Decision Trees

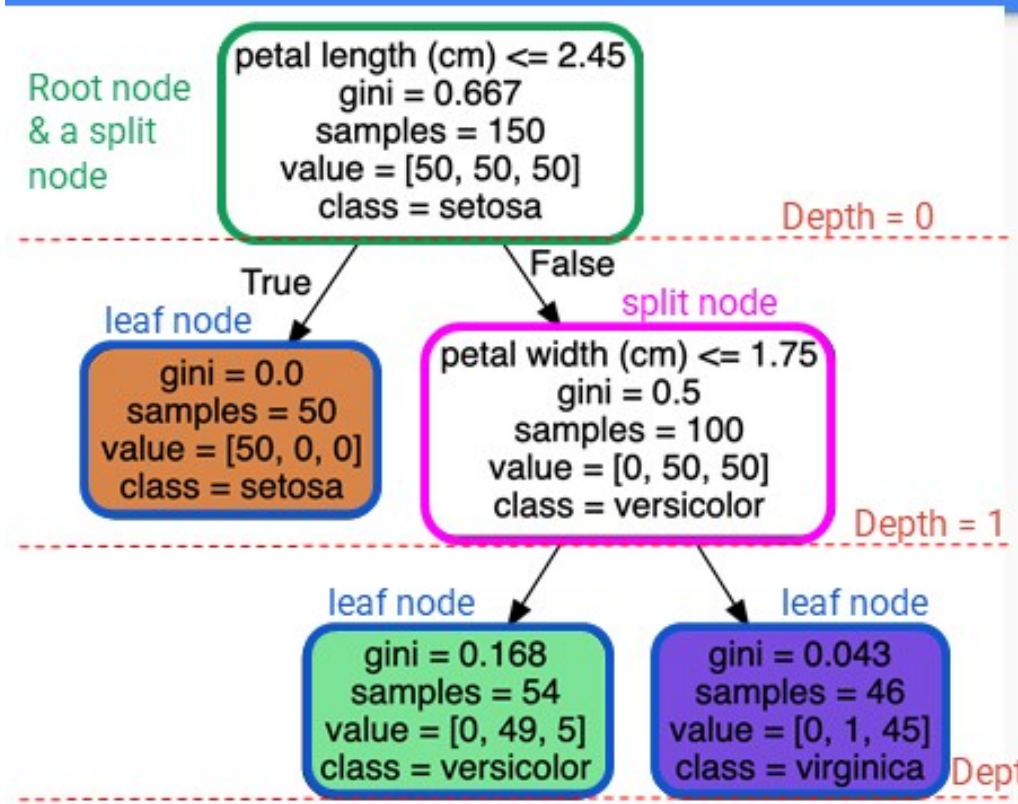
By Waad ALMASRI

Introduction

Decision Trees:

- Versatile Machine Learning (ML) algorithms
- Performs classification, regression, & even multi output tasks
- Algorithm capable of fitting complex datasets
- Nonparametric model
- Fundamental component of random forests, a very powerful ML model

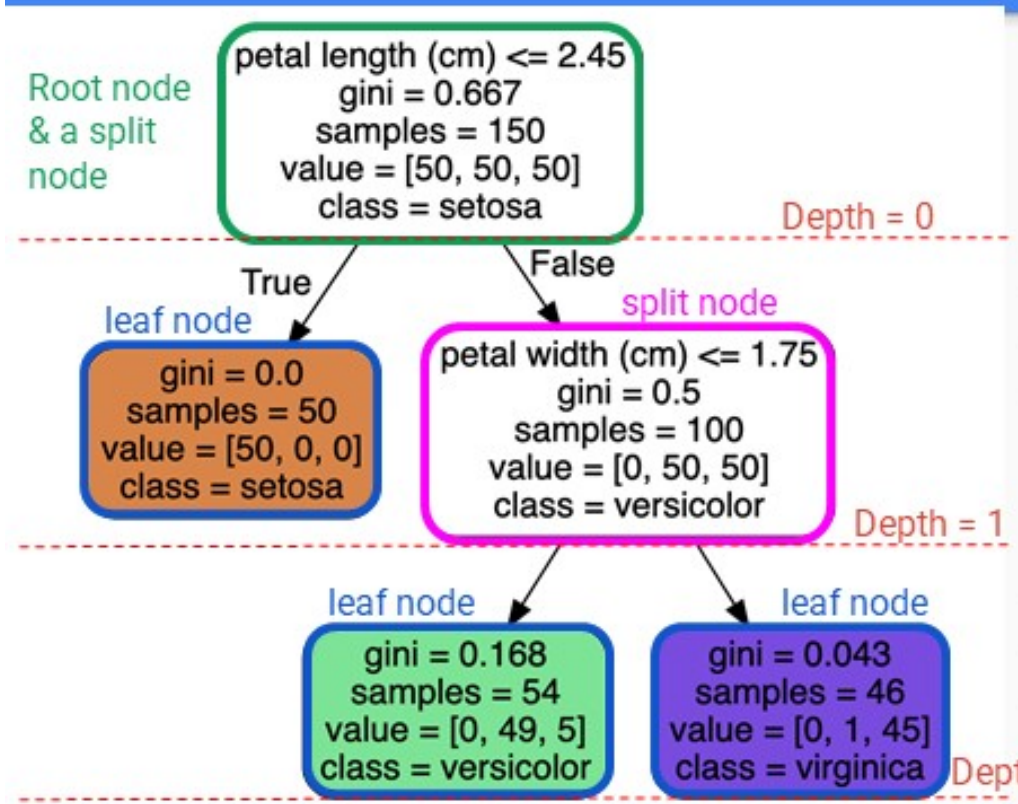
Introduction



At every node, there are 4 attributes:

- *Samples*: this attribute counts how many training samples validate this node's condition.
- *Value*: it counts the number of training instances per class, this nodes applies to; it is a list of length = to the number of classes.
- *Class*: the majority class of the training samples belonging to this node; the class with the highest *value*.
- *Gini*: it measures the Gini impurity; a node is pure (i.e., gini=0) if all training samples it applies to belong to the same class.

Impurity measures



To identify the optimum split, the decision tree algorithm computes an impurity measure. It is defined by the *criterion* hyperparameter, it could be either:

- The Gini Impurity:

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

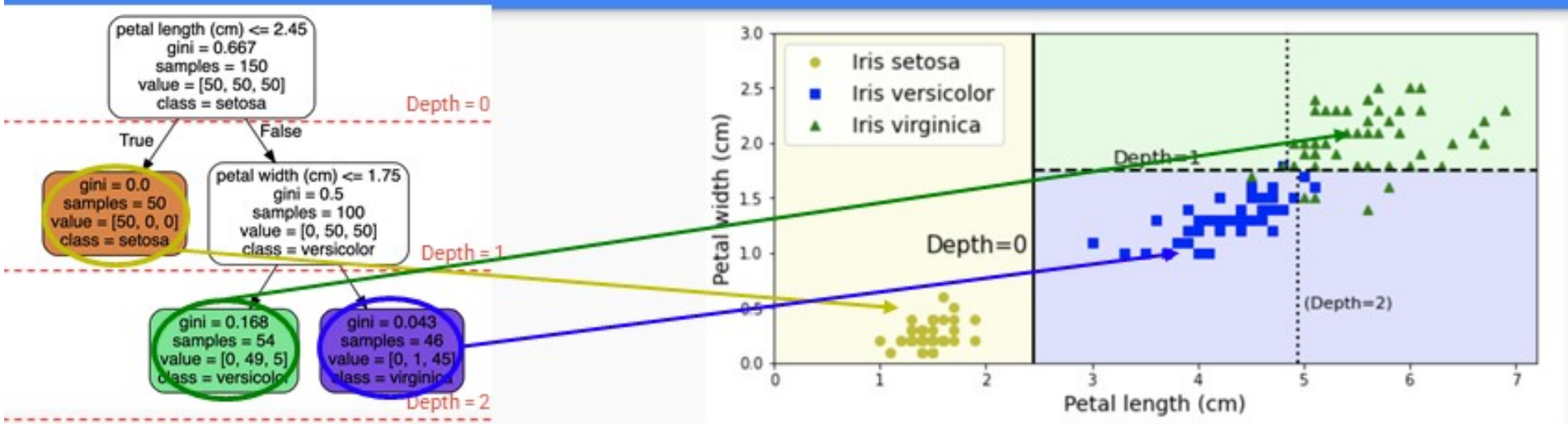
- G_i the gini impurity of the node i
- $p_{i,k}$ the ratio of class k instances among the training samples in the ith node.
- n = the total number of classes
- Example: for the depth 2, left node, the gini impurity is = $1 - \left(\left(\frac{0}{54}\right)^2 + \left(\frac{49}{54}\right)^2 + \left(\frac{5}{54}\right)^2\right) = 0.168$

- The Entropy:

$$H_i = -\sum_{k=1}^{n} p_{i,k} \log_2(p_{i,k})$$

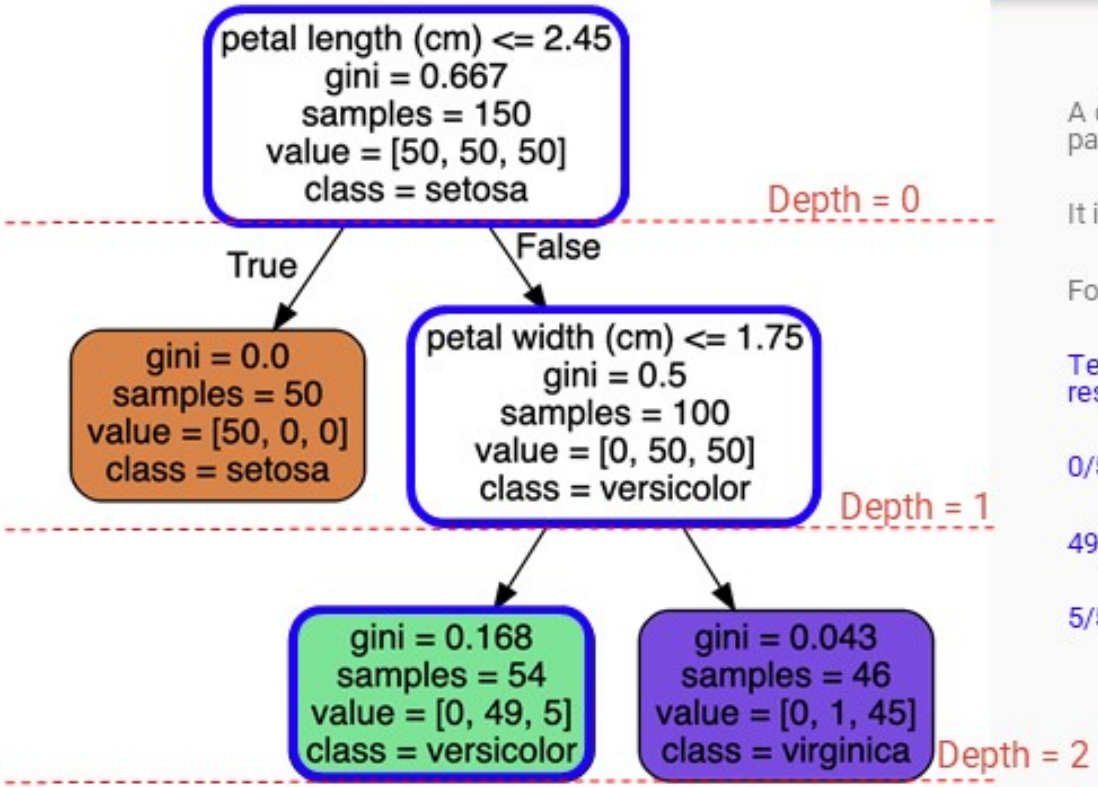
- $p_{i,k}$ the ratio of class k instances among the training samples in the ith node and $p_{i,k} \neq 0$
- n = the total number of classes
- Example: for the depth 2, left node, the entropy is = $-\left(\frac{0}{54}\right)\log_2\left(\frac{0}{54}\right) - \left(\frac{49}{54}\right).\log_2\left(\frac{49}{54}\right) - \left(\frac{5}{54}\right).\log_2\left(\frac{5}{54}\right) = 0.445$

Decision Boundary



The thick vertical line represents the decision boundary of the root node (depth 0): petal length=2.45cm.
Since the lefthand area is pure (only Iris setosa), it cannot be split any further.
However, the righthand area is pure (only Iris setosa), so the depth-1 right node splits it at petal width = 1.75cm (represented by the dashed line).
Since max_depth was set to 2, the decision tree stops right there.
If max_depth is set to 3, then the two depth-2 nodes would each add another decision boundary (represented by the two vertical dotted lines).

Estimating class probabilities



A decision tree estimates the probability that an instance belongs to a particular class k.

It is the ratio of the training samples of class k in this node.

For example:

Test sample: petal length = 5cm long and petal width = 1.5cm →the result is the depth-2 left node and the probabilities output are:

0/54=0% for Iris setosa

49/54=90.7% for Iris versicolor

5/54=9.3% for Iris virginica

Classification And Regression Tree (CART)

- Scikit-Learn decision tree model is based on the CART algorithm.
- The training instances are first splitted into 2 subsets using a single feature k and a threshold t_k.
- The algorithm searches for the pair (k, t_k) that produces the purest subsets, weighted by their sizes by minimizing the following cost function:
  - $J(k, t_k) = \frac{n_{left}}{n} G_{left} + \frac{n_{right}}{n} G_{right}$
  - Such that $G_{left\ or\ right}$ measures the impurity of the left/right subset,
  - $n_{left\ or\ right}$ is the number of instances in the left/right subset,
  - and n the total number of training instances in the current node.
- After the first split, the subsets get splitted using the same logic.
- The splits stop once the maximum depth is reached (the *max_depth* hyperparameter) or when the algorithm cannot find a split that will reduce the impurity.

Regression Decision Tree

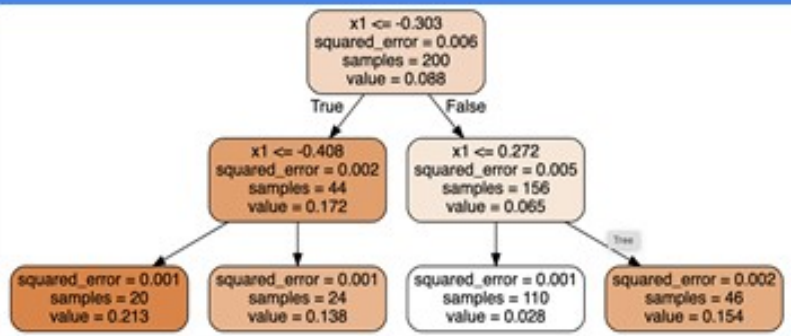
A regression tree is different from a classification tree in the prediction: instead of a class, it predicts a value.
At each node, there are 3 attributes:

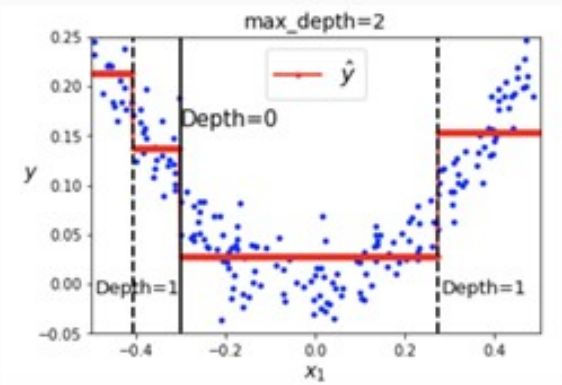
- Squared error: this is the mean squared error over the *n'* training samples associated with this node
- Samples: the number of training samples associated with this node *n'*
- Value: this prediction is the average target value of the *n'* training samples associated with this node

The CART algorithm for regression splits the training set to minimize MSE. The cost function to be minimized:

- $J(k, t_k) = \frac{n_{left}}{n} MSE_{left} + \frac{n_{right}}{n} MSE_{right}$
- Such that $MSE_{right\ or\ left} = \frac{\sum_{i \in node}(\hat{y}_{node} - y_i)^2}{n_{node}}$ and $\hat{y}_{node} = \frac{\sum_{i \in node} y_i}{n_{node}}$
- $\$n_{left\ or\ right}$ is the number of instances in the left/right subset,
- and n the total number of training instances in the current node.

Scikit-Learn: DecisionTreeRegressor.





Pros vs Cons of decision trees

- Model requiring very little data preparation; they don't need feature scaling (unlike SVMs) or centering at all.
- Very intuitive, easy to interpret; white box models.
- Fast predictor even with large training sets; prediction complexity independent of the number of features ($\Theta(\log_2(n))$)

- Greedy algorithm→ solution can be good, but rarely optimal.
- Sensitive to the data's orientation because it only supports orthogonal decision boundaries (all splits are perpendicular to an axis)
- Prone to overfitting the training data → need for regularization
- High variance: a small change in the hyperparameter/data can produce a completely different model → solution=ensemble of trees (e.g., random forest)

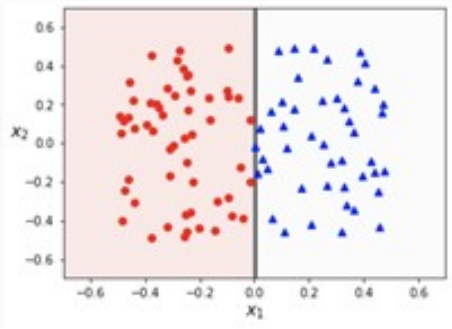
Sensitivity to axis orientation

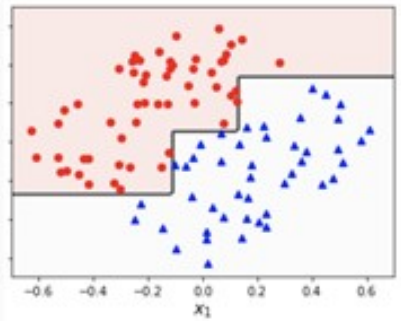
In the Pros vs Cons of decision trees, we have mentioned that the decision tree is sensitive to the data's orientation.

Indeed, decision trees have orthogonal decision boundaries, which can affect their generalization performance.

To overcome this constraint, scaling could be applied, like a PCA. a PCA rotates the data in a way to reduce the correlation between the features.

Which of the models on the right will not generalize well?





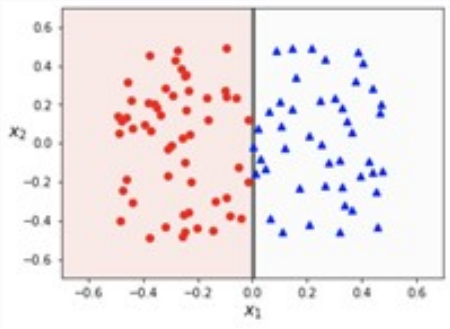
Sensitivity to axis orientation

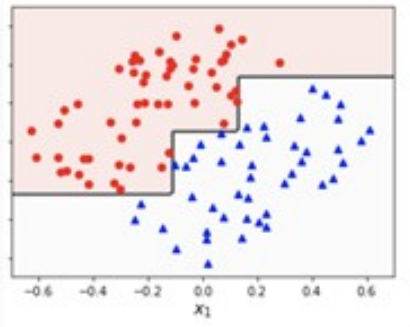
In the Pros vs Cons of decision trees, we have mentioned that the decision tree is sensitive to the data's orientation.

Indeed, decision trees have orthogonal decision boundaries, which can affect their generalization performance.

To overcome this constraint, scaling could be applied, like a PCA. a PCA rotates the data in a way to reduce the correlation between the features.

Which of the models on the right will not generalize well?





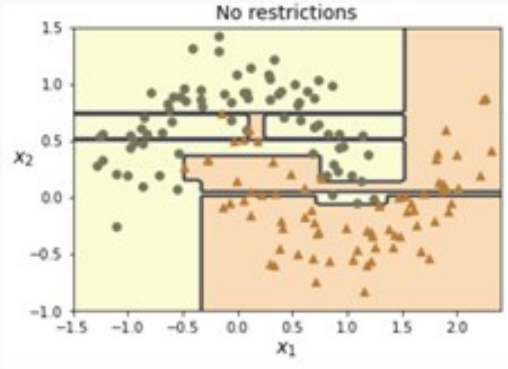
Regularization hyperparameters

With the decision tree being a nonparametric model, i.e., a model that fits the training data closely, it risks overfitting this training data.

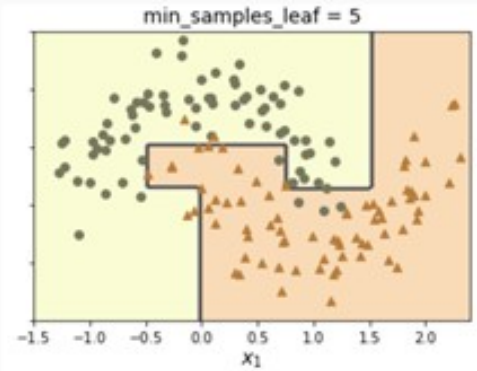
Thus, to avoid this, we can restrict the decision tree's freedom during training, called regularization, via finetuning some hyperparameters (by increasing the min_... and decreasing the max_...):

- Max_depth: the maximum depth of a tree; reducing its value reduces the risk of overfitting
- Max_features: Maximum number of features evaluated for splitting each node
- Min_samples_split: minimum number of samples a node must have before it can be split
- Min_samples_leaf: minimum number of samples a leaf node must have to be created
- Min_weight_fraction_leaf: same as Min_samples_leaf, but expressed as a fraction of the total number of weighted instances.

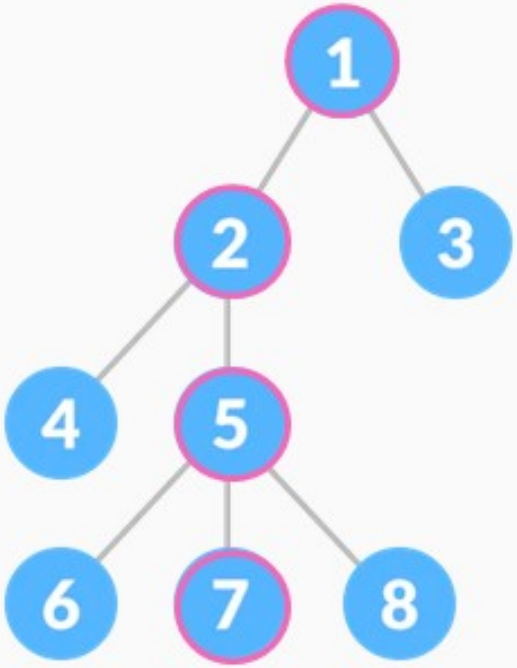
overfitting



regularized



Computational Complexity



To predict on new input using the decision tree model, the input must traverse the tree.

The worst case scenario is that the prediction is in the leaf node belonging to the longest path.

The longest path is equal to the maximum depth of the tree, which is $\log_2(n)$; n = total number of training samples.

Moreover, each node along the path only requires checking the value of one feature.

Thus, the prediction complexity is $\Theta(\log_2(n)+1)=\Theta(\log_2(n))$

On the other side, during the training, the algorithm compares all features (m=total number of features) on all samples at each node (to find the best k and threshold t_k. This comparison costs $\Theta(nm)$, since traversing the tree costs $\Theta(\log_2(n))$ → the training complexity is $\Theta(nm\log_2(n))$.

Fun facts

A decision tree with max_depth = 1 (i.e., a tree composed of a single decision node plus 2 leaf nodes) is called a decision stump; it is sometimes used with Boosting ensemble method.

References

https://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf

Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. " O'Reilly Media, Inc.", 2022.