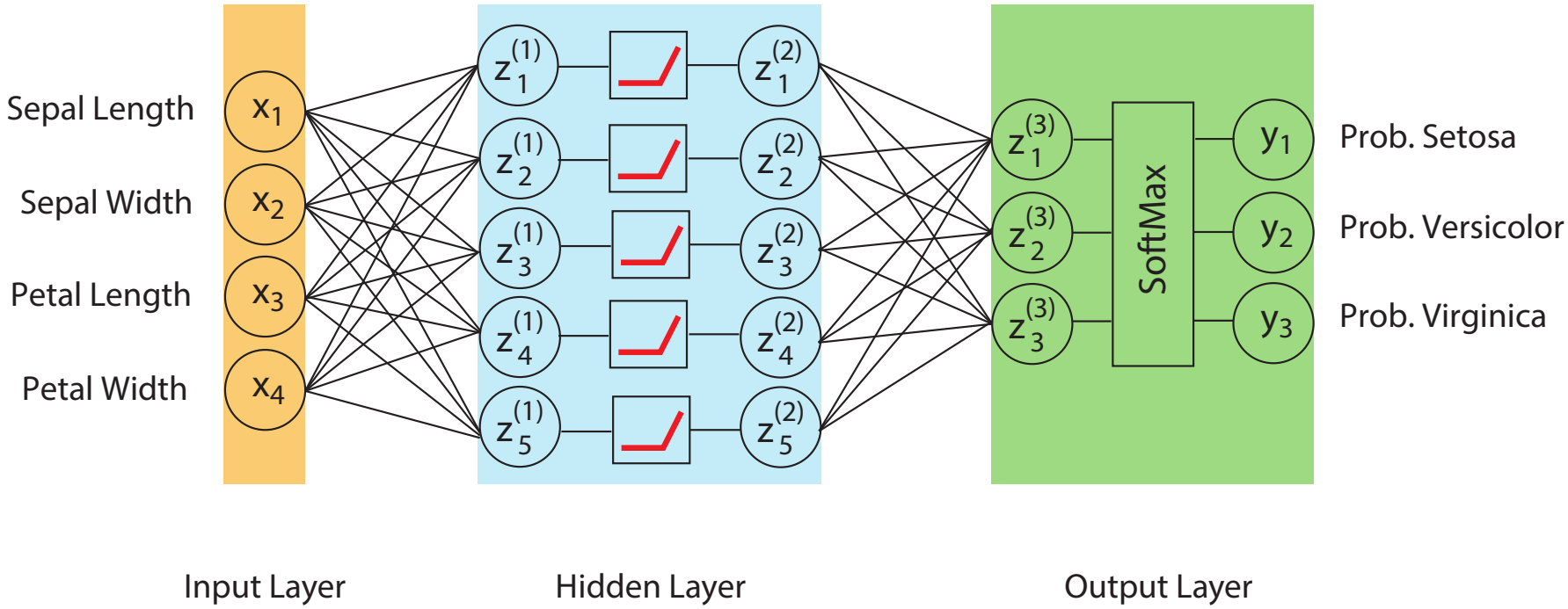
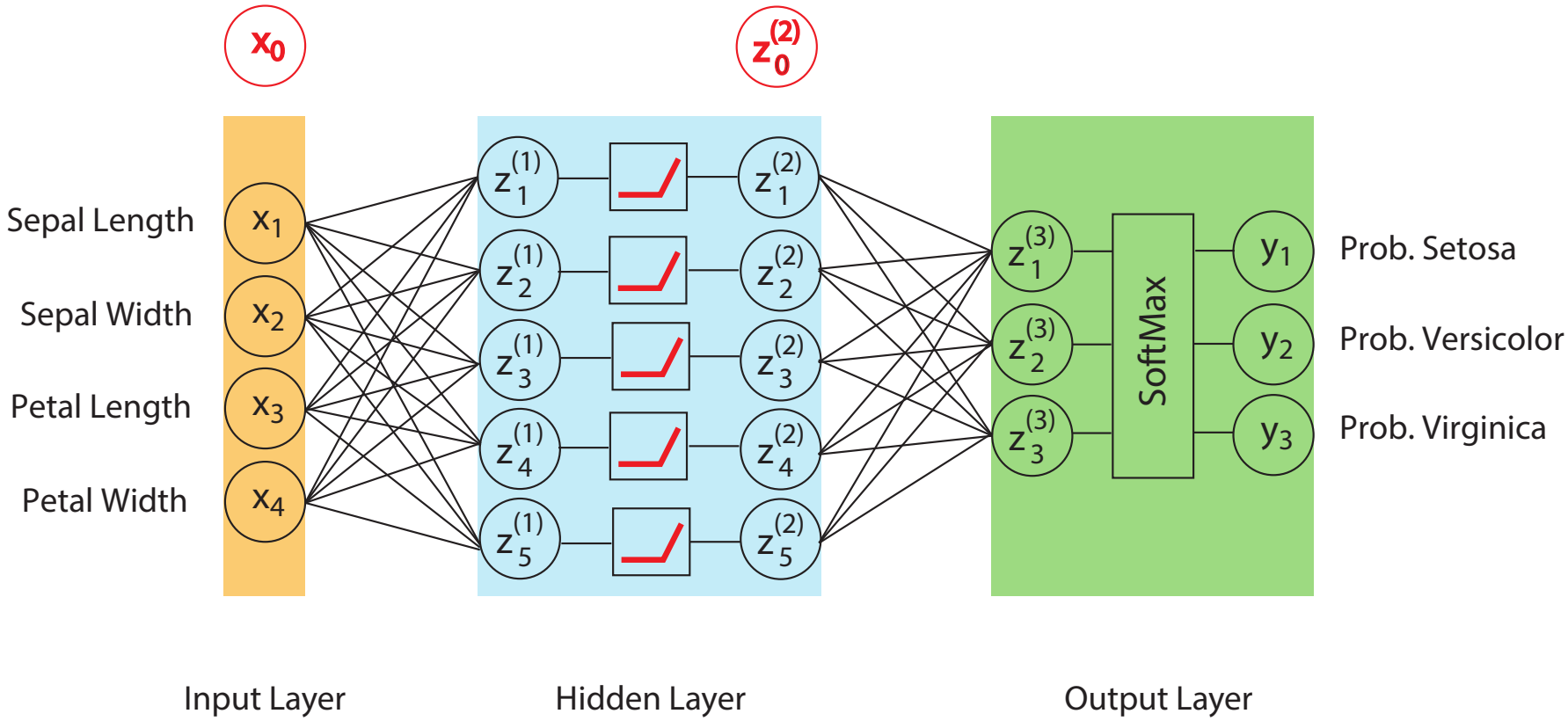
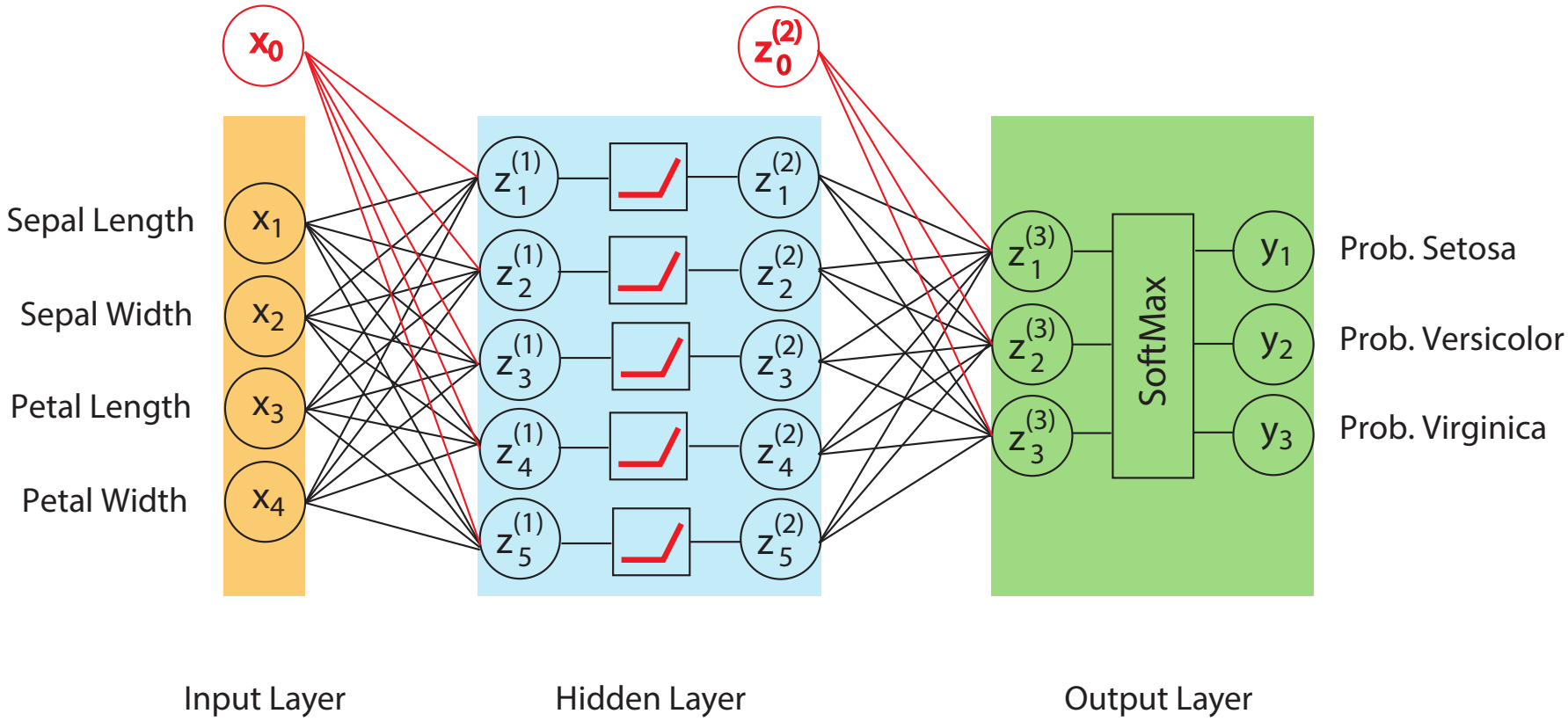
Bonus 1: Bias node



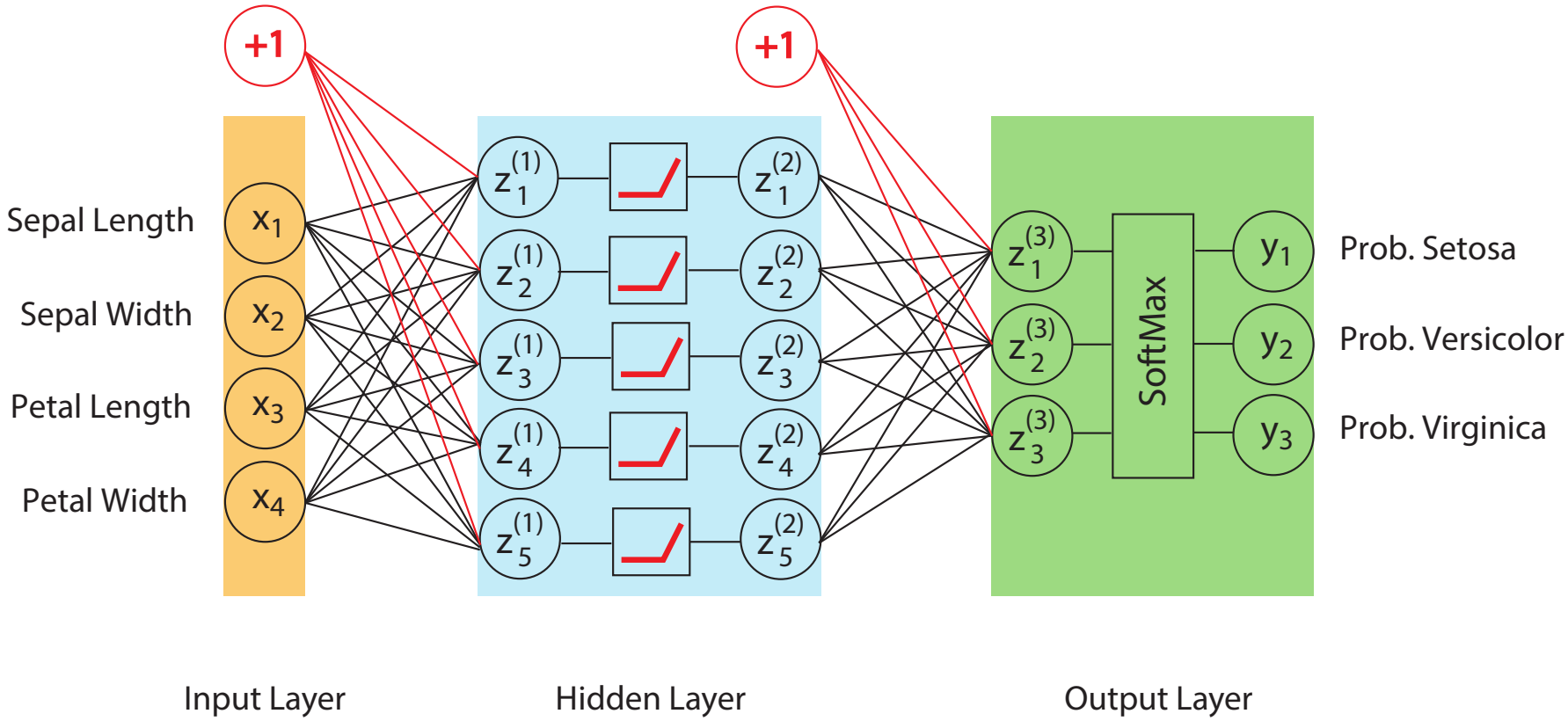
Bonus 1: Bias node



Bonus 1: Bias node



Bonus 1: Bias node



Bonus 1: Bias node

We have

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \\ w_{51} & w_{52} & w_{53} & w_{54} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} w_{11}\ x_1 + w_{12}\ x_2 + w_{13}\ x_3 + w_{14}\ x_4 \\ w_{21}\ x_1 + w_{22}\ x_2 + w_{23}\ x_3 + w_{24}\ x_4 \\ w_{31}\ x_1 + w_{32}\ x_2 + w_{33}\ x_3 + w_{34}\ x_4 \\ w_{41}\ x_1 + w_{42}\ x_2 + w_{43}\ x_3 + w_{44}\ x_4 \\ w_{51}\ x_1 + w_{52}\ x_2 + w_{53}\ x_3 + w_{54}\ x_4 \end{pmatrix}$$

Bonus 1: Bias node

We have

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \\ w_{51} & w_{52} & w_{53} & w_{54} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} w_{11}\ x_1 + w_{12}\ x_2 + w_{13}\ x_3 + w_{14}\ x_4 \\ w_{21}\ x_1 + w_{22}\ x_2 + w_{23}\ x_3 + w_{24}\ x_4 \\ w_{31}\ x_1 + w_{32}\ x_2 + w_{33}\ x_3 + w_{34}\ x_4 \\ w_{41}\ x_1 + w_{42}\ x_2 + w_{43}\ x_3 + w_{44}\ x_4 \\ w_{51}\ x_1 + w_{52}\ x_2 + w_{53}\ x_3 + w_{54}\ x_4 \end{pmatrix}$$

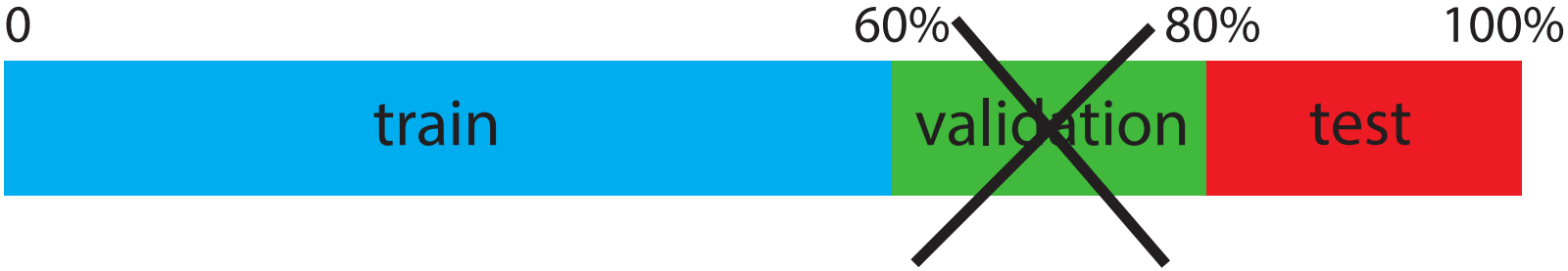
We now add a bias node:

$$\begin{pmatrix} w_{10} & w_{11} & w_{12} & w_{13} & w_{14} \\ w_{20} & w_{21} & w_{22} & w_{23} & w_{24} \\ w_{30} & w_{31} & w_{32} & w_{33} & w_{34} \\ w_{40} & w_{41} & w_{42} & w_{43} & w_{44} \\ w_{50} & w_{51} & w_{52} & w_{53} & w_{54} \end{pmatrix} \begin{pmatrix} +1 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} w_{10} + w_{11}\ x_1 + w_{12}\ x_2 + \dots \\ w_{20} + w_{21}\ x_1 + w_{22}\ x_2 + \dots \\ w_{30} + w_{31}\ x_1 + w_{32}\ x_2 + \dots \\ w_{40} + w_{41}\ x_1 + w_{42}\ x_2 + \dots \\ w_{50} + w_{51}\ x_1 + w_{52}\ x_2 + \dots \end{pmatrix}$$

The bias node introduces the intercepts w_{10}, w_{20}, w_{30}, w_{40}, w_{50}

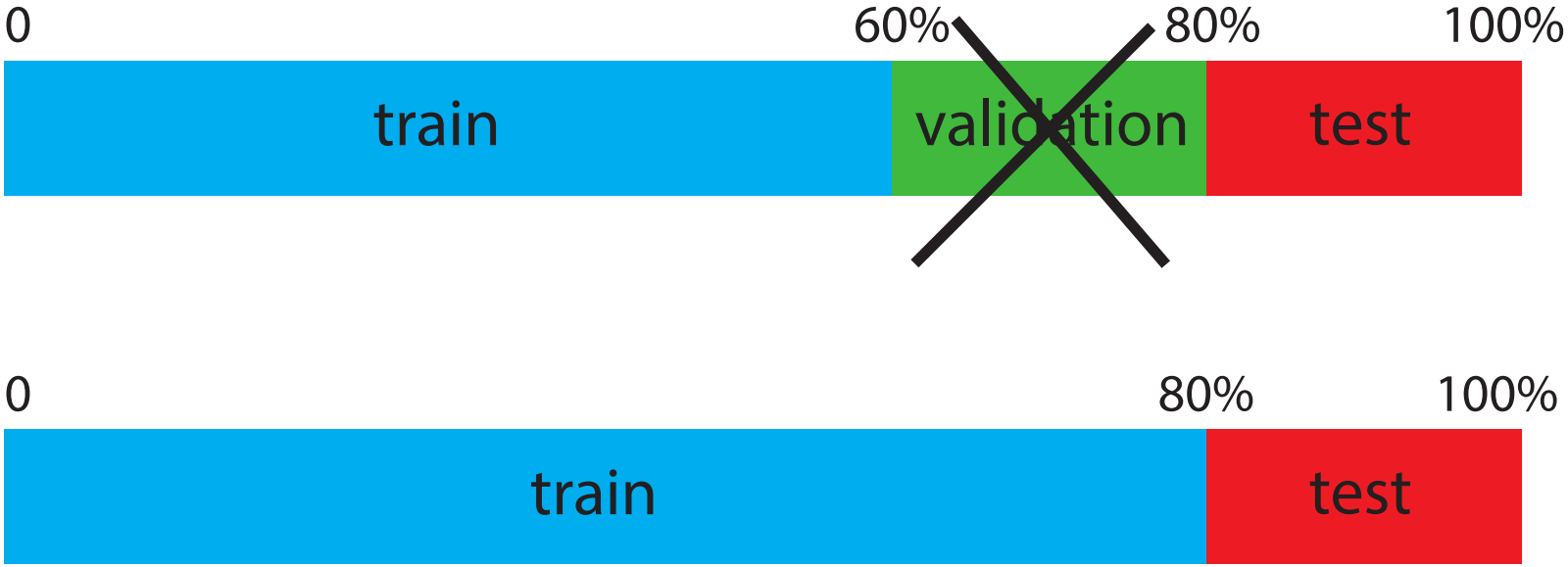
Bonus 2: K-fold cross-validation for model selection

For small datasets, one may not want to keep a separate validation dataset:



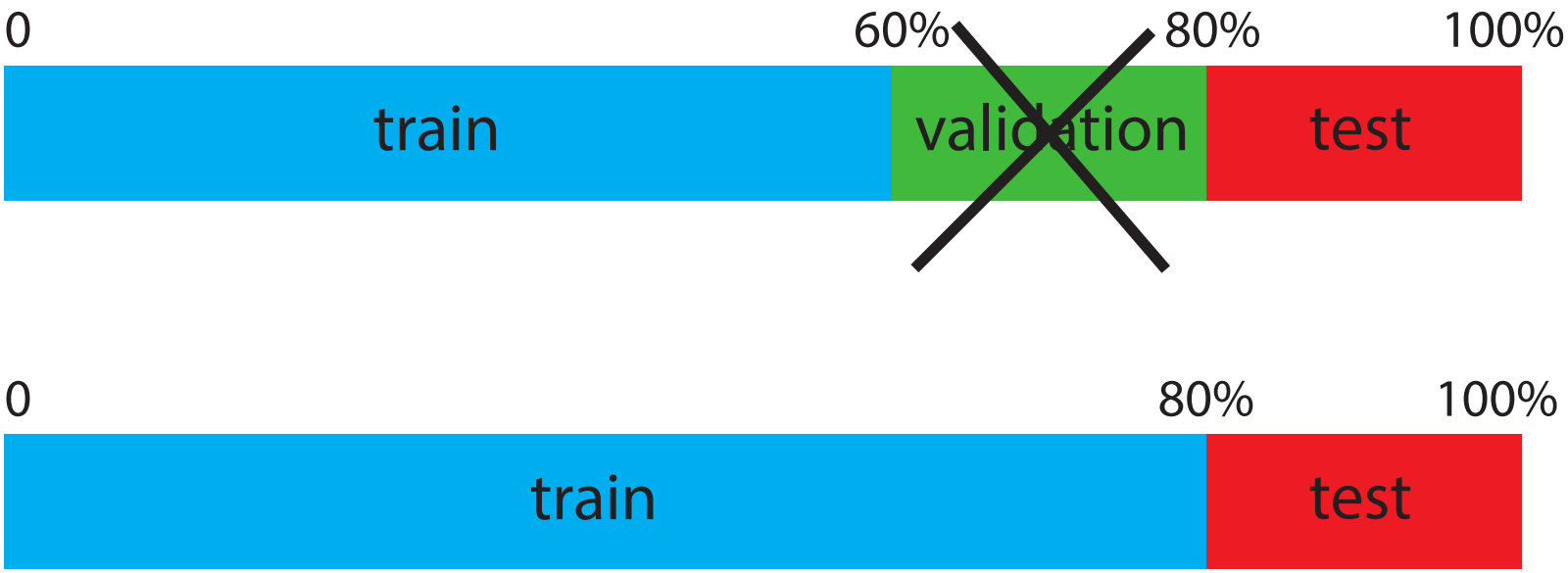
Bonus 2: K-fold cross-validation for model selection

For small datasets, one may not want to keep a separate validation dataset:



Bonus 2: K-fold cross-validation for model selection

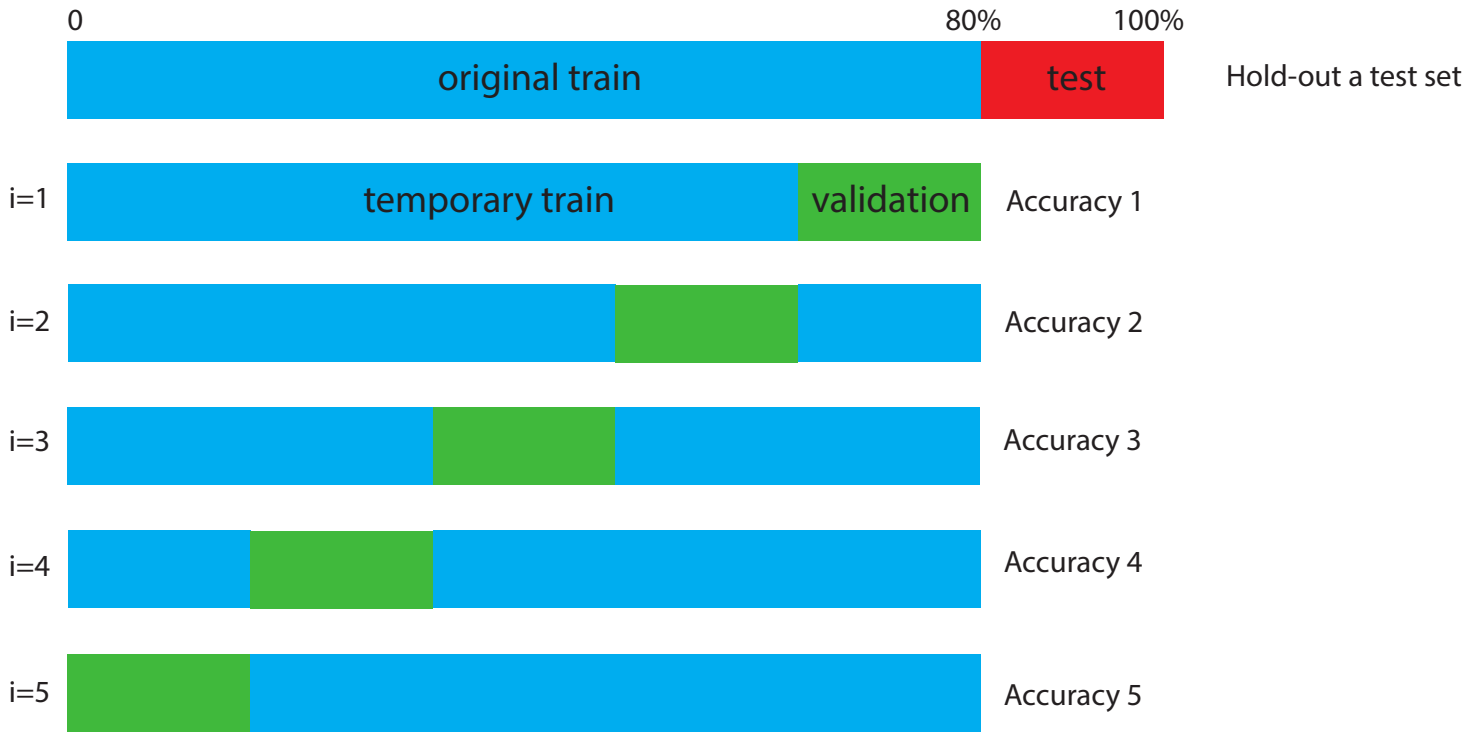
For small datasets, one may not want to keep a separate validation dataset:



- Model selection: Evaluate the validation accuracy of different models by using K-fold cross-validation.
- Model assessment: Use the training set to fit the best selected model and the test set to evaluate its accuracy.

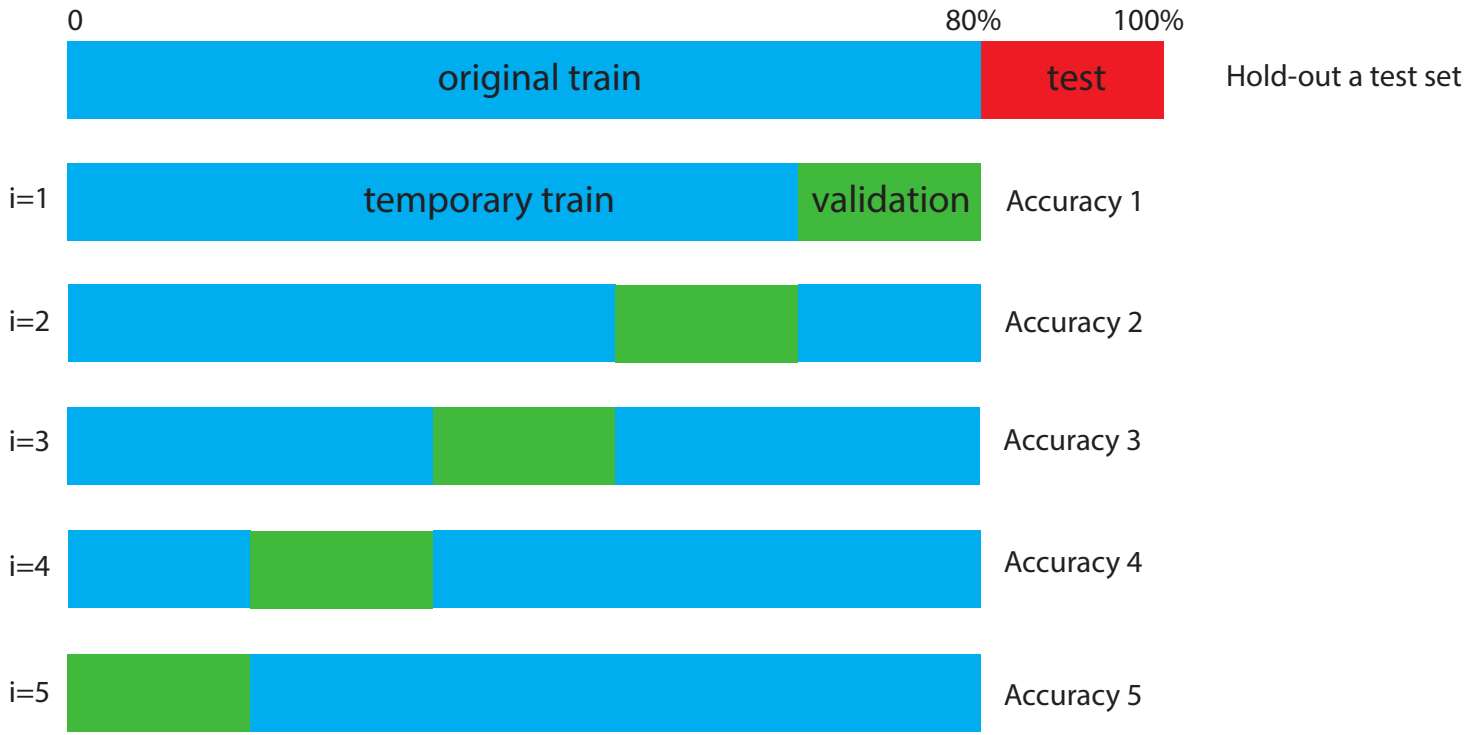
Bonus 2: K-fold cross-validation for model selection

Estimate the validation accuracy of a model by K-fold cross-validation (K=5):



Bonus 2: K-fold cross-validation for model selection

Estimate the validation accuracy of a model by K-fold cross-validation (K=5):

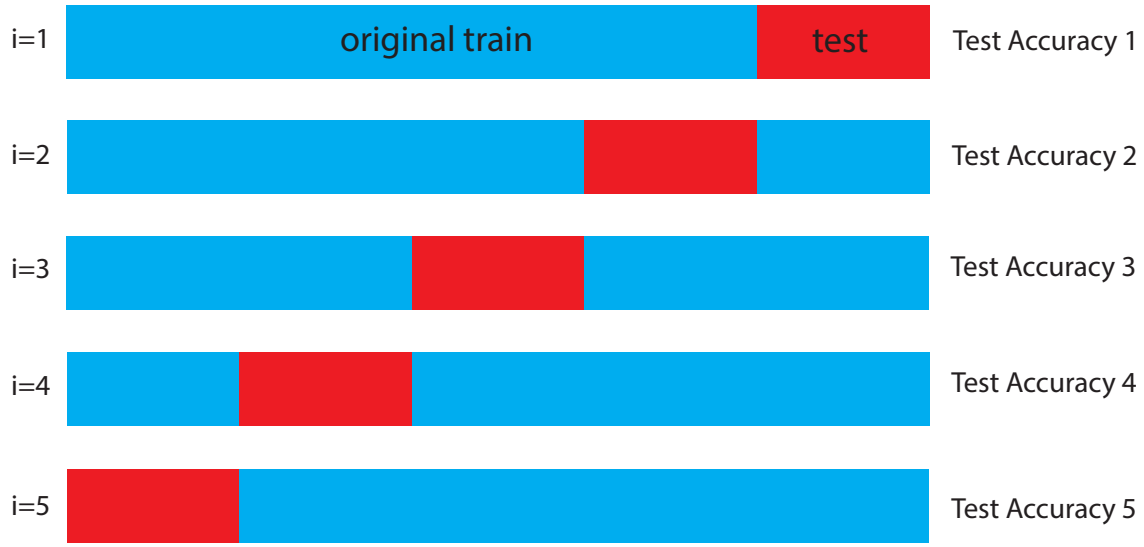


The Cross-Validation accuracy rate:

$$\text{CV} = \frac{1}{K}\sum_{i=1}^{K} \text{Accuracy}_i$$

Bonus 2: K-fold CV for model assessment

The best selected model (obtained by **inner** K-fold CV) and its accuracy may depend on the particular training/test splitting. One may check the stability of the best model and its prediction accuracy by using **outer** K-fold CV (K=5):

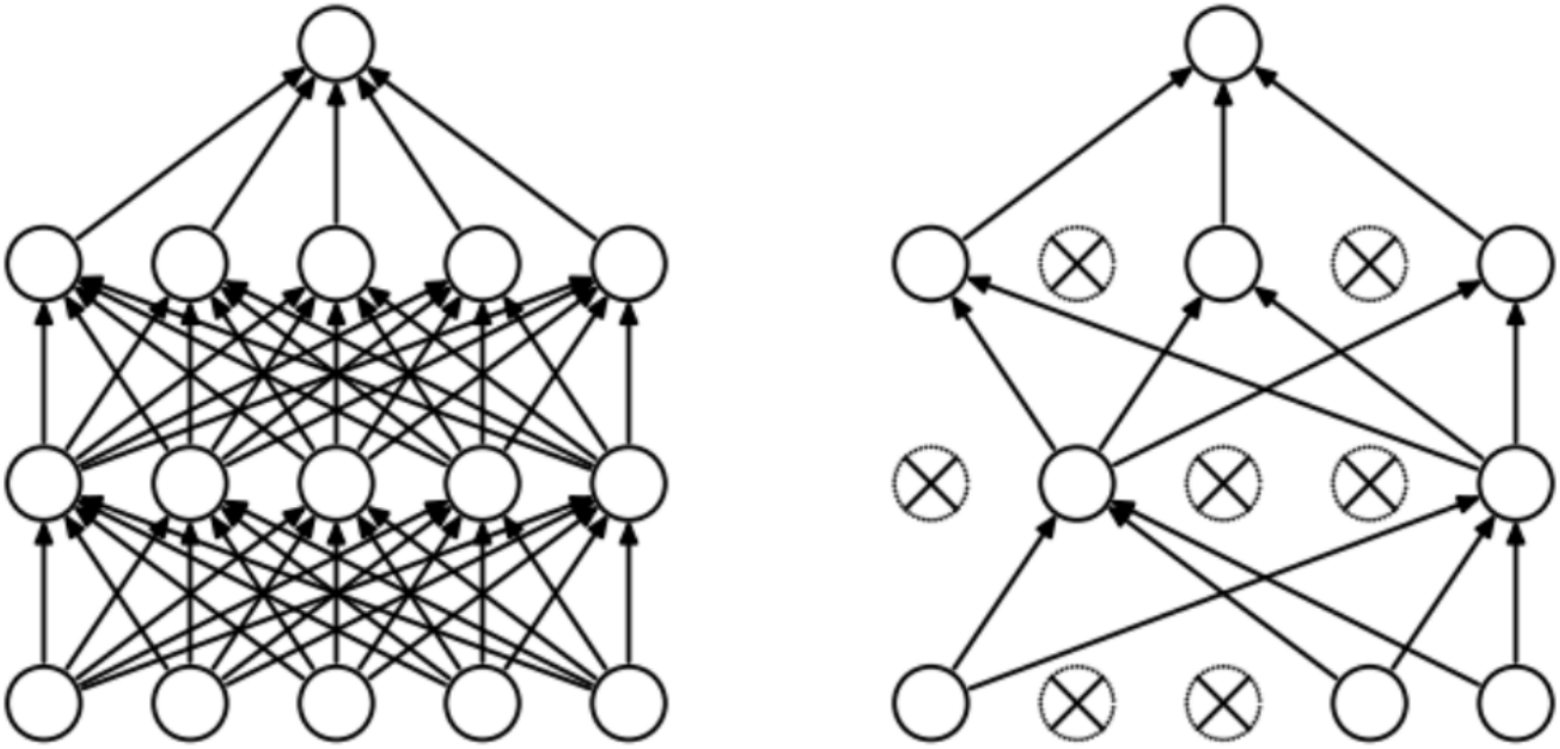


The Cross-Validation Test accuracy rate:

$$\text{CV Test} = \frac{1}{K}\sum_{i=1}^{K} \text{Test Accuracy}_i$$

Bonus 3: Dropout nodes

Dropout is a regularization technique for reducing overfitting in neural networks. One drops out units (both hidden and visible) during training according to a random distribution. This prevents units from co-adapting too much.



(a) Standard Neural Net (b) After applying dropout.

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

Bonus 3: Dropout nodes

- Training Phase: For each selected layer and for each training iteration, ignore a random fraction 1-p of nodes. Different nodes will be dropped at each iteration.

Bonus 3: Dropout nodes

- Training Phase: For each selected layer and for each training iteration, ignore a random fraction 1-p of nodes. Different nodes will be dropped at each iteration.

- Testing Phase: Use all nodes but make normalization (to account for the missing nodes during training).

Bonus 4: Back-propagation

The **error function**:

$$E_n(W) = -\sum_{k=1}^{3} t_{nk} \cdot \log y_{nk}(W)$$

The weights are updated with the **gradient descent algorithm**:

$$(w_{ij}^{(1)})^{\tau} = (w_{ij}^{(1)})^{\tau-1} - \eta \cdot \frac{\partial E_n}{\partial w_{ij}^{(1)}}$$

$$(w_{ij}^{(2)})^{\tau} = (w_{ij}^{(2)})^{\tau-1} - \eta \cdot \frac{\partial E_n}{\partial w_{ij}^{(2)}}$$

Bonus 4: Back-propagation

The **error function**:

$$E_n(W) = -\sum_{k=1}^{3} t_{nk} \cdot \log y_{nk}(W)$$

The weights are updated with the **gradient descent algorithm**:

$$(w_{ij}^{(1)})^{\tau} = (w_{ij}^{(1)})^{\tau-1} - \eta \cdot \frac{\partial E_n}{\partial w_{ij}^{(1)}}$$

$$(w_{ij}^{(2)})^{\tau} = (w_{ij}^{(2)})^{\tau-1} - \eta \cdot \frac{\partial E_n}{\partial w_{ij}^{(2)}}$$

The gradients are computed by **back-propagation**:

$$(1)\frac{\partial E_n}{\partial w_{ij}^{(2)}} \qquad (2)\frac{\partial E_n}{\partial w_{ij}^{(1)}}$$

Bonus 4: Back-propagation

After some analytical computations:

$$\frac{\partial E_n}{\partial w_{ij}^{(2)}} = \Delta_{ni}^{(2)} \cdot z_{nj}^{(2)}$$

$$\frac{\partial E_n}{\partial w_{ij}^{(1)}} = \Delta_{ni}^{(1)} \cdot x_{nj}$$

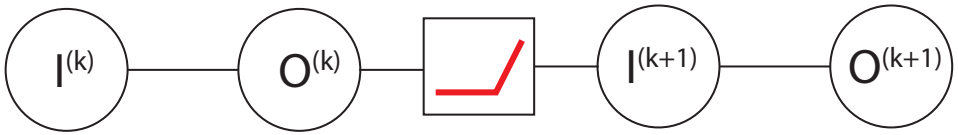
with errors:

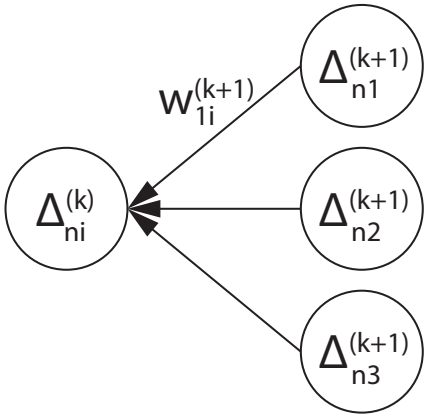
$$\Delta_{ni}^{(2)} = (y_{ni} - t_{ni})$$

$$\Delta_{ni}^{(1)} = ReLU'(z_{ni}^{(1)}) \sum_{p=1}^{3} w_{pi}^{(2)} \Delta_{np}^{(2)}$$

Bonus 4: Back-propagation

Relations between layer k and $k+1$:





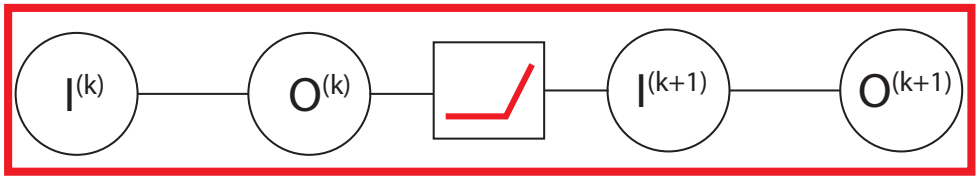
The **back-propagation equations**:

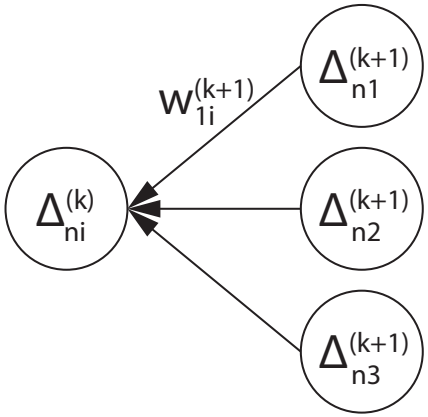
$$\frac{\partial E_n}{\partial w_{ij}^{(k)}} = \Delta_{ni}^{(k)} \cdot I_{nj}^{(k)}$$

$$\Delta_{ni}^{(k)} = ReLU'(O_{ni}^{(k)}) \sum_{p=1}^{3} w_{pi}^{(k+1)} \Delta_{np}^{(k+1)}$$

Bonus 4: Back-propagation

Relations between layer k and $k+1$:





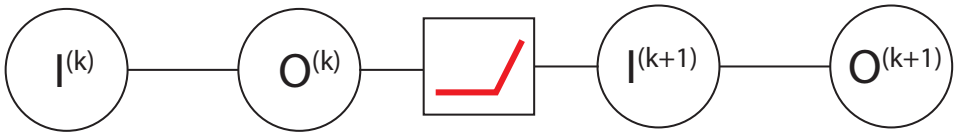
The **back-propagation equations**:

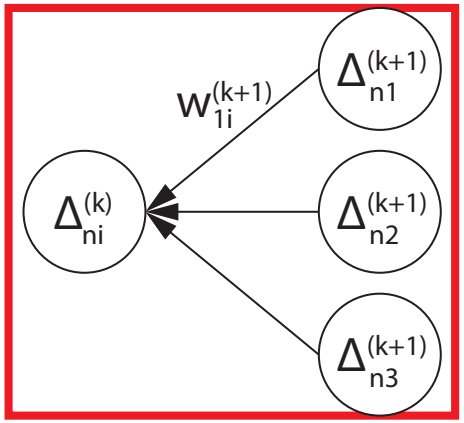
$$\frac{\partial E_n}{\partial w_{ij}^{(k)}} = \Delta_{ni}^{(k)} \cdot I_{nj}^{(k)}$$

$$\Delta_{ni}^{(k)} = ReLU'(O_{ni}^{(k)}) \sum_{p=1}^{3} w_{pi}^{(k+1)} \Delta_{np}^{(k+1)}$$

Bonus 4: Back-propagation

Relations between layer k and $k+1$:





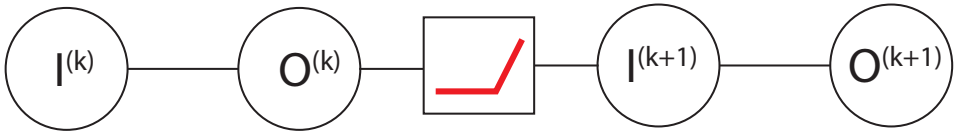
The **back-propagation equations**:

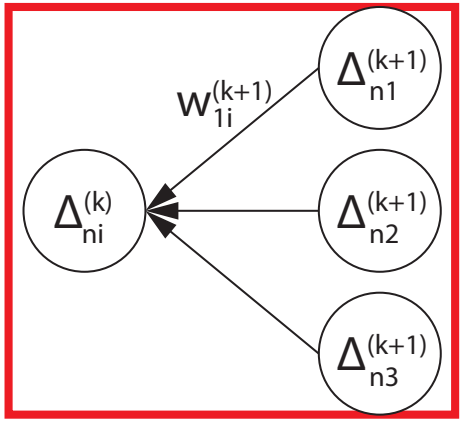
$$\frac{\partial E_n}{\partial w_{ij}^{(k)}} = \Delta_{ni}^{(k)} \cdot I_{nj}^{(k)}$$

$$\Delta_{ni}^{(k)} = ReLU'(O_{ni}^{(k)}) \sum_{p=1}^{3} w_{pi}^{(k+1)} \Delta_{np}^{(k+1)}$$

Bonus 4: Back-propagation

Relations between layer k and $k+1$:





The **back-propagation equations**:

$$\frac{\partial E_n}{\partial w_{ij}^{(k)}} = \Delta_{ni}^{(k)} \cdot I_{nj}^{(k)}$$

$$\Delta_n^{(k)} = ReLU'(O_n^{(k)}) \circ \left(W^{(k+1)}\right)^{\mathrm{T}} \Delta_n^{(k+1)}$$

Bonus 5: Neural Network for Regression

Two modifications to go

from NN classification to NN regression

Modification 1

Remove the SoftMax function

Modification 2

Replace the cross-entropy error by

the sum-of-squares error:

$$E(W) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} |t_{nk} - y_{nk}(W)|^2$$

QUESTIONS ?

Coffee break

COFFEE BREAK

The iris decision tree

The iris decision tree

input output

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.3	3.3	6.0	2.5	virginica

The iris decision tree

A decision tree is a model

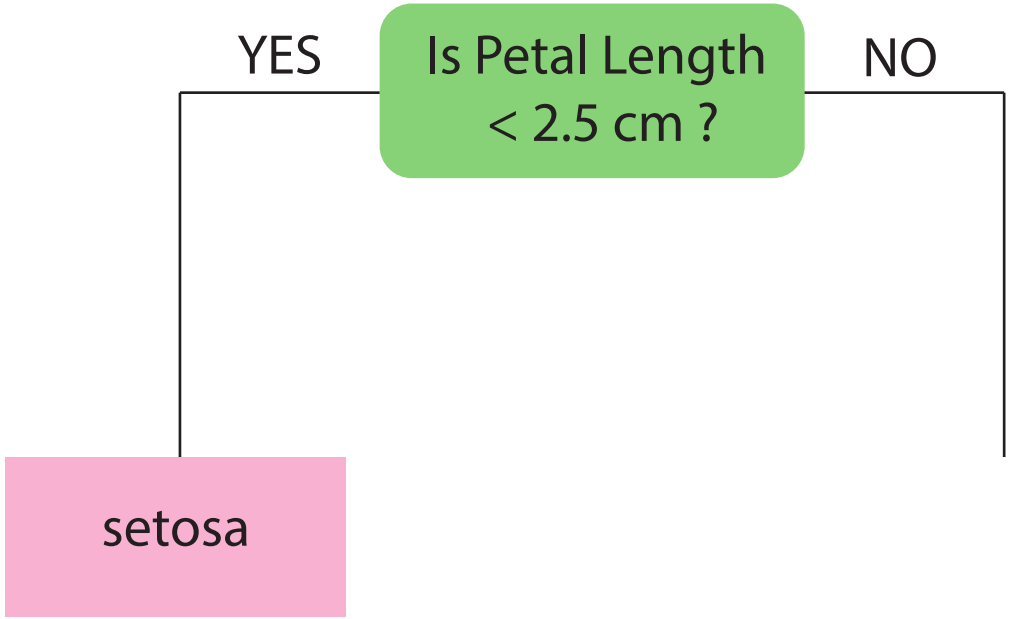
that predicts the output

by answering questions on the input

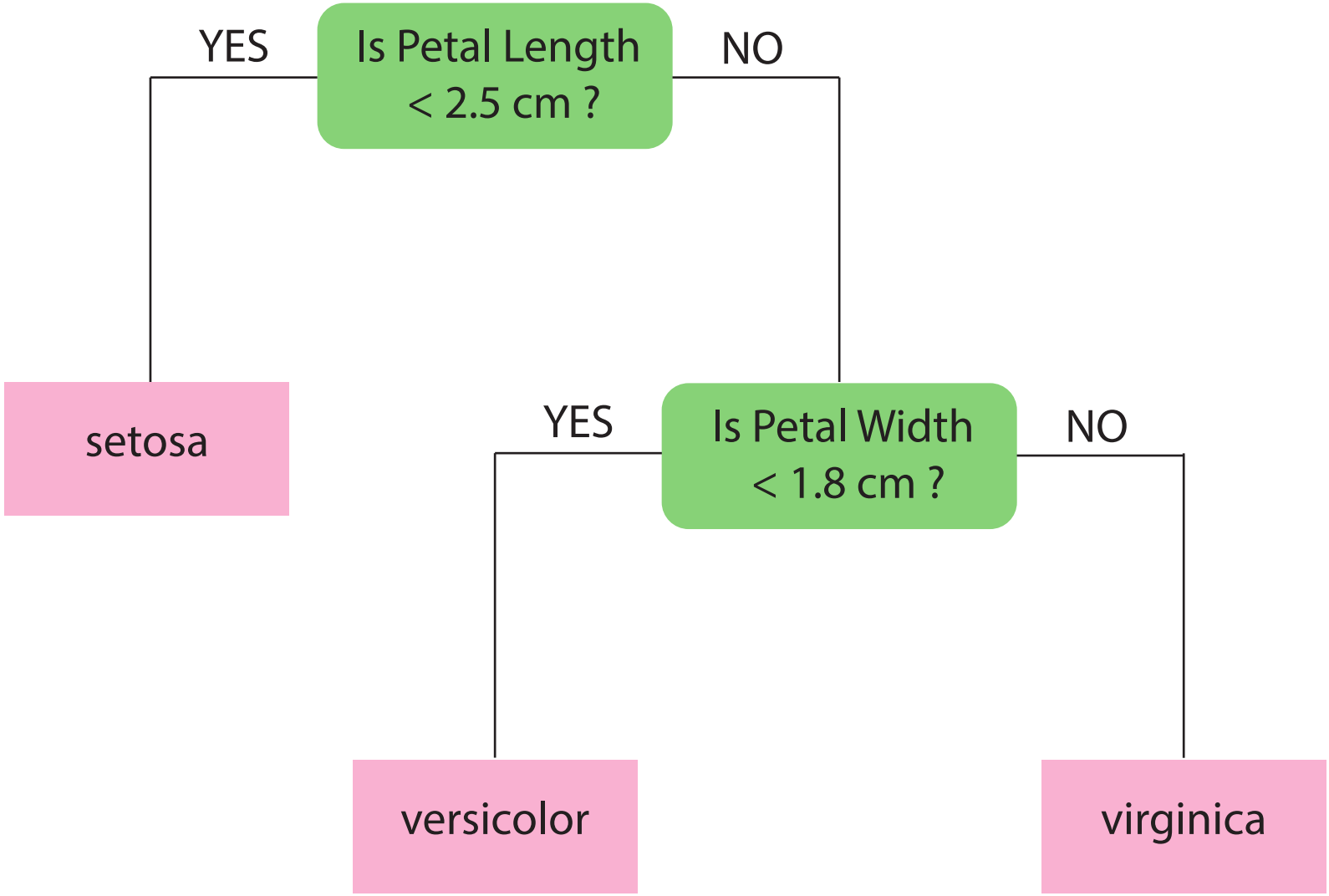
The iris decision tree

EXAMPLE

The iris decision tree

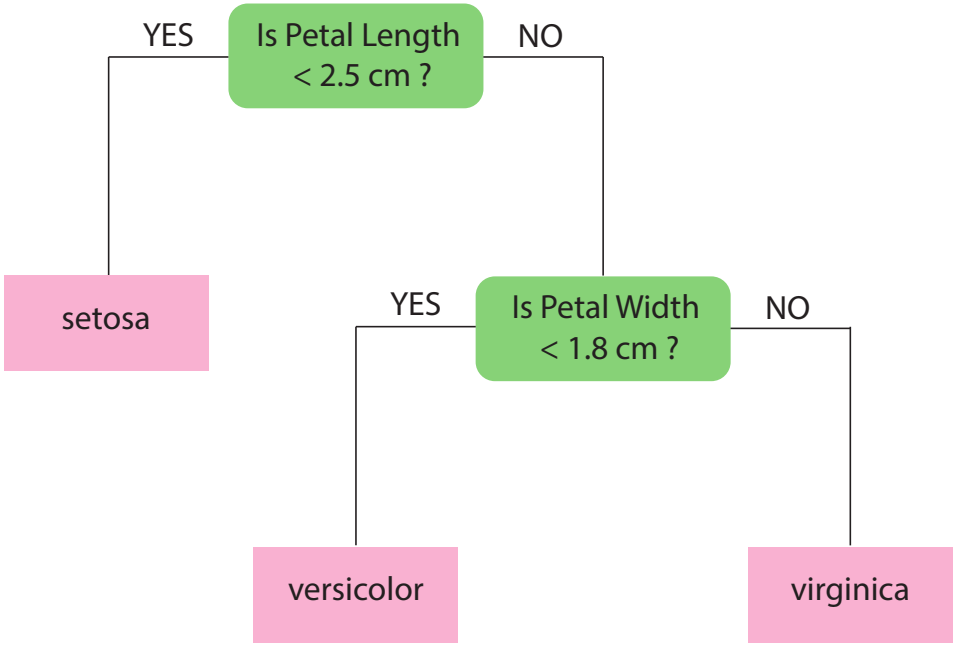


The iris decision tree



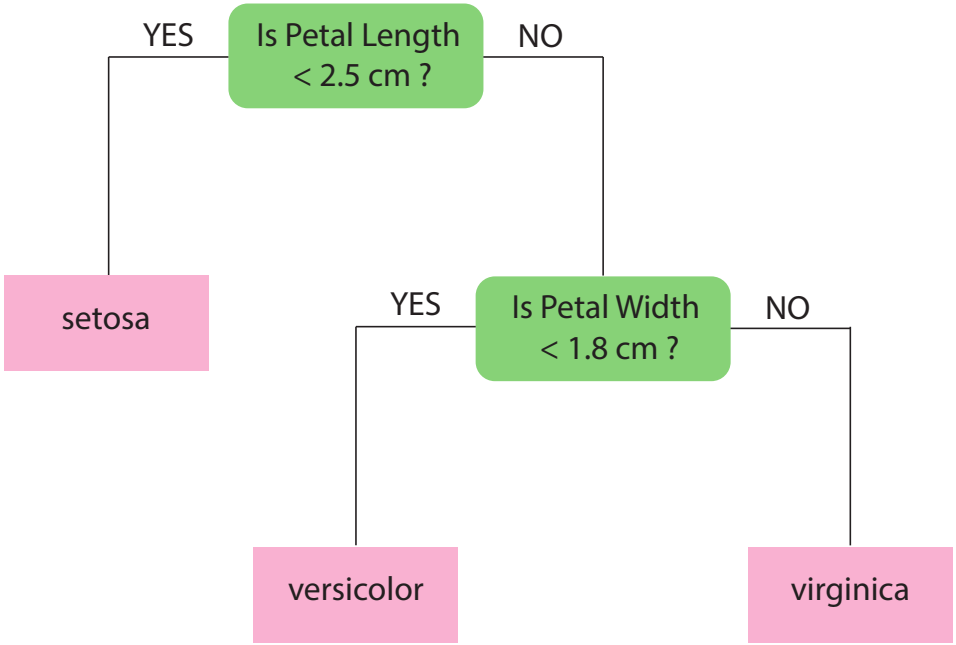
The iris decision tree

input				output
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.3	3.3	6.0	2.5	virginica



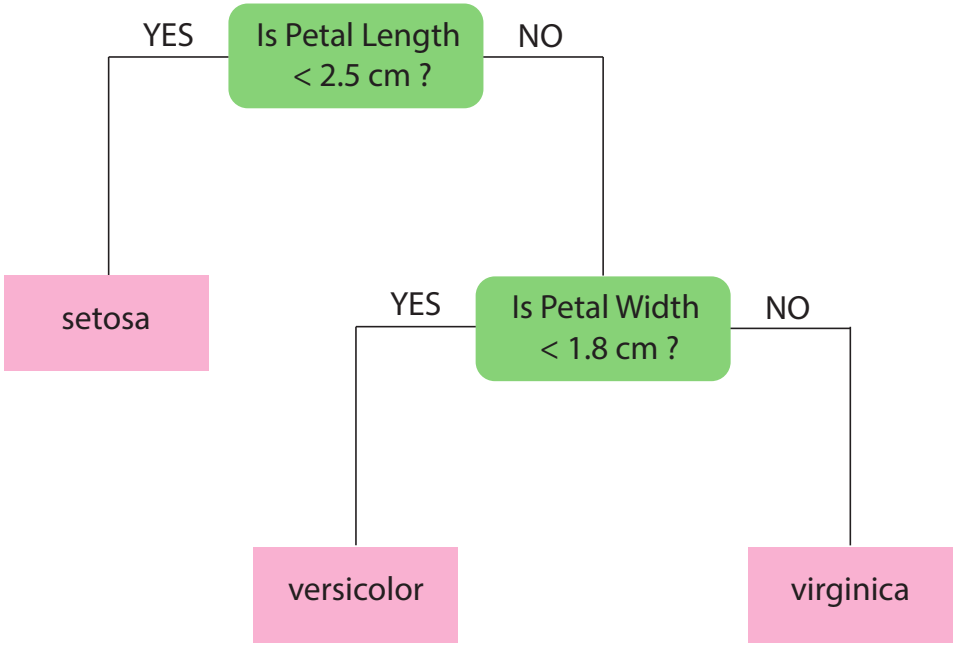
The iris decision tree

input				output
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.3	3.3	6.0	2.5	virginica



The iris decision tree

input				output
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.3	3.3	6.0	2.5	virginica



How to choose the splitting features ?

How to choose

the questions ?

How to choose the splitting features ?

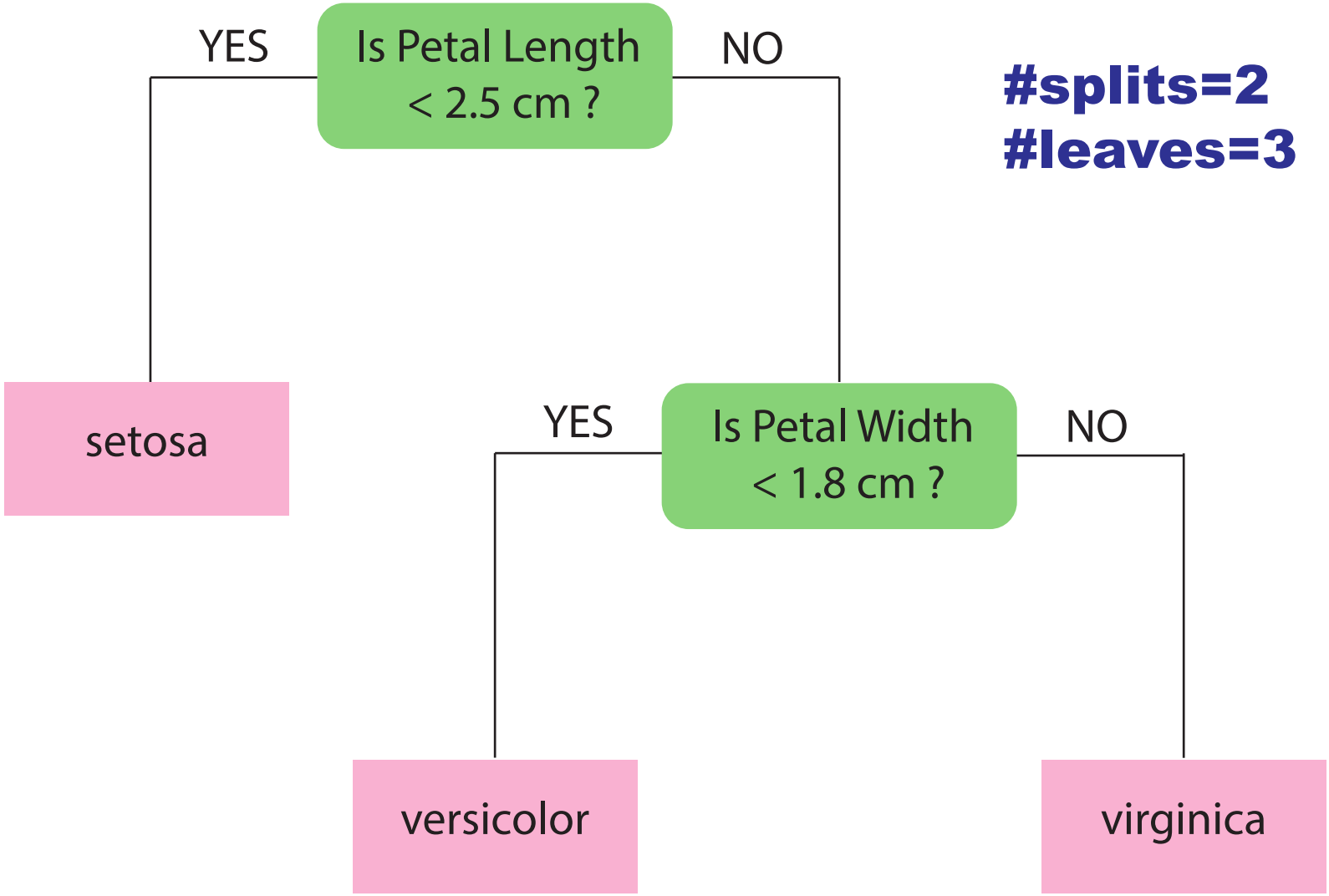
One wants to answer

as few as possible questions

to determine which species

a given plant belongs to

The iris decision tree



#splits=2
#leaves=3

How to choose the splitting features ?

This is a global optimization problem

that cannot be handled computationally

How to choose the splitting features ?

We start with all the training data

and choose a **locally** optimal question

that splits the data at each stage

Information

Information content of a leaf

If I take randomly a sample from the leaf,

how much do I know about its class

The iris decision tree

EXAMPLE

The iris decision tree

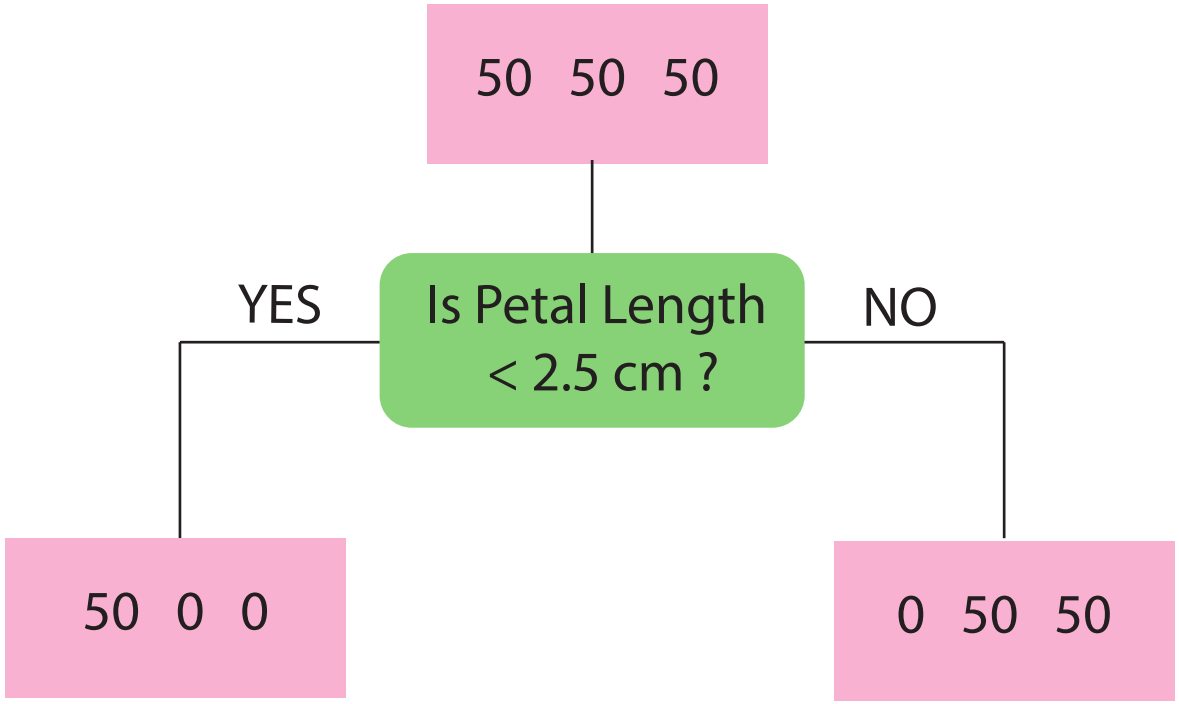
50 50 50

The iris decision tree

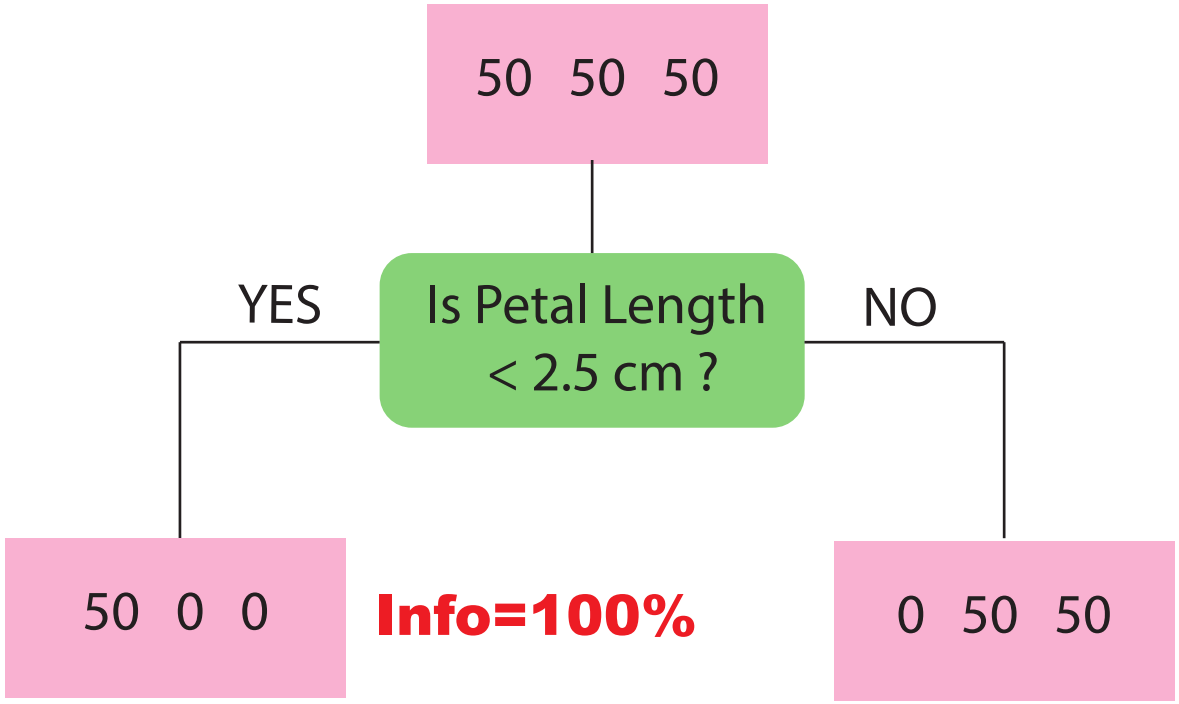
50 50 50

Info=0

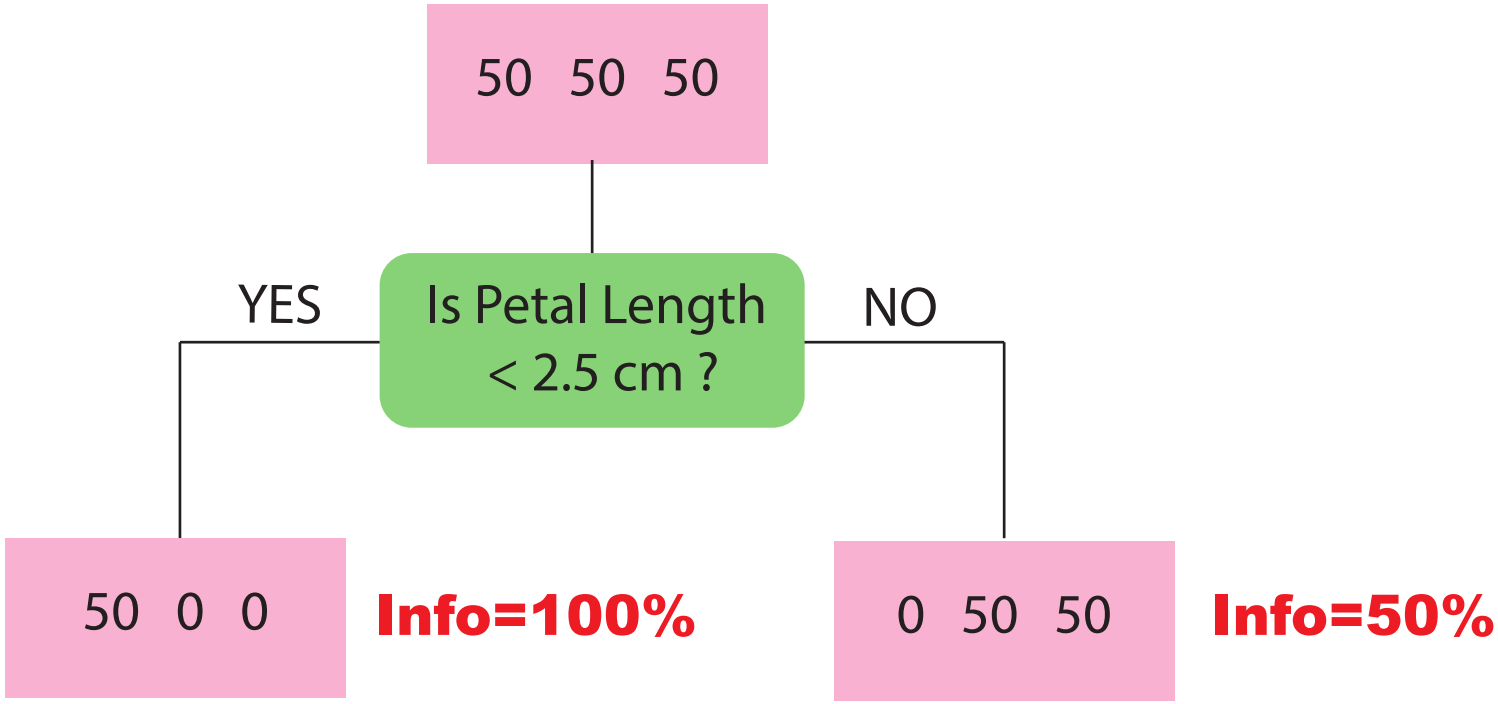
The iris decision tree



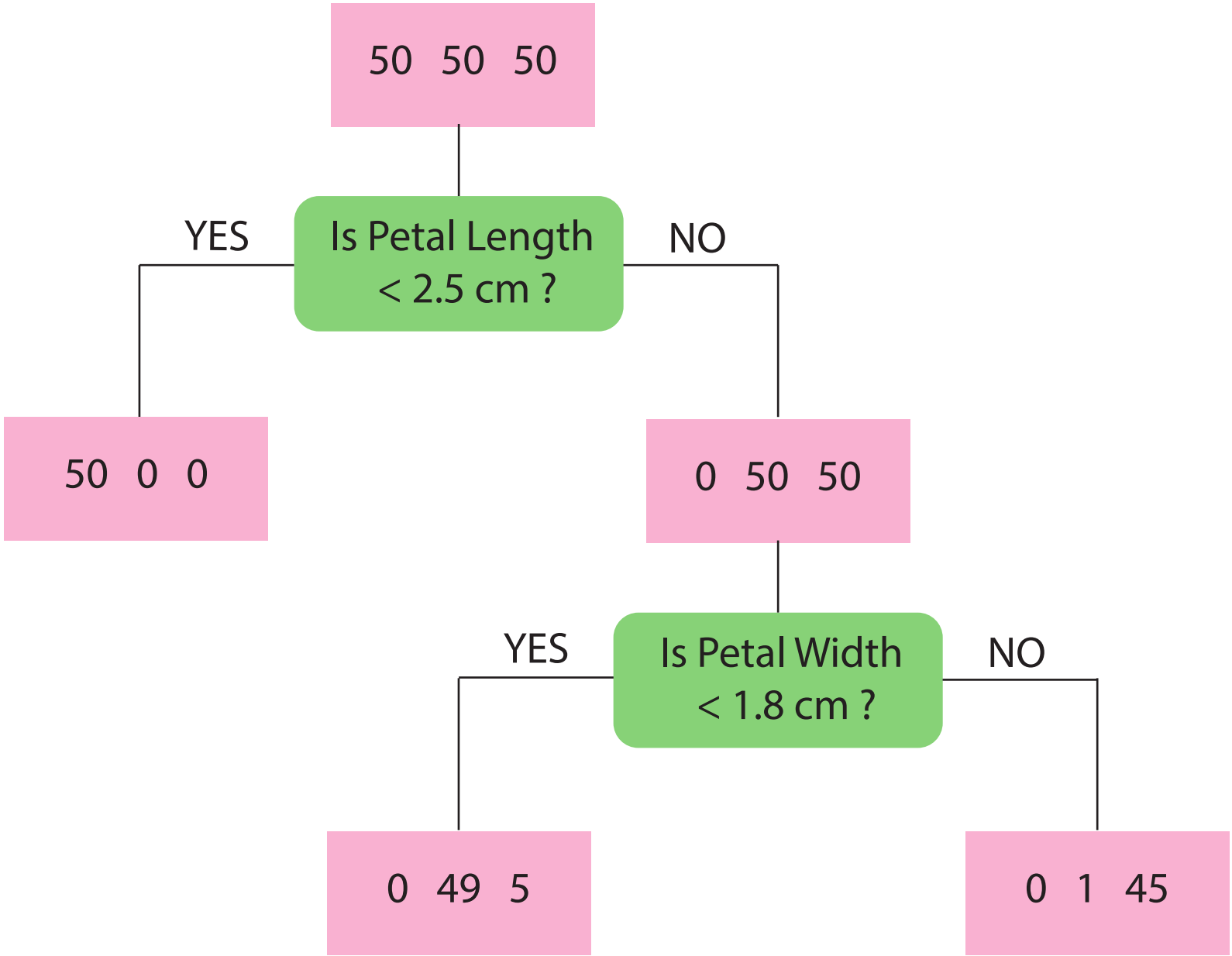
The iris decision tree



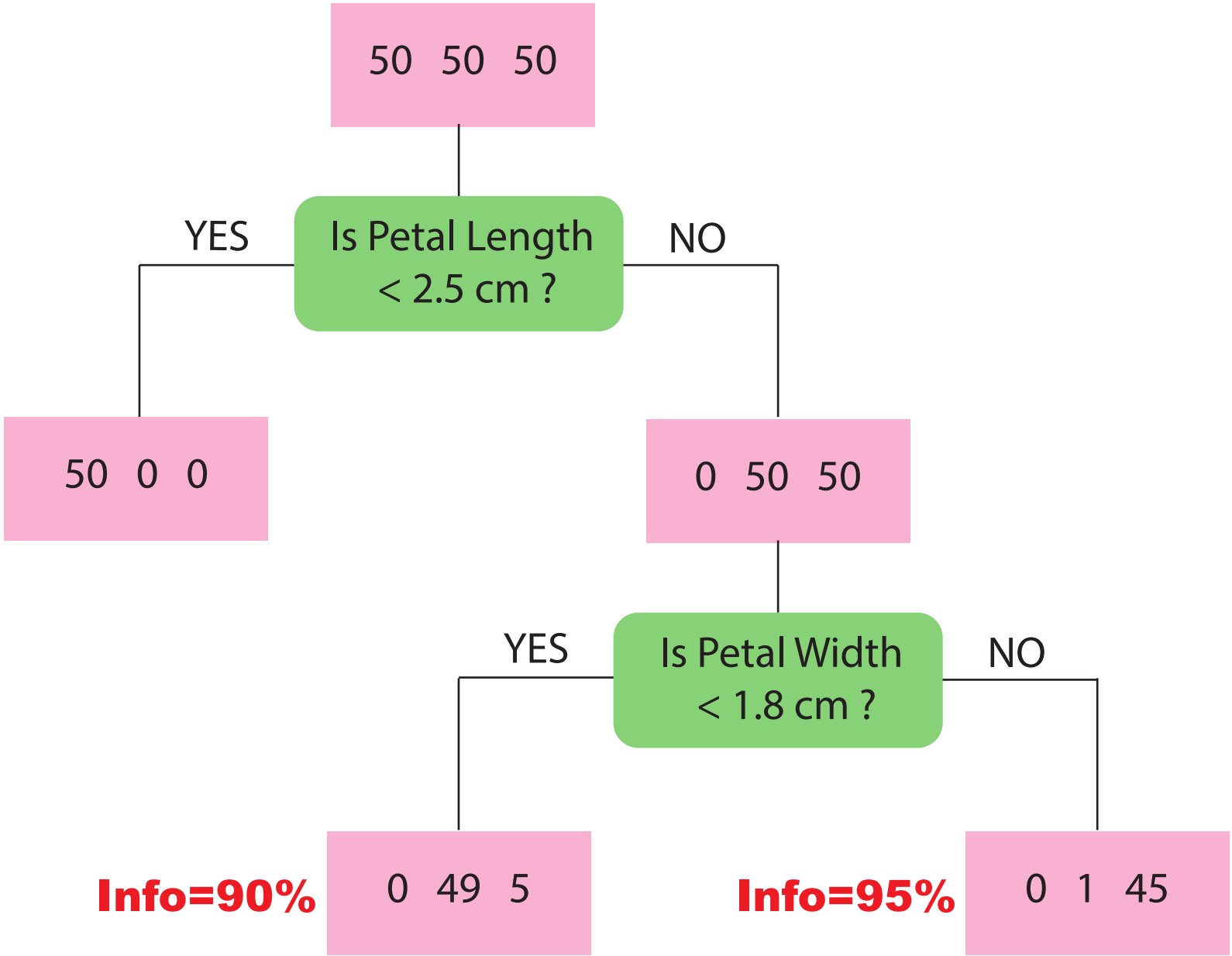
The iris decision tree



The iris decision tree



The iris decision tree



How to choose the splitting features ?

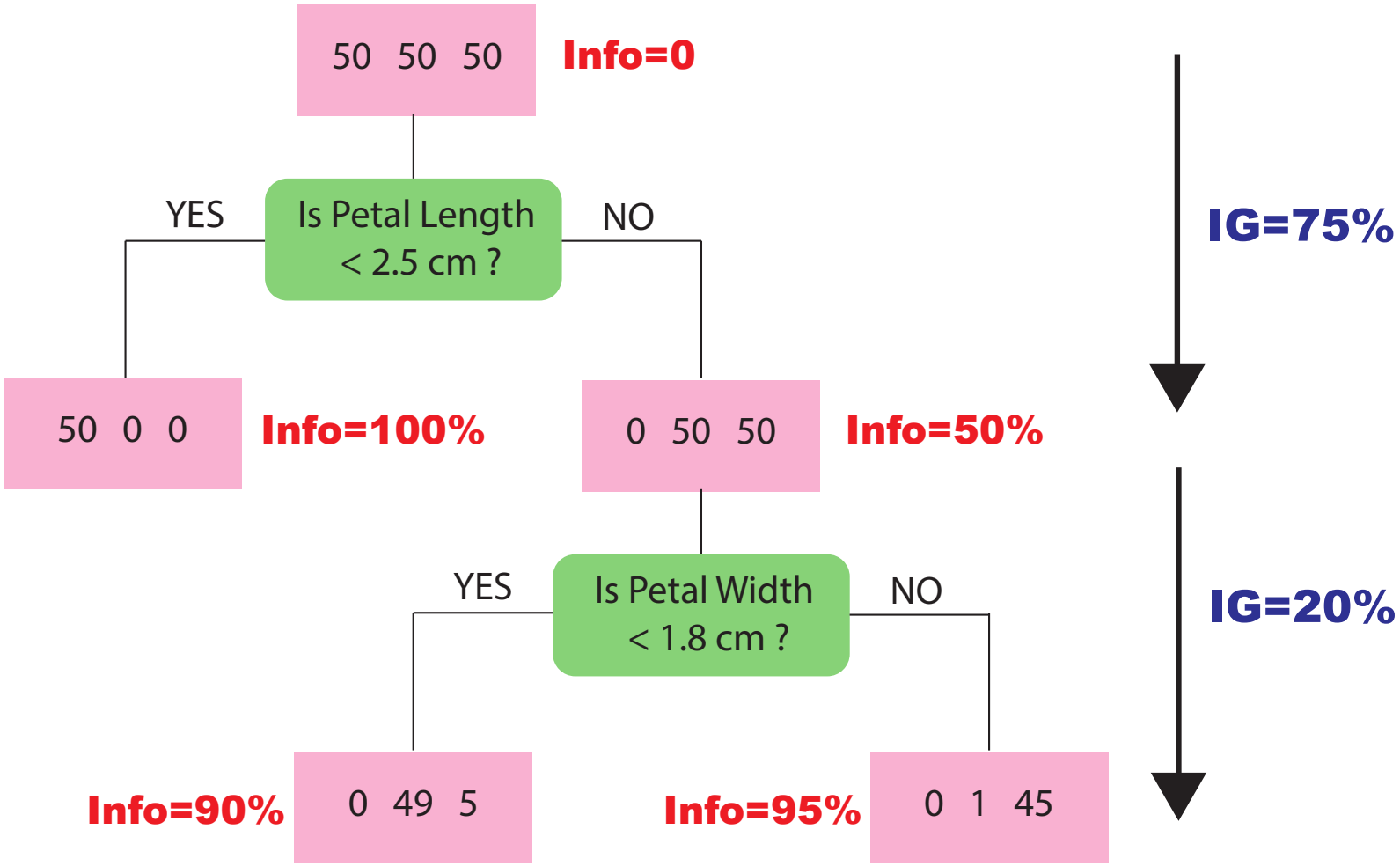
We start at the root

and split the training data on the feature

that results in the largest **information gain**

$$IG = Info(D_{\text{children}}) - Info(D_{\text{parent}})$$

How to choose the splitting features ?



How to choose the splitting features ?

Assume $Info(D_{\text{parent}}) \in [0, 1]$:

$$
\begin{aligned}
IG &= Info(D_{\text{children}}) - Info(D_{\text{parent}}) \\
&= [1 - Info(D_{\text{parent}})] - [1 - Info(D_{\text{children}})] \\
&= LackInfo(D_{\text{parent}}) - LackInfo(D_{\text{children}})
\end{aligned}
$$

How to choose the splitting features ?

LackInfo is called the **impurity index I**:

$$IG = I(D_{\text{parent}}) - \left[\frac{N_{\text{left child}}}{N_{\text{parent}}} I(D_{\text{left child}}) + \frac{N_{\text{right child}}}{N_{\text{parent}}} I(D_{\text{right child}}) \right]$$

How to choose the splitting features ?

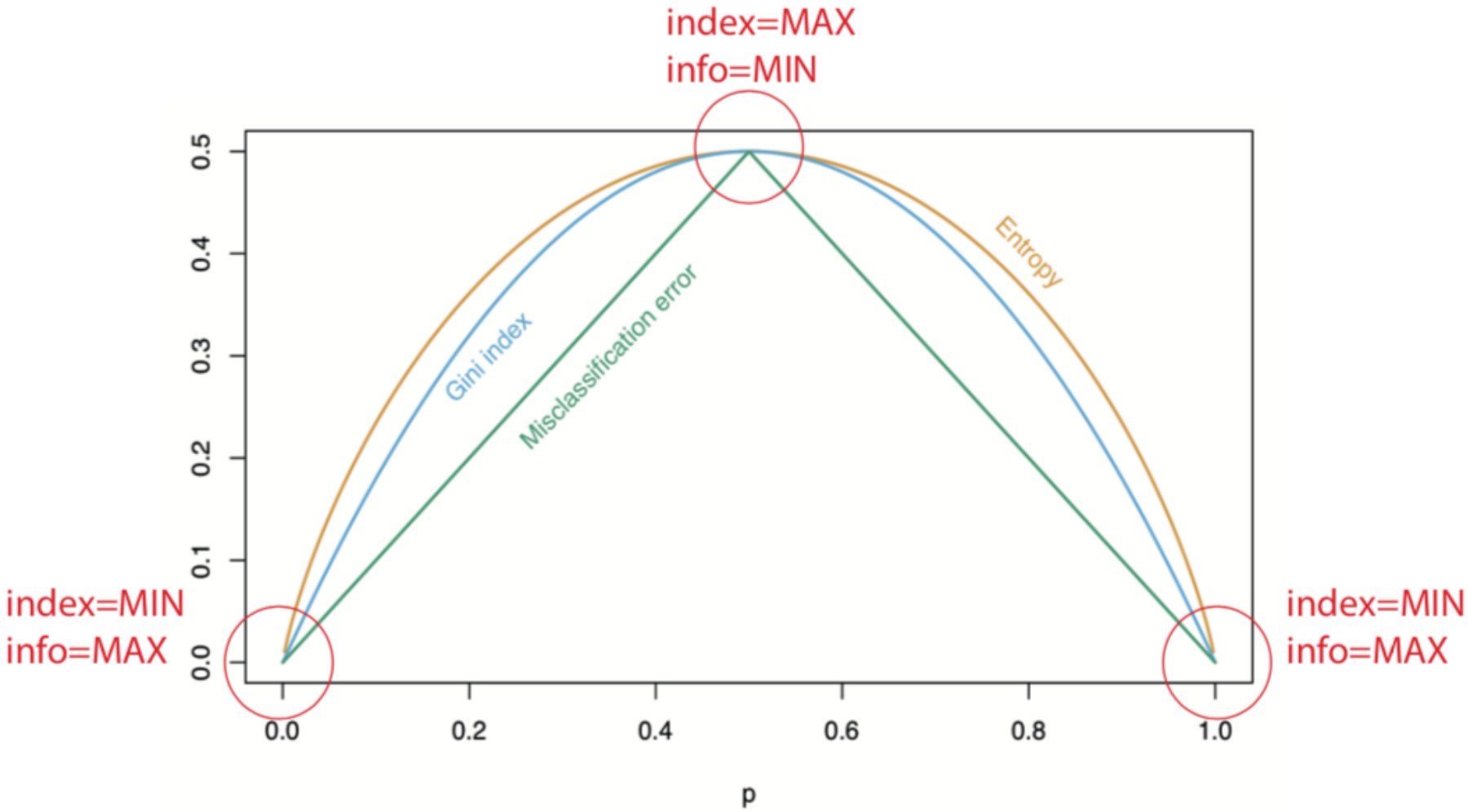
The **impurity index**:

$$
\begin{aligned}
I_E(D_\ell) &= 1 - \max_{c=1,2,3} \{p_{\ell c}\} \quad \text{(Misclassification error)} \\
I_H(D_\ell) &= -\sum_{c=1}^{3} p_{\ell c} \log_2(p_{\ell c}) \quad \text{(Entropy or Deviance)} \\
I_G(D_\ell) &= 1 - \sum_{c=1}^{3} p_{\ell c}^2 \quad \text{(Gini index)}
\end{aligned}
$$

where $p_{\ell c}$ is the proportion of data points at leaf ℓ that belongs to class $c = 1, 2, 3$.

How to choose the splitting features ?

In the case of 2 classes:



Conclusion: we want info=MAX or I=MIN at the children leaves

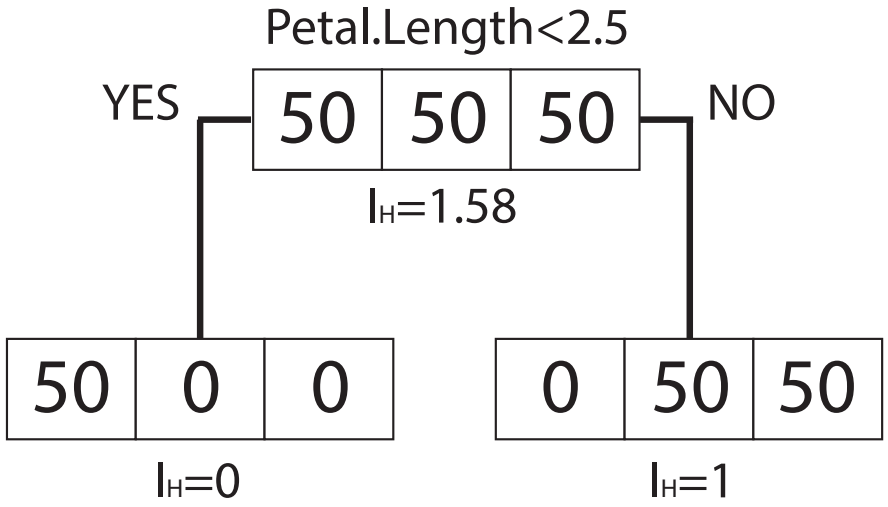
Ref: The Elements of Statistical Learning

How to choose the splitting features ?

EXAMPLE

How to choose the splitting features ?

Case A

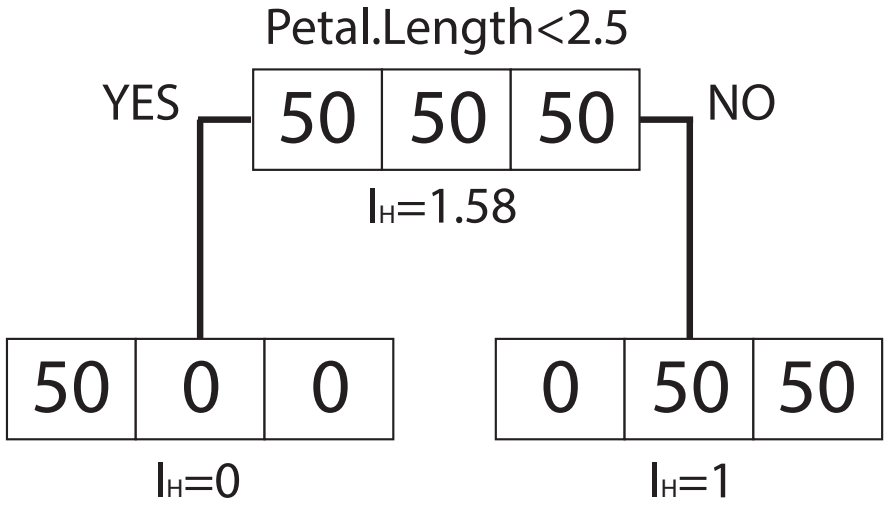


Information Gain:

IG=1.58-[(50/150)*0+(100/150)*1]
=0.92

How to choose the splitting features ?

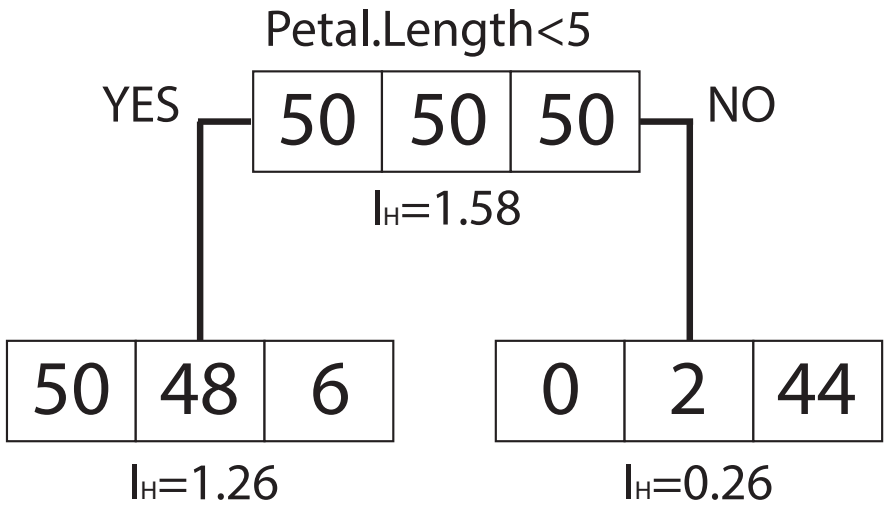
Case A



Information Gain:

IG=1.58-[(50/150)*0+(100/150)*1]
=0.92

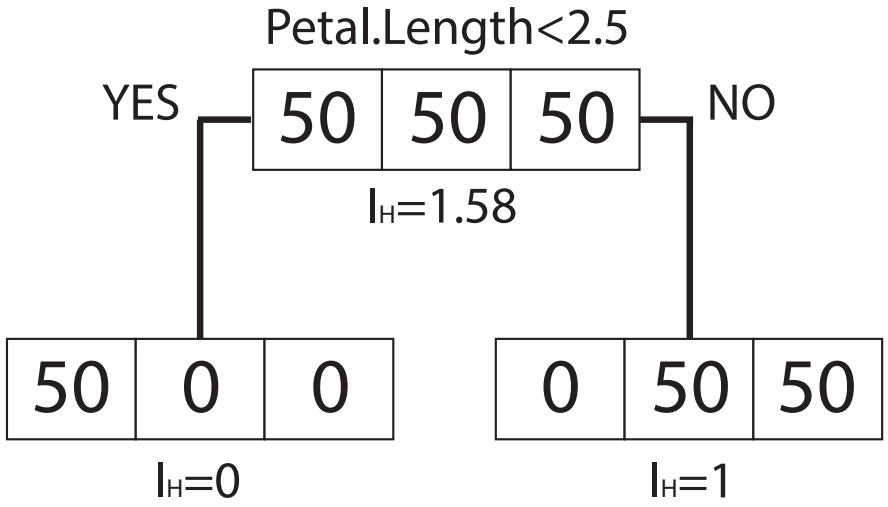
Case B



IG=1.58-[(104/150)*1.26+(46/150)*0.26]
=0.63

How to choose the splitting features ?

Case A

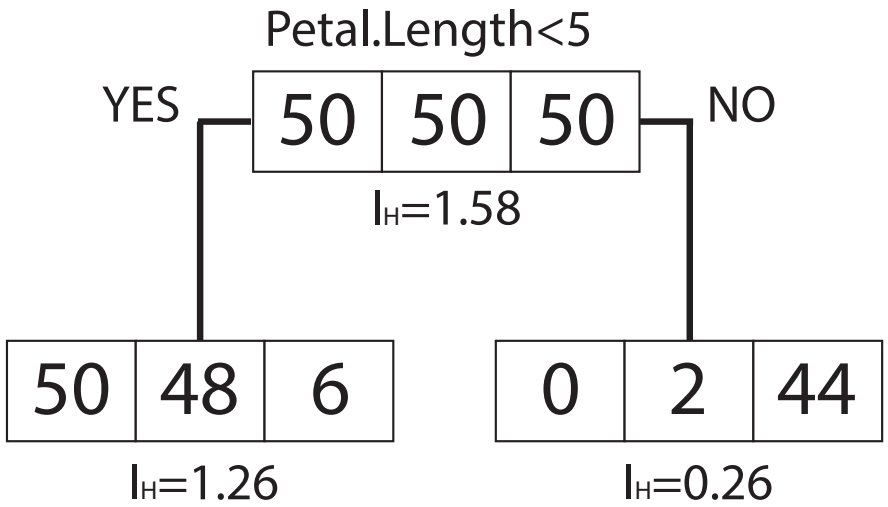


Information Gain:

IG=1.58-[(50/150)*0+(100/150)*1]
=0.92

IG(Case A) > IG(Case B)

Case B



IG=1.58-[(104/150)*1.26+(46/150)*0.26]
=0.63

How to choose the splitting features ?

How many splitting features

should be tested ?

How to choose the splitting features ?

Order the values from smallest to biggest:

n	1	2	...	149	150
Petal Length	1.0	1.1	...	6.7	6.9
Petal Width	0.1	0.1	...	2.5	2.5
Sepal Length	4.3	4.4	...	7.7	7.9
Sepal Width	2.0	2.2	...	4.2	4.4

There are 4x150=600 possible questions (splitting features)

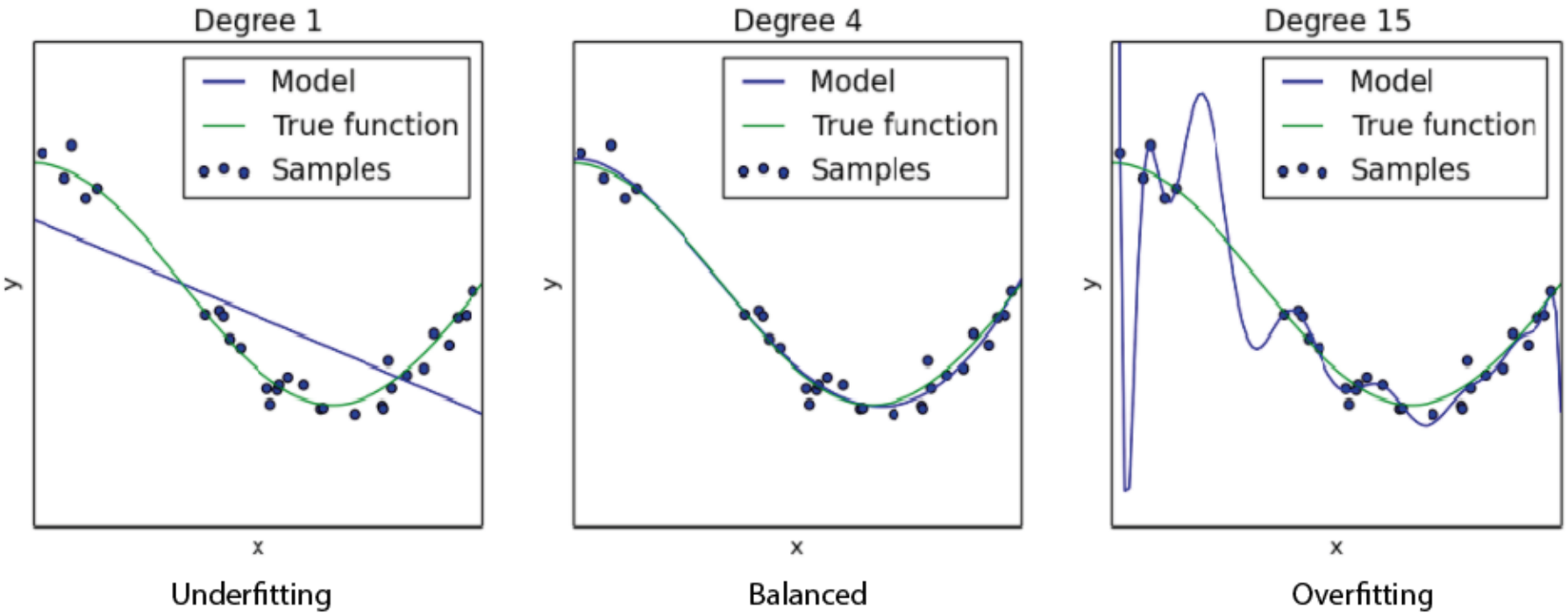
When to stop splitting features ?

When to stop splitting features ?

The answer is related to the problem of

under-fitting and over-fitting

When to stop splitting features ?



Ref: scikit-learn 0.18 documentation

When to stop splitting features ?

If the tree is too small

(too few splits),

it may under-fit the data

When to stop splitting features ?

If the tree is too big

(too many splits),

it may over-fit the data

When to stop splitting features ?

Method 1

Use stopping rules

that will prevent any node being split

if those conditions are not met

When to stop splitting features ?

minsplit

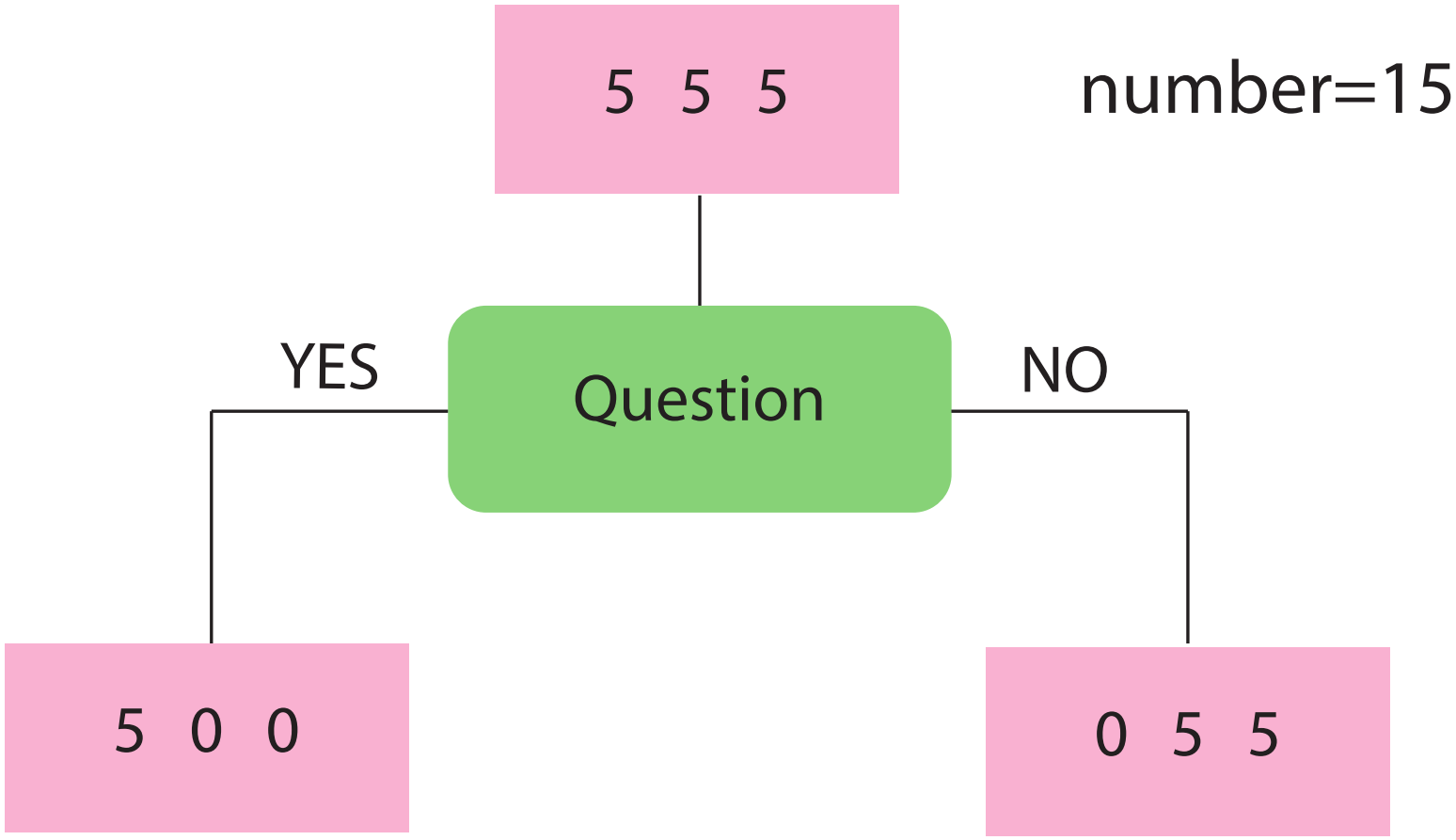
The minimum number of samples

that must exist in a node

in order for a split to be attempted

When to stop splitting features ?

minsplit=10



When to stop splitting features ?

minsplit=10

5 0 4

number=9

DO NOT SPLIT

When to stop splitting features ?

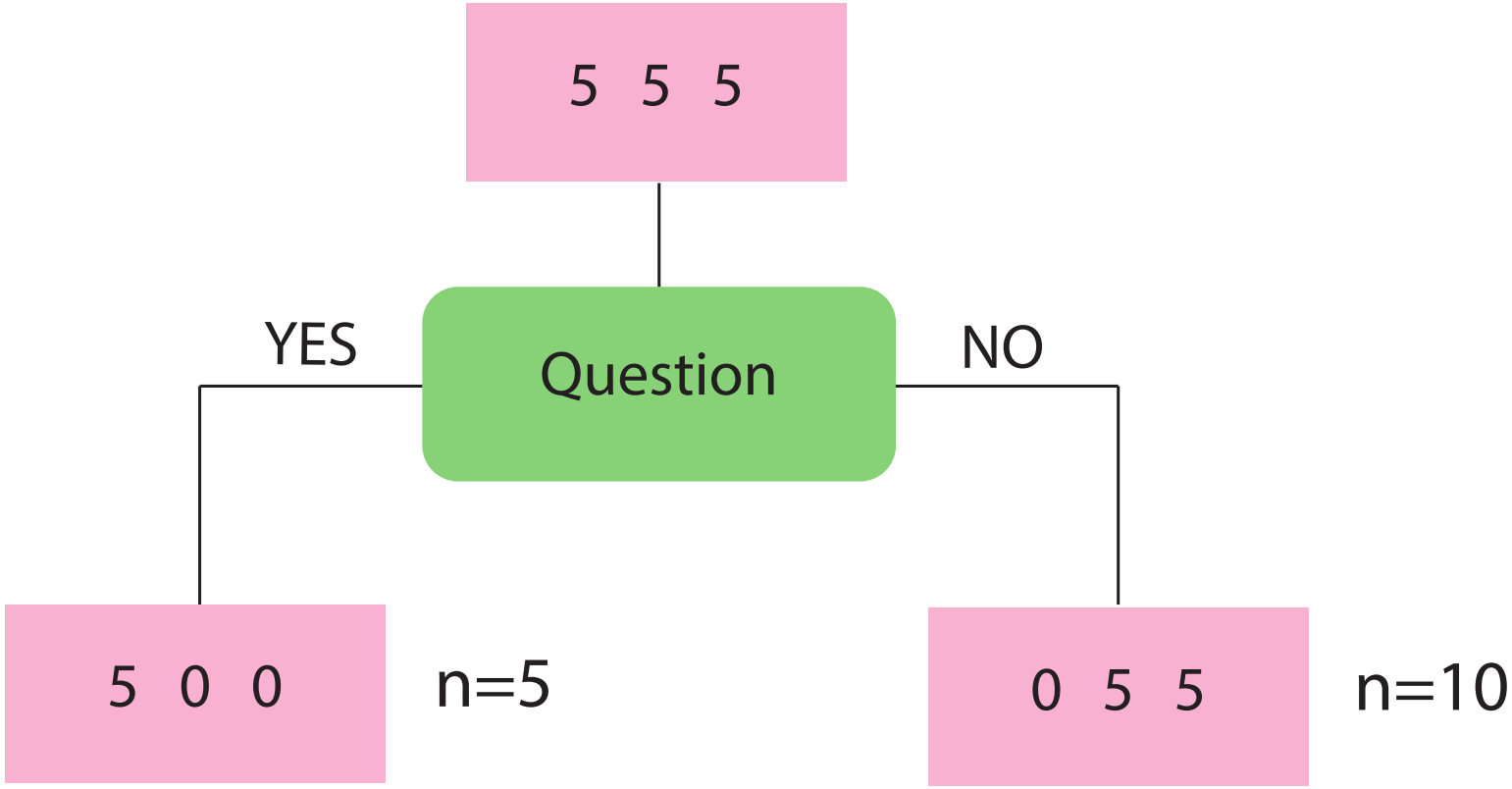
minbucket

The minimum number of samples

in any terminal leaf

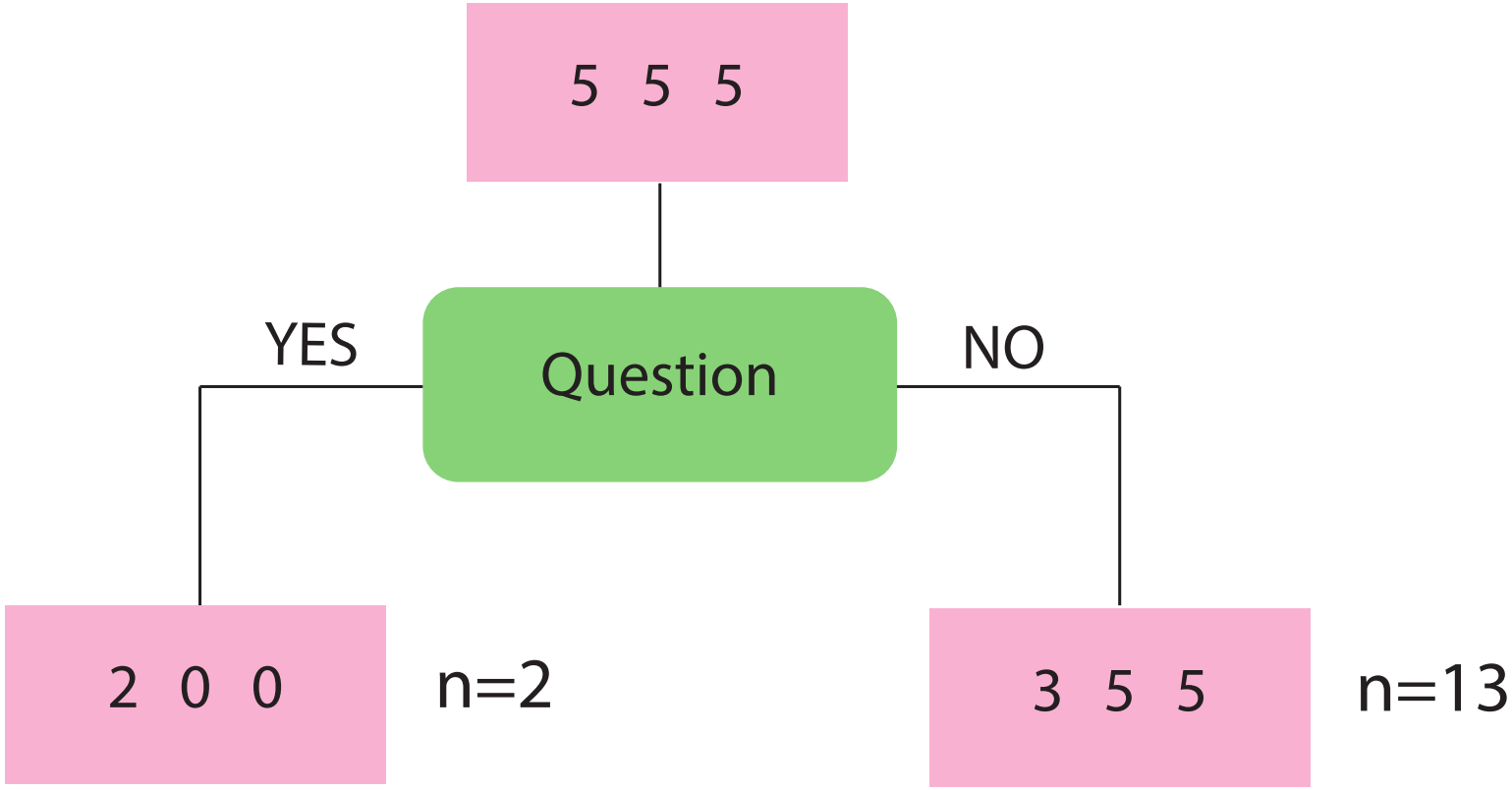
When to stop splitting features ?

minbucket=5



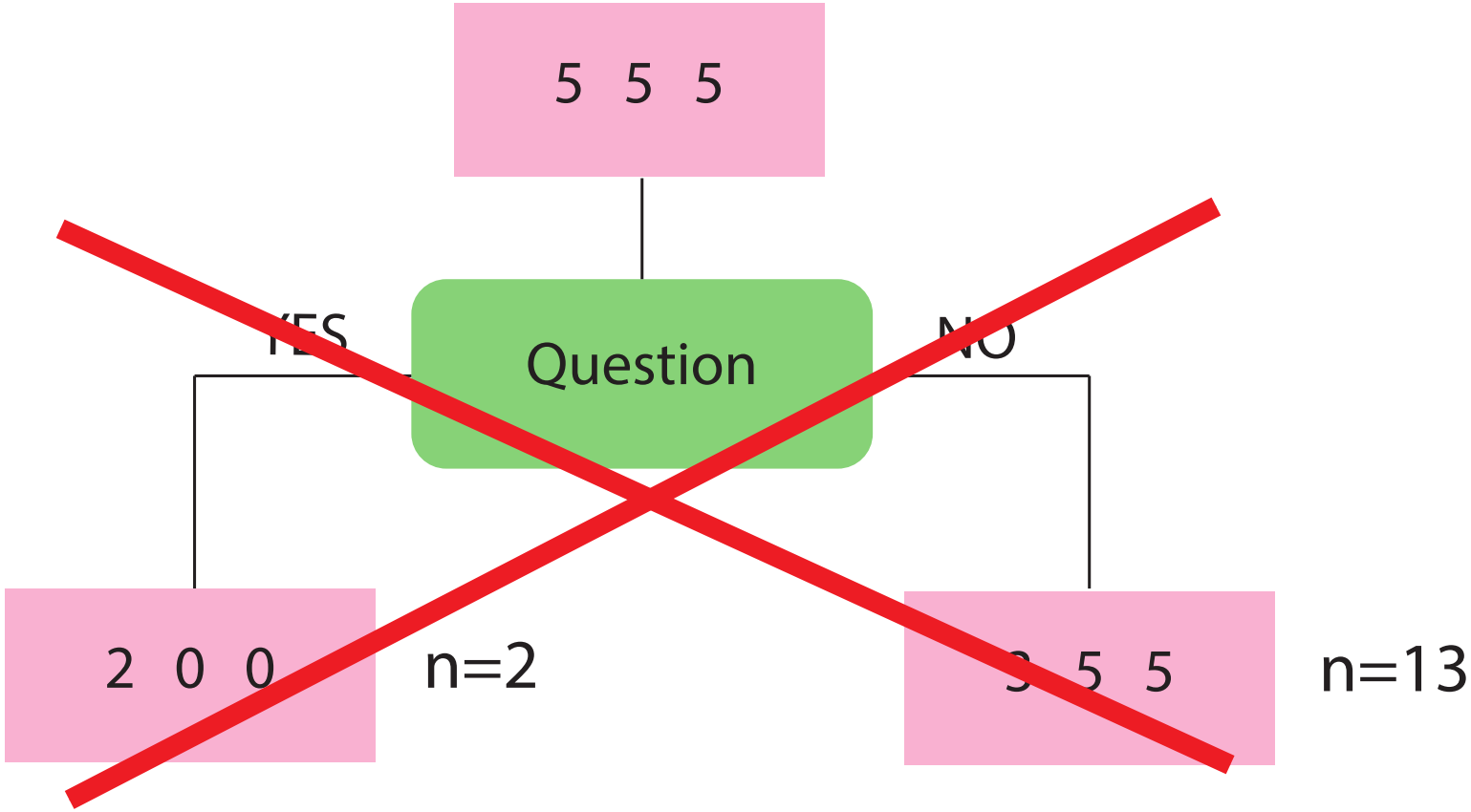
When to stop splitting features ?

minbucket=5



When to stop splitting features ?

minbucket=5



When to stop splitting features ?

Method 2

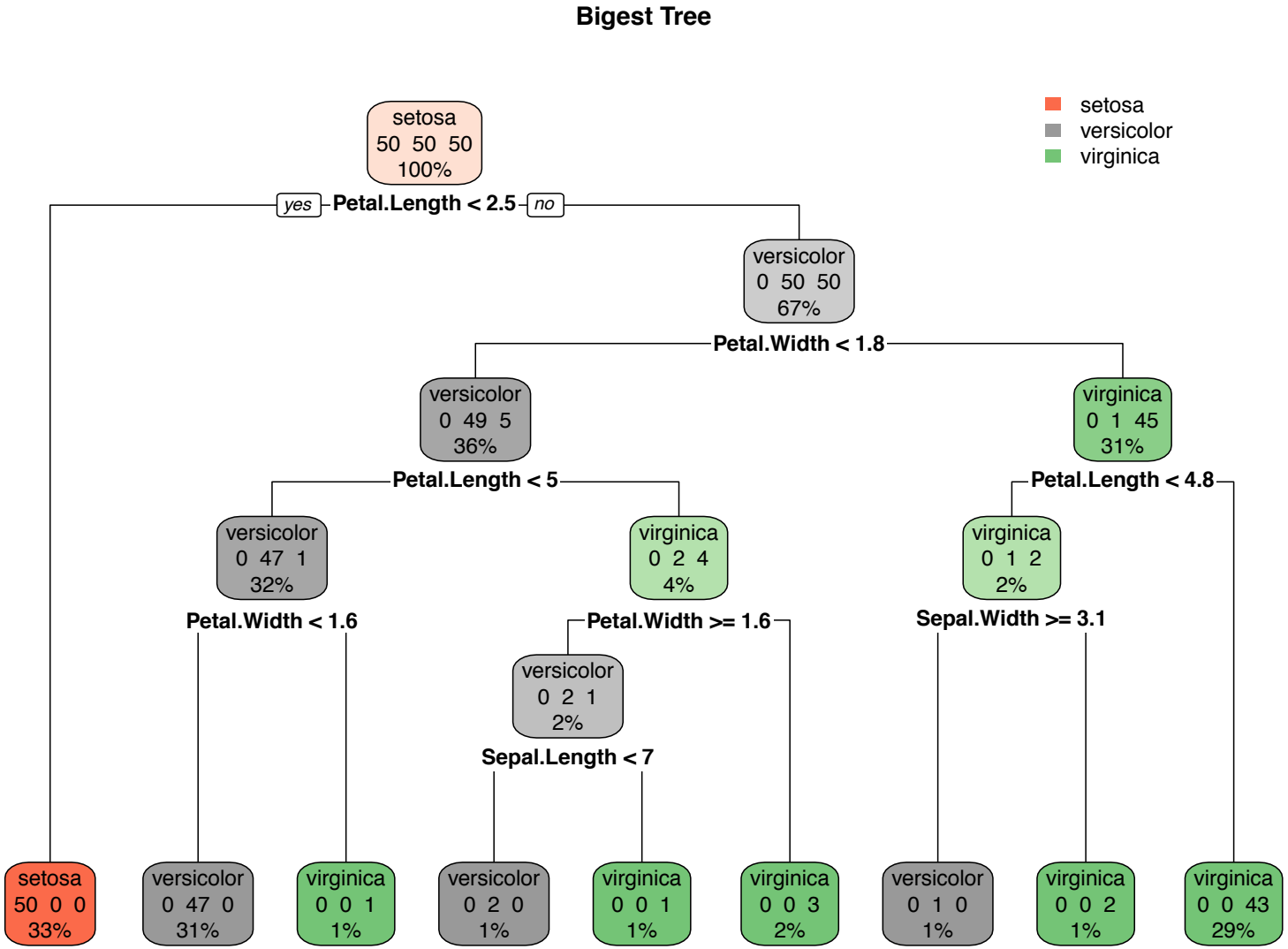
Prune the biggest tree from bottom-up:

stop when reaching minimum validation error

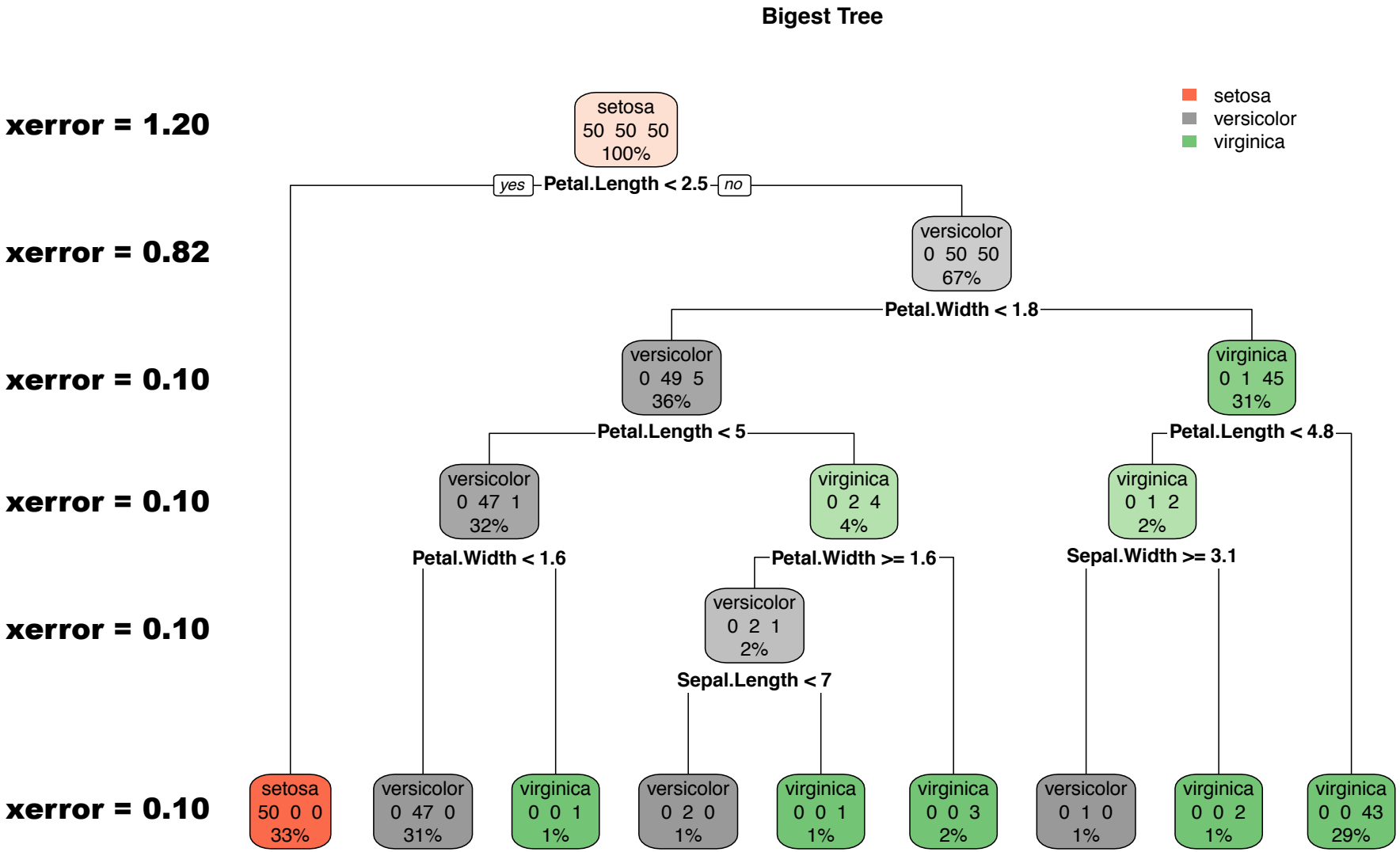
When to stop splitting features ?

EXAMPLE

When to stop splitting features ?



When to stop splitting features ?



When to stop splitting features ?

xerror = 1.20

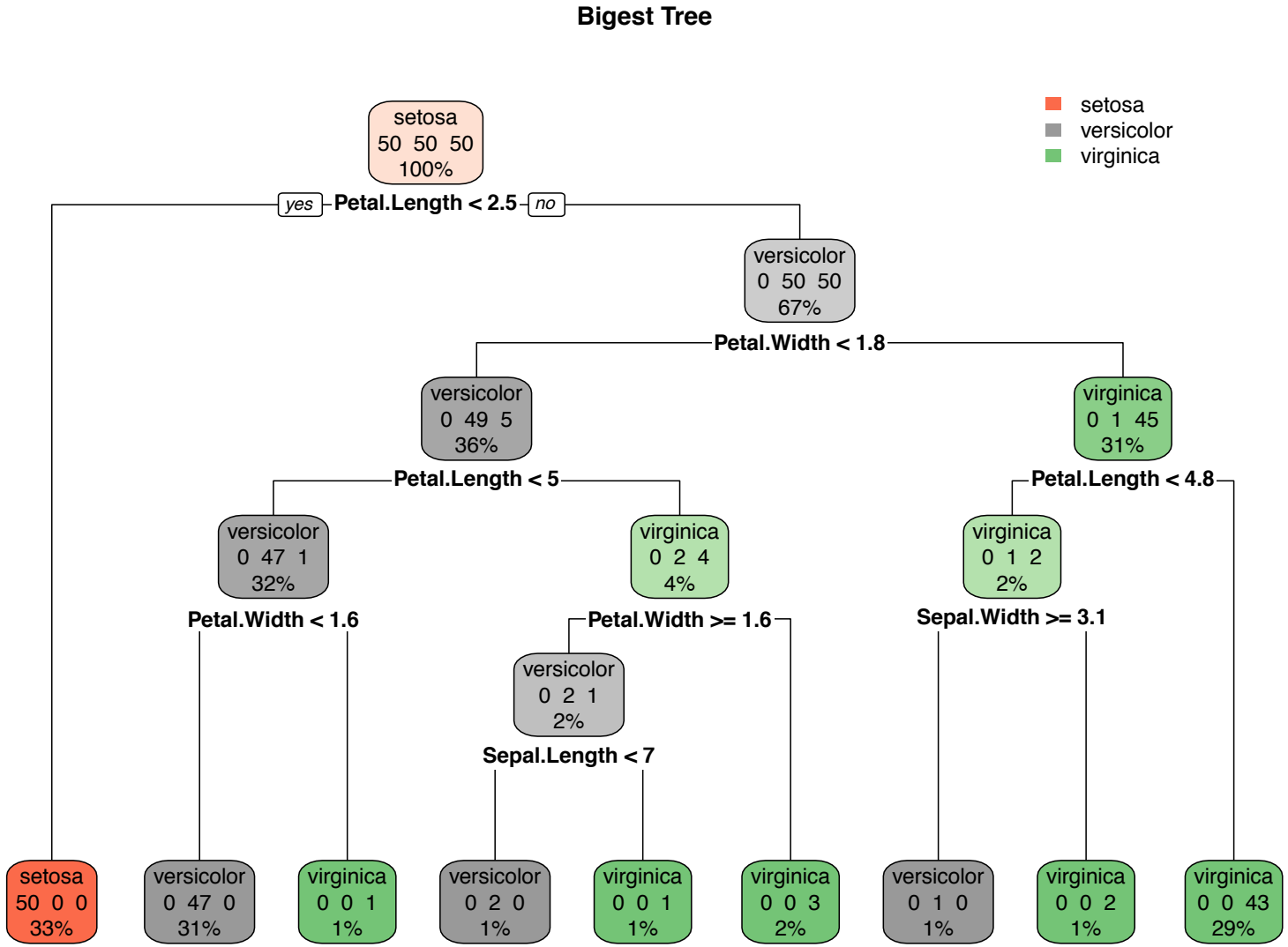
xerror = 0.82

xerror = 0.10

xerror = 0.10

xerror = 0.10

xerror = 0.10



When to stop splitting features ?

xerror = 1.20

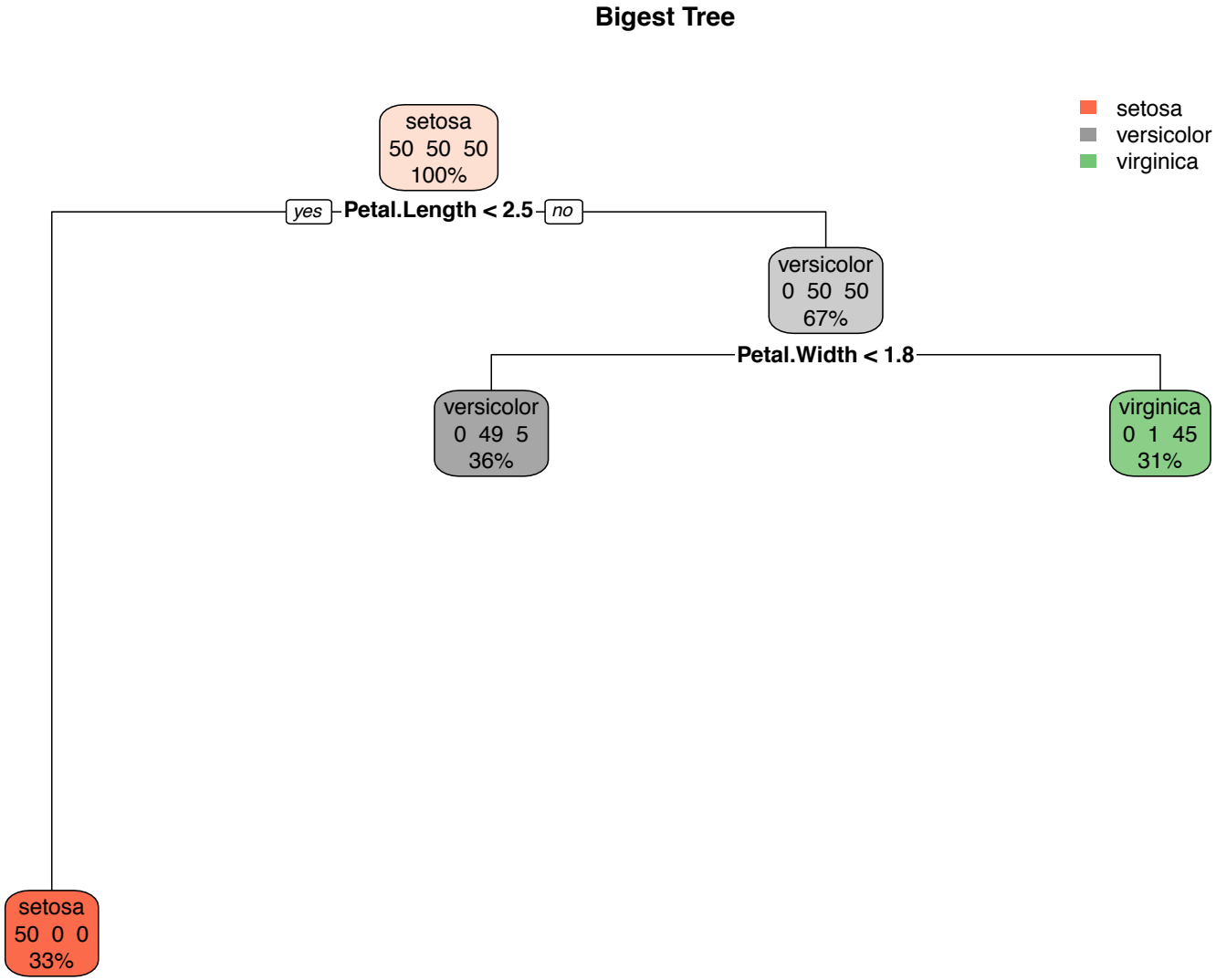
xerror = 0.82

xerror = 0.10

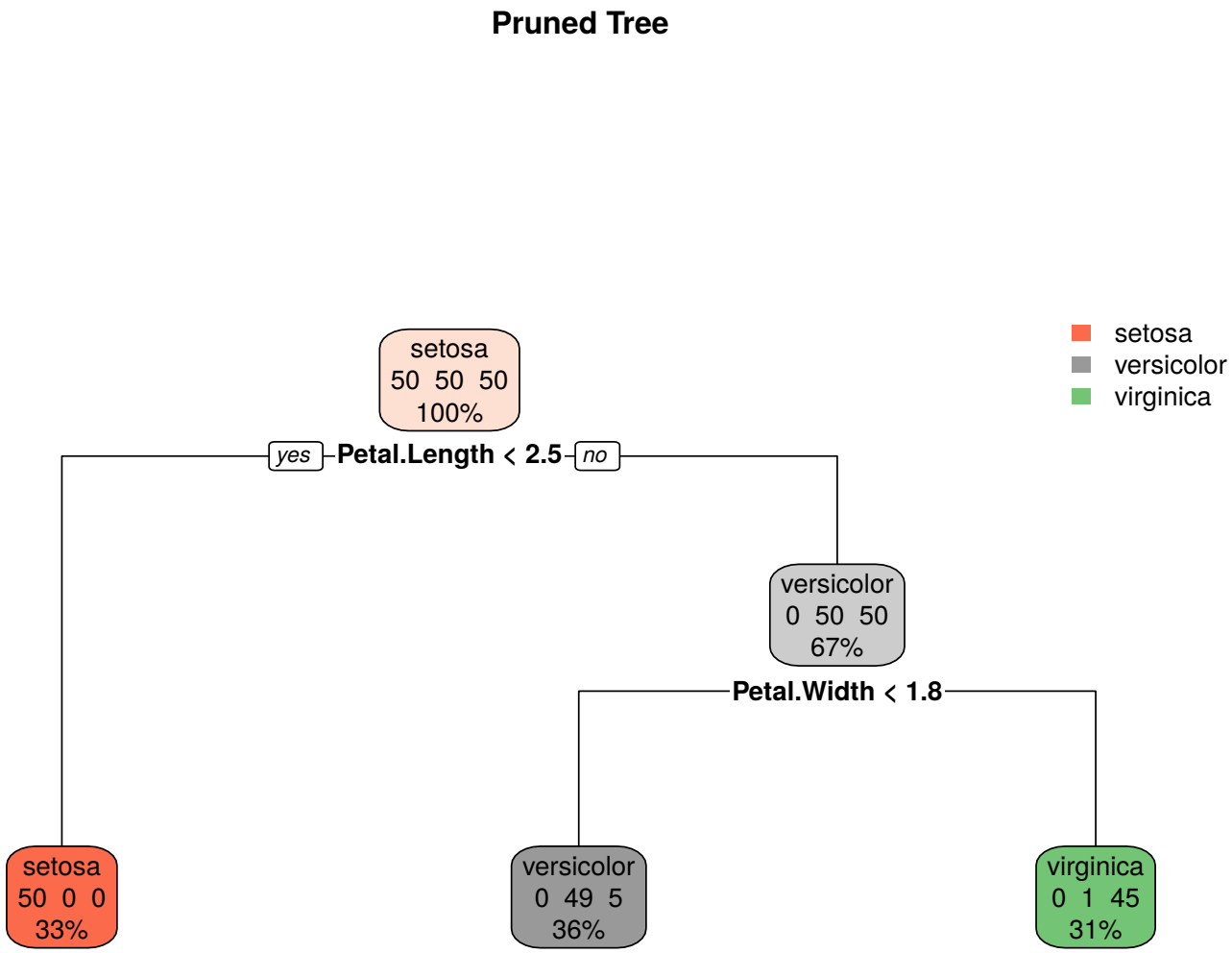
xerror = 0.10

xerror = 0.10

xerror = 0.10



When to stop splitting features ?



How to evaluate

the decision tree performances ?

How to evaluate the decision tree performances ?

At first we apply the **majority class rule** at each end node:

49 11 5	→	setosa
1 **39** 13	→	versicolor
10 3 **29**	→	virginica

How to evaluate the decision tree performances ?

Apply the decision tree to test data to get **confusion matrix**:

	predictions		
actuals	setosa	versicolor	virginica
setosa	14	0	0
versicolor	0	9	0
virginica	0	2	10

Use the following **performance metrics**:

$$\text{Accuracy rate} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{33}{35} = 94\%$$

$$\text{Error rate} = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}} = \frac{2}{35} = 6\%$$

Weakness

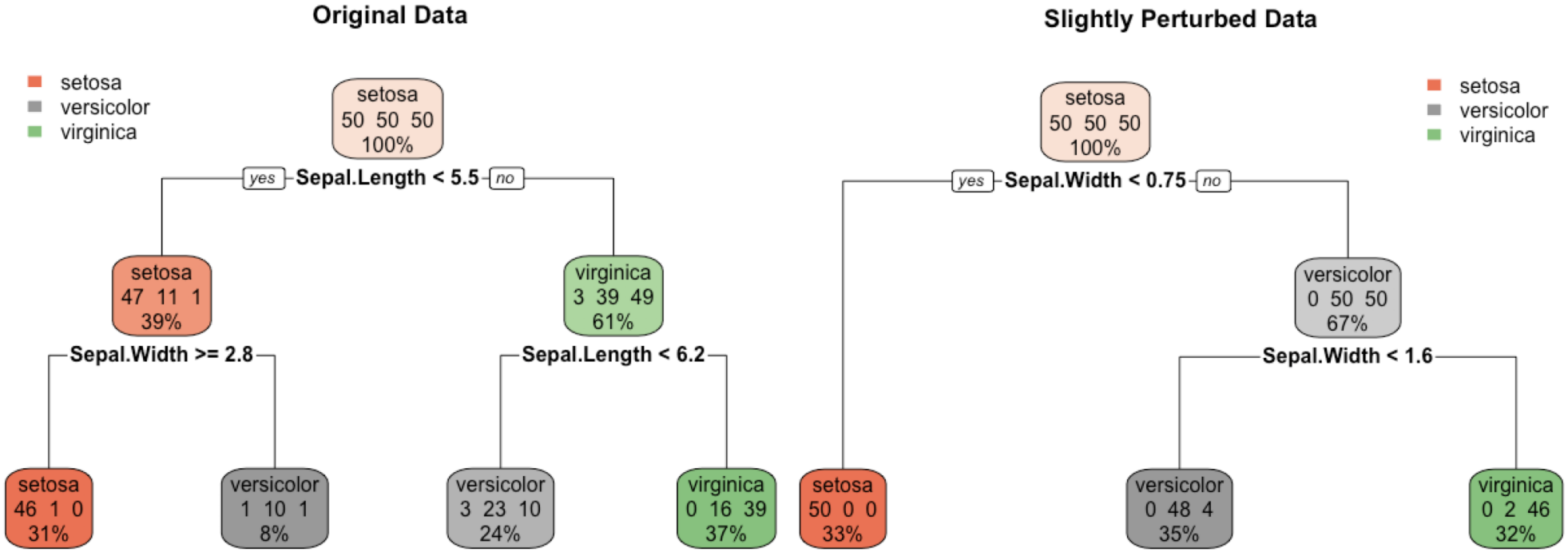
A decision tree is sensitive

to small data modifications

(a small change may give a very different tree)

Weakness

We perturb the data with ± 0.1 random numbers



Conclusion on decision tree

Hyperparameters:

- Splitting rule: impurity index
- Stopping rule: minsplit, minbucket, prune

Conclusion on decision tree

Hyperparameters:

- Splitting rule: impurity index
- Stopping rule: minsplit, minbucket, prune

Main advantage:

- Easy to visualise graphically and to interpret

Conclusion on decision tree

Hyperparameters:

- Splitting rule: impurity index
- Stopping rule: minsplit, minbucket, prune

Main advantage:

- Easy to visualise graphically and to interpret

Main disadvantage:

- Sensitive to training data (a small change in data may give a very different tree). This may be solved partially using a random forest, which is an ensemble of decision trees.

QUESTIONS ?

Random Forest

Random forest

A decision tree has an important weakness:

a small change in the training dataset

may give a very different tree

EXAMPLE

If we split randomly the training data

into two parts and fit a DT to both halves,

they may be very different

Random forest

Decision tree has high variance

How to reduce variance ?

Random forest

Average over a set of predictions

Probabilistic setting

1 tree: X and Y are random variables with $V(Y) = \sigma^2$

Probabilistic setting

1 tree: X and Y are random variables with $V(Y) = \sigma^2$

Random forest with M trees:

$$Y = \frac{1}{M}\sum_{i=1}^{M} Y_i \qquad \text{Corr}(Y_i, Y_j) = \varrho$$

Then

$$V(Y) = \frac{1}{M}\sigma^2 + \frac{M-1}{M}\varrho\sigma^2$$

If M is large and ϱ is small, the RF output has small variance

How to generate

many uncorrelated trees ?

Random forest

- Generate many training datasets

Random forest

- Generate many training datasets
- Build a decision tree for each training dataset

Random forest

- Generate many training datasets
- Build a decision tree for each training dataset
- Apply the majority rule for the prediction

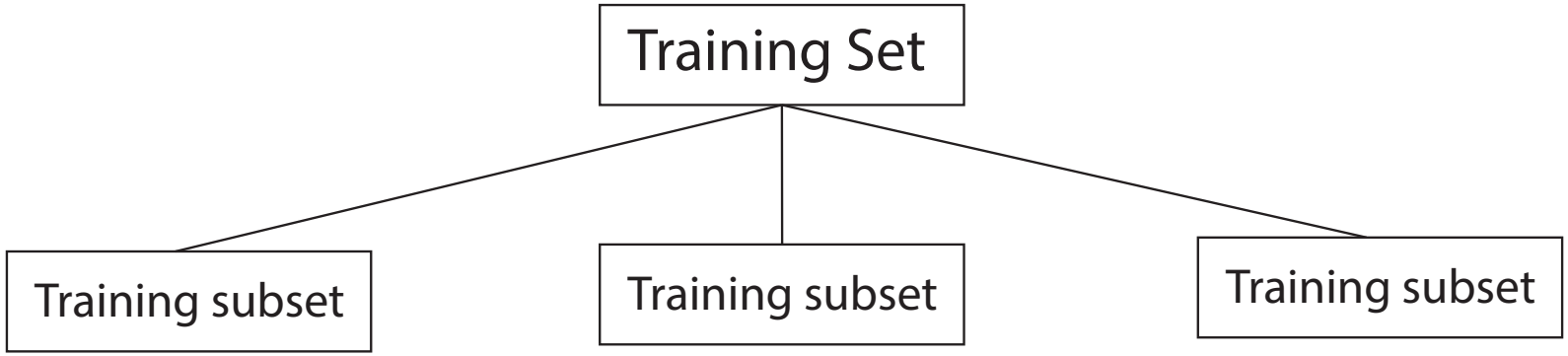
How to choose the splitting features ?

EXAMPLE

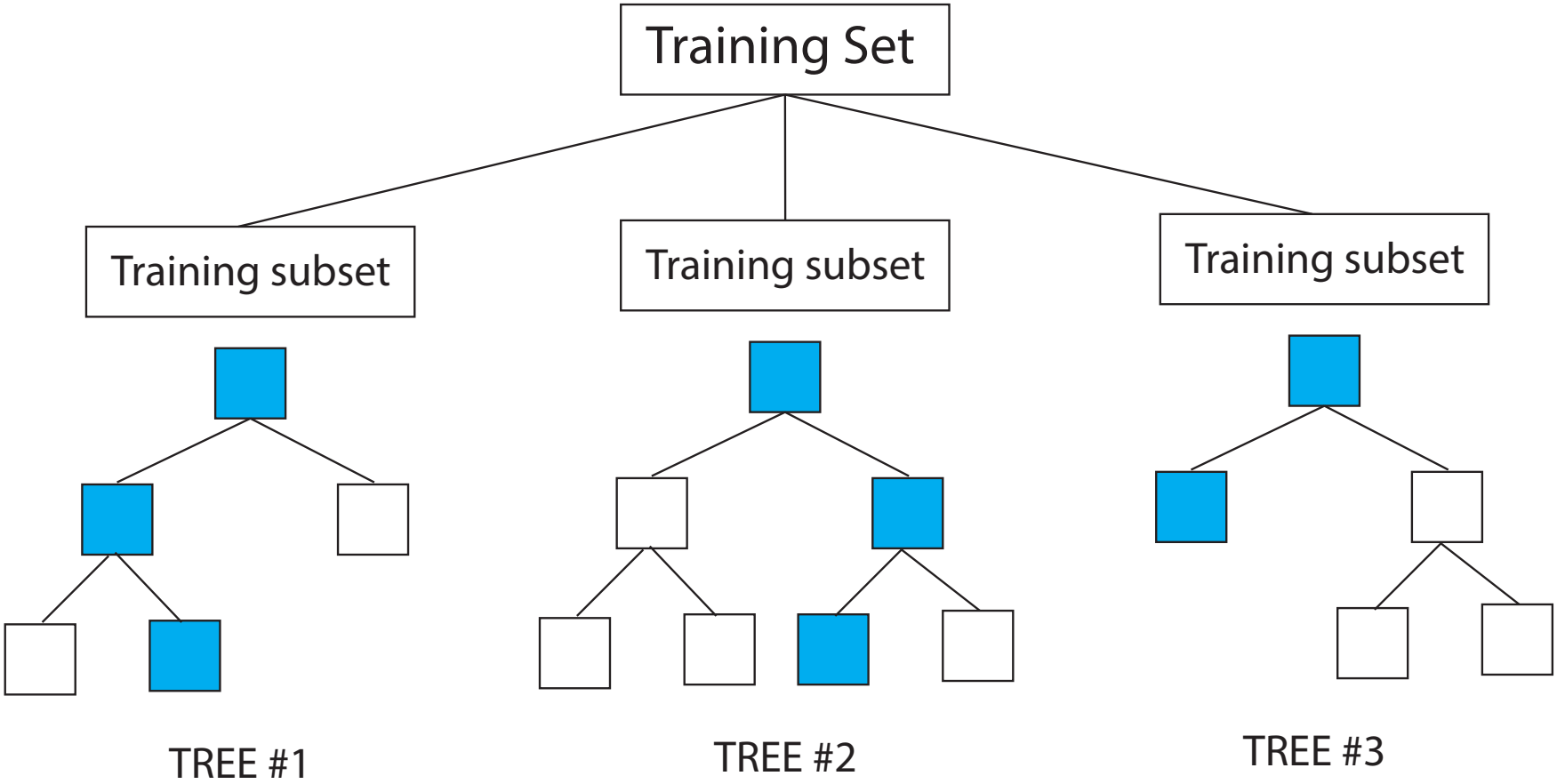
Random forest with 3 decision trees

Training Set

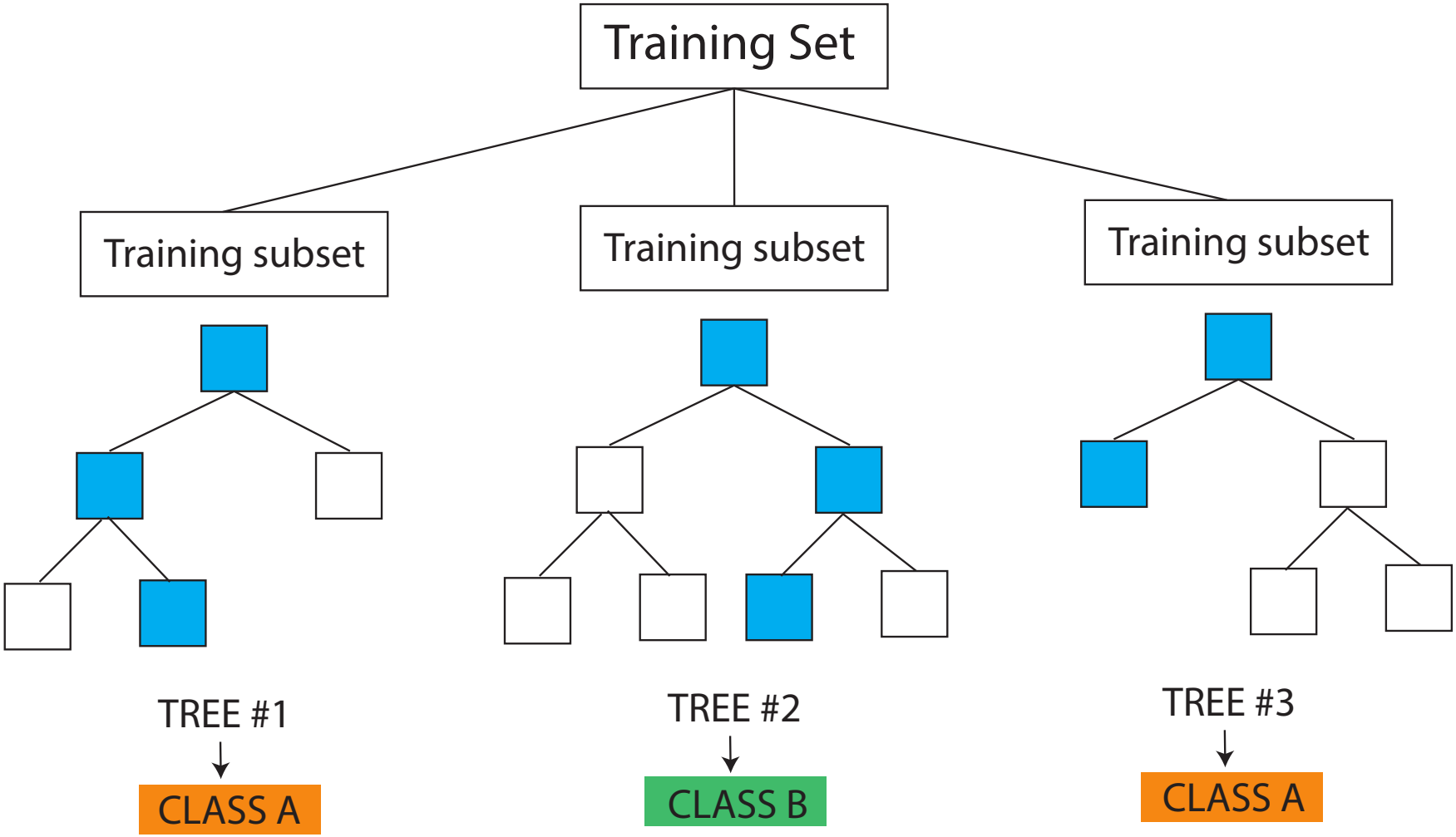
Random forest with 3 decision trees



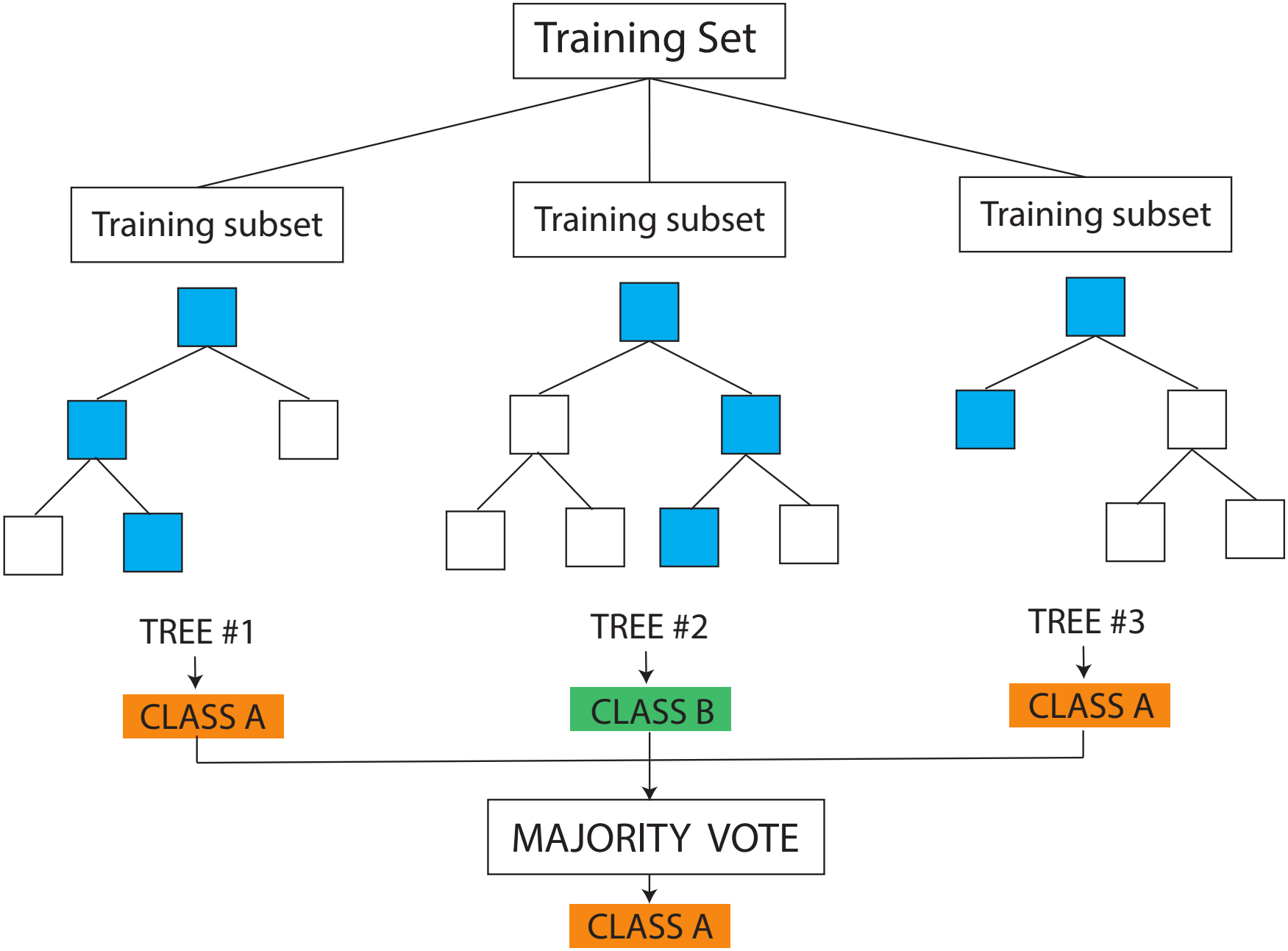
Random forest with 3 decision trees



Random forest with 3 decision trees



Random forest with 3 decision trees



Generate many training datasets

Algorithm: Draw B bootstrap samples of size $n < N$ with or without replacement ($B = 4, n = 4, N = 10$, no replacement):

Plant 1
Plant 2
Plant 3
Plant 4
Plant 5
Plant 6
Plant 7
Plant 8
Plant 9
Plant 10

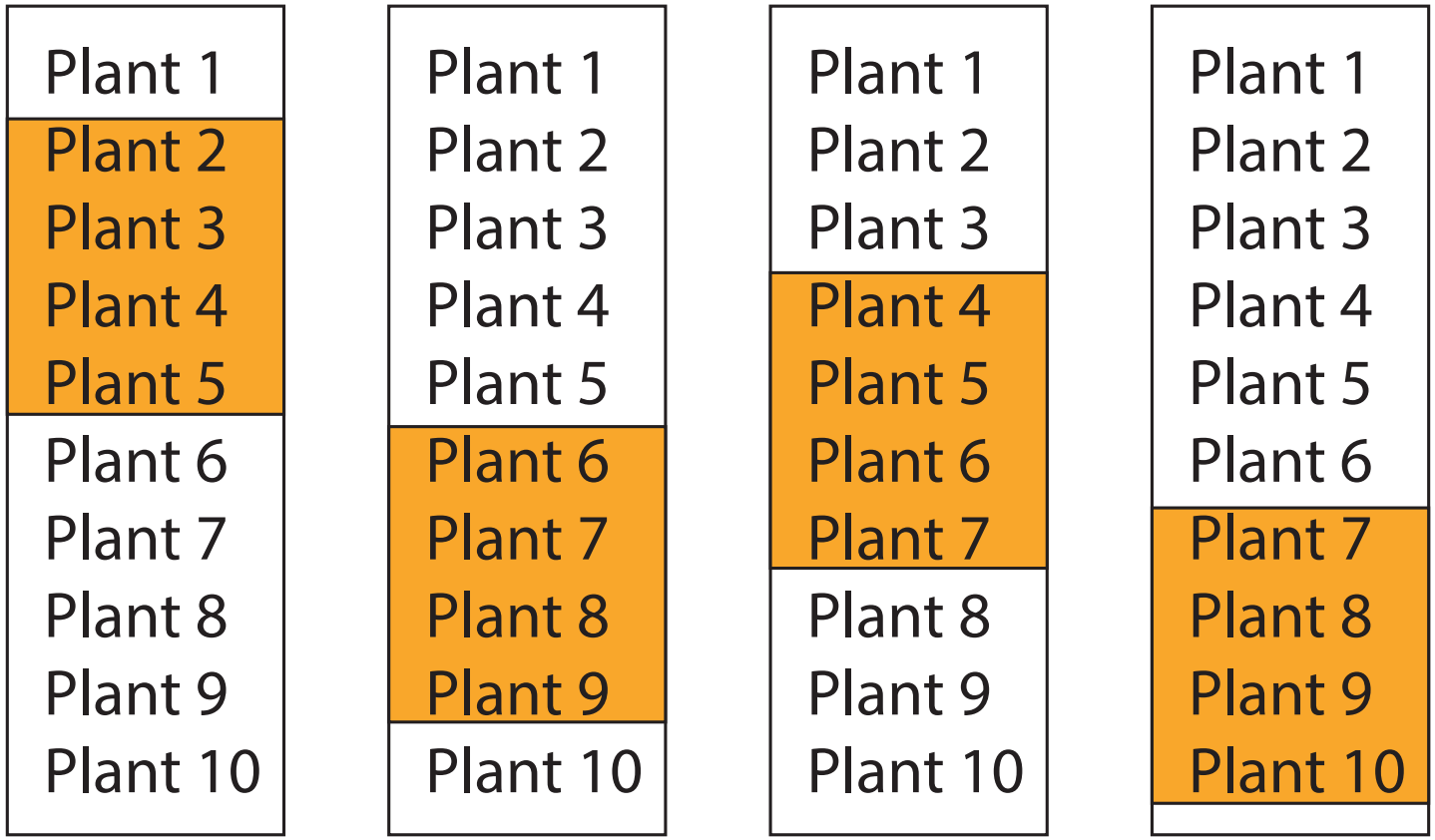
Generate many training datasets

Algorithm: Draw B bootstrap samples of size $n < N$ with or without replacement ($B = 4, n = 4, N = 10$, no replacement):

Plant 1	Plant 1	Plant 1	Plant 1
Plant 2	Plant 2	Plant 2	Plant 2
Plant 3	Plant 3	Plant 3	Plant 3
Plant 4	Plant 4	Plant 4	Plant 4
Plant 5	Plant 5	Plant 5	Plant 5
Plant 6	Plant 6	Plant 6	Plant 6
Plant 7	Plant 7	Plant 7	Plant 7
Plant 8	Plant 8	Plant 8	Plant 8
Plant 9	Plant 9	Plant 9	Plant 9
Plant 10	Plant 10	Plant 10	Plant 10

Generate many training datasets

Algorithm: Draw B bootstrap samples of size $n < N$ with or without replacement ($B = 4, n = 4, N = 10$, no replacement):



Build a decision tree for each training dataset

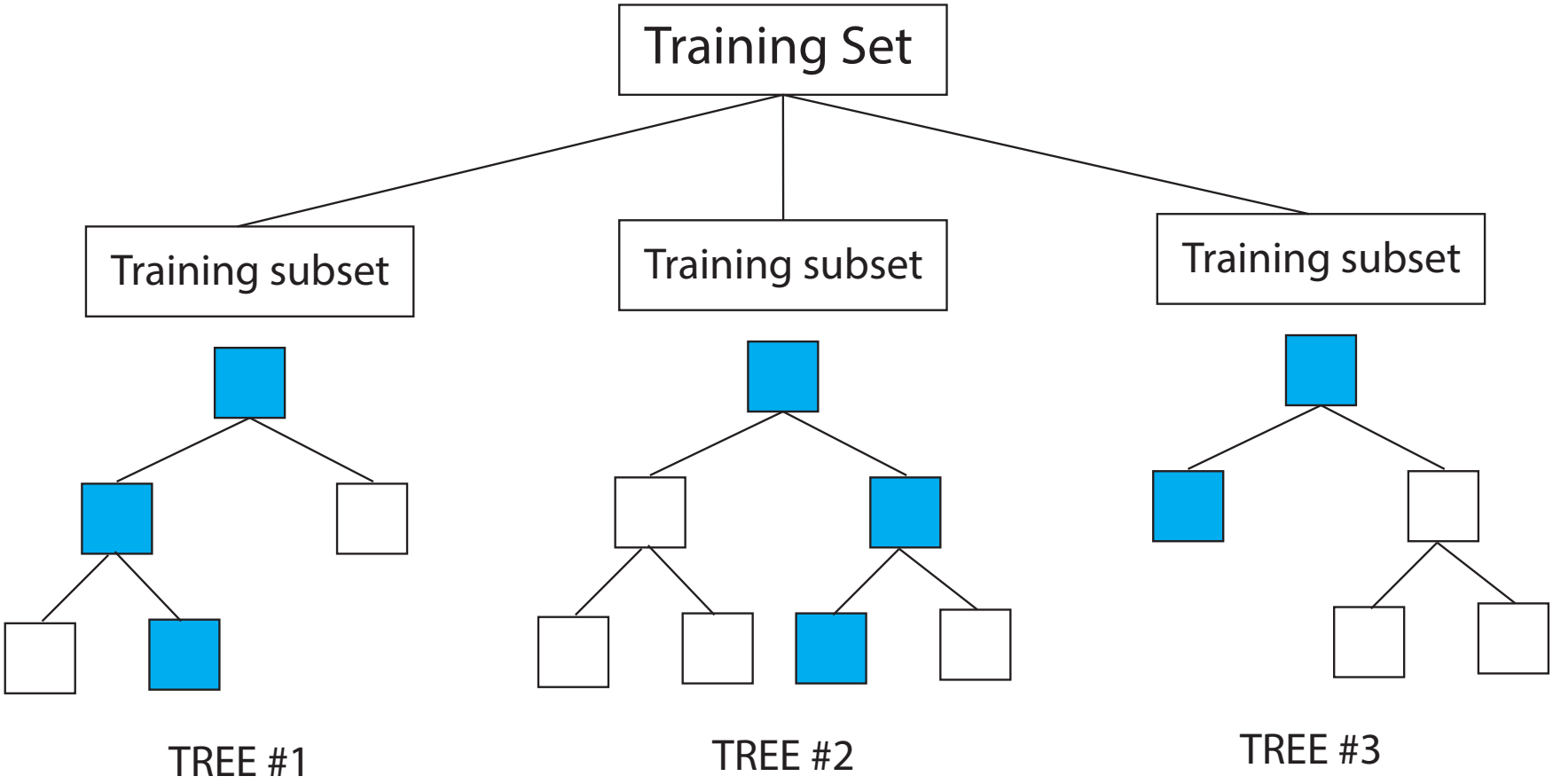
For each bootstrap sample:

1. Select randomly m variables from the p variables.
   Important to obtain **decorrelated** trees
   Typically $m = \sqrt{p}$ ($p = 4$ so $m = 2$ for iris)

2. Split the node into two daughter nodes based on the m variables (the value m is same at each node)

3. Continue until no more split is possible

Build a decision tree for each training dataset



What is the importance of each variable ?

PROBLEM

What is the importance of each variable ?

It is not clear in a random forest

which variables are important in predicting

the species a given plant belongs to

What is the importance of each variable ?

SOLUTION

What is the importance of each variable ?

The **importance** of a given predictor

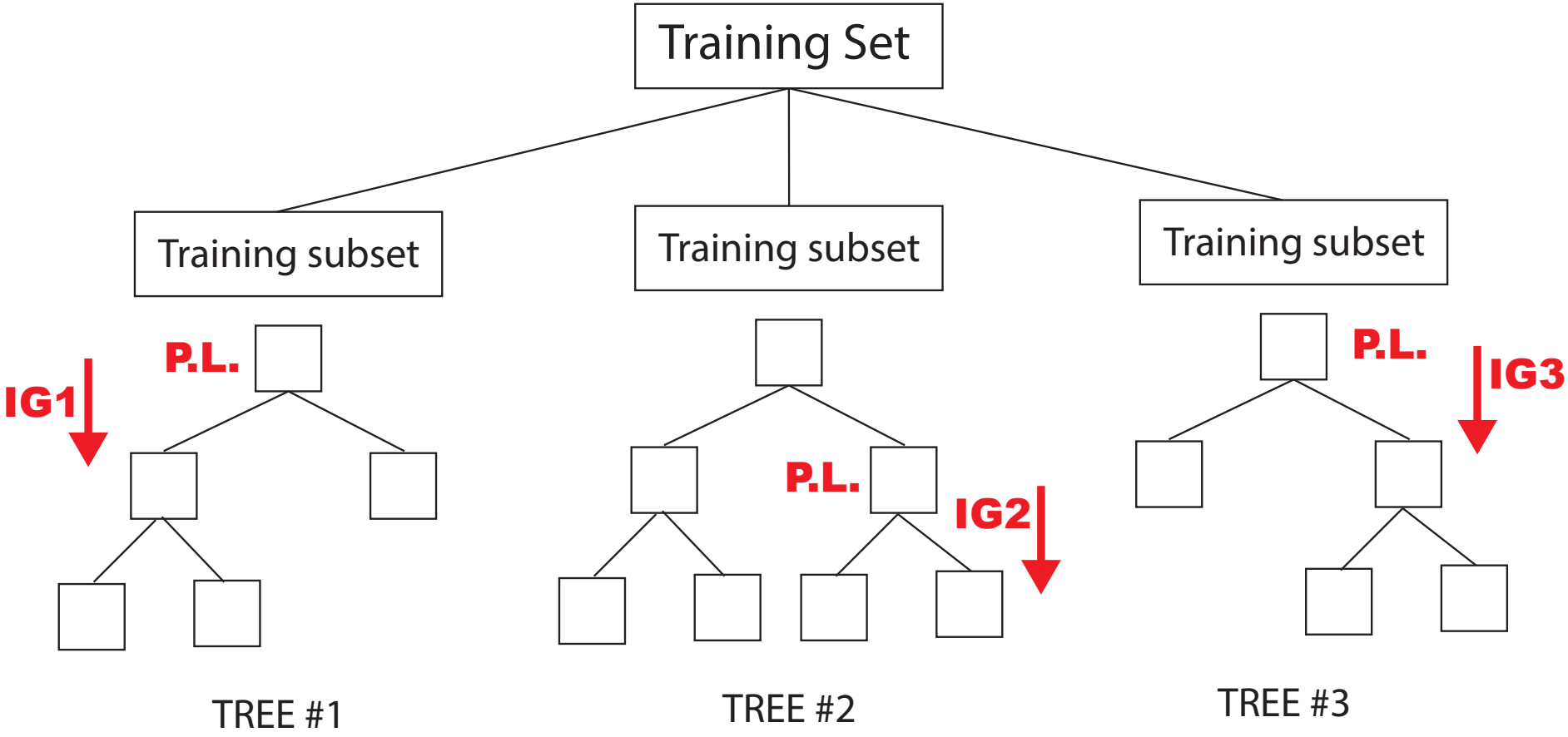
may be computed by adding up

the information gain increases averaged

over all splits in the random forest involving

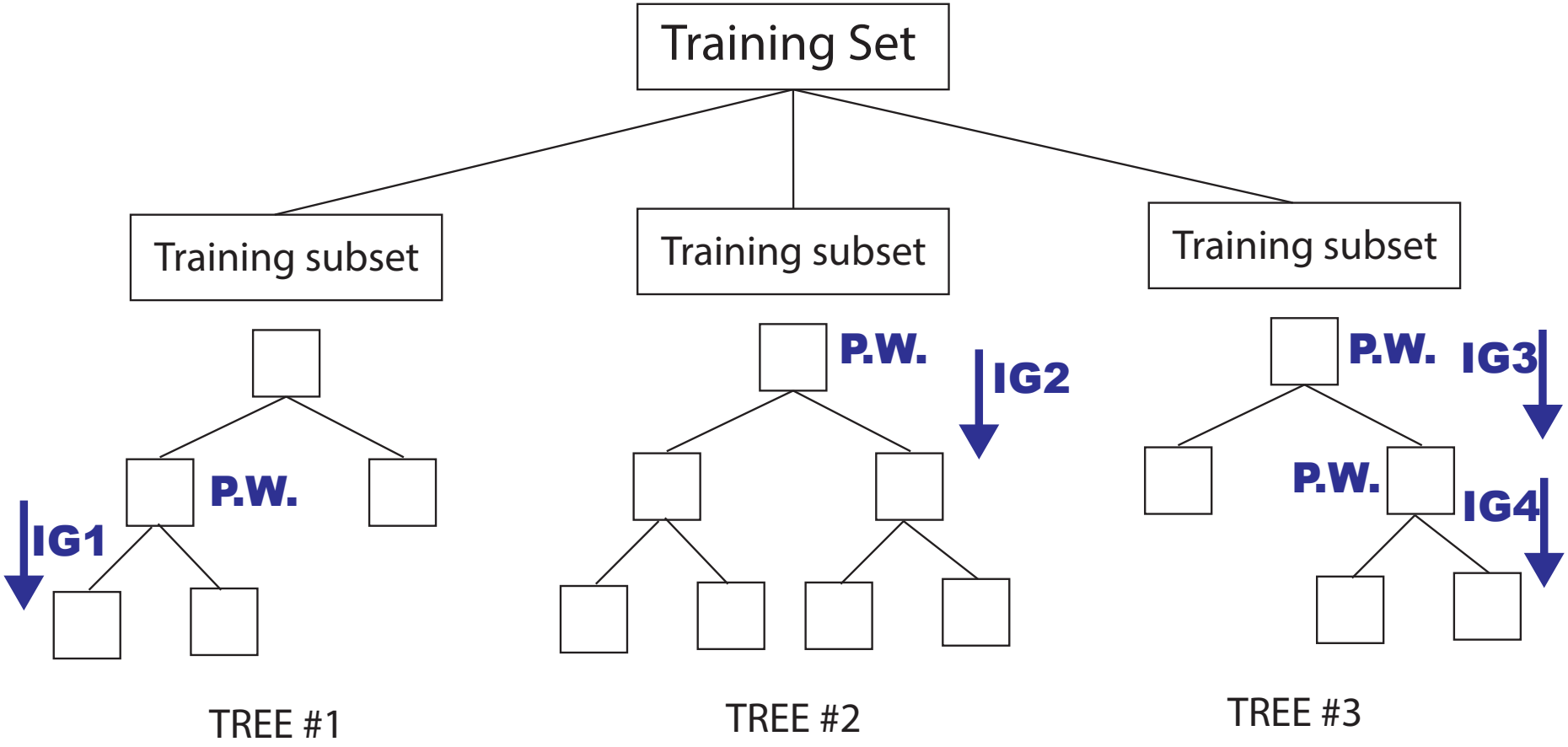
the predictor in question

Build a decision tree for each training dataset



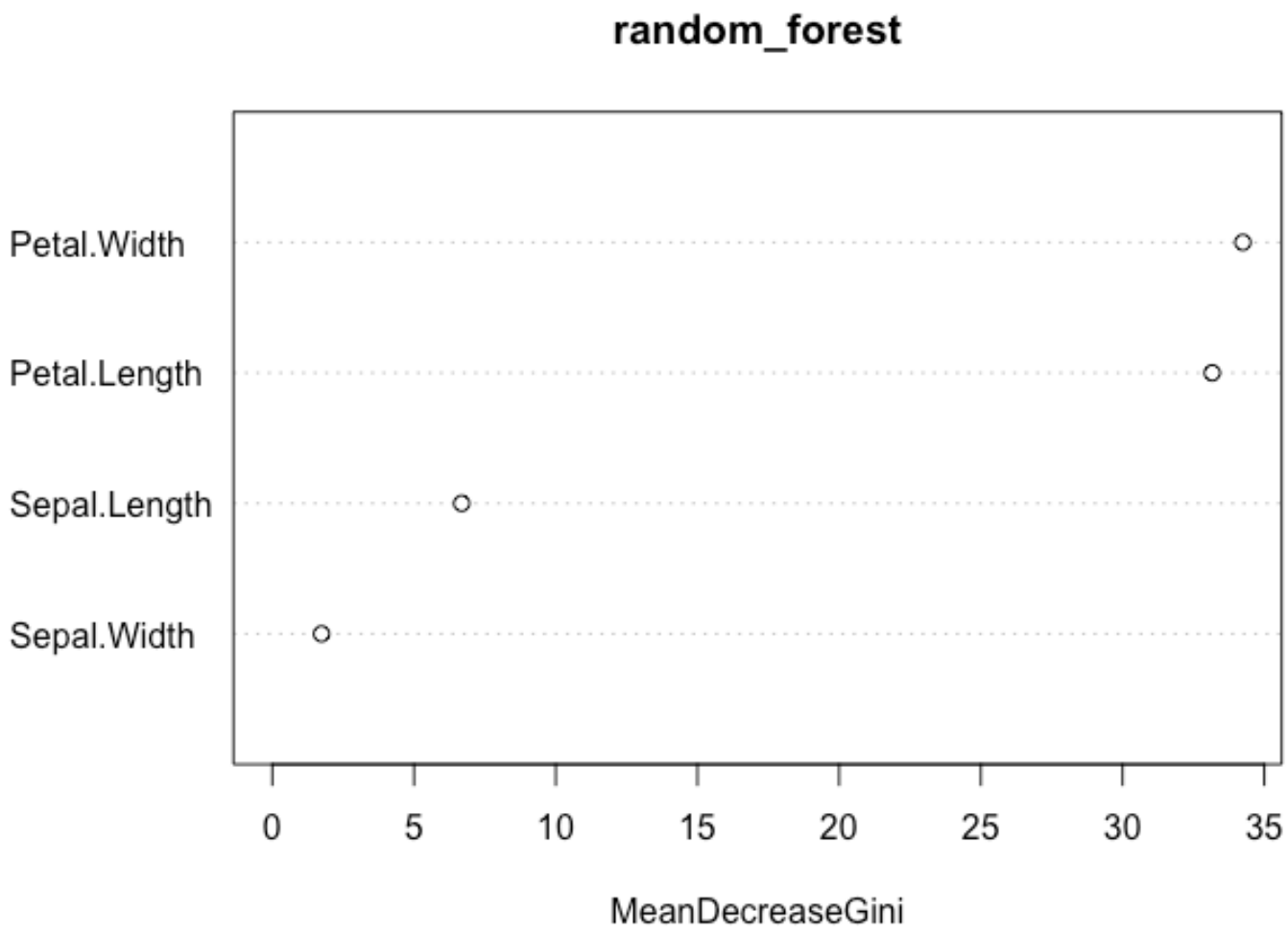
Importance=(IG1+IG2+IG3)/3

Build a decision tree for each training dataset



Importance=(IG1+IG2+IG3+IG4)/3

What is the importance of each variable ?



Conclusion on random forest

New hyperparameters compared to decision tree:

- Number of trees, size of train subset (with or without replacement), number of variables used at each split

Conclusion on random forest

New hyperparameters compared to decision tree:

- Number of trees, size of train subset (with or without replacement), number of variables used at each split

Main advantage compared to decision tree:

- Robust to training data change

Conclusion on random forest

New hyperparameters compared to decision tree:

- Number of trees, size of train subset (with or without replacement), number of variables used at each split

Main advantage compared to decision tree:

- Robust to training data change

Main disadvantage compared to decision tree:

- Not easy to visualise graphically and to interpret

QUESTIONS ?

Applications

Application 1

Predict the risk of

kidney transplantation rejection

Ref: Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation, Torgyn Shaikhina, DaveLowe, Sunil Dagade, David Briggs, Robert Higgins, Natasha Khovanov, Biomedical Signal Processing and Control (2017)

Application 1

Some kidney diseases require transplantation

to save the life of the patient

Application 1

But the kidney may be rejected

Application 1

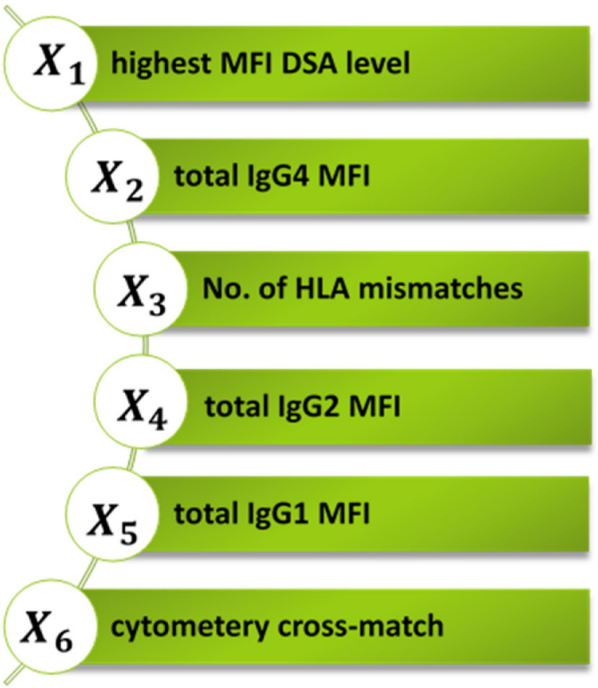
What are the risk factors associated with rejection ?

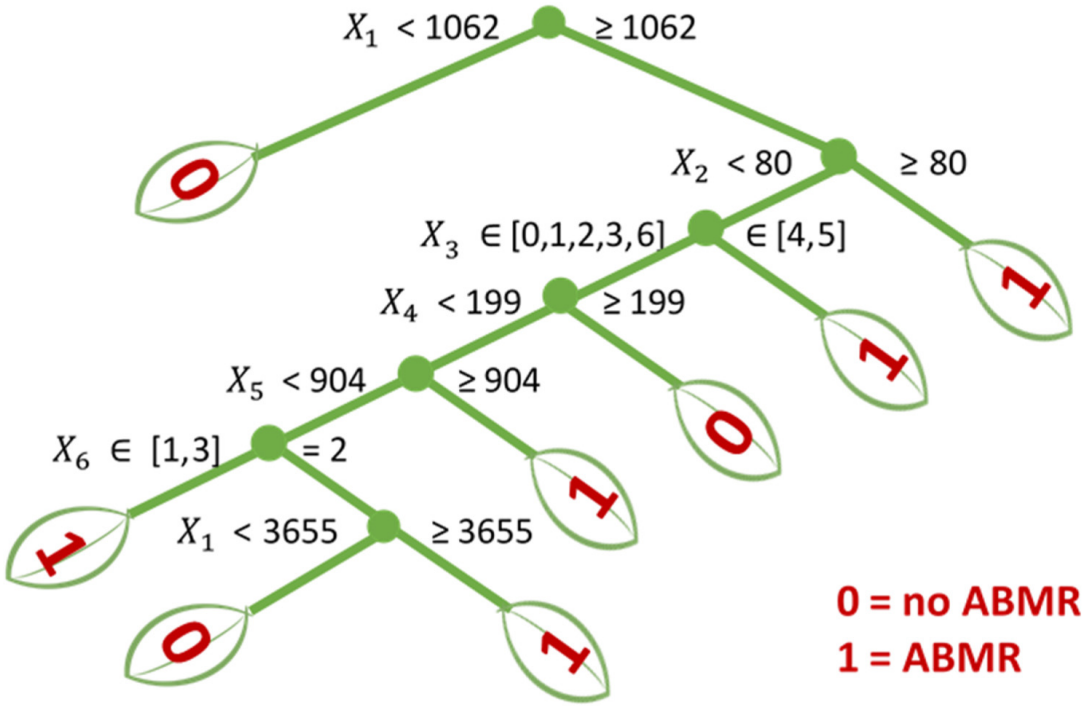
Application 1

They used 6 predictors in a decision tree

(data: 80 patients)

Application 1





0 = No Kidney Rejection

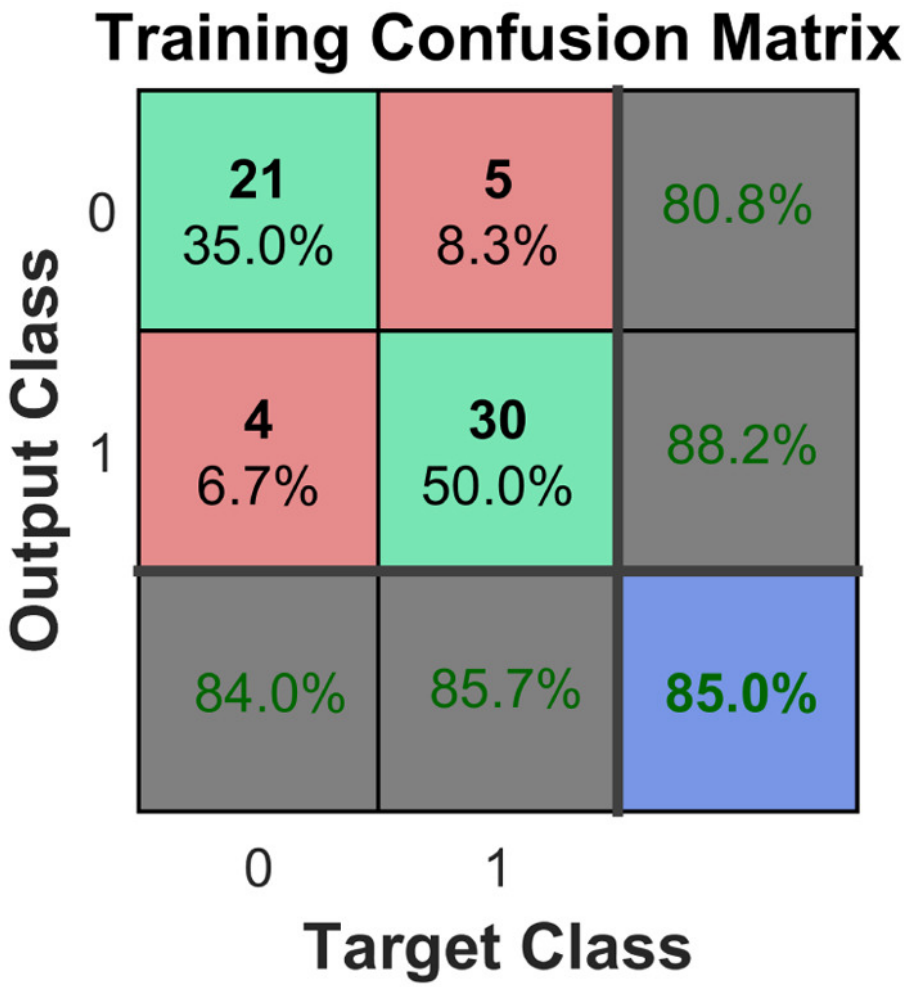
1 = Kidney Rejection

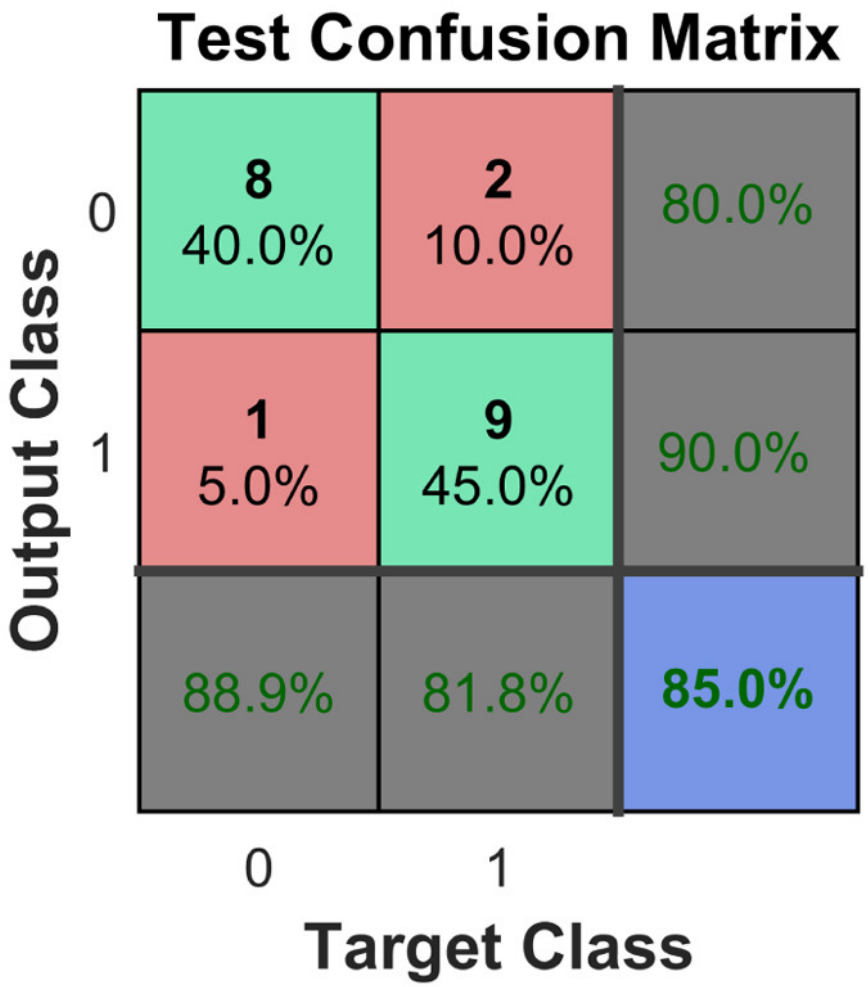
Impurity index: Gini

minsplit=10, minbucket=1

Application 1

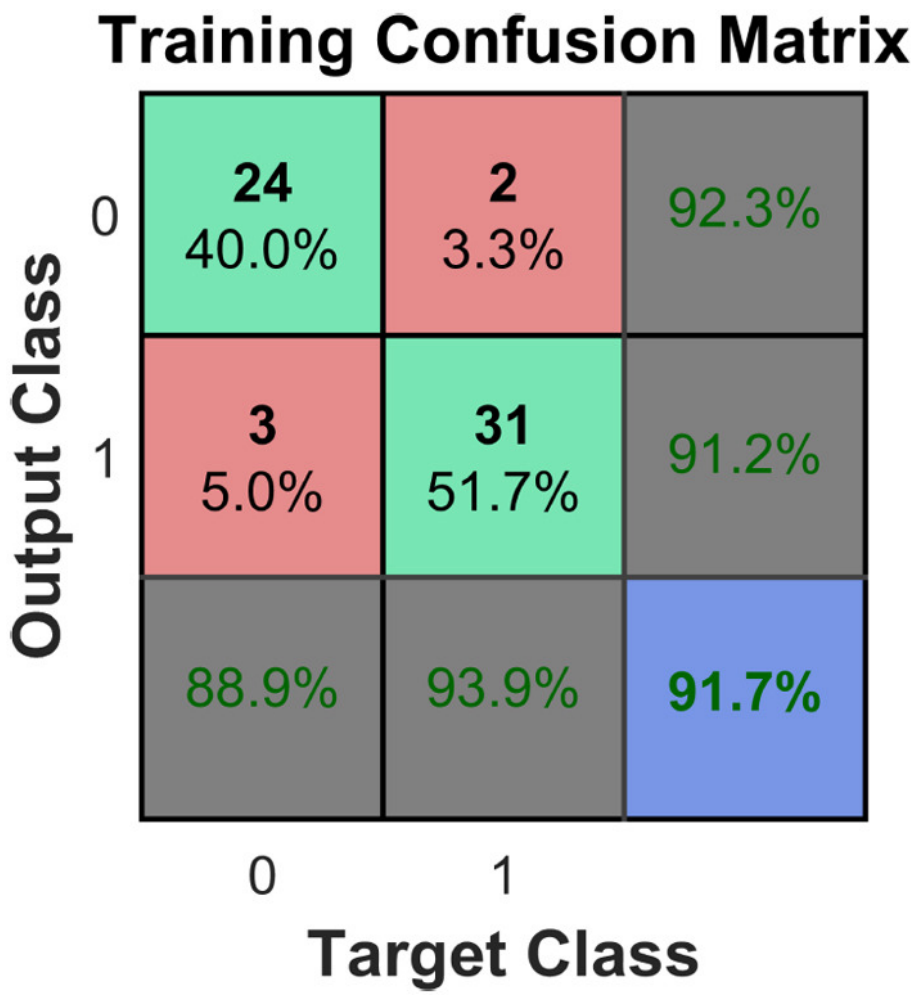
Decision Tree:

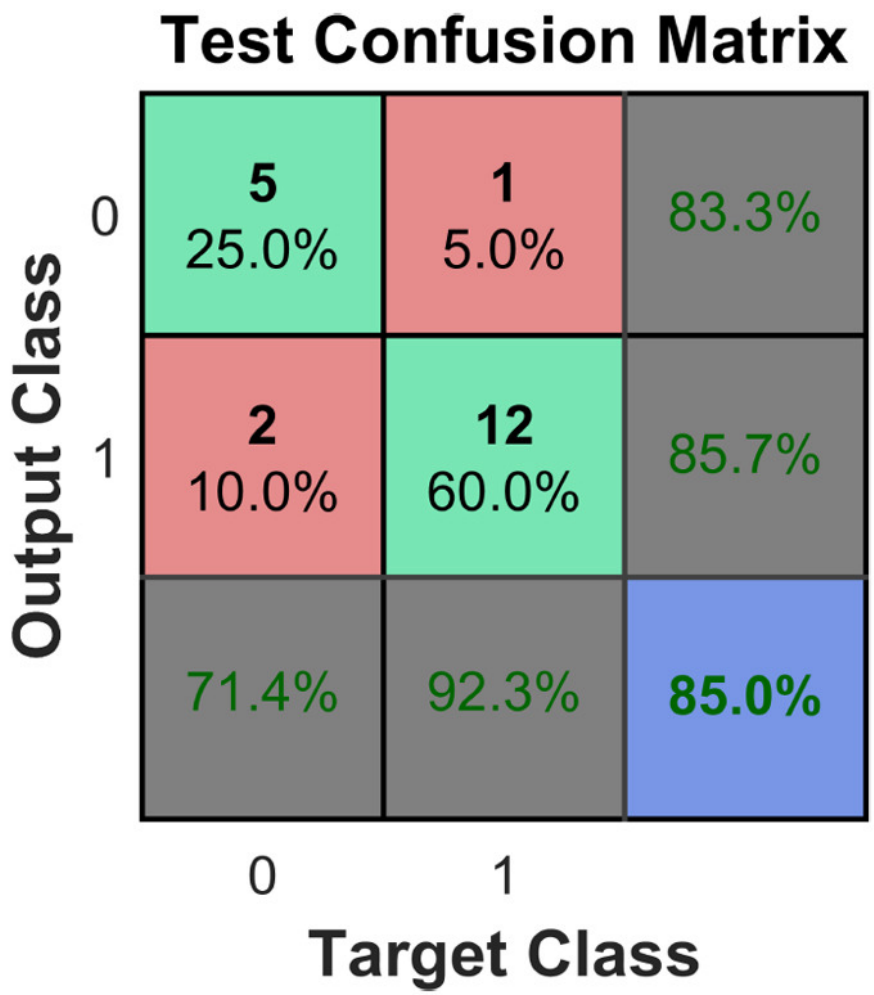




Application 1

Random Forest (600 trees):





Application 2

Predict the risk of

type 2 diabetes

Ref: Type 2 Diabetes Mellitus Screening and Risk Factors Using Decision Tree: Results of Data Mining, Shafi Habibi, Maryam Ahmadi, and Somayeh Alizadeh, Glob J Health Sci. (2015)

Application 2

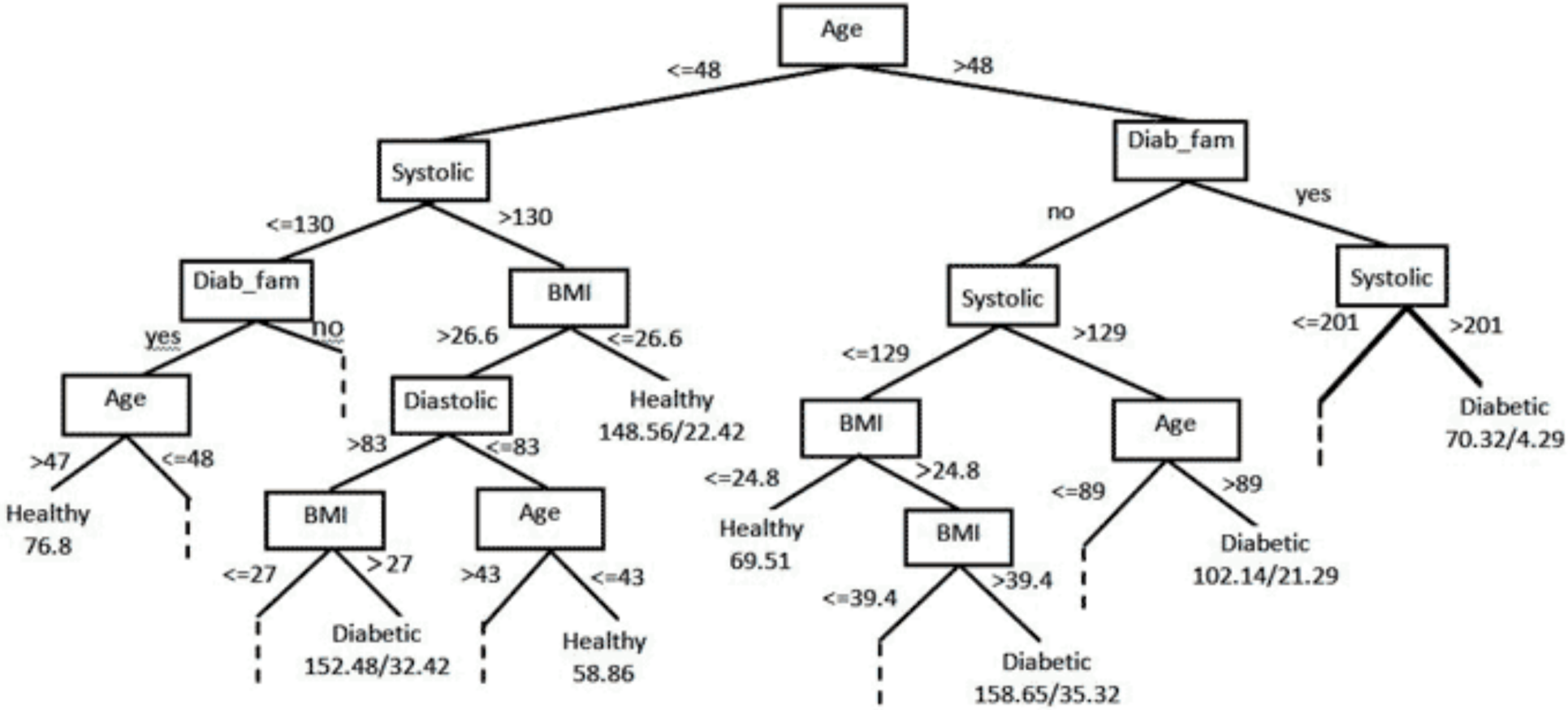
DATA

20’000 patient records

(age, gender, BMI, etc)

are classified as diabetic or healthy

Application 2



Rem: Gender was used but does not appear in the tree

Application 2

PREDICTION

Table 2. Confusion matrix of the decision tree model

	Classes	Diabetic	Healthy
TRUE	Healthy	253	21221
	Diabetic	641	283

Accuracy = 98 %

Application 2

PREDICTION

Table 2. Confusion matrix of the decision tree model

	Classes	Diabetic	Healthy
TRUE	Healthy	253	21221
	Diabetic	641	283

Accuracy = 98 %

You may want to minimize FNR

QUESTIONS ?

BONUS

Bonus 1: The best learning algorithm

Which is

the best learning algorithm ?

Bonus 1: The best learning algorithm

No universal machine learning algorithm:

- No free lunch theorem: In a nutshell, it states that there is no learning algorithm that works best for all problems.

Bonus 1: The best learning algorithm

No universal machine learning algorithm:

- No free lunch theorem: In a nutshell, it states that there is no learning algorithm that works best for all problems.
- As a consequence, one should try several reasonable learning algorithms based on the nature of the problem, type and amount of data, error function, etc.

Bonus 1: The best learning algorithm

No universal machine learning algorithm:

- No free lunch theorem: In a nutshell, it states that there is no learning algorithm that works best for all problems.
- As a consequence, one should try several reasonable learning algorithms based on the nature of the problem, type and amount of data, error function, etc.
- And use the validation set accuracy as a performance measure to select the best one.

Bonus 1: The best learning algorithm

No universal machine learning algorithm:

- No free lunch theorem: In a nutshell, it states that there is no learning algorithm that works best for all problems.
- As a consequence, one should try several reasonable learning algorithms based on the nature of the problem, type and amount of data, error function, etc.
- And use the validation set accuracy as a performance measure to select the best one.

Example: Let us put aside the interpretability and robustness. Then, the accuracy for the iris data:

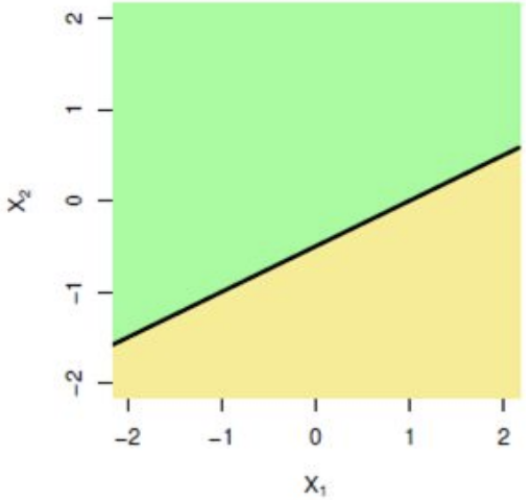
NN= 0.94, Tree= 0.94, RF= 0.91 –> Best: NN=Tree

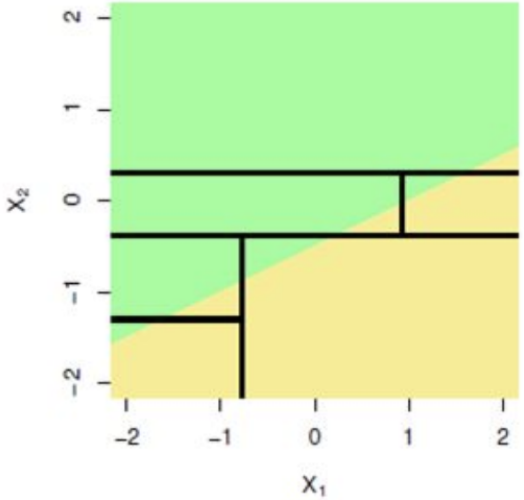
Bonus 2: Decision Tree versus Linear Model

Decision Trees vs. Linear Models – Depends on underlying data structure

1. **True linear boundary:**

Linear model fits perfectly



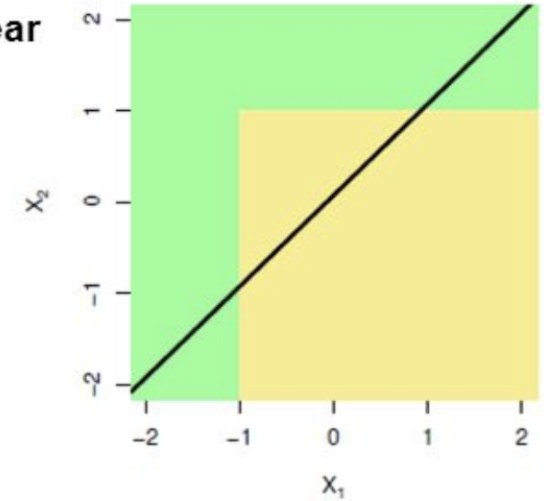


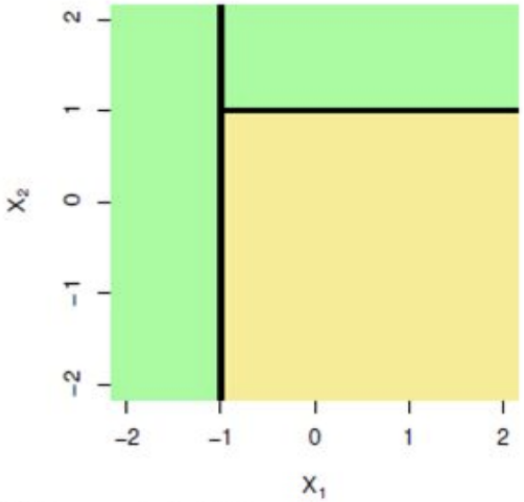
3. **True linear boundary:**

Flexible decision tree can approximate

2. **True nonlinear boundary:**

Linear model fits poorly





4. **True nonlinear boundary:**

Simple decision tree fits perfectly

Source: "An Introduction to Statistical Learning with Applications in R", James et al., 2013

Bonus 3: Decision Tree for Regression

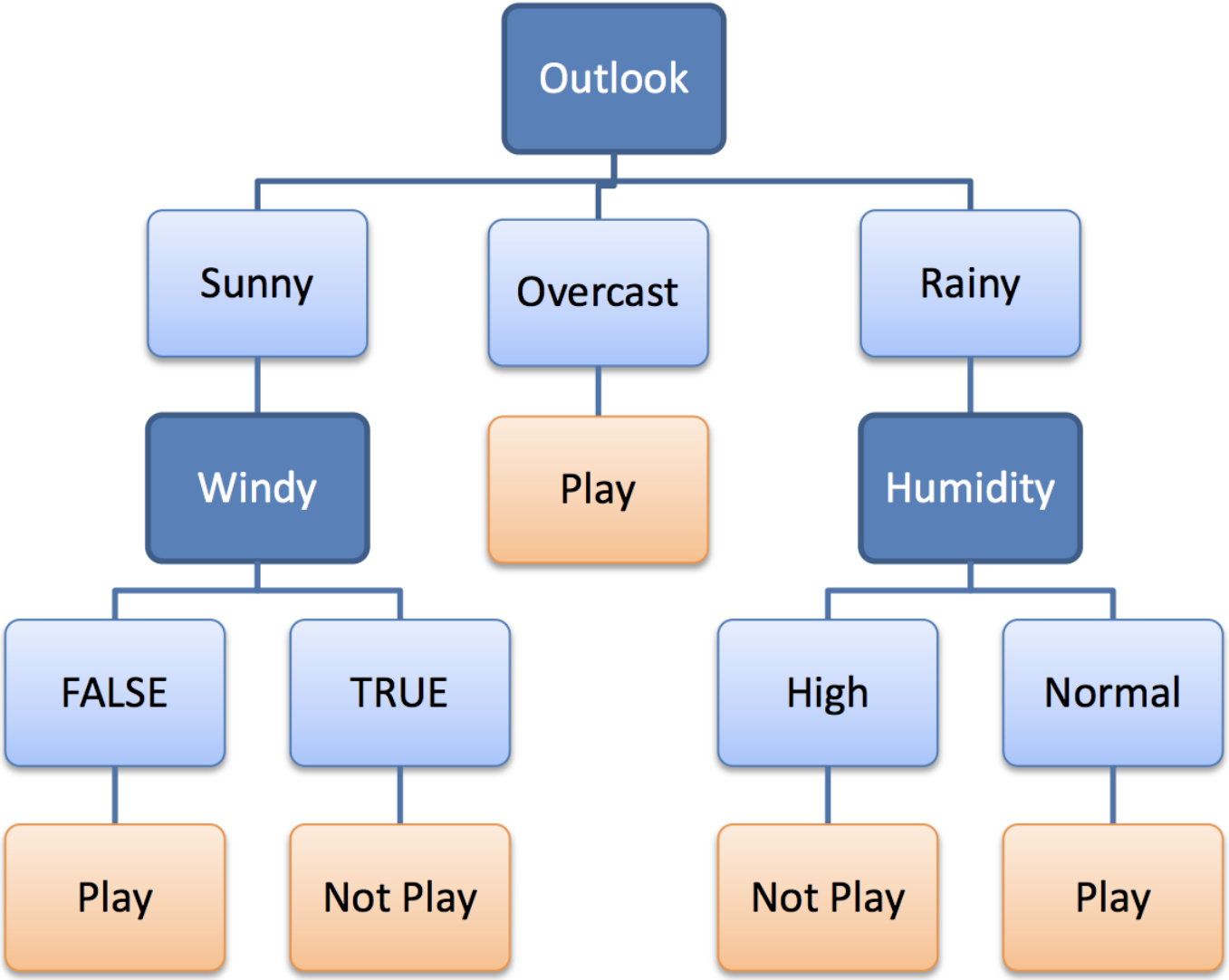
Decision Tree for Classification:

Predictors | Target

Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Bonus 3: Decision Tree for Regression

Decision Tree for Classification:



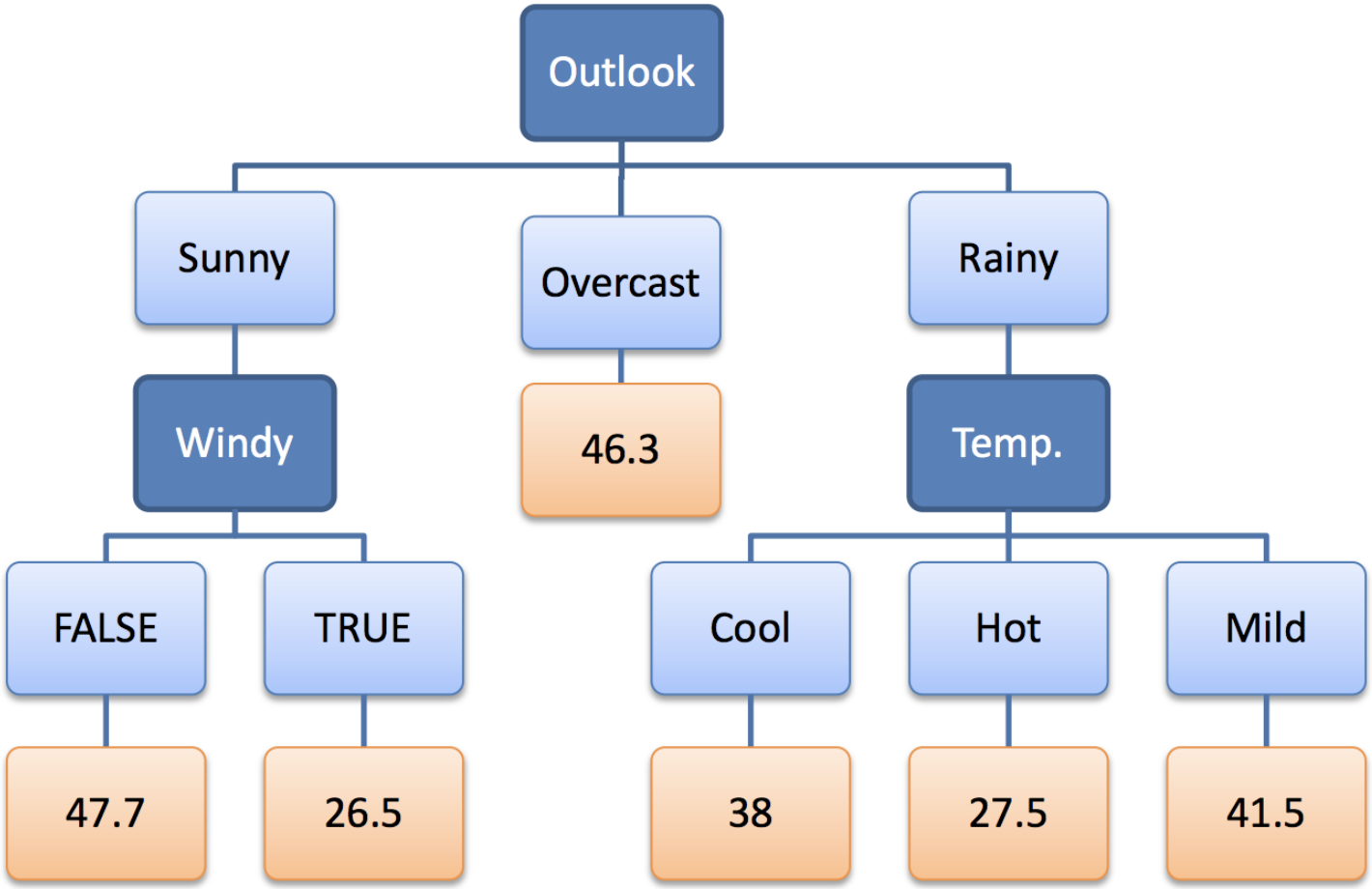
Bonus 3: Decision Tree for Regression

Decision Tree for Regression:

Predictors				Target
Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	25
Rainy	Hot	High	True	30
Overcast	Hot	High	False	46
Sunny	Mild	High	False	45
Sunny	Cool	Normal	False	52
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	35
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	46
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	52
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

Bonus 3: Decision Tree for Regression

Decision Tree for Regression:



Bonus 3: Decision Tree for Regression

Two modifications to go

from DT classification to DT regression

Modification 1

Information Gain

becomes

Standard Deviation Reduction

Modification 2

Most commonly occurring class in each leaf

becomes

Mean response of the training output values

Bonus 3: Decision Tree for Regression

1. Standard Deviation Reduction:

		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

		Hours Played (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
SDR=0.17		

		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
SDR=0.28		

		Hours Played (StDev)
Windy	False	7.87
	True	10.59
SDR=0.29		

Bonus 3: Decision Tree for Regression

1. Standard Deviation Reduction:

		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

		Hours Played (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
SDR=0.17		

		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
SDR=0.28		

		Hours Played (StDev)
Windy	False	7.87
	True	10.59
SDR=0.29		

Bonus 3: Decision Tree for Regression

1. Standard Deviation Reduction:

Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48

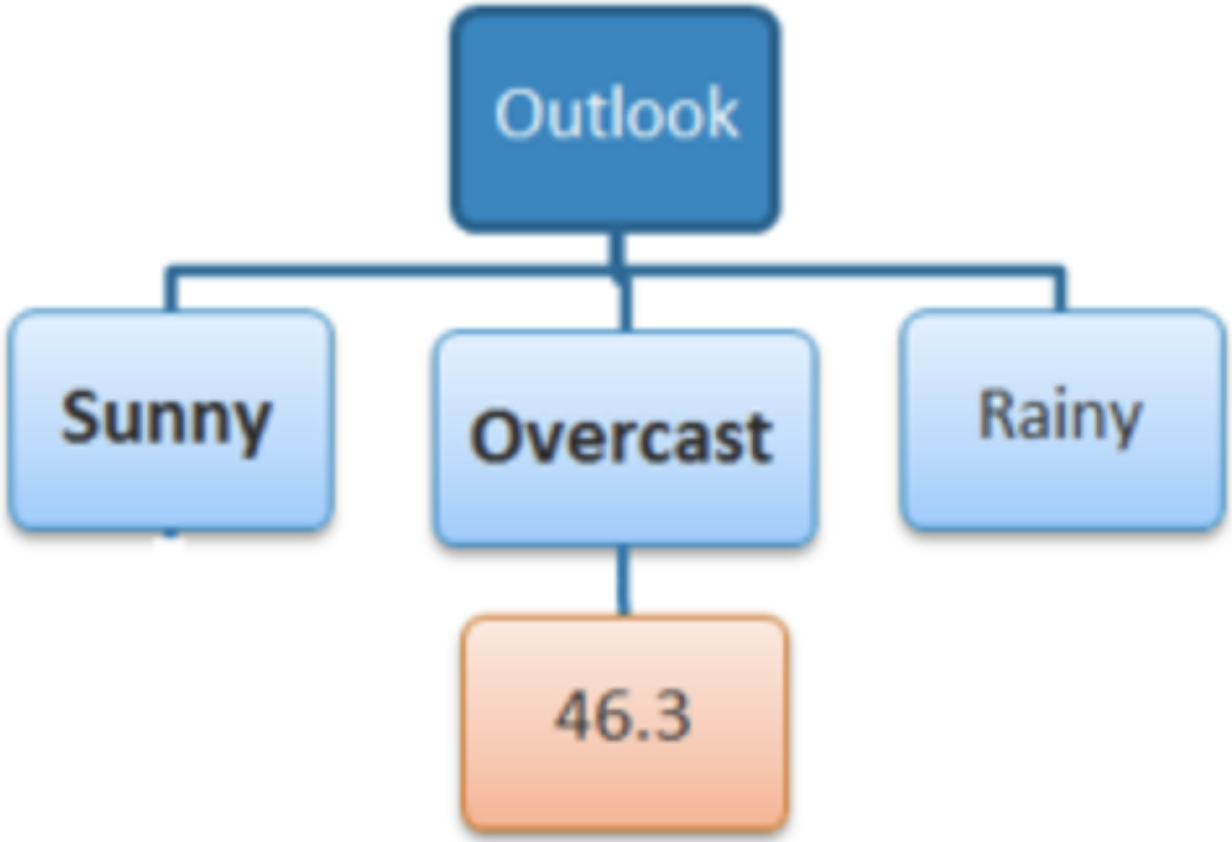
Bonus 3: Decision Tree for Regression

1. Standard Deviation Reduction:

Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48

Bonus 3: Decision Tree for Regression

2. Mean response of the training output values:



Bonus 4: Multicollinearity

Multicollinearity

occurs when two or more predictor variables

are intercorrelated

Bonus 4: Multicollinearity

Is multicollinearity a problem ?

Bonus 4: Multicollinearity

Yes for interpretation

(to know which variables are important)

(or to know the effects of individual predictors)

No really for prediction

(more variables may give better accuracy)

THANK YOU

FOR YOUR ATTENTION