Data Mining Laboratory

Ewa Figielska

Decision trees

- A **decision tree**, is a collection of **decision nodes**, connected by **branches**, extending downward from the **root node** until terminating in **leaf nodes**.

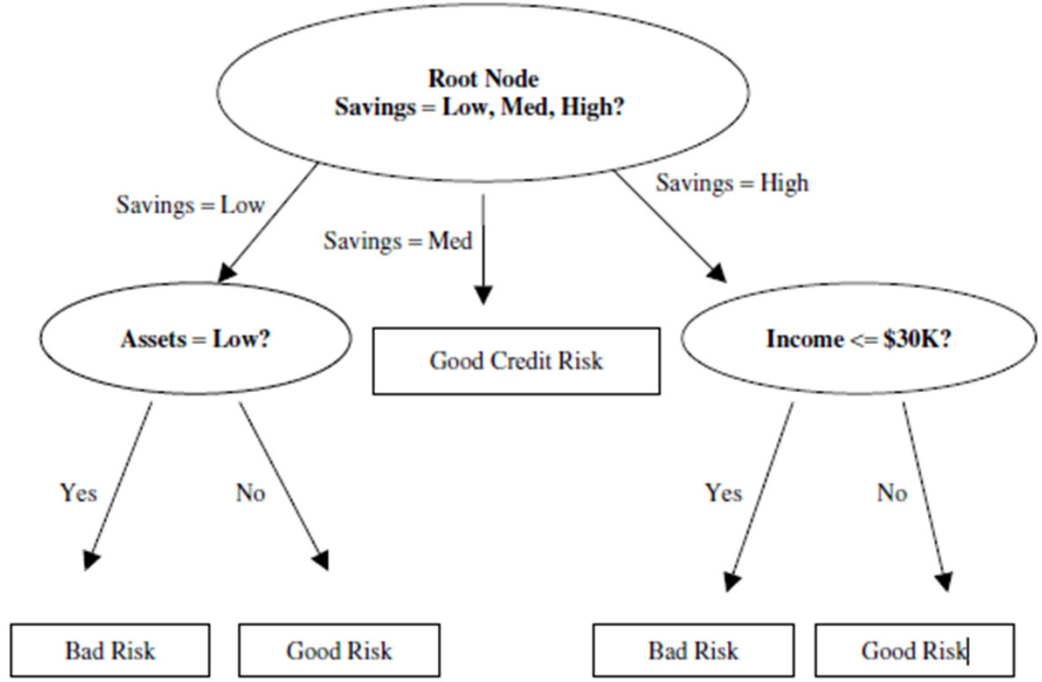


Figure 1. Simple decision trees (source: Larose, 2005)

- In the figure:
  - The target variable is *credit risk*, with potential customers being classified as either good or bad credit risks.
  - The predictor variables are *savings* (low, medium, and high), *assets* (low or not low), and *income* (≤$50,000 or >$50,000).
  - The root node represents a decision node, testing the savings level of the records.
  - The data set is partitioned (*split)* according to the values of the savings attribute.
  - The records with low savings are sent via the leftmost branch (*savings* = *low*) to another decision node.
  - The records with high savings are sent via the rightmost branch to a different decision node.
  - The records with medium savings are sent via the middle branch directly to a leaf node, indicating the termination of this branch (the customer with medium savings predicts good credit with 100% accuracy).

Requirements for decision trees

Certain requirements must be met before decision tree algorithms may be applied:

1. Decision tree algorithms represent supervised learning, and as such require preclassified target variables.
2. A training data set with the values of the target variable must be supplied.
3. This training data set should be rich and varied.
4. The target attribute classes must be discrete.

Classification and regression trees (CART) algorithm

- The *classification and regression trees* (CART) method was introduced in 1984.
- The CART algorithm grows the decision tree by recursively partitioning (splitting) the records in the training data set into subsets of records.
- The decision tree is strictly binary, containing exactly two branches for each decision node.
- The CART algorithm selects the optimal split according to the following criterion:

$$\Phi(s|t) = 2P_L P_R \sum_{j=1}^{K} |P(j|t_L) - P(j|t_R)|$$

where

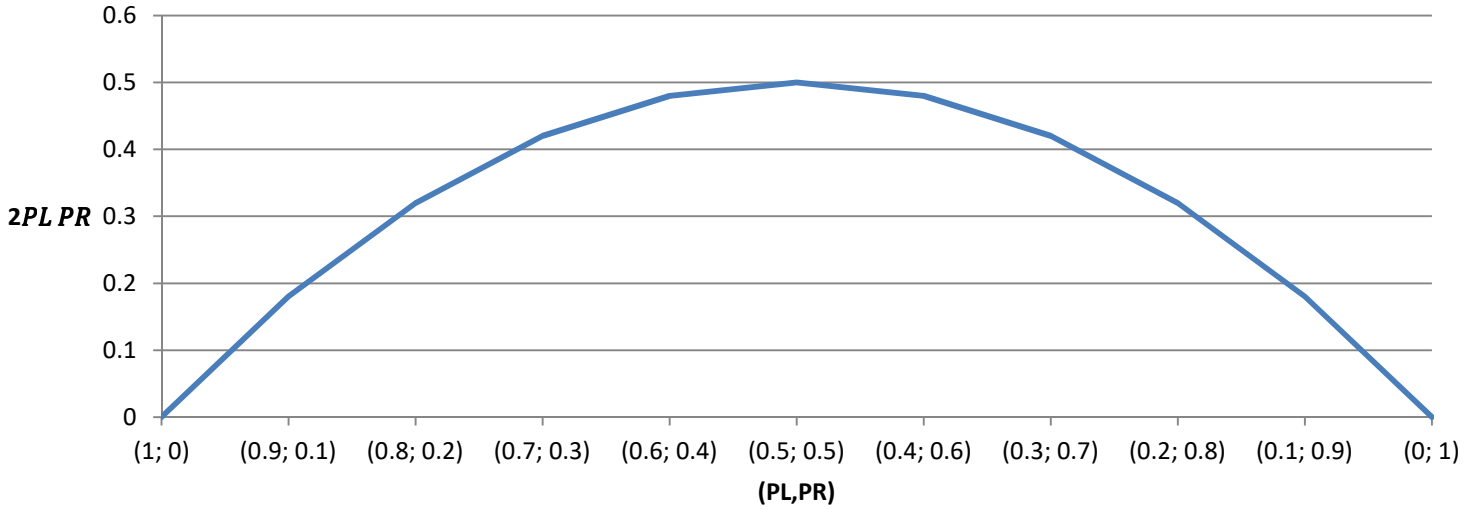
$\Phi(s|t)$ is a measure of the "goodness" of a candidate split s at node t,
t_L = left child of node t (set of records)
t_R = right child of node t (set of records)
P_L = (number of records at t_L) / (number of records at t)
P_R = (number of records at t_R) / (number of records at t)
$P(j|t_L)$ = (number of class j records at t_L) / (number of records in t_L)
$P(j|t_R)$= (number of class j records at t_R) / (number of records in t_R)
K = number of the target variable classes

Optimal split

- The criterion $\Phi(s|t)$ is maximized.
- The best split s of set t is the one with the greatest value of $\Phi(s|t)$.

$$\Phi(s|t) = 2P_L P_R \sum_{j=1}^{K} |P(j|t_L) - P(j|t_R)|$$

- If the child nodes are homogeneous (each contains only records of the same class), factor $\sum_{j=1}^{K} |P(j|t_L) - P(j|t_R)|$ has the maximum value (equal to the number of classes).
- If there are the same number of records in both the child nodes, factor $2P_L P_P$ has the maximum value (equal to 0.5).



Example

Customer	Savings	Assets	Income ($1000s)	Credit Risk
1	Medium	High	75	Good
2	Low	Low	50	Bad
3	Hight	Medium	25	Bad
4	Medium	Medium	50	Good
5	Low	Medium	100	Good
6	Hight	High	25	Good
7	Low	Low	25	Bad
8	Medium	Medium	75	Good

Training set

We are interested in using CART to build a decision tree for predicting whether a particular customer should be classified as being a good or a bad credit risk.

Target variable = Credit Risk
Predictor variables = Savings, Assets, Income

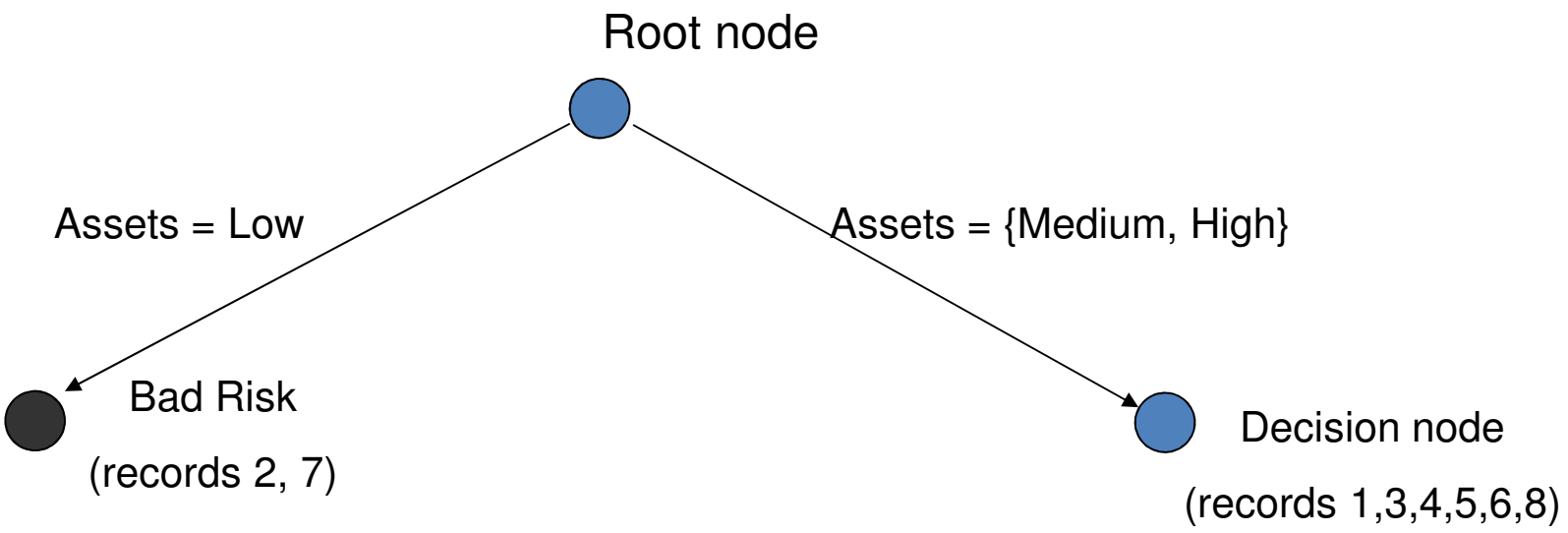
First split

Cust.	Savings	Assets	Income	Credit Risk
1	Medium	High	75	Good
2	Low	Low	50	Bad
3	Hight	Medium	25	Bad
4	Medium	Medium	50	Good
5	Low	Medium	100	Good
6	Hight	High	25	Good
7	Low	Low	25	Bad
8	Medium	Medium	75	Good

Candidate split	Left child node	Right child node
1	Savings = Low	Savings = {Medium, High}
2	Savings = Medium	Savings = {Low, High}
3	Savings = High	Savings = {Low, Medium}
4	Assets = Low	Assets = {Medium, High}
5	Assets = Medium	Assets = {Low, High}
6	Assets = High	Assets = {Low, Medium}
7	Income <= 25	Income > 25
8	Income <= 50	Income > 50
9	Income <= 75	Income > 75

Candidate split	P_L	$P(j\|t_L)$ Good Risk	$P(j\|t_L)$ Bad Risk	P_R	$P(j\|t_R)$ Good Risk	$P(j\|t_R)$ Bad Risk	$\Sigma_{j=1...K} \|P(j\|t_L) - P(j\|t_R)\|$	$\Phi(s\|t)$
1	3/8 = 0.375	1/3 = 0.333	2/3 = 0.667	5/8 = 0.625	4/5 = 0.800	1/5 = 0.200	|0.33-0.80|+ |0.67-0.2|= 0.933	0.438
2	3/8 = 0.375	3/3 = 1.000	0/3 = 0.000	5/8 = 0.625	2/5 = 0.400	3/5 = 0.600	1.200	0.563
3	2/8 = 0.250	1/2 = 0.500	1/2 = 0.500	6/8 = 0.750	4/6 = 0.667	2/6 = 0.333	0.333	0.125
4	2/8 = 0.250	0/2 = 0.000	2/2 = 1.000	6/8 = 0.750	5/6 = 0.833	1/6 = 0.167	1.667	0.625
5	4/8 = 0.500	3/4 = 0.750	1/4 = 0.250	4/8 = 0.500	2/4 = 0.500	2/4 = 0.500	0.500	0.250
6	2/8 = 0.250	2/2 = 1.000	0/2 = 0.000	6/8 = 0.750	3/6 = 0.500	3/6 = 0.500	1.000	0.375
7	3/8 = 0.375	1/3 = 0.333	2/3 = 0.667	5/8 = 0.625	4/5 = 0.800	1/5 = 0.200	0.933	0.438
8	5/8 = 0.625	2/5 = 0.400	3/5 = 0.600	3/8 = 0.375	3/3 = 1.000	0/3 = 0.000	1.200	0.563
9	7/8 = 0.875	4/7 = 0.571	3/7 = 0.429	1/8 = 0.125	1/1 = 1.000	0/1 = 0.000	0.857	0.188

CART decision tree after the first split



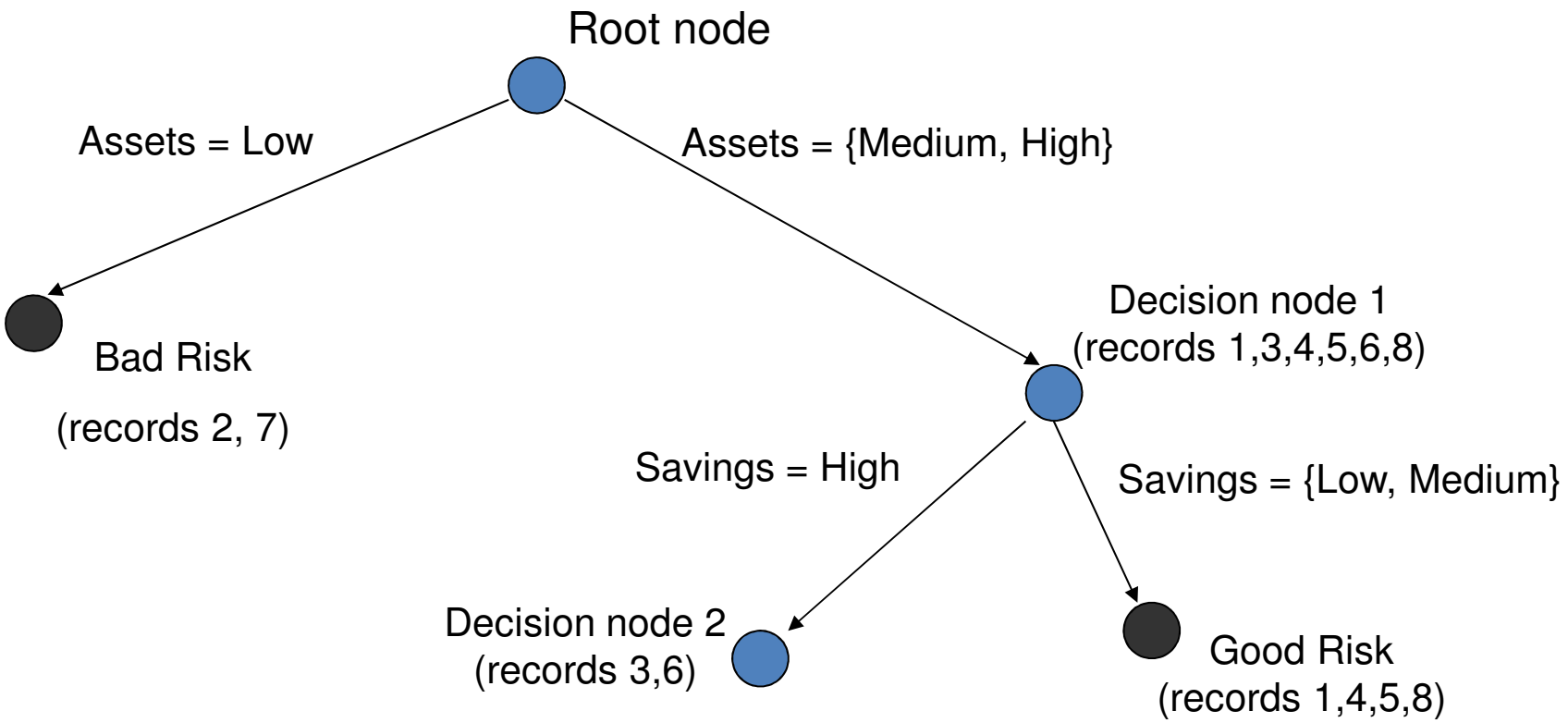
Second split

Candidate split	Left child node	Right child node
1	Savings = Low	Savings = {Medium, High}
2	Savings = Medium	Savings = {Low, High}
3	Savings = High	Savings = {Low, Medium}
4	Assets = Low NA	
5	Assets = Medium	Assets = High
6	Assets = High	Assets = Medium
7	Income <= 25	Income > 25
8	Income <= 50	Income > 50
9	Income <= 75	Income > 75

Cust.	Savings	Assets	Income	Credit Risk
1	Medium	High	75	Good
3	Hight	Medium	25	Bad
4	Medium	Medium	50	Good
5	Low	Medium	100	Good
6	Hight	High	25	Good
8	Medium	Medium	75	Good

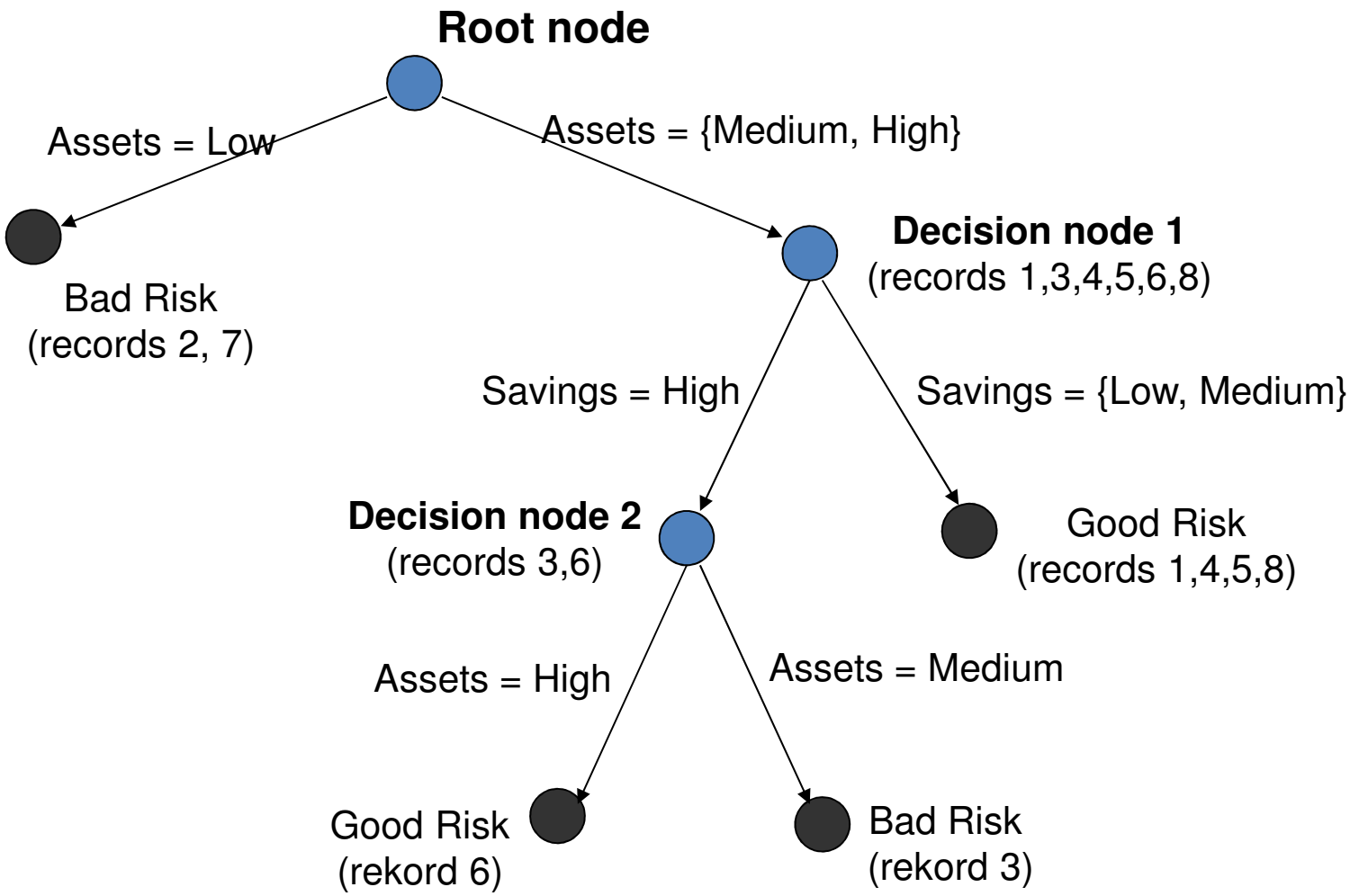
Candidate split		$P(j\|t_L)$			$P(j\|t_R)$		$\Sigma_{j=1...K} \|P(j\|t_L) - P(j\|t_R)\|$	$\Phi(s\|t)$
	P_L	Good Risk	Bad Risk	P_R	Good Risk	Bad Risk		
1	1/6 = 0.167	1/1 = 1.000	0/1 = 0.000	5/6 = 0.833	4/5 = 0.800	1/5 = 0.200	|1.0-0.8|+ |0-0.2|= 0.400	0.111
2	3/6 = 0.500	3/3 = 1.000	0/3 = 0.000	3/6 = 0.500	2/3 = 0.667	1/3 = 0.333	0.667	0.333
3	2/6 = 0.333	1/2 = 0.500	1/2 = 0.500	4/6 = 0.667	4/4 = 1.000	0/4 = 0.000	1.000	0.444
4	Assets = Low NA							
5	4/6 = 0.667	3/4 = 0.750	1/4 = 0.250	2/6 = 0.333	2/2 = 1.000	0/2 = 0.000	0.500	0.222
6	2/6 = 0.333	2/2 = 1.000	0/2 = 0.000	4/6 = 0.667	3/4 = 0.750	1/4 = 0.250	0.500	0.222
7	2/6 = 0.333	1/2 = 0.500	1/2 = 0.500	4/6 = 0.667	4/4 = 1.000	0/4 = 0.000	1.000	0.444
8	3/6 = 0.500	2/3 = 0.667	1/3 = 0.333	3/6 = 0.500	3/3 = 1.000	0/3 = 0.000	0.667	0.333
9	5/6 = 0.833	4/5 = 0.800	1/5 = 0.200	1/6 = 0.167	1/1 = 1.000	0/1 = 0.000	0.400	0.111

CART decision tree after the second split

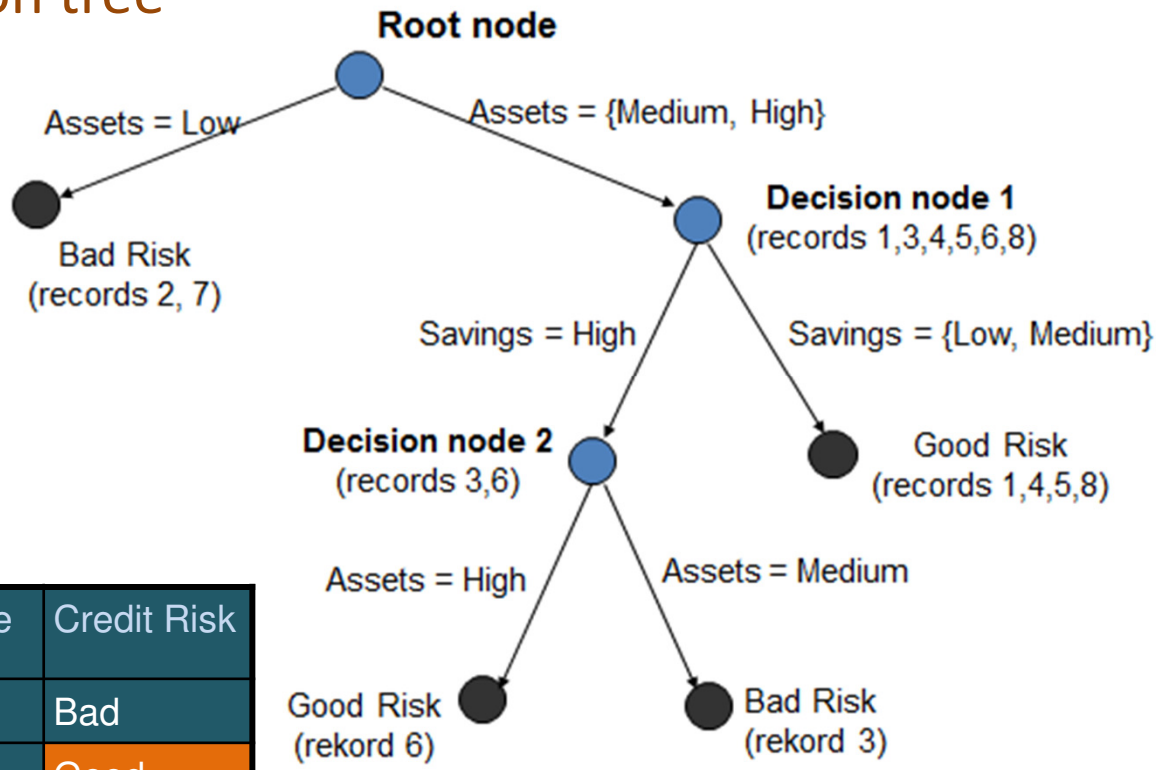


Decision node 2 is diverse, so we again seek the optimal split. Only two records remain in this decision node, each with the same value for savings (high) and income (25). Therefore, the only possible split is assets = high versus assets = medium.

Fully grown form of the CART decision tree



Evaluating the decision tree



Test set

Customer	Savings	Assets	Income	Credit Risk
9	Low	Low	50	Bad
10	Medium	Low	100	Good
11	Low	Medium	25	Bad
12	High	Medium	100	Good
13	Low	High	25	Good
14	Low	Low	25	Bad
15	Medium	Medium	25	Good

- Misclassified records: 10, 11, 12
- error rate = 3/7 = 0.43

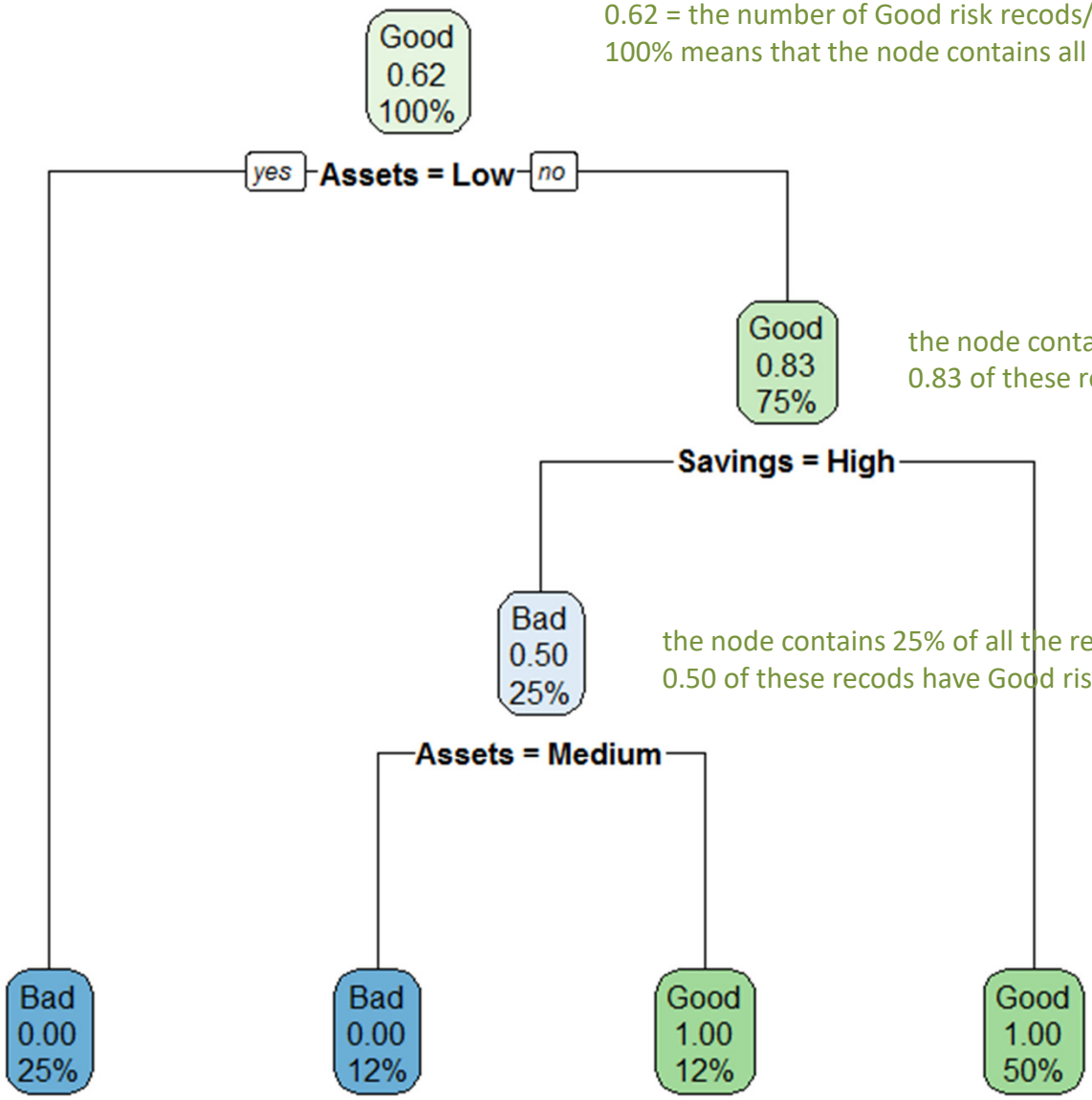
Creating and testing the CART decision tree in R

1. #install.packages(c("rpart","rpart.plot"))
2. #library("rpart");library("rpart.plot")
3. **d<-read.csv(file="credits.csv", stringsAsFactors =TRUE)**

4. **train_indices<-c(1:8)**
5. **d.train <- d[train_indices, 1:4]**
6. **d.test <- d[-train_indices,1:3]**
7. **d.test.class<- d[-train_indices,4]**

8. **model<-rpart(CreditRisk~Savings+Assets+Income, data = d.train, method = "class", control=rpart.control(minsplit=2))** #growing the tree by the CART algorithm
9. **rpart.plot(model)** #plotting the decision tree model

10. **pred <- predict(model, d.test,type="class")** #using the model to predict the target variable values for the test data
11. **error_rate<-mean(pred != d.test.class)**



0.62 = the number of Good risk recods/the number of records in the data set
100% means that the node contains all the records from the used data set

the node contains 75% of all the records
0.83 of these recods have Good risk

the node contains 25% of all the records
0.50 of these recods have Good risk

No records with
Good risk

All records have
Good risk

C4.5 algorithm. Entropy.

- The C4.5 algorithm was developed in 1993.
- It recursively visits each decision node, selecting the optimal split, until no further splits are possible.
- C4.5 algorithm is not restricted to binary splits.
- For categorical attributes, it by default produces a separate branch for each value of the categorical attribute.
- The C4.5 algorithm uses the concept of **information gain or entropy reduction** to select the optimal split.

- Suppose that we are given a variable X whose k possible values have probabilities $p_1, p_2, \ldots, p_k$. The smallest number of bits, on average per symbol, needed to transmit a stream of symbols representing the values of X is called the **entropy of X** and is defined as follows:

$$\boldsymbol{H(X) = -\sum_{j=1}^{k} p_j log_2(p_j)}$$

- For example, the result of a fair coin toss, with probability 0.5, can be transmitted using $-\sum_{j=1}^{2} 0.5 \log_2(0.5) = 1$ bit, which is 0 or 1, depending on the result of the toss.

C4.5

- C4.5 uses this concept of entropy as follows.
- Suppose that we have a candidate split S, which partitions the training data set T into k subsets, $T_1, T_2, \ldots, T_k$.
- The **mean information requirement** can be calculated as the weighted sum of the entropies for the individual subsets, as follows:

$$H_S(T) = -\sum_{i=1}^{k} P_i H_s(T_i)$$

where

k = number of subsets,

P_i = proportion of records in subset i,

$H_s(T_i)$ = entropy for subset T_i in split S.

- The **information gain,** i.e. the increase in information produced by partitioning the training data T according to candidate split S, is calculated as follows:

$$gain(S) = H(T) - H_S(T)$$

- At each decision node, C4.5 chooses the split that has the greatest information gain, $gain(S)$.

Example

Customer	Savings	Assets	Income ($1000s)	Credit Risk
1	Medium	High	75	Good
2	Low	Low	50	Bad
3	Hight	Medium	25	Bad
4	Medium	Medium	50	Good
5	Low	Medium	100	Good
6	Hight	High	25	Good
7	Low	Low	25	Bad
8	Medium	Medium	75	Good

The entropy before splitting:

$$H(T) = -\sum_j p_j log_2(p_j) = -5/8 \log_2(5/8) - 3/8 \log_2(3/8) = 0.9544$$

(Good credid risk: 5 records, Bad credit risk: 3 records).

Possible splits in the root:

Split	Subsets		
	T1	T2	T3
(S1) Savings	Low	Medium	High
(S2) Assets	Low	Medium	High
(S3) Income	≤ 25	>25	
(S4) Income	≤ 50	>50	
(S5) Income	≤ 75	>75	

Example. Entropy for each split

Cust.	Savings	Assets	Income	Credit Risk
1	Medium	High	75	Good
2	Low	Low	50	Bad
3	Hight	Medium	25	Bad
4	Medium	Medium	50	Good
5	Low	Medium	100	Good
6	Hight	High	25	Good
7	Low	Low	25	Bad
8	Medium	Medium	75	Good

1st split (S1). Savings.

$$P_{T1} = 3/8,$$
$$H_{S1}(T1) = -1/3\ log_2(1/3) - 2/3\ log_2(2/3) = 0.9183$$

$$P_{T2} = 3/8,$$
$$H_{S1}(T2) = -3/3\ log_2(3/3) - 0/3\ log_2(0/3) = 0.$$

$$P_{T3} = 2/8,$$
$$H_{S1}(T3) = -1/2\ log_2(1/2) - 1/2\ log_2(1/2) = 1.$$

Mean information requirement after splitting $S1$:

$$H_{S1}(T) = -\sum_{i=1}^{3} P_i H_{S1}(T_i) = 3/8 \cdot 0.9183 + 3/8 \cdot 0 + 2/8 \cdot 1 = 0.5944$$

Information gain: $gain(S1) = H(T) - H_{s1}(T) = 0.9544 - 0.5944 = 0.36$

2nd split (S2). Assets.

$$H_{S2}(T) = -\sum_{i=1}^{3} P_i H_{S2}(T_i) = 2/8 \cdot 0 + 4/8(-3/4\ log_2(3/4) - 1/4\ log_2(1/4)) + 2/8 \cdot 0 = 4/8 \cdot 0.8113 = 0.4057$$

$$gain(S2) = H(T) - H_{S2}(T) = 0.9544 - 0.4057 = 0.5487$$

Example. Entropy for each split

Customer	Savings	Assets	Income	Credit Risk
1	Medium	High	75	Good
2	Low	Low	50	Bad
3	Hight	Medium	25	Bad
4	Medium	Medium	50	Good
5	Low	Medium	100	Good
6	Hight	High	25	Good
7	Low	Low	25	Bad
8	Medium	Medium	75	Good

3rd split (S3). Income:

$$H_{S3}(T) = -\sum_{i=1}^{2} P_i H_{S3}(T_i)$$
$$= 3/8\,(-1/3\log_2(1/3) - 2/3\log_2(2/3)) + 5/8(-4/5\log_2(4/5) - 1/5\log_2(1/5))$$
$$= 0.7219$$

$$\boldsymbol{gain(S3) = H(T) - H_{S3}(T) = 0.9544 - 0.7956 = 0.1588}$$

4th split (S4). Income:

$$H_{S4}(T) = -\sum_{i=1}^{2} P_i H_{S4}(T_i)$$
$$= 5/8\,(-2/5\log_2(2/5) - 3/5\log_2(3/5)) + 3/8(-3/3\log_2(3/3) - 0/3\log_2(0/3))$$
$$= 0.6069$$

$$\boldsymbol{gain(S4) = H(T) - H_{S4}(T) = 0.9544 - 0.6069 = 0.3475}$$

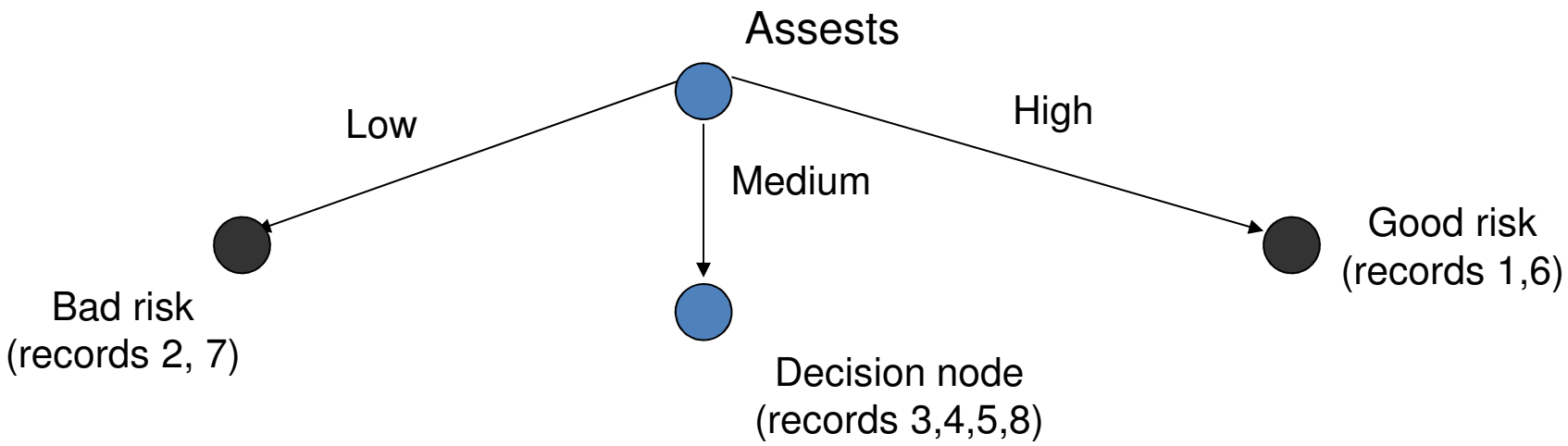
The worst split = S5.
The best split = S2.

5th split (S5). Income:

$$H_{S5}(T) = -\sum_{i=1}^{2} P_i H_{S5}(T_i)$$
$$= 7/8\,(-4/7\log_2(4/7) - 3/7\log_2(3/7)) + 1/8(-1/1\log_2(1/1) - 0/1\log_2(0/1))$$
$$= 0.8621$$

$$\boldsymbol{gain(S5) = H(T) - HS_5(T) = 0.9544 - 0.8621 = 0.0923}$$

Growing the tree



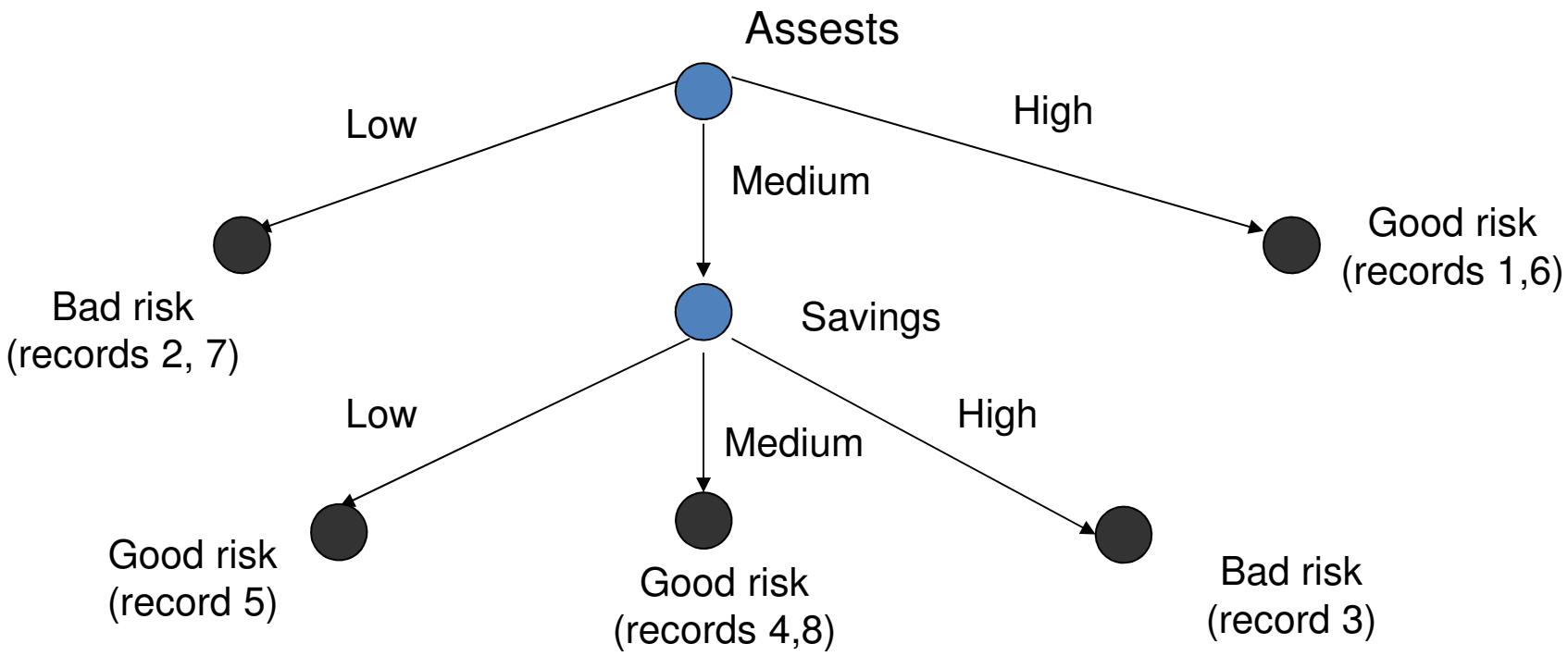
Records in the decision node

Cust.	Savings	Assets	Income	Credit Risk
3	Hight	Medium	25	Bad
4	Medium	Medium	50	Good
5	Low	Medium	100	Good
8	Medium	Medium	75	Good

Possible splits

Split	Subsets		
	T1	T2	T3
(S1) Savings	Low	Medium	High
(S3) Income	≤ 25	>25	
(S4) Income	≤ 50	>50	
(S5) Income	≤ 75	>75	

The tree



Customer	Savings	Assets	Income	Credit Risk
9	Low	Low	50	Bad
10	Medium	Low	100	Good
11	Low	Medium	25	Bad
12	High	Medium	100	Good
13	Low	High	25	Good
14	Low	Low	25	Bad
15	Medium	Medium	25	Good

Test set

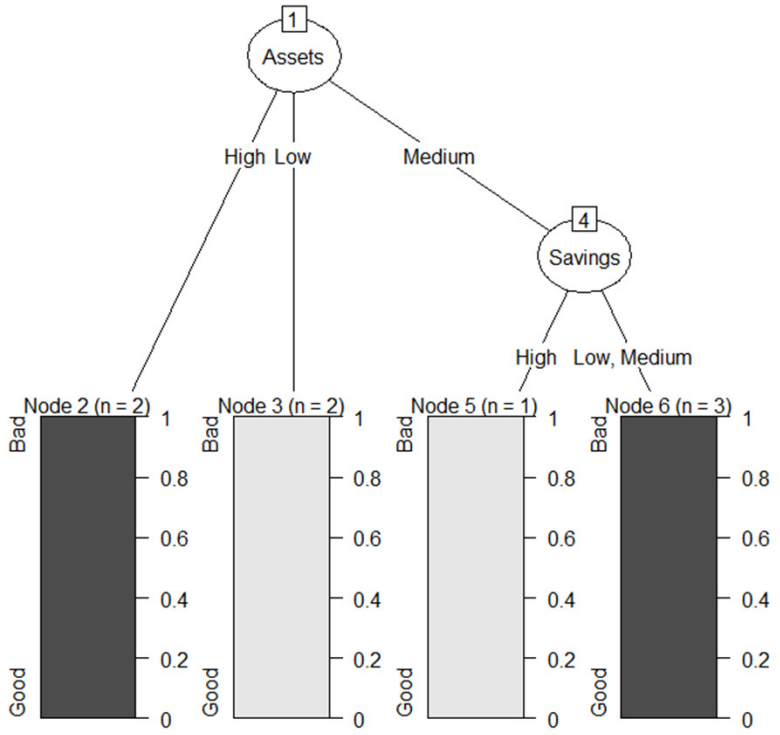
- Misclassified records: 10, 11, 12
- error rate = 3/7 = 0.43

Creating and testing the C4.5 decision tree in R

```
#install.packages("C50")
#library("C50")
d<-read.csv(file="credits.csv", stringsAsFactors =TRUE )

train_indices<-c(1:8)
d.train <- d[train_indices, 1:3] #!!!
d.train.class<- d[train_indices,4]
d.test <- d[-train_indices,1:3]
d.test.class<- d[-train_indices,4]

model<-C5.0(x=d.train,y=d.train.class,control=C5.0Control(CF=1.0,minCases = 1)) #growing the tree by the C5.0 algorithm, the result is provided as a decision tree, to create decision rules  set parameter rules=TRUE

summary(model) #shows the details of the model either in the form of a tree or decision rules
plot(model) #plots the tree model

pred <- predict(model, d.test,type="class")
pred

error_rate<-mean(pred != d.test.class)
error_rate
```

C5.0 results

```
Decision tree:

Assets = High: Good (2)
Assets = Low: Bad (2)
Assets = Medium:
:...Savings = High: Bad (1)
    Savings in {Low,Medium}: Good (3)


Rules:

Rule 1: (2, lift 2.0)
        Assets = Low
        ->  class Bad  [0.750]

Rule 2: (1, lift 1.8)
        Savings = High
        Assets = Medium
        ->  class Bad  [0.667]

Rule 3: (3, lift 1.3)
        Savings in {Low, Medium}
        Assets = Medium
        ->  class Good  [0.800]

Rule 4: (2, lift 1.2)
        Assets = High
        ->  class Good  [0.750]
```



Customer	Savings	Assets	Income	Credit Risk
1	Medium	High	75	Good
2	Low	Low	50	Bad
3	Hight	Medium	25	Bad
4	Medium	Medium	50	Good
5	Low	Medium	100	Good
6	Hight	High	25	Good
7	Low	Low	25	Bad
8	Medium	Medium	75	Good

Experiment. Discovering overfitting by using the k-fold cross validation

- In the following experiment, we investigate how the error rate of a model (a decision tree) is influenced by the complexity (size) of the model.

- The size of a decision tree is controlled by **prunning parameters.** In the experiment, we use the prunning parameters dealing with the number of records in leaves.

- Data set used in the experiment: “heart_disease_male.csv”.

age	chest_pain	rest_bpress	blood_sugar	rest_electro	max_heart_rate	exercice_angina	disease
43	atyp_angina	120	f	normal	160	yes	negative
39	non_anginal	160	t	normal	160	no	negative
39	non_anginal	160	f	normal	146	no	negative
42	asympt	140	f	normal	130	no	negative
49	asympt	140	f	normal	135	no	negative
50	non_anginal	140	f	st_t_wave_abnormalit	170	no	negative
59	atyp_angina	140	t	normal	140	no	negative
54	non_anginal	120	f	left_vent_hyper	135	no	negative
59	atyp_angina	140	f	normal	150	no	negative
56	atyp_angina	190	f	normal	106	no	negative
52	atyp_angina	140	f	normal	138	yes	negative
60	non_anginal	180	f	normal	100	no	negative
55	atyp_angina	130	f	normal	110	no	negative
57	atyp_angina	130	f	normal	120	no	negative
38	non_anginal	120	f	normal	185	no	negative

Experiment design

- The following pruning parameters are used:
  - for CART:
    - `minsplit` (the minimal size of a data set in a node which is splitted);
    - the values of `minsplit` are set to: 2, 8, 14, 20, 26, 32, 38, 44 and 50.
  - for C4.5:
    - `minCases` (the maximal size of a data set in a node which can be left without splitting);
    - the values of `minCases` are set to 1, 4, 7, 10, 13, 16, 19, 22 and 25.

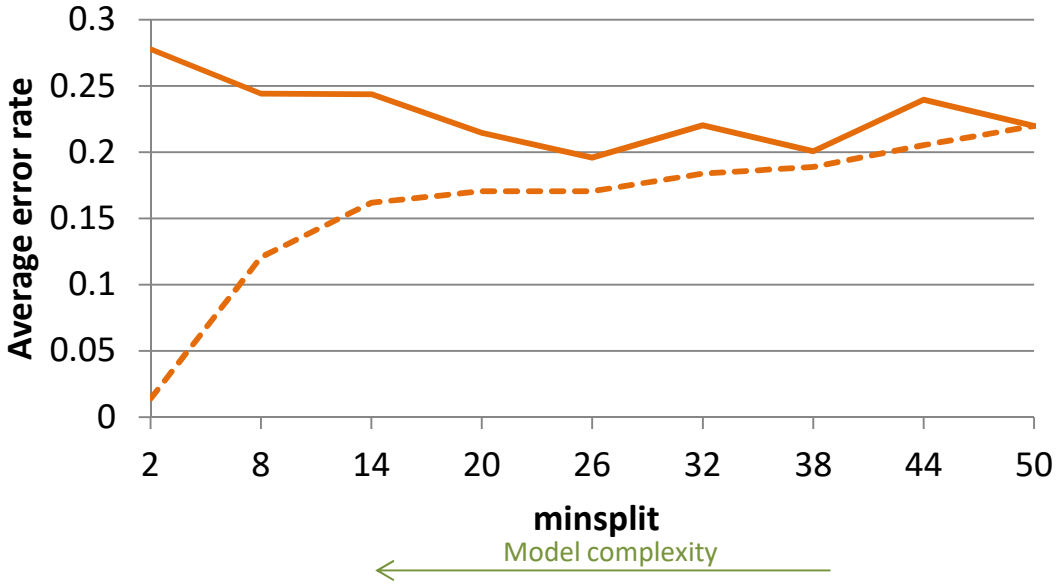
Other pruning parameters , **CF** (C4.5) and **cp** (CART), are set as shown below:

```
C5.0(x=…,y=…,control=C5.0Control(minCases = …,CF=1.0))
rpart(…, data = …, method =…, control=rpart.control(minsplit=… cp=0.0))
```

- The k-fold cross validation is performed for every value of a prunning parameter.

- In the k-fold cross validation:
  - k set to 10;
  - the **error rates** are determined for both the **test set** and the **training set**.

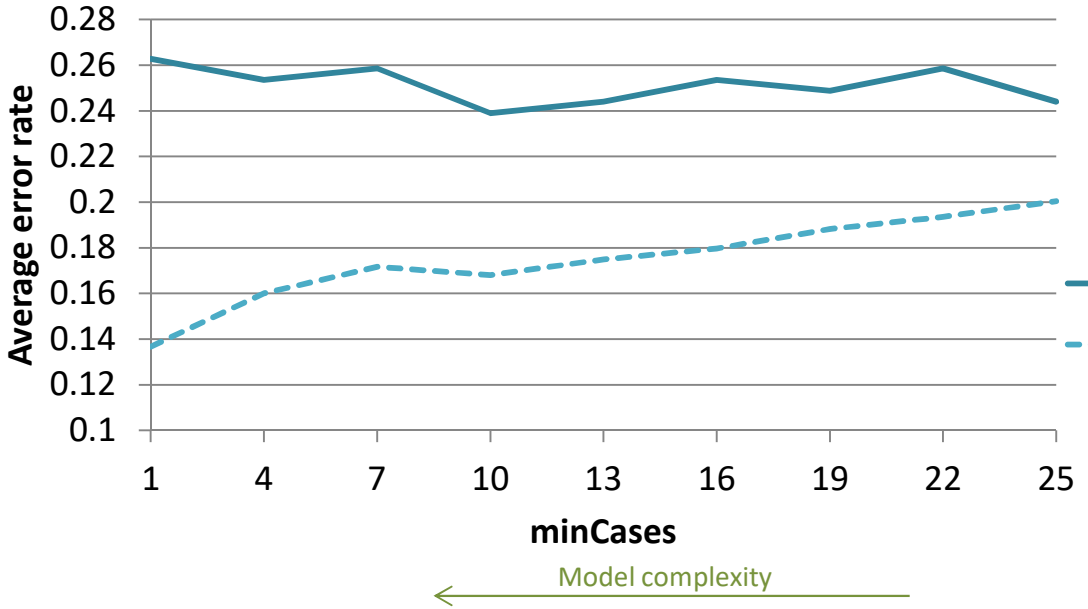
Experiment results



CART on the test set

CART on the training set

- The overfitting (for the test data set) can be observed for small values of minsplit and minCases, i.e. when the model has a great size). The best average error rate is achieved for smaller models (for minsplit = 26 (CART) and for minCases = 10 (C4.5)).
- For the training set, the smallest average error rate is observed when the model is of great size, i.e. when it is "fitted" to the data.

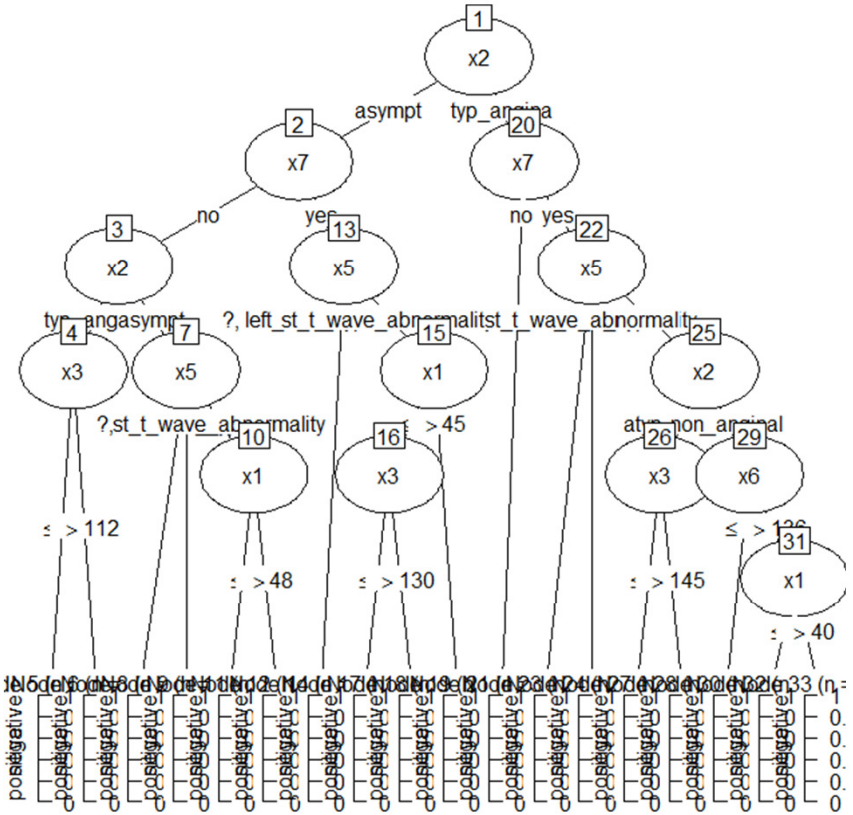


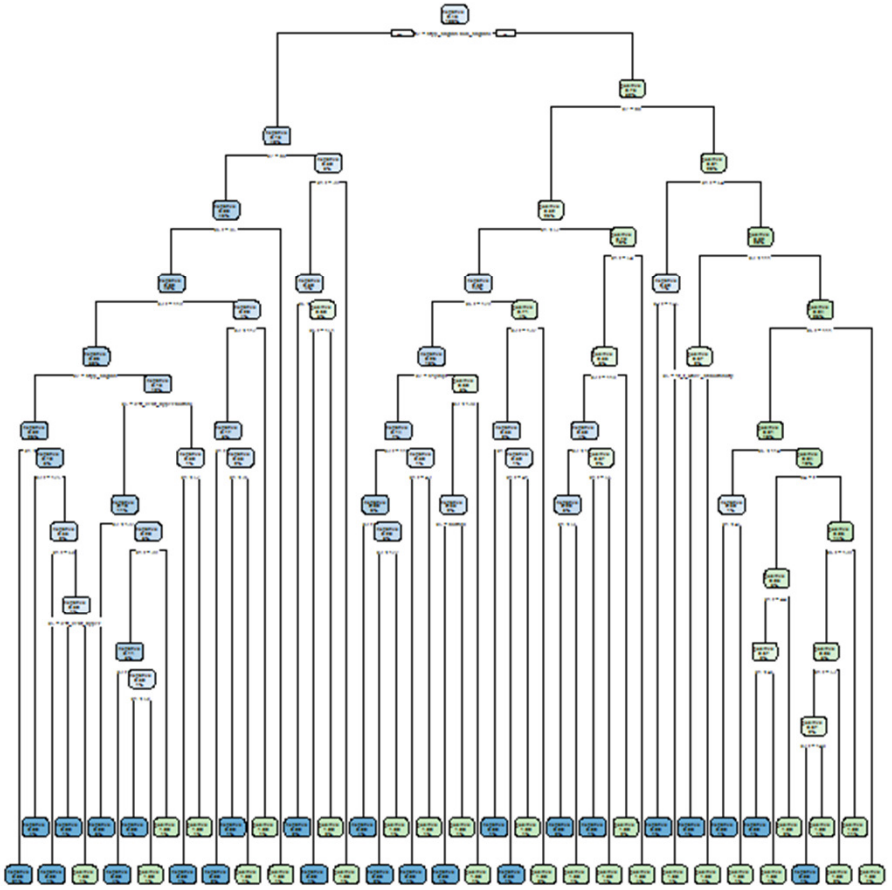
C4.5 on the test set

C4.5 on the training set

Average error rate is the average of 10 error rates resulting from the 10-fold cross validation.

Decision trees

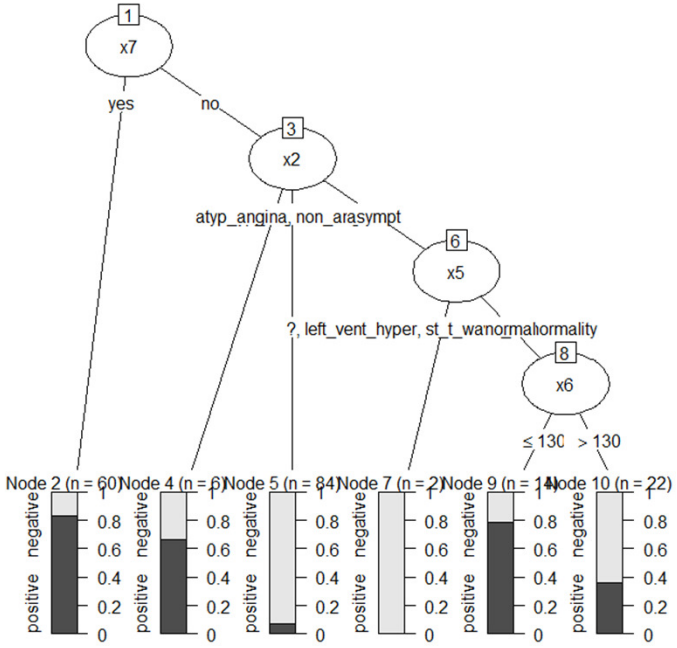


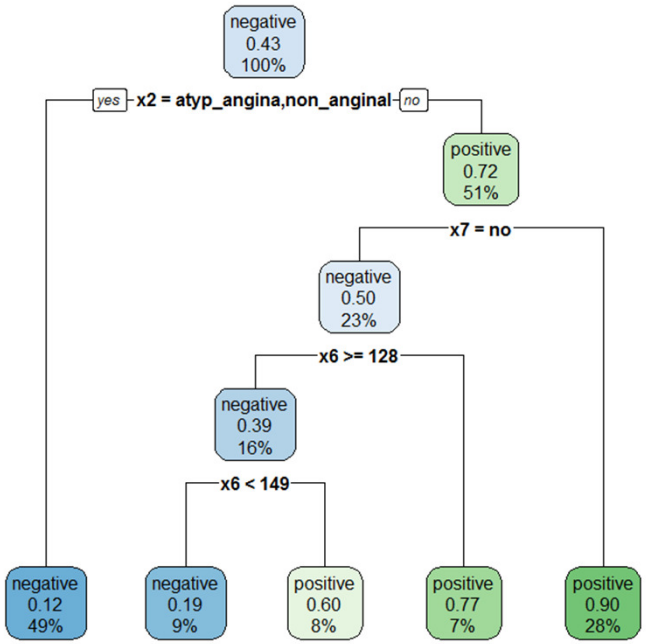


- C4.5 `minCases = 1`

CART `minsplit = 2`

Decision trees





- C4.5 minCases = 13

CART minsplit = 26