Laboration report in Machine Learning

Computer lab 2 block 1

732A99

Simge Cinar
Duc Tran
William Wiik

Division of Statistics and Machine Learning
Department of Computer Science
Linköping University

26 November 2023

Contents

1 Assignment 2. Decision trees and logistic regression for bank marketing.

The data in this assignment is related with direct marketing campaigns of a Portuguese banking institution. Data consists of 21 variables, where 20 are input variables about the clients and the output variable is if the client has a term deposit (yes/no).

1.1 2.1

Question: Import the data to R, **remove variable "duration"** and divide into training/validation/test as 40/30/30: use data partitioning code specified in Lecture 2a.

Answer: Data was loaded into R, where all character variables were made into factor variables, and the variable duration was removed.

```
# Index 12 is the variable duration
data <- read.csv2("bank-full.csv", stringsAsFactors = TRUE)[, -12]

# Data partitioning (40/30/30)
# Training data
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.4))
train=data[id,]

# Validation data
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.3))
valid=data[id2,]

# Test data
id3=setdiff(id1,id2)
test=data[id3,]
```

After the splitting the number of observations in the data sets are as follows:

- Training: 18 084 observations.
- Validation: 13 563 observations.
- Test: 13 563 observations.

1.2 2.2

Question: Fit decision trees to the training data so that you change the default settings one by one (i.e. not simultaneously):
a. Decision Tree with default settings.
b. Decision Tree with smallest allowed node size equal to 7000.
c. Decision trees minimum deviance to 0.0005.
and report the misclassification rates for the training and validation data. Which model is the best one among these three? Report how changing the deviance and node size affected the size of the trees and explain why.

Answer:

The default setting in tree for smallest allowed node size is 10 and the default for minimum deviance is 0.01. The three different trees were fitted and training error, validation error and number of leaves are presented in table 1.

```
# Tree a: default setting
tree_a <- tree(y ~., data=train)
# Misclassification for training data for tree a
pred_train <- predict(tree_a, newdata=train, type="class")
table_train <- table(train$y, pred_train)
miss_train <- 1 - (sum(diag(table_train)) / sum(table_train))
# Misclassification for validation data for tree a
pred_valid <- predict(tree_a, newdata=valid, type="class")
table_valid <- table(valid$y, pred_valid)
miss_valid <- 1 - (sum(diag(table_valid)) / sum(table_valid))
# Number of leaves
leaf <- sum(tree_a$frame$var == "<leaf>")
# Misclassification for tree a
model_a <- c(miss_train, miss_valid, leaf)



# Tree b: smallest size changed from 10 (default) to 7 000
tree_b <- tree(y ~., data=train, control=tree.control(nobs = nrow(train),
                                                      minsize = 7000))
# Misclassification for training data for tree b
pred_train <- predict(tree_b, newdata=train, type="class")
table_train <- table(train$y, pred_train)
miss_train <- 1 - (sum(diag(table_train)) / sum(table_train))
# Misclassification for validation data for tree b
pred_valid <- predict(tree_b, newdata=valid, type="class")
table_valid <- table(valid$y, pred_valid)
miss_valid <- 1 - (sum(diag(table_valid)) / sum(table_valid))
# Number of leaves
leaf <- sum(tree_b$frame$var == "<leaf>")
# Misclassification for tree b
model_b <- c(miss_train, miss_valid, leaf)
```

```r
# Tree c: mindev changed from 0.01 (default) to 0.0005
tree_c <- tree(y ~., data=train, control=tree.control(nobs = nrow(train),
                                                      mindev = 0.0005))
# Misclassification for training data for tree c
pred_train <- predict(tree_c, newdata=train, type="class")
table_train <- table(train$y, pred_train)
miss_train <- 1 - (sum(diag(table_train)) / sum(table_train))
# Misclassification for validation data for tree c
pred_valid <- predict(tree_c, newdata=valid, type="class")
table_valid <- table(valid$y, pred_valid)
miss_valid <- 1 - (sum(diag(table_valid)) / sum(table_valid))
# Number of leaves
leaf <- sum(tree_c$frame$var == "<leaf>")
# Misclassification for tree c
model_c <- c(miss_train, miss_valid, leaf)
```

Table 1: Misclassification error for the three trees on training and validation data.

	Training error	Validation error	Number of leaves
Tree a: default setting	0.104844	0.109268	6
Tree b: smallest node 7000	0.104844	0.109268	5
Tree c: min deviance 0.0005	0.094006	0.111922	122

From table 1, the training error and validation error for tree a and b are the same, however there is a difference in number of leaves where tree a has one more leaf. In tree b, the minimum node size is changed to 7 000 from 10 which forces the tree to have least 7 000 observations in each node. The increase in minimum node size makes that the tree can not split nodes smaller than 7 000 and thus the tree will have less leaves.

Training error is lowest for tree c, but the validation error is also highest for tree c. This indicates that tree c is more overfitted on training data compared to the tree a and b. The difference for tree c is that minimum deviance is changed from 0.01 to 0.0005. Minimum deviance is how much within-node deviance a node have to have compared to the root node to be able to split. By having a smaller number, the tree is allowed to split nodes with smaller deviance, which in turns allows the tree to grow and have smaller nodes than before.

Between tree a, b, and c the tree b is best since it has lowest validation error but also has the least amounts of leaves. By having less leaves the model is less complex and is easier to interpret.

1.3 2.3

Question: Use training and validation sets to choose the optimal tree depth in the model 2c: study the trees up to 50 leaves. Present a graph of the dependence of deviances for the training and the validation data on the number of leaves and interpret this graph in terms of bias-variance tradeoff. Report the optimal amount of leaves and which variables seem to be most important for decision making in this tree. Interpret the information provided by the tree structure (not everything but most important findings).

Answer: Tree c is used where the tree is post-pruned to trees with between 2 and 50 leaves. The deviance for training data and validation data are calculated and the code is as follows:

```
fit <- tree(y ~., data=train,
            control=tree.control(nobs = nrow(train),
                                 mindev = 0.0005))
trainScore=rep(0,50)
testScore=rep(0,50)
for(i in 2:50){
  prunedTree=prune.tree(fit,best=i)
  pred=predict(prunedTree, newdata=valid,
               type="tree")
  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(pred)
}
```

The tree with the lowest validation error had 22 leaves.

```
# Finds min, add +1 since index 1 is tree with 2 leaves.
which(min(testScore[2:50]) == testScore[2:50])+1
```

```
## [1] 22
```

The dependence of deviances for training and validation data for different numbers of leaves is presented in figure 1.

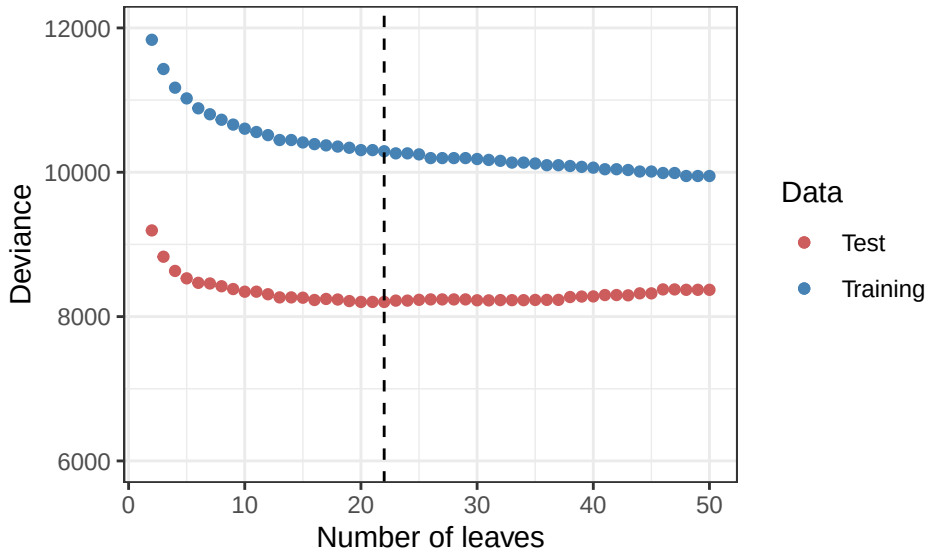

Figure 1: Dependence of deviances for training- and testdata.

In figure 1 the dashed line shows where the tree with the lowest deviance for testdata is located. For tree models, the complexity increases when the number of leaves increase. In terms of bias-variance tradeoff a model with low complexity (few leaves) have high bias but low variance and is considered to be underfitting. In contrast a model with high complexity have low bias but high variance and is considered to be overfitting. In figure 1 the tree with 22 leaves (dashed line) is considered to be the tree where we have a balance between low/high values for bias and variance which gives us that the tree has the optimal fit since it is not underfitting or overfitting.

The tree with 22 leaves is considered to be optimal and is presented in figure 2.

```
best_fit <- prune.tree(fit, best=22)
plot(best_fit)
text(best_fit, pretty=0)
```

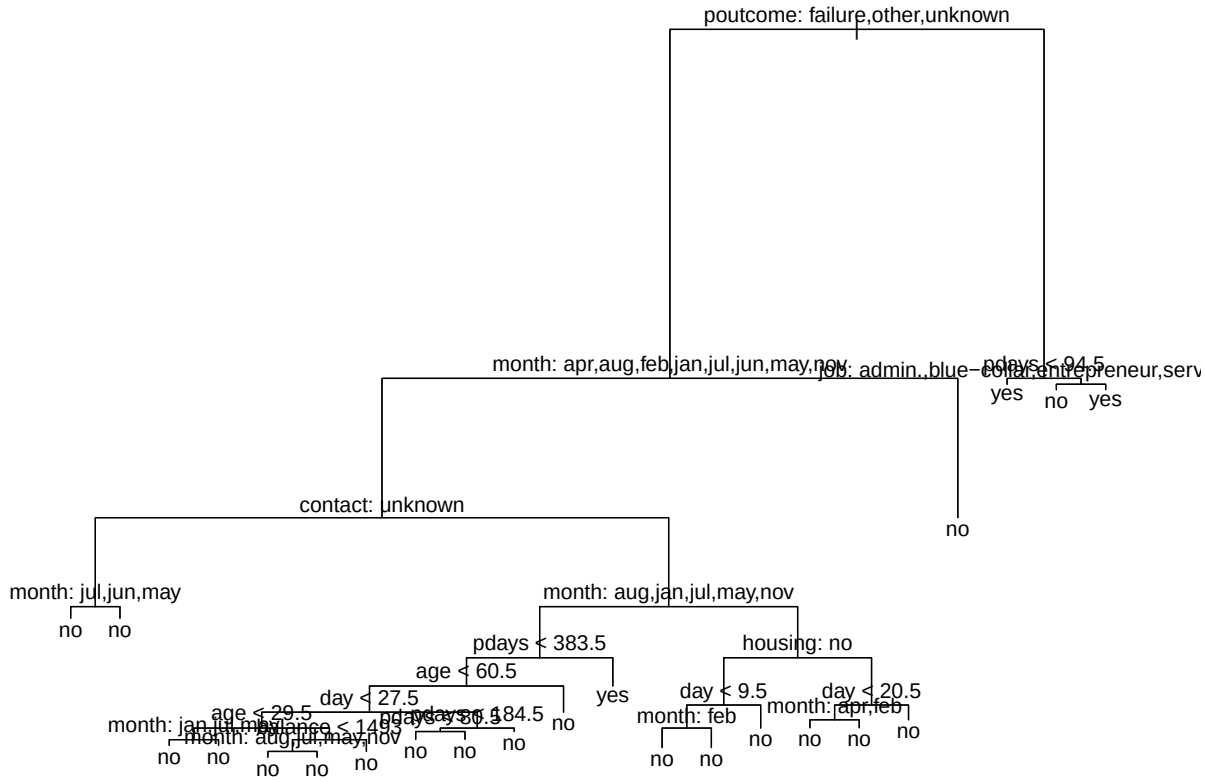

Figure 2: Tree with 22 leaves.

In figure 2 most of the leaves have the prediction "no". The leaves with the predictions "yes" used the variables: poutcome, month, contact, pdays, and job. These 5 variables seems to be the most important for decision making for this tree. The tree structure is presented in the following output:

```
best_fit
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
```

```
##    1) root 18084 12850.00 no ( 0.88576 0.11424 )
##      2) poutcome: failure,other,unknown 17468 11030.00 no ( 0.90422 0.09578 )
##        4) month: apr,aug,feb,jan,jul,jun,may,nov 16828  9772.00 no ( 0.91520 0.08480 )
##          8) contact: unknown 5130  1599.00 no ( 0.96374 0.03626 )
##           16) month: jul,jun,may 5074  1502.00 no ( 0.96610 0.03390 ) *
##           17) month: apr,aug,feb,jan,nov 56    62.98 no ( 0.75000 0.25000 ) *
##          9) contact: cellular,telephone 11698  7914.00 no ( 0.89391 0.10609 )
##           18) month: aug,jan,jul,may,nov 9284  5503.00 no ( 0.91265 0.08735 )
##             36) pdays < 383.5 9246  5373.00 no ( 0.91510 0.08490 )
##               72) age < 60.5 9097  5107.00 no ( 0.91920 0.08080 )
##                144) day < 27.5 7670  4588.00 no ( 0.91147 0.08853 )
##                  288) age < 29.5 754   637.10 no ( 0.85013 0.14987 )
##                    576) month: jan,jul,may 681   528.00 no ( 0.86931 0.13069 ) *
##                    577) month: aug,nov 73    92.46 no ( 0.67123 0.32877 ) *
##                  289) age > 29.5 6916  3918.00 no ( 0.91816 0.08184 )
##                    578) balance < 1493 5180  2635.00 no ( 0.92973 0.07027 )
##                     1156) month: aug,jul,may,nov 5164  2596.00 no ( 0.93087 0.06913 ) *
##                     1157) month: jan 16    21.93 no ( 0.56250 0.43750 ) *
##                    579) balance > 1493 1736  1249.00 no ( 0.88364 0.11636 ) *
##                145) day > 27.5 1427   472.40 no ( 0.96076 0.03924 )
##                  290) pdays < 184.5 1308   462.50 no ( 0.95719 0.04281 )
##                    580) pdays < 80.5 1277   402.60 no ( 0.96319 0.03681 ) *
##                    581) pdays > 80.5 31    37.35 no ( 0.70968 0.29032 ) *
##                  291) pdays > 184.5 119     0.00 no ( 1.00000 0.00000 ) *
##               73) age > 60.5 149   190.10 no ( 0.66443 0.33557 ) *
##             37) pdays > 383.5 38    47.40 yes ( 0.31579 0.68421 ) *
##           19) month: apr,feb,jun 2414  2262.00 no ( 0.82187 0.17813 )
##             38) housing: no 1123  1321.00 no ( 0.72484 0.27516 )
##               76) day < 9.5 691   668.20 no ( 0.81187 0.18813 )
##                152) month: feb 505   364.20 no ( 0.88317 0.11683 ) *
##                153) month: apr,jun 186   247.30 no ( 0.61828 0.38172 ) *
##               77) day > 9.5 432   586.10 no ( 0.58565 0.41435 ) *
##             39) housing: yes 1291   803.20 no ( 0.90627 0.09373 )
##               78) day < 20.5 1210   655.90 no ( 0.92314 0.07686 )
##                156) month: apr,feb 1154   565.70 no ( 0.93328 0.06672 ) *
##                157) month: jun 56    67.01 no ( 0.71429 0.28571 ) *
##               79) day > 20.5 81   104.40 no ( 0.65432 0.34568 ) *
##        5) month: dec,mar,oct,sep 640   852.70 no ( 0.61562 0.38438 ) *
##      3) poutcome: success 616   806.40 yes ( 0.36201 0.63799 )
##        6) pdays < 94.5 170   185.50 yes ( 0.23529 0.76471 ) *
##        7) pdays > 94.5 446   603.90 yes ( 0.41031 0.58969 )
##         14) job: admin.,blue-collar,entrepreneur,services,technician 213   295.20 no ( 0.50704 0.4929
##         15) job: housemaid,management,retired,self-employed,student,unemployed,unknown 233   292.80 y
```

From the output the four leaves that predicts "yes" have the probabilities between 0.58969 and 0.76471.
What more should we comment?

1.4 2.4

Question: Estimate the confusion matrix, accuracy and F1 score for the test data by using the optimal model from step 3. Comment whether the model has a good predictive power and which of the measures (accuracy or F1-score) should be preferred here.

Answer: The confusion matrix is estimated for the test data and is presented in the following output:

```
# Confusion matrix
pred_test <- predict(best_fit, newdata=test, type="class")
conf_mat_tree <- table(True = test$y, Predicton = pred_test)
conf_mat_tree
```

```
##       Predicton
## True      no   yes
##   no   11872   107
##   yes   1371   214
```

The accuracy of the model on test data is around 89%.

```
# Accuracy
acc_test <- sum(diag(conf_mat_tree)) / sum(conf_mat_tree)
acc_test
```

```
## [1] 0.8910351
```

The F1-score test data is around 0.2246.

```
# F1-score
# True positive
tp <- conf_mat_tree[2,2]
# False positive
fp <- conf_mat_tree[1,2]
# False negative
fn <- conf_mat_tree[2,1]

precision <- tp / (tp+fp)
recall <- tp / (tp+fn)
# F-score is between 0 and 1, where the closer to 1, the better.
f1_score <- 2 * (precision*recall) / (precision + recall)
f1_score
```

```
## [1] 0.224554
```

The accuracy of the model is around 89%, which looks high. But examining the class "yes", the model only predicted around 13.5% (214/(214+1371)) of "yes" observations as "yes. Confusion matrix indicate that we have unbalanced data, which makes the model able to predict"no" (the dominant class) better than "yes". F1-score focuses more on the prediction "yes" compared to accuracy. In this case we have really unbalanced classes and F1-score is therefore a better measurement. F1-score is between 0 and 1, where the closer F1-score is to 1 the better. The model has a F1-score of 0.22 and is not considered to have good predictive power.

1.5 2.5

Question: Perform a decision tree classification of the test data with the following loss matrix,

$$L = \begin{matrix} \\ Observed \quad yes \\ \qquad\quad no \end{matrix} \begin{matrix} Predicted \\ \begin{pmatrix} 0 & 5 \\ 1 & 0 \end{pmatrix} \end{matrix}$$

and report the confusion matrix for the test data. Compare the results with the results from step 4 and discuss how the rates has changed and why.

Answer: From the slides from the lectures we get the following from the loss matrix:

$E(y = no, \hat{y} = yes) = 1$
$E(y = yes, \hat{y} = no) = 5$

We can use this for classification of the model with:

$\frac{p(y=no|x)}{p(y=yes|x)} > 5 \rightarrow$ *classify as no.*

The code used in R for this is as follows:

```
# Predicts the probability of each class.
pred_best <- predict(best_fit, test, type="vector")
prediction <- as.vector(ifelse(pred_best[,1]/pred_best[,2] > 5, "no", "yes"))
table(test$y, prediction)
```

```
##      prediction
##          no   yes
##   no  11030   949
##   yes   771   814
```

Compared to the confusion matrix in 2.4, the model has now more predictions for the class "yes". The loss matrix for the model now says that a wrong prediction on a class that is "yes" is 5 times more severe than a wrong prediction on a class that is "no". Before the model predicted "yes" was if the probability was above 50%, now the model predicts "yes" if the probability is above 16.7% (100/6).

1.6 2.6

Question: Use the optimal tree and a logistic regression model to classify the test data by using the following principle:

$$\hat{Y} = \text{yes if } p(Y = \text{'yes'}|X) > \pi, \text{otherwise } \hat{Y} = no$$

where $\pi = 0.05, 0.1, 0.15, ..., 0.9, 0.95$. Compute the TPR and FPR values for the two models and plot the corresponding ROC curves. Conclusion? Why precision-recall curve could be a better option here?

Answer: True positive rate (TPR) and false positive rate (FPR) were calculated and the ROC curves for tree and logistic regression model are presented in figure 3.

```r
# ROC for tree
d <- data.frame(TPR = 1, FPR = 1)
roc_logistic <- d[FALSE, ]
roc_tree <- d[FALSE, ]

pred_best <- predict(best_fit, test, type="vector")
probs <- seq(from=0.05, to=0.95, by=0.05)
for(i in 1:length(probs)){
  prediction <- ifelse(pred_best[,2] > probs[i], "yes", "no")
  conf_mat <- table(test$y, prediction)
  if(dim(conf_mat)[2] == 1){
    TP <- 0
    FP <- 0
  } else {
    TP <- conf_mat[2,2]
    FP <- conf_mat[1,2]
  }
  TN <- conf_mat[1,1]
  FN <- conf_mat[2,1]

  # TPR, true positive rate
  roc_tree[i,1] <- TP / (TP+FN)
  # FPR, false positive rate
  roc_tree[i,2] <- FP / (FP+TN)
}
plot_data <- data.frame(roc_tree)

# ROC for logistic
model_logistic <- glm(y~., family="binomial", train)
pred_logistic <- predict(model_logistic, test, type="response")
d <- data.frame(TPR = 1, FPR = 1)
roc_logistic <- d[FALSE, ]
probs <- seq(from=0.05, to=0.95, by=0.05)
roc_logistic <- d[FALSE, ]

for(i in 1:length(probs)){
  pred <- ifelse(pred_logistic > probs[i], "yes", "no")
  conf_mat <- table(test$y, pred)
  TP <- conf_mat[2,2]
  TN <- conf_mat[1,1]
  FN <- conf_mat[2,1]
  FP <- conf_mat[1,2]

  # TPR, true positive rate
  roc_logistic[i,1] <- TP / (TP+FN)
  # FPR, false positive rate
  roc_logistic[i,2] <- FP / (FP+TN)
}
plot_data_logistic <- data.frame(roc_logistic)
```

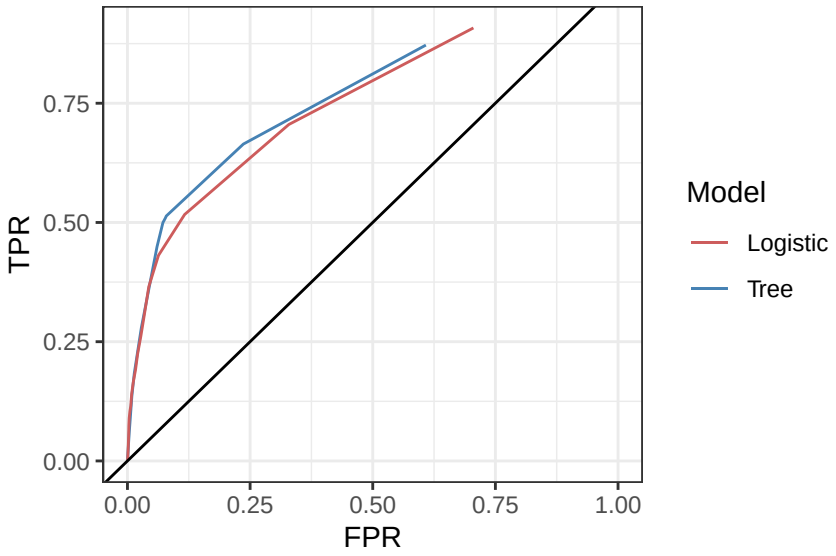

Figure 3: Roc curves for tree and logistic model.

In figure 3 the black line represents the theoretical value for a random classifier. A model with a ROC curve further away from this line towards upper left part of the plot is considered better. The tree model is therefore considered to be better than the logistic model.

In precision-recall curve, recall is defined the same as TPR in ROC curve and precision is defined as: $precision = \frac{TP}{TP+FP}$, where TP is true positive and FP is false positive. Since we have unbalanced data, where the class "yes" is a lot smaller, a precision-recall curve can give us better information on how well the models predicts the class "yes".

2 Statement of Contribution

We worked on the assignment individually for the computer labs (to be more efficient when asking questions), William on task 1, Duc on task 2, and Simge on task 3. We later solved all assignment individually and compared and discussed our solutions before dividing the task of writing the laboration report.

2.1 Question 1

Text written by William.

2.2 Question 2

Text written by Duc.

2.3 Question 3

Text written by Simge.

3 Appendix

The code used in this laboration report are summarised in the code as follows:

```
library(ggplot2)
library(tree)
library(knitr)
library(dplyr)
library(kableExtra)
knitr::opts_chunk$set(
  echo = TRUE,
  fig.width = 4.5,
  fig.height = 3)
# Index 12 is the variable duration
data <- read.csv2("bank-full.csv", stringsAsFactors = TRUE)[, -12]

# Data partitioning (40/30/30)
# Training data
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.4))
train=data[id,]

# Validation data
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.3))
```

```r
valid=data[id2,]

# Test data
id3=setdiff(id1,id2)
test=data[id3,]
# Tree a: default setting
tree_a <- tree(y ~., data=train)
# Misclassification for training data for tree a
pred_train <- predict(tree_a, newdata=train, type="class")
table_train <- table(train$y, pred_train)
miss_train <- 1 - (sum(diag(table_train)) / sum(table_train))
# Misclassification for validation data for tree a
pred_valid <- predict(tree_a, newdata=valid, type="class")
table_valid <- table(valid$y, pred_valid)
miss_valid <- 1 - (sum(diag(table_valid)) / sum(table_valid))
# Number of leaves
leaf <- sum(tree_a$frame$var == "<leaf>")
# Misclassification for tree a
model_a <- c(miss_train, miss_valid, leaf)



# Tree b: smallest size changed from 10 (default) to 7 000
tree_b <- tree(y ~., data=train, control=tree.control(nobs = nrow(train),
                                                      minsize = 7000))
# Misclassification for training data for tree b
pred_train <- predict(tree_b, newdata=train, type="class")
table_train <- table(train$y, pred_train)
miss_train <- 1 - (sum(diag(table_train)) / sum(table_train))
# Misclassification for validation data for tree b
pred_valid <- predict(tree_b, newdata=valid, type="class")
table_valid <- table(valid$y, pred_valid)
miss_valid <- 1 - (sum(diag(table_valid)) / sum(table_valid))
# Number of leaves
leaf <- sum(tree_b$frame$var == "<leaf>")
# Misclassification for tree b
model_b <- c(miss_train, miss_valid, leaf)


# Tree c: mindev changed from 0.01 (default) to 0.0005
tree_c <- tree(y ~., data=train, control=tree.control(nobs = nrow(train),
                                                      mindev = 0.0005))
# Misclassification for training data for tree c
pred_train <- predict(tree_c, newdata=train, type="class")
table_train <- table(train$y, pred_train)
miss_train <- 1 - (sum(diag(table_train)) / sum(table_train))
# Misclassification for validation data for tree c
pred_valid <- predict(tree_c, newdata=valid, type="class")
table_valid <- table(valid$y, pred_valid)
```

```r
miss_valid <- 1 - (sum(diag(table_valid)) / sum(table_valid))
# Number of leaves
leaf <- sum(tree_c$frame$var == "<leaf>")
# Misclassification for tree c
model_c <- c(miss_train, miss_valid, leaf)
# Summarised results
table <- data.frame(rbind(model_a, model_b, model_c))
colnames(table) <- c("Training error", "Validation error", "Number of leaves")
rownames(table) <- c("Tree a: default setting", "Tree b: smallest node 7000",
                     "Tree c: min deviance 0.0005")

kable(table, digits=6, booktabs=T,
      caption="Misclassification error for the three trees on training and validation data.") %>%
   kable_styling(latex_options = "HOLD_position")
fit <- tree(y ~., data=train,
            control=tree.control(nobs = nrow(train),
                                 mindev = 0.0005))
trainScore=rep(0,50)
testScore=rep(0,50)
for(i in 2:50){
  prunedTree=prune.tree(fit,best=i)
  pred=predict(prunedTree, newdata=valid,
               type="tree")
  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(pred)
}
# Finds min, add +1 since index 1 is tree with 2 leaves.
which(min(testScore[2:50]) == testScore[2:50])+1
plot_data <- data.frame(training = trainScore[-1], test = testScore[-1])
ggplot(plot_data, aes(x=2:50)) +
  geom_point(aes(y=training, color="training")) +
  geom_point(aes(y=test, color="test")) +
  geom_vline(xintercept=22, linetype="dashed") +
  ylim(6000, 12000) +
  labs(x="Number of leaves", y="Deviance") +
  scale_colour_manual(name="Data",
                      values=c("indianred", "steelblue"),
                      labels=c("Test", "Training")) +
  theme_bw()

#
# plot(2:50, trainScore[2:50], type="b", col="red",ylim=c(8000,12000))
# points(2:50, testScore[2:50], type="b", col="blue")
# # Fill the point with the min value on validation
# points(22, testScore[22], pch=16, col="blue")
best_fit <- prune.tree(fit, best=22)
plot(best_fit)
text(best_fit, pretty=0)
best_fit
```

```r
# Confusion matrix
pred_test <- predict(best_fit, newdata=test, type="class")
conf_mat_tree <- table(True = test$y, Predicton = pred_test)
conf_mat_tree

# Accuracy
acc_test <- sum(diag(conf_mat_tree)) / sum(conf_mat_tree)
acc_test
# F1-score
# True positive
tp <- conf_mat_tree[2,2]
# False positive
fp <- conf_mat_tree[1,2]
# False negative
fn <- conf_mat_tree[2,1]

precision <- tp / (tp+fp)
recall <- tp / (tp+fn)
# F-score is between 0 and 1, where the closer to 1, the better.
f1_score <- 2 * (precision*recall) / (precision + recall)
f1_score
# Predicts the probability of each class.
pred_best <- predict(best_fit, test, type="vector")
prediction <- as.vector(ifelse(pred_best[,1]/pred_best[,2] > 5, "no", "yes"))
table(test$y, prediction)
# ROC for tree
d <- data.frame(TPR = 1, FPR = 1)
roc_logistic <- d[FALSE, ]
roc_tree <- d[FALSE, ]

pred_best <- predict(best_fit, test, type="vector")
probs <- seq(from=0.05, to=0.95, by=0.05)
for(i in 1:length(probs)){
  prediction <- ifelse(pred_best[,2] > probs[i], "yes", "no")
  conf_mat <- table(test$y, prediction)
  if(dim(conf_mat)[2] == 1){
    TP <- 0
    FP <- 0
  } else {
    TP <- conf_mat[2,2]
    FP <- conf_mat[1,2]
  }
  TN <- conf_mat[1,1]
  FN <- conf_mat[2,1]

  # TPR, true positive rate
  roc_tree[i,1] <- TP / (TP+FN)
  # FPR, false positive rate
  roc_tree[i,2] <- FP / (FP+TN)
```

```
}
plot_data <- data.frame(roc_tree)

# ROC for logistic
model_logistic <- glm(y~., family="binomial", train)
pred_logistic <- predict(model_logistic, test, type="response")
d <- data.frame(TPR = 1, FPR = 1)
roc_logistic <- d[FALSE, ]
probs <- seq(from=0.05, to=0.95, by=0.05)
roc_logistic <- d[FALSE, ]

for(i in 1:length(probs)){
  pred <- ifelse(pred_logistic > probs[i], "yes", "no")
  conf_mat <- table(test$y, pred)
  TP <- conf_mat[2,2]
  TN <- conf_mat[1,1]
  FN <- conf_mat[2,1]
  FP <- conf_mat[1,2]

  # TPR, true positive rate
  roc_logistic[i,1] <- TP / (TP+FN)
  # FPR, false positive rate
  roc_logistic[i,2] <- FP / (FP+TN)
}
plot_data_logistic <- data.frame(roc_logistic)
ggplot(plot_data) +
  geom_line(aes(x=FPR, y=TPR, color="Tree")) +
  geom_line(data=plot_data_logistic, aes(x=FPR, y=TPR, color="Logistic")) +
  scale_x_continuous(limits=c(0,1)) +
  geom_abline(intercept = 0, slope = 1) +
  scale_color_manual(name="Model",
                     values=c("indianred", "steelblue"),
                     labels=c("Logistic", "Tree")) +
  theme_bw()
```