Decision Trees

Decision Trees

- Decision Trees are versatile ML algorithms that can perform both classification and regression tasks, even multi-output tasks
- They are also fundamental components of Random Forests

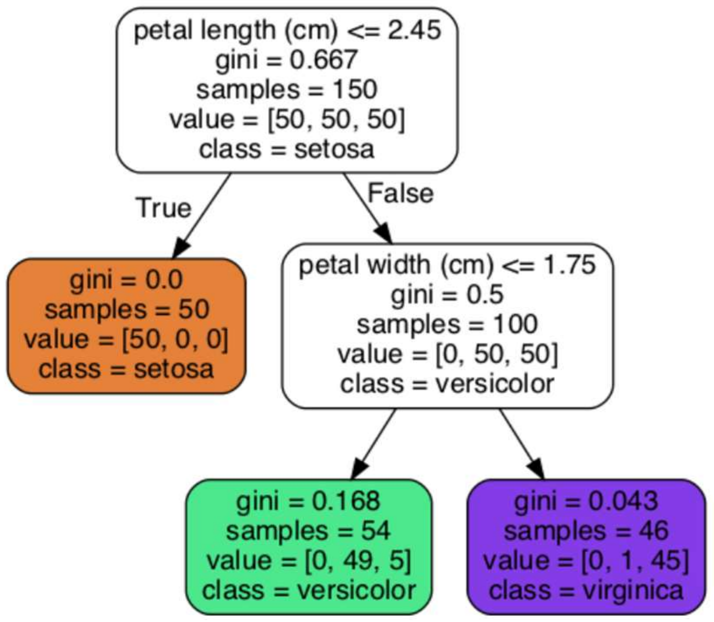
Training and Visualizing a Decision Tree

- The decision tree can be classified using **DecisionTreeClassifier** from scikit-learn
- You can visualize the trained Decision Tree by first using the **export_graphviz()** method to output a graph definition file called <filename>.dot

Demo

Making Predictions

- Suppose you find an iris flower and you want to classify it.
- You start at the root node (depth 0, at the top): this node asks whether the flower's petal length is smaller than 2.45 cm.
- 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 (i.e., it does not have any children nodes), so it does not ask any questions: you can simply look at the predicted class for that node
- The Decision Tree predicts that your flower is an Iris Setosa (class=setosa).



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.
- For example, suppose you have found a flower whose petals are 5 cm long and 1.5 cm wide. The corresponding leaf node is the depth-2 left node
- The Decision Tree should output the following probabilities: 0% for Iris-Setosa (0/54), 90.7% for Iris-Versicolor (49/54), and 9.3% for Iris-Virginica (5/54).

Gini Impurity

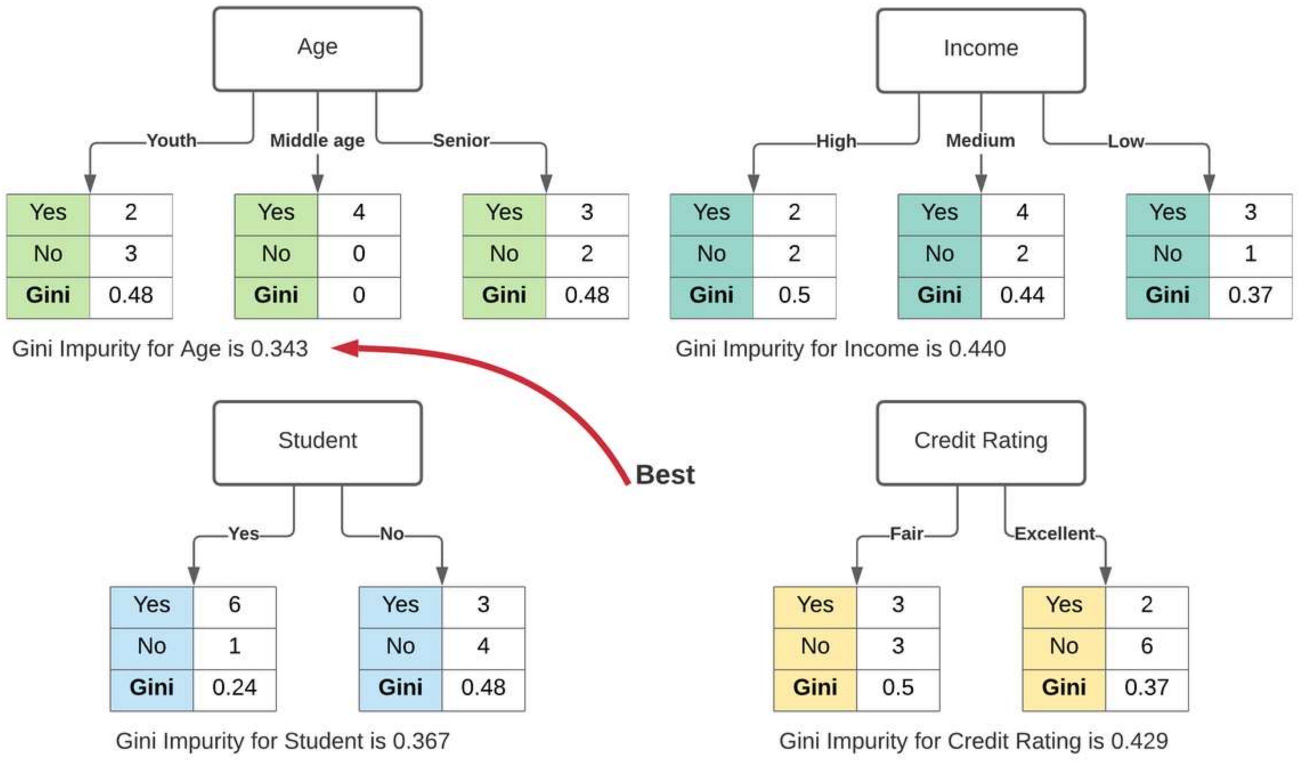
- Gini Impurity is a measurement used to build Decision Trees to determine how the features of a dataset should split nodes to form the tree.
- More precisely, the Gini Impurity of a dataset is a number between 0-0.5, which indicates the likelihood of new, random data being misclassified if it were given a random class label according to the class distribution in the dataset.
- Consider a dataset D that contains samples from k classes. The probability of samples belonging to class i at a given node can be denoted as pi. Then the Gini Impurity of D is defined as:

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

Gini Impurity

- For example, say you want to build a classifier that determines if someone will default on their credit card. You have some labeled data with features, such as bins for age, income, credit rating, and whether or not each person is a student.
- To find the best feature for the first split of the tree – the root node – you could calculate how poorly each feature divided the data into the correct class, default ("yes") or didn't default ("no").
- This calculation would measure the **impurity** of the split, and the feature with the lowest impurity would determine the best feature for splitting the current node.

Gini Impurity



Gini Impurity

"Will I Go Running" Data Set

Day	Weather	Just Ate	Late at Work	Will I go Running?
1	'Sunny'	'yes'	'no'	'yes'
2	'Rainy'	'yes'	'yes'	'no'
3	'Sunny'	'no'	'yes'	'yes'
4	'Rainy'	'no'	'no'	'no'
5	'Rainy'	'no'	'no'	'yes'
6	'Sunny'	'yes'	'no'	'yes'
7	'Rainy'	'no'	'yes'	'no'

In the data set above, there are two classes in which data can be classified: “yes” (I will go running) and “no” (I will not go running).

If we were using the entire data set above as training data for a new decision tree (not enough data to train an accurate tree… but let’s roll with it) the gini impurity for the set would be calculated:

Gini Impurity

G(will I go running) = P("yes") * 1 - P("yes") + P("no") * 1 - P("no")

G(will I go running) = 4 / 7 * (1 - 4/7) + 3 / 7 * 1 - P(3/7)

G(will I go running) = 0.489796

This means there is a 48.97% chance of a new data point being incorrectly classified, based on the observed training data we have at our disposal. This number makes sense, since there are more yes class instances than no, so the probability of mis-classifying something is less than a coin flip.

When training a decision tree, the best split is chosen by maximizing the Gini Gain, which is calculated by subtracting the weighted impurities of the branches from the original impurity.

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".
- In Machine Learning, it is frequently used as an impurity measure: a set's entropy is zero when it contains instances of only one class.

$$H_i = -\sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^{n} p_{i,k} \log_2 \left(p_{i,k}\right)$$

Should you use Gini or Entropy: Most of the times it doesn't make a big difference, they lead to similar trees

Regularization of Hyperparameters

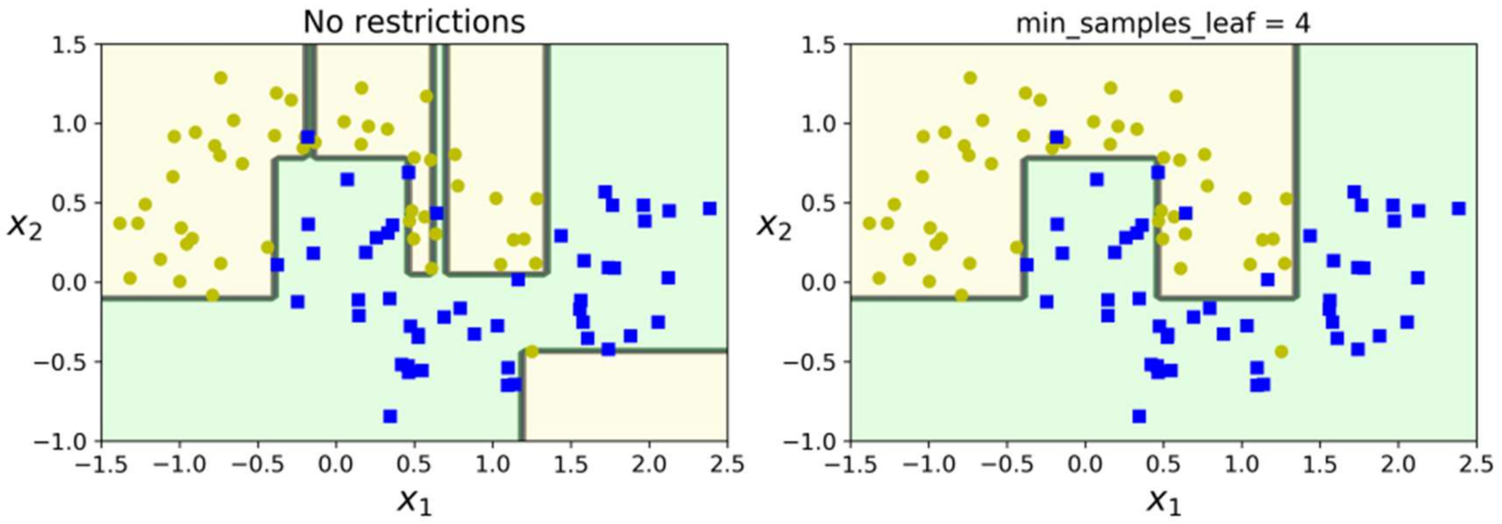
- Decision Trees make very few assumptions about the training data
- **Non-parametric model**: If left unconstrained, the tree structure will adapt itself to the training data, fitting it very closely, and most likely overfitting it.
- **Parametric model:** It has pre-determined number of parameters, so its degree of freedom is limited, reducing the risk of overfitting – but also increasing the risk of underfitting
- Constraining the Decision Tree’s freedom is called **Regularization**

Regularization of Hyperparameters

- In Scikit-Learn, this is controlled by the **max_depth** hyperparameter. Reducing max_depth will regularize the model and thus reduce the risk of overfitting.
- The **DecisionTreeClassifier** class has a few other parameters that similarly restrict the shape of the Decision Tree
- Increasing **min_*** hyperparameters or reducing **max_*** hyperparameters will regularize the model.

See other hyperparameters of the DecisionTreeClassifier in scikit-learn

Regularization of Hyperparameters



Pruning

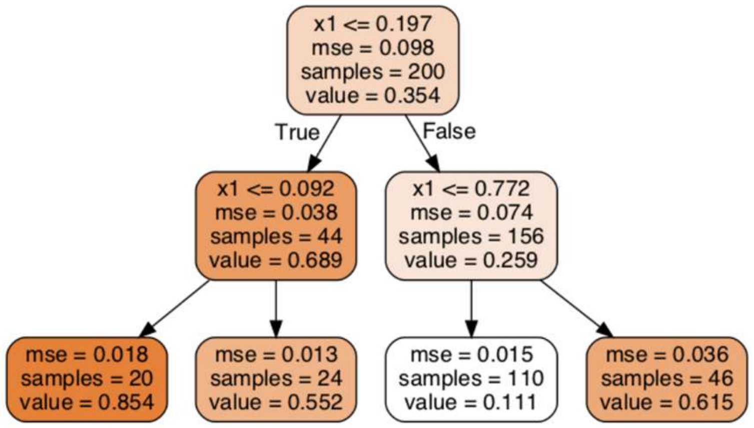
- Other algorithms work by first training the Decision Tree without restrictions, then pruning (deleting) unnecessary nodes.
- A node whose children are all leaf nodes is considered unnecessary if the purity improvement it provides is not statistically significant.
- Standard statistical tests, such as the $\chi 2$ test, are used to estimate the probability that the improvement is purely the result of chance (which is called the null hypothesis).
- If this probability, called the p-value, is higher than a given threshold (typically 5%, controlled by a hyperparameter), then the node is considered unnecessary and its children are deleted.
- The pruning continues until all unnecessary nodes have been pruned.

Decision Tree for Regression

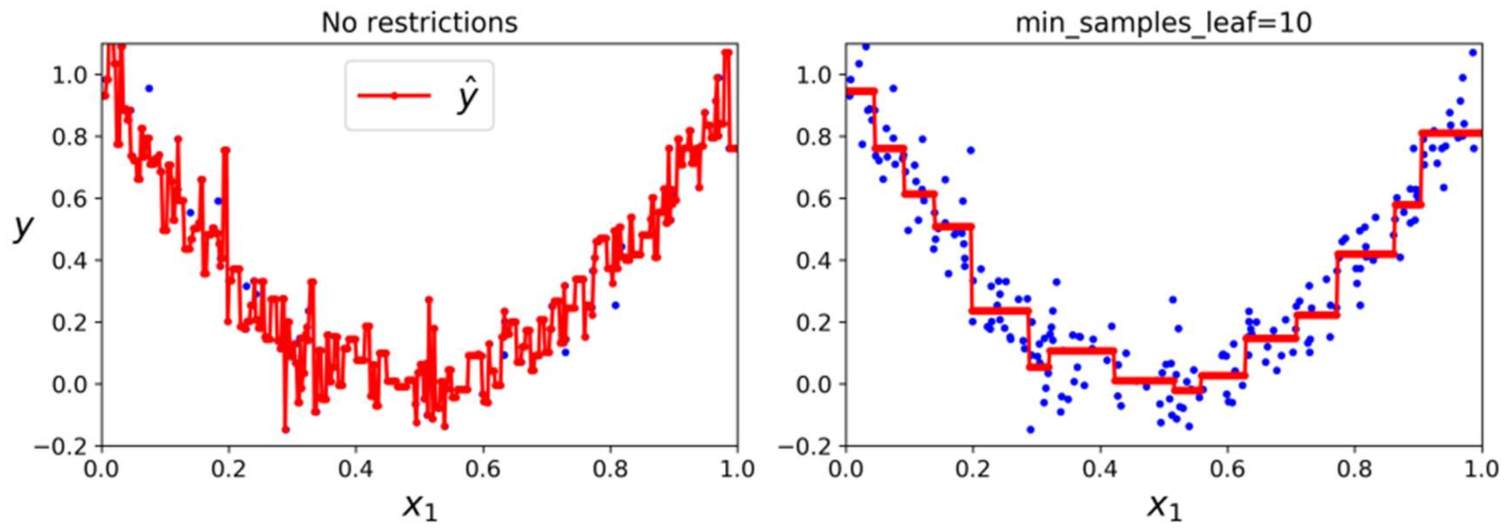
- Decision Trees are also capable of performing regression tasks.
- Regression tree using Scikit-Learn’s **DecisionTreeRegressor** class
- This tree looks very similar to the classification tree. The main difference is that instead of predicting a class in each node, it predicts a value.

Decision Tree for Regression

- For example, suppose you want to make a prediction for a new instance with x1 = 0.6. You traverse the tree starting at the root, and you eventually reach the leaf node that predicts value=0.1106.

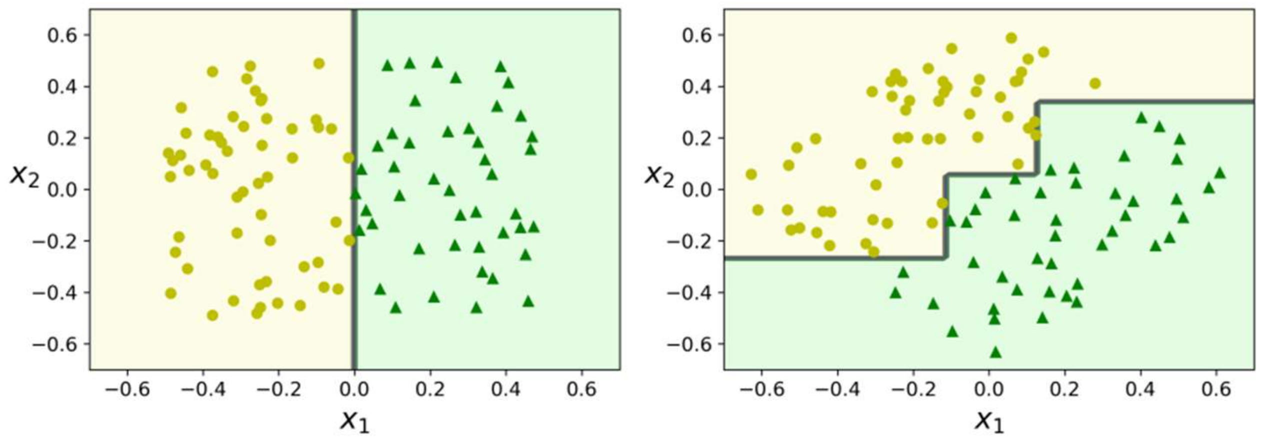


Regularizing a Decision Tree Regressor



Instability

- Decision Trees likes the orthogonal decision boundaries (all splits are perpendicular to an axis)



After the dataset is rotated by 45°, the decision boundary looks unnecessarily convoluted. Although both Decision Trees fit the training set perfectly, it is very likely that the model on the right will not generalize well. One way to limit this problem is to use PCA (see Chapter 8), which often results in a better orientation of the training data.