Scikit-learn and Tour of Classifiers

Decision Tree learning

Decision Tree learning

- Based on recursive binary **partition** of the **feature space**
- **Decide** class based on a series of “**questions**”
- Partition based on **axis-oriented** hyperplanes (“zero weights only bias”)

- The **number of partitions** is called **tree depth** and determines model complexity

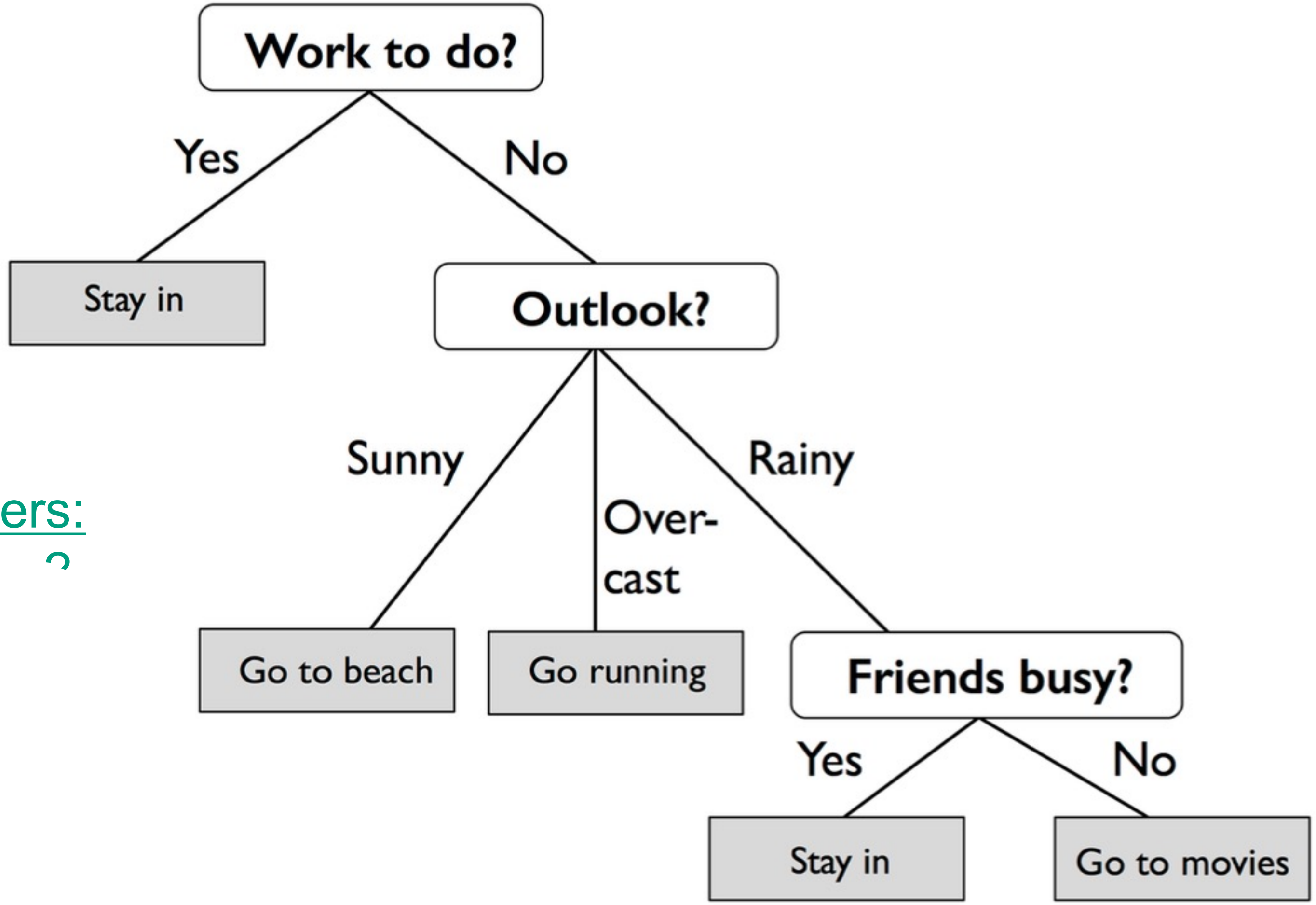
- Very popular due to easy **interpretability** of the **resulting decision function**

Decision trees – Illustrative example

Question

Decision

Outcome



Question for real numbers:

Is Petal len

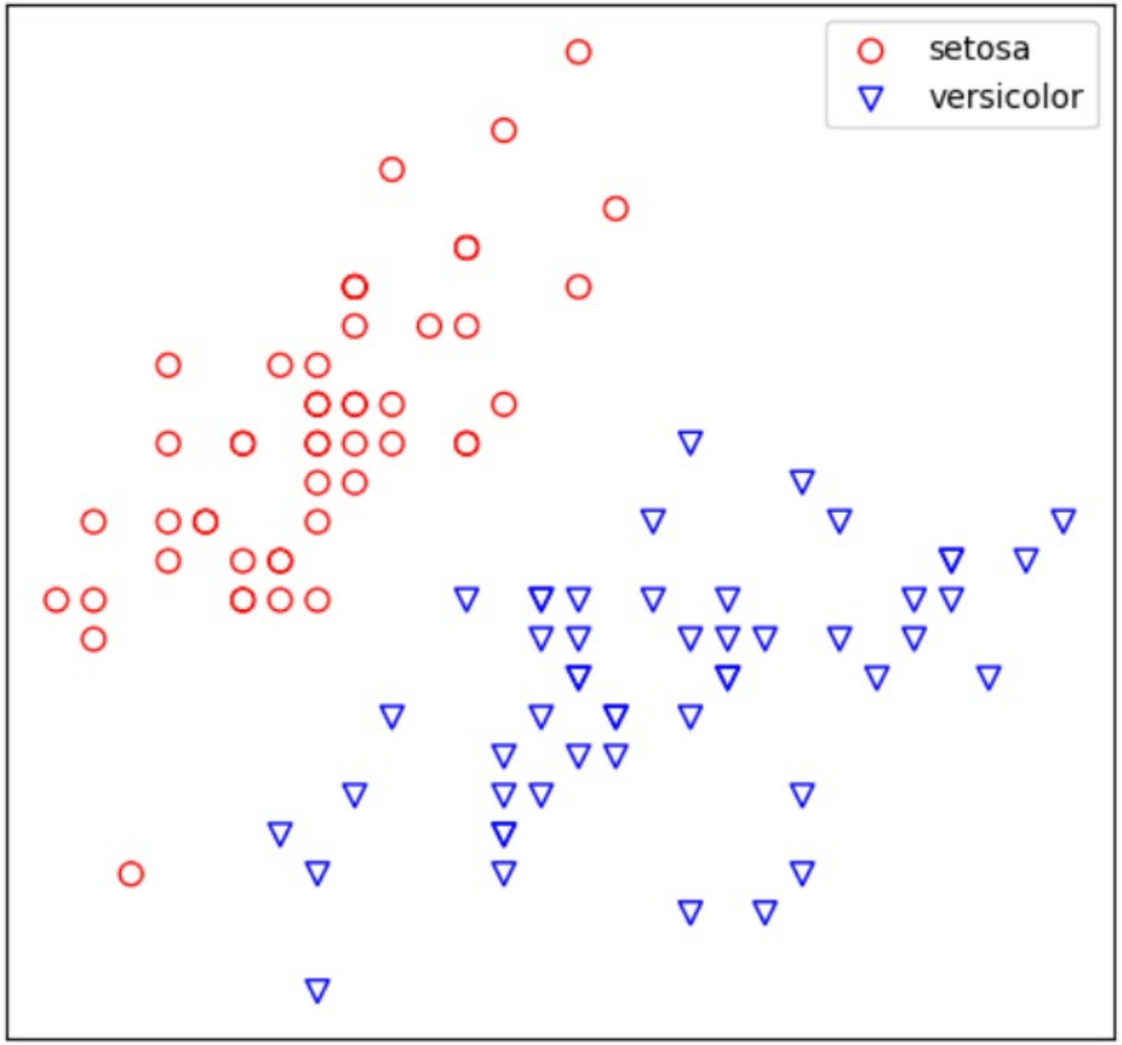
Tree depth = 3

Figure from Python Machine Learning (Raschka & Mirjalili)

Decision trees – Learning goal

- **Learn a series of questions** that will lead to a decision
- Which question? That is, which split do we choose?
  - Choose the split with the largest **information gain (IG)**
- We can repeat adding splits until **leaf nodes** are **pure** (all samples associated with the leaf node belong to the same class)
- Trees with only pure leaf nodes are usually too deep (overfitting!)
- **Pruning** a tree means removing branches or setting a limit for the maximum depth

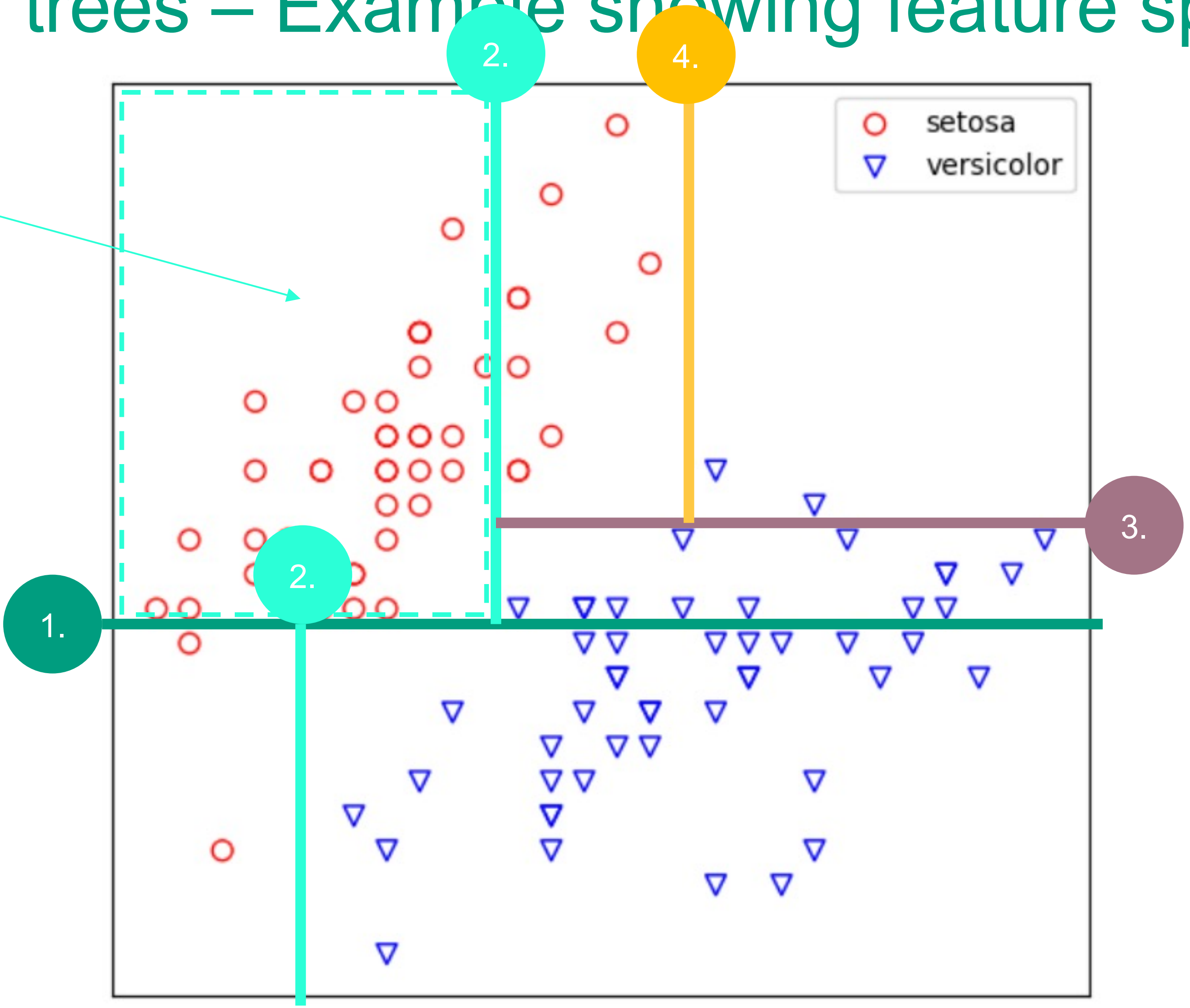
Decision trees – Example showing feature space

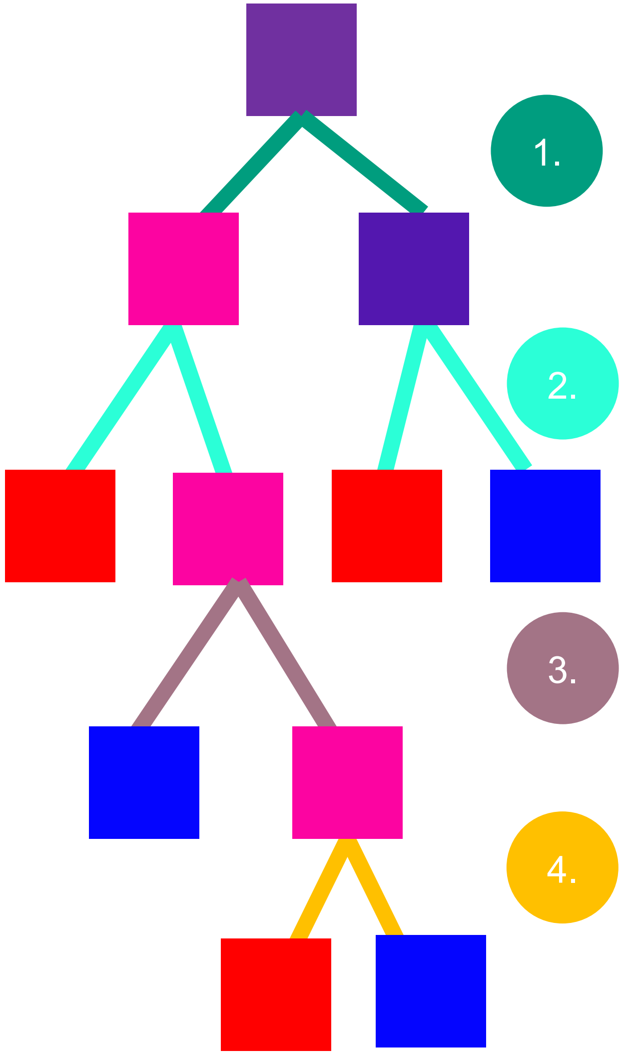


Decision trees – Example showing feature space

Pure leaf node
(here: “region”)

Only contains
“setosa” samples





Tree depth = 4

Decision trees – Maximize information gain (IG)

- **Information gain (IG)** of splitting along feature “f” with threshold “t”, can be measured by

$$IG\big(D_p,(f,t)\big) = I\big(D_p\big) - \sum_{j=1}^{C=2} \frac{N_j(t)}{N_p} I(D_j(t))$$

- $D_n \subseteq D = \{\boldsymbol{X}, y\}$ is the **subset** of the **data** at **node** *n* (*“p”: parent, “j”: j-th child of parent p)*
- N_n is the number of samples in D_n
- Typically, C=2 (**number of child nodes**); yields efficient data structures (**binary trees**)
- Finally, *I(D)* is the **impurity** of the data
- Choose split along the **feature with** the **maximum information gain** $\underset{\{f,t\}}{\operatorname{argmax}}\ IG(D_p, f, t)$

Decision trees – Impurity measures (Classification error)

- **Impurity measure (I)** of a dataset associated with a node: a **measure** of how far away from a **pure** node (all samples one class) we are
- Let's give each node a unique index *n*; and *K* is the number of classes
- $D_n \subseteq D = \{\boldsymbol{X}, y\}$ is the subset of the data at node *n*
- "Minimize probability of misclassification"

Classification error: $$I_E(D_n) = 1 - \max_{c \in K}\{\, p(c \mid D_n) \,\}$$

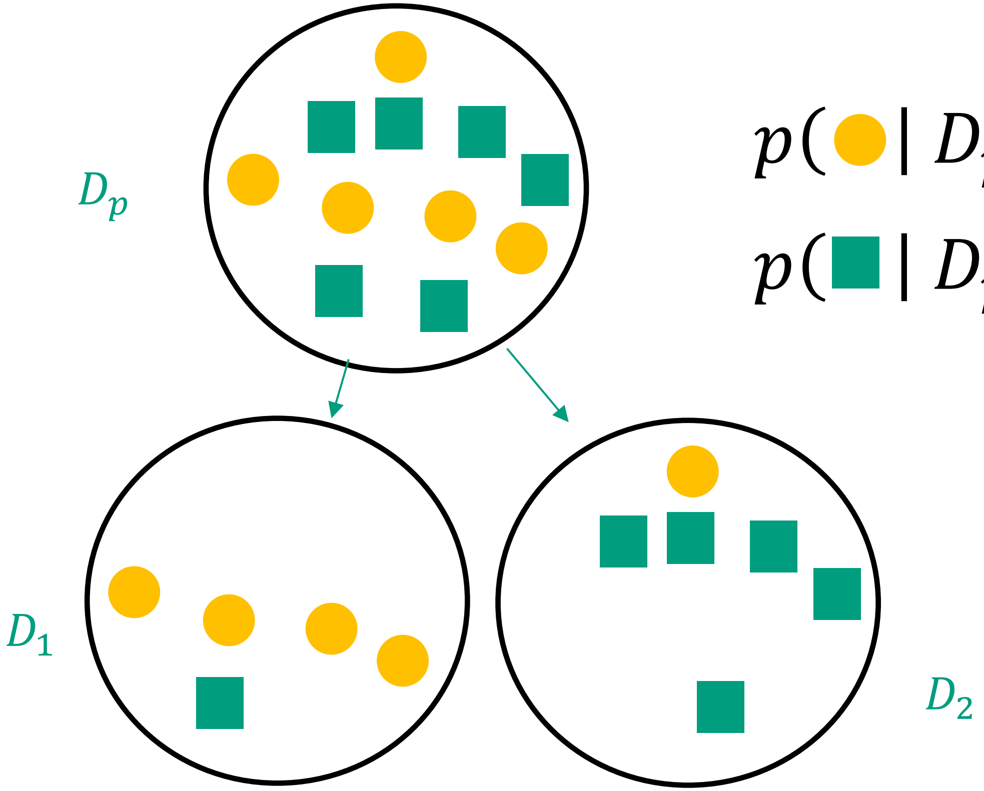
$p(c \mid D_n)$: Sample probability of a sample having class *c* in data subset D_n

$$\rightarrow p(c \mid D_n) = \frac{N_c}{\sum_{c \in K} N_c} \qquad \sum_{c \in K} p\,(c \mid D_n) = 1$$

Decision trees – Sample probability at node

$p(c \mid D_n)$: Sample probability of a sample having class *c* in data subset D_n

$$\rightarrow p(c \mid D_n) = \frac{N_c}{\sum_{c \in K} N_c} \qquad \sum_{c \in K} p\,(c \mid D_n) = 1$$



$$p(\bullet \mid D_p) = 5/11 = 0.45$$

$$p(\blacksquare \mid D_p) = 6/11 = 0.55 = 1 - p(\bullet \mid D_p)$$

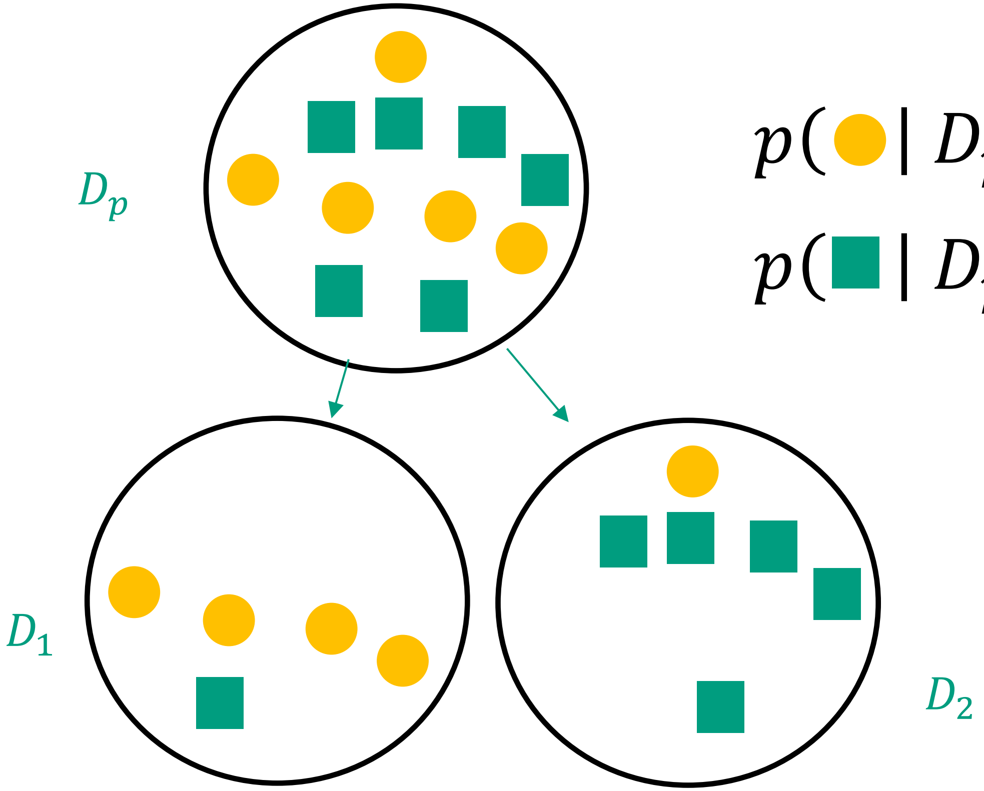
$$p(\bullet \mid D_1) = 4/5 = 0.8$$

$$p(\blacksquare \mid D_1) = 1/5 = 0.2 = 1 - p(\bullet \mid D_1)$$

Decision trees – Sample probability at node

$p(c \mid D_n)$: Sample probability of a sample having class *c* in data subset D_n

$$\rightarrow p(c \mid D_n) = \frac{N_c}{\sum_{c \in K} N_c} \qquad \sum_{c \in K} p\,(c \mid D_n) = 1$$



$$p(\bullet \mid D_p) = 5/11 = 0.45$$

$$p(\blacksquare \mid D_p) = 6/11 = 0.55 = 1 - p(\bullet \mid D_p)$$

$$p(\bullet \mid D_2) = 1/6 = 0.1\bar{6}$$

$$p(\blacksquare \mid D_2) = 5/6 = 0.8\bar{3} = 1 - p(\bullet \mid D_2)$$

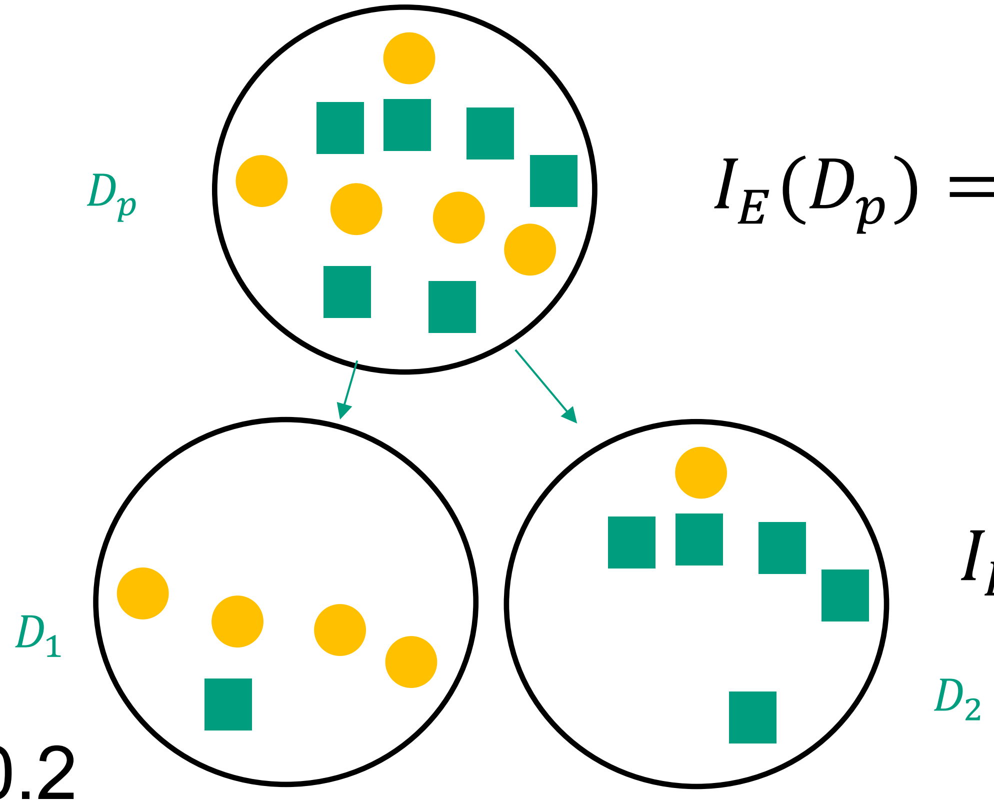
Decision trees – Impurity at node

$p(c \mid D_n)$: Sample probability of a sample having class *c* in data subset D_n

$$\rightarrow p(c \mid D_n) = \frac{N_c}{\sum_{c \in K} N_c}$$

Using classification error

$$I_E(D_n) = 1 - \max_{c \in K}\{\, p(c \mid D_n) \,\}$$



$$I_E(D_p) = 0.45$$

$$I_E(D_1) = 0.2$$

$$I_E(D_2) = 0.1\overline{6}$$

Decision trees – Information gain by split

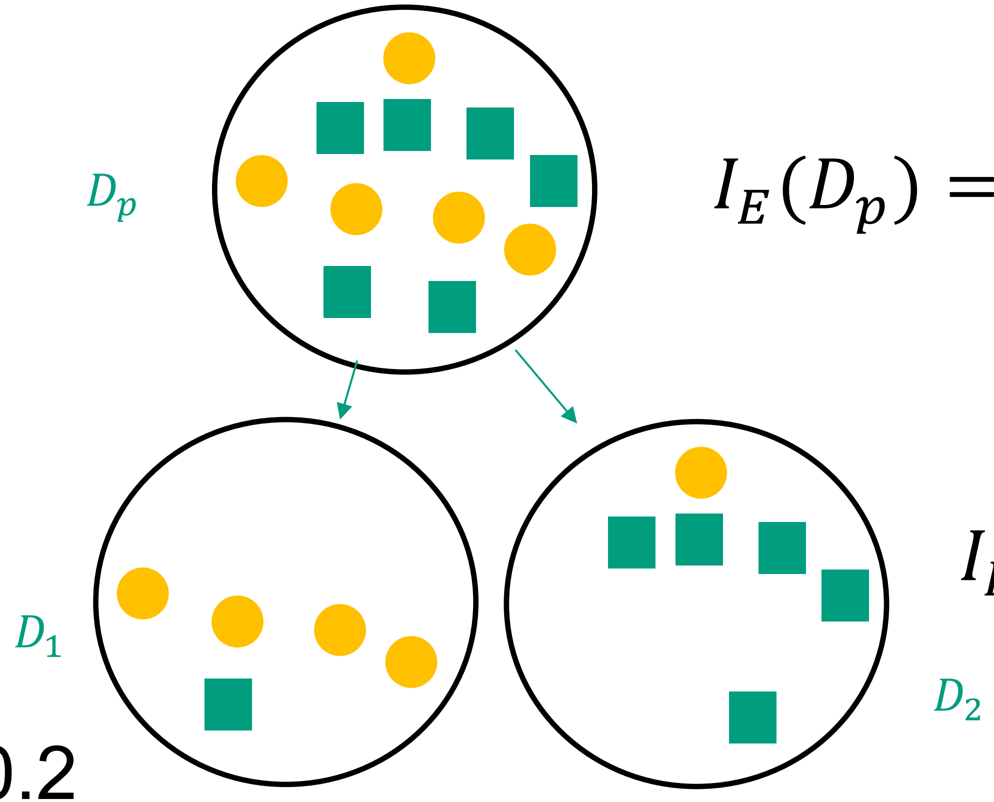
Using classification error

$$I_E(D_n) = 1 - \max_{c \in K}\{\, p(c \mid D_n) \,\}$$

Information gain

$$IG = I(D_p) - \sum_{j=1}^{2} \frac{N_j}{N_p} I(D_j)$$

$$= 0.45 \quad - (5/11 \cdot 0.2)$$
$$\quad - (6/11 \cdot 0.1\overline{6})$$
$$\approx 0.268$$



$I_E(D_p) = 0.45$

$I_E(D_2) = 0.1\overline{6}$

$I_E(D_1) = 0.2$

Decision trees – Information gain by split 2 (hypothetical)

Using classification error

$$I_E(D_n) = 1 - \max_{c \in K}\{\, p(c \mid D_n) \,\}$$

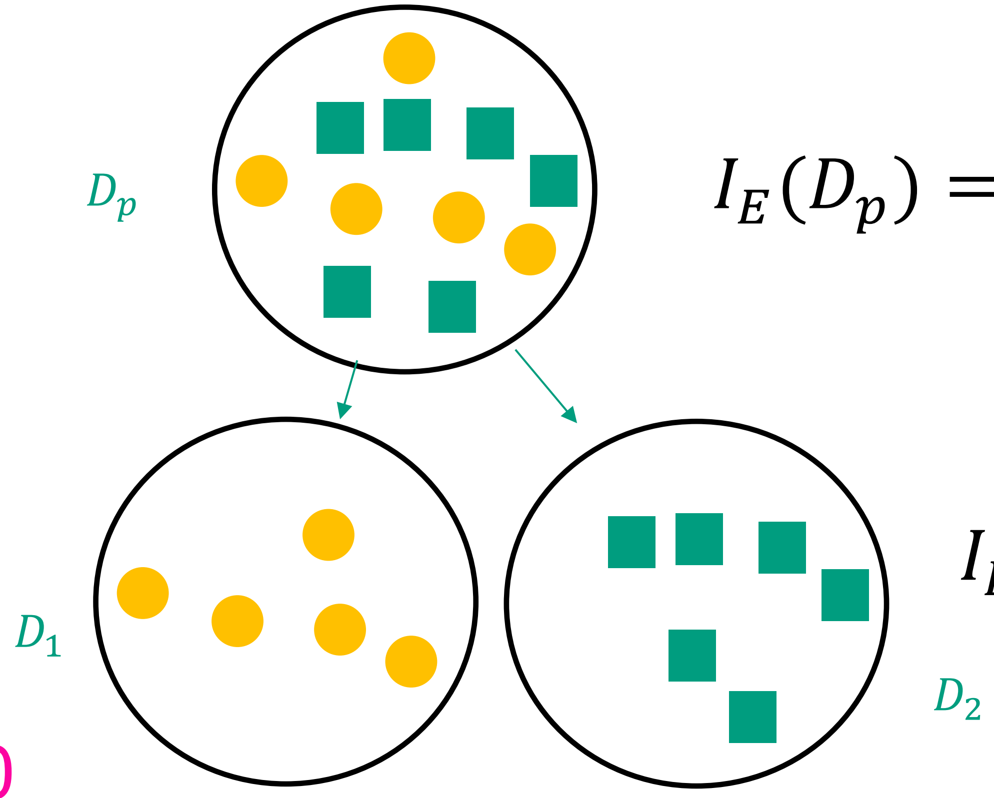
Information gain

$$IG = I(D_p) - \sum_{j=1}^{2} \frac{N_j}{N_p} I(D_j)$$

$= 0.45 \quad - (5/11 \cdot 0)$

$\qquad\qquad - (6/11 \cdot 0)$

$= 0.45$



$I_E(D_p) = 0.45$

$I_E(D_2) = 0$

$I_E(D_1) = 0$

pure leaf nodes

Decision trees – Impurity measures

Classification error: $I_E(D_n) = 1 - \max_{c \in K}\{ p(c \mid D_n) \}$

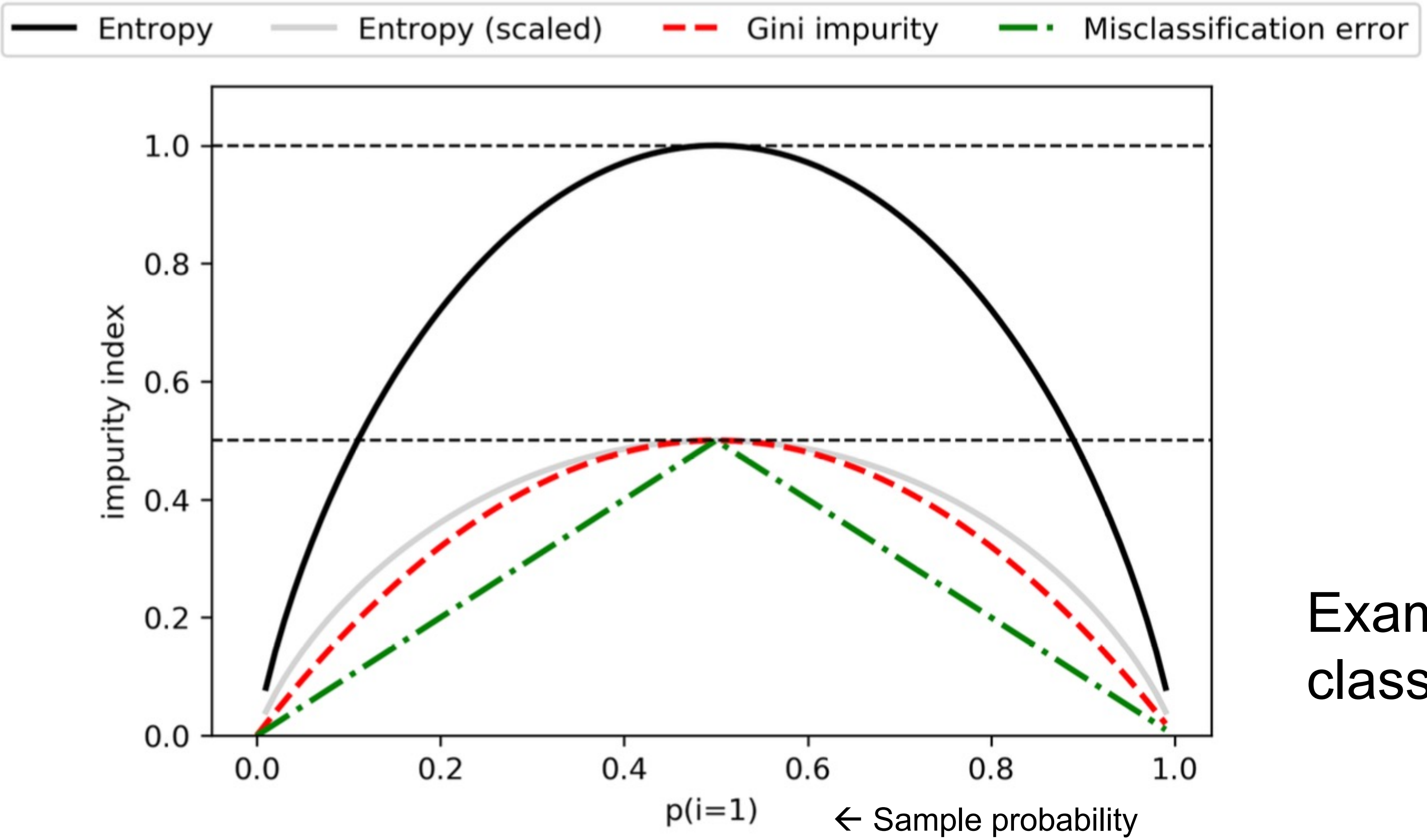
Entropy: $I_H(D_n) := -\sum_{c \in K} p(c|D_n) \log_2 p(c|D_n)$

(Information theory, Shannon 1948)

Gini impurity: $I_{Gini}(D_n) = \sum_{c \in K} p(c|D_n)\left(1 - p(c|D_n)\right) = 1 - \sum_{c \in K} p(c|D_n)^2$

Impurity measure (I) of a dataset associated with a node: a **measure** of how far away from a **pure** node (all samples one class) we are

Decision trees – Impurity measures (overview)



← Sample probability

Example for binary classification

Figure from Python Machine Learning (Raschka & Mirjalili)

Decision trees – Impurity measures (overview)

- In practice **Gini impurity** or **Entropy** are used
- They are **differentiable** functions (classification error is not) which simplifies finding optimal split
- They are more sensitive to changes in class probability and prefer splits which result in pure nodes with higher probability

Decision trees – Scikit-learn example Iris

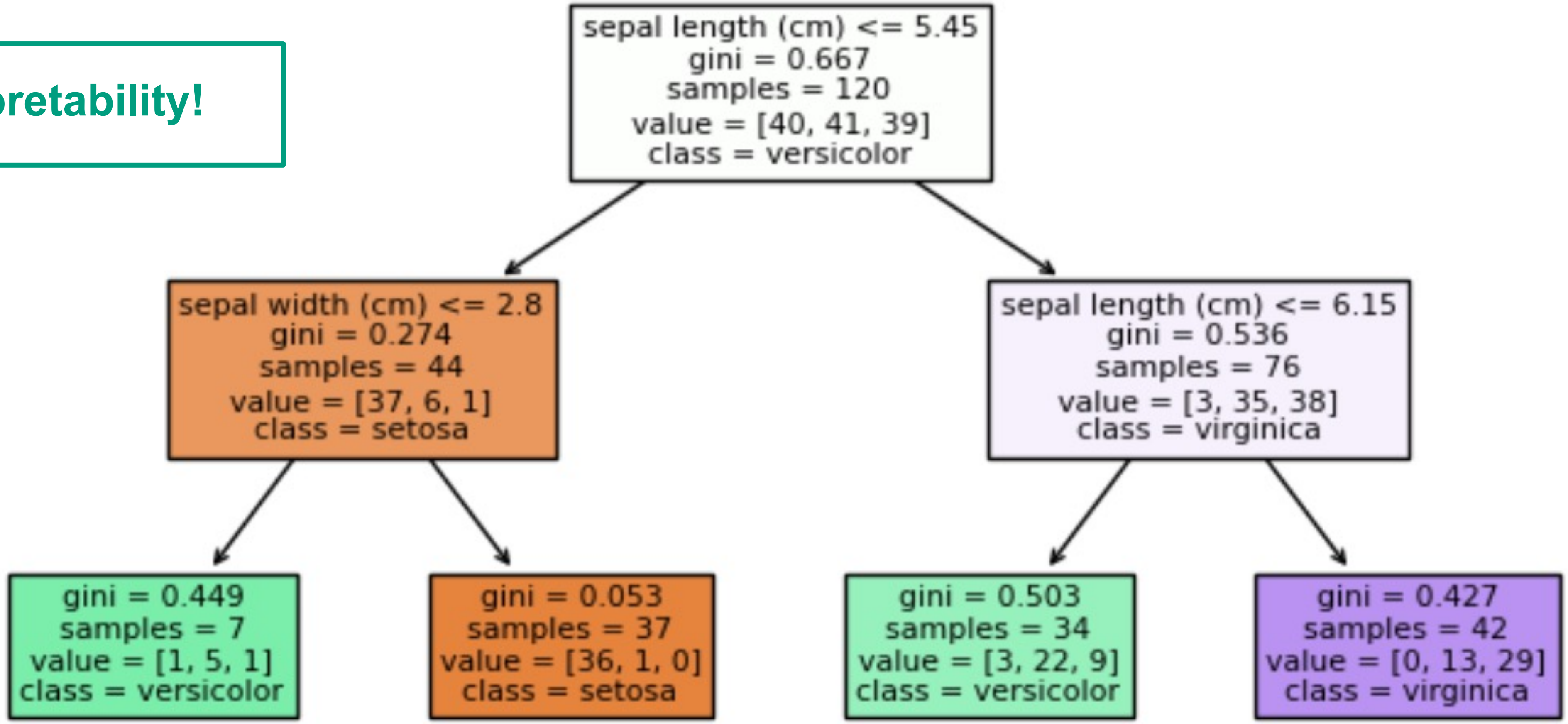
- Train a decision tree on the iris data set
- `from sklearn.tree import DecisionTreeClassifier`
- Plot the decision tree graphically
- `from sklearn.tree import plot_tree`

`03_decision_tree_iris.ipynb`

- Examples for hyperparameters: `max_depth`, `min_samples_leaf`, `criterion`
- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

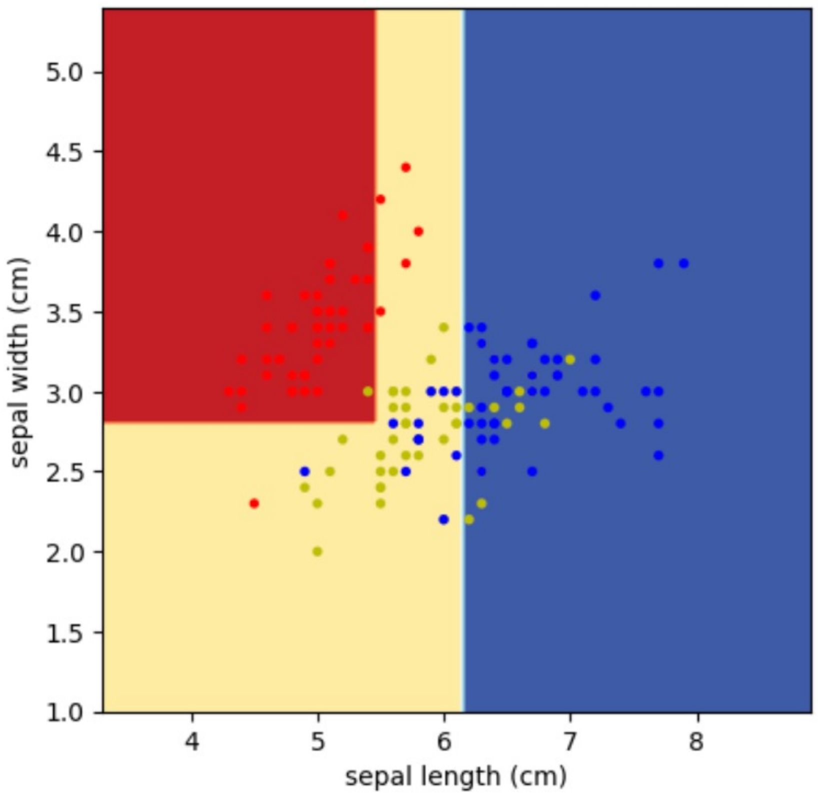
Decision trees – Scikit-learn example Iris

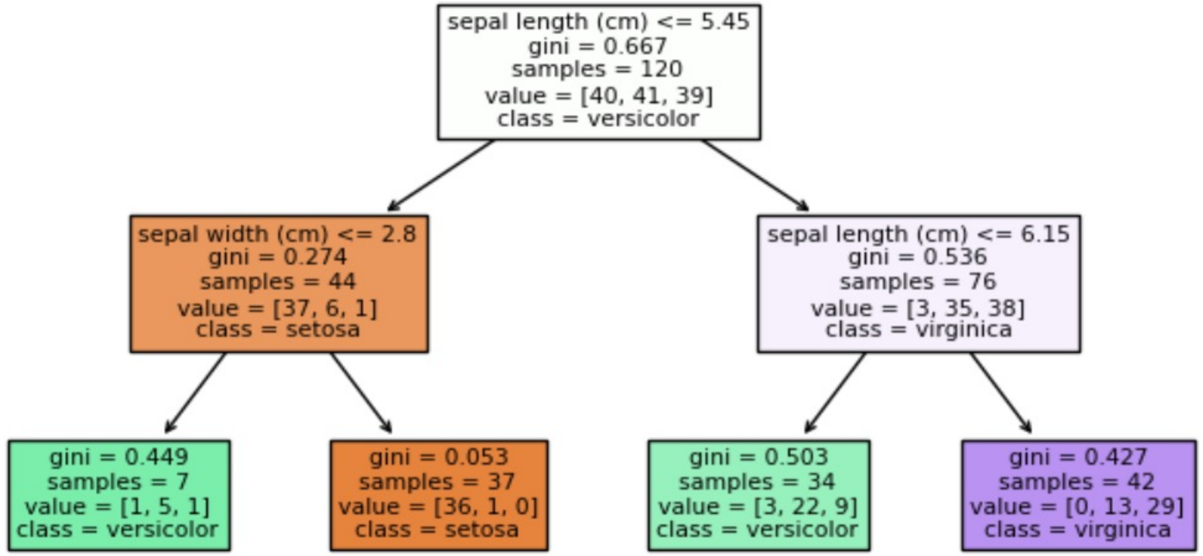
Interpretability!



`03_decision_tree_iris.ipynb`

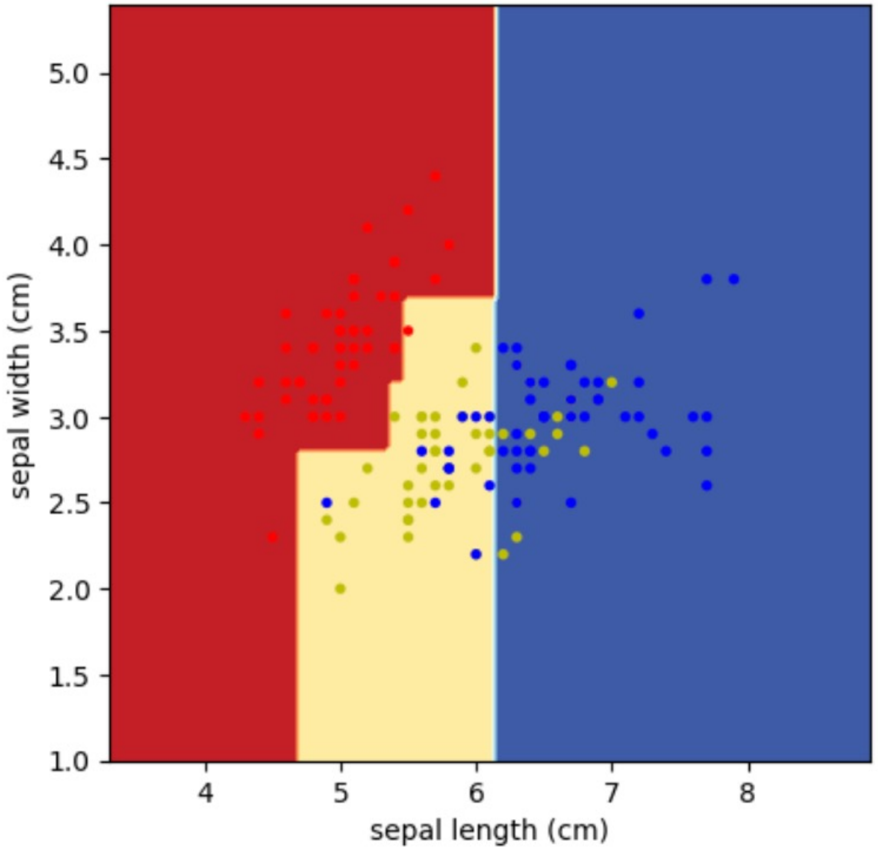
Decision trees – Scikit-learn example Iris (max_depth=2)

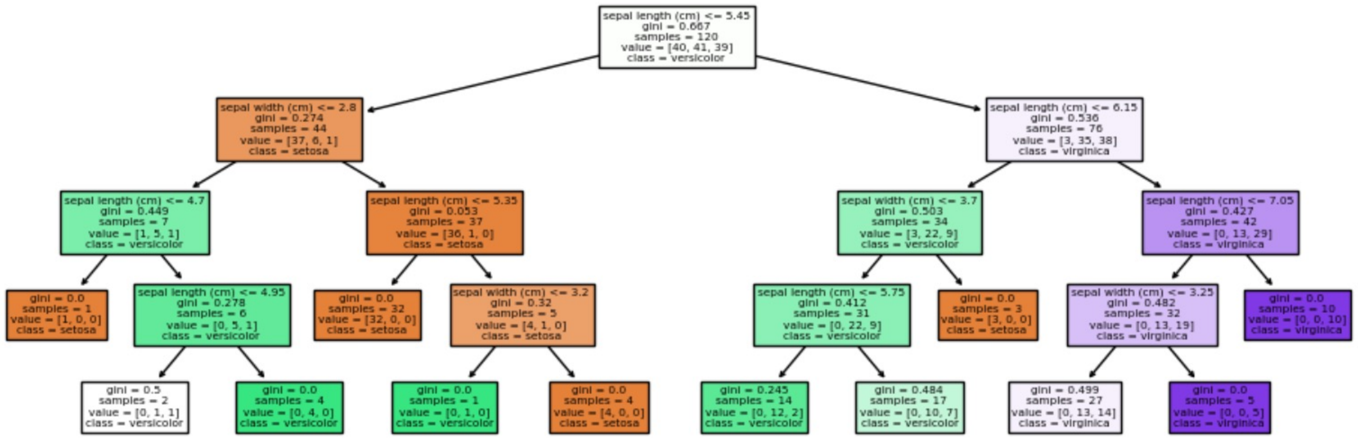




`03_decision_tree_iris.ipynb`

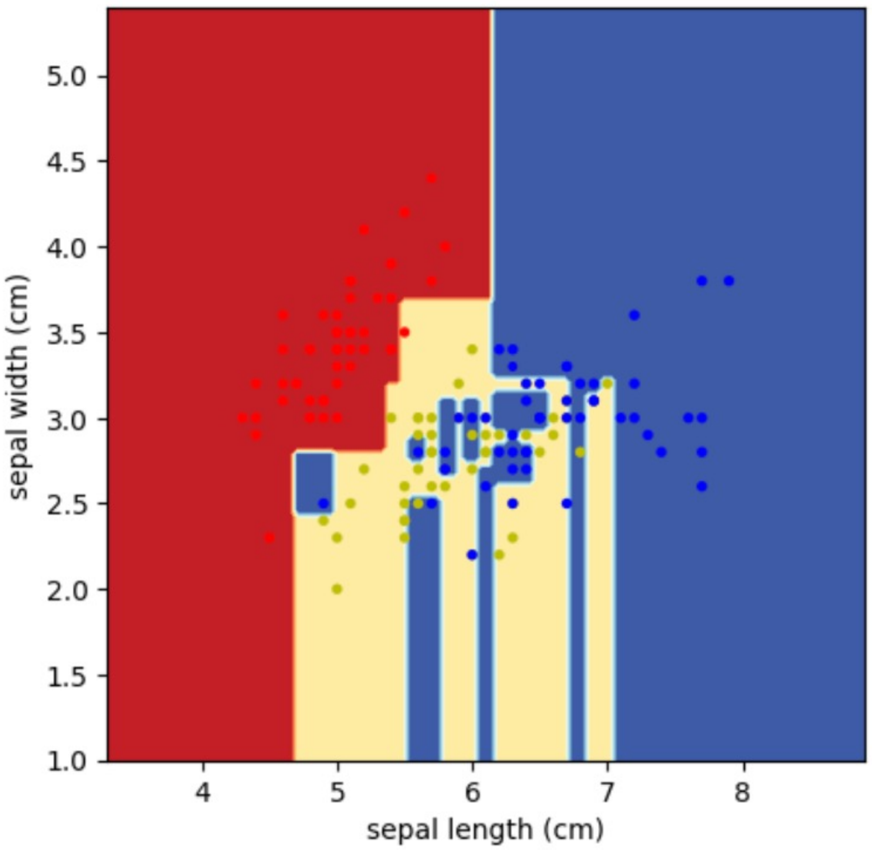
Decision trees – Scikit-learn example Iris (max_depth=4)

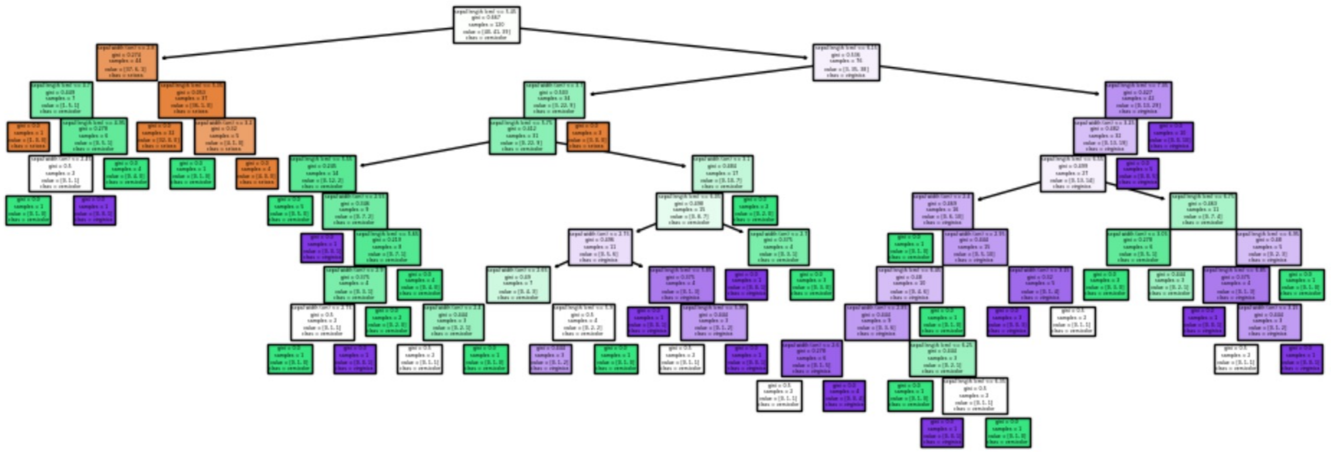




03_decision_tree_iris.ipynb

Decision trees – Scikit-learn example Iris (max_depth=20)





03_decision_tree_iris.ipynb

Overfitting!

Decision trees – Pruning trees

- Tuning the **tree depth** or **minimal number of samples in a leaf node** e.g. via **grid search**
- Or, for example, pruning via "*cost-complexity analysis*" in sci-kit learn (not syllabus)

`03_decision_tree_cost_complexity.ipynb`

Decision trees (summary)

- **Decision trees** can build **complex (non-linear)** decision boundaries by dividing the feature space into **rectangles**
- They are easy to interpret
- Need to be careful regarding **depth** of the decision tree
  - more complex the decision boundaries can easily result in **overfitting**
- **Note:** Feature scaling is **not** a requirement for decision tree algorithms
  - splits are easier to interpret with original scale

Scikit-learn and Tour of Classifiers

Random Forests

Random forests

- Combining **multiple decision trees** into a powerful classifier
- **Bagging** and **boosting** are methods to combine trees (more after Easter break)
- Instance of an **ensemble learning** algorithm (more detailed after Easter break)
- **Average** multiple trees (each **high variance**) to obtain a more **robust** model
- Very popular due to
  - their good classification performance
  - their scalability
  - their ease of use

Random forests – Bagging

- **Goal:** Build a more **robust** model that has a **better generalisation** performance and is **less susceptible** to **overfitting**

- **Bagging** stands for **bootstrap aggregating** (Breiman 1994)
- Averaging **decreases** the **variance** of the model, **without increasing** the **bias**

Random forests – Bagging

Random forest algorithm can be summarised in **four** simple steps

1. Draw a random **bootstrap sample** of size n (**randomly choose** n samples from the training set **with replacement**)

2. **Grow a decision tree** from the bootstrap sample. At each node
   - **Randomly select** d **features (**without replacement**)**
   - **Split** the node using the feature that provides the **best split** according to the objective function, for instance, **maximising the information gain**

3. **Repeat** the steps 1. and 2. **k times** (k: number of trees to be computed)

4. **Aggregate** the **prediction** by each tree to assign the class label by **majority vote**

Random forests – Bagging in images (bootstrap sample)

Volume	Clusters	Shape factor	Malignant?
350	4	1	no
100	1	1	no
260	1	3.5	yes
10	3	2.5	yes

Random forests – Bagging in images (bootstrap sample)

Volume	Clusters	Shape factor	Malignant?
350	4	1	no
100	1	1	no
260	1	3.5	yes
10	3	2.5	yes

1. Random pick

Volume	Clusters	Shape factor	Malignant?
350	4	1	no

Random forests – Bagging in images (bootstrap sample)

Volume	Clusters	Shape factor	Malignant?
350	4	1	no
100	1	1	no
260	1	3.5	yes
10	3	2.5	yes

2. Random pick

Volume	Clusters	Shape factor	Malignant?
350	4	1	no
260	1	3.5	yes

Random forests – Bagging in images (bootstrap sample)

Volume	Clusters	Shape factor	Malignant?
350	4	1	no
100	1	1	no
260	1	3.5	yes
10	3	2.5	yes

Volume	Clusters	Shape factor	Malignant?
350	4	1	no
260	1	3.5	yes
100	1	1	no

3. Random pick

Random forests – Bagging in images (bootstrap sample)

Volume	Clusters	Shape factor	Malignant?
350	4	1	no
100	1	1	no
260	1	3.5	yes
10	3	2.5	yes

Volume	Clusters	Shape factor	Malignant?
350	4	1	no
260	1	3.5	yes
100	1	1	no
100	1	1	no

4. Random pick

Random forests – Bagging in images (bootstrap sample)

Original data

Volume	Clusters	Shape factor	Malignant?
350	4	1	no
100	1	1	no
260	1	3.5	yes
10	3	2.5	yes

Bootstrapped sample

Volume	Clusters	Shape factor	Malignant?
350	4	1	no
260	1	3.5	yes
100	1	1	no
100	1	1	no

Sample can occur multiple times

Some samples don’t occur

Random forests – Bagging in images (select features)

Volume	Clusters	Shape factor	Malignant?
350	4	1	no
100	1	1	no
260	1	3.5	yes
10	3	2.5	yes

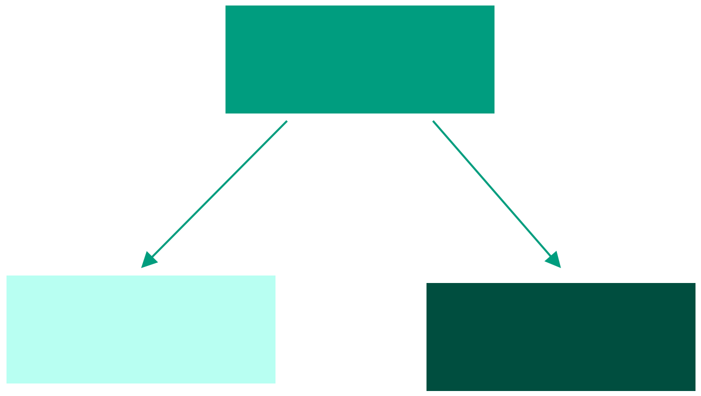
Randomly select features

Volume	Shape factor	Malignant?
350	1	no
100	1	no
260	3.5	yes
10	2.5	yes

Compute split that maximizes information gain among the selected features

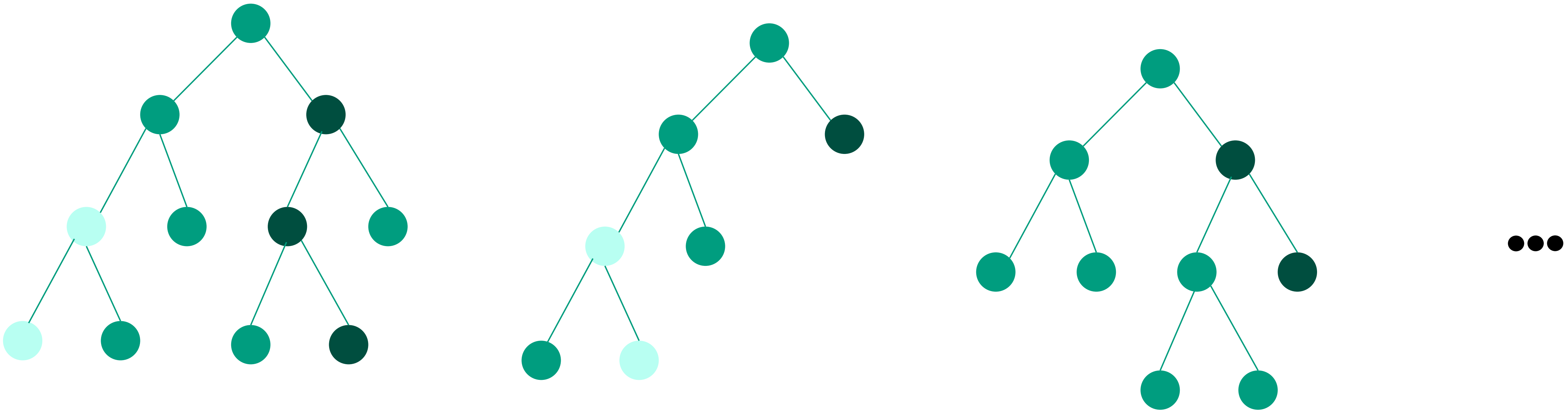
Repeat for every node

Each split at any node may use a different subset of features



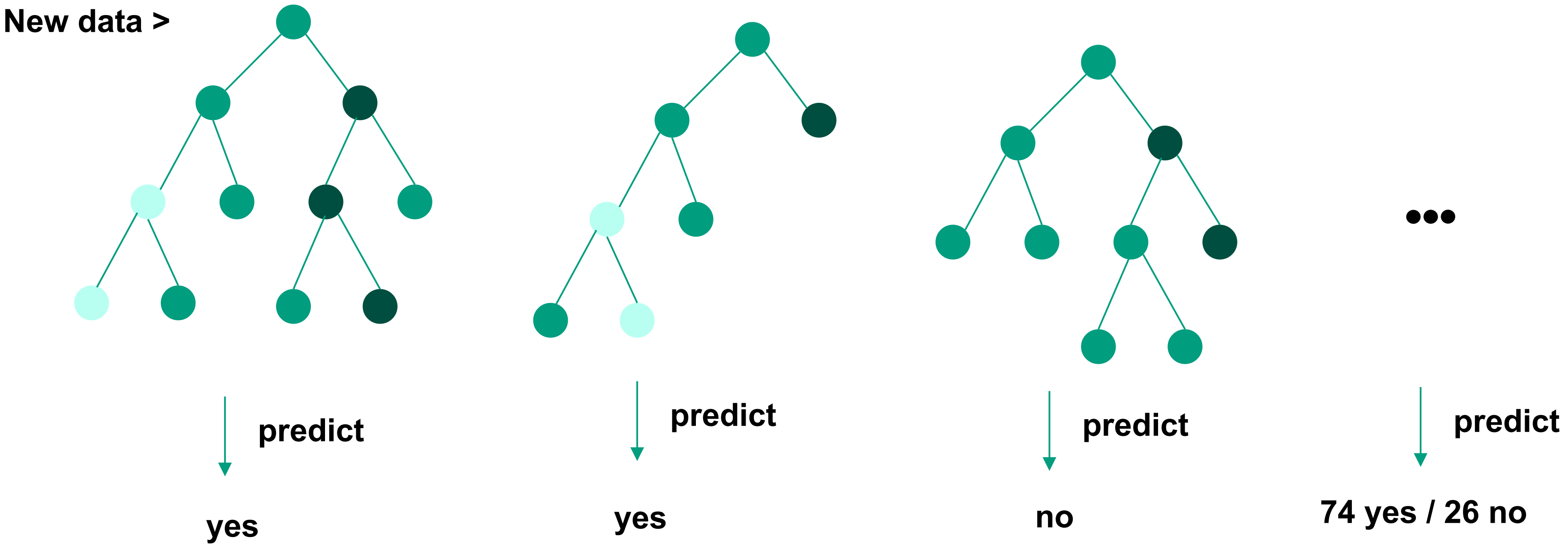
Random forests – Bagging in images (many trees)

Train many trees – each will look a bit different



Random forests – Bagging in images (voting)

Train many trees – each will look a bit different



Majority vote: **yes**

Random forests – Bagging summary

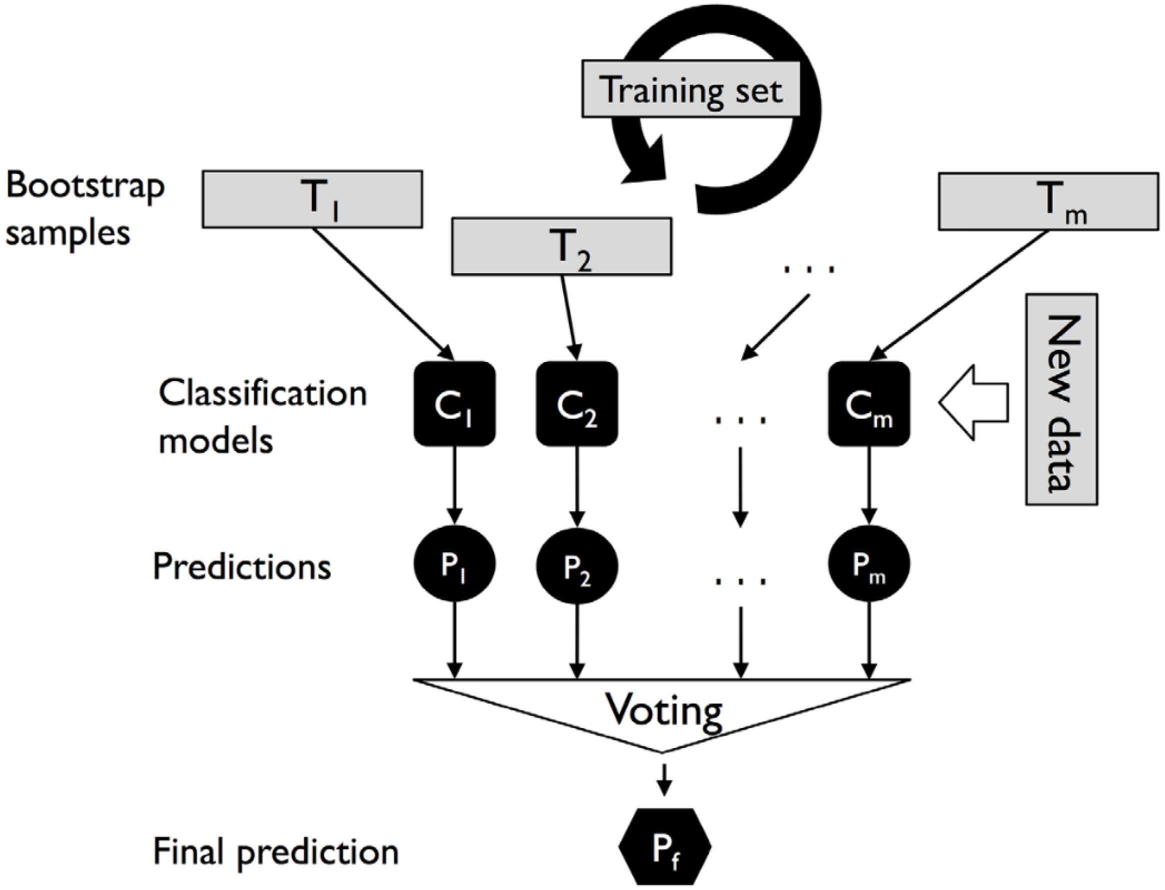
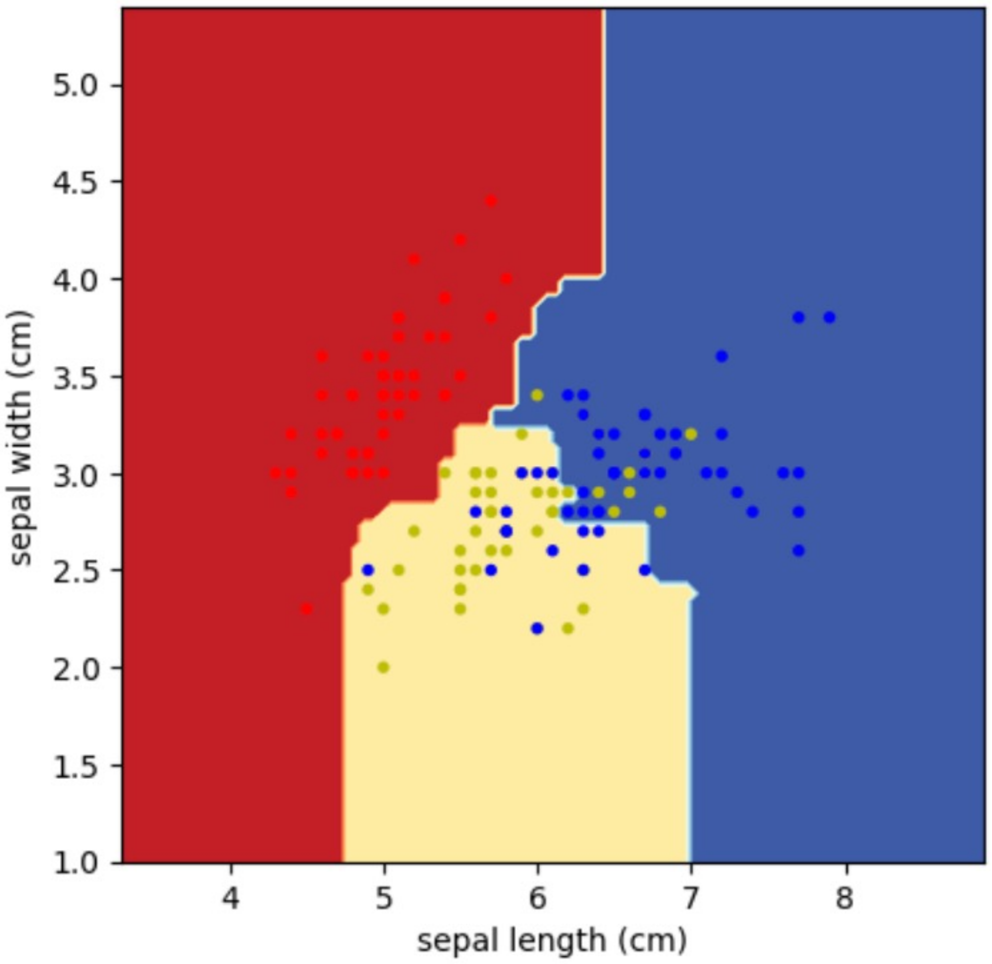


Figure from Python Machine Learning (Raschka & Mirjalili)

Random forest classifier (code example iris)



max_depth=3
n_estimators=100
n_jobs=2 (compute tree in parallel)

Decision boundary is smoother because it's obtained by a majority vote over 100 high variance decision trees

Random forest has usually a better generalization error than a single tree

```
03_random_forest_iris.ipynb
```

```
03_randomforest_and_decisiontree.ipynb
```

Random forest classifier (summary)

- Robust classifier by averaging over many trees
- Do not offer the same level of interpretability as plain decision trees
- Typically, the more trees are used, the better
- But: using more than one tree results in higher computational effort
- Less hyperparameters to tune due to robustness
  - Number of trees (`n_estimators`)
  - Size of bootstrap samples → usually fixed to training size
  - Number of features to randomly select → usually fixed to $\sqrt{m}$ for m features