548   44  0.02763819
## spam     85  924  0.08424182
```

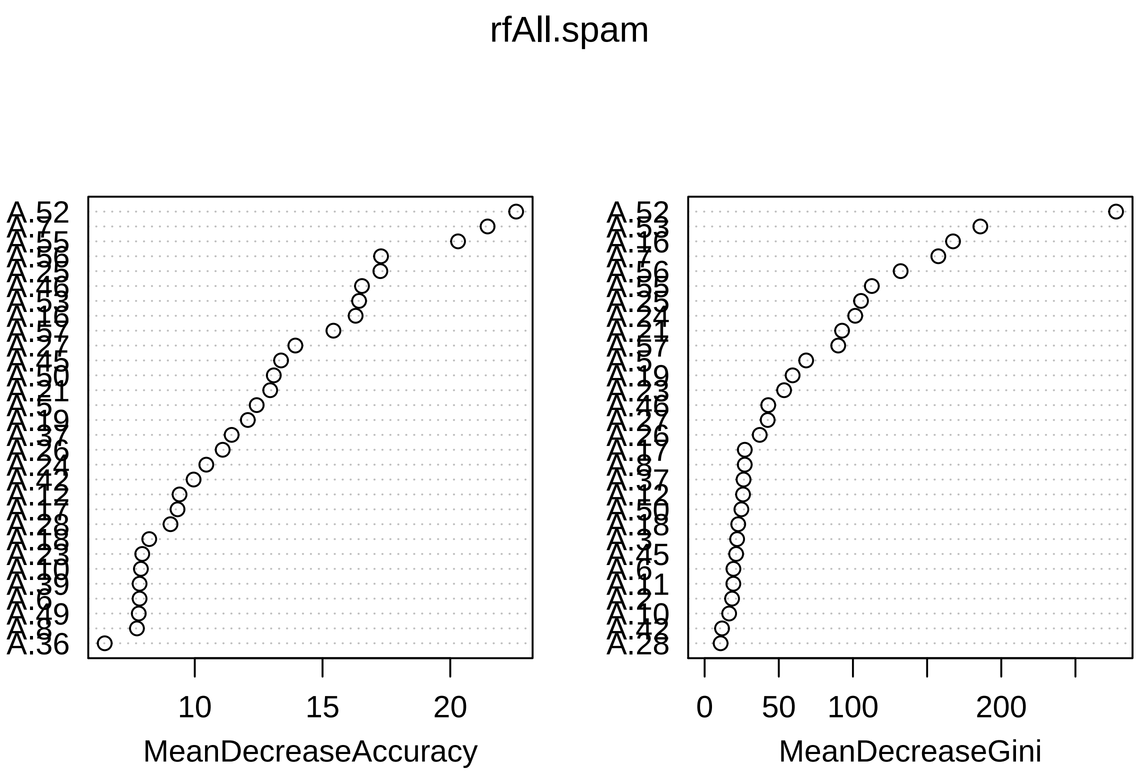
```r
pred<-predict(rf3.spam,newdata=spam[-fitSet,])
confusion(object=pred,true=spam[-fitSet,58])
```

```
##          true
## predicted email spam
##     email  1165   79
##     spam     31  725
```

f) Suppose we don't want to compute all variables for each new incoming mail, but only use the best 5. Which 5 variables should we choose? Compare the OOB error using all variables, the best 5 and the worst 5 (according to decrease in accuracy; use ntree = 100 and seed = 123).

```r
set.seed(123)
rfAll.spam<-randomForest(spam~.,data=spam,importance=T,ntree=100)

varImpPlot(rfAll.spam)
```




```r
importance(rfAll.spam)
```

```
##            email      spam MeanDecreaseAccuracy MeanDecreaseGini
## A.1    2.9153923 3.7172615             4.110481        9.9297727
## A.2    4.4524677 5.1152764             5.651437       18.4839588
```

```
## A.3   2.8424422  6.6919526             5.949023       21.8907532
## A.4   3.3355669  0.3902599             3.140267        1.5837023
## A.5   8.8506827 12.3516473            12.423526       68.4413809
## A.6   5.8840412  6.3416106             7.838628       19.4050886
## A.7  19.2856230 16.4575942            21.461694      157.5592715
## A.8   7.6682539  5.3614645             7.735778       27.0500202
## A.9   4.2177019  4.1111277             5.301587        9.4434507
## A.10  4.6099072  7.0310912             7.891455       16.5142773
## A.11  6.2279957  3.6202488             6.070576       19.3493391
## A.12  4.5786465  8.7790129             9.403398       25.8811570
## A.13  2.6410869  4.9957643             5.185319        8.8118162
## A.14  3.7532916  4.4363364             5.142657        4.7485668
## A.15  3.0108882  2.3997579             3.810204        2.3620173
## A.16 15.4659416 12.3740378            16.296030      167.4706849
## A.17  8.6010673  6.6008146             9.324523       27.0691813
## A.18  7.3436999  5.8406586             8.216818       22.6325570
## A.19  7.1060507 12.0222664            12.076010       59.3157527
## A.20  4.0684939  3.5530731             4.291483       10.1306610
## A.21  9.3824522 11.3328218            12.952131       92.6671719
## A.22  5.0529483  4.5674376             5.857561        6.9653391
## A.23  7.6299645  5.4512083             7.943342       53.5645920
## A.24  9.1125452  8.1168517            10.451127      101.5313567
## A.25 12.3723296 15.7652223            17.264008      105.4278940
## A.26  5.9958068 10.1499621            11.091323       37.1493299
## A.27  8.5974795 13.1672130            13.937549       42.4924078
## A.28  5.7103984  6.9428629             9.049371       10.7792290
## A.29  0.2164038  3.9614304             4.034284        2.8865216
## A.30  3.4621361  5.0899771             5.687904        6.2939240
## A.31  1.2312943  3.9758017             4.020393        3.9230821
## A.32  0.6198613  3.4556578             3.366126        1.6107848
## A.33  3.2016826  4.3344354             5.191401        5.7788080
## A.34  1.0827993  3.2230728             3.309442        2.2992775
## A.35  2.8889239  5.7406096             6.170489        6.9157937
## A.36  4.5448311  4.2390216             6.474240        7.2407657
## A.37  7.3336967 11.2908733            11.443116       26.2387975
## A.38 -0.4524552  2.4075808             1.752617        1.0429460
## A.39  2.8314985  7.2719421             7.839250        6.2612732
## A.40  3.1854459  1.6545890             3.728981        2.6174651
## A.41  0.3920695  4.3180442             4.594217        2.2217989
## A.42  7.0398343  8.7611080             9.953598       11.7394161
## A.43  2.5393220  4.5852328             4.960438        3.4719049
## A.44  1.9740509  4.7033516             5.383094        4.0421911
## A.45 10.5140729 10.5577919            13.377374       21.2462822
## A.46 11.1675253 15.5442885            16.545219       42.8860166
## A.47 -1.0050378  2.1857452             1.826781        0.4163933
## A.48  2.4375596  3.8255632             4.423217        2.5498125
## A.49  4.3875662  7.5668732             7.806714        9.3561579
## A.50  8.1185110 11.1862867            13.092245       24.7786765
## A.51  3.6636100  4.7391886             5.183238        5.6521260
## A.52 19.2065746 19.5935721            22.579589      277.4016488
## A.53 15.7360565 12.6693014            16.431508      185.8911569
## A.54  5.2848240  4.2142014             6.174244        7.4508385
## A.55 13.7915608 15.0494891            20.305324      112.7295835
## A.56 12.3369767 14.0182660            17.290458      132.1604182
```

```
## A.57 11.9857797  9.7731050            15.427016       90.0770229
```

```r
#verwende Criterium MeanDecreaseGini
decreaseGini<-(subset(importance(rfAll.spam),select="MeanDecreaseGini"))
sort(decreaseGini[,1])
```

```
##        A.47        A.38         A.4        A.32        A.41        A.34
##   0.4163933   1.0429460   1.5837023   1.6107848   2.2217989   2.2992775
##        A.15        A.48        A.40        A.29        A.43        A.31
##   2.3620173   2.5498125   2.6174651   2.8865216   3.4719049   3.9230821
##        A.44        A.14        A.51        A.33        A.39        A.30
##   4.0421911   4.7485668   5.6521260   5.7788080   6.2612732   6.2939240
##        A.35        A.22        A.36        A.54        A.13        A.49
##   6.9157937   6.9653391   7.2407657   7.4508385   8.8118162   9.3561579
##         A.9         A.1        A.20        A.28        A.42        A.10
##   9.4434507   9.9297727  10.1306610  10.7792290  11.7394161  16.5142773
##         A.2        A.11         A.6        A.45         A.3        A.18
##  18.4839588  19.3493391  19.4050886  21.2462822  21.8907532  22.6325570
##        A.50        A.12        A.37         A.8        A.17        A.26
##  24.7786765  25.8811570  26.2387975  27.0500202  27.0691813  37.1493299
##        A.27        A.46        A.23        A.19         A.5        A.57
##  42.4924078  42.8860166  53.5645920  59.3157527  68.4413809  90.0770229
##        A.21        A.24        A.25        A.55        A.56         A.7
##  92.6671719 101.5313567 105.4278940 112.7295835 132.1604182 157.5592715
##        A.16        A.53        A.52
## 167.4706849 185.8911569 277.4016488
```

```r
#Besten 5: 52,53,16,7,56
set.seed(123)
rfBest5.spam<-randomForest(spam~A.52+A.53+A.16+A.7+A.56,data=spam,importance=T,ntree=100)
rfBest5.spam
```

```
##
## Call:
##  randomForest(formula = spam ~ A.52 + A.53 + A.16 + A.7 + A.56,      data = spam, importance = T,
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 8.54%
## Confusion matrix:
##       email spam class.error
## email  2671  117  0.04196557
## spam    276 1537  0.15223387
```

```r
# OOB estimate of  error rate: 8.54%

#Schlechtesten 5: 47,38,4,32,41
set.seed(123)
rfWorst5.spam<-randomForest(spam~A.47+A.38+A.4+A.32+A.41,data=spam,importance=T,ntree=100)
rfWorst5.spam
```

```
##
## Call:
##  randomForest(formula = spam ~ A.47 + A.38 + A.4 + A.32 + A.41,      data = spam, importance = T,
##                Type of random forest: classification
```

```
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 38.38%
## Confusion matrix:
##       email spam class.error
## email  2781    7  0.00251076
## spam   1759   54  0.97021511
```

```
#OOB estimate of  error rate: 38.38%
```