6. Classification Trees

For feature vectors of real-value and discrete-valued numbers, and there is a natural measure of distance between vectors.

For example, two real-valued vectors

$$\mathbf{x}_1 = [0.1 \quad 0.7]^T$$

$$\mathbf{x}_2 = [0.2 \quad 0.8]^T$$

The Euclidean distance between the two vectors directly reflects the similarity between the two vectors:

$$d = \sqrt{(0.2 - 0.1)^2 + (0.8 - 0.7)^2}$$

Some applications involve **nominal data**: data used for naming or labelling variables without any quantitative value. For example,

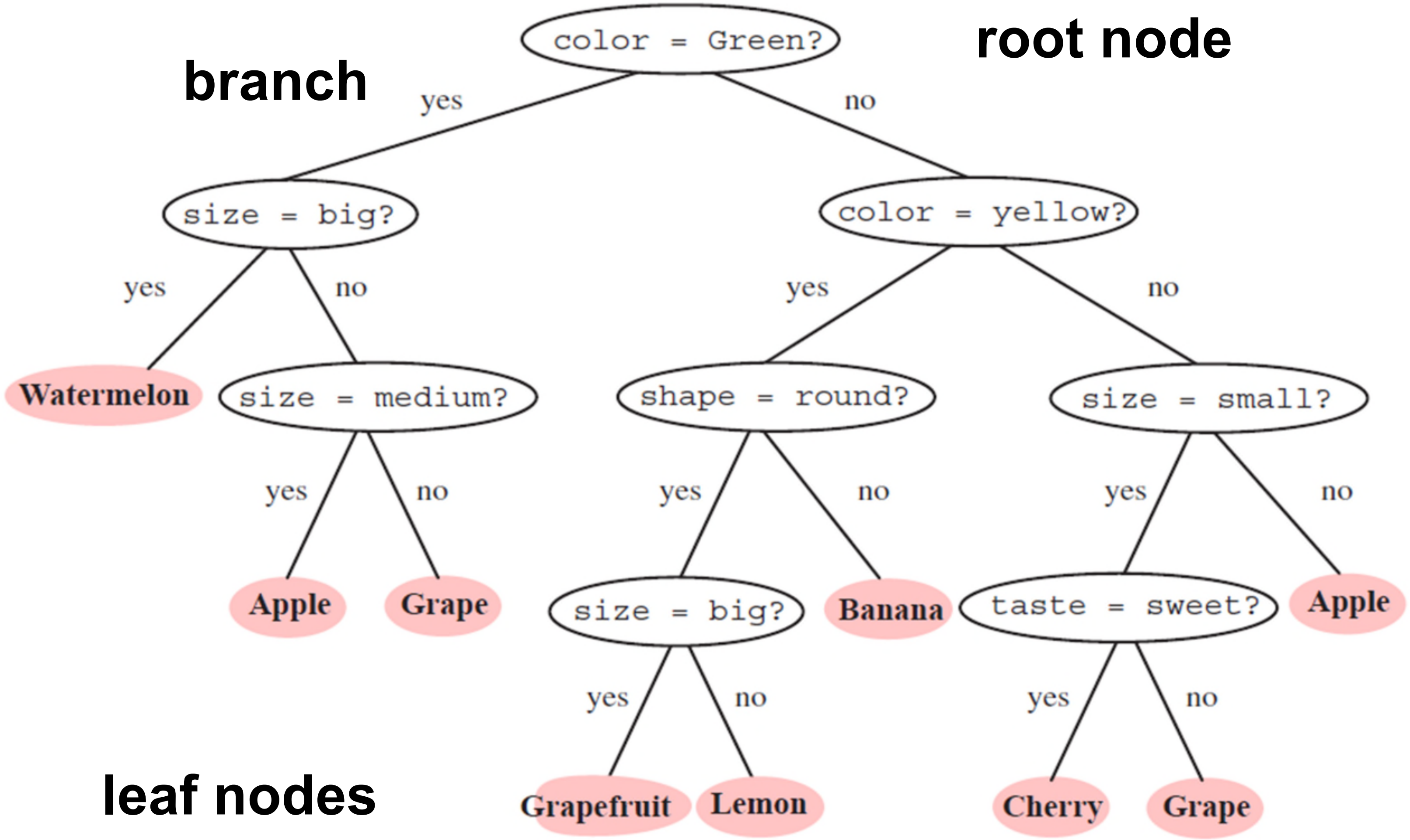
(i) Attribute “transportation mode”, may take values of “train”, “car”, “air”, “sea” etc.

(ii) Attribute “colour” may take values of “red”, “yellow”, ”blue”, “green” etc.

If we use 1, 2, 3, 4 etc to denotes them, respectively, the Euclidean distance does not reflect the similarity. As a result, the conventional distance metrics-based classifiers may not work well

To tackle the problem, we may use nonmetric method such as classification tree.

What is a classification tree?



branch

root node

leaf nodes

As shown above, a classification tree is a flowchart with tree structure. It classifies a pattern through a sequence of questions.

❑ The questions asked at each node concern a particular property (i.e. feature or attribute) of the sample.

❑ All of the questions are asked in a “yes/no”, or “true/false”, or “value(property) *is an element of* set of values” style.

Why classification trees?

The simple classification tree illustrates one benefit of trees: interpretability. Such interpretability has two manifestations:

- ❑ We can easily interpret the decision

- ❑ Classification trees provide a natural way to incorporate prior knowledge from human experts

How to build a classification tree?

Now we turn to the key problem: how to use training data to create a classification tree?

To build a classification tree, basically, we need to answer the following questions:

1) How many questions to ask (i.e. how many splits) at each node?

2) Which property should be used at a node?

3) When should be a node be declared as a leaf node?

4) If the tree becomes “too large,” how can it be made smaller and simpler, i.e., pruned?

5) If a leaf node is impure, how should the category label be assigned?

- **Number of splits at each node**

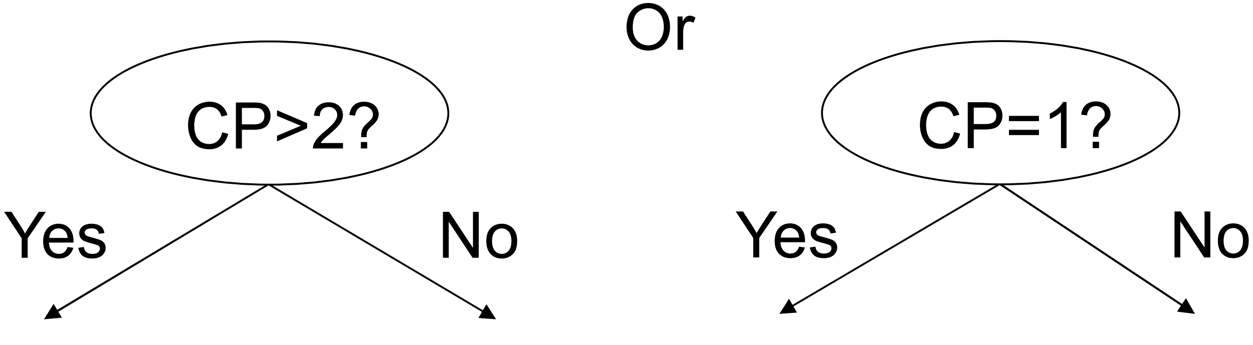
Each node asks one or more questions, which split a subset of the training data. The root node splits the full training set; each successive nodes splits a subset of the data.

The number of splits at a node is closely related to question 1, specifying *which* particular split will be made at a node. In general, the number of splits is set by the designer, and could vary throughout the tree.

The number of links descending from a node is sometimes called the node’s *branching factor* or *branching ratio.*

For example, if an attribute CP has 4 values: 1,2,3,4

(i) Binary splits



CP>2?

Yes

No

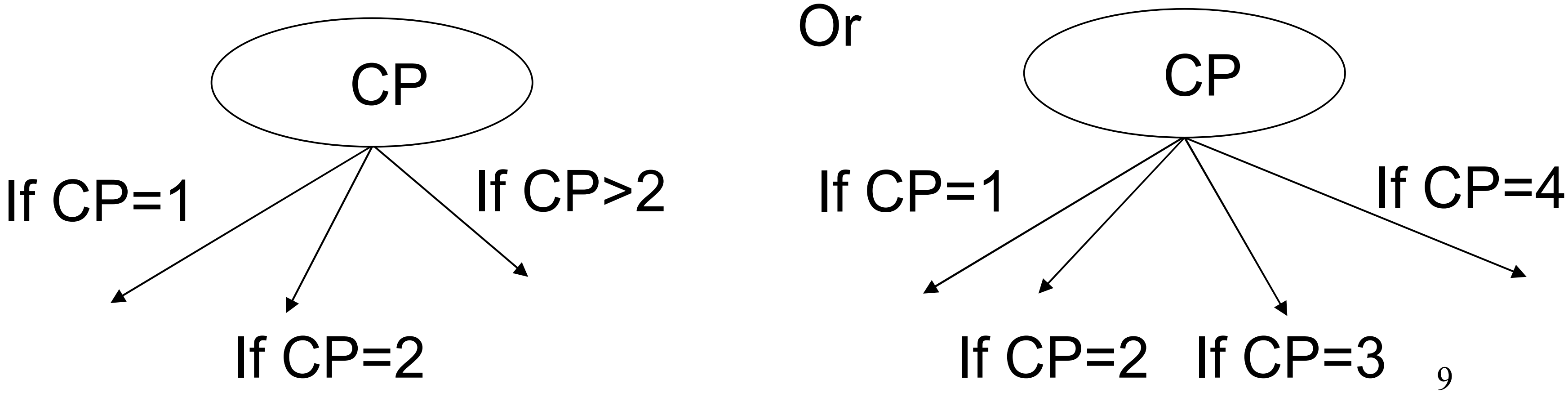
Or

CP=1?

Yes

No

(ii) Multiple splits



CP

If CP=1

If CP=2

If CP>2

Or

CP

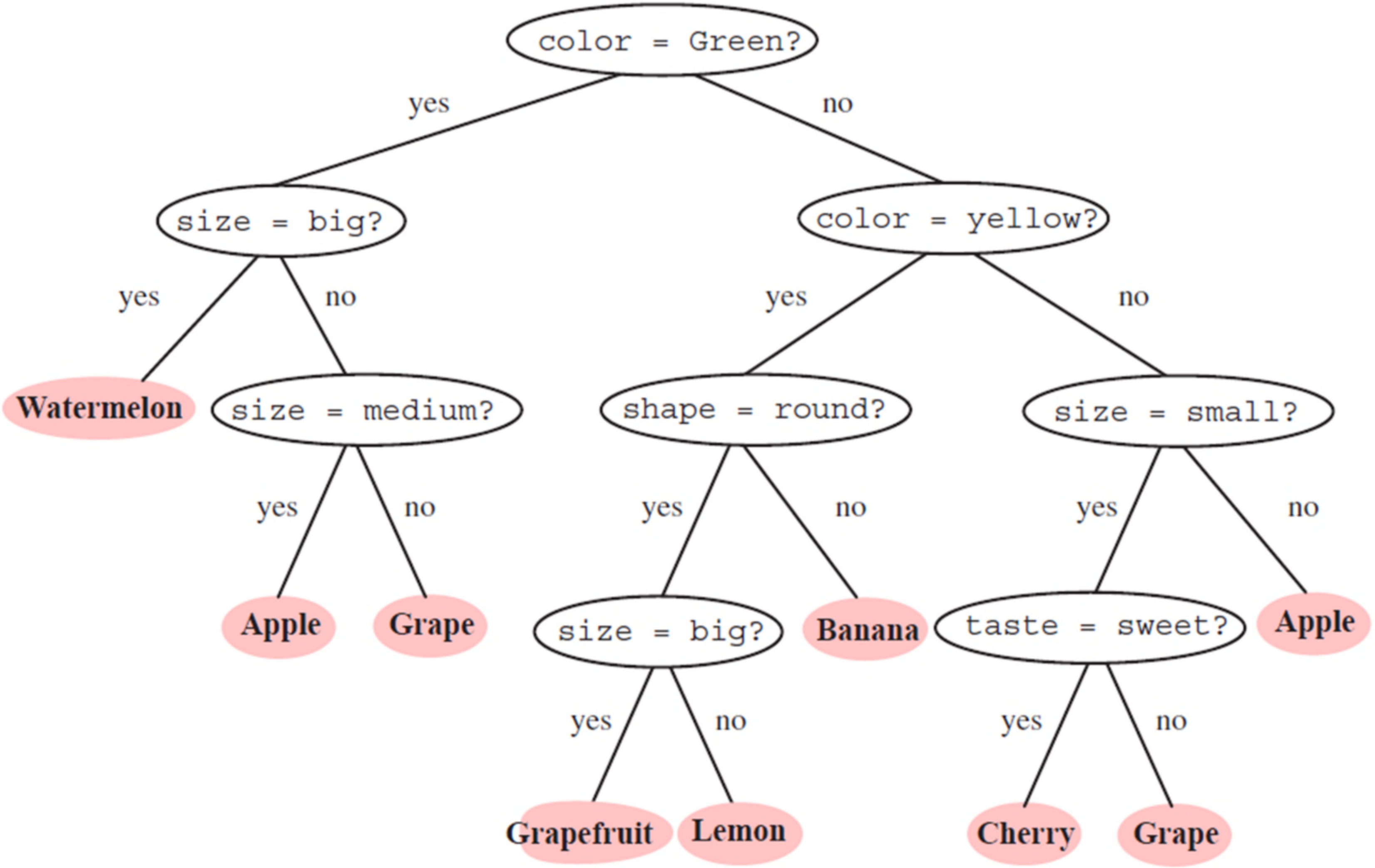
If CP=1

If CP=2

If CP=3

If CP=4

In a binary tree, all nodes use binary splits



- **Query selection and node impurity**

The fundamental principle in tree creation is simplicity: we prefer decisions that lead to a simple, compact tree with few nodes.

To this end, we seek a property test T at each node N that makes the data reaching the immediate descendent nodes as “pure” as possible. In formalizing this notion, the impurity rather than purity of a node is more convenient.

Several different measures of impurity can be used. We next introduce some of the most popular ones.

Let $i(N)$ denote the impurity of node *N*. In all cases, we want $i(N)$ to be 0 if all of the samples that reach the node bear the same category label, and to be large if the categories are equally represented

(i) Entropy impurity

The most popular measure is the entropy impurity, which is defined as follows:

$$i_E(N) = -\sum_j P(\omega_j) \log_2 P(\omega_j)$$

where $P(\omega_j)$ is the fraction of samples at node N that are in category ω_j.

Example: we have the following dataset at node N:

Class	No. of Samples
Class 1	10
Class 2	90

Then we have:

$$P(\omega_1) = 0.1 \qquad P(\omega_2) = 0.9$$

Then the entropy impurity measure is:

$$i_E = -0.1 \times \log_2(0.1) - 0.9 \times \log_2 0.9 = 0.469$$

For a dataset with 50:50 distribution, then the entropy impurity measure is:

$$i_E = -0.5 \times \log_2(0.5) - 0.5 \times \log_2 0.5 = 1$$

(ii) Variance and Gini impurity

Given the desire to have zero impurity when the node represents only patterns of a single category, the simplest form is:

$$i_{Var}(N) = P(\omega_1)P(\omega_2)$$

A generalization of the variance impurity, applicable to two or more category, is the *Gini impurity*:

$$i_{Gini}(N) = \sum_{j \neq k} P(\omega_j)P(\omega_k) = \frac{1}{2}\left[1 - \sum_j P^2(\omega_j)\right]$$

Consider the following dataset at node N:

Class	No. of Samples
Class 1	10
Class 2	90

Then we have:

$$P(\omega_1) = 0.1 \qquad P(\omega_2) = 0.9$$

Then the variance impurity measure is:

$$i_{Var} = P(\omega_1)P(\omega_2) = 0.1 \times 0.9 = 0.09$$

The Gini impurity measure is:

$$i_{Gini} = \frac{1}{2}[1 - P(\omega_1)^2 - P(\omega_2)^2] = \frac{1}{2}[1 - 0.1^2 - 0.9^2] = 0.09$$

(iii) Misclassification impurity

This impurity measures the minimum probability that a training sample would be misclassified at node *N:*

$$i_{MC}(N) = 1 - \max_j P(\omega_j)$$

For the above example,

$$i_{MC} = 1 - \max_j P\left(\omega_j\right) = 1 - 0.9 = 0.1$$

The entropy impurity is frequently used because of its computational simplicity and basis in information theory, Usually, the results are not affected much by the different impurity measures.

What attribute and what value should we choose for the property test at a node? An obvious heuristic is to choose the one that decreases the impurity as much as possible.

The drop in impurity is defined by:

$$\begin{aligned}\Delta i(N) &= i(N) - P_L i(N_l) - P_R i(N_R) \\ &= i(N) - [P_L i(N_l) + P_R i(N_R)]\end{aligned}$$

Where N_L and N_R are the left and right descendent nodes, $i(N_L)$ and $i(N_R)$ are their impurities, P_L and P_R are the fraction of data at node *N* that will go to N_L and N_R, respectively.

The maximum drop in impurity is equivalent to minimum impurity at the descendent nodes $[P_L i(N_l) + P_R i(N_R)]$

For multi-split, an intuitive extension of the binary-split impurity drop is as follow:

$$\Delta i(N) = i(N) - \sum_{k=1}^{B} P_k i(N_k)$$

Where P_k is the fraction of training data at node N that will be sent down the link to node N_k, B is the number of branches at node N, and

$$\sum_{k=1}^{B} P_k = 1$$

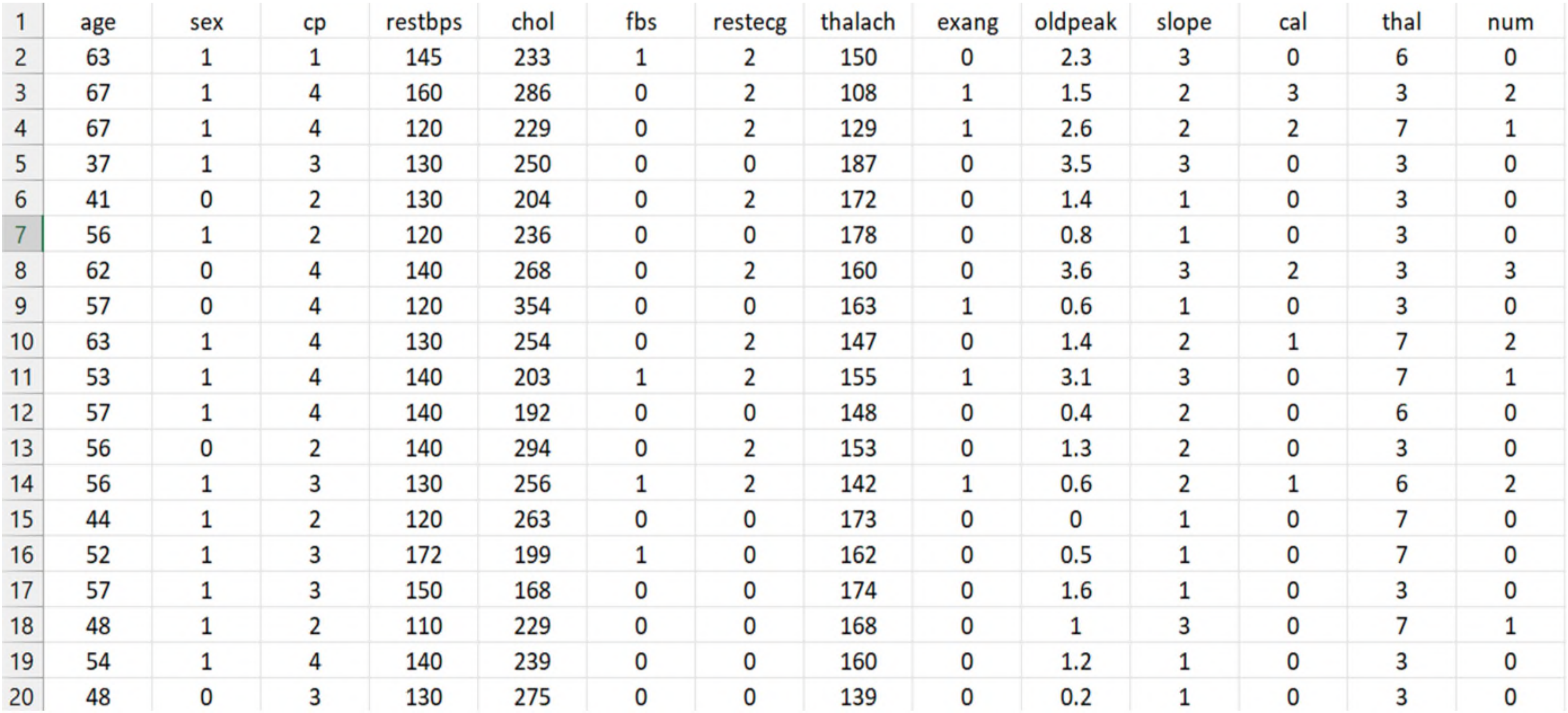
It is found that this measure favours large B whether or not the large splits in fact represent meaningful structure in the data.

To avoid this drawback, we may scale the above measure as follows:

$$\Delta i'(N) = \frac{\Delta i(N)}{-\sum_{k=1}^{B} P_k \log_2 P_k}$$

The best test value is the choice that maximizes $\Delta i(N)$ for binary-split or $\Delta i'(N)$ for multi-split. The above measure is also called gain ratio.

Next, we use a real example (heart disease data) to illustrate the node selection problem (please refer to the Excel file Data_HeartDisease uploaded for the data).



1	age	sex	cp	restbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	cal	thal	num
2	63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
3	67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
4	67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
5	37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
6	41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
7	56	1	2	120	236	0	0	178	0	0.8	1	0	3	0
8	62	0	4	140	268	0	2	160	0	3.6	3	2	3	3
9	57	0	4	120	354	0	0	163	1	0.6	1	0	3	0
10	63	1	4	130	254	0	2	147	0	1.4	2	1	7	2
11	53	1	4	140	203	1	2	155	1	3.1	3	0	7	1
12	57	1	4	140	192	0	0	148	0	0.4	2	0	6	0
13	56	0	2	140	294	0	2	153	0	1.3	2	0	3	0
14	56	1	3	130	256	1	2	142	1	0.6	2	1	6	2
15	44	1	2	120	263	0	0	173	0	0	1	0	7	0
16	52	1	3	172	199	1	0	162	0	0.5	1	0	7	0
17	57	1	3	150	168	0	0	174	0	1.6	1	0	3	0
18	48	1	2	110	229	0	0	168	0	1	3	0	7	1
19	54	1	4	140	239	0	0	160	0	1.2	1	0	3	0
20	48	0	3	130	275	0	0	139	0	0.2	1	0	3	0

The dataset contains 303 samples from 2 classes:

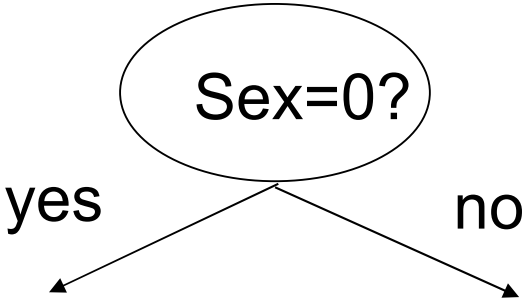
Class	No. of Samples
Class 1 (with heart attack)	139
Class 2 (without heart attack)	164

Each sample is represented by 13 attributes:

1. age
2. sex
3. cp
4. trestbps
5. chol
6. fbs
7. restecg)
8. thalach
9. exang
10. oldpeak
11. slope
12. ca
13. thal

We first decide the root node.

If “sex” is selected



Class	Samples
Class 1	25
Class 2	72

Class	Samples
Class 1	114
Class 2	92

Gini impurity of left and right branch:

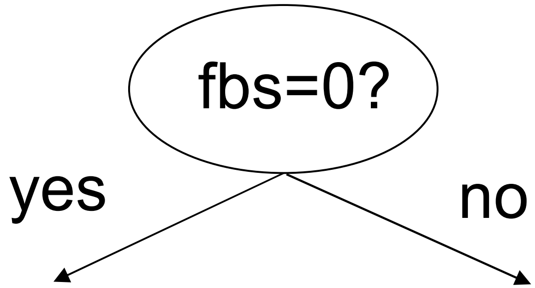
$$i_{left} = \frac{1}{2}[1 - P(\omega_1)^2 - P(\omega_2)^2] = \frac{1}{2}\left[1 - \left(\frac{25}{25+72}\right)^2 - \left(\frac{72}{25+72}\right)^2\right]=0.1913$$

$$i_{right} = \frac{1}{2}[1 - P(\omega_1)^2 - P(\omega_2)^2] = \frac{1}{2}\left[1 - \left(\frac{114}{114+92}\right)^2 - \left(\frac{92}{114+92}\right)^2\right]=0.2471$$

Then the Gini impurity of feature “sex” is obtained as:

$$
\begin{aligned}
i_{sex} &= P_L i_{left} + P_R i_{right} \\
&= \frac{97}{303} \times 0.1913 + \frac{206}{303} \times 0.2471 = 0.2276
\end{aligned}
$$

If “fbs” is selected



Class	Samples
Class 1	117
Class 2	141

Class	Samples
Class 1	22
Class 2	23

Gini impurity of left and right branch:

$$i_{left} = \frac{1}{2}[1 - P(\omega_1)^2 - P(\omega_2)^2] = \frac{1}{2}\left[1 - \left(\frac{117}{117+141}\right)^2 - \left(\frac{141}{117+141}\right)^2\right] = 0.2478$$

$$i_{right} = \frac{1}{2}[1 - P(\omega_1)^2 - P(\omega_2)^2] = \frac{1}{2}\left[1 - \left(\frac{22}{22+23}\right)^2 - \left(\frac{22}{22+23}\right)^2\right] = 0.2499$$

The impurity of “fbs” (i.e. fasting blood sugar) is thus obtained as:

$$
\begin{aligned}
i_{fbs} &= P_L i_{left} + P_R i_{right} \\
&= \frac{258}{303} \times 0.2478 + \frac{45}{303} \times 0.2499 = 0.2481
\end{aligned}
$$

If “age” is selected

“age” is a continuous variable. How to find the splitting point and impurity measure for continuous variable?

(1) First sort the data in ascending order:

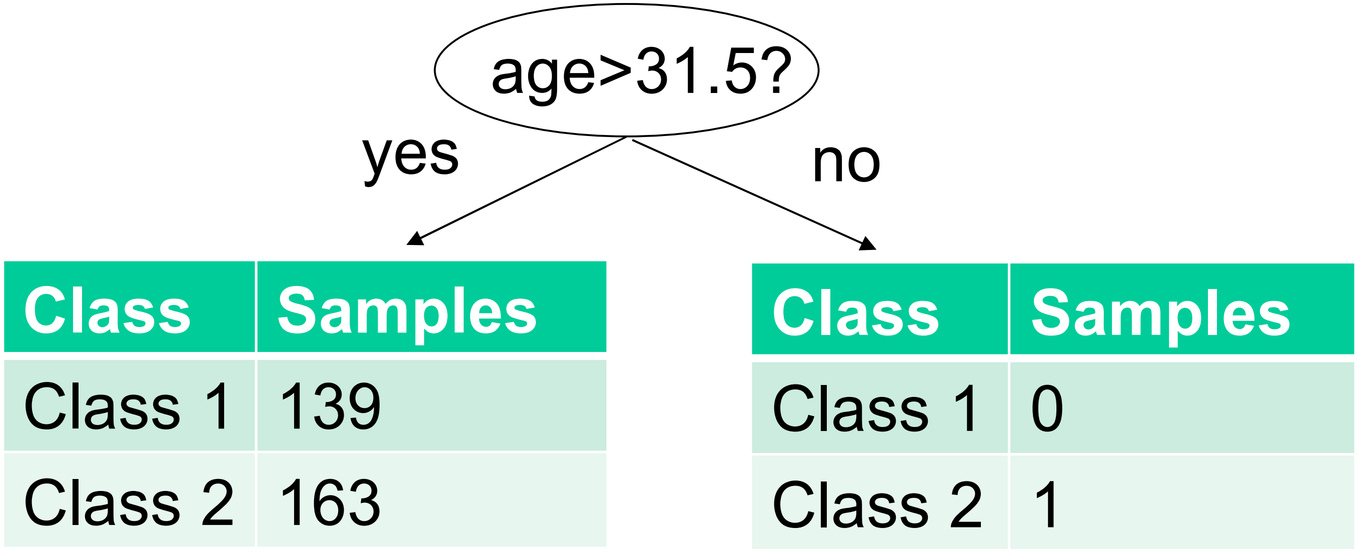
$$29, 34, 35, 37, \dots.$$

(2) Find the middle point of two adjacent values

$$31.5, 34.5, 36, \dots$$

(3) Take each of the middle point as the candidate splitting point and calculate the corresponding impurity. The one leading to the lowest impurity will be selected as the splitting point

If 31.5 is the splitting point of “age”



age>31.5?

yes

no

Class	Samples
Class 1	139
Class 2	163

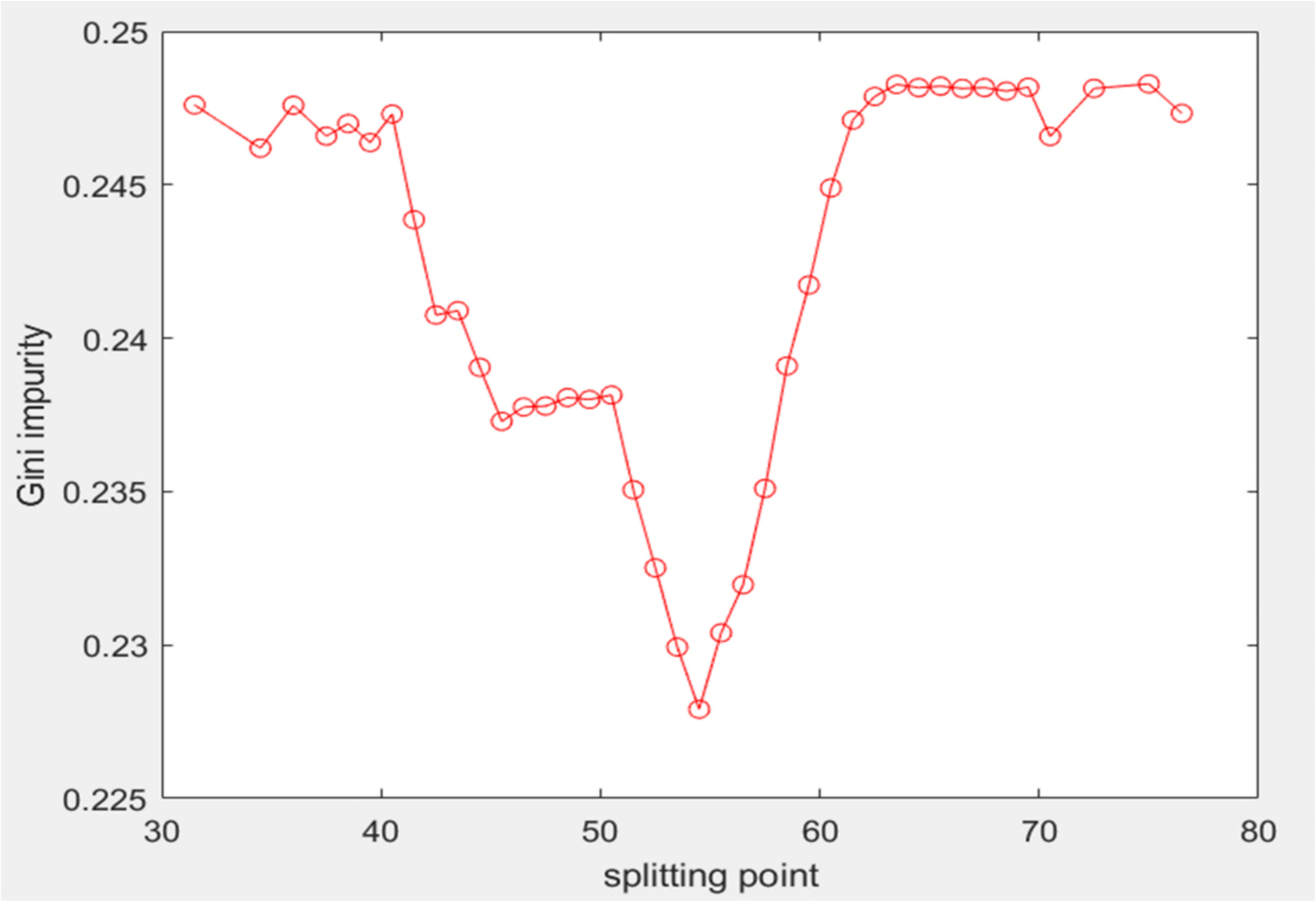
Class	Samples
Class 1	0
Class 2	1

$$i_{left} = \frac{1}{2}[1 - P(\omega_1)^2 - P(\omega_2)^2] = \frac{1}{2}\left[1 - \left(\frac{139}{139+163}\right)^2 - \left(\frac{163}{139+163}\right)^2\right] = 0.2484$$

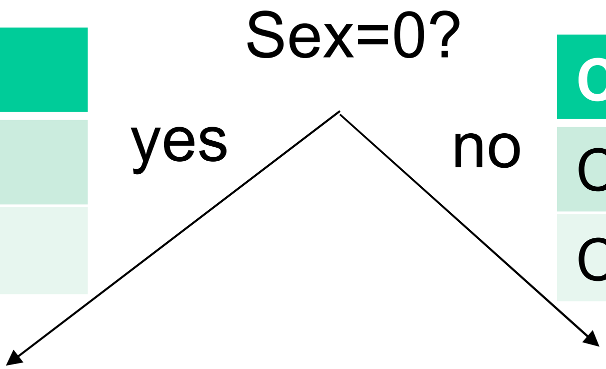
$$i_{right} = \frac{1}{2}[1 - P(\omega_1)^2 - P(\omega_2)^2] = \frac{1}{2}\left[1 - \left(\frac{0}{0+1}\right)^2 - \left(\frac{1}{0+1}\right)^2\right] = 0$$

$$\begin{aligned} i_{31.5} &= P_L i_{left} + P_R i_{right} \\ &= \frac{302}{302+1} \times 0.2484 + \frac{1}{302+1} \times 0 = 0.2476 \end{aligned}$$

Similarly, we can obtain $i_{34.5}$, i_{36},… The lowest Gini impurity 0.228 is achieved when the splitting point is set to 54.5.



Similarly, we can obtain the Gini impurities of other features. Assume the Gini impurity of “sex” is the lowest, then it is selected as the root node, and the result is as follow:



Sex=0?

yes

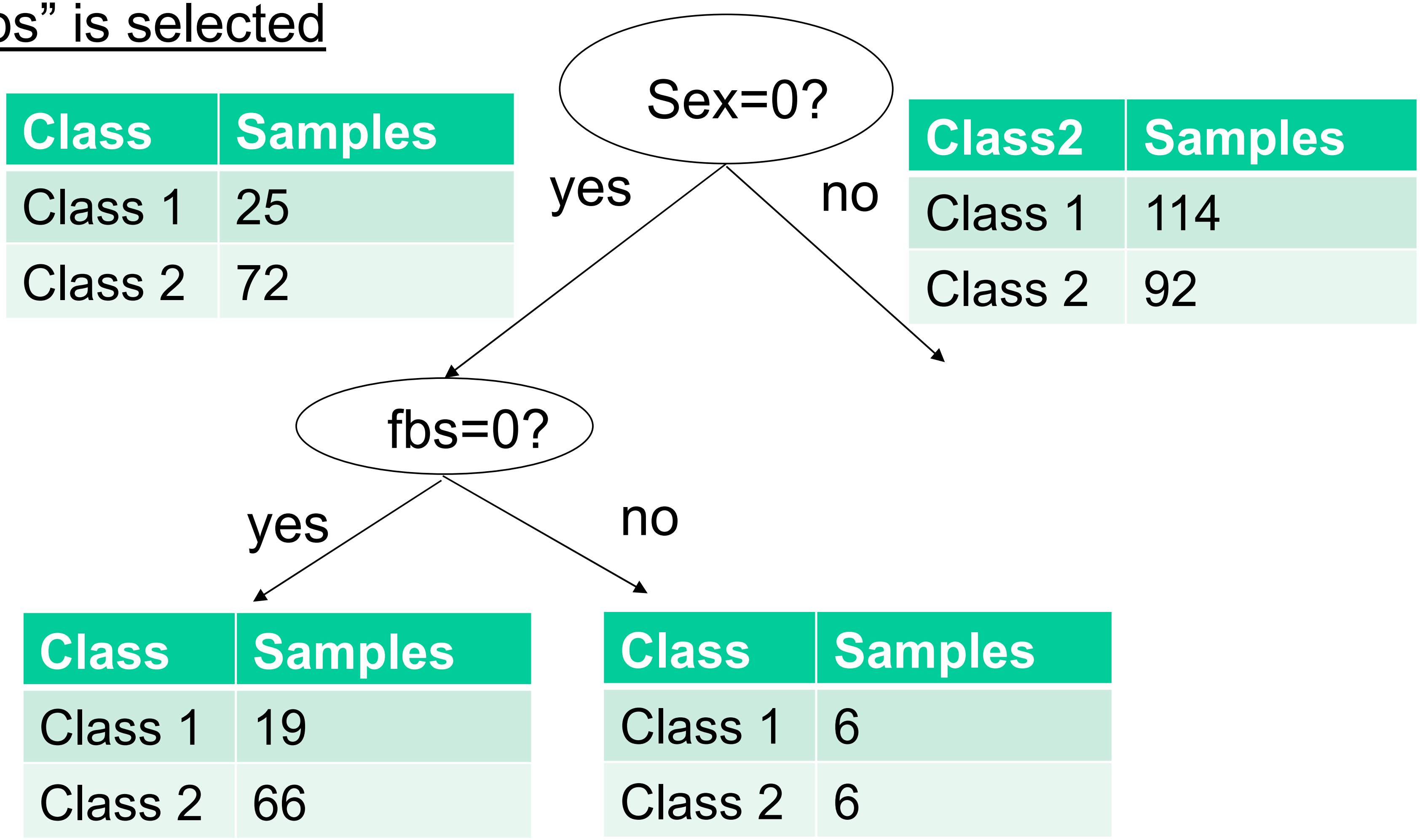
no

Class	Samples
Class 1	25
Class 2	72

Class2	Samples
Class 1	114
Class 2	92

Next, we continue to grow the classification tree. We first look at the left branch.

If “fbs” is selected



Sex=0?

yes

Class	Samples
Class 1	25
Class 2	72

no

Class2	Samples
Class 1	114
Class 2	92

fbs=0?

yes

Class	Samples
Class 1	19
Class 2	66

no

Class	Samples
Class 1	6
Class 2	6

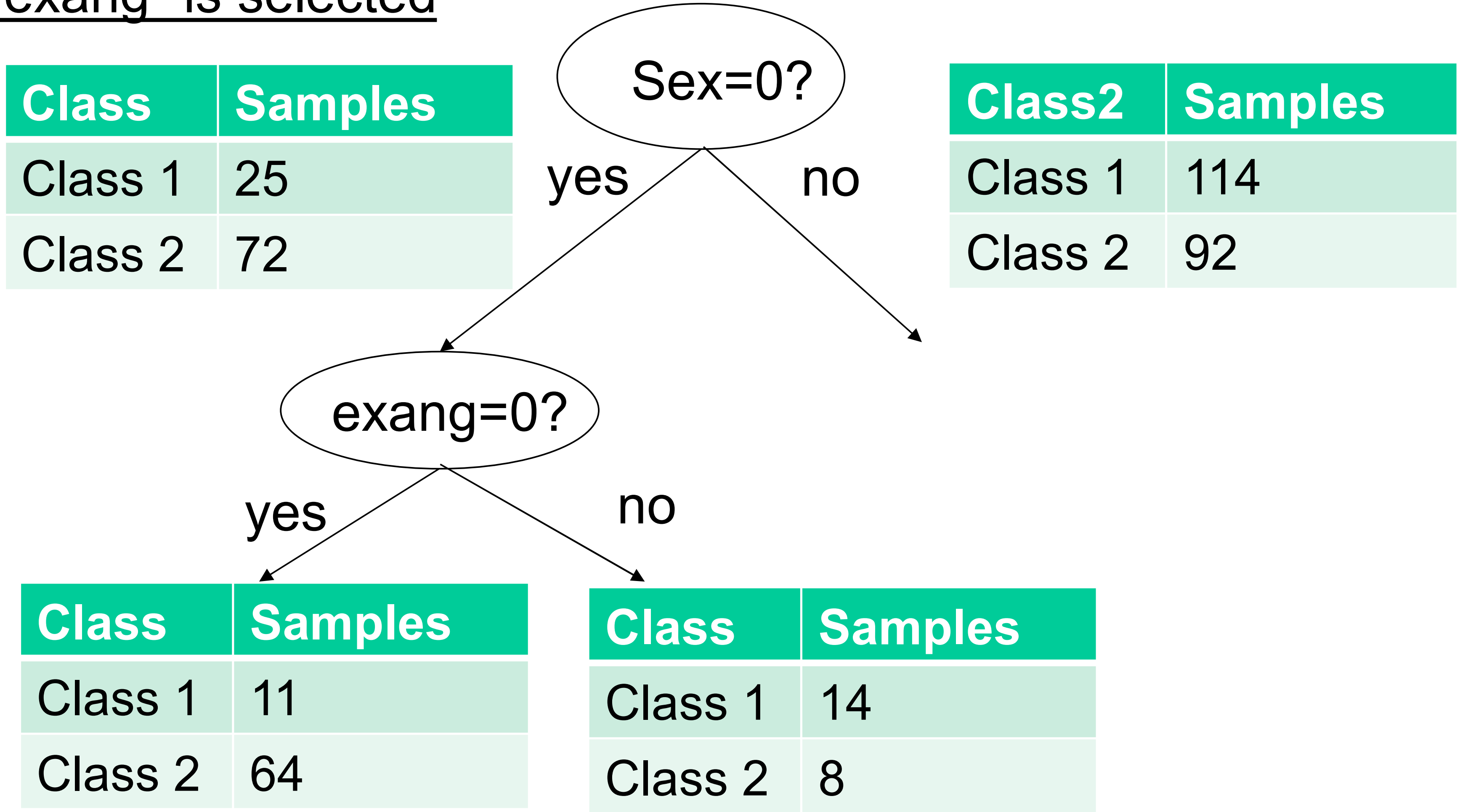
$$i_{left} = \frac{1}{2}[1 - P(\omega_1)^2 - P(\omega_2)^2] = \frac{1}{2}\left[1 - \left(\frac{19}{19+66}\right)^2 - \left(\frac{66}{19+66}\right)^2\right] = 0.1736$$

$$i_{right} = \frac{1}{2}[1 - P(\omega_1)^2 - P(\omega_2)^2] = \frac{1}{2}\left[1 - \left(\frac{6}{6+6}\right)^2 - \left(\frac{6}{6+6}\right)^2\right] = 0.25$$

The impurity of “fbs” (i.e. fasting blood sugar) is thus obtained as:

$$
\begin{aligned}
i_{fbs} &= P_L i_{left} + P_R i_{right} \\
&= \frac{85}{85+12} \times 0.1736 + \frac{12}{85+12} \times 0.25 = 0.1831
\end{aligned}
$$

If “exang” is selected



Sex=0?

yes

no

Class	Samples
Class 1	25
Class 2	72

Class2	Samples
Class 1	114
Class 2	92

exang=0?

yes

no

Class	Samples
Class 1	11
Class 2	64

Class	Samples
Class 1	14
Class 2	8

$$i_{left} = \frac{1}{2}[1 - P(\omega_1)^2 - P(\omega_2)^2] = \frac{1}{2}\left[1 - \left(\frac{11}{11+64}\right)^2 - \left(\frac{64}{11+64}\right)^2\right] = 0.1252$$

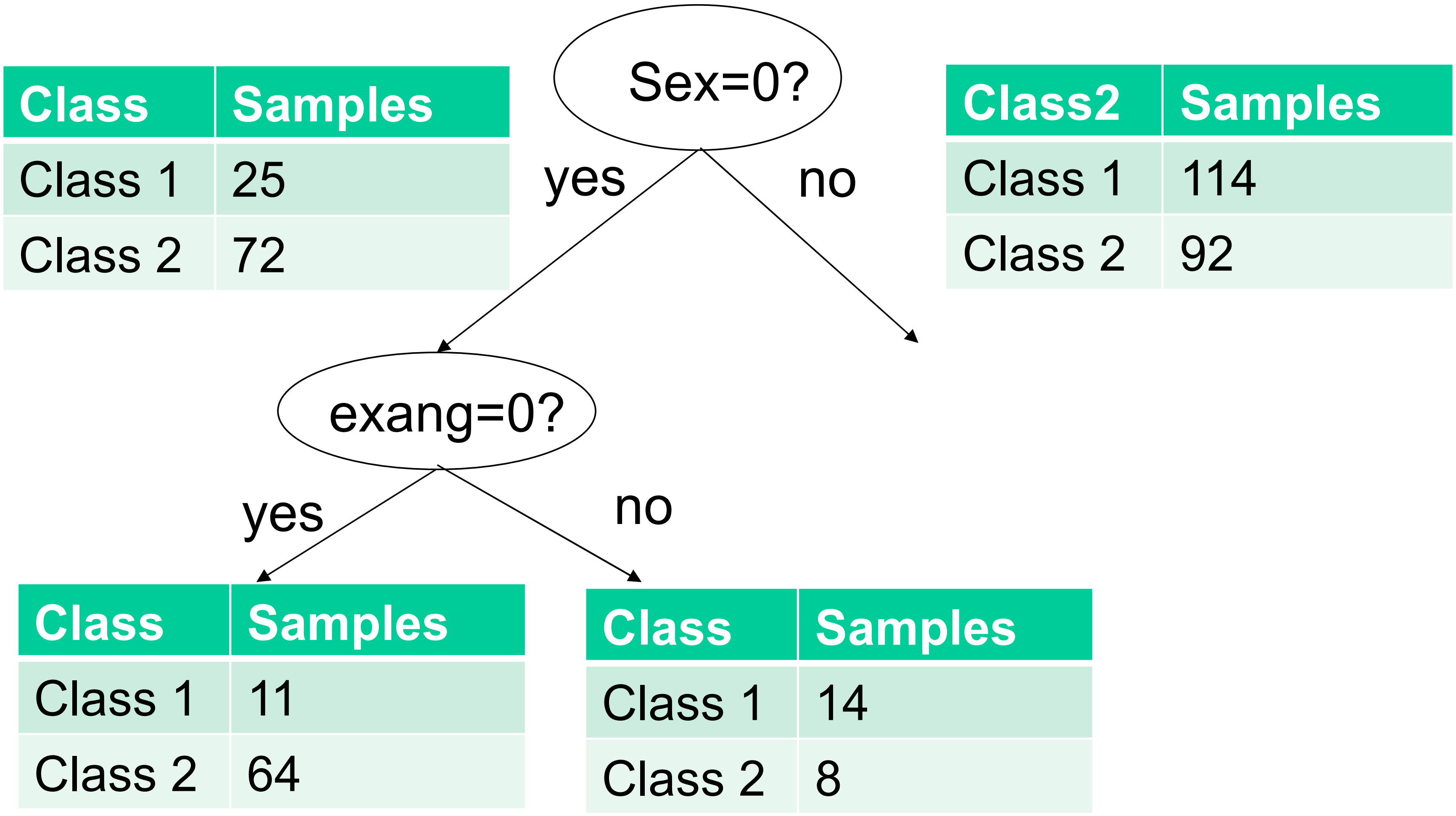
$$i_{right} = \frac{1}{2}[1 - P(\omega_1)^2 - P(\omega_2)^2] = \frac{1}{2}\left[1 - \left(\frac{14}{14+8}\right)^2 - \left(\frac{8}{14+8}\right)^2\right] = 0.2314$$

The impurity of “exang” (i.e. excise induced angina) is thus obtained as:

$$i_{exang} = P_L i_{left} + P_R i_{right}$$
$$= \frac{75}{75+22} \times 0.1252 + \frac{22}{75+2} \times 0.2314 = 0.1493$$

Obviously, “exang” has lower Gini impurity than “fbs”.

Similarly, we can obtain the Gini impurities of other features. Assume the Gini impurity of “exang” is the lowest, then it is selected as the descendent node of the left branch of root node “sex”.



Sex=0?

yes

Class	Samples
Class 1	25
Class 2	72

no

Class2	Samples
Class 1	114
Class 2	92

exang=0?

yes

Class	Samples
Class 1	11
Class 2	64

no

Class	Samples
Class 1	14
Class 2	8

- **When to stop splitting**

Consider now the problem of deciding when to stop splitting during the training of a binary tree. If we continue to grow the tree fully until each leaf node corresponds to the lowest impurity, then the data has typically been overfitted.

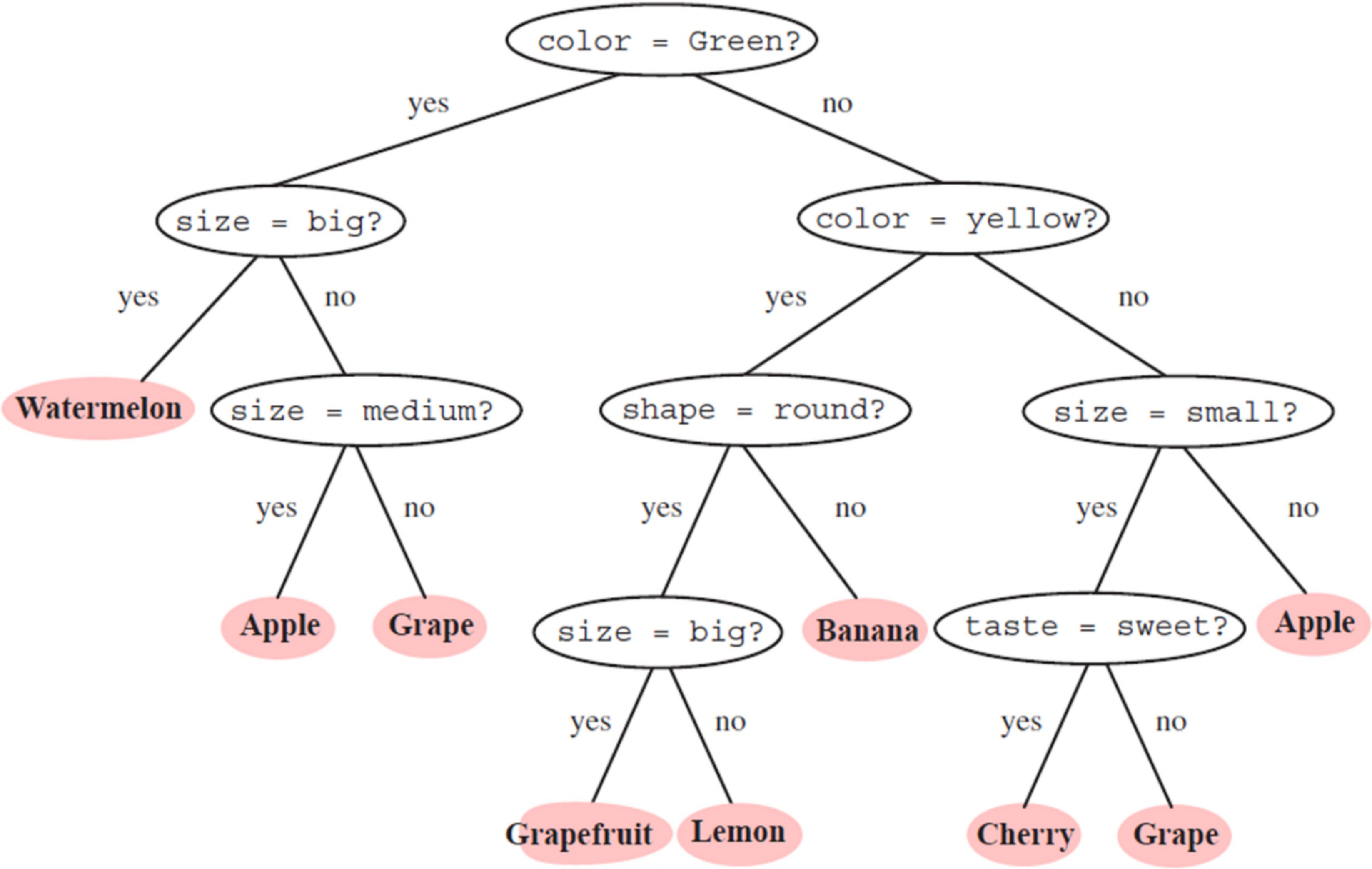
(1) In the extreme but rare case, each leaf corresponds to a single training data and the full tree is merely a convenient implementation of a lookup table; it thus cannot be expected to generalize well.

(2) Conversely, if splitting is stopped too early, then the error on the training data is not sufficiently low and hence performance may suffer.

How shall we decide when to stop splitting?

- One traditional approach is to use cross-validation.
  - ❑ The tree is trained using the training data
  - ❑ We continue splitting nodes in successive layers until the error on the validation data is minimized.

- Another method is to set a (small) threshold value in the reduction in impurity. Splitting is stopped if the best candidate split at a node reduces the impurity by less than the pre-set amount.
  - ❑ Unlike cross-validation, the tree is trained directly using all the training data.
  - ❑ Leaf nodes can lie in different levels of the tree.


color = Green?
yes
no
size = big?
color = yellow?
yes
no
yes
no
Watermelon
size = medium?
shape = round?
size = small?
yes
no
yes
no
yes
no
Apple
Grape
size = big?
Banana
taste = sweet?
Apple
yes
no
yes
no
Grapefruit
Lemon
Cherry
Grape

We can also stop splitting when a node has fewer than some threshold number of points, say 10, or some fixed percentage of the total training set.

A **trade-off criterion** can also be used:

$$J = \alpha \times size + \sum_{N \in leaf\ node} i(N)$$

Here $size$ could be the number of nodes or links and α is some positive constant.

This trade-off criterion aims for a balance between tree complexity and tree performance.

- **Pruning**

As the name implies, pruning involves cutting back the tree. After a tree has been built, it may be overfitted.

In pruning, all pairs of neighbouring leaf nodes (i.e., ones linked to a common antecedent node, one level above) are considered for elimination. Any pair whose elimination yields a satisfactory (small) increase in impurity is eliminated, and the common antecedent node declared a leaf (This antecedent, in turn, could itself be pruned)..

Clearly, such *merging* or *joining* of the two leaf nodes is the inverse of splitting

- **Assignment of leaf node labels**

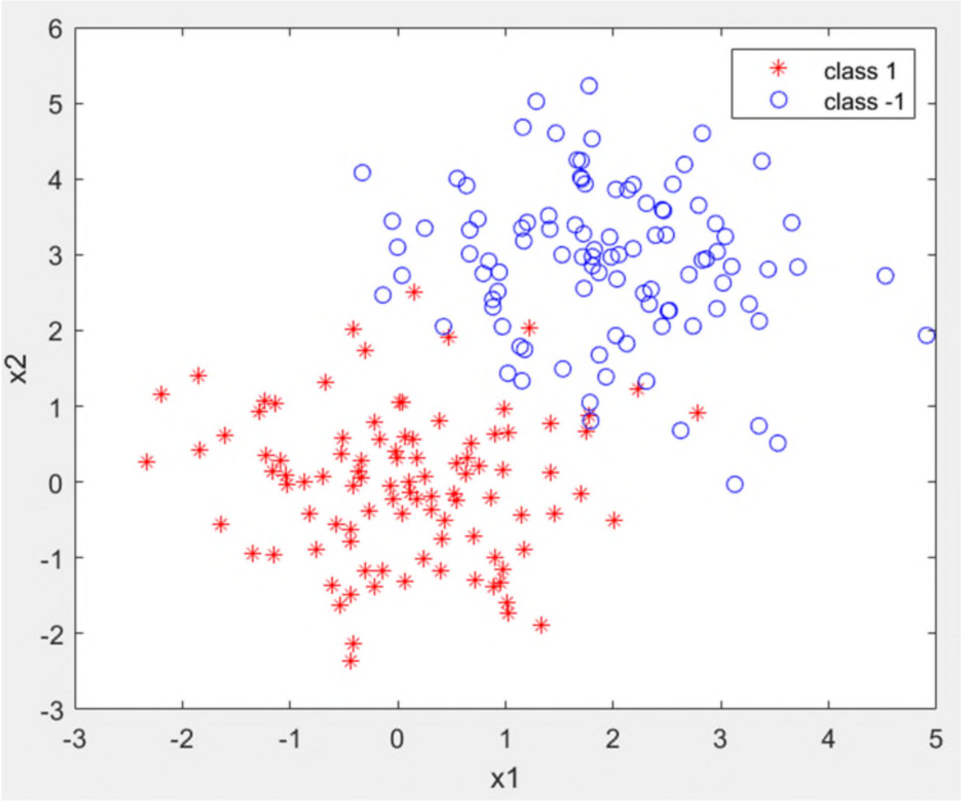
Assigning category labels to the leaf nodes is the simplest step in tree construction.

❑ If successive nodes are split as far as possible, and each leaf node corresponds to patterns in a single category (zero impurity), then of course this category label is assigned to the leaf

❑ In the more typical case, the leaf nodes have positive impurity. Thus, each leaf node should be labelled by the category that has the most samples.

Example:

Consider the following 2-class problem

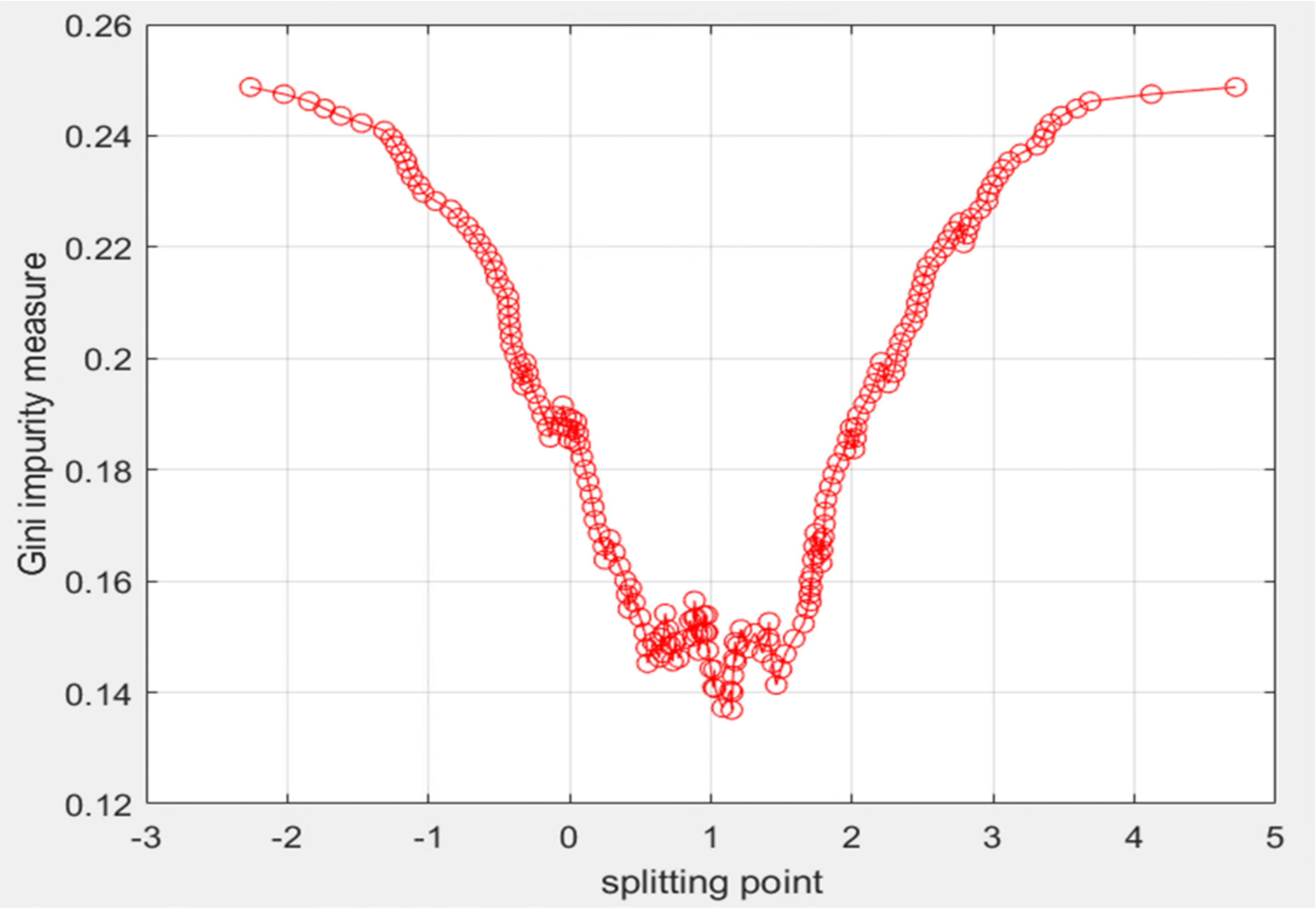


First, we select root node.

(1) Consider x_1 as the candidate attribute. Since it is a continuous variable, we sort the data and take the average of two adjacent values as the candidate splitting points and evaluate the corresponding Gini impurity measure.

The Gini impurity measure obtained is shown next.

Lowest Gini impurity measure 0.1367 is achieved when 1.147 is used as the splitting point.



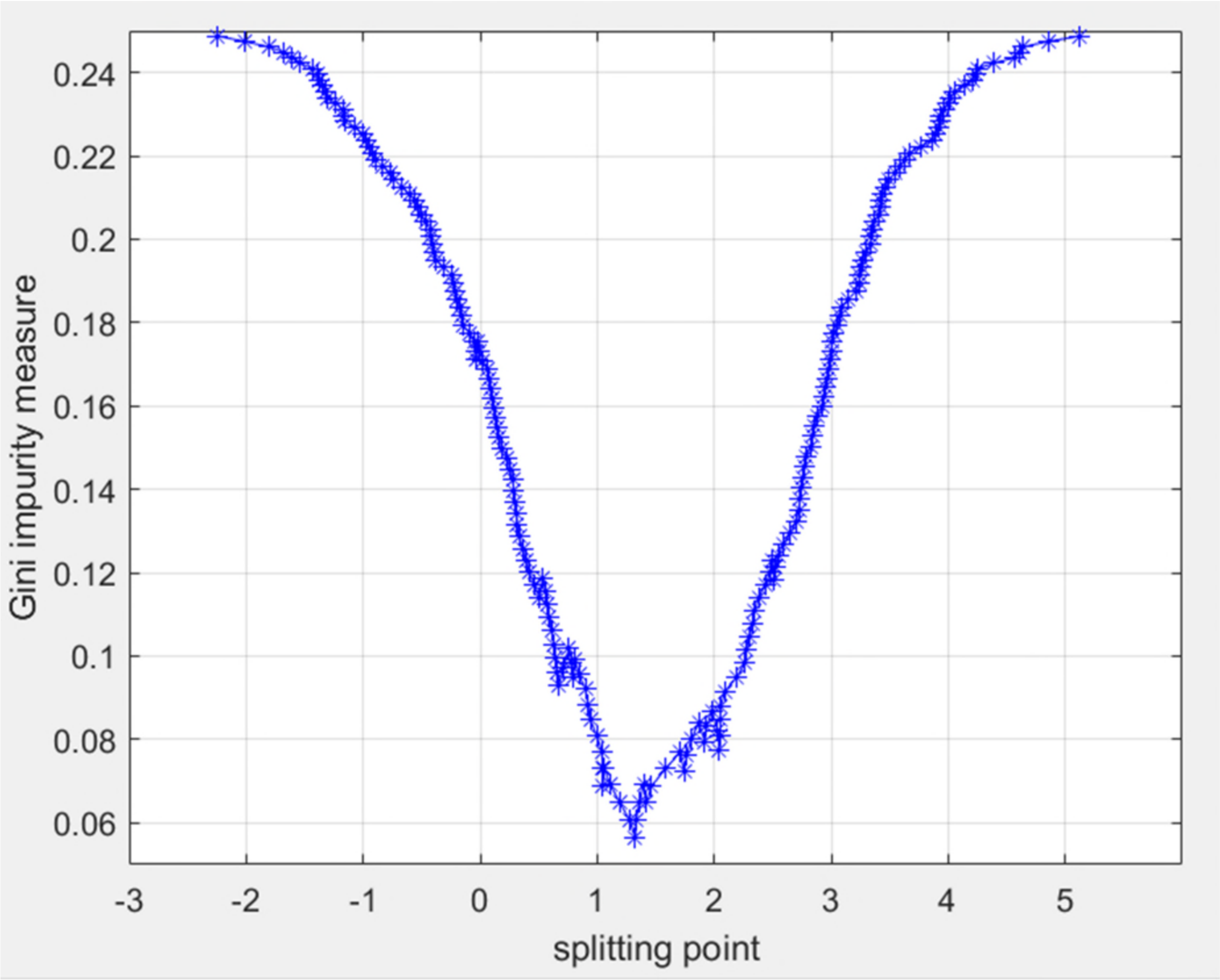
Gini impurity measure of x_1 of different splitting point

(2) Similarly, x_2 is considered as the candidate attribute for root node. Since it is a continuous variable, we sort the data and take the average of two adjacent values as the candidate splitting points and evaluate the corresponding Gini impurity measure.

The Gini impurity measure obtained is shown next.

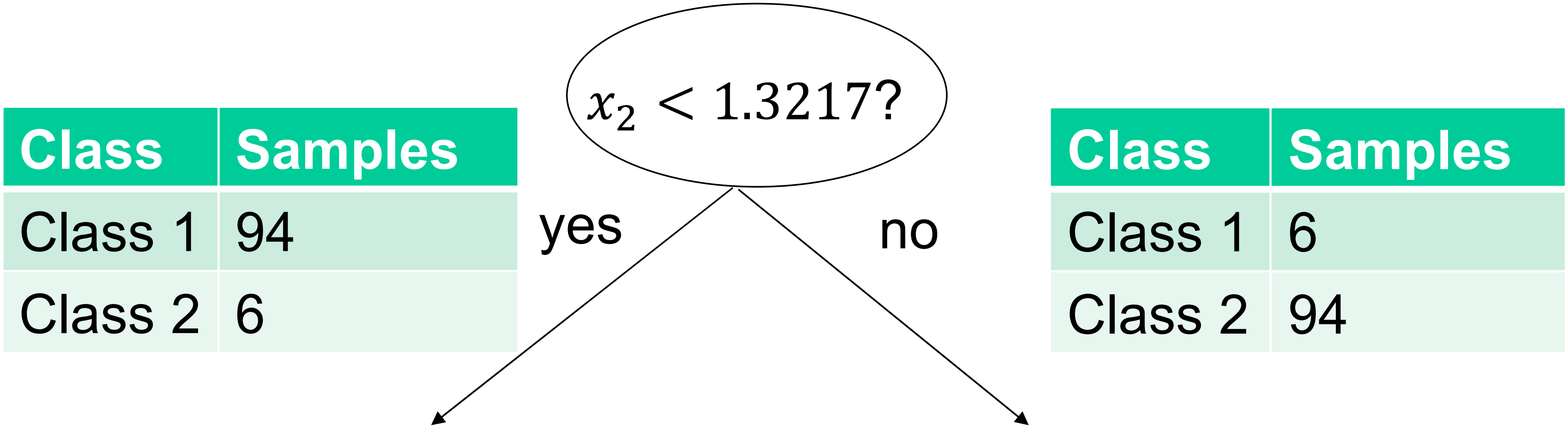
Lowest Gini impurity measure 0.0524 is achieved if 1.3271 is used as the splitting point.

Obviously, x_2 could lead to lower Gini impurity measure, and hence is selected as the root node, with splitting point of 1.3271



Gini impurity measure of x_2 of different splitting point

Then we have:



$x_2 < 1.3217?$

yes

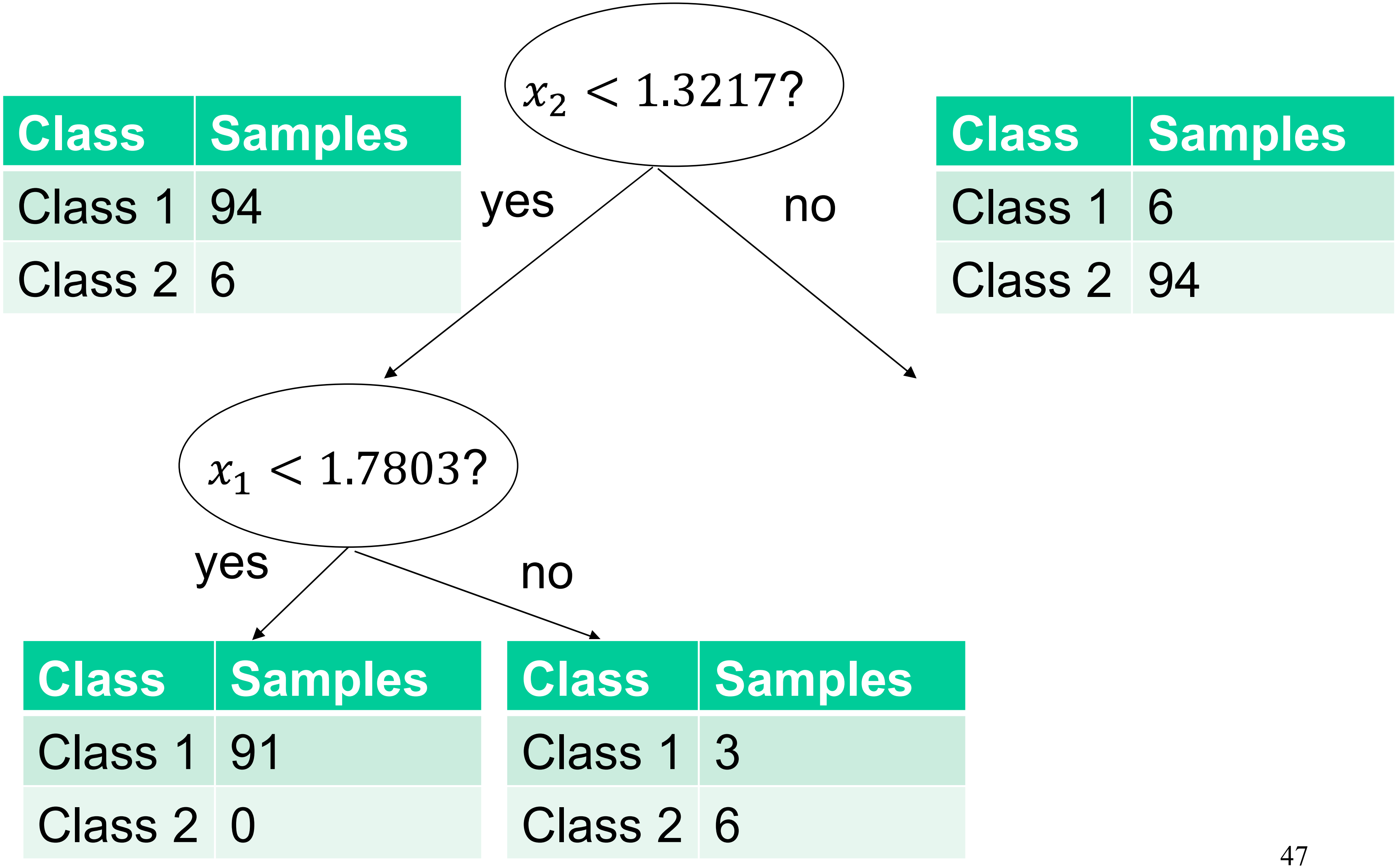
no

Class	Samples
Class 1	94
Class 2	6

Class	Samples
Class 1	6
Class 2	94

For the left branch, we search the optimal splitting point of x_1. From the figure below, we can see when 1.7803 is used, the measure is 0.02

Then we have:



$x_2 < 1.3217?$

yes

Class	Samples
Class 1	94
Class 2	6

no

Class	Samples
Class 1	6
Class 2	94

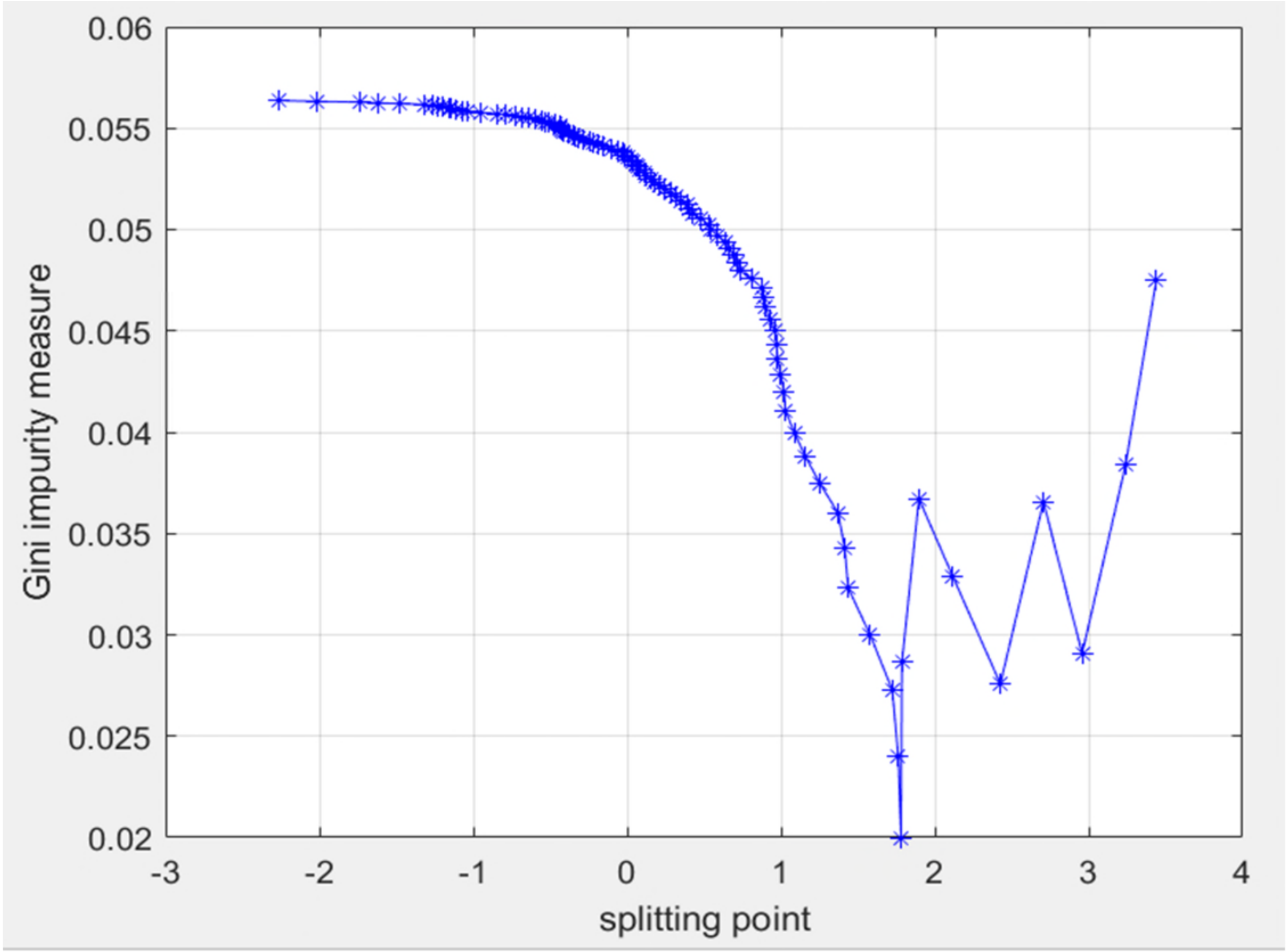
$x_1 < 1.7803?$

yes

Class	Samples
Class 1	91
Class 2	0

no

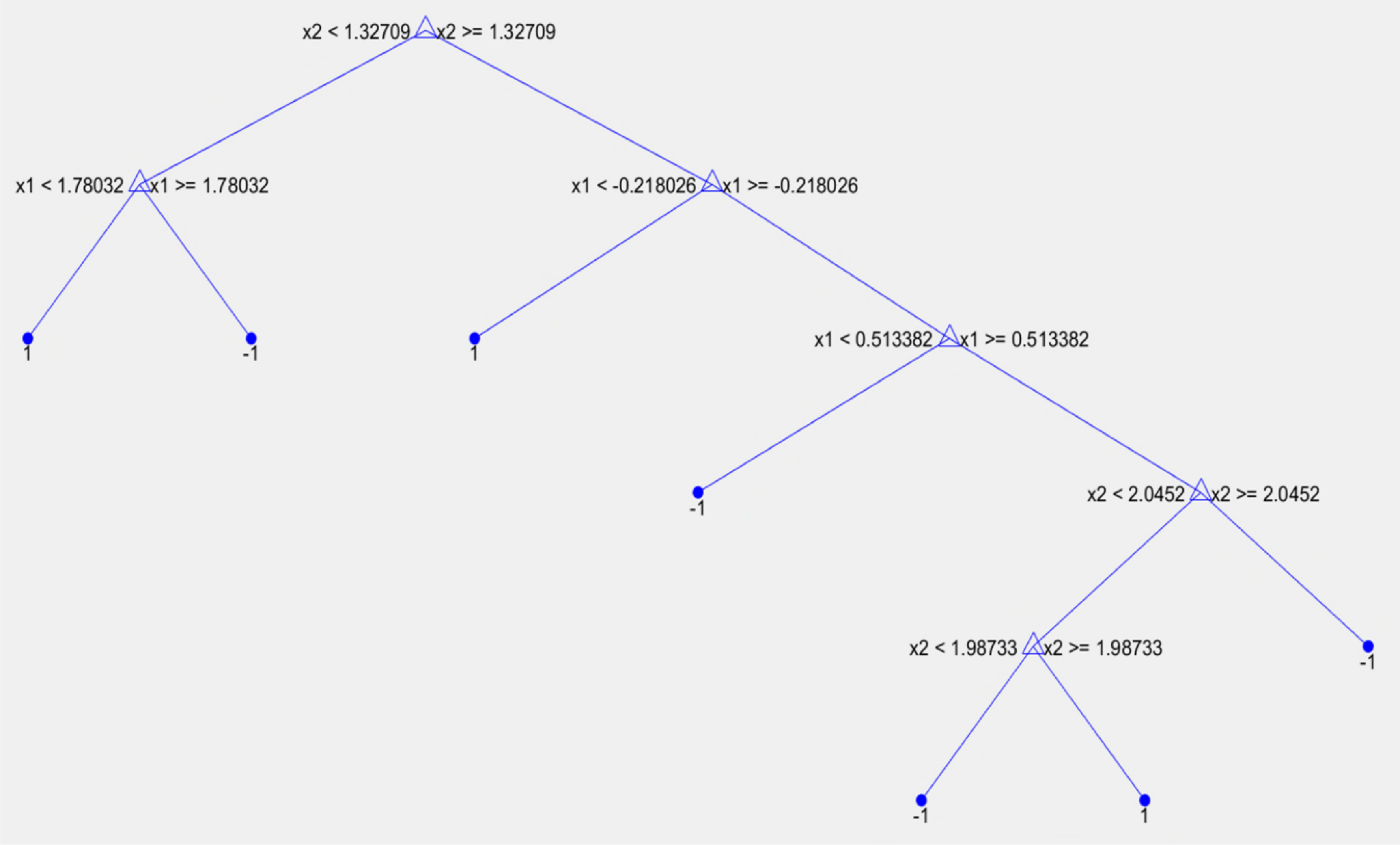
Class	Samples
Class 1	3
Class 2	6



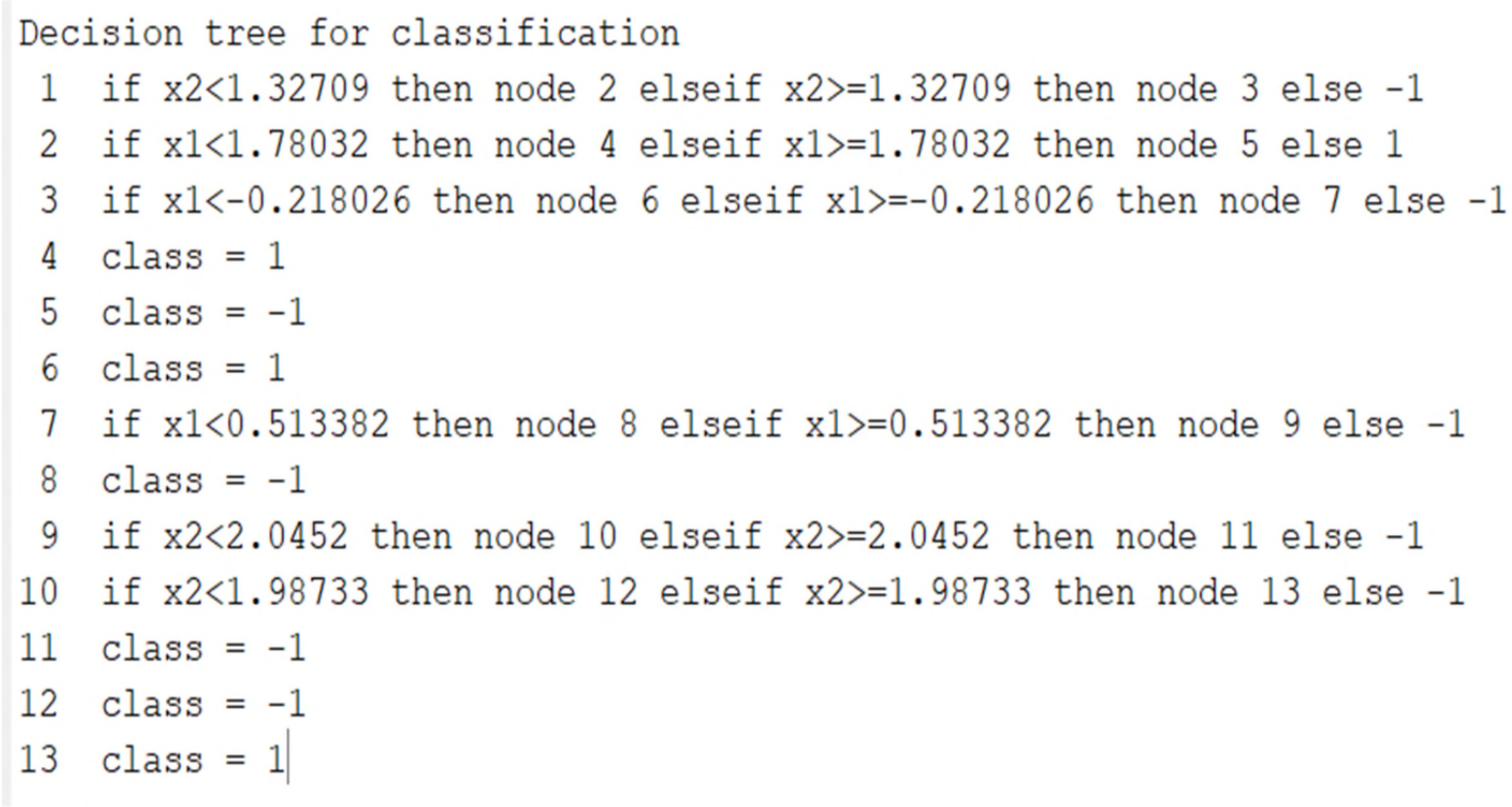
Gini impurity measure of x_1 of different splitting point

The branching can be continued until the stopping criterion, such as number of samples in leaf nodes is fewer than a pre-set threshold, is satisfied.

If we use the MATLAB function “fitctree”, we obtain the following tree:

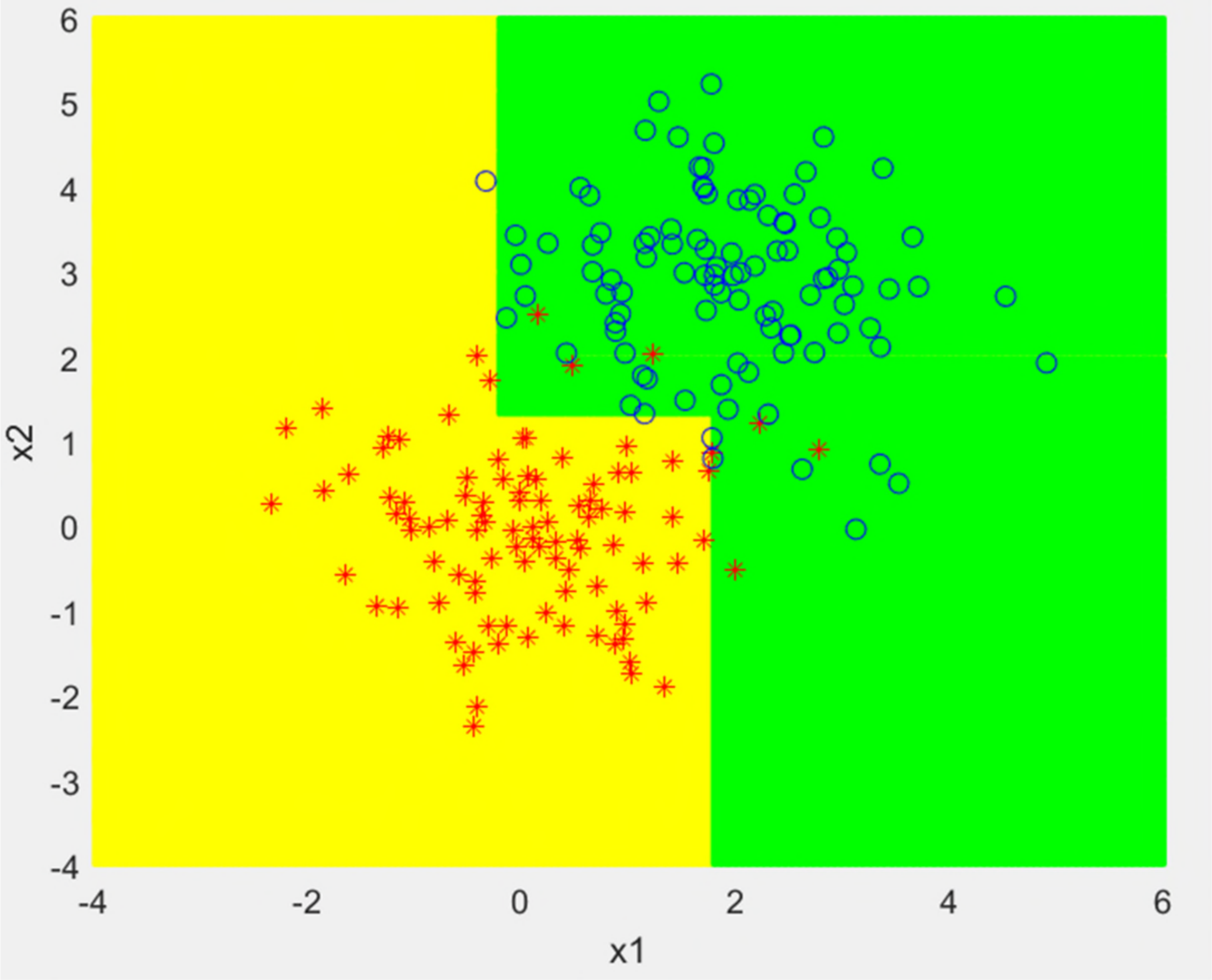


The classification tree actually is a graphical representation of the following rules:

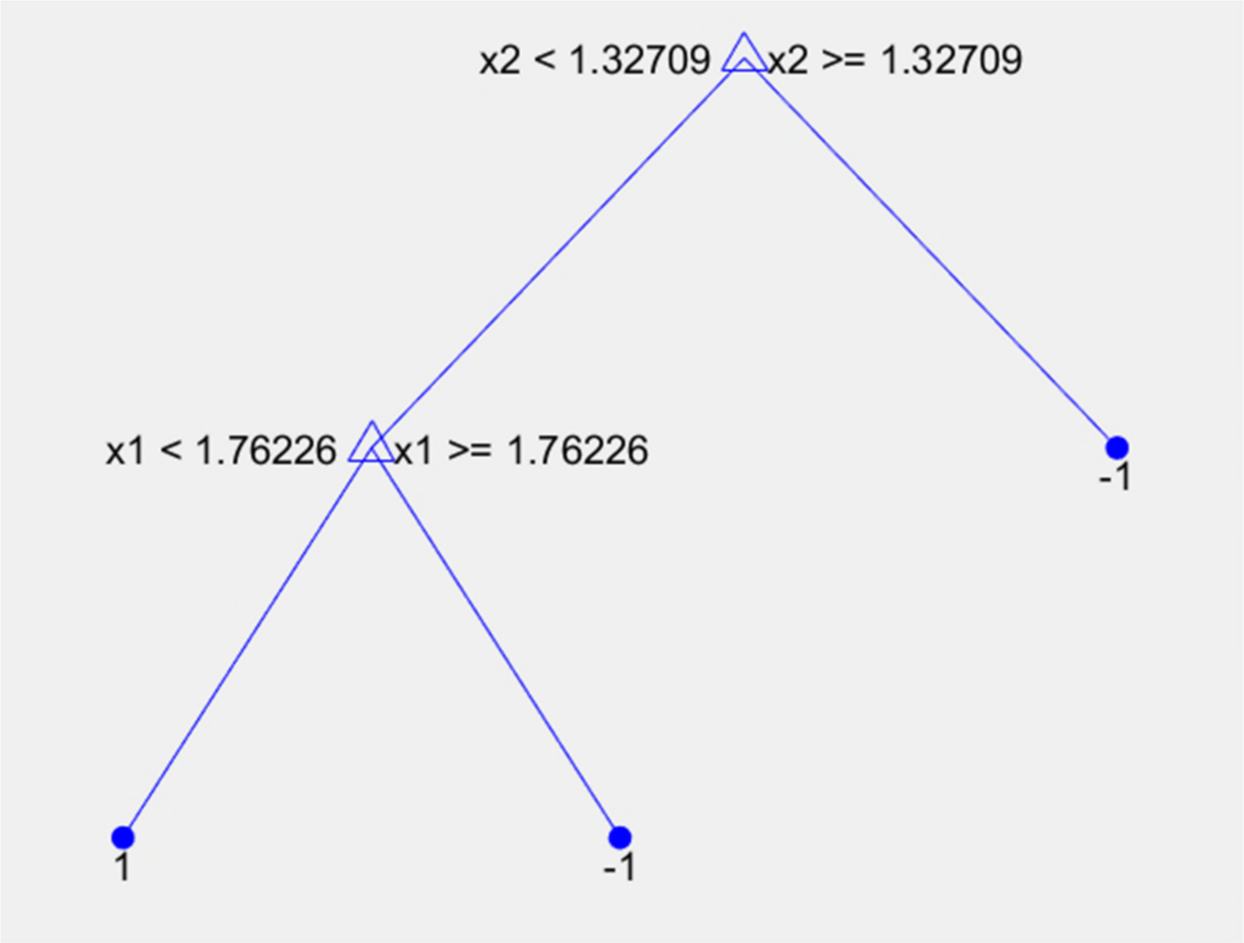


```
Decision tree for classification
 1  if x2<1.32709 then node 2 elseif x2>=1.32709 then node 3 else -1
 2  if x1<1.78032 then node 4 elseif x1>=1.78032 then node 5 else 1
 3  if x1<-0.218026 then node 6 elseif x1>=-0.218026 then node 7 else -1
 4  class = 1
 5  class = -1
 6  class = 1
 7  if x1<0.513382 then node 8 elseif x1>=0.513382 then node 9 else -1
 8  class = -1
 9  if x2<2.0452 then node 10 elseif x2>=2.0452 then node 11 else -1
10  if x2<1.98733 then node 12 elseif x2>=1.98733 then node 13 else -1
11  class = -1
12  class = -1
13  class = 1
```

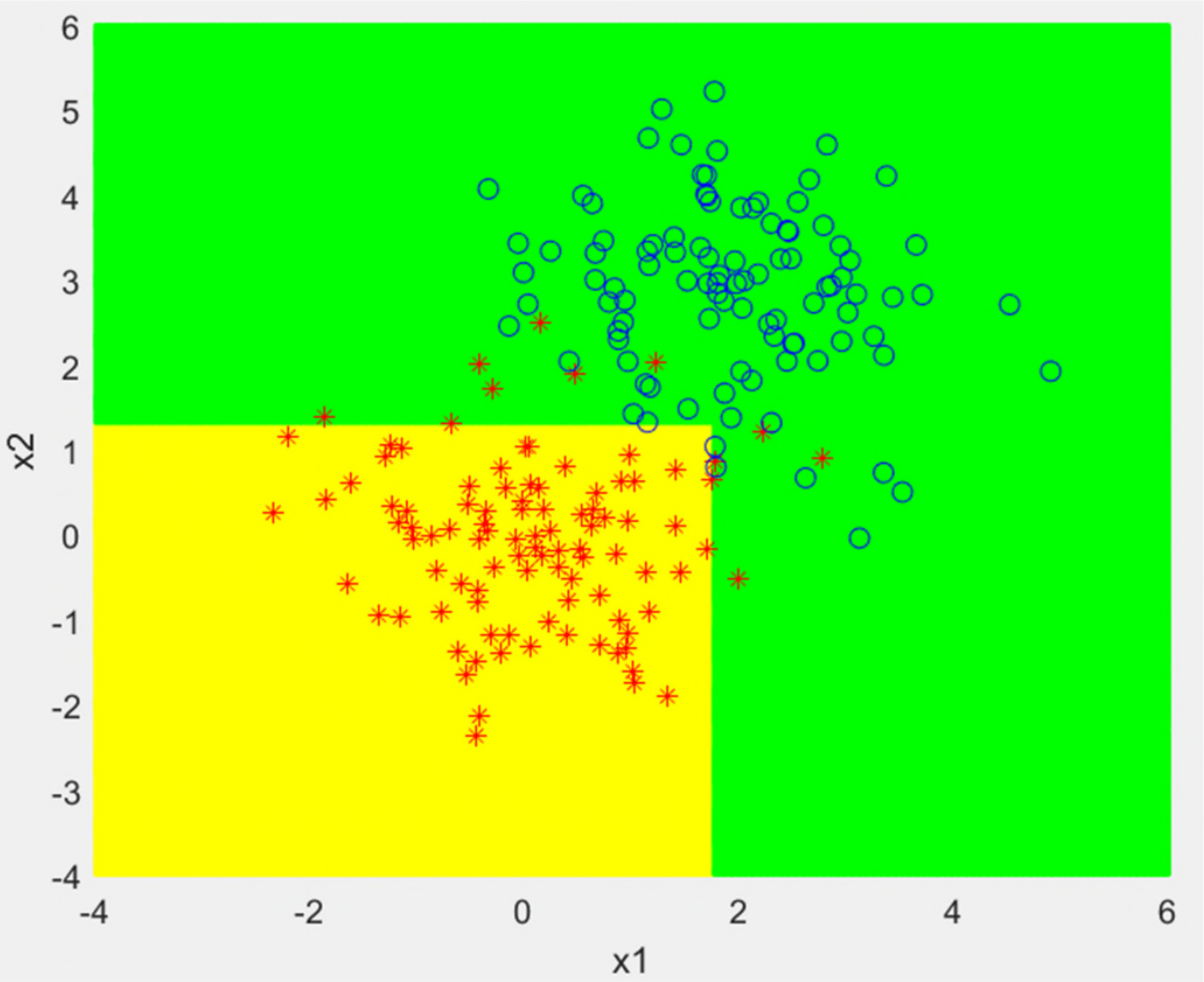
Decision boundary of the classification tree: 6 training errors.



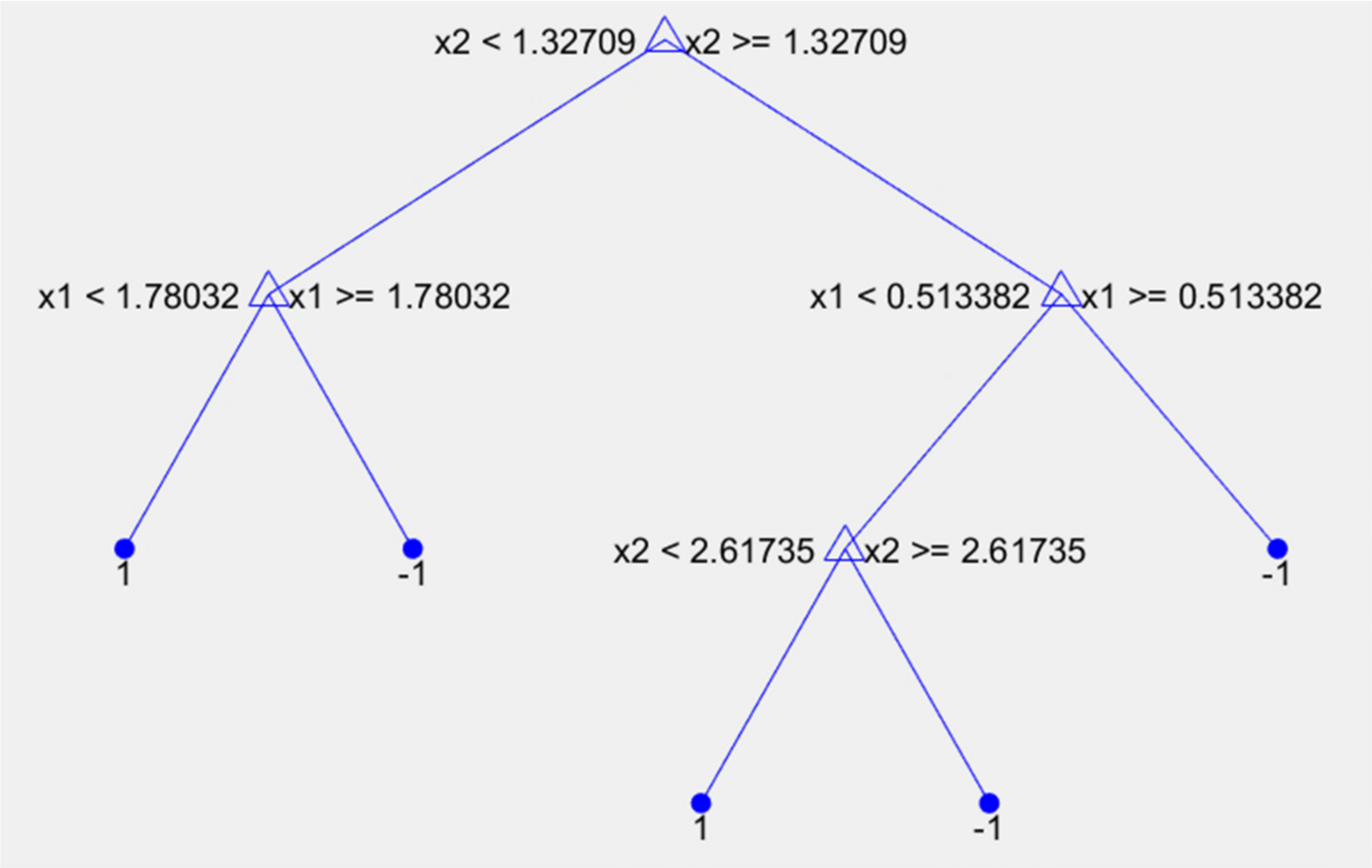
If we require at least 10 samples at a leaf node, then we obtain a much simpler tree:



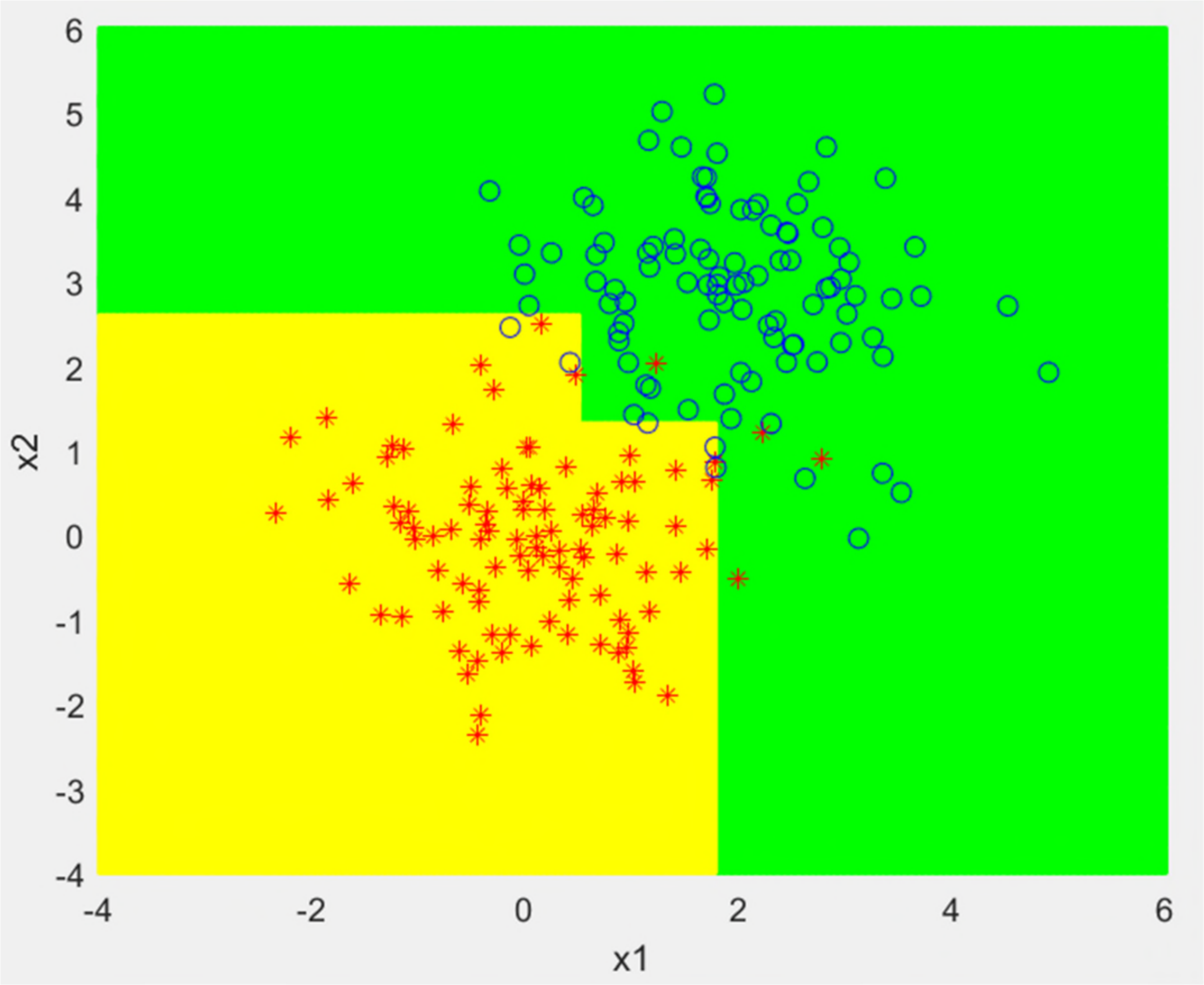
Decision boundary of the classification tree: 10 training errors



If we require at least 5 samples at a leaf node, then we obtain the following tree:



Decision boundary of the classification tree: 6 training errors



Some typical classification trees

Virtually, all tree-based classification techniques employ the fundamental techniques described above. The following are some typical classification trees:

(i) CART (Classification and Regression Tree). In fact, the techniques discussed above are the core ideas of CART.

(ii) ID3 (Iterative Dichotomiser 3)

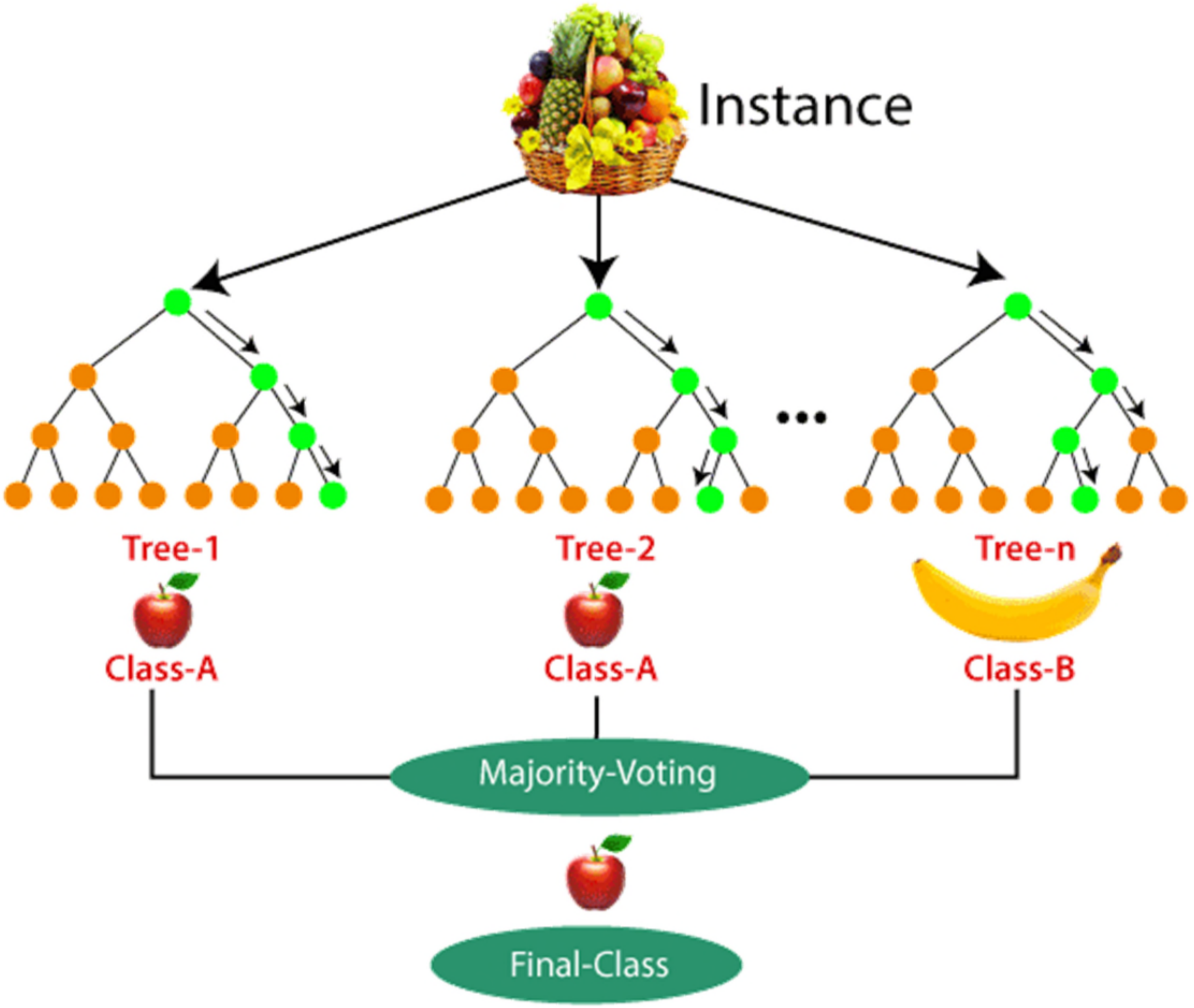
- ❑ It is intended for use with nominal (unordered) inputs only.
- ❑ If the problem involves real-valued variables, they are first binned into intervals, each interval being treated as an unordered nominal attribute.

(iii) C4.5. The C4.5 algorithm is the successor and refinement of ID3. In C4.5,

- □ Real-valued variables are treated the same as in CART.
- □ Multiple ($B > 2$) splits are used with nominal data, as in ID3 with a gain ratio impurity.

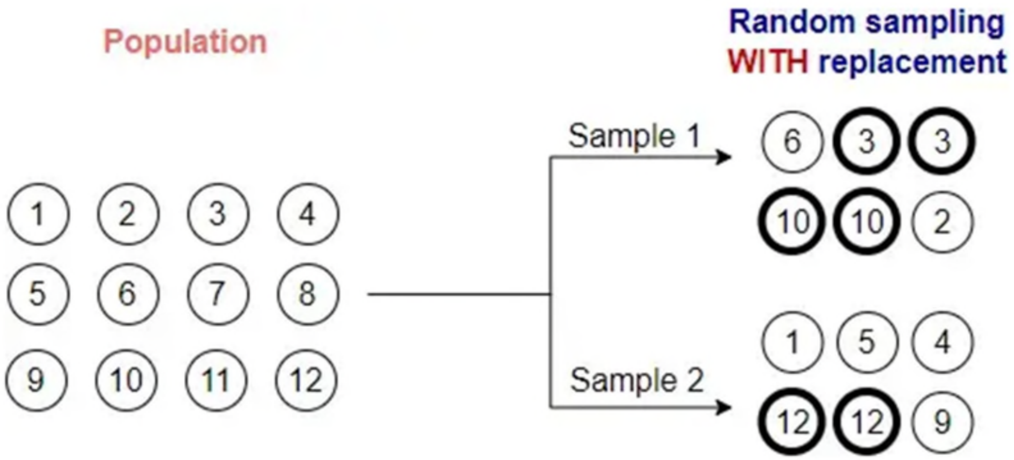
(iv) Random forest

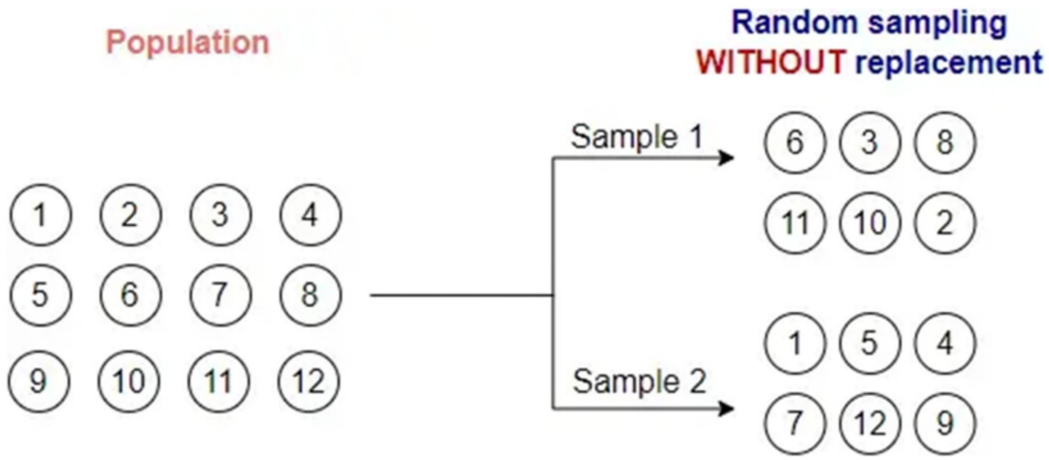
- □ Random forest consists of many decision trees, trained on different subsets of the full training data.
- □ Random forest is an ensemble learning method: each individual tree in the random forest produces a class prediction and the class with the most votes becomes the model's prediction, as illustrated below.


Instance
Tree-1
Tree-2
...
Tree-n
Class-A
Class-A
Class-B
Majority-Voting
Final-Class

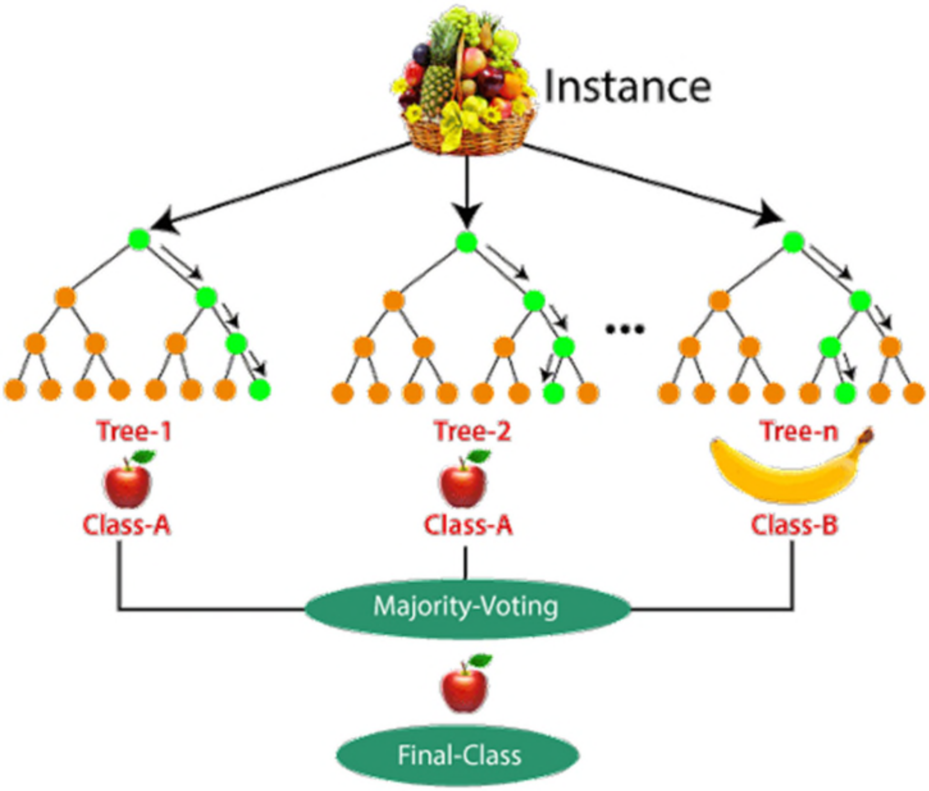
Random Forest is developed based on the ensemble technique of Bagging, short for Bootstrap Aggregating.

The Bootstrapping part of Bagging refers to the resampling method in which several random samples are drawn with replacement from a dataset. Thus, Bootstrapping creates multiple, smaller random datasets drawn from the same distribution.


Population
Random sampling
WITH replacement
Sample 1
Sample 2
1 2 3 4
5 6 7 8
9 10 11 12
6 3 3
10 10 2
1 5 4
12 12 9


Population
Random sampling
WITHOUT replacement
Sample 1
Sample 2
1 2 3 4
5 6 7 8
9 10 11 12
6 3 8
11 10 2
1 5 4
7 12 9

Each of the bootstrapped datasets is used to train a classification tree. For a test sample, the outputs from the multiple classification trees are then Aggregated into one final result, by picking the most common results from all the trees, i.e. majority voting.



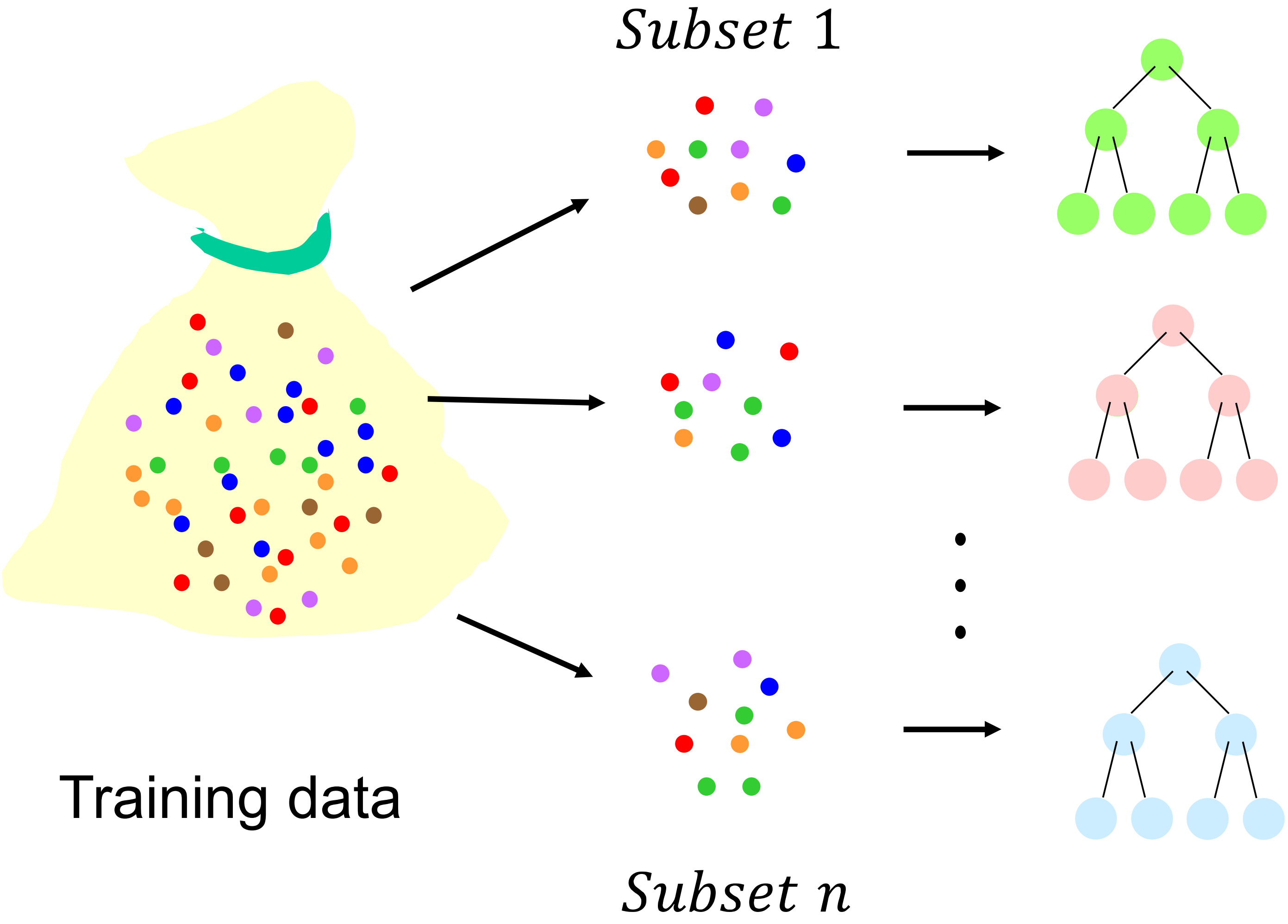
Steps to build a random forest

Assume the training dataset consists of N samples with d features.

(1) Create n data subsets through bootstrapping: $subset\ 1, subset\ 2, \dots, subset\ n$, where $n < N$.

(2) For each data subset, randomly select m features from the original d features, and use the m features to construct a classification tree, where $m < d$.

(3) Repeat Step (2) until all n subsets have been used.



$Subset\ 1$

Training data

$Subset\ n$

Steps to use a random forest for classification

Given a test sample with unknown class label

(1) Run the test sample through each of the n classification trees to obtain the predicted class from each classification tree.

(2) Calculate the votes for each of the predicted class.

(3) Output the most highly voted predicted class as the final class prediction

Features of Random Forest

(1) Diversity: Not all attributes or features are considered while making an individual tree; each tree is unique.

(2) Immune to the curse of dimensionality: Since each tree does not consider all the features, the feature space is reduced.

(3) Parallelization: Each tree is created independently out of different data and attributes.

(4) Train-Test split: In a random forest, we don’t have to split the data for train and test as there will always be data unseen by the decision tree.

(5) Stability: Stability arises because the result is based on majority voting/averaging.

Random forest vs Classification tree

Aspect	Random forest	Classification tree
Nature	Ensemble of multiple classification trees	A single classification tree
Variance	Lower variance, reduced overfitting	Higher variance, prone to overfitting
Accuracy	Generally higher due to ensemble	Prone to overfitting, may vary
Robustness	More robust to outliers and noise	Sensitive to outliers and noise
Training time	Slower due to multiple tree construction	Faster as it builds a single tree
Interpretability	Less interpretable due to ensemble	More interpretable as a single tree
Usage	Suitable for complex tasks, high-dimensional data	Simple tasks, easy interpretation