

Module 8: Solutions to Recommended Exercises

TMA4268 Statistical Learning V2020

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

March 05, 2020

Problem 1 – Theoretical

- a) Provide a detailed explanation of the algorithm that is used to fit a regression tree. What is different for a classification tree?
1. *Recursive binary splitting*: We find the best single partitioning of the data such that the reduction of RSS is the greatest. This process is applied sequentially to each of the split parts until a predefined minimum number of leave observation is reached.
 2. *Cost complexity pruning* of the large tree from previous step, in order to obtain a sequence of best trees as a function of a parameter α . Each value of α corresponds to a subtree that minimize the following equation (several α s for the same tree):

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|,$$

where $|T|$ is the number of terminal nodes.

3. *K-fold cross-validation* to choose α . For each fold:
 - Repeat Steps 1 and 2 on all but the kth folds of the training data.
 - Evaluate the mean squared prediction on the data in the left-out kth fold, as a function of α .
 - Average the results for each value of α and choose α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α .

For a **classification** tree we replace RSS with Gini index or entropy.

- b) What are the advantages and disadvantages of regression and classification trees?

Advantages

- Very easy to explain
- Can be displayed graphically
- Can handle both quantitative and qualitative predictors without the need to create dummy variables

Disadvantages

- The predictive accuracy is usually not very high
- They are non-robust. That is a small change in the data can cause a large change in the estimated tree

- c) What is the idea behind bagging and what is the role of bootstrap? How do random forests improve that idea?

Decision trees suffer from high variance. Recall that if we have B *i.i.d* observations of a random variable X with the same mean and variance σ^2 . We calculate the mean $\bar{X} = \frac{1}{B} \sum_{b=1}^B X_b$, and the variance of the mean is $\text{Var}(\bar{X}) = \frac{\sigma^2}{B}$. That is by averaging we get reduced variance.

For decision trees, if we have B training sets, we could estimate $\hat{f}_1(\mathbf{x}), \hat{f}_2(\mathbf{x}), \dots, \hat{f}_B(\mathbf{x})$ and average them as

$$\hat{f}_{avg}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^B \hat{f}_b(\mathbf{x}) .$$

However we do not have many data independent data sets, and we bootstrap to create B datasets. These datasets are however not completely independent and the reduction in variance is therefore not as large as for independent training sets.

To make the different trees that are built from each bootstrapped dataset more different, random forests use a random subset of the predictors to split the tree into new branches at each step. This decorrelates the different trees that are built from the B bootstrapped datasets, and consequently reduces variance.

- d) What is an out-of bag (OOB) error estimator and what percentage of observations are included in an OOB sample? (Hint: The result from RecEx5-Problem 4c can be used)

An OOB is the set of observations that were not chosen to be in a specific bootstrap sample. From RecEx5-Problem 4c we have that on average $1 - 0.632 = 0.368$ are included in the OOB sample.

- e) Bagging and Random Forests typically improve the prediction accuracy of a single tree, but it can be difficult to interpret, for example in terms of understanding which predictors are how relevant. How can we evaluate the importance of the different predictors for these methods?

Variable importance based on node impurity

Regression Trees: The total amount that the RSS is decreased due to splits of each predictor, averaged over the B trees.

Classification Trees: The importance is the mean decrease (over all B trees) in the Gini index by splits of a predictor.

Variable importance based on randomization

This measure is based on how much the predictive accuracy (MSE or gini index) is decreased when the variable is replaced by a permuted version of it. You find a drawing [here](#).

Problem 2 – Regression (Book Ex. 8)

In the lab, a classification tree was applied to the Carseats data set after converting the variable **Sales** into a qualitative response variable. Now we will seek to predict **Sales** using regression trees and related approaches, treating the response as a quantitative variable.

- a) Split the data set into a training set and a test set. (Hint: Use 70% of the data as training set and the rest 30% as testing set)

```
library(ISLR)
data("Carseats")
set.seed(4268)
n = nrow(Carseats)
train = sample(1:n, 0.7 * nrow(Carseats), replace = F)
test = (1:n)[-train]
```

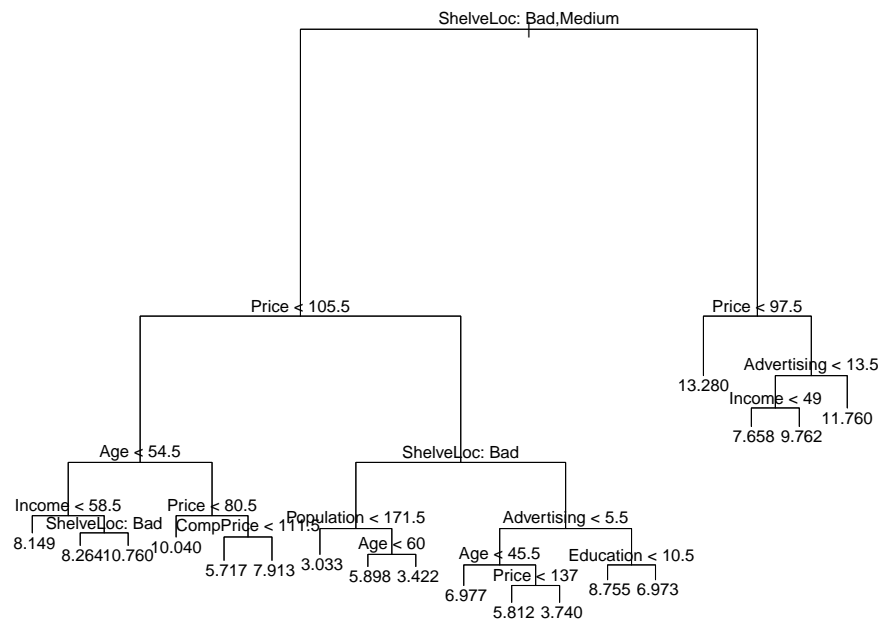
```
Carseats.train = Carseats[train, ]
Carseats.test = Carseats[-train, ]
```

- b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

```
library(tree)
tree.mod = tree(Sales ~ ., Carseats, subset = train)
summary(tree.mod)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = Carseats, subset = train)
## Variables actually used in tree construction:
## [1] "ShelveLoc" "Price" "Age" "Income" "CompPrice"
## [6] "Population" "Advertising" "Education"
## Number of terminal nodes: 18
## Residual mean deviance: 2.609 = 683.6 / 262
## Distribution of residuals:
## Min. 1st Qu. Median Mean 3rd Qu. Max.
## -3.74000 -1.12400 -0.06522 0.00000 1.06800 4.47200

plot(tree.mod)
text(tree.mod, pretty = 0)
```



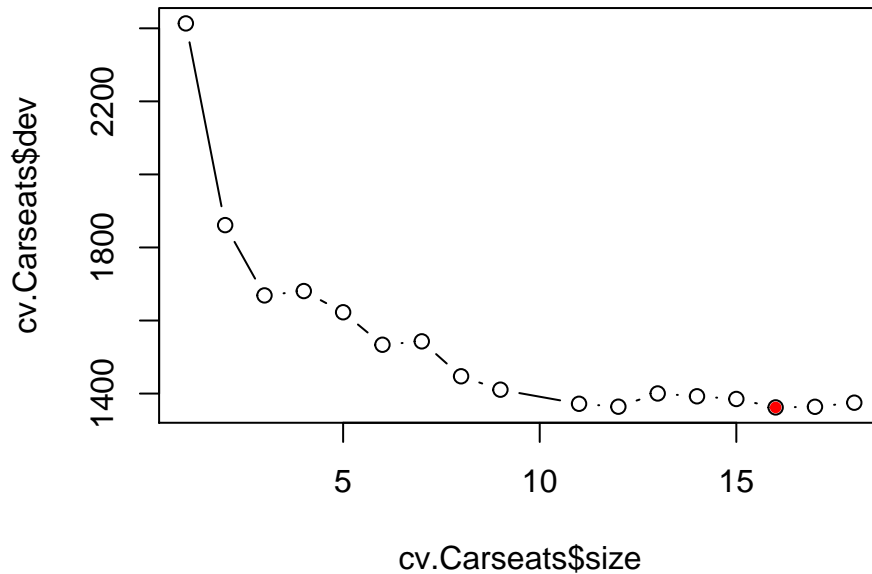
```
yhat = predict(tree.mod, newdata = Carseats.test)
mse = mean((yhat - Carseats.test$Sales)^2)
mse
```

```
## [1] 4.585249
```

- c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

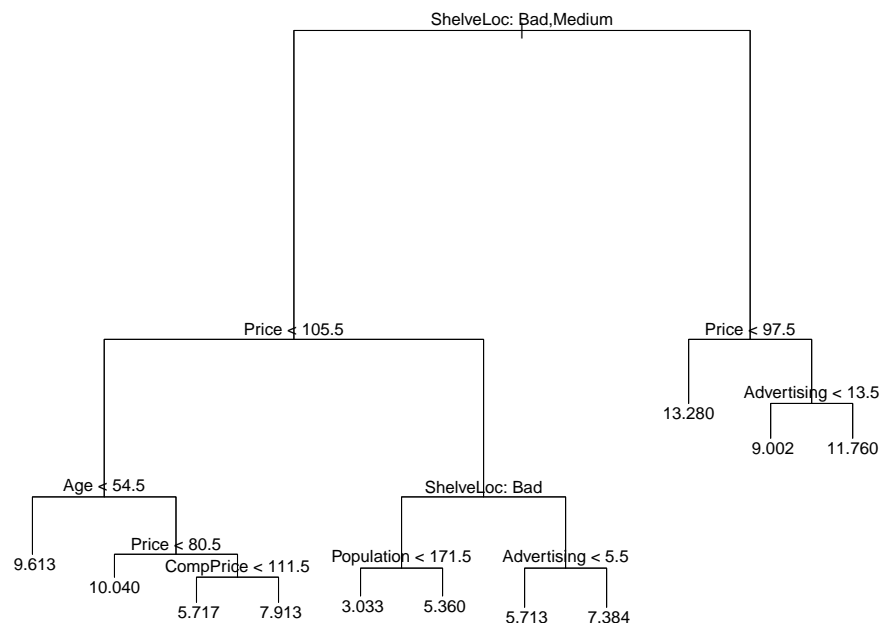
```
set.seed(4268)
cv.Carseats = cv.tree(tree.mod)
tree.min = which.min(cv.Carseats$dev)
```

```
best = cv.Carseats$size[tree.min]
plot(cv.Carseats$size, cv.Carseats$dev, type = "b")
points(cv.Carseats$size[tree.min], cv.Carseats$dev[tree.min], col = "red", pch = 20)
```



We see that trees with sizes 11, 12, 16 and 17 have similar deviance values. We might choose the tree of size 11 as it gives the simpler tree.

```
pr.tree = prune.tree(tree.mod, best = 11)
plot(pr.tree)
text(pr.tree, pretty = 0)
```



```
yhat = predict(pr.tree, newdata = Carseats.test)
mse = mean((yhat - Carseats.test$Sales)^2)
mse
```

```
## [1] 4.378499
```

There is a slight reduction in MSE for the pruned tree with 11 leaves.

- d) Use the bagging approach with 500 trees in order to analyze the data. What test MSE do you obtain? Use the `importance()` function to determine which variables are most important.

```
library(randomForest)
dim(Carseats)
```

```
## [1] 400 11
```

```
bag.Carseats = randomForest(Sales ~ ., Carseats.train, mtry = ncol(Carseats) -
  1, ntree = 500, importance = TRUE)
yhat.bag = predict(bag.Carseats, newdata = Carseats.test)
mse.bag = mean((yhat.bag - Carseats.test$Sales)^2)
mse.bag
```

```
## [1] 2.122958
```

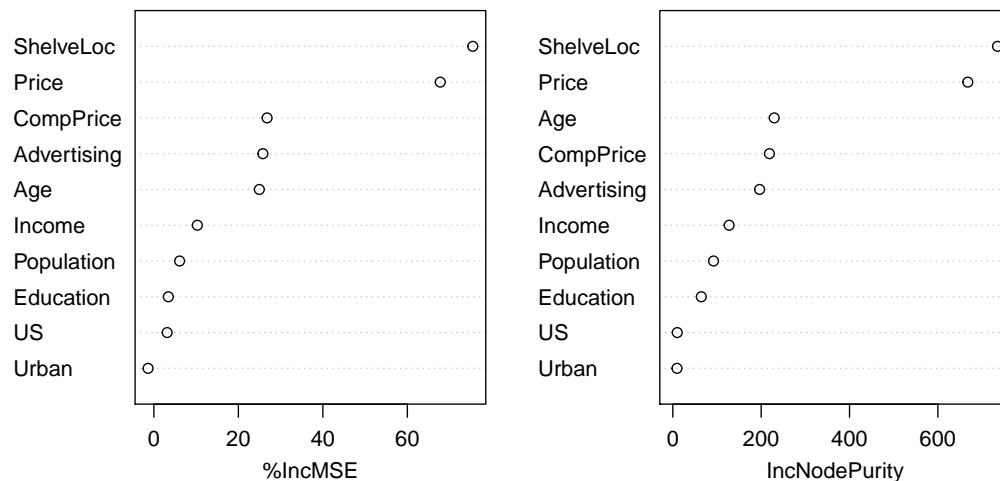
Bagging decreases the test MSE significantly to 2.12. From the importance plots we might conclude that Price and ShelfLoc are the most important Variables.

```
importance(bag.Carseats)
```

```
##           %IncMSE  IncNodePurity
## CompPrice    26.803869    218.740455
## Income       10.284817    127.447480
## Advertising  25.795425    196.438893
## Population    6.084270     92.149065
## Price        67.791459    667.696518
## ShelfLoc     75.485534    734.902022
## Age          24.961130    229.491494
## Education     3.423565     64.510742
## Urban        -1.373635      9.423406
## US           3.141449     10.105870
```

```
varImpPlot(bag.Carseats)
```

bag.Carseats



- e) Use random forests and to analyze the data. Include 500 trees and select 3 variables for each split.

What test MSE do you obtain? Use the `importance()` function to determine which variables are most important. Describe the effect of m , the number of variables considered at each split, on the error rate obtained.

```
rf.Carseats = randomForest(Sales ~ ., data = Carseats.train, mtry = 3, ntree = 500,
  importance = TRUE)
yhat.rf = predict(rf.Carseats, newdata = Carseats.test)
mse_forest <- mean((yhat.rf - Carseats.test$Sales)^2)
mse_forest
```

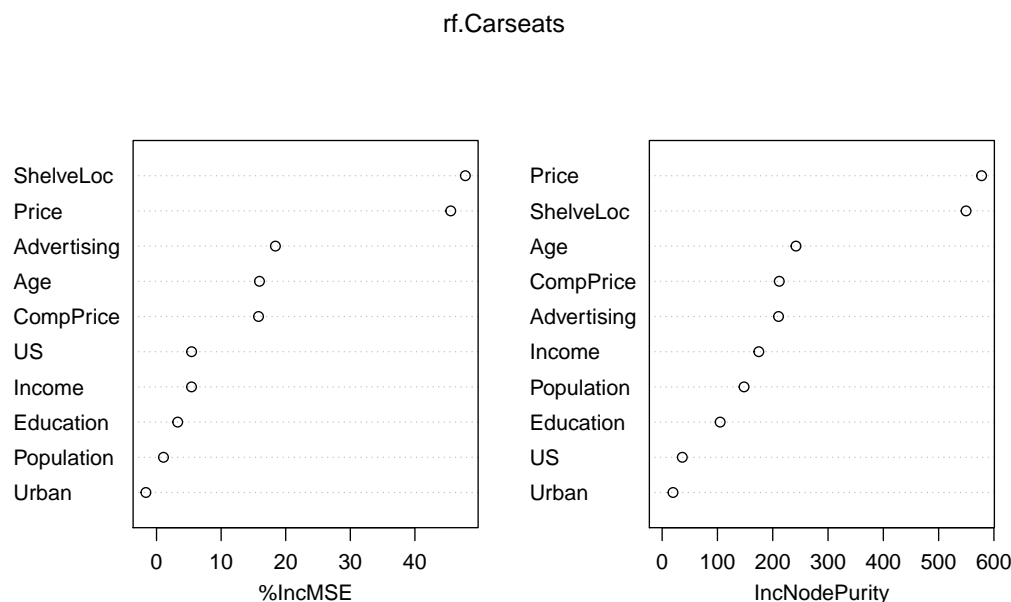
```
## [1] 2.25397
```

We use $\sqrt{p} = \sqrt{10} \approx 3$ trees, and we obtain an MSE of 2.25 which is slightly larger than Bagging MSE. The two most important Variables are again Price and ShelfLoc.

```
importance(rf.Carseats)
```

```
##           %IncMSE IncNodePurity
## CompPrice 15.789484    211.79213
## Income    5.415374    174.79625
## Advertising 18.402600    210.47149
## Population 1.076874    148.09993
## Price     45.548596    577.68865
## ShelfLoc  47.810006    549.62278
## Age       15.936114    241.99130
## Education  3.275725    104.89503
## Urban     -1.646580     19.63668
## US        5.427599     36.45647
```

```
varImpPlot(rf.Carseats)
```



f) Finally use boosting with 500 trees and an interaction depth $d = 4$ on our data. Compare the MSE to all other methods.

```
library(gbm)
r.boost = gbm(Sales ~ ., Carseats.train, distribution = "gaussian", n.trees = 500,
  interaction.depth = 4)
yhat.boost = predict(r.boost, newdata = Carseats.test, n.trees = 500)
```

```
mse_boost <- mean((yhat.boost - Carseats.test$Sales)^2)
mse_boost
```

```
## [1] 2.151292
```

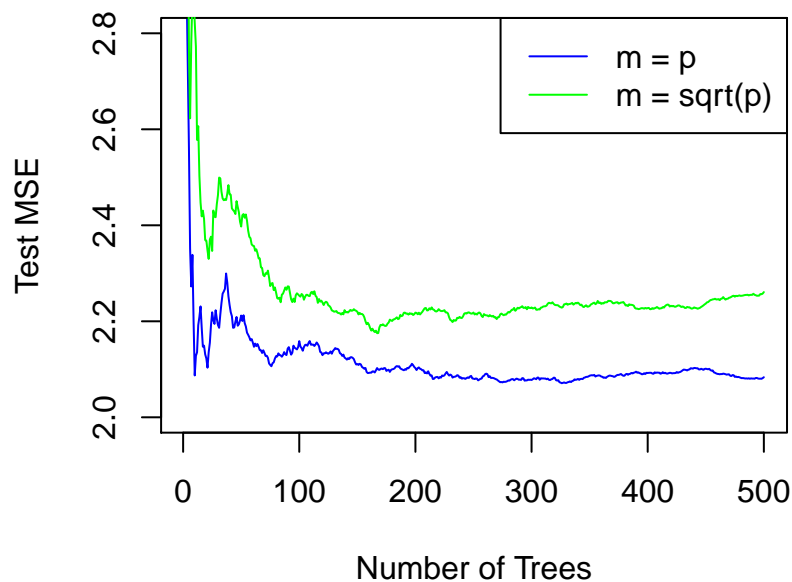
We see a further decrease in MSE by boosting our trees.

- g) What is the effect of the number of trees (`ntree`) on the test error? Plot the test MSE as a function of `ntree` for both the bagging and the random forest method.

```
train.predictors = Carseats.train[, -1]
test.predictors = Carseats.test[, -1]
Y.train = Carseats.train[, 1]
Y.test = Carseats.test[, 1]

bag.Car = randomForest(train.predictors, y = Y.train, xtest = test.predictors,
  ytest = Y.test, mtry = 10, ntree = 500)
rf.Car = randomForest(train.predictors, y = Y.train, xtest = test.predictors,
  ytest = Y.test, mtry = 3, ntree = 500)
plot(1:500, bag.Car$test$mse, col = "blue", type = "l", xlab = "Number of Trees",
  ylab = "Test MSE", ylim = c(2, 2.8))
lines(1:500, rf.Car$test$mse, col = "green")

legend("topright", c("m = p", "m = sqrt(p)"), col = c("blue", "green"), cex = 1,
  lty = 1)
```



Problem 3 – Classification

In this exercise you are going to implement a spam filter for e-mails by using tree-based methods. Data from 4601 e-mails are collected and can be uploaded from the kernlab library as follows:

```
library(kernlab)
data(spam)
```

Each e-mail is classified by `type` (`spam` or `nonspam`), and this will be the response in our model. In addition there are 57 predictors in the dataset. The predictors describe the frequency of different words in the e-mails

and orthography (capitalization, spelling, punctuation and so on).

a) Study the dataset by writing `?spam` in R.

```
?(spam)
```

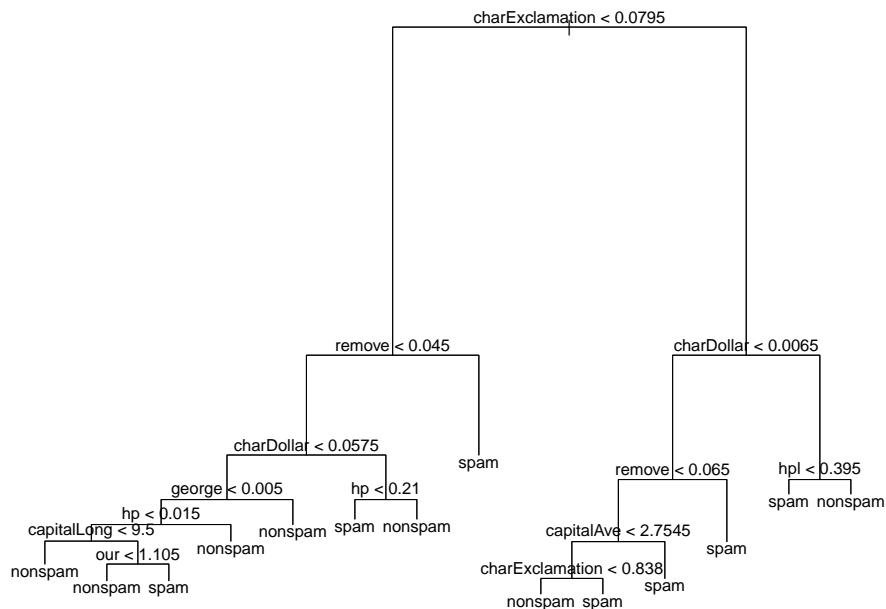
b) Create a training set and a test set for the dataset. (Hint: Use 70% of the data as training set and the rest 30% as testing set)

```
library(ISLR)
set.seed(4268)
n = nrow(spam)
train = sample(1:n, 0.7 * n, replace = F)
test = (1:n)[-train]
spam.train = spam[train, ]
spam.test = spam[-train, ]
```

c) Fit a tree to the training data with `type` as the response and the rest of the variables as predictors. Study the results by using the `summary()` function. Also create a plot of the tree. How many terminal nodes does it have?

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

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



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

```
##
## Classification tree:
## tree(formula = type ~ ., data = spam, subset = train)
## Variables actually used in tree construction:
## [1] "charExclamation" "remove" "charDollar" "george"
## [5] "hp" "capitalLong" "our" "capitalAve"
## [9] "hpl"
## Number of terminal nodes: 14
## Residual mean deviance: 0.4801 = 1539 / 3206
```



```
## Misclassification error rate: 0.08975 = 289 / 3220
```

d) Predict the response on the test data. What is the misclassification rate?

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

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

```
##           response.test
## yhat      nonspam spam
## nonspam    781   67
## spam       67  466
```

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

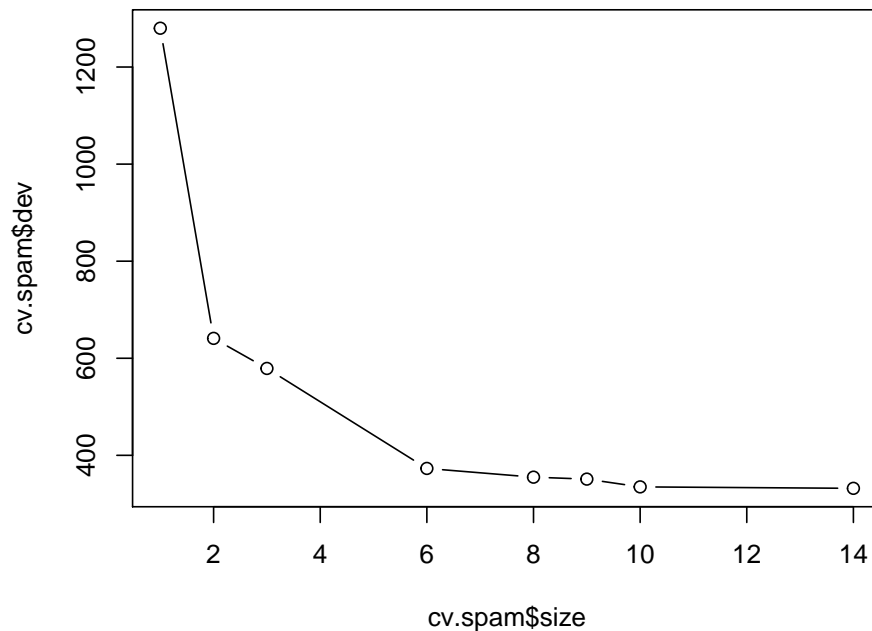
```
## [1] 0.09703114
```

e) Use the `cv.tree()` function to find the optimal tree size. Prune the tree according to the optimal tree size by using the `prune.misclass()` function and plot the result. Predict the response on the test data by using the pruned tree. What is the misclassification rate in this case?

```
set.seed(4268)
```

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

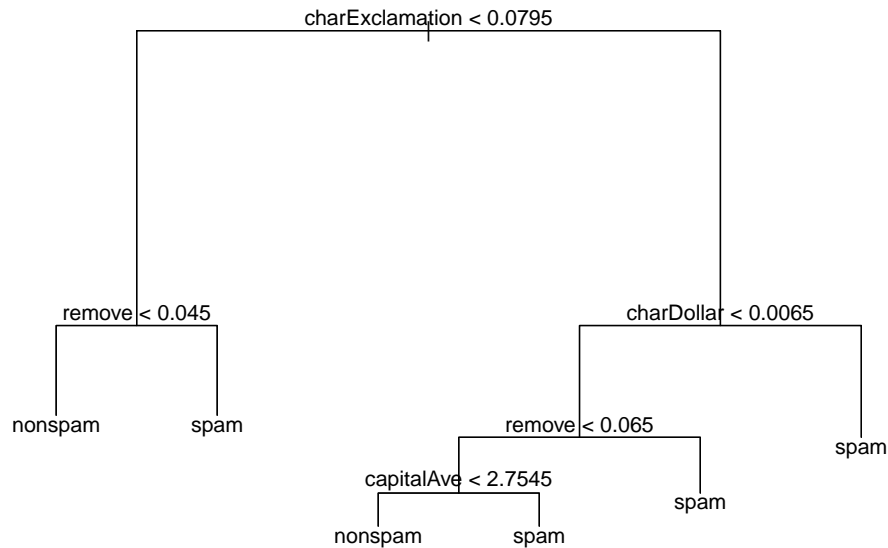
```
plot(cv.spam$size, cv.spam$dev, type = "b")
```



According to the plot the optimal number of terminal nodes is 6 (or larger). We choose 6 as this gives the simplest tree, and prune the tree according to this value.

```
prune.spam = prune.misclass(spam.tree, best = 6)
```

```
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)
misclass.prune

```

```

##           response.test
## yhat.prune nonspam spam
##   nonspam    796  104
##   spam       52  429

```

The misclassification rate is

```

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

## [1] 0.1129616

```

- f) Create a decision tree by using the bagging approach. Use the function `randomForest()` and consider all of the predictors in each split. Predict the response on the test data and report the misclassification rate.

```

library(randomForest)
bag.spam = randomForest(type ~ ., data = spam, subset = train, mtry = ncol(spam) -
  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)
misclass.bag

```

```

##           response.test
## yhat.bag  nonspam spam
##   nonspam    810  43
##   spam       38  490

```

The misclassification rate is

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

```
## [1] 0.05865315
```

- g) Apply the `randomForest()` function again, but this time consider only a subset of the predictors in each split. This corresponds to the random forest-algorithm. Study the importance of each variable by using the function `importance()`. Are the results as expected based on earlier results? Again, predict the response for the test data and report the misclassification rate.

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`.

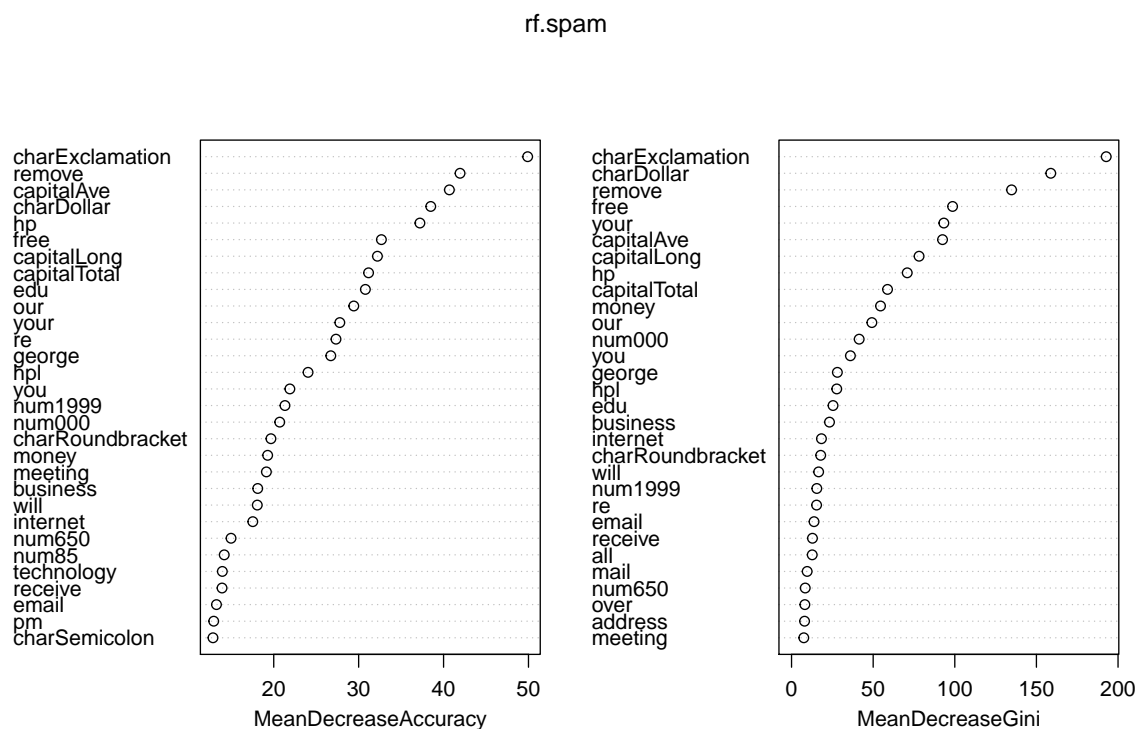
```
set.seed(4268)
rf.spam = randomForest(type ~ ., data = spam, subset = train, mtry = round(sqrt(ncol(spam) - 1)), ntree = 500, importance = TRUE)
```

We study the importance of each variable

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

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] 0.044895
```

The misclassification rate is given by

```
misclass.rf

##           response.test
## yhat.rf   nonspam spam
## nonspam   824    38
## spam      24   495
```

- h) Use `gbm()` to construct a boosted classification tree using 5000 trees, an interaction depth of $d = 3$ and a shrinkage parameter of $\lambda = 0.001$. Predict the response for the test data and report the misclassification rate.

The `gbm()` function does not allow factors, so we have to use '1' and '0' instead of `spam` and `nonspam`

```
library(gbm)
set.seed(4268)

spamboost = spam
spamboost$type = c()
spamboost$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 = 3, shrinkage = 0.001)
```

We predict the response for the test data

```
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])

misclass.boost
```

```
##
## yhat.boost   0    1
##             0 812  54
##             1  36 479
```

and the misclassification rate is

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

```
## [1] 0.06517017
```

- i) Compare the misclassification rates in d-h. Which method gives the lowest misclassification rate for the test data? Are the results as expected?

We get lower misclassification rates for bagging, random forest and boosting than for a simple tree, which is expected.