Data Analytics
EEE 4774 & 6777

Module 4 - Classification

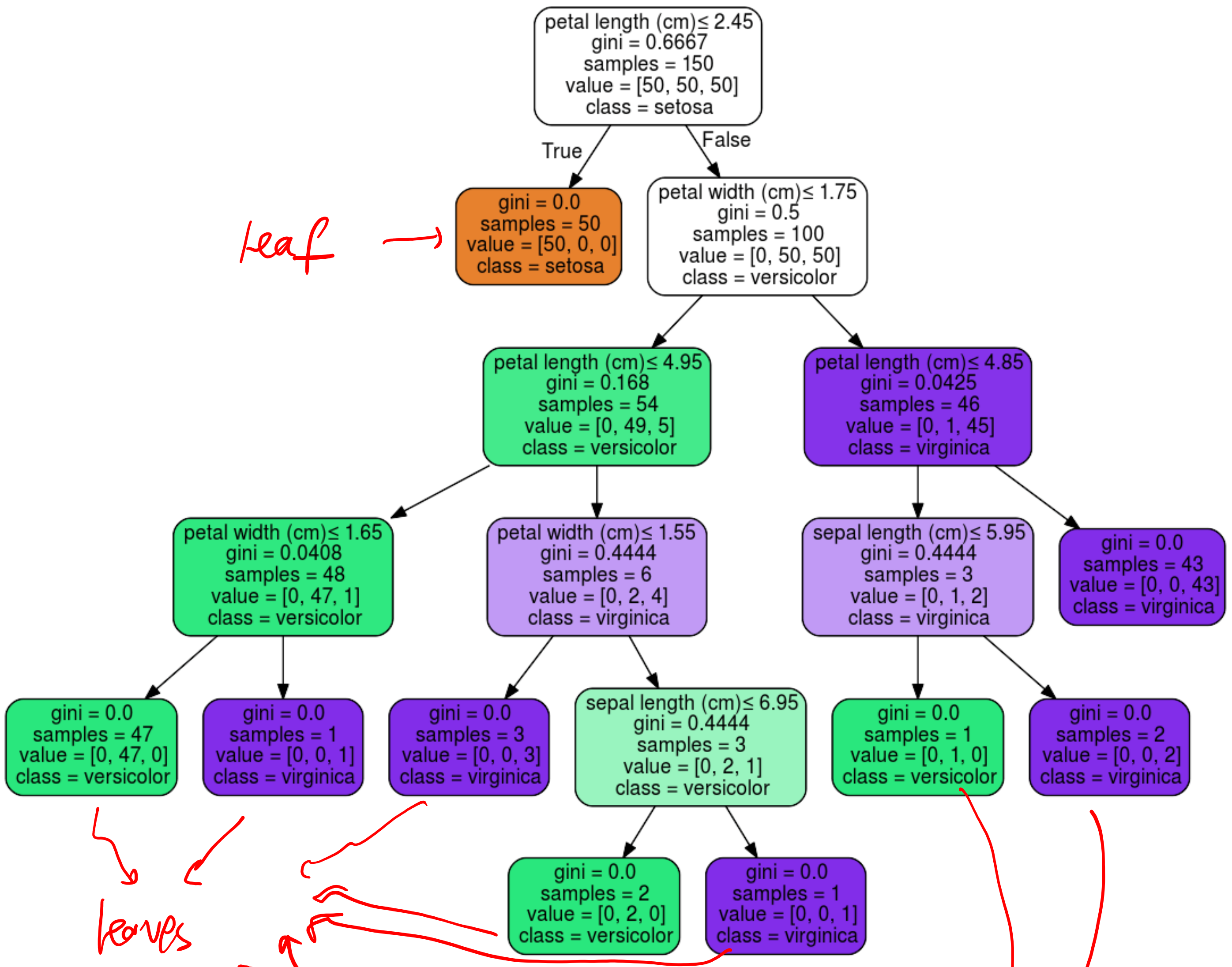
Decision Tree

Spring 2022

Decision Tree

- Non-parametric supervised learning method used for both classification and regression
- Goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
- Visually and explicitly represent decisions and decision making. A tree-like model of decisions *drawn upside down with its root at the top.*
- Condition/**internal node**, based on which the tree splits into branches/**edges**. The end of the branch that doesn't split anymore is the decision/**leaf**
- Growing a tree involves deciding on **which features to choose**, **what conditions to use** for splitting, and knowing **when to stop**.

root



Decision tree for the Iris dataset

Tree algorithms: ID3

- Iterative Dichotomiser 3 (1986)
- Creates a multiway tree for categorical features in a greedy way:
  - Find the feature (attribute) that yields the largest information gain
  - Repeat (recurse) for the following nodes until the max depth is reached
  - Then apply a pruning step
- Recursion on a branch stops if
  - All instances have the same label
  - No more features to select (leaf node labeled with the most common label)
- Training: build the tree and store it in memory
- Test: use the tree to classify new instances

Tree algorithms: ID3

Information gain $IG(A)$ is the measure of the difference in entropy from before to after the set S is split on an attribute A. In other words, how much uncertainty in S was reduced after splitting set S on attribute A.

$$IG(S,A) = \mathrm{H}(S) - \sum_{t \in T} p(t)\mathrm{H}(t) = \mathrm{H}(S) - \mathrm{H}(S|A).$$

feature

Where,

- $\mathrm{H}(S)$ – Entropy of set S
- T – The subsets created from splitting set S by attribute A such that $S = \bigcup_{t \in T} t$
- $p(t)$ – The proportion of the number of elements in t to the number of elements in set S
- $\mathrm{H}(t)$ – Entropy of subset t

In ID3, information gain can be calculated (instead of entropy) for each remaining attribute. The attribute with the **largest** information gain is used to split the set S on this iteration.

Entropy $\mathrm{H}(S)$ is a measure of the amount of uncertainty in the (data) set S (i.e. entropy characterizes the (data) set S).

$$\mathrm{H}(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$

Where,

- S – The current dataset for which entropy is being calculated
  - This changes at each step of the ID3 algorithm, either to a subset of the previous set in the case of splitting on an attribute or to a "sibling" partition of the parent in case the recursion terminated previously.
- X – The set of classes in S
- $p(x)$ – The proportion of the number of elements in class x to the number of elements in set S

When $\mathrm{H}(S) = 0$, the set S is perfectly classified (i.e. all elements in S are of the same class).

Tree algorithms: C4.5

- Successor to ID3 – removed the restriction that features must be categorical
  - Dynamically defines a discrete attribute (based on numerical variables) that partitions the continuous attribute value into a discrete set of intervals
- C4.5 converts the trained trees (i.e. the output of the ID3 algorithm) into sets of if-then rules.
- The accuracy of each rule is then evaluated to determine the order in which they should be applied.

Pruning:

- Pruning is done by removing a rule's precondition if the accuracy of the rule improves without it.

Tree algorithms: CART

- Classification and Regression Trees (CART) is very similar to C4.5, but it differs in that
  - it supports numerical target variables (regression) and
  - does not compute rule sets.
- CART constructs binary trees using the feature and threshold that yield the largest information gain at each node.
- Splitting criteria:
  - Gini impurity measure: a variation of the usual entropy measure for decision trees. For K classes,

$$G = \sum_{k=1}^{K} p_k(1 - p_k)$$

  - p_k is proportion of class k instances present in the set. A perfect class purity occurs when a set contains all instances from the same class, in which case p_k is either 1 or 0 and $G = 0$. A node having a 50–50 split of classes in a set has the worst purity, so for a binary classification it will have $p_k = 0.5$ and $G = 0.5$.

Tree algorithms: CART

similar (Dimensionality reduction)
feature extraction
different (Original features are preserved or not)

Advantages of CART

- Simple to understand, interpret, visualize.
- Decision trees *implicitly perform variable screening or feature selection.*
- Can *handle both numerical and categorical data*. Can also *handle multi-output problems.*
- Decision trees require relatively *little effort from users for data preparation.*
- *Nonlinear relationships between parameters do not affect tree performance.*

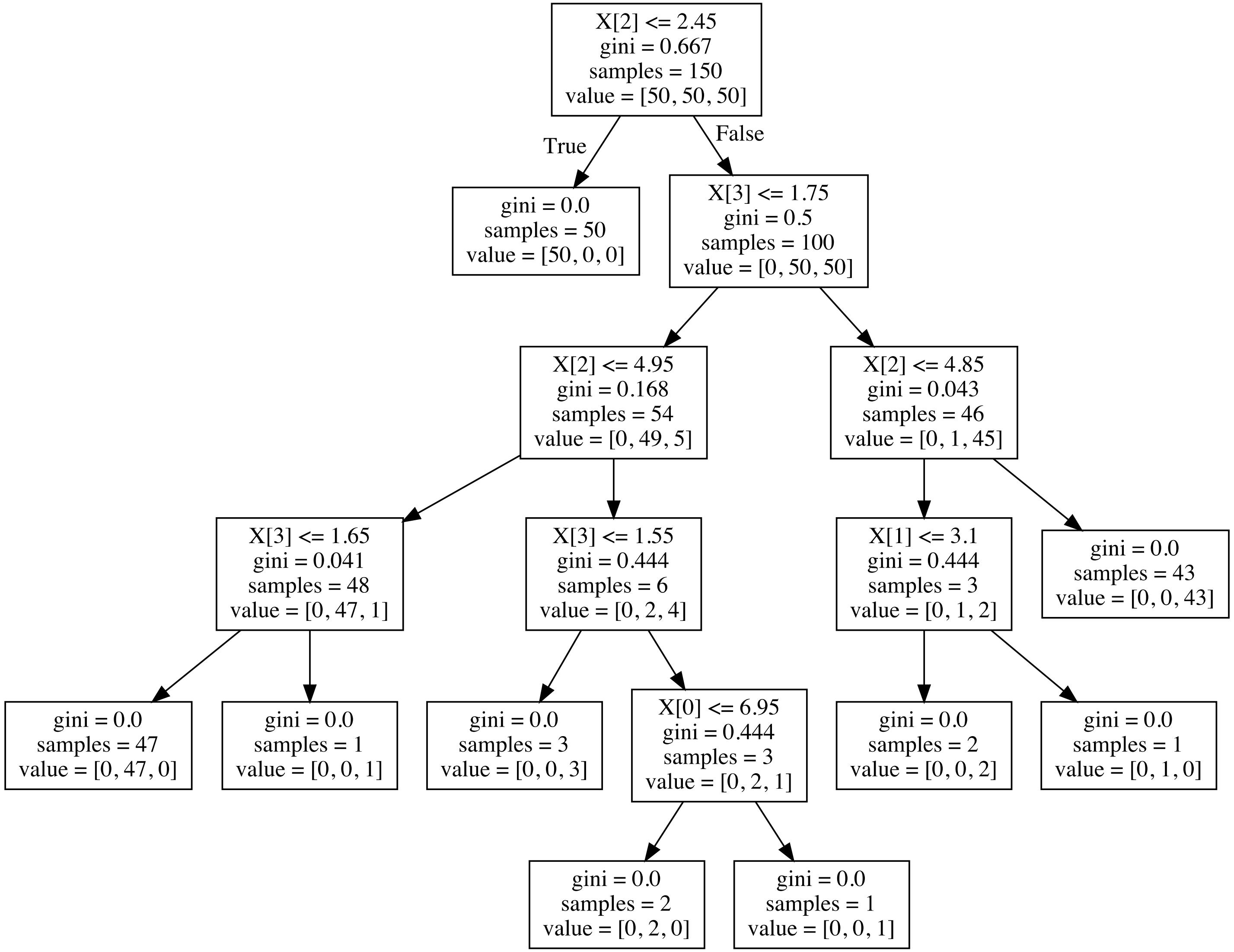
Disadvantages of CART

- Decision-tree learners *can create over-complex trees* that do not generalize the data well. This is called *overfitting*.
- Decision trees can be unstable because *small variations in the data might result in a completely different tree being generated.* This is called <u>*variance*</u>, which needs to be *lowered by methods like* <u>***bagging***</u> *and* <u>***boosting***</u>.
- Greedy algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees, where the features and samples are randomly sampled with replacement.
- Decision tree learners create <u>*biased*</u> *trees if some classes dominate*. It is therefore recommended to balance the data set prior to fitting with the decision tree.

Tips on practical use in scikit-learn

- scikit-learn uses an optimised version of the CART algorithm; however, scikit-learn implementation does not support categorical variables for now.
- Decision trees tend to overfit on data with a large number of features. Getting the right ratio of samples to number of features is important, since a tree with few samples in high dimensional space is very likely to overfit.
- Consider performing dimensionality reduction (PCA, ICA, or Feature selection) beforehand to give your tree a better chance of finding features that are discriminative.
- Understanding the decision tree structure will help in gaining more insights about how the decision tree makes predictions, which is important for understanding the important features in the data.
- Visualize your tree as you are training by using the export function. Use max_depth=3 as an initial tree depth to get a feel for how the tree is fitting to your data, and then increase the depth.
- Remember that the number of samples required to populate the tree doubles for each additional level the tree grows to. Use max_depth to control the size of the tree to prevent overfitting.
- Use min_samples_split or min_samples_leaf to ensure that multiple samples inform every decision in the tree, by controlling which splits will be considered. While min_samples_split can create arbitrarily small leaves, min_samples_leaf guarantees that each leaf has a minimum size. For classification with few classes, min_samples_leaf=1 is often the best choice.

Python Exercise



Decision tree for the Iris dataset