Decision Trees

Decision trees are versatile machine learning algorithms that can perform classification, regression, and multi-output tasks.

Training and Visualizing a Decision Tree

- We can use the **DecisionTreeClassifier()** class from Scikit-Learn to build a classification model.
- We can also visualize the tree using the export_graphviz() class and then the **Source.from_file()** method from the **graphviz** class to load and display the file.

Making Predictions

- A Decision Tree functions the same way as a binary tree.
    - The root node at depth 0 is at the top.
    - The tree will move down to the left child if the value is smaller than the parent node, and it moves to the right child if the value is greater than the parent node.
    - If the child node is a leaf node, the model assigns the predicted class to that node.
    - The node might be a split node, in which case, the split node is compared against its children and moves left or right.
    - The process repeats until the tree's depth is reached.

```
                root
             /          \
      leaf node      split node
                         /         \
                leaf node     leaf node
```

- Decision trees don't require feature scaling or centering.
- Attributes of each node:
    - The "samples" attribute counts how many instances it applies to.
    - The "value" attribute tells us how many training instances of each class the current node applies to.
    - The "gini" attribute measures the Gini impurity.
        - A node is pure (gini = 0) if all training instances it applies to belong to the same class.
        - The value of Gini equals 1 minus the ratio of each instance that belongs to each class squared.
            - Example: We have three classes, Classes 1, 2, and 3. We have 54 total training instances in this node. Then, the gini impurity is equal to $1 - (0/54)^2 - (49/54)^2 - (5/54)^2 = 0.168$ since 0 of the instances belong to class 1, 49 belong to class 2, and 5 belong to class 3.

- The maximum depth of the tree determines the number of decision boundaries.
- We can use **help(name_of_tree.tree_)** to get information about the tree's structure.
- Decision trees are called white box models since they provide simple classification rules that can be applied manually. On the other hand, black box models, such as random forests and neural networks, provide complex classification rules.

Estimating Class Probabilities

- A decision tree can estimate the probability that an instance belongs to a particular class k.
    - It traverses the tree to find the lead node for this instance and returns the ratio of the training instances of class k in this node.
    - We can use the **predict_proba()** function to calculate this ratio.
    - We can also use the tree's **predict()** function to find the class that has the highest probability.

The CART Training Algorithm

- Scikit-Learn uses the Classification and Regression Tree algorithm to train decision trees (also called "growing" trees).
    - The algorithm splits the training set into two subsets using a single feature k and a threshold t_k.
    - Then, the algorithm searches for the (k, t_k) pair that produces the purest subset, weighted by their size.
    - The algorithm continues splitting the subsets recursively until it reaches the maximum depth or it cannot find a split that will reduce impurity.
- Keep in mind that the CART algorithm is a greedy algorithm, meaning that it will return a reasonably good solution, but not the most optimal.

Computational Complexity

- The overall prediction complexity of a decision tree is $O(\log_2(m))$.
    - The decision tree requires traversing through roughly $O(\log_2(m))$ nodes where m is the number of instances.
- Comparing all the features on all samples at each node results in $O(n \times m \times \log_2(m))$ where n is the number of nodes in the tree.

Gini Impurity vs Entropy

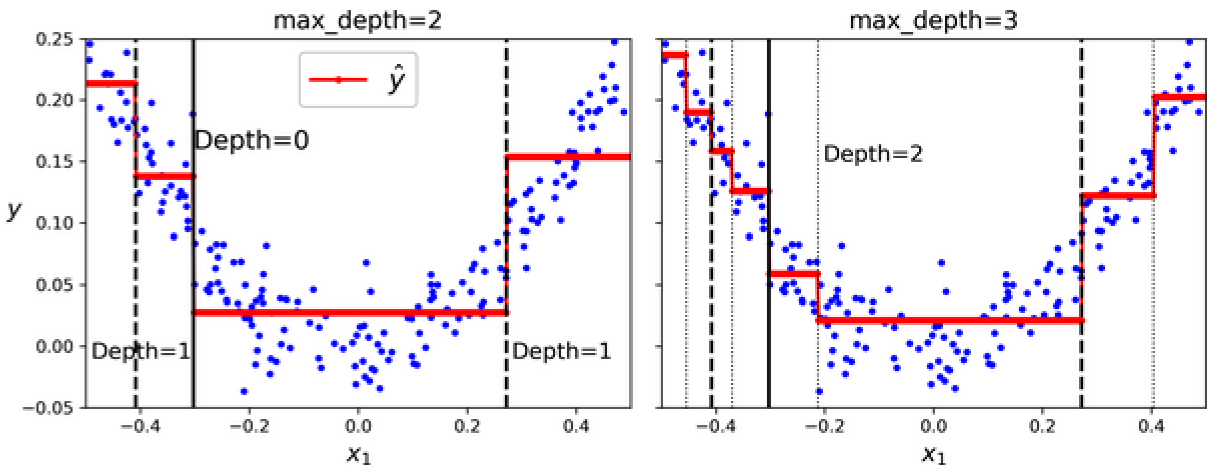
- By default, the DecisionTreeClassifier() class uses the Gini impurity measure.
- We can change this by setting the criterion hyperparameter to "entropy".
- In ML, entropy is frequently used as an impurity measure.
    - A set's entropy is zero when it contains instances of only one class.
- Gini impurity is slightly faster to compute than entropy.
    - If they differ, Gini impurity tends to isolate the most frequent class in its own brance of the tree.
    - Entropy tends to produce slightly more balanced trees.

Regularization Hyperparameters

- 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.
  - There's also nonparametric models, which are unconstrained and will most likely overfit the data.
- Regularization hyperparameters can constrain models. For decision tree classifiers, we have:
  - "max_depth" which determines the maximum number of levels the tree has.
  - "max_features" which sets the maximum number of features that are evaluated for splitting at each node.
  - "max_Leaf_nodes" which control the maximum number of leaf nodes.
  - "min_samples_split" which sets the maximum number of samples a node must have before it can be split.
  - "min_samples_leaf" which controls the maximum number of samples a leaf node must have to be created.
  - "min_weight_fraction_leaf" which does the same thing as min_samples_leaf but expressed as a fraction of the total number of weighted instances.
- Some algorithms first train the decision tree without restrictions, and then deleting unnecessary nodes.
  - A node whose children are all leaf nodes is unnecessary if the purity improvement it provides is not statistically significant.
  - We can use statistical tests like chi-squared tests to calculate the p-value, which tells us whether the purity improvement is statistically significant or not.
- We can use the tree's **score()** method to get the accuracy of the tree's predictions.

Decision Tree Regression

- We can use the **DecisionTreeRegressor()** class to perform regression tasks.
- Instead of predicting a class in each node, the tree predicts a value.
  - The prediction value is the average target value of all the training instances in this leaf node.



- In regression, the CART algorithm splits the training set in a way that minimizes the mean squared error (MSE).

Problems with Decision Trees

- Decision trees have some weaknesses.
    - Decision trees love orthogonal decision boundaries (all splits are perpendicular to the axis), which makes them sensitive to the data's orientation.
        - To solve this issue, we can scale the data and rotate it to reduce the correlation between features.
        - We can use the **PCA()** class and a StandardScaler() in a pipeline to rotate the data.
    - Decision trees have a high variance.
        - Small changes to the hyperparameters or to the data may produce very different models.
        - Even retraining the same decision tree on the same data may produce different models unless the random_state hyperparameter is set.
        - By averaging predictions over many trees, we can reduce the variance significantly. This ensemble of trees is called a random forest.