DECISION TREE API ASSIGNMENT

NAME: SAMEEKSHA KINI

ROLL NO: J029

A decision tree is a graphical representation of all possible solutions to a decision based on certain conditions. On each step or node of a decision tree, used for classification, we try to form a condition on the features to separate all the labels or classes contained in the dataset to the fullest purity.

CODE:-

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)

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:

a) If int, then consider min_samples_split as the minimum number.
b) If float, then min_samples_split is a fraction and ceil(min_samples_split * n_samples) are the minimum number of samples for each split.

- **min_samples_leaf: int or float, default=1**
  The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf

training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

a) If int, then consider min_samples_leaf as the minimum number.
b) If float, then min_samples_leaf is a fraction and ceil(min_samples_leaf * n_samples) are the minimum number of samples for each 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:

a) If int, then consider max_features features at each split.
b) If float, then max_features is a fraction and int(max_features * n_features) features are considered at each split.
c) If "auto", then max_features = sqrt(n_features)
d) If "sqrt", then max_features = sqrt(n_features)
e) If "log2", then max_features = log2(n_features)
f) If None, then max_features = n_features

- **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.

- **min_impurity_split: float, default=0**
  Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf.

- **class_weight: dict, list of dict or "balanced", default=None**
  Weights associated with classes in the form {class_label: weight}. If None, all classes are supposed to have weight one.

- **ccp_alpha: non-negative float, default=0.0**
  Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than ccp_alpha will be chosen.

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_: int**
  The number of features when fit is performed.

- **n_outputs_:int**
  The number of outputs when fit is performed.

- **tree_: Tree instance**
  The underlying Tree object.

ADVANTAGES OF DECISION TREES:-

- They are simple to understand, interpret as well as visualise.
- This algorithm requires less effort for data preparation.
- Requires neither scaling nor normalization of the data.
- Missing values do not affect the process of building a decision tree.

DISADVANTAGES OF DECISION TREES:-

- Model training requires higher time.
- Calculations performed here can be more complex than other algorithms.
- This model can be unstable because small variations in data might result in a completely different tree being generated.