Machine Learning

| Decision Tree: Visualizing CART

Prof. Dr. Mostafa Elhosseini

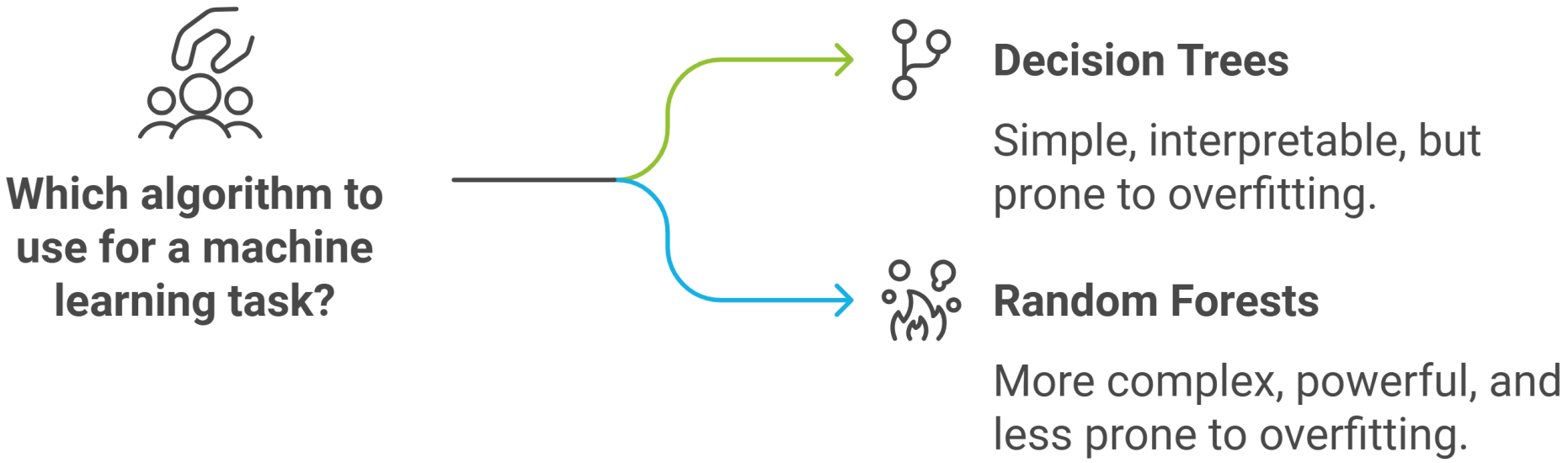
https://youtube.com/ElhosseiniAcademy

Agenda

- DT Vs. Random Forest
- White Box Model
- CART Algorithm
- Greedy Algorithm
- NP-complete problem
- Computational Complexity
- Visualizing DT - Graphviz

Decision Tree

- Decision trees are also the fundamental components of **random forests**.



White Box Vs. Black Box

- Decision Trees are Intuitive and **interpretable**; known as **white-box models** because their decision paths are transparent.
- Provide clear, simple rules that can be easily explained or applied manually.
- The field of interpretable ML aims at creating ML systems that can **explain their decisions** in a way humans can understand

White Box Vs. Black Box

- **Random Forests and Neural Networks** are Considered **black-box** models; they make **accurate** predictions, but the reasoning behind these predictions is often **hard to interpret**.
- Difficult to pinpoint specific features or aspects that influenced a prediction.
- if a neural network says that a **particular person appears in a picture**, it is hard to know what contributed to this prediction: Did the model recognize that person's eyes? Their mouth? Their nose? Their shoes? Or even the couch that they were sitting on?

CART Algorithm

- **CART** stands for **C**lassification **a**nd **R**egression **T**rees

- Scikit-Learn uses the **CART algorithm**, which produces only **binary trees**, meaning trees where split nodes always have exactly two children (i.e., questions only have yes/no answers).

The CART Training Algorithm

- Scikit-Learn uses the Classification and Regression Tree (**CART**) algorithm to train **decision trees** (also called "**growing**" trees).
- The algorithm works by first **splitting the training set** into two subsets using a single **feature k** and a **threshold t_k** (e.g., "petal length ≤ 2.45 cm").

- How does it choose k and t_k ?
- It searches for the pair (k, t_k) that produces the **purest subsets**, weighted by their size

The CART Training Algorithm

- Once the CART algorithm has successfully split the training set in two, it splits the subsets using the same logic, then the sub-subsets, and so on.
- It stops recursing once it reaches the **maximum depth** (defined by the **max_depth** hyperparameter), or if it cannot find a split that will reduce impurity.
- A few **other hyperparameters** control additional stopping conditions: min_samples_split, min_samples_leaf, min_weight_fraction_leaf, and max_leaf_nodes.

CART is a Greedy algorithm

- it greedily searches for an **optimum split at the top level**, then repeats the process at each subsequent level.
- It does not check whether or not the split will lead to the lowest possible impurity several levels down.
- A greedy algorithm often produces a solution that's reasonably good but not guaranteed to be optimal.
- Unfortunately, finding the optimal tree is known to be an **NP-complete problem**. It requires $O(\exp(m))$ time, making the problem **intractable** even for small training sets. This is why we must settle for a "**reasonably good**" solution when training decision trees.

NP-complete problem

- **Class NP** includes problems for which a given solution can be **verified quickly** (in polynomial time) by a deterministic computer.
- This **doesn't mean** they can be solved quickly, only that if someone provides a solution, we can **verify** its correctness quickly.

- What Makes a Problem NP-Complete?
  - It's in NP (its solutions are verifiable in polynomial time).
  - Every other problem in NP can be transformed (reduced) to it in polynomial time.

NP-complete problem

Examples of NP-Complete Problems:

- Traveling Salesman Problem
- Boolean Satisfiability Problem (SAT)
- Knapsack Problem
- Graph Coloring

CART is a Greedy algorithm

- It requires $O(\exp(m)) = O(2^m)$ - where m : # instances
- In Big-O notation, the base of an exponential function does not change the complexity class. Both $O(e^m)$ and $O(2^m)$ describe functions that grow exponentially with m.
- It means that as m increases, the time required to compute grows extremely fast, often **doubling** with each additional feature

ID3 – C4.5

- **ID3** stands for **I**terative **D**ichotomiser 3
- **ID3** can produce decision trees with nodes that have more than two children.
- ID3 utilizes concepts from information theory, specifically **entropy** and information **gain**
- C4.5 Algorithm: An improvement over ID3

Training and Visualizing

- **DecisionTreeClassifier** - Iris dataset

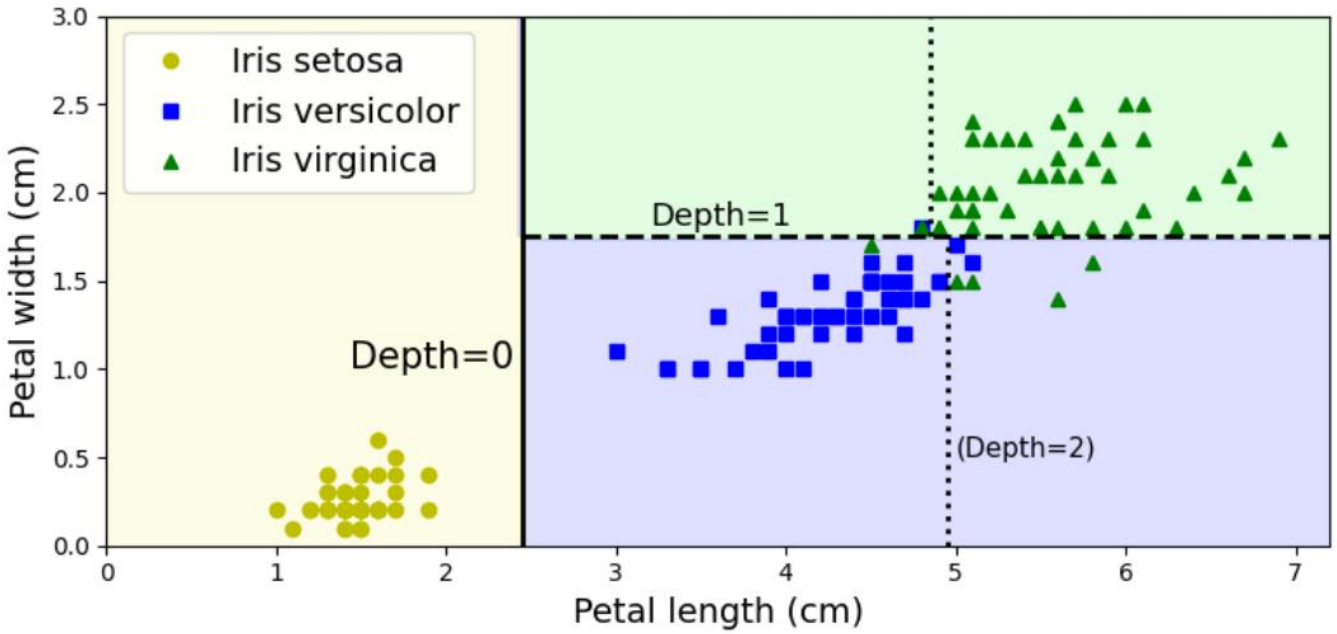
```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
```

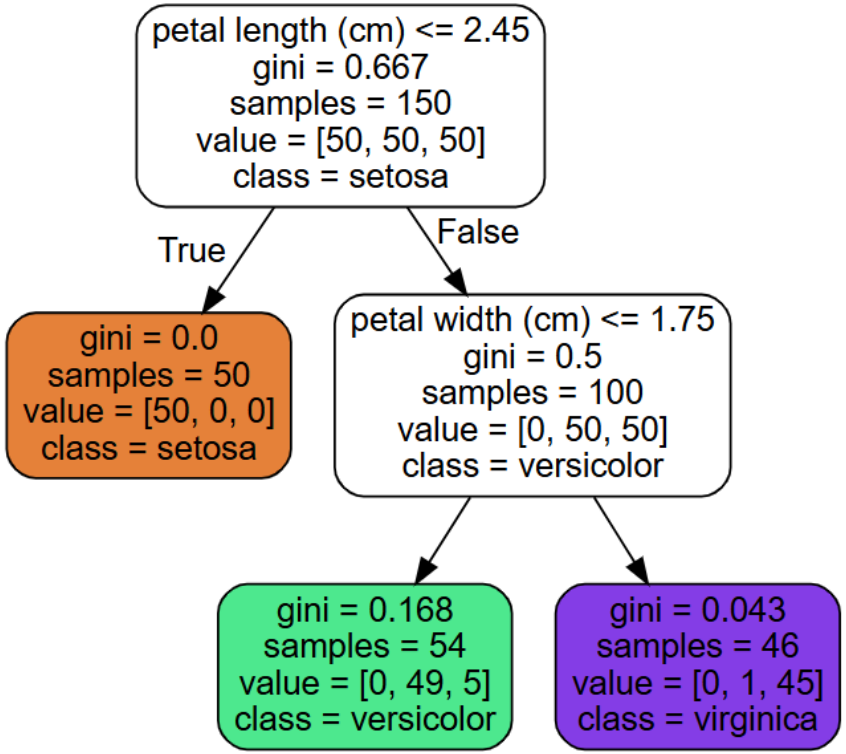
```
iris = load_iris(as_frame=True)
X_iris = iris.data[["petal length (cm)", "petal width (cm)"]].values
y_iris = iris.target
```

```
tree_clf = DecisionTreeClassifier(max_depth=2, random_state=42)
tree_clf.fit(X_iris, y_iris)
```

Training and Visualizing

- **Graphviz** is an open-source graph visualization software package

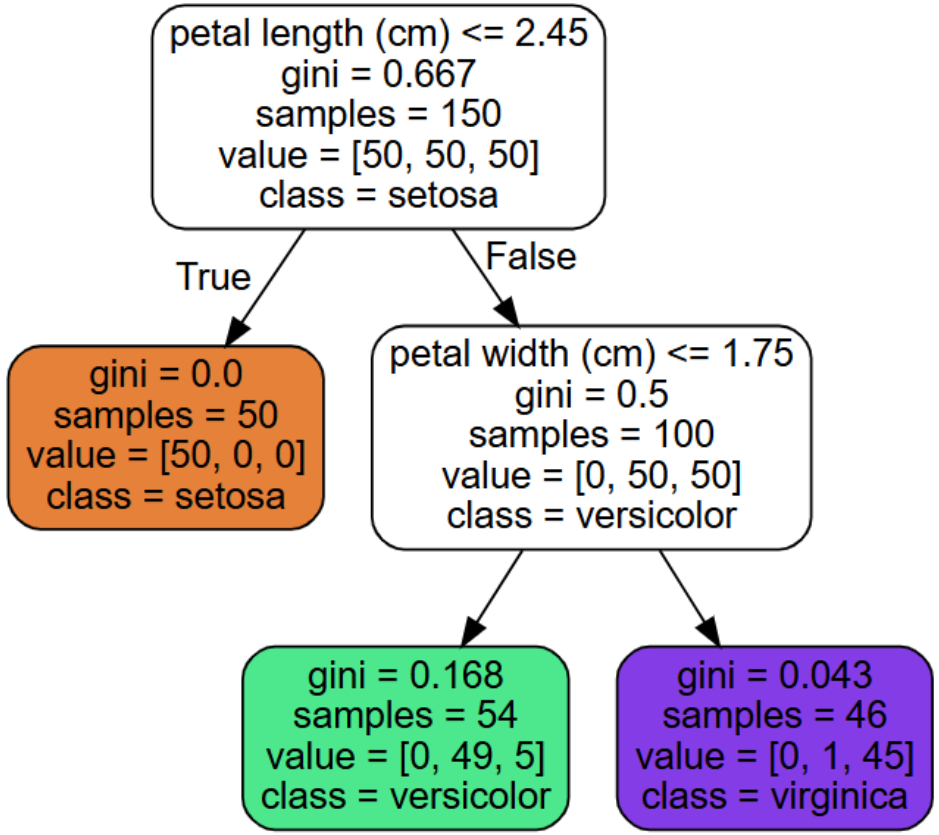




Training and Visualizing

- **Samples attribute** counts how many training instances it applies to.

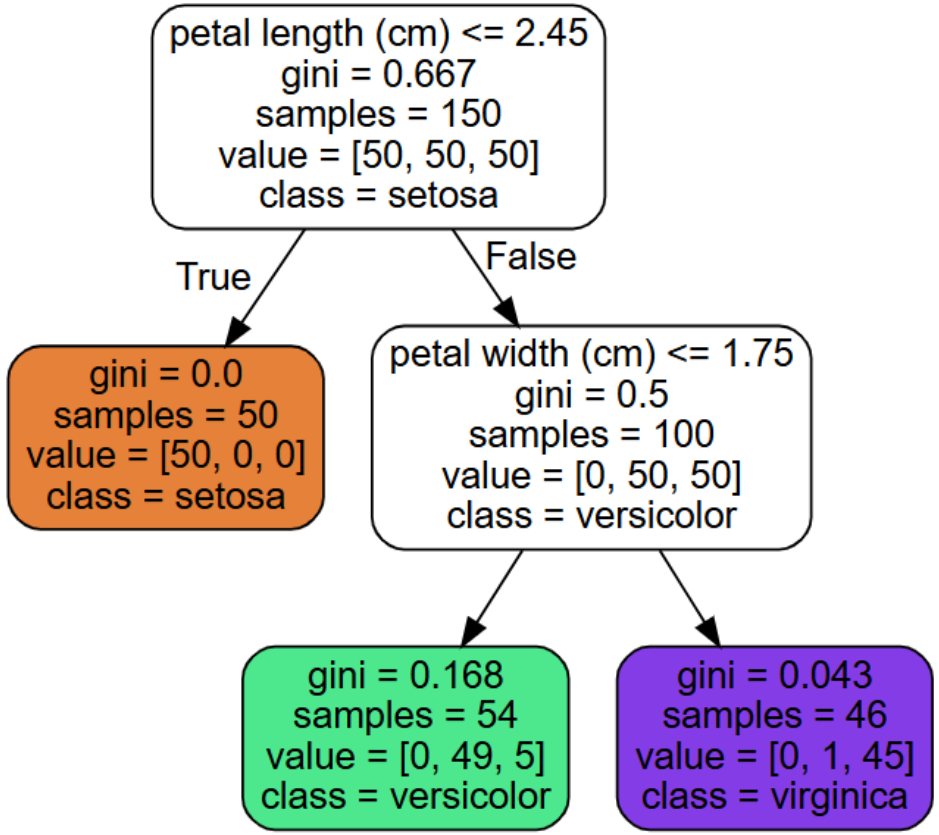
- For example, 100 training instances have a petal length greater than 2.45 cm (depth 1, right)



Training and Visualizing

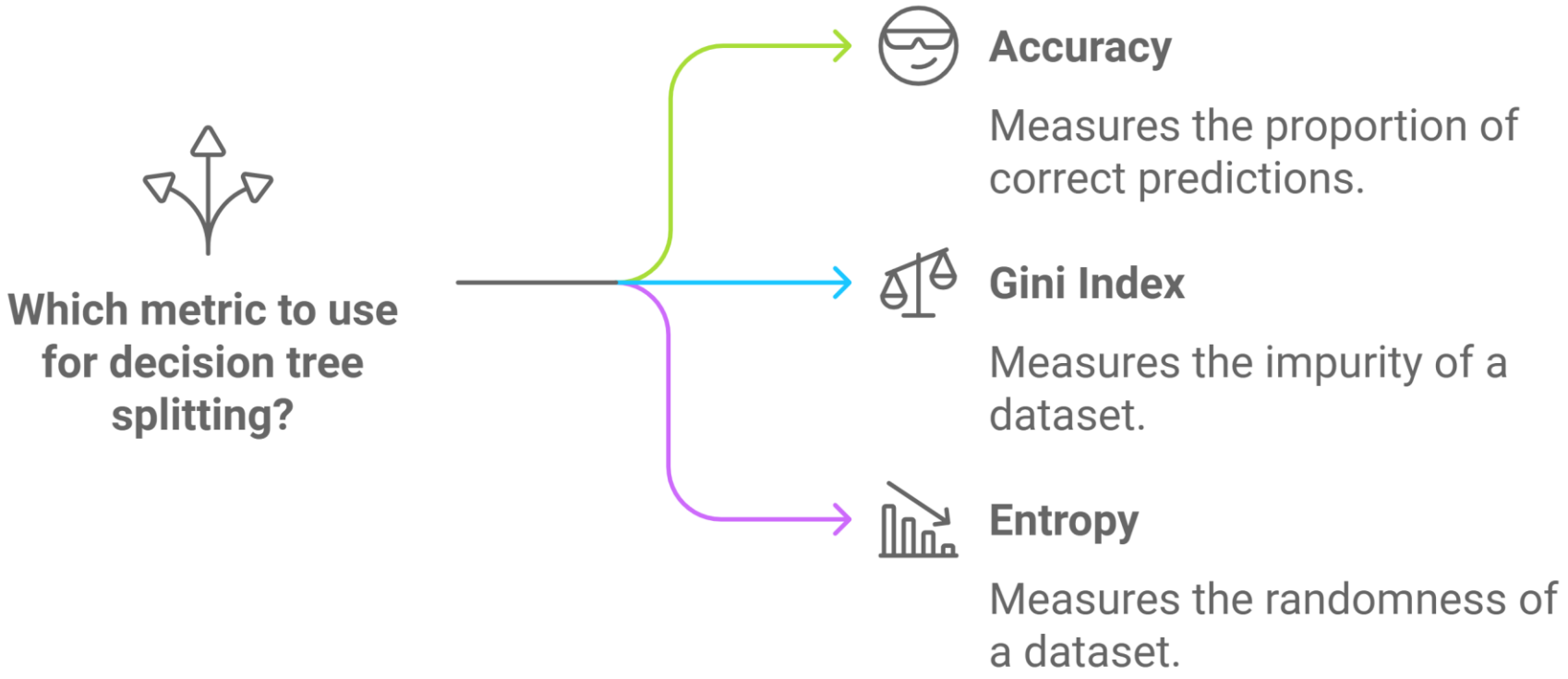
- **Value attribute** tells you how many training instances of each class this node applies to

- For example, the bottom-right node applies to 0 Iris setosa, 1 Iris versicolor, and 45 Iris virginica.



Metrics

- How exactly do you decide which is the best possible question to ask?

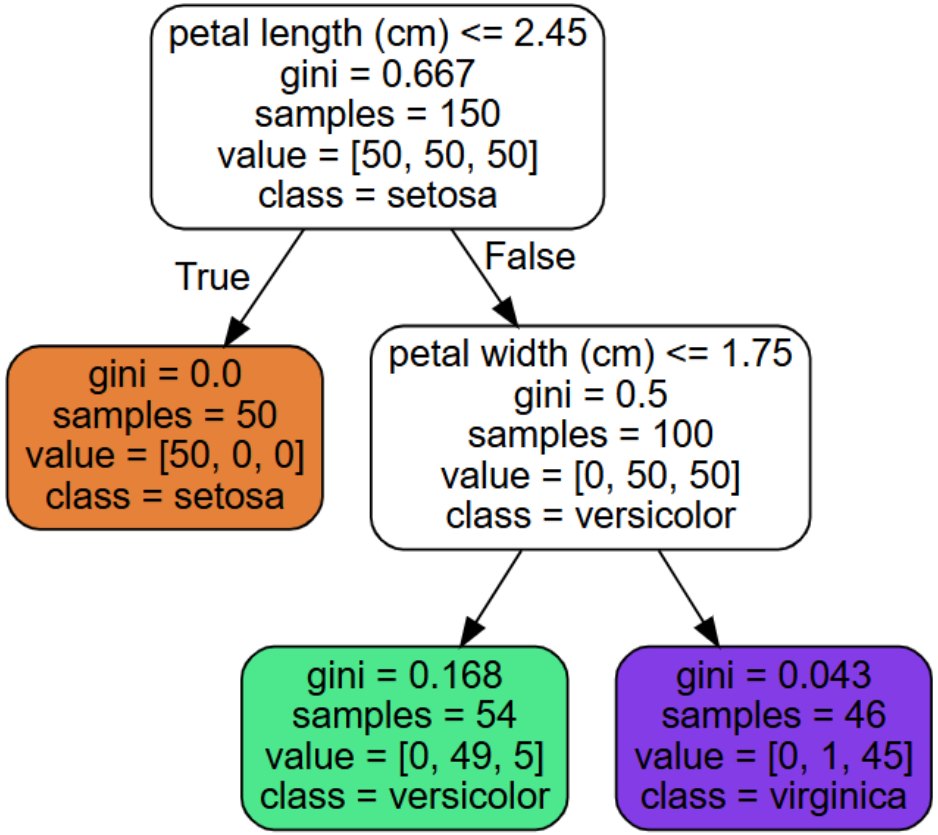


Gini Index

- The **Gini Index**, also known as **Gini Impurity**, is a metric used to evaluate the quality of a split in decision trees.
- In a set with m elements and n classes, with a_i elements belonging to the $i-th$ class, the Gini impurity index is

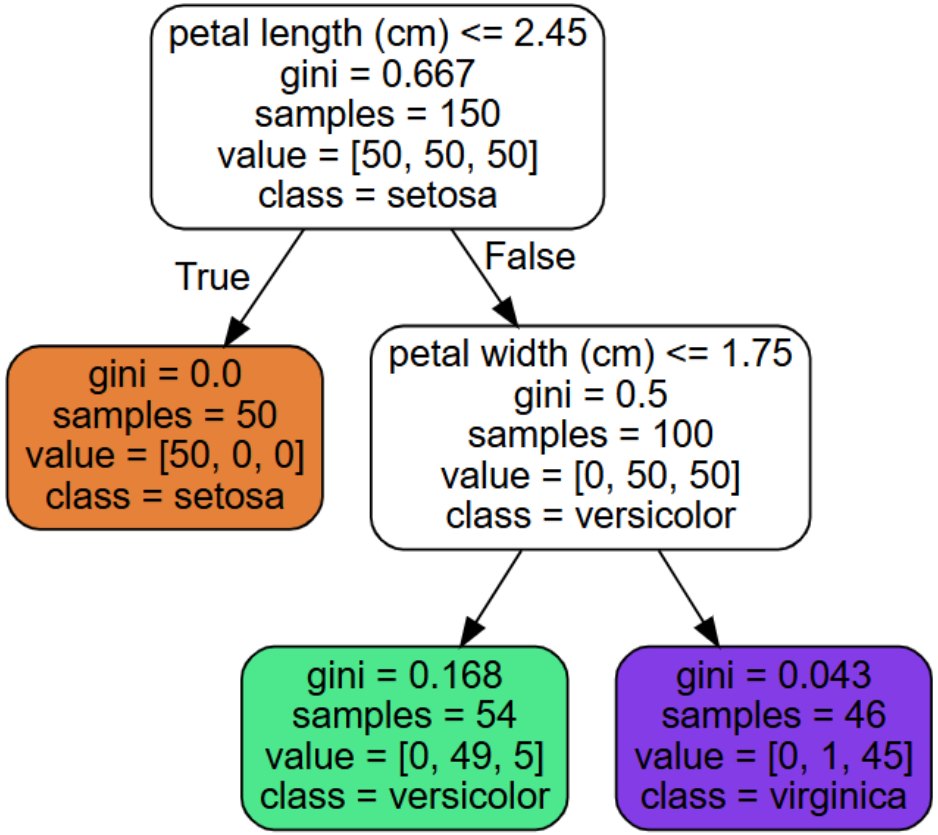
$$\text{Gini} = 1 - {p_1}^2 - {p_2}^2 - {p_3}^2 \ldots {p_n}^2$$

- $p_i = \frac{a_i}{m}$



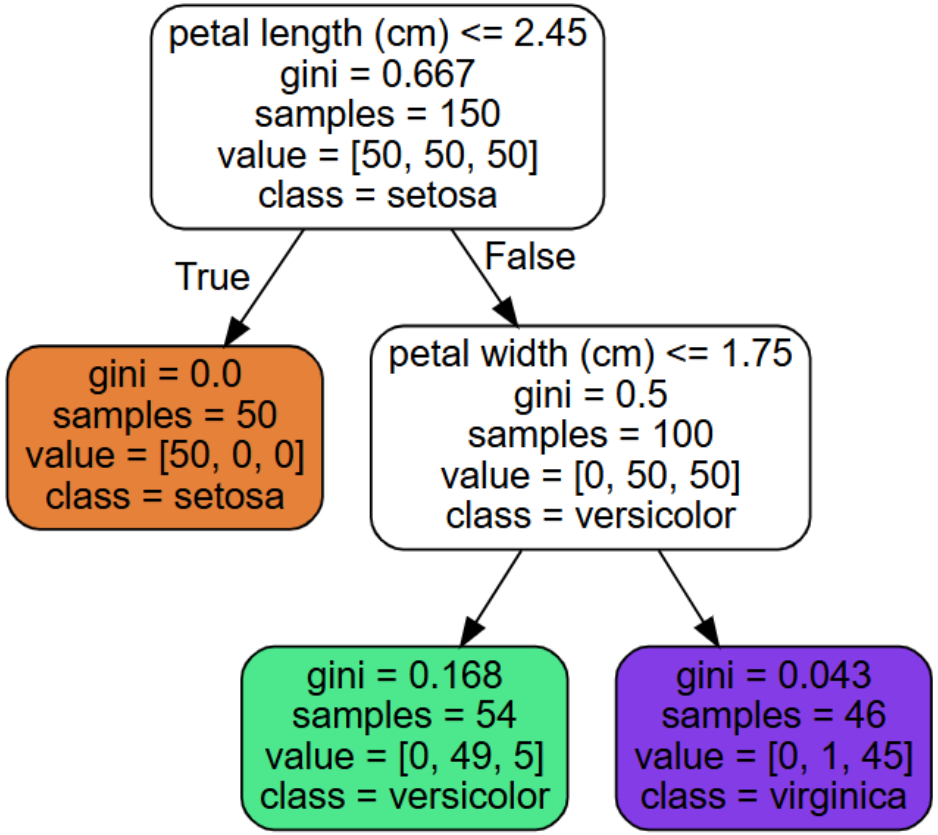
Gini Index

- A node is “pure” (Gini=0) if all training instances it applies to belong to the same class.
- For example, since the depth- 1 left node applies only to Iris setosa training instances, it is pure and its Gini impurity is 0.
- The depth-2 left node has a Gini impurity equal to
- Gini impurity index $= 1 - \left(\frac{0}{54}\right)^2 - \left(\frac{49}{54}\right)^2 - \left(\frac{5}{54}\right)^2 = 0.168$



Estimating Class Probabilities

- A decision tree can also estimate the probability that an instance belongs to a particular class k. First it traverses the tree to find the leaf node for this instance, and then it returns the ratio of training instances of class k in this node
- flower whose petals are 5 cm long and 1.5 cm wide.



Estimating Class Probabilities

```
tree_clf.predict_proba([[5, 1.5]]).round(3)
```

```
array([[0.   , 0.907, 0.093]])
```

```
tree_clf.predict([[5, 1.5]])
```

```
array([1])
```