Decision Tree Classifier

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, class_weight=None, ccp_alpha=0.0)

```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
```

```
clf=DecisionTreeClassifier(random_state=101)
iris=load_iris()
cross_val_score(clf,iris.data,iris.target,cv=10).mean()
```

```
0.9533333333333334
```

Parameters :

- **Criterion : *{“gini”, “entropy”}, default=”gini”* -** *The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.*
- **Splitter : *{“best”, “random”}, default=”best”* -** *The strategy used to choose the split at each node. Supported strategies are “best” to choose the best split and “random” to choose the best random split.*
- **max_depth : *int, default=None* -** *The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.*
- **min_samples_split : *int or float, default=2* -** *The minimum number of samples required to split an internal node.*
- **min_samples_leaf : *int or float, default=1* -** *The minimum number of samples required to be at a leaf node.*
- **min_weight_fraction_leaf : *float, default=0.0* -** *The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node.*
- **max_features : *int, float or {“auto”, “sqrt”, “log2”}, default=None* -** *The number of features to consider when looking for the best split.*
- **random_state : *int, RandomState instance or None, default=None* -** *Controls the randomness of the estimator.*
- **max_leaf_nodes : *int, default=None* -** *Grow a tree with max_leaf_nodes in best-first fashion.*
- **min_impurity_decrease : *float, default=0.0* -** *A node will be split if this split induces a decrease of the impurity greater than or equal to this value.*
- **class_weight : *dict, list of dict or “balanced”, default=None* -** *Weights associated with classes in the form {class_label: weight}.*
- **ccp_alpha : *non-negative float, default=0.0* -** *Complexity parameter used for Minimal Cost-Complexity Pruning.*

Attributes :

- **classes_ndarray of shape (n_classes,) or list of ndarray -***The classes labels (single output problem), or a list of arrays of class labels (multi-output problem).*
- **feature_importances_ndarray of shape (n_features,) -** *Return the feature importances.*
- **max_features_int -** *The inferred value of max_features*.
- **n_classes_int or list of int -** *The number of classes (for single output problems), or a list containing the number of classes for each output (for multi-output problems).*
- **n_features_in_int -** *Number of features seen during fit.*
- **feature_names_in_ndarray of shape (n_features_in_,) -** *Names of features seen during fit. Defined only when X has feature names that are all strings.*
- **n_outputs_int -** *The number of outputs when fit is performed.*
- **tree_Tree instance -***The underlying Tree object.*

Advantages :

1. Simple to understand and to interpret. Trees can be visualized.
2. Requires little data preparation.
3. The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
4. Able to handle both numerical and categorical data.
5. Able to handle multi-output problems.
6. Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
7. Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

Disadvantages :

1. Decision-tree learners can create over-complex trees that do not generalize the data well. This is called overfitting.
2. Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
3. 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.
4. 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.