DECISION TREE API

KABIR CHATURVEDI

J011

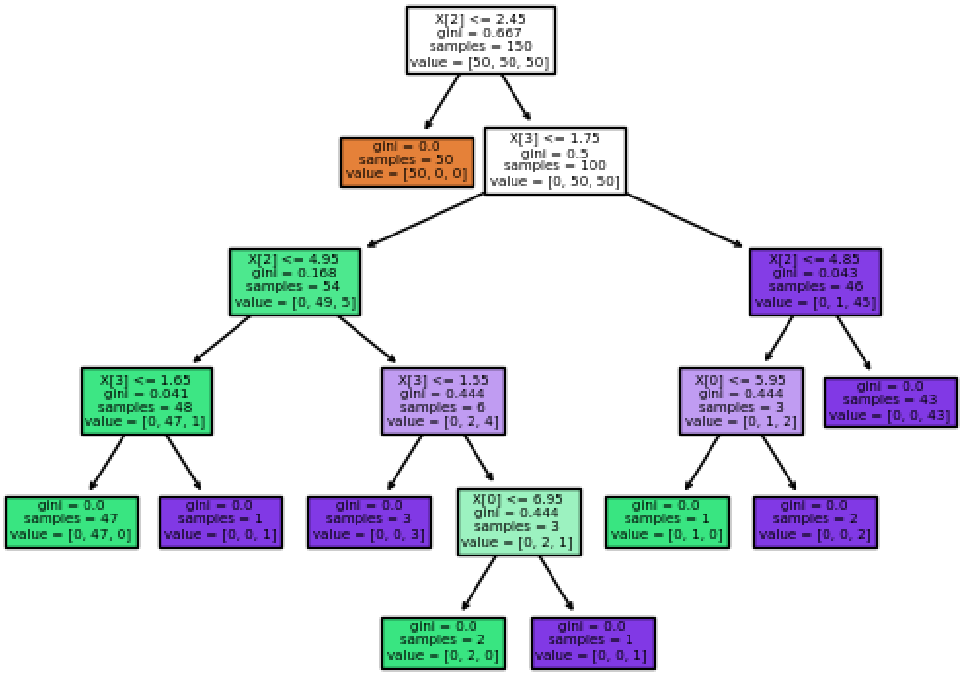
Decision Tree :

Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

API:

class sklearn.tree.DecisionTreeClassifier(*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, ccp_alpha=0.0)

- **CODE:**
  - `from sklearn.datasets import load_iris`
  - `from sklearn import tree`
  - `iris = load_iris()`
  - `X, y = iris.data, iris.target`
  - `clf = tree.DecisionTreeClassifier()`
  - ` clf = clf.fit(X, y)`



- **Important Parameters are:-**

- 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. A very small number will usually mean the tree will overfit, whereas a large number will prevent the tree from learning the data. Try min_samples_leaf=5 as an initial value. If the sample size varies greatly, a float number can be used as percentage in these two parameters.
- Splitter – determines how the decision tree searches the features for a split. Default value is set to ‘best’ but could be changed to ‘random’.
- Max_depth – will determine the maximum depth of a tree. The default value is none but this should be regularised to prevent overfitting.
- Min_samples_split – minimum number of samples a node must contain to consider splitting.
- Min_sample_leaf- minimum number of samples needed to be considered a lead node. The default value is set to one.

▪ Decision tree working:

Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

- **Important Methods : -**

Fit(X, y)- fit the linear model.

Predict(X)-predict using linear model.

Score(X,y)-returns the coefficient of determination R^2 of the prediction.

- **Advantages :**

- Simple to understand and to interpret. Trees can be visualised.
- Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed.
- Able to handle both numerical and categorical data. However scikit-learn implementation does not support categorical variables for now. Other techniques are usually specialised in analysing datasets that have only one type of variable.
- Able to handle multi-output problems.
- Uses a white box model.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.

- **Disadvantages:**

- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.

- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated.
- Predictions of decision trees are neither smooth nor continuous, but piecewise constant approximations as seen in the above figure. Therefore, they are not good at extrapolation.
- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts.
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.