Module 8: Solutions to Recommended Exercises

TMA4268 Statistical Learning V2020

Martina Hall, Michail Spitieris, Stefanie Muff, Department of Mathematical Sciences, NTNU

March 05, 2020

Solutions to Recommended Exercises

1. Theoretical

See Module 5 Recommended exercise solution for the bootstrap result.

2. Understanding

See solutions to exercise Q1 and Q4 here: https://rstudio-pubs-static.s3.amazonaws.com/65564_925dfde884e14ef9b5735eddd16c263e.html

See solutions to 2c-f: https://www.math.ntnu.no/emner/TMA4268/2019v/8Trees/TMA4268M8RecEx2ctof.pdf

3. Implementation:

a)

We have 4601 observations and 58 variables, 57 of them will be used as covariates.

```
library(kernlab)
`?`(spam)
```

b)

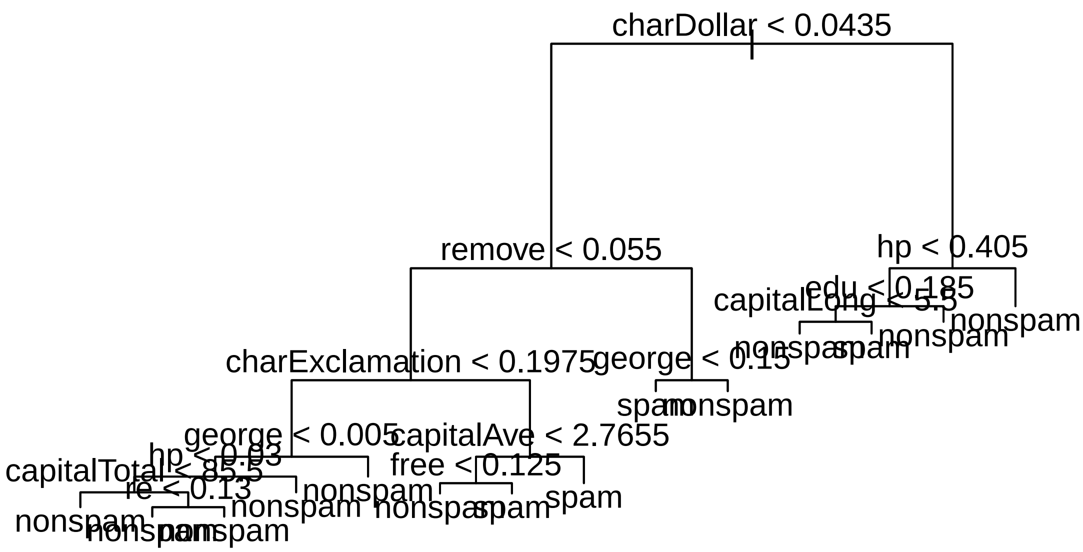
We use approximately 2/3 of the observations as the train set and 1/3 as the test set.

```
library(tree)

set.seed(1)

data(spam)
N = dim(spam)[1]

train = sample(1:N, 3000)
test = (1:N)[-train]
```

c)

We fit a classification tree to the training data.

```r
tree.spam = tree(type ~ ., spam, subset = train)

plot(tree.spam)

text(tree.spam, pretty = 1)
```



For this training set we have 11 terminal nodes.

```r
summary(tree.spam)
```

```
##
## Classification tree:
## tree(formula = type ~ ., data = spam, subset = train)
## Variables actually used in tree construction:
##  [1] "charDollar"      "remove"          "charExclamation"
##  [4] "george"          "hp"              "capitalTotal"
##  [7] "re"              "capitalAve"      "free"
## [10] "edu"             "capitalLong"
## Number of terminal nodes:  14
## Residual mean deviance:  0.4663 = 1392 / 2986
## Misclassification error rate: 0.08533 = 256 / 3000
```

d)

We predict the response for the test data.

```r
yhat = predict(tree.spam, spam[test, ], type = "class")
response.test = spam$type[test]
```

and make a confusion table:

```r
misclass = table(yhat, response.test)
print(misclass)
```

```
##           response.test
```

```
## yhat      nonspam spam
##   nonspam     950   99
##   spam         47  505
```

The misclassification rate is given by:

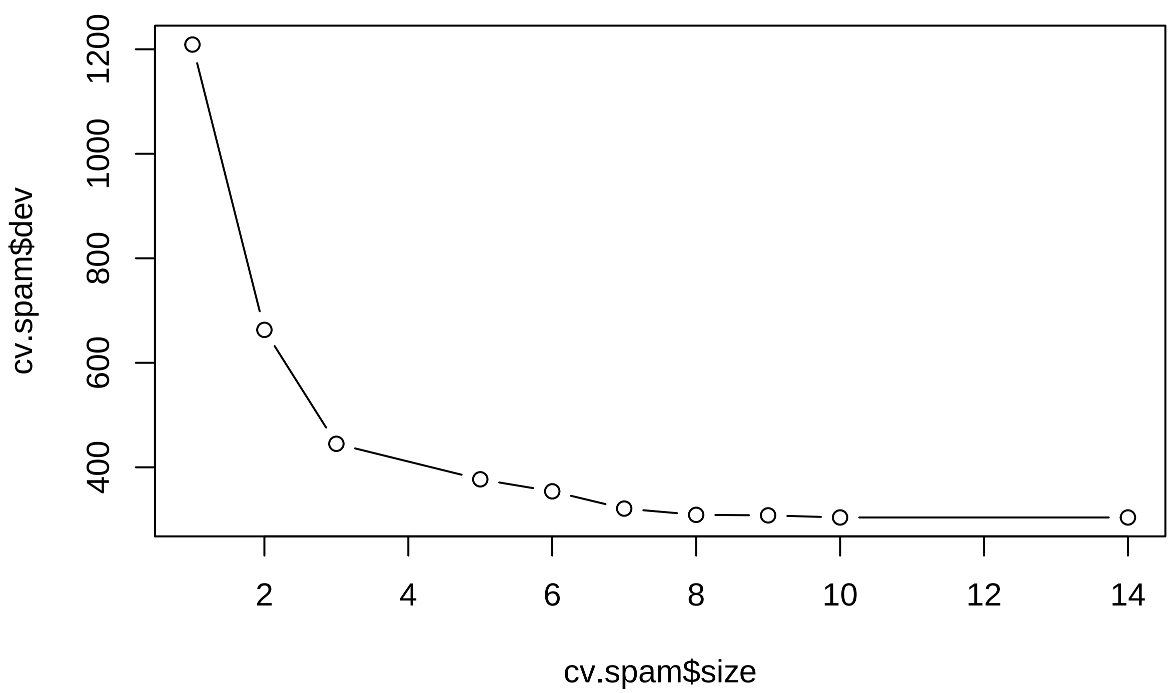
```r
1 - sum(diag(misclass))/sum(misclass)
```

```
## [1] 0.091193
```

e)

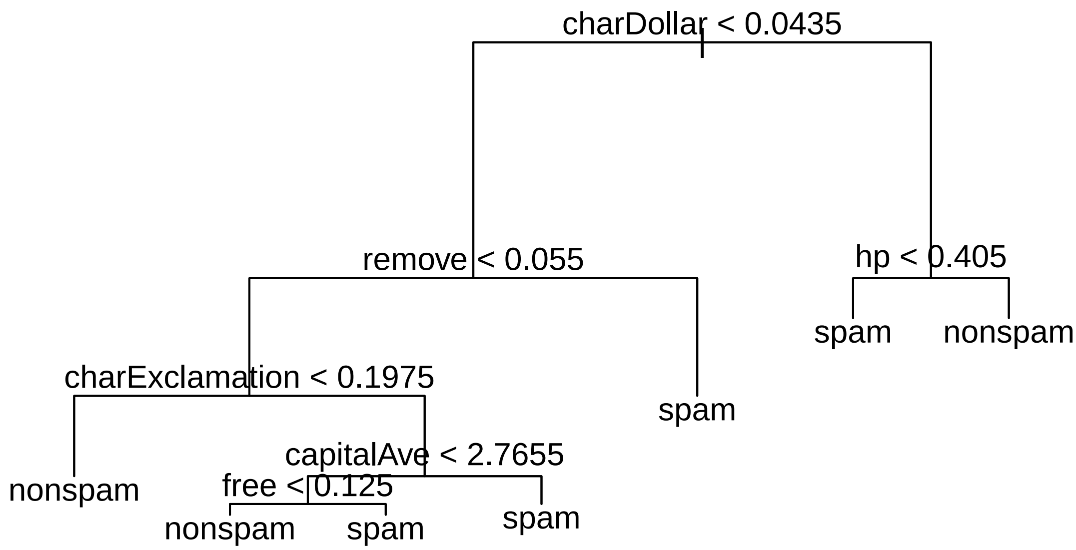
We use *cv.tree()* to find the optimal tree size.

```r
set.seed(1)

cv.spam = cv.tree(tree.spam, FUN = prune.misclass)

plot(cv.spam$size, cv.spam$dev, type = "b")
```



According to the plot the optimal number of terminal nodes is 7 (or larger). We choose 7 as this gives the simplest tree, and prune the tree according to this value.

```r
prune.spam = prune.misclass(tree.spam, best = 7)

plot(prune.spam)
text(prune.spam, pretty = 1)
```



We predict the response for the test data:

```
yhat.prune = predict(prune.spam, spam[test, ], type = "class")

misclass.prune = table(yhat.prune, response.test)
print(misclass.prune)
```

```
##            response.test
## yhat.prune nonspam spam
##    nonspam     937   94
##    spam         60  510
```

The misclassification rate is

```
1 - sum(diag(misclass.prune))/sum(misclass.prune)
```

```
## [1] 0.09618988
```

f)

We create a decision tree by bagging.

```
library(randomForest)
bag.spam = randomForest(type ~ ., data = spam, subset = train, mtry = dim(spam)[2] -
    1, ntree = 500, importance = TRUE)
```

We predict the response for the test data as before:

```
yhat.bag = predict(bag.spam, newdata = spam[test, ])

misclass.bag = table(yhat.bag, response.test)
print(misclass.bag)
```

```
##           response.test
## yhat.bag  nonspam spam
##   nonspam     964   53
##   spam         33  551
```

The misclassification rate is

```r
1 - sum(diag(misclass.bag))/sum(misclass.bag)
```

```
## [1] 0.05371643
```

g)

We now use the random forest-algorithm and consider only $\sqrt{57} \approx 8$ of the predictors at each split. This is specified in *mtry*.

```r
set.seed(1)

rf.spam = randomForest(type ~ ., data = spam, subset = train, mtry = round(sqrt(dim(spam)[2] -
    1)), ntree = 500, importance = TRUE)
```

We study the importance of each variable

```r
importance(rf.spam)
```

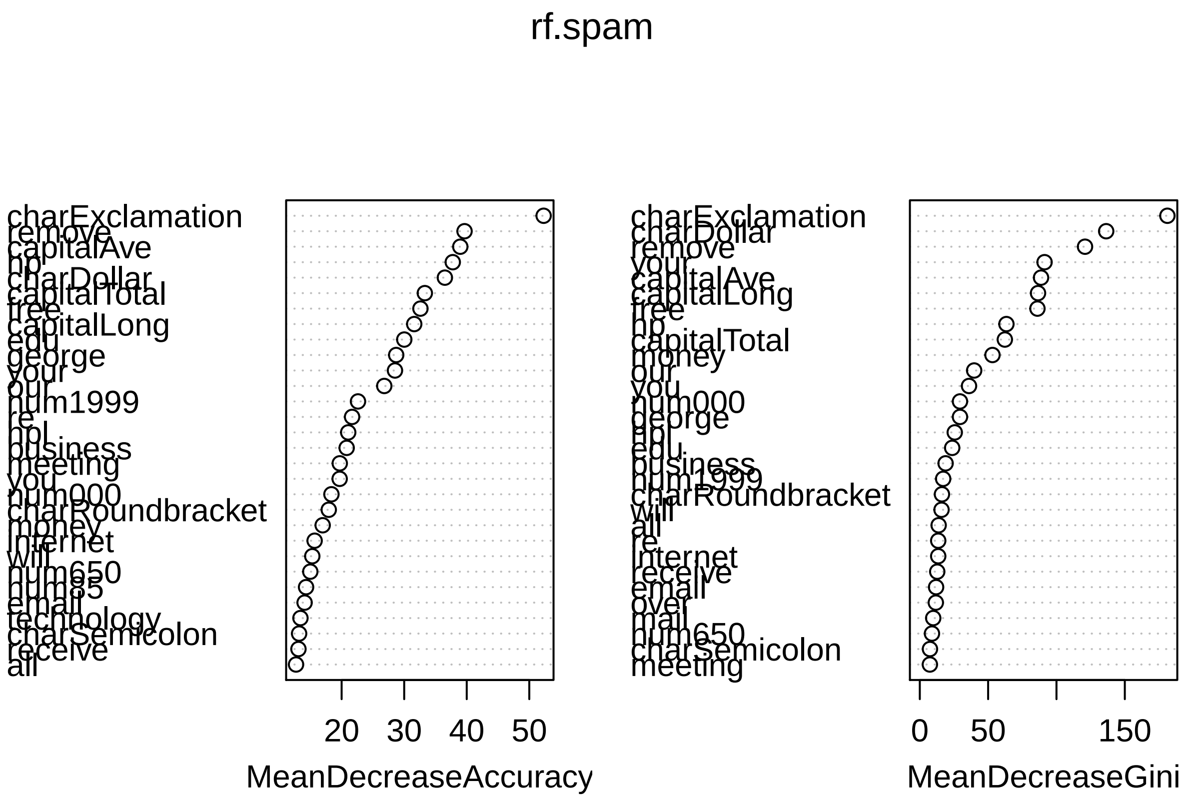
```
##                      nonspam        spam MeanDecreaseAccuracy
## make               4.9152911  5.41745971            7.1272184
## address            6.6082347  6.39616947            8.6424650
## all                3.7623871 13.38409051           12.7096561
## num3d              6.1486863  1.38525613            5.5227168
## our               21.8589103 21.08883211           26.8137166
## over              11.8436173  8.50780383           12.2836325
## remove            37.3651553 30.12285958           39.6505623
## internet          14.4469859  9.00299692           15.6829228
## order              8.5231174  6.18906573           10.2290415
## mail               8.0660481  6.94139036           10.4527384
## receive           13.4944484  4.02347567           13.0949560
## will               6.1608437 15.48489404           15.3052134
## people             2.6359362  7.53309843            7.4218176
## report             5.0999952  9.98477933            9.2494181
## addresses          7.2873994  3.80203954            8.1415793
## free              28.7491646 23.95906339           32.5970656
## business          19.2327228 12.90476732           20.8057002
## email             12.3733993  8.64261776           14.0801341
## you               13.5037854 16.81113609           19.6843500
## credit            10.2773077  5.99062255           11.1953169
## your              20.0800416 24.85034151           28.5272746
## font               9.3669641  2.00146290            9.6523743
## num000            17.2922196  8.63339294           18.3668304
## money             14.6183673 13.88434979           16.9681313
## hp                25.2535721 34.75362191           37.7646254
## hpl               11.6069451 19.86963854           21.0505307
## george            19.0778732 25.50570085           28.7211247
## num650            10.6824826 11.90991651           14.9856992
## lab                0.3372010  9.06855329            9.1866249
## labs               4.6636729  9.73641004           10.8304912
## telnet             4.0505041  7.10454447            7.5211077
## num857             2.5340239  6.32819636            6.4496282
## data               4.0702200  7.75052392            8.1453195
## num415             4.1024106  6.55080792            7.3255371
## num85              7.7476476 12.82545808           14.3156126
```

```
## technology        10.8892593  7.95737246           13.4126520
## num1999           13.7347159 21.32978237           22.6136471
## parts              0.4852073  3.68838179            3.2720179
## pm                 2.7691505 10.29380049            9.8360477
## direct             7.1609636  0.07398713            7.5530075
## cs                 2.4566729  7.95438516            8.0647661
## meeting            9.7817022 18.90953421           19.7000176
## original           0.7496996 11.94326459           11.5248576
## project            1.4166917  8.07075380            8.2328867
## re                14.2508608 18.46965458           21.6704090
## edu               22.4647630 27.14157538           30.0117662
## table             -0.8291370  1.63657439            0.7965145
## conference         4.7445783  6.49331553            7.4547271
## charSemicolon      9.3176571 10.51718113           13.2015714
## charRoundbracket   7.3846811 17.70749039           17.9388820
## charSquarebracket  7.8770475  6.93249127            9.8952485
## charExclamation   37.5196716 42.97521675           52.2951665
## charDollar        31.8878152 27.86857931           36.5023715
## charHash           8.7386799  6.51787572           11.0979178
## capitalAve        28.8554125 28.56300770           38.9684741
## capitalLong       24.7708567 22.38827384           31.6011983
## capitalTotal      26.5759426 20.31122421           33.2994667
##                   MeanDecreaseGini
## make                     5.1155425
## address                  6.4881500
## all                     13.7961289
## num3d                    1.5160261
## our                     39.7705074
## over                    11.7300966
## remove                 120.8819930
## internet                13.4770211
## order                    5.3904925
## mail                     9.8257255
## receive                 12.7626807
## will                    15.8940426
## people                   5.7833772
## report                   3.4244233
## addresses                1.7442492
## free                    86.0756131
## business                18.8819051
## email                   11.9289880
## you                     36.0055700
## credit                   6.4540848
## your                    91.2810658
## font                     2.8332895
## num000                  29.4069872
## money                   53.1738233
## hp                      63.3746313
## hpl                     25.5778403
## george                  29.4020696
## num650                   8.9036821
## lab                      2.5379545
## labs                     4.9613153
## telnet                   2.1317091
```

```
## num857                    1.1347365
## data                      3.6723269
## num415                    1.2262174
## num85                     5.3136245
## technology                4.7331045
## num1999                  17.0965842
## parts                     0.6302254
## pm                        3.5348141
## direct                    1.5445946
## cs                        1.6667927
## meeting                   7.3758323
## original                  2.7642920
## project                   2.8664311
## re                       13.4826965
## edu                      23.7037302
## table                     0.3192404
## conference                1.8178769
## charSemicolon             7.4326828
## charRoundbracket         16.1982376
## charSquarebracket         3.2856362
## charExclamation         181.1496058
## charDollar              136.3648818
## charHash                  5.3325230
## capitalAve               88.6994807
## capitalLong              86.5005735
## capitalTotal             62.2243211
```

If *MeanDecreaseAccuracy* and *MeanDecreaseGini* are large, the corresponding covariate is important.

```
varImpPlot(rf.spam)
```



In this plot we see that *charExclamation* is the most important covariate, followed by *remove* and *charDollar*. This is as expected as these variables are used in the top splits in the classification trees we have seen so far.

We now predict the response for the test data.

```
yhat.rf = predict(rf.spam, newdata = spam[test, ])

misclass.rf = table(yhat.rf, response.test)
1 - sum(diag(misclass.rf))/sum(misclass.rf)
```

```
## [1] 0.04871955
```

The misclassification rate is given by

```
print(misclass.rf)
```

```
##           response.test
## yhat.rf   nonspam spam
##   nonspam     965   46
##   spam         32  558
```

h)

Finally, we create a tree by using the boosting algorithm. The *gbm()* function does not allow factors in the response, so we have to use "1" and "0" instead of "spam" and "nonspam":

```
library(gbm)
set.seed(1)

spamboost = spam
spamboost$type = c()
```

```r
spamboost$type[spam$type == "spam"] = 1
spamboost$type[spam$type == "nonspam"] = 0

boost.spam = gbm(type ~ ., data = spamboost[train, ], distribution = "bernoulli",
    n.trees = 5000, interaction.depth = 4, shrinkage = 0.001)
```

We predict the response for the test data:

```r
yhat.boost = predict(boost.spam, newdata = spamboost[-train, ], n.trees = 5000,
    distribution = "bernoulli", type = "response")

yhat.boost = ifelse(yhat.boost > 0.5, 1, 0)  #Transform to 0 and 1 (nonspam and spam).

misclass.boost = table(yhat.boost, spamboost$type[test])

print(misclass.boost)
```

```
##
## yhat.boost   0   1
##          0 960  62
##          1  37 542
```

The misclassification rate is

```r
1 - sum(diag(misclass.boost))/sum(misclass.boost)
```

```
## [1] 0.06183635
```

i)

We get lower misclassification rates for bagging, boosting and random forest as expected.

Solutions to Compulsory Exercises 3, 2018

Problem 1 - Classification with trees: https://www.math.ntnu.no/emner/TMA4268/2018v/CompEx/Compulsory3solutions.html

Solutions to Exam question 2018 Problem 4

Classification of diabetes cases c), with Q20, Q21, Q22.

https://www.math.ntnu.no/emner/TMA4268/Exam/e2018sol.pdf