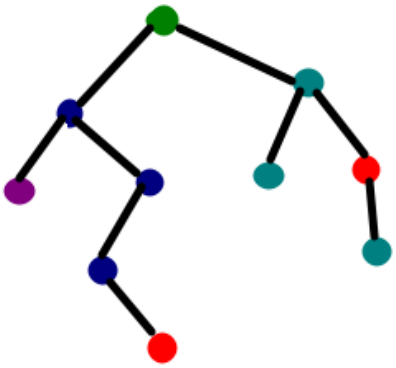
Decision tree implementation using Python

Prerequisites: Decision Tree, DecisionTreeClassifier, sklearn, numpy, pandas
Decision Tree is one of the most powerful and popular algorithm. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.



In this article, We are going to implement a Decision tree algorithm on the *Balance Scale Weight & Distance Database* presented on the UCI.

Data-set Description :

```
Title          : Balance Scale Weight & Distance Database

Number of Instances: 625 (49 balanced, 288 left, 288 right)

Number of Attributes: 4 (numeric) + class name = 5

Attribute Information:
  1. Class Name (Target variable): 3
  •  L [balance scale tip to the left]
  •  B [balance scale be balanced]
  •  R [balance scale tip to the right]
  2. Left-Weight: 5 (1, 2, 3, 4, 5)
  3. Left-Distance: 5 (1, 2, 3, 4, 5)
  4. Right-Weight: 5 (1, 2, 3, 4, 5)
  5. Right-Distance: 5 (1, 2, 3, 4, 5)

Missing Attribute Values: None
Class Distribution:
  1. 46.08 percent are L
  2. 07.84 percent are B
  3. 46.08 percent are R

You can find more details of the dataset here.
```

Used Python Packages :

1. **sklearn :**
   - In python, sklearn is a machine learning package which include a lot of ML algorithms.
   - Here, we are using some of its modules like train_test_split, DecisionTreeClassifier and accuracy_score.
2. **NumPy :**
   - It is a numeric python module which provides fast maths functions for calculations.
   - It is used to read data in numpy arrays and for manipulation purpose.
3. **Pandas :**
   - Used to read and write different files.
   - Data manipulation can be done easily with dataframes.

Installation of the packages :

In Python, sklearn is the package which contains all the required packages to implement Machine learning algorithm. You can install the sklearn package by following the commands given below.

using pip :

```
pip install -U scikit-learn
```

Before using the above command make sure you have *scipy* and *numpy* packages installed.
If you don't have pip. You can install it using

```
python get-pip.py
```

using conda :

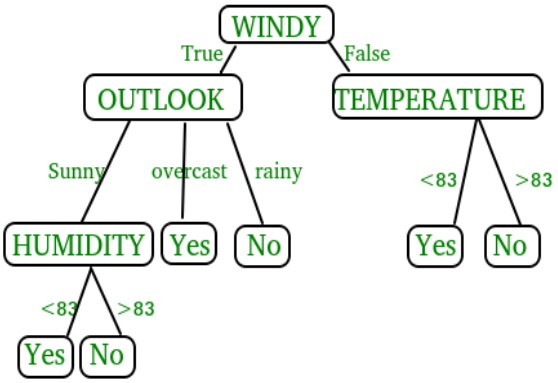
```
conda install scikit-learn
```

Assumptions we make while using Decision tree :

- At the beginning, we consider the whole training set as the root.
- Attributes are assumed to be categorical for information gain and for gini index, attributes are assumed to be continuous.
- On the basis of attribute values records are distributed recursively.
- We use statistical methods for ordering attributes as root or internal node.

Pseudocode :

1. Find the best attribute and place it on the root node of the tree.
2. Now, split the training set of the dataset into subsets. While making the subset make sure that each subset of training dataset should have the same value for an attribute.
3. Find leaf nodes in all branches by repeating 1 and 2 on each subset.



While implementing the decision tree we will go through the following two phases:

4. Building Phase
   - Preprocess the dataset.
   - Split the dataset from train and test using Python sklearn package.
   - Train the classifier.
5. Operational Phase
   - Make predictions.
   - Calculate the accuracy.

Data Import :

- To import and manipulate the data we are using the *pandas* package provided in python.
- Here, we are using a URL which is directly fetching the dataset from the UCI site no need to download the dataset. When you try to run this code on your system make sure the system should have an active Internet connection.
- As the dataset is separated by “,” so we have to pass the sep parameter’s value as “,”.
- Another thing is notice is that the dataset doesn’t contain the header so we will pass the Header parameter’s value as none. If we will not pass the header parameter then it will consider the first line of the dataset as the header.

Data Slicing :

- Before training the model we have to split the dataset into the training and testing dataset.
- To split the dataset for training and testing we are using the sklearn module *train_test_split*
- First of all we have to separate the target variable from the attributes in the dataset.

```
X = balance_data.values[:, 1:5]

Y = balance_data.values[:,0]
```

- Above are the lines from the code which separate the dataset. The variable X contains the attributes while the variable Y contains the target variable of the dataset.
- Next step is to split the dataset for training and testing purpose.

```
X_train, X_test, y_train, y_test = train_test_split(

          X, Y, test_size = 0.3, random_state = 100)
```

- Above line split the dataset for training and testing. As we are splitting the dataset in a ratio of 70:30 between training and testing so we are pass *test_size* parameter's value as 0.3.
- *random_state* variable is a pseudo-random number generator state used for random sampling.

Terms used in code :

Gini index and information gain both of these methods are used to select from the *n* attributes of the dataset which attribute would be placed at the root node or the internal node.

Gini index

$$\text{Gini Index} = 1 - \sum_j P_j^2$$

- Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified.
- It means an attribute with lower gini index should be preferred.
- Sklearn supports "gini" criteria for Gini Index and by default, it takes "gini" value.

Entropy

if a random variable x can take N different value ,the i^{th} value x_i with probability $p(x_{ii})$,we can assocoate the folloeing entropy with x:

$$H(x) = -\sum_{i=1}^{N} p(x_i) \log_2 p(x_i)$$

- Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy the more the information content.

Information Gain

Definition: Suppose S is a set of instances, A is an attribute, S_v is the subset of s with A = v and Values(A) is the set of all possible of A,then

$$Gain(S,A) = Entropy(S) - \sum_{v\ :va(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$|S|$ denotes the size of set S

- The entropy typically changes when we use a node in a decision tree to partition the training instances into smaller subsets. Information gain is a measure of this change in entropy.
- Sklearn supports “entropy” criteria for Information Gain and if we want to use Information Gain method in sklearn then we have to mention it explicitly.

Accuracy score

- Accuracy score is used to calculate the accuracy of the trained classifier.

Confusion Matrix

- Confusion Matrix is used to understand the trained classifier behavior over the test dataset or validate dataset.