Decision Trees

Decision Trees

In this session we'll learn about **Decision Trees**.

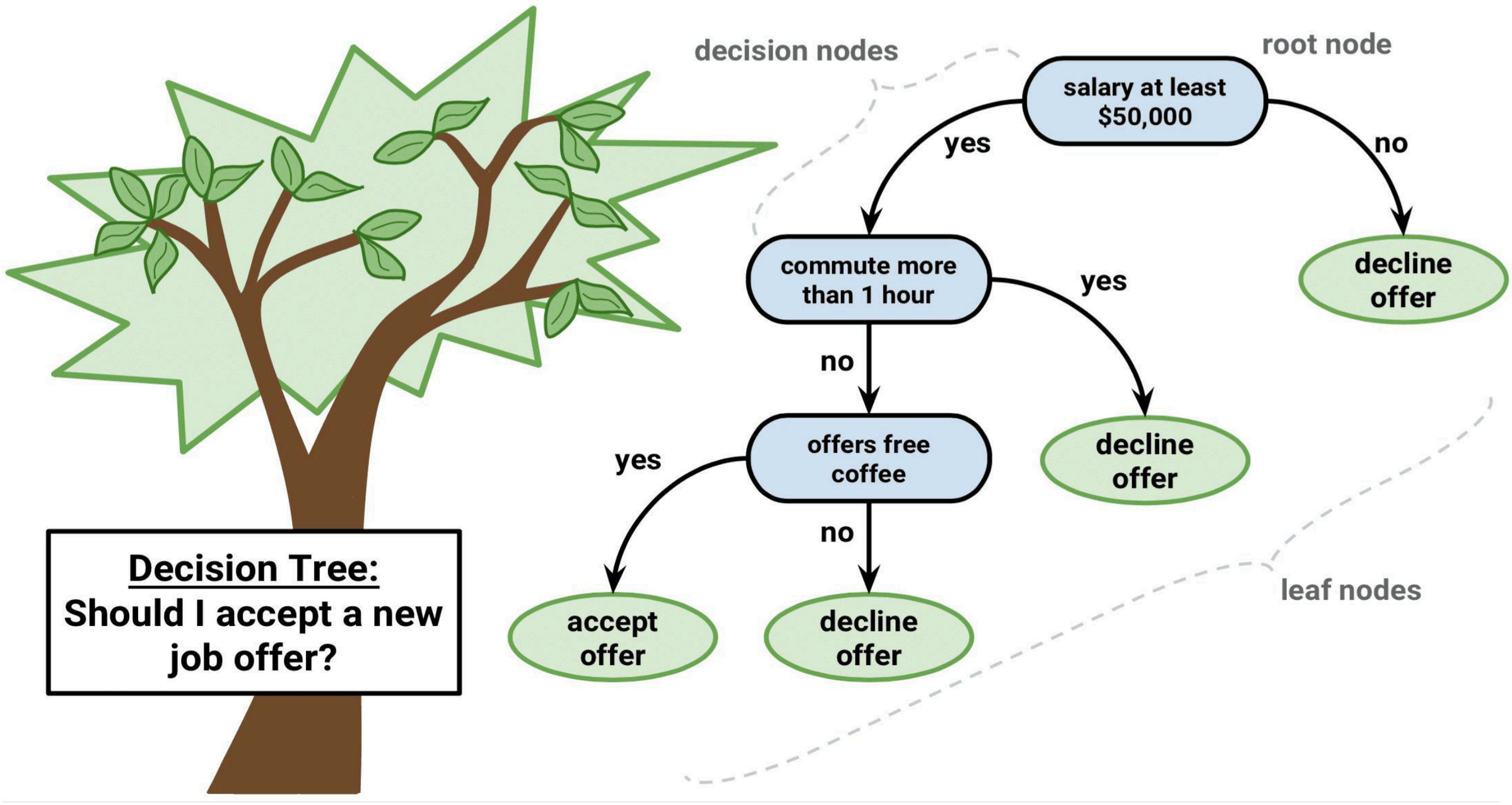
Decision Trees are versatile Machine Learning algorithms that can perform

- Classification Tasks
- Regression Tasks
- And even multi-output tasks

They are very powerful algorithms, capable of fitting complex datasets. Also used in Random Forest (remember our project?)

Decision Trees

Example of a Decision Tree



Decision Trees

What we'll learn in this session ?

- Working with **Decision Trees**
  - Train
  - Visualize
  - Make prediction with Decision tree
- The **CART** training algorithm used by **Scikit-Learn**
- How to **regularize** trees and use them for regression tasks
- Regression with Decision Trees
- Limitations of Decision Trees

Decision Trees

Training and Visualizing a Decision Tree

Let's just build a Decision Tree and take a look at how it makes predictions

We'll train a DecisionTreeClassifier using Scikit Learn on the famous **Iris dataset**.

And then we'll see how it works.

Decision Trees

Iris Dataset - Training and Visualizing a Decision Tree

The iris dataset consists of 4 features namely :

- Petal Length
- Petal Width
- Sepal Length
- Sepal Width

There are three classes namely :

- Iris Setosa
- Iris Versicolor
- Iris Virginica

Decision Trees

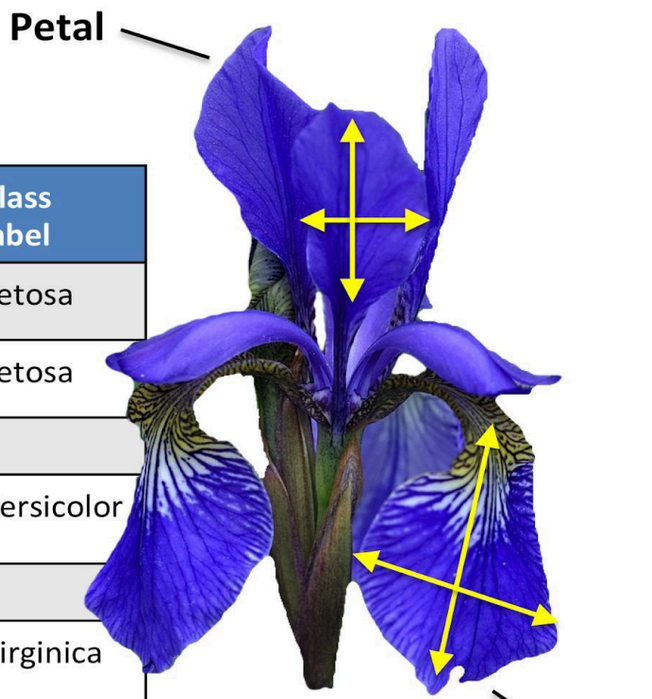
Iris Dataset - Training and Visualizing a Decision Tree

Samples
(instances, observations)

	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica

Features
(attributes, measurements, dimensions)

Class labels
(targets)



Decision Trees

Training and Visualizing a Decision Tree

The iris dataset has 4 features petal length, petal width, sepal length and sepal width.

But here we'll only use two features i.e. petal length and petal width.

Decision Trees

Training and Visualizing a Decision Tree

We will follow the following steps to train and visualize our decision tree

1. Load the Iris dataset using Scikit Learn
2. Select only the Petal length and Petal width features
3. Train our Decision Tree classifier on the Iris Dataset
4. Visualize our Decision Tree using export_graphviz()
5. export_graphviz() gives us a file in .dot format which we will convert to png using the dot command line tool

Decision Trees

Training and Visualizing a Decision Tree

1. Load the Iris dataset using Scikit Learn

```
>>> from sklearn.datasets import load_iris
>>> iris = load_iris()
```

Run it in jupyter notebook

Decision Trees

Training and Visualizing a Decision Tree

2. Select only the Petal length and Petal width features

```
>>> X = iris.data[:, 2:]  # petal length and width
>>> y = iris.target
```

Run it in jupyter notebook

Decision Trees

Training and Visualizing a Decision Tree

3. Train our Decision Tree classifier on the Iris Dataset

```
>>> from sklearn.tree import DecisionTreeClassifier
>>> tree_clf = DecisionTreeClassifier(max_depth=2)
>>> tree_clf.fit(X, y)
```

Run it in jupyter notebook

Decision Trees

Training and Visualizing a Decision Tree

4. Visualize our Decision Tree using export_graphviz()

We can visualize the trained decision tree using the **export_graphviz()** method.

```
>>> from sklearn.tree import export_graphviz
>>> export_graphviz( tree_clf,
    out_file=image_path("iris_tree.dot"),
    feature_names=iris.feature_names[2:],
    class_names=iris.target_names, rounded=True,
    filled=True )
```

Decision Trees

Training and Visualizing a Decision Tree

5. Converting to Png file

Then you can convert this .dot file to a variety of formats such as PDF or PNG using the **dot command- line tool** from the graphviz package.

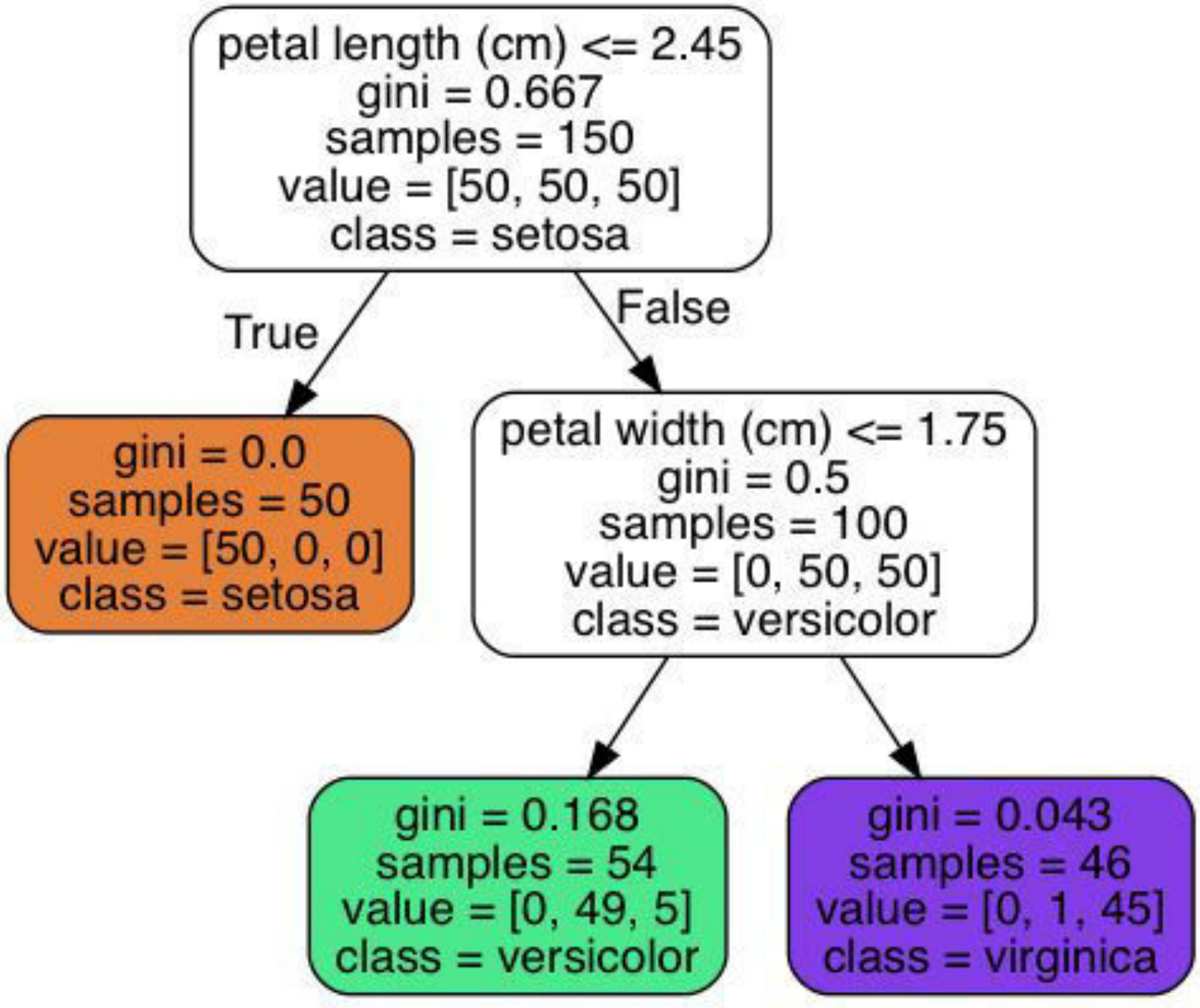
This command line converts the .dot file to a .png image file:

```
>>> dot -Tpng iris_tree.dot -o iris_tree.png
```

Run it in the console

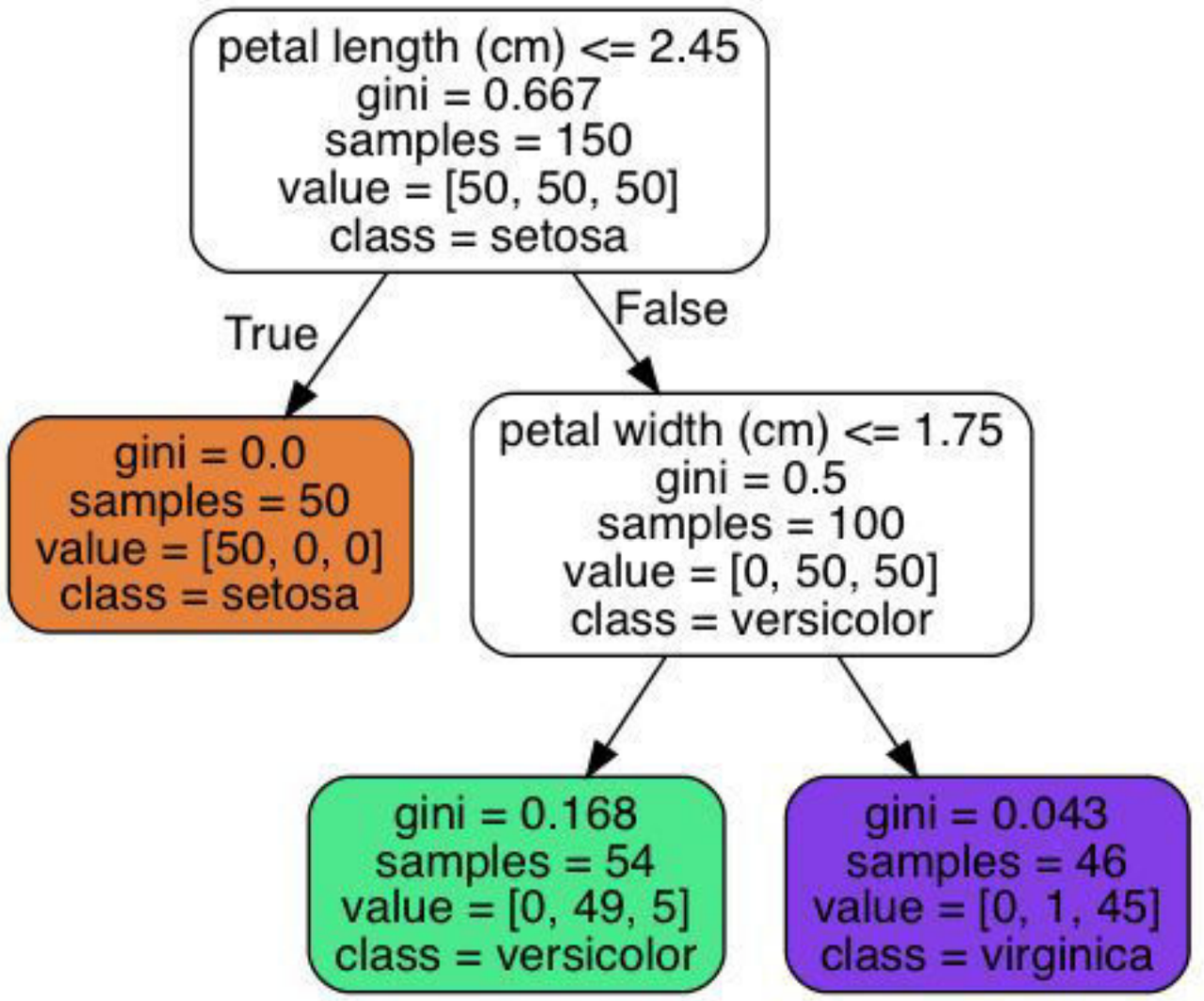
Decision Trees

Training and Visualizing a Decision Tree



Decision Trees

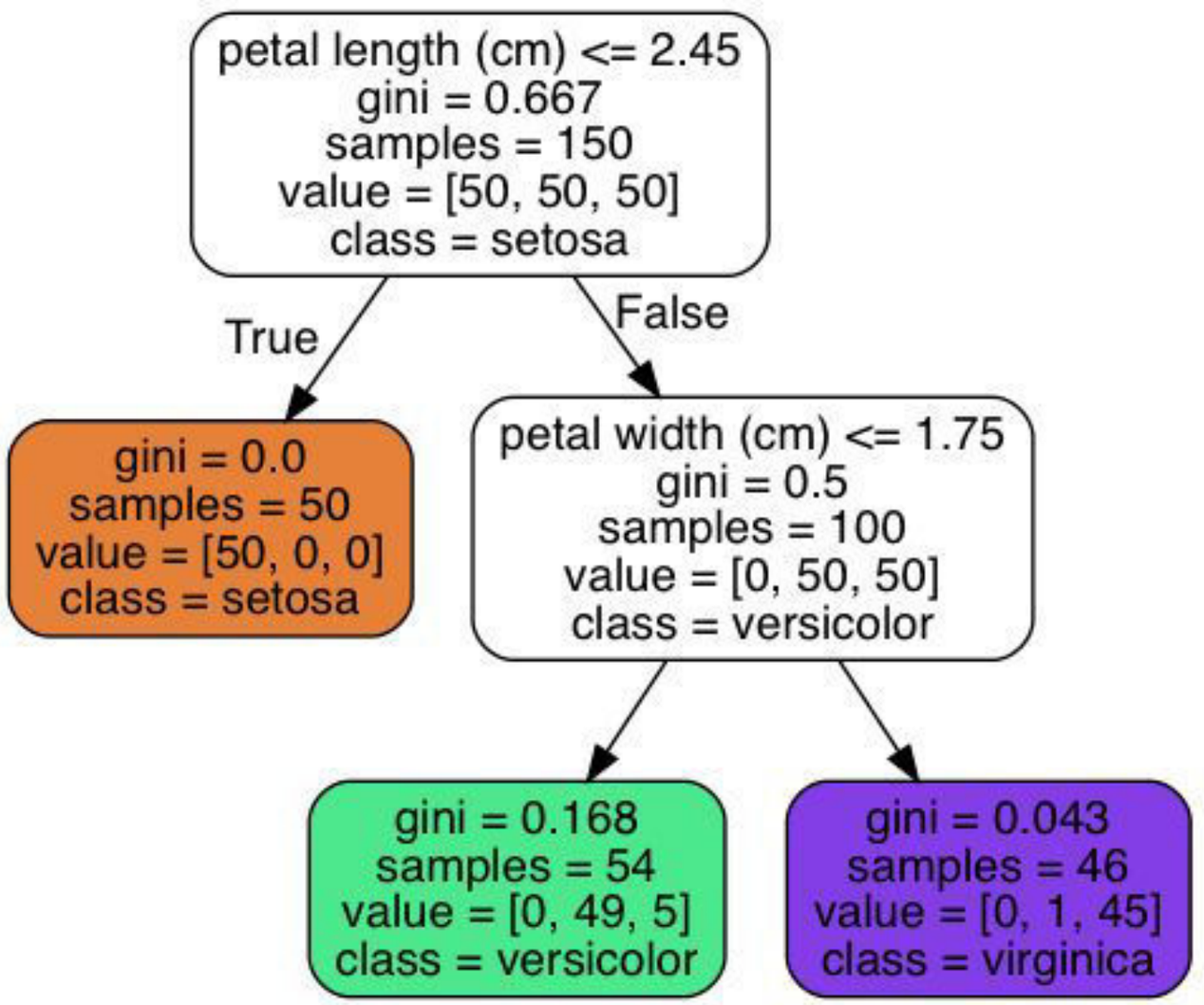
Understanding the Decision Tree



A **node's 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.

Decision Trees

Understanding the Decision Tree

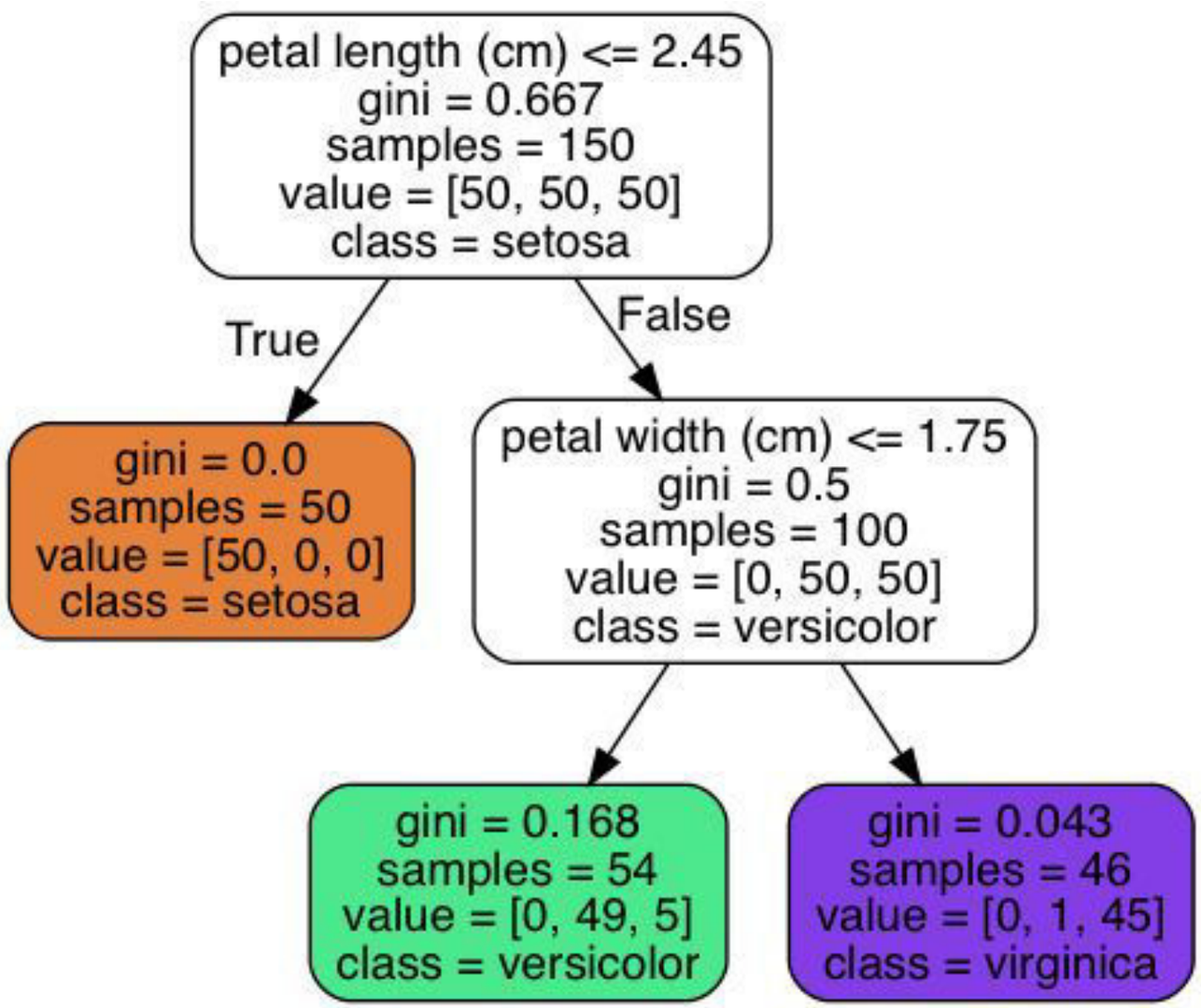


A **node's gini attribute** measures its impurity: a node is "pure" (gini=0) if all training instances it applies to belong to the same class.

For example, since the depth-1 left node applies only to Iris-Setosa training instances, it is pure and its gini score is 0.

Decision Trees

Making Predictions

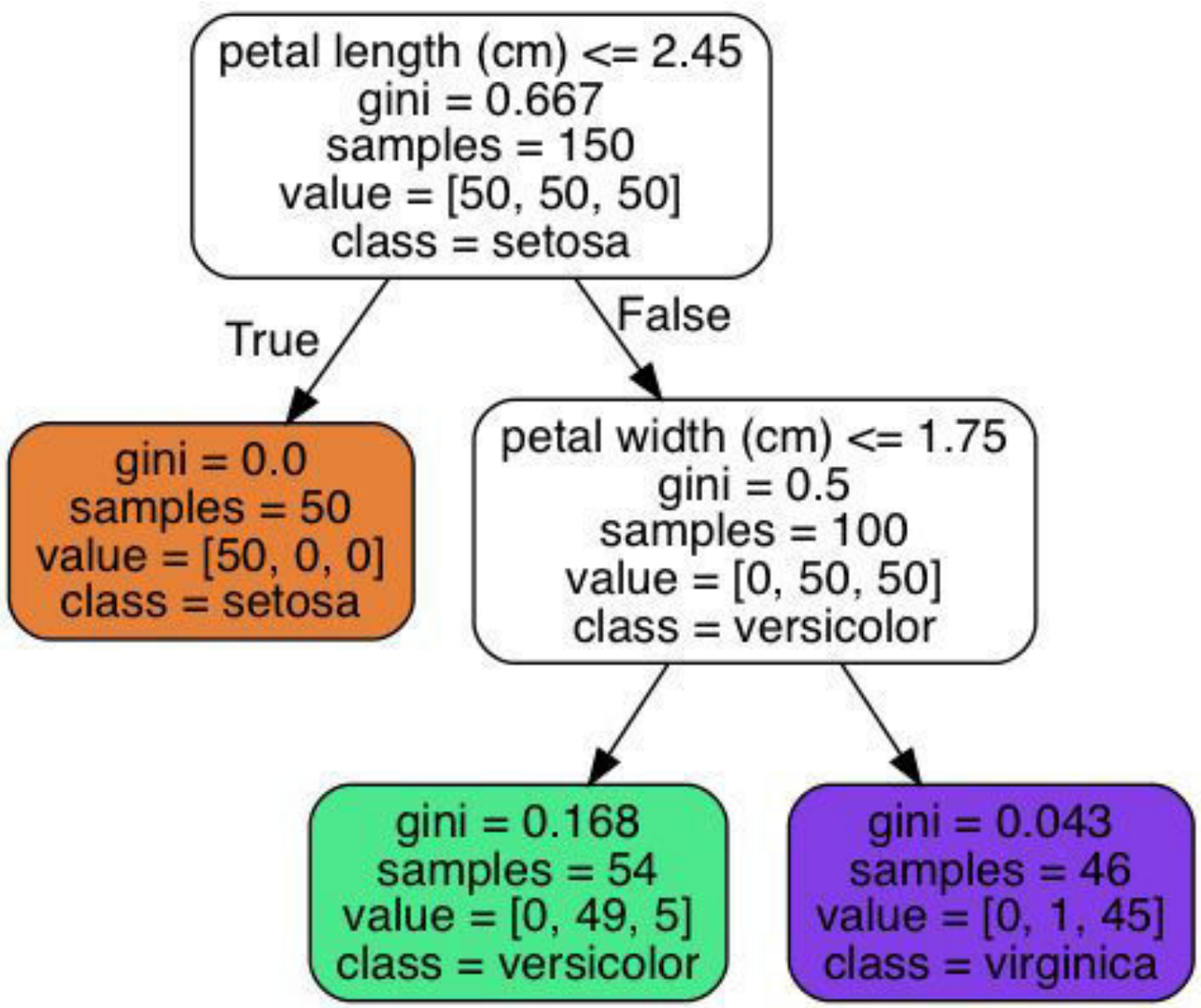


To make a prediction the decision classifier follows these steps :

- Start at the root node (depth 0, at the top), this node asks whether the flower's petal length is smaller than or equal to 2.45 cm:

Decision Trees

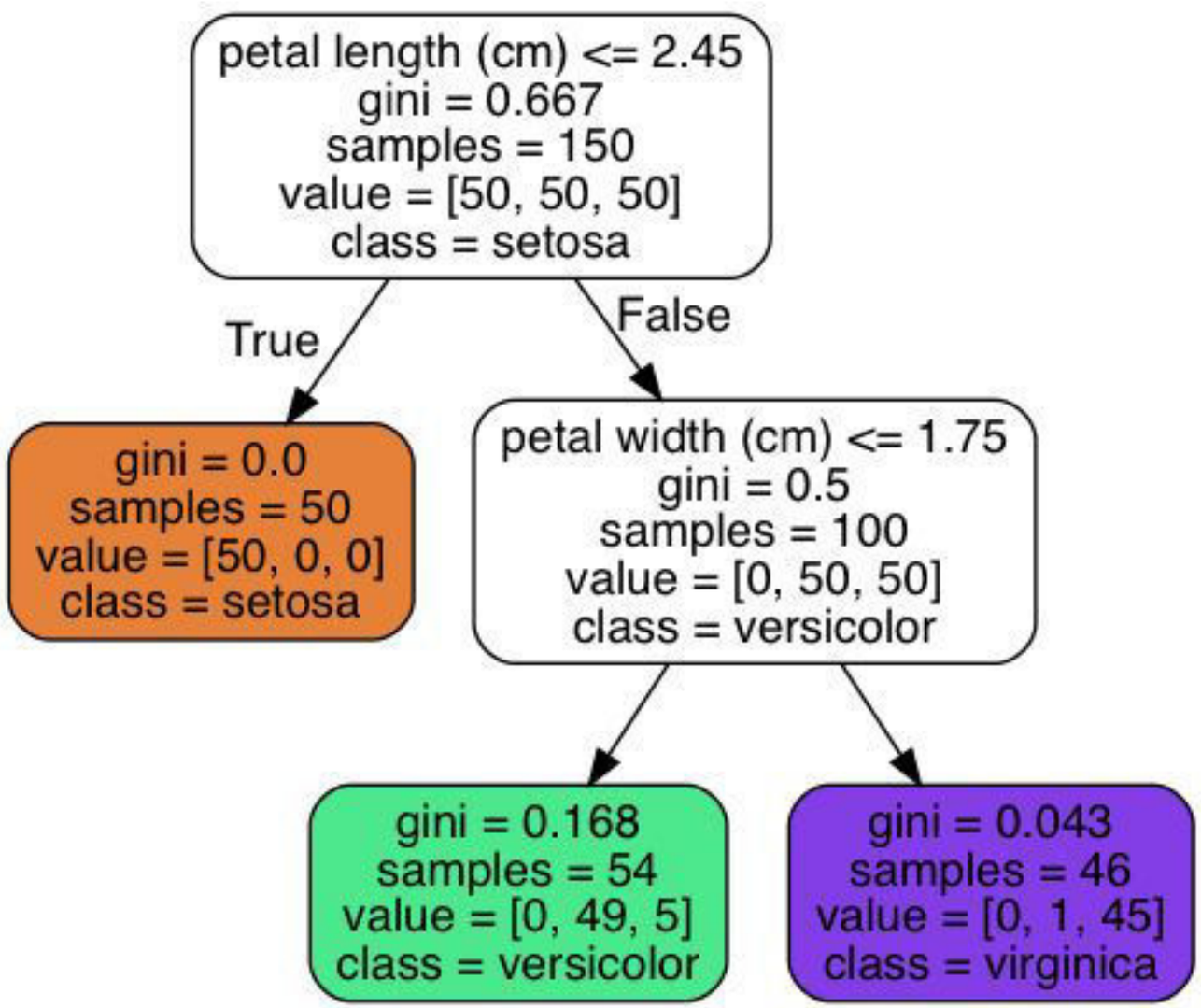
Making Predictions



- If it is, then you move down to the root's left child node (depth 1, left). In this case it is a leaf node hence the flower is predicted as setosa.

Decision Trees

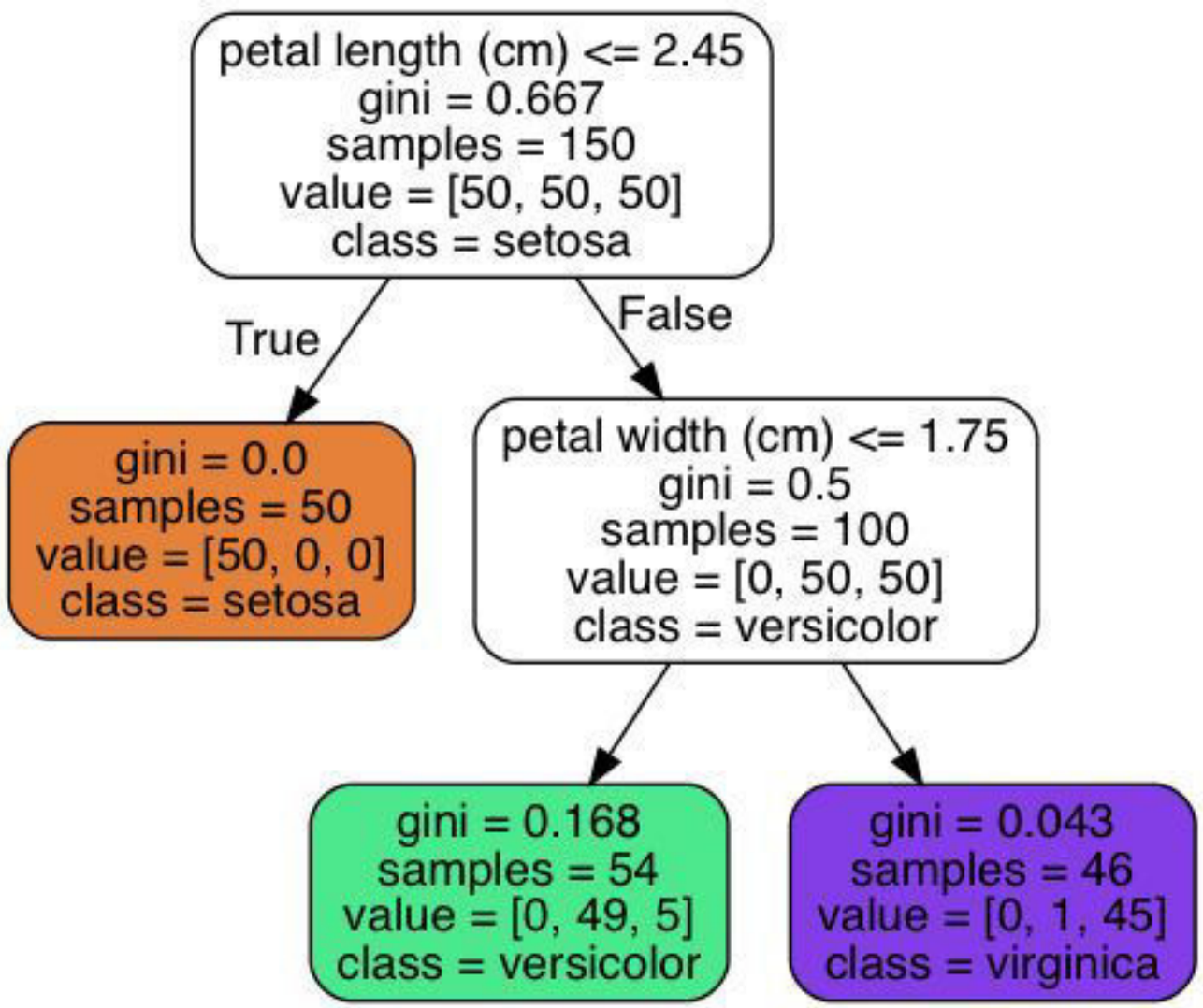
Making Predictions



- If it is not, then you move down to the root's right child node (depth 1, right), since it is not a leaf node it asks further questions as, is the petal width smaller than or equal to 1.75 cm?

Decision Trees

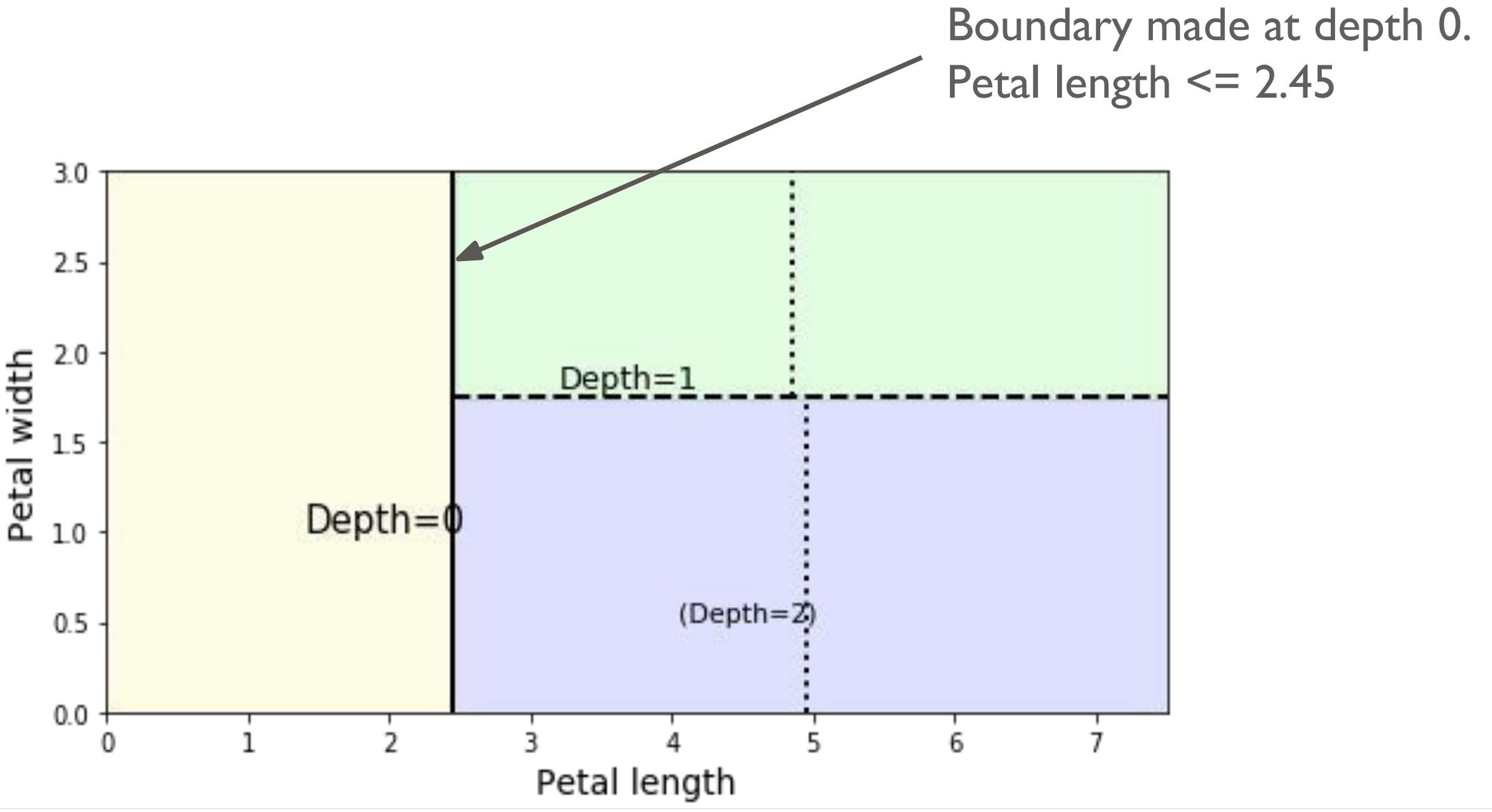
Making Predictions



- If it is, then your flower is most likely an Iris- Versicolor (depth 2, left).
- If it is not, If not, it is likely an Iris-Virginica (depth 2, right).

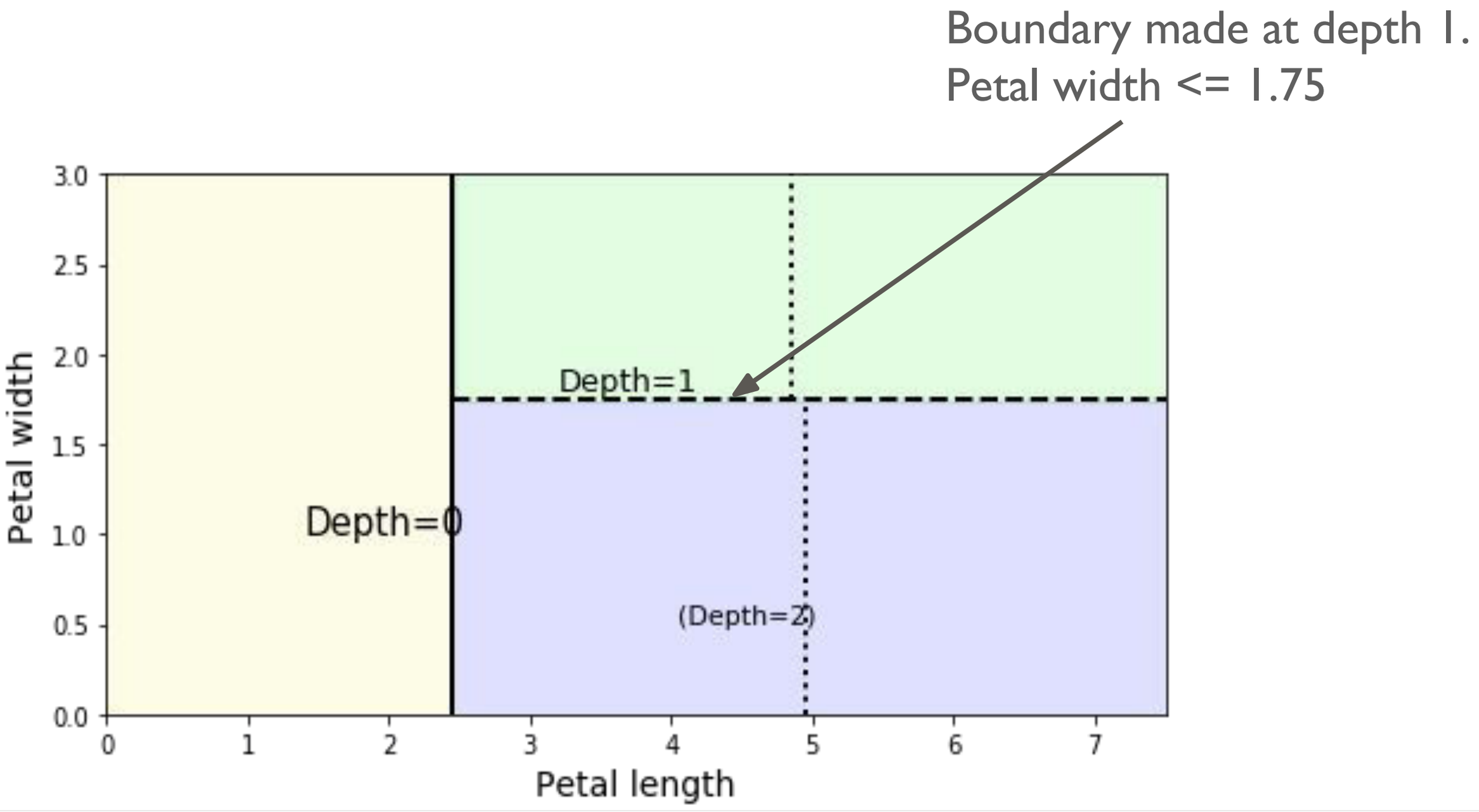
Decision Trees

Decision Tree's decision boundaries.



Decision Trees

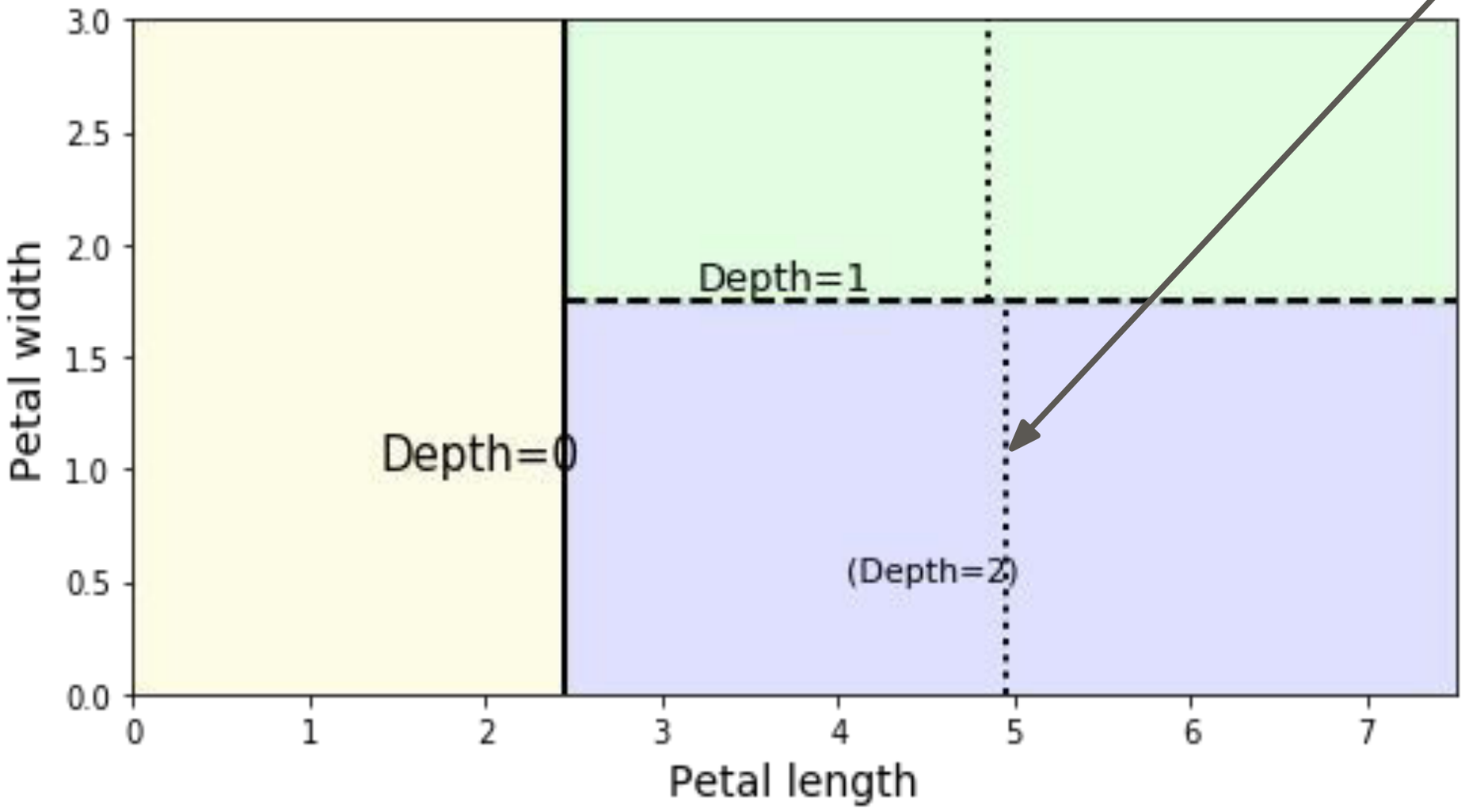
Decision Tree's decision boundaries.



Decision Trees

Decision Tree's decision boundaries.

This would have been the decision boundary to divide the tree further i.e. if max_depth was set to 3



Decision Trees

How is Gini Impurity Calculated?

A node's **gini attribute** measures its **impurity**: a node is

- "Pure" (gini=0) if all training instances it applies to belong to the same class.

A Gini coefficient of 1 expresses maximal inequality among the training samples.

Decision Trees

How is Gini Impurity Calculated?

The formula for finding the gini impurity score of a particular level is :

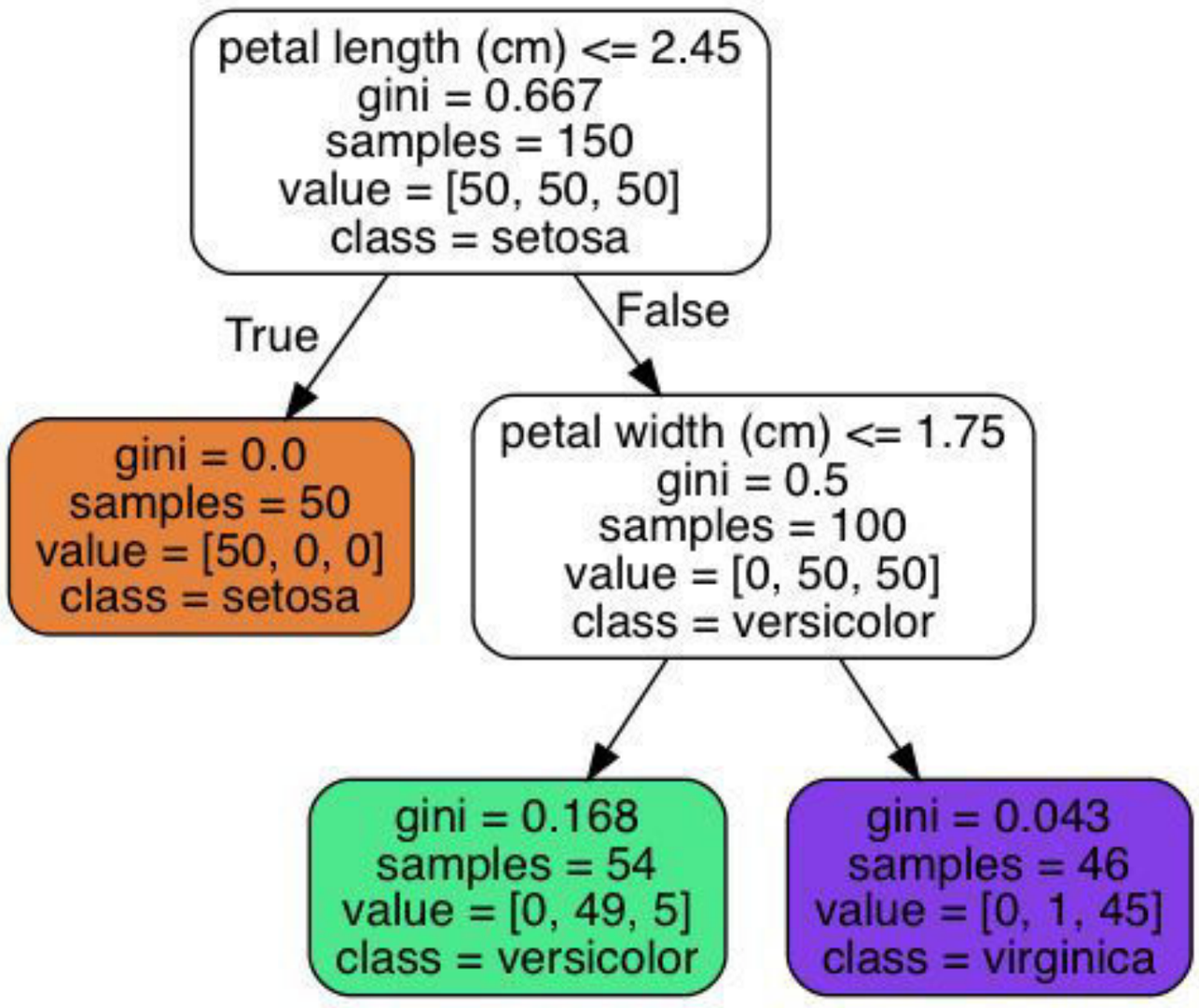
$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Here p_i is the ratio of class i in the node whose gini index is being calculated.

There are total of c classes.

Decision Trees

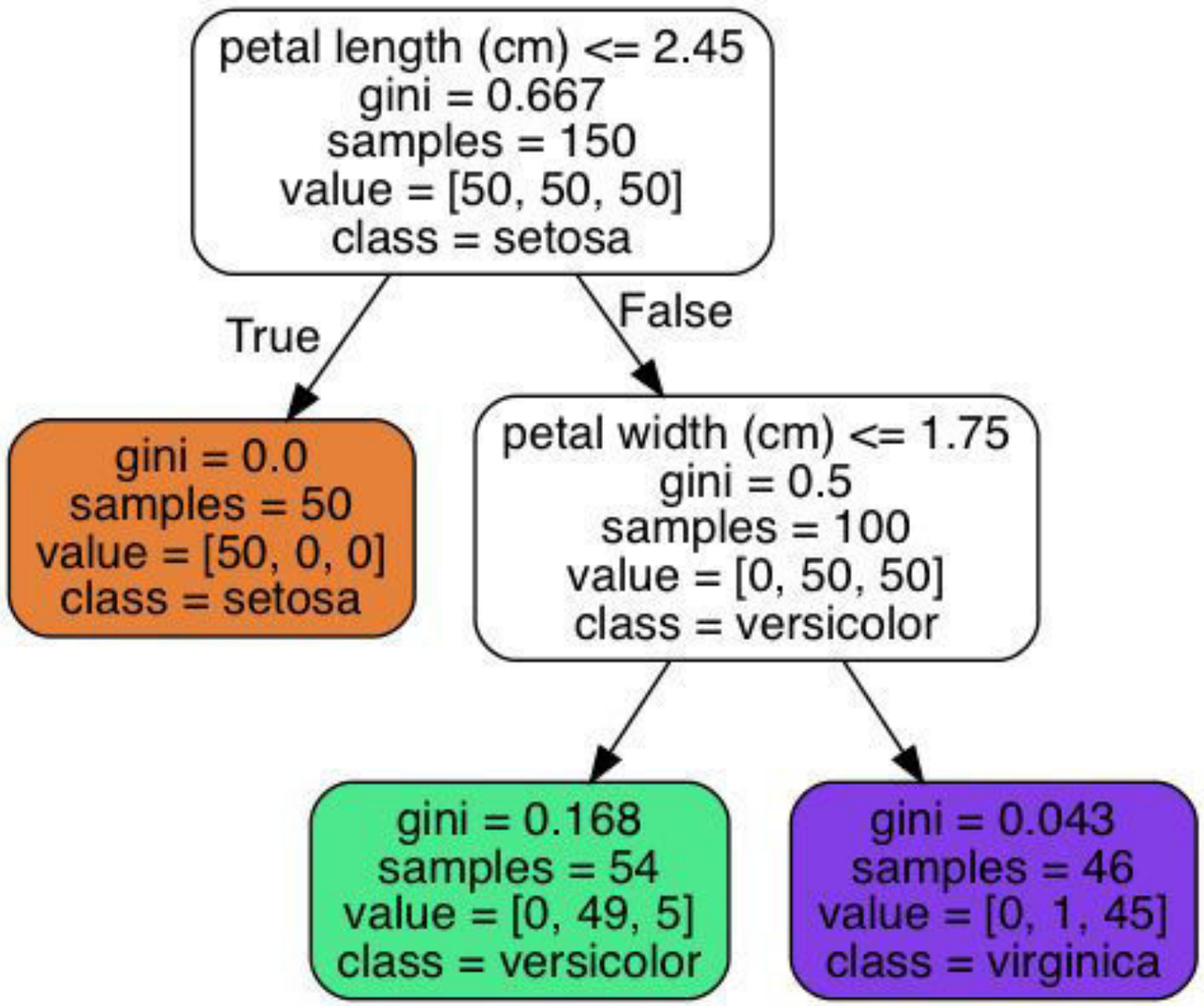
How is Gini Impurity Calculated?



For example, since the depth-1 left node applies only to Iris-Setosa training instances, it is pure and its gini score is 0.

Decision Trees

How is Gini Impurity Calculated?



In the our example the depth-2 left node has a gini score equal to

$1 - (0/54)^2 - (49/54)^2 - (5/54)^2 \approx 0.168$

Decision Trees

Model Interpretation : White Box versus Black Box

A **White Box Model** is :

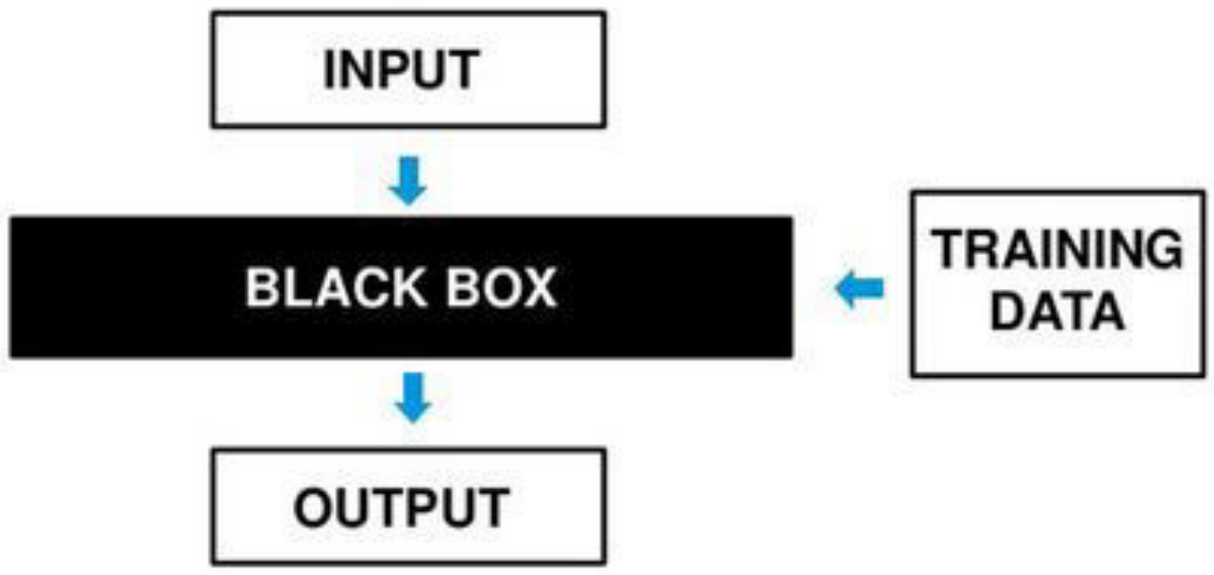
- Fairly intuitive
- Their decisions are easy to interpret
- Eg. Decision Trees

Decision Trees

Model Interpretation : White Box versus Black Box

A **Black Box Model** is :

- They make great predictions
- Easy to check the calculations that were performed to make these predictions



Decision Trees

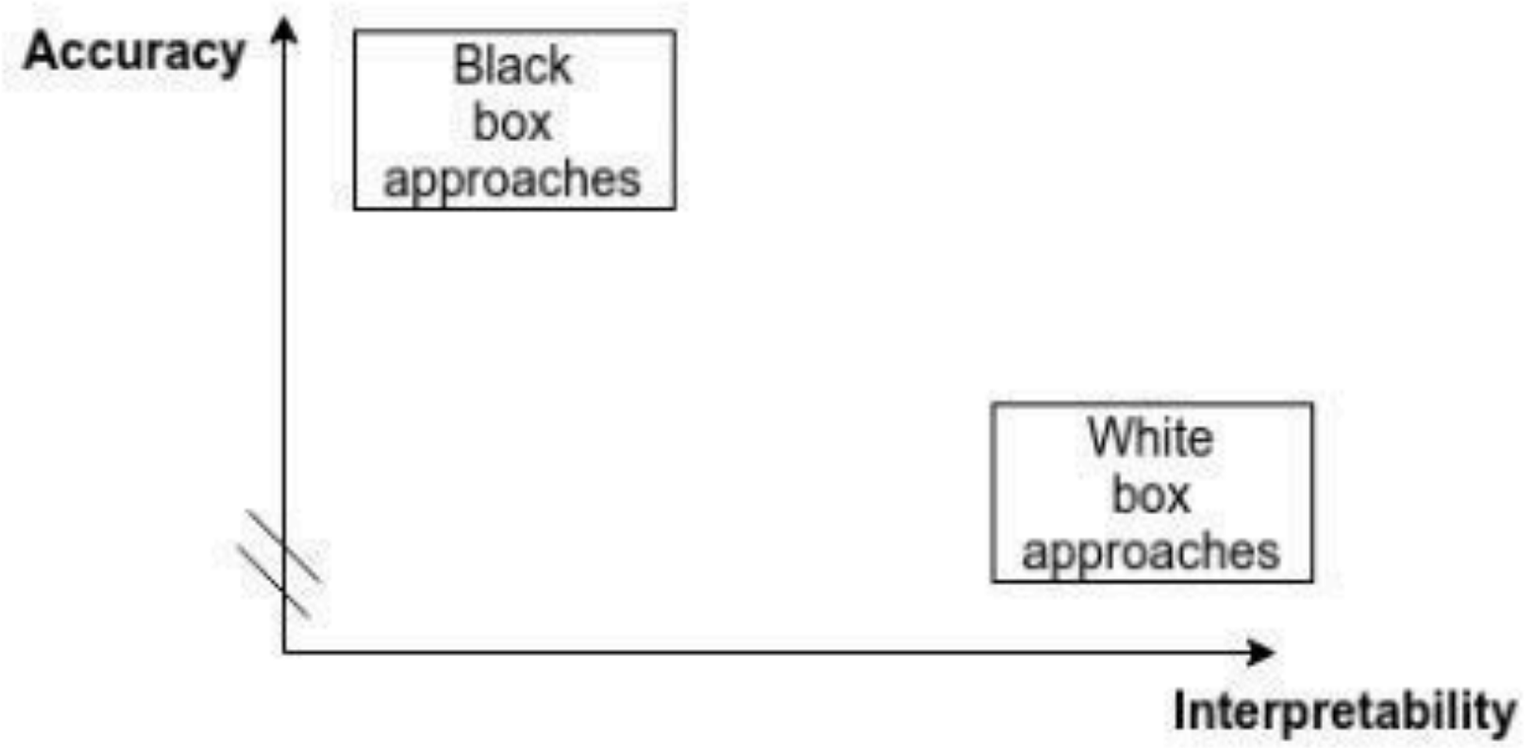
Model Interpretation : White Box versus Black Box

A **Black Box Model** is :

- It is usually hard to explain in simple terms why the predictions were made
- Eg. Random Forests , Neural networks

Decision Trees

Model Interpretation : White Box versus Black Box



Decision Trees

Estimating Class Probabilities

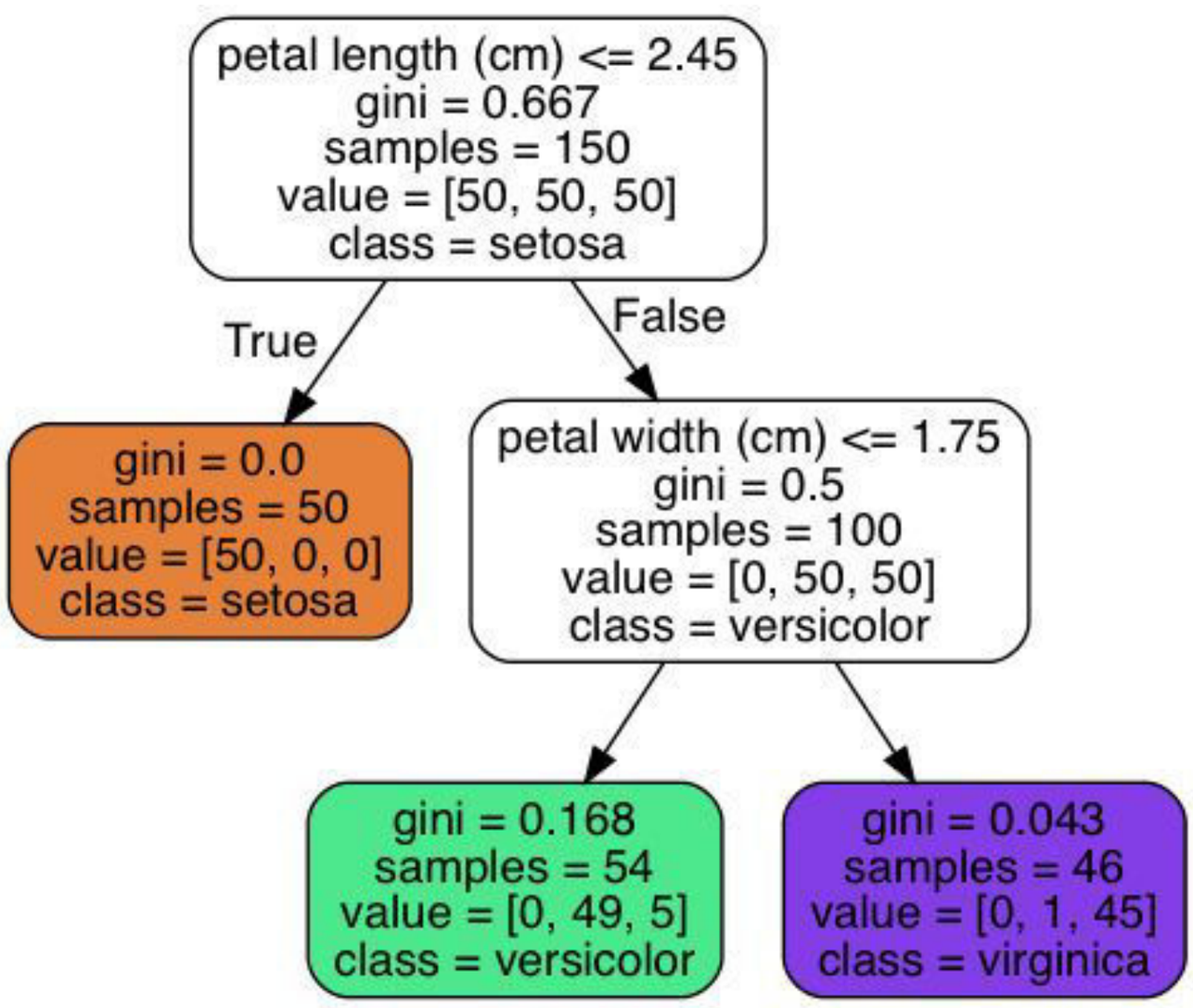
A Decision Tree can also estimate the probability that an instance belongs to a particular class k.

To do this it follows the following steps:

- First it traverses the tree to find the leaf node for this instance
- Then it returns the ratio of training instances of class k in this node.

Decision Trees

Estimating Class Probabilities

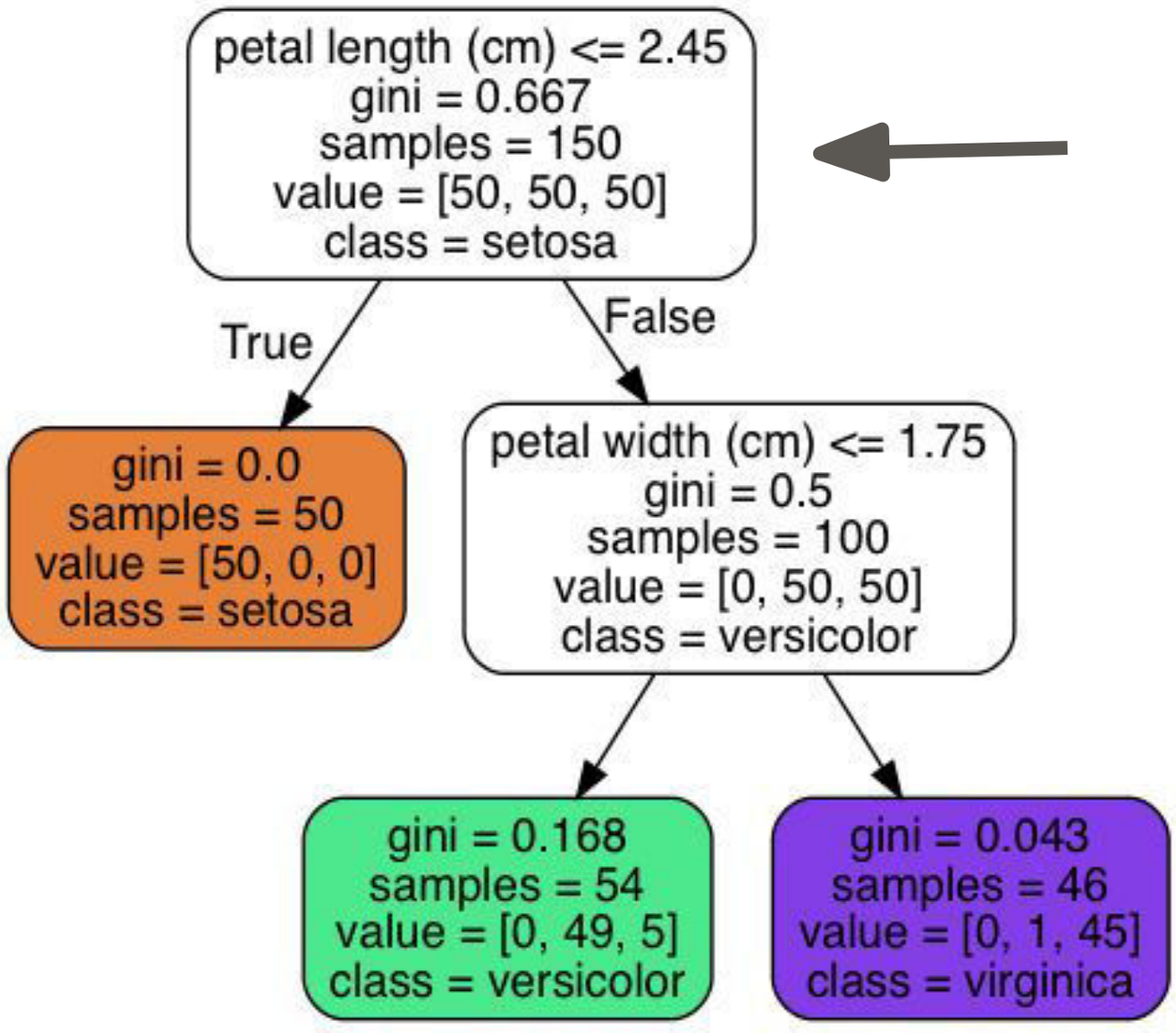


Suppose you have found a flower whose petals are

- 5 cm long and
- 1.5 cm wide

Decision Trees

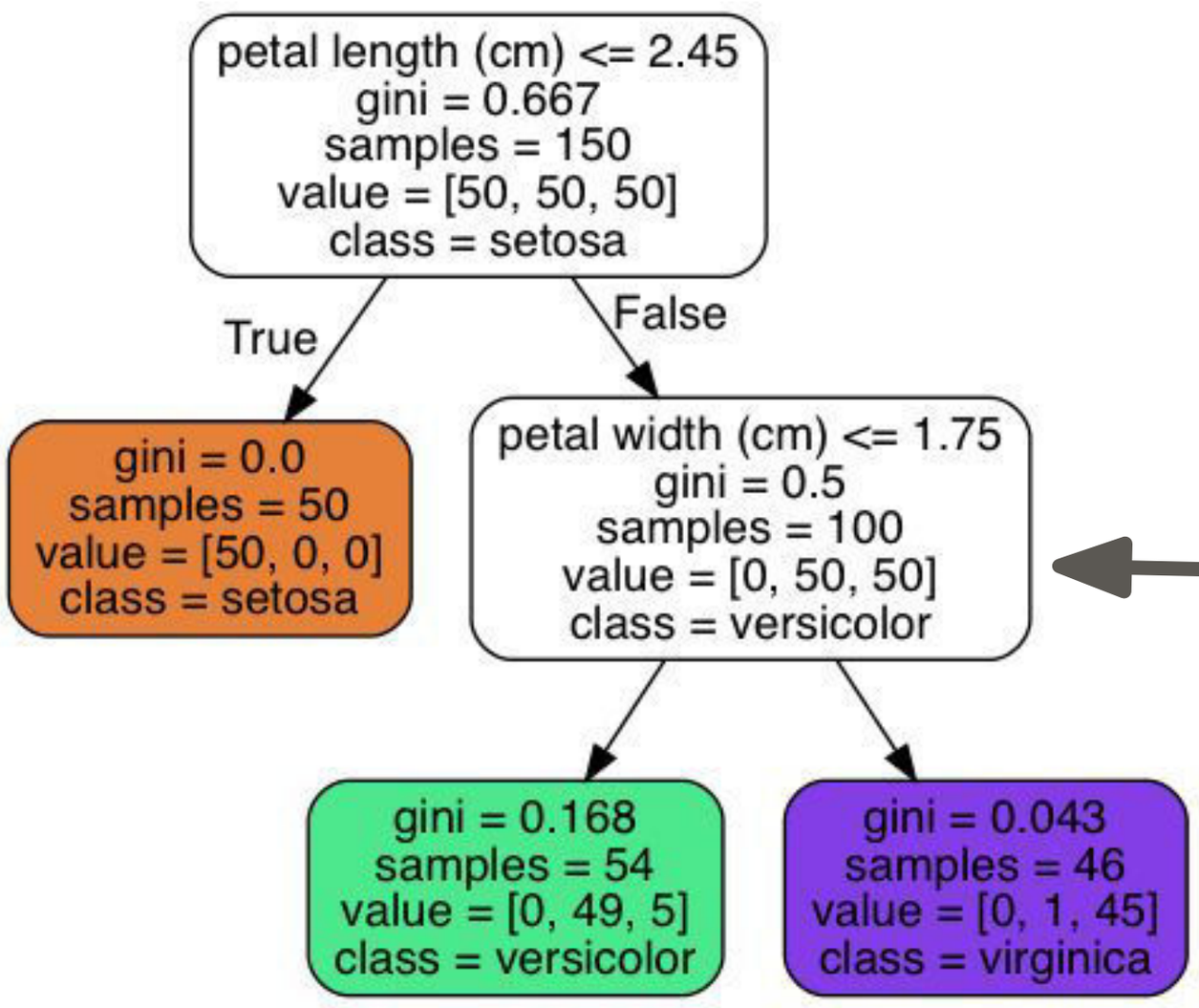
Estimating Class Probabilities



It first asks the question whether petal length <= 2.45. Since the condition is false it will go to the right subset

Decision Trees

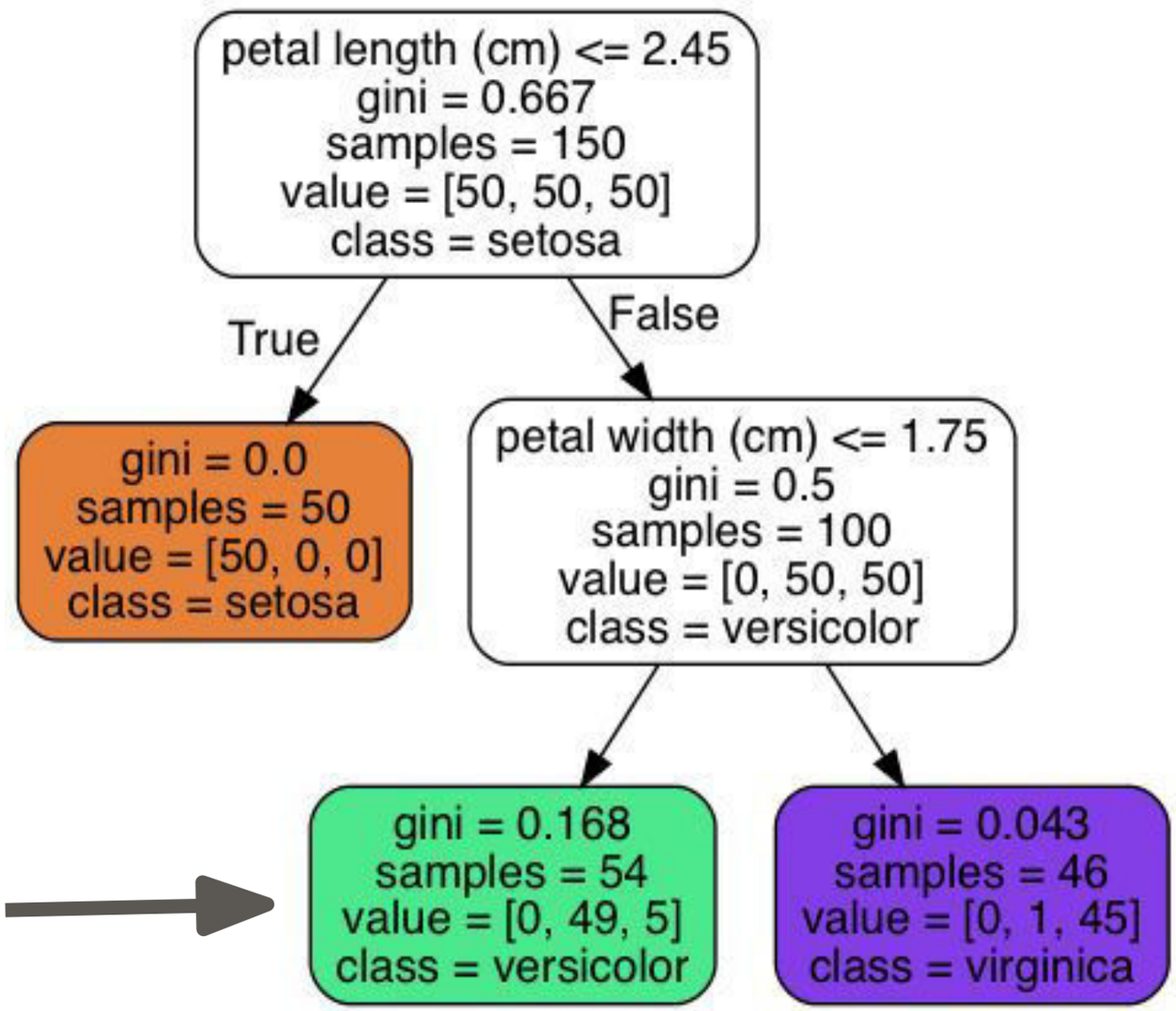
Estimating Class Probabilities



It will then ask if petal width <= 1.75. Since the condition is true it will go the left subset.

Decision Trees

Estimating Class Probabilities



Decision Tree should output the following probabilities:

- 0% for Iris-Setosa (0/54)
- 90.7% for Iris-Versicolor (49/54)
- 9.3% for Iris- Virginica (5/54)

Decision Trees

Estimating Class Probabilities

If we ask it to predict the class, it should output Iris-Versicolor (class 1) since it has the highest probability.

Let's verify it:

```
>>> tree_clf.predict_proba([[5, 1.5]])
array([[ 0. , 0.90740741, 0.09259259]])
>>> tree_clf.predict([[5, 1.5]])
array([1])
```

Run it in jupyter notebook

Decision Trees

The CART Training Algorithm

Scikit-Learn uses the Classification And Regression Tree (CART) algorithm to train Decision Trees.

The idea is really quite simple:

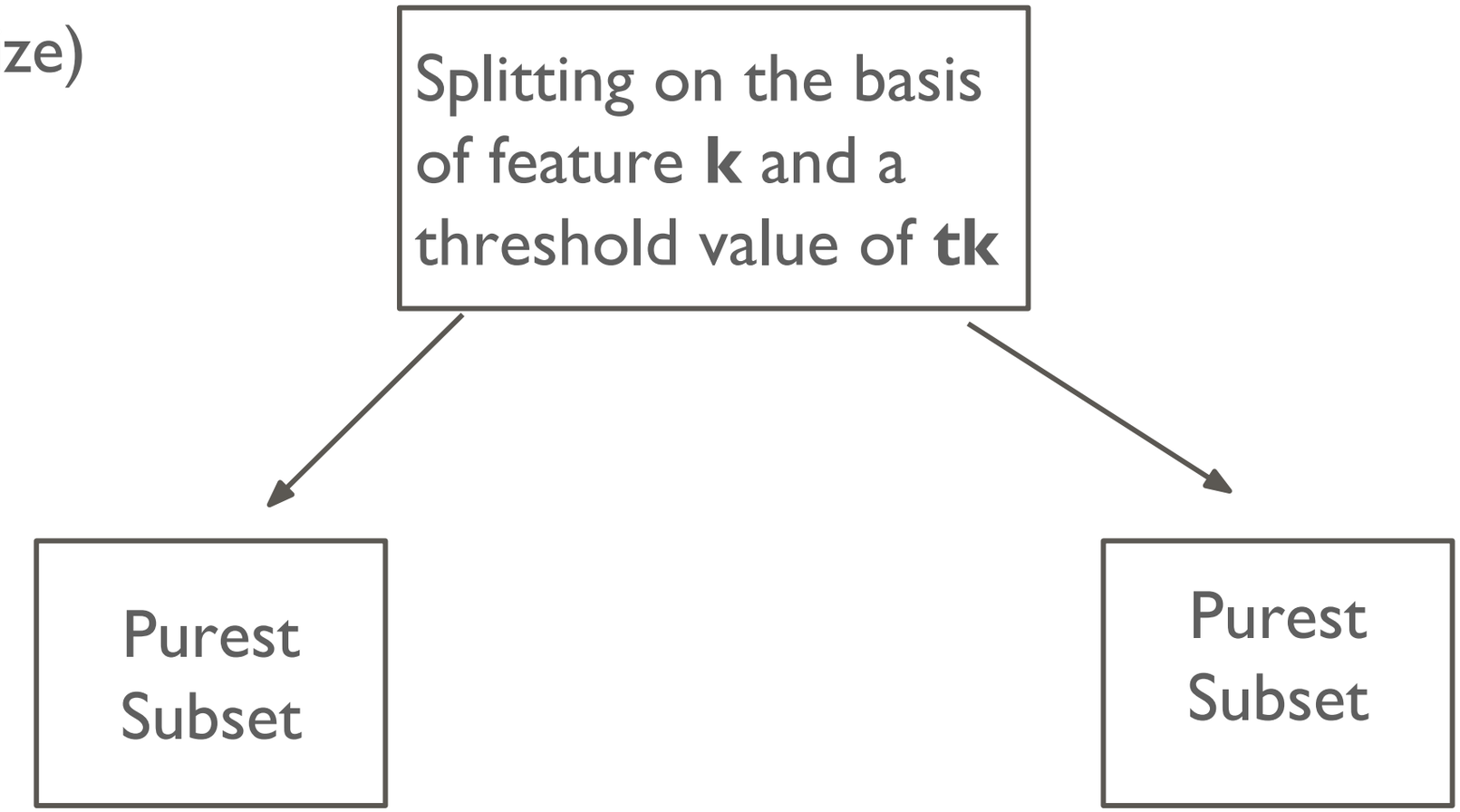
- The algorithm first splits the training set in two subsets using a single feature k and a threshold tk

Decision Trees

The CART Training Algorithm

How does it choose k and tk???

It searches for the pair (k, tk) that produces the purest subsets (weighted by their size)



Decision Trees

The CART Training Algorithm

The cost function that the algorithm tries to minimize is given by the equation :

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

where $\begin{cases} G_{\text{left/right}} \text{ measures the impurity of the left/right subset,} \\ m_{\text{left/right}} \text{ is the number of instances in the left/right subset.} \end{cases}$

Decision Trees

The CART Training Algorithm

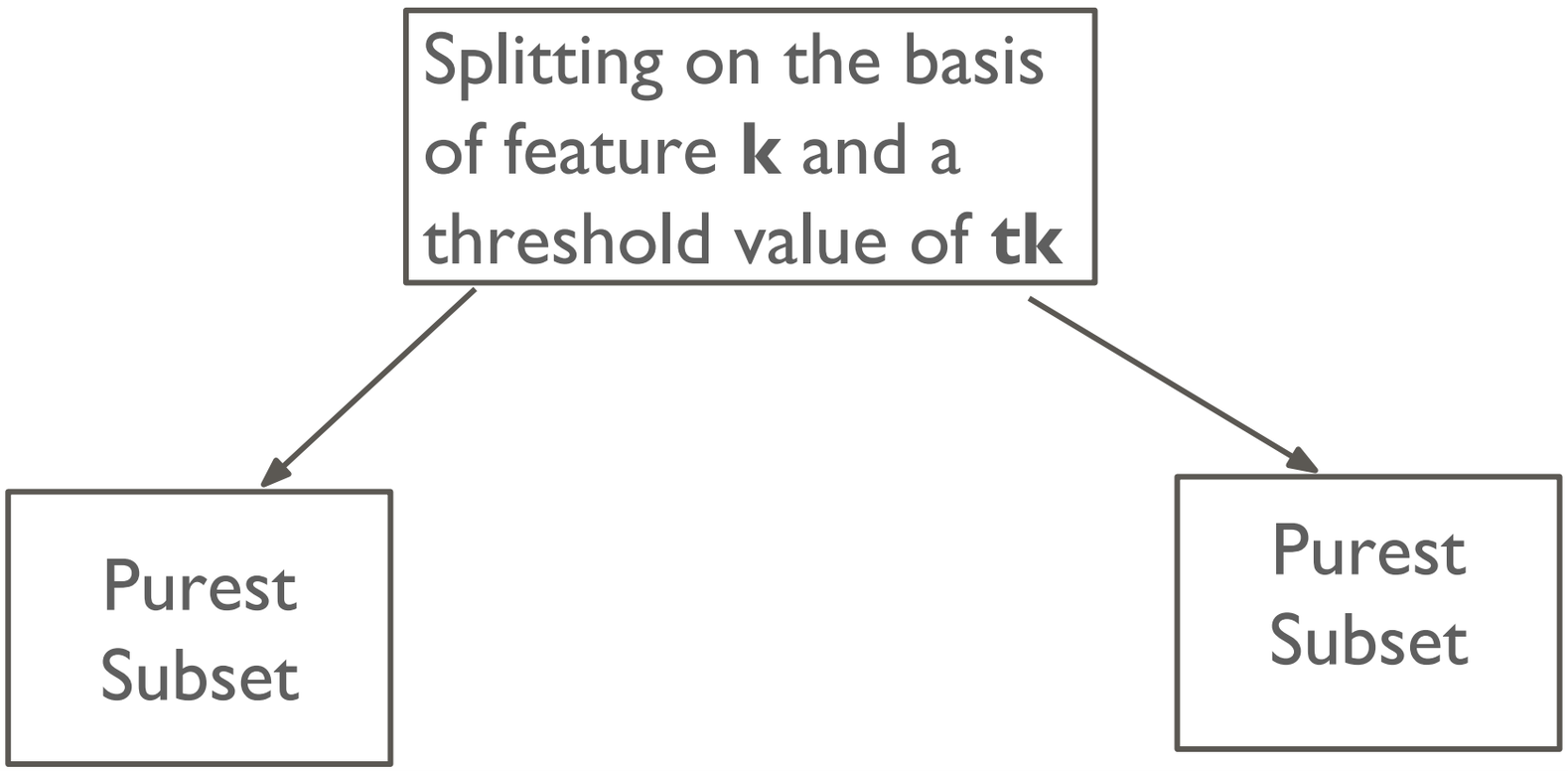
- Once it has successfully split the training set in two, it splits the subsets using the same logic, then the sub- subsets and so on, recursively

- It stops recursion once it reaches the maximum depth (defined by the max_depth hyperparameter), or if it cannot find a split that will reduce impurity.

Decision Trees

Important points on the CART Training Algorithm

- It is a greedy algorithm as it greedily searches for an optimum split at the top level
- Then repeats the process at each level.



Decision Trees

Important points on the CART Training Algorithm

- 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 reasonably good solution, but it is not guaranteed to be the optimal solution

Decision Trees

Computational Complexity of Decision Trees

Complexity of Prediction :

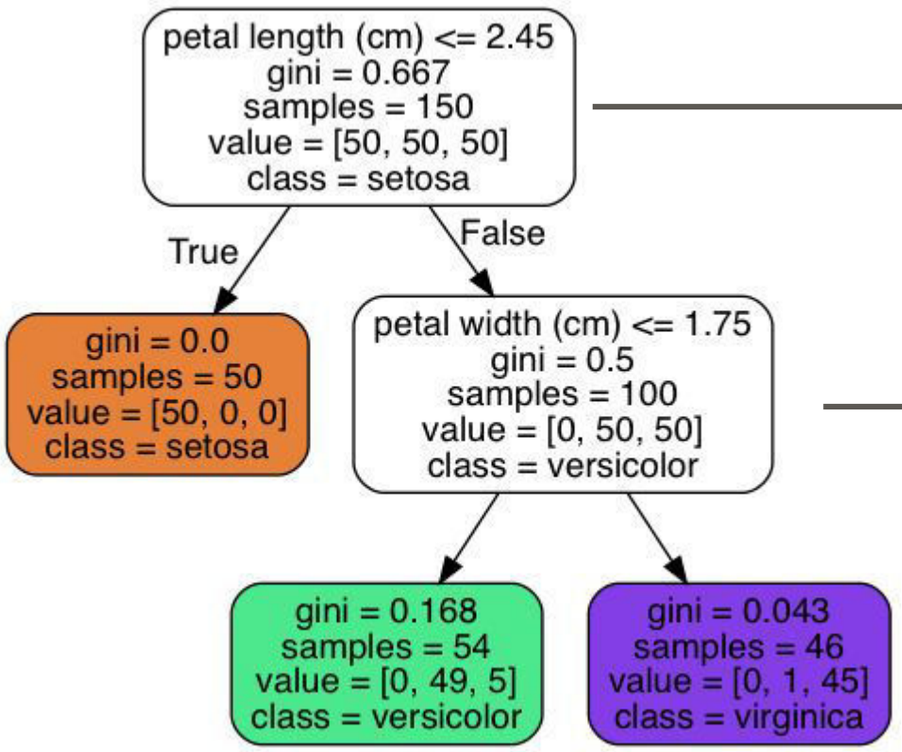
- Making predictions requires traversing the Decision Tree from the root to a leaf.
- Decision Trees are generally approximately balanced, so traversing the Decision Tree requires going through roughly $\mathbf{O(\log_2(m))}$ nodes, where **m** is total number of training instances.

Decision Trees

Computational Complexity of Decision Trees

<u>Complexity of Prediction</u> :

- Since each node only requires checking the value of one feature, the overall prediction complexity is just **$O(\log_2(m))$**



One feature checked i.e. petal length

One feature checked i.e. petal width

Decision Trees

Computational Complexity of Decision Trees

Complexity of Prediction :

- Hence, the complexity of prediction is independent of the number of features.

So predictions are **very fast**, even when dealing with **large training sets**.

Decision Trees

Computational Complexity of Decision Trees

Complexity of Training :

- The training algorithm compares **all features** (or less if max_features is set) on all samples at each node.
- This results in a training complexity of **O(n × m log(m))**, where n is the number of features, we have to compare all the n features at each of the m nodes.

Decision Trees

Computational Complexity of Decision Trees

Complexity of Training :

For small training sets (less than a few thousand instances),

Scikit-Learn can speed up training by presorting the data (set presort=True),

but this slows down training considerably for larger training sets.

Decision Trees

Which measure to use ? Gini Impurity or Entropy?

By default, the Gini impurity measure is used, but you can select the entropy impurity measure instead by setting the criterion **hyperparameter** to **"entropy"**.

Entropy measures the degree of randomness

Decision Trees

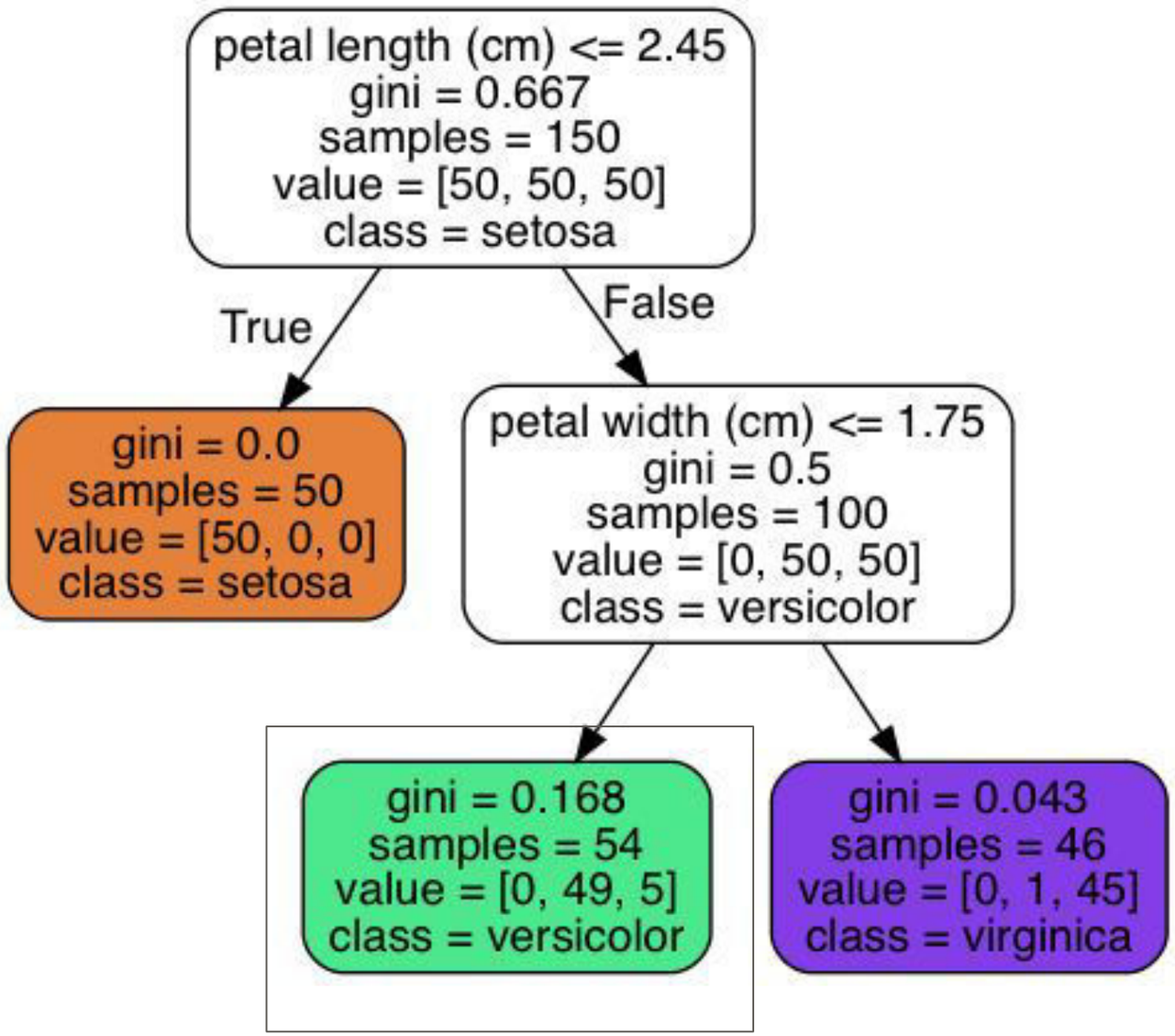
Which measure to use ? Gini Impurity or Entropy?

The formula for measuring entropy is

$$H_i = -\sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^{n} p_{i,k} \log\left(p_{i,k}\right)$$

Decision Trees

Which measure to use ? Gini Impurity or Entropy?



The entropy for the depth-2 left node is

$$-\frac{49}{54}\log\left(\frac{49}{54}\right)-\frac{5}{54}\log\left(\frac{5}{54}\right)$$

≈ 0.31

Decision Trees

Which measure to use ? Gini Impurity or Entropy?

The truth is, most of the time it does not make a big difference: they lead to similar trees.

- **Gini impurity is slightly faster** to compute, so it is a good default.
- However, Gini impurity tends to isolate the most frequent class in its own branch of the tree
- While entropy tends to produce **slightly more balanced trees**.

Decision Trees

Regularization Hyperparameter

- If left unconstrained, the tree structure will adapt itself to the training data, fitting it very closely, and **most likely overfitting it**.
- To avoid overfitting the training data, you need to restrict the Decision Tree's freedom during training.

Decision Trees

Regularization Hyperparameter

- Putting restriction on the freedom of the model during training is called **regularization**. The regularization hyperparameters depend on the algorithm used, but generally you can at least restrict the maximum depth of the Decision Tree.

Decision Trees

Parametric and Non-parametric models

- Models like Decision Tree models are often called **nonparametric model** because the number of parameters is not determined prior to training, so the model structure is free to stick closely to the data.
- In contrast, a **parametric model** such as a linear model has a predetermined number of parameters, so its degree of freedom is limited, reducing the risk of overfitting but increasing the risk of underfitting.

Decision Trees

Regularization parameters for DecisionTreeClassifier class

- **max_depth** → restricts the maximum depth of the Decision Tree
- **min_samples_split** → the minimum number of samples a node must have before it can be split
- **min_samples_leaf** → the minimum number of samples a leaf node must have

Decision Trees

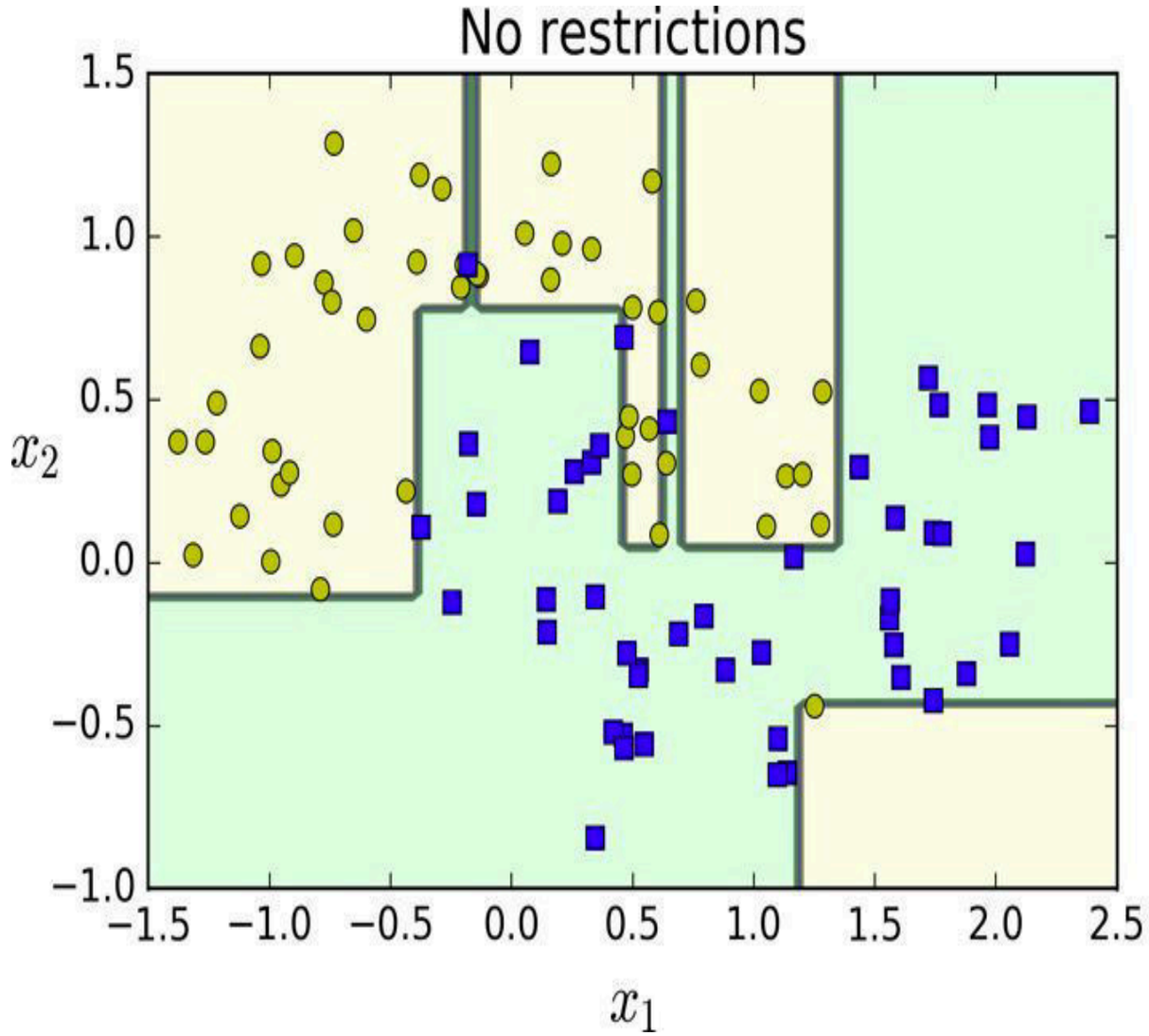
Regularization parameters for DecisionTreeClassifier class

- **min_weight_fraction_leaf** → same as min_samples_leaf but expressed as a fraction of the total number of weighted instances
- **max_leaf_nodes** → maximum number of leaf nodes
- **max_features** → maximum number of features that are evaluated for splitting at each node

Increasing min_* hyperparameters or **reducing max_* hyperparameters** will **regularize** the model.

Decision Trees

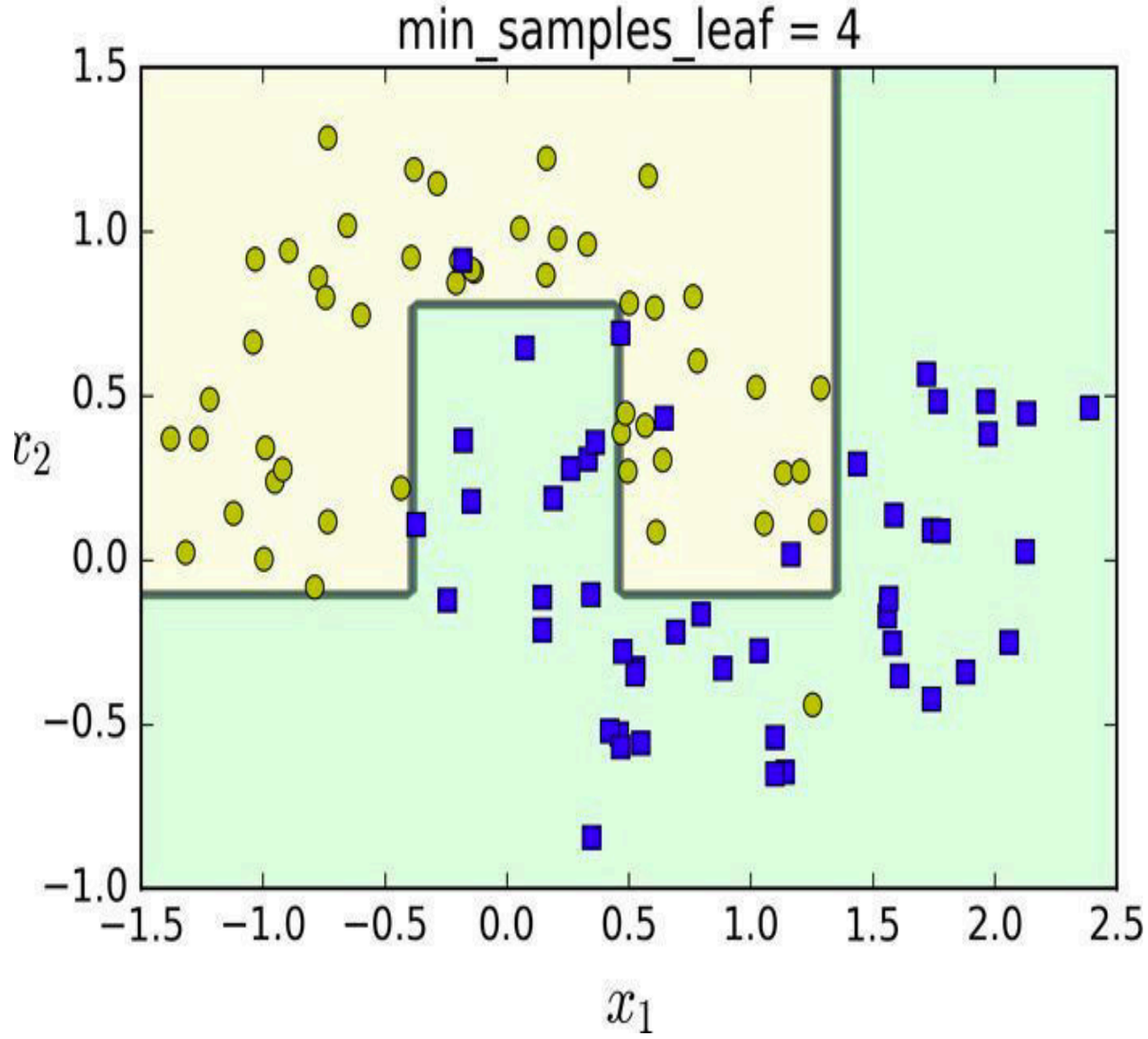
Regularization parameters for DecisionTreeClassifier class



The above Decision Tree is trained with the default hyperparameters i.e., no restrictions. This model is **Overfitting** the data.

Decision Trees

Regularization parameters for DecisionTreeClassifier class



The above Decision Tree is trained with min_samples_leaf=4. This model will probably **generalize** better.

Decision Trees

Regression with Decision Trees

Decision Trees are also capable of performing regression tasks. Let's build a regression tree using Scikit- Learn's DecisionTreeRegressor class, training it on a noisy quadratic dataset with max_depth=2

Decision Trees

Regression with Decision Trees

```
>>> from sklearn.tree import DecisionTreeRegressor
>>> tree_reg = DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X, y)
```

Run it on Notebook

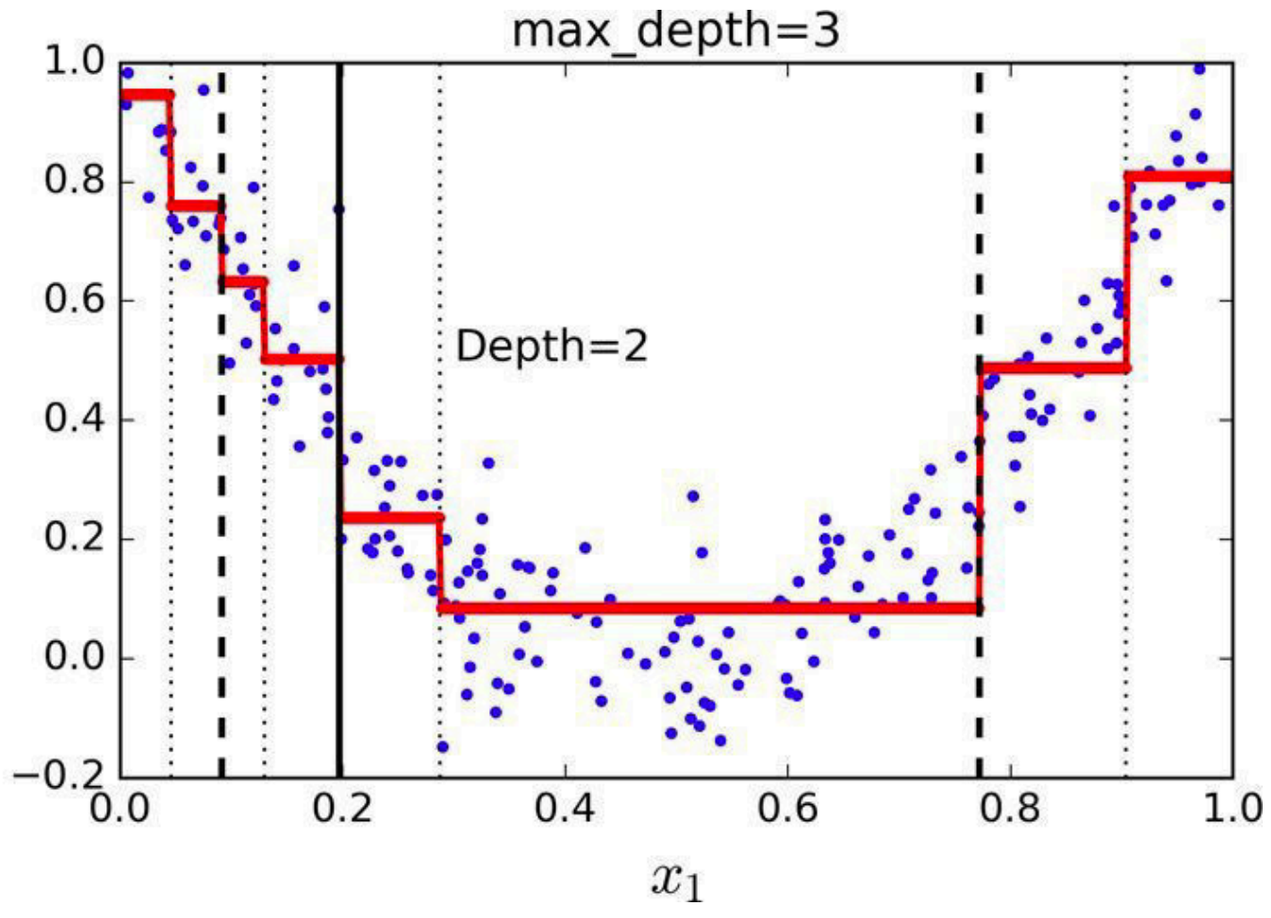
Decision Trees

Regression with Decision Trees

The main difference is that instead of predicting a class in each node, it predicts a value.

Decision Trees

Regression with Decision Trees



Notice how the predicted value for each region is always the average target value of the instances in that region.

Decision Trees

Regression with Decision Trees

The **CART algorithm** works the same way except instead of trying to split the training set in a way that **minimizes impurity**, it now tries to split the training set in a way that **minimizes the MSE**.

Decision Trees

Regression with Decision Trees

The formula for cost function for regression is -

$$J(k, t_k) = \frac{m_{\text{left}}}{m}\text{MSE}_{\text{left}} + \frac{m_{\text{right}}}{m}\text{MSE}_{\text{right}} \quad \text{where} \begin{cases} \text{MSE}_{\text{node}} = \sum_{i \in \text{node}} (\hat{y}_{\text{node}} - y^{(i)})^2 \\ \hat{y}_{\text{node}} = \frac{1}{m_{\text{node}}} \sum_{i \in \text{node}} y^{(i)} \end{cases}$$

Decision Trees

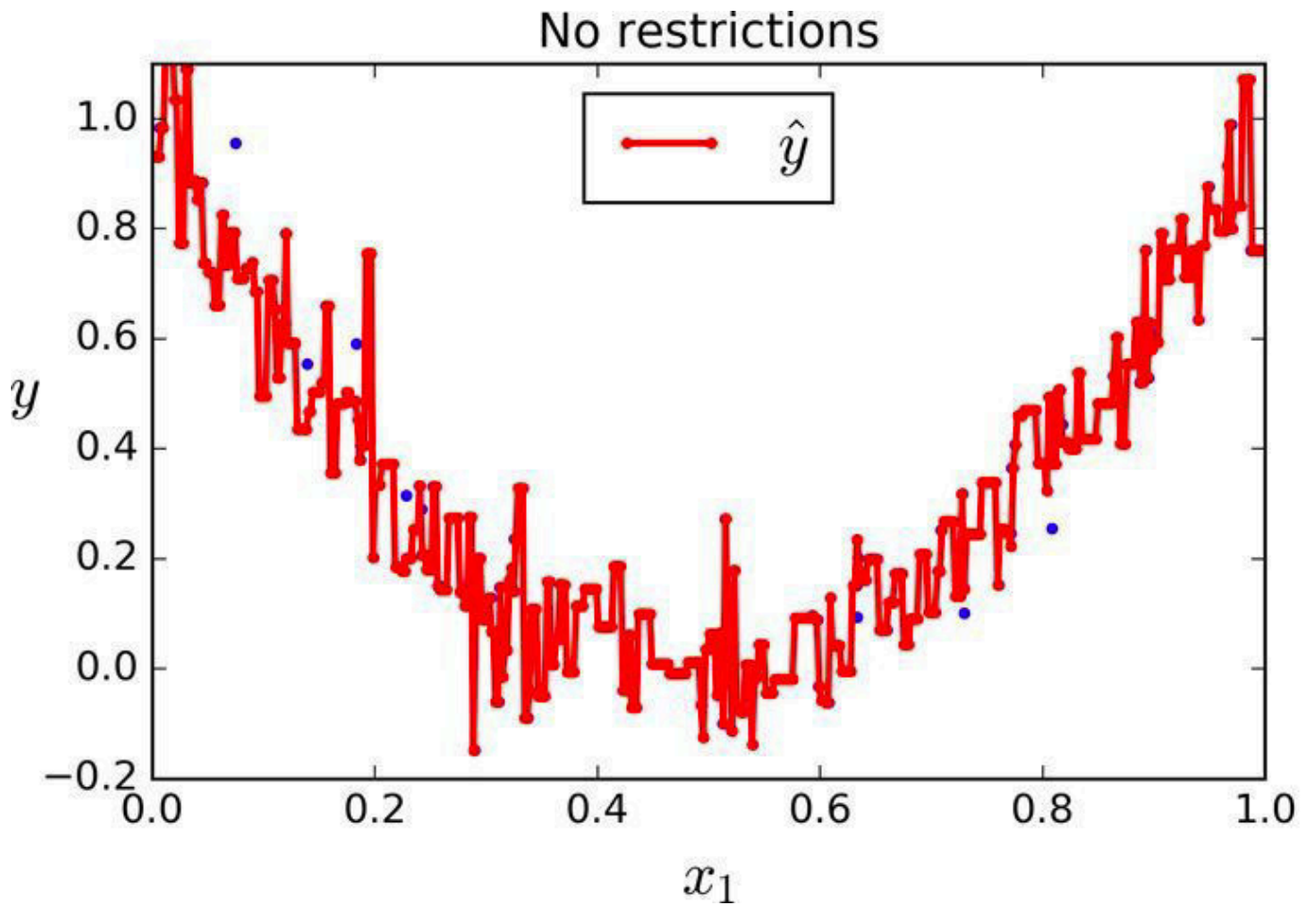
Regression with Decision Trees

- Just like for classification tasks, Decision Trees are prone to overfitting when dealing with regression tasks.

- Without any regularization the model may overfit the data.

Decision Trees

Regression with Decision Trees

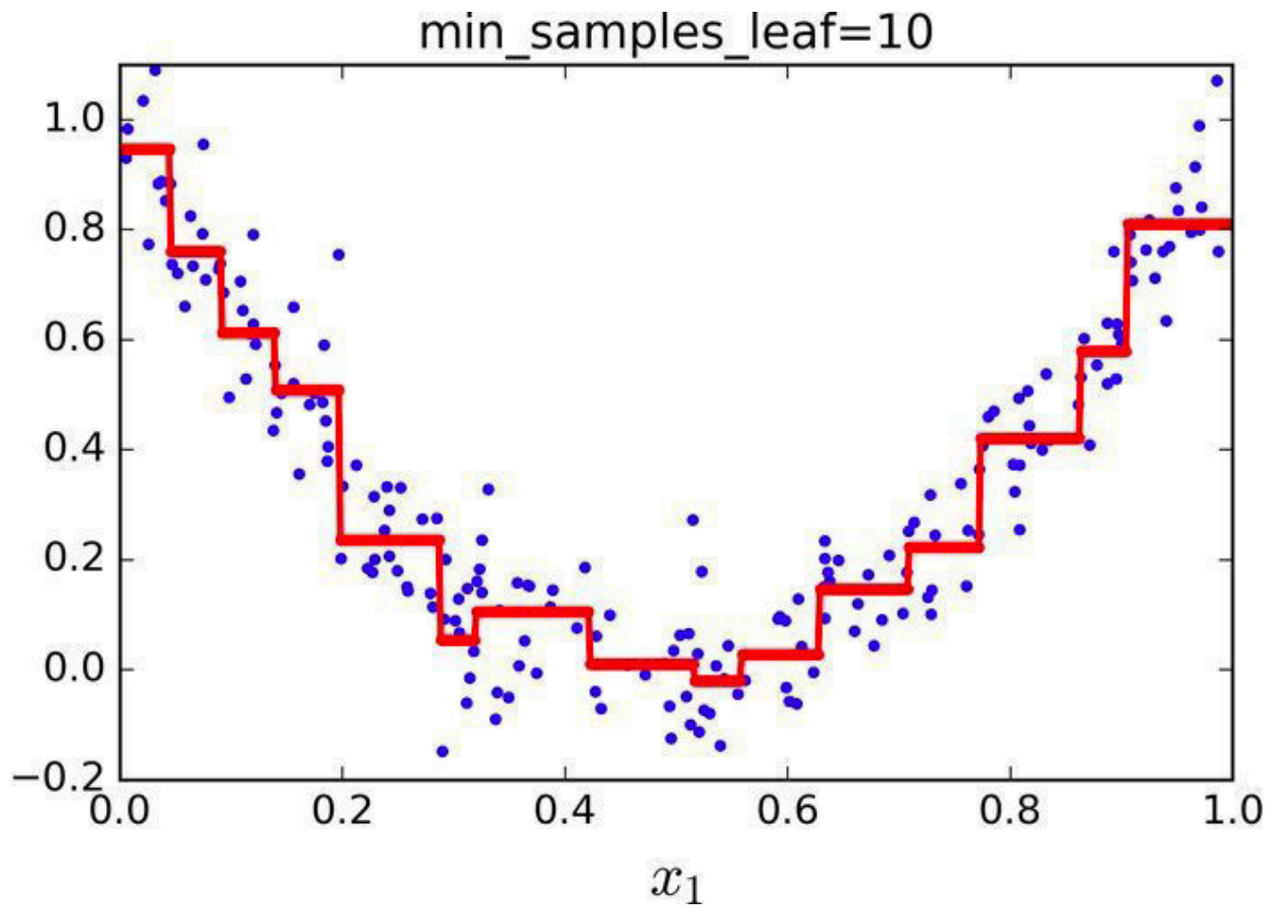


Without any regularization i.e., using the default hyperparameters, you get the above model.

It is obviously overfitting the training set very badly.

Decision Trees

Regression with Decision Trees



Just setting min_samples_leaf=10 results in a much more reasonable model, represented by the above figure.

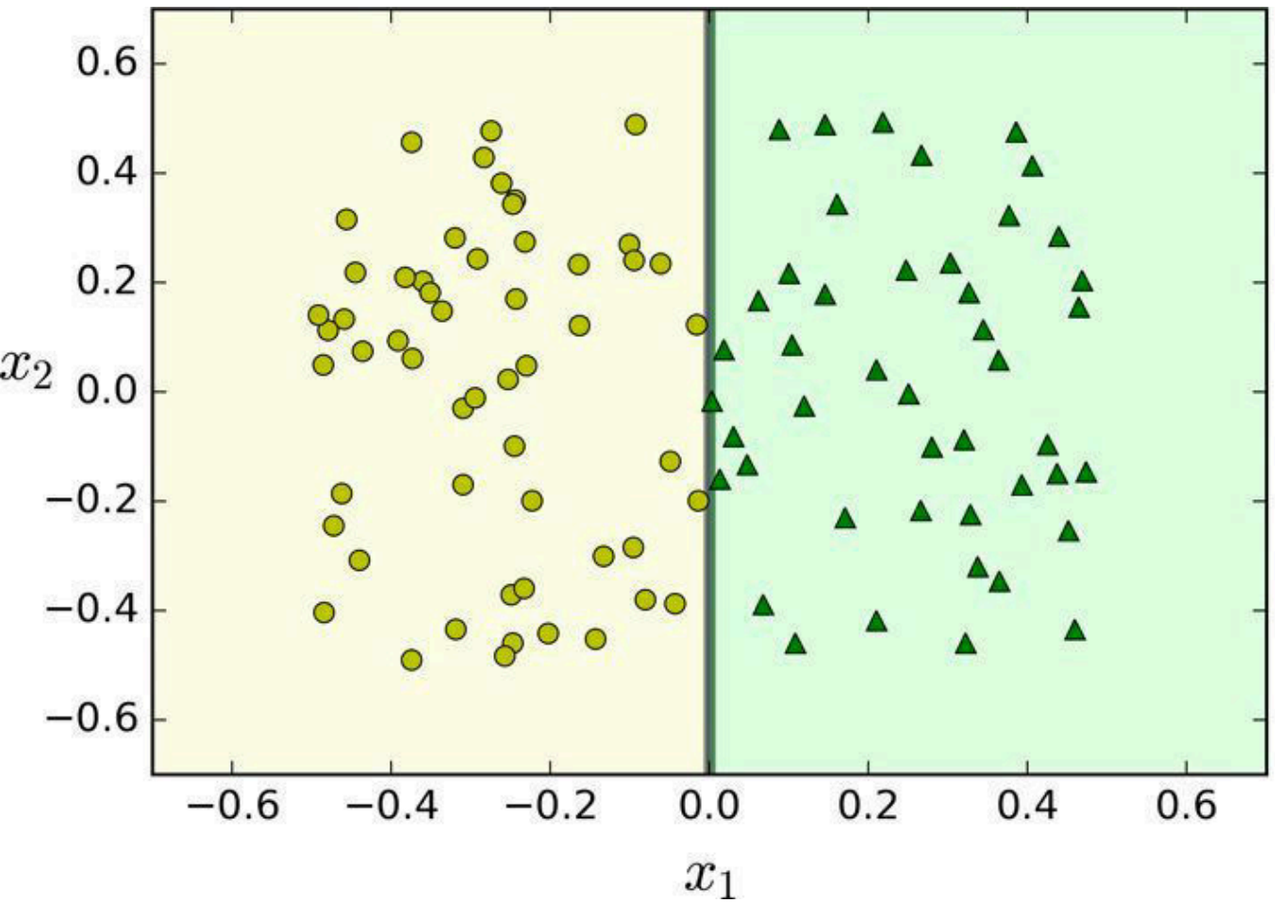
Decision Trees

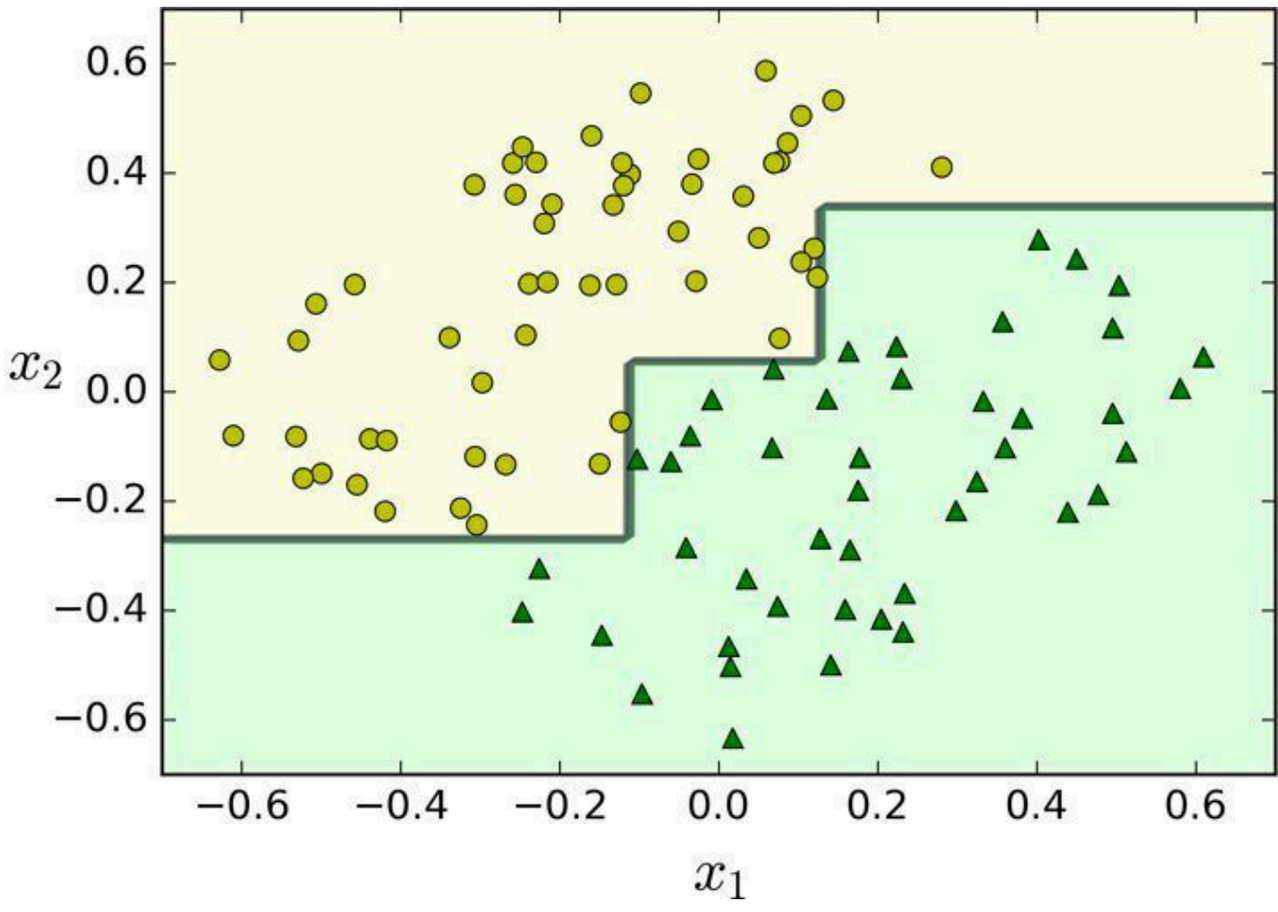
Demerits of Decision Trees

- Decision Trees love orthogonal decision boundaries (all splits are perpendicular to an axis)
- This makes them sensitive to training set rotation.
- They are very sensitive to small variations in the training data.

Decision Trees

Demerits of Decision Trees

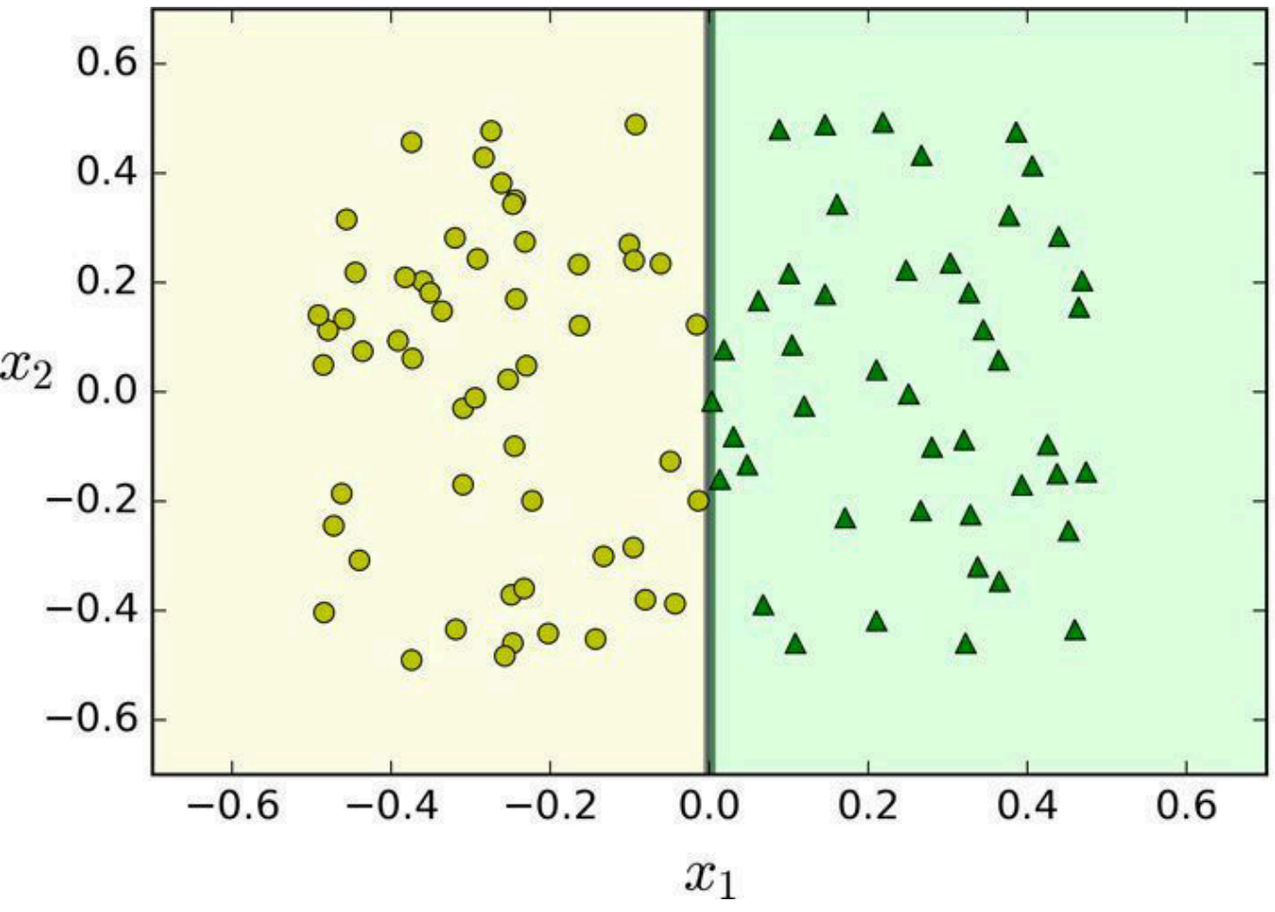


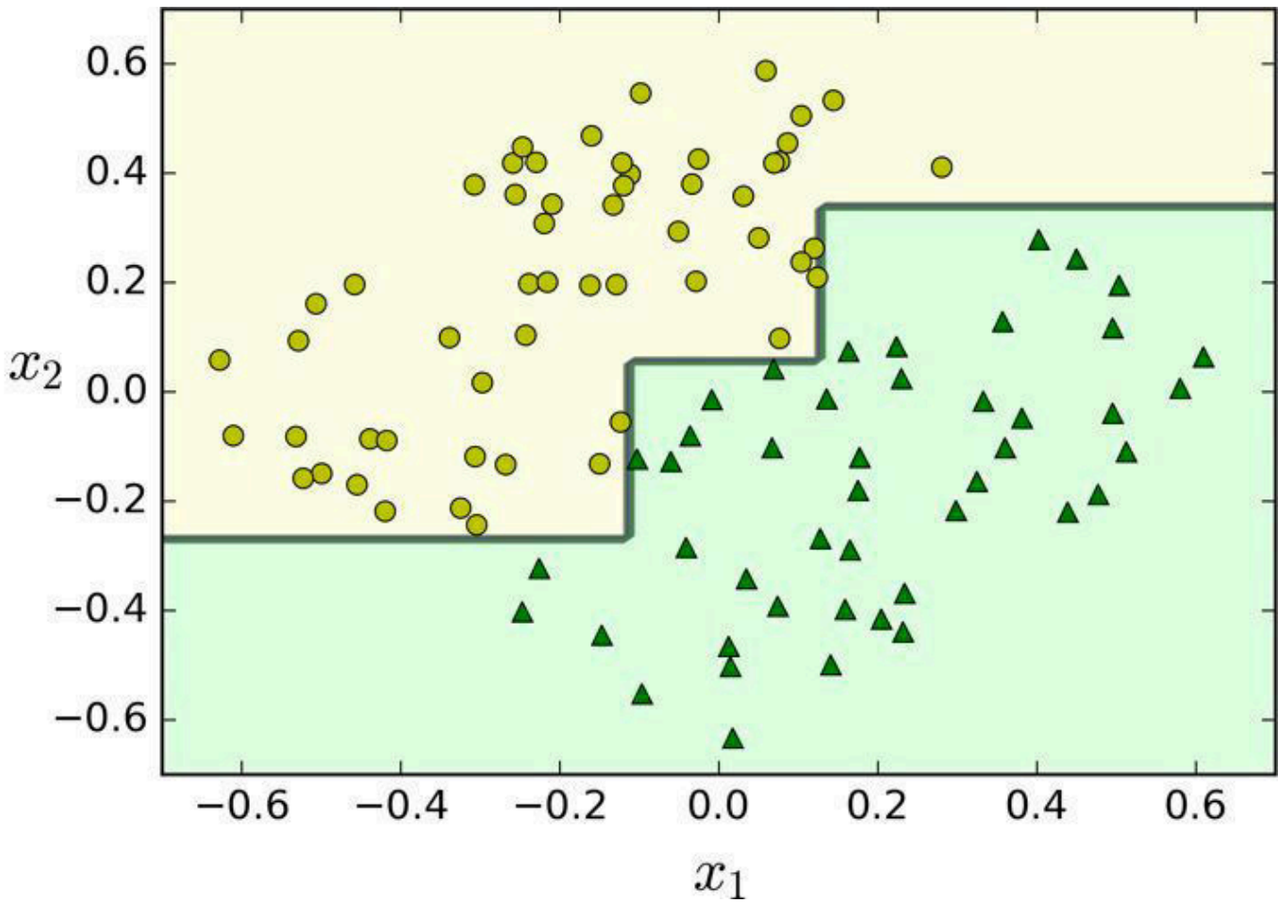


Above figure shows a simple linearly separable dataset: on the left, a Decision Tree can split it easily, while on the right, after the dataset is rotated by 45°, the decision boundary looks unnecessarily convoluted.

Decision Trees

Demerits of Decision Trees





Although both Decision Trees fit the training set perfectly, it is very likely that the model on the right will not generalize well.

Questions?

Thank You