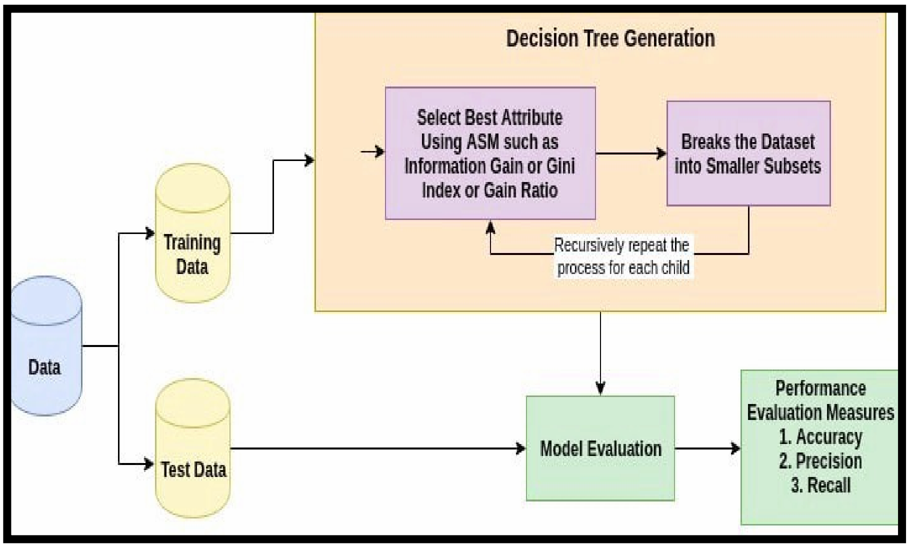
WEEK-3

AIM: Implement ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

DESCRIPTION:

Decision Tree is a **Supervised learning technique** that can be used for both **classification and Regression problems**, but mostly it is preferred for solving **Classification problems.** It is a **tree-structured classifier**, where **internal nodes** represent the **features of a dataset**. **Branches represent the decision rules. Each leaf node represents the outcome.**

Decision Tree Generation:



Decision Tree Algorithm:

Step-1: Begin the tree with the root node, says S, which contains the **complete dataset**.

Step-2: Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**

Step-3: **Divide the S into subsets** that contains possible values **for the best attributes.**

Step-4: **Generate the decision tree node**, which **contains the best attribute.**

Step-5: **Recursively make new decision trees** using the subsets of the dataset created **in step -3.** Continue this process until a **stage is reached** where you **cannot further classify** the **nodes** and called the **final node as a leaf node.**

Attribute Selection Measure:

It is a heuristic for **selecting the splitting criterion** that **"best" separates a given data partition, D**, of a **class-labeled training tuples into individual classes.**

The **Three Important attribute Selection measures** are

Information gain: ID3/C4.5

Gain Ratio : C4.5

Gini Index CART

ID3 Algorithm: Entropy is the main concept of this algorithm, which helps determine a feature or attribute that gives maximum information about a class is called Information gain or ID3 algorithm.

Steps in ID3 algorithm:

1. It **begins with the original set S as the root node.**
2. On **each iteration of the algorithm**, it iterates through the very unused attribute of the set S and **calculates Entropy(H) and Information gain(IG)** of this attribute.
3. It then selects the attribute which has the **smallest Entropy or Largest Information gain.**
4. The **set S is then split by the selected attribute** to produce a subset of the data.

Entropy: By using this method, we can **reduce the level of entropy** from the **root node to the leaf node.**

$$\boxed{E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i}$$

$$\textbf{Entropy = - (p(0) * log(P(0)) + p(1) * log(P(1)))}$$

Where $\mathbf{p_i}$ is the probability of **randomly picking an element of class i** (i.e. the proportion of the dataset made up of class i).

Information Gain: Information gain is used for determining the best features/attributes that render maximum information about a class. It follows the concept of entropy while aiming at decreasing the level of entropy, beginning from the root node to the leaf nodes.

Information Gain = entropy (parent) – [average entropy (children)]

Information Gain = Entropy(S) – [(weighted Avg.) * Entropy (Each Feature)]

It involves below steps:

1. Install the Packages:

(a) Pip: it is a standard package manager used to install and maintain packages for Python. The Python standard library comes with a collection of built-in functions and built-in packages. **Note:** If you have Python version 3.4 or later, PIP is included by default.

```
!pip installs decision-tree-id3
```

(b) Six: **It is a Python 2 and 3 compatibility library**. It provides **utility functions for smoothing over the differences between the Python versions** with the goal of writing Python code that is **compatible on both Python versions.**

```
import six
```

(c) Sys : It is a **sys module** in Python provides **various functions and variables** that are used to **manipulate different parts** of the **Python runtime environment.**

```
import sys
sys.modules ['sklearn.externals.six'] = six
```

(d) ID3 Estimator: **decision-tree-id3** is a module created to derive decision trees using the **ID3 algorithm**. It is written to be compatible with **Scikit-learn's API**

2. Importing datasets:

read_csv() function: Now to import the dataset, we will use **read_csv() function** of **pandas library**, which is used to read a csv file and performs various operations on it. Using this function, we can read a csv file locally as well as through an URL. **We can use read_csv function as below:**

For Eg: tennis_data = pd.read_csv ('PlayTennis.csv')

Here, **tennis_data** is a name of the variable to store our dataset, and inside the function, we have passed the name of our dataset.

And also to print the head () : The head() returns the first n rows for the object based on position. **tennis_data.head(5)**

3. Extracting Independent and Dependent Variables:

In machine learning, it is important to distinguish the matrix of features (independent variables) and dependent variables from dataset.

For Eg: In our dataset, there are **FOUR independent variables** that are **Outlook, Temperature, Humidity, Wind**, and one is a dependent variable which is **Play Tennis**

4. Encoding Categorical Data: Since machine learning model completely works on mathematics and numbers, but if our dataset would have a categorical variable, then it may create trouble while building the model. So it is necessary to encode these categorical variables into numbers.

For Independent variable: Firstly, we will convert the country variables into categorical data. So to do this, we will use **LabelEncoder()** class from **pre-processing** library.
Categorical data is data which has some categories such as, in our dataset; there are four categorical variable, **Outlook, Temperature, Humidity, Wind**

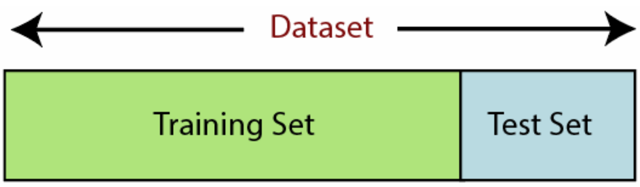
```
from sklearn.preprocessing import LabelEncoder
Le = Label Encoder()
# converting each column
tennis_data ['Outlook'] = Le.fit_transform(tennis_data['Outlook'])
```

```
tennis_data['Temperature'] = Le.fit_transform(tennis_data['Temperature'])
tennis_data['Humidity'] = Le.fit_transform(tennis_data['Humidity'])
tennis_data['Wind'] = Le.fit_transform(tennis_data['Wind'])
```

Separating independent and Dependent Variable:

```
y = tennis_data['Play Tennis']
X = tennis_data. Drop(['Play Tennis'],axis=1)
```

5. Splitting dataset into training and test set: we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model.



Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

For **splitting the dataset, we will use the below lines of code:**

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.9, random_state = 0)
```

Explanation:

- In the above code, the first line is used for splitting arrays of the dataset into random train and test subsets.
- In the second line, we have used four variables for our output that are
  - x_train: features for the training data
  - x_test: features for testing data
  - y_train: Dependent variables for training data
  - y_test: dependent variable for testing data

6. Decision Tree Creation: (Print Text Representation): Here we will create a random decision tree with the help of **sci-kit learn library.** We will use the **Play Tennis dataset** for decision tree creation.

1. Firstly, We need to import the **DecisionTreeClassifier** from s**klearn.tree module**.

2. **<u>Invoking sklearn export text –</u> O**nce we have created the decision tree, We can export the decision tree into textual format. But to achieve this, we need to **import export_text** from **sklearn.tree.export package.** After it, We will invoke the export_text() function by passing the decision tree **object as an argument**. we can easily solve the mystery of the decision tree with the above self-explanatory **export_text() function**. Here **show_weights are set are True**. It will give more info about each node. Let's run the complete code together and check the output.

from sklearn import tree

print(tree.export_text(clf))

```
|--- feature_0 <= 0.50
|   |--- class: 1
|--- feature_0 >  0.50
|   |--- feature_2 <= 0.50
|   |   |--- feature_0 <= 1.50
|   |   |   |--- feature_3 <= 0.50
|   |   |   |   |--- class: 0
|   |   |   |--- feature_3 >  0.50
|   |   |   |   |--- class: 1
|   |   |--- feature_0 >  1.50
|   |   |   |--- class: 0
|   |--- feature_2 >  0.50
|   |   |--- feature_3 <= 0.50
|   |   |   |--- feature_0 <= 1.50
|   |   |   |   |--- class: 0
|   |   |   |--- feature_0 >  1.50
|   |   |   |   |--- class: 1
|   |   |--- feature_3 >  0.50
|   |   |   |--- class: 1
```

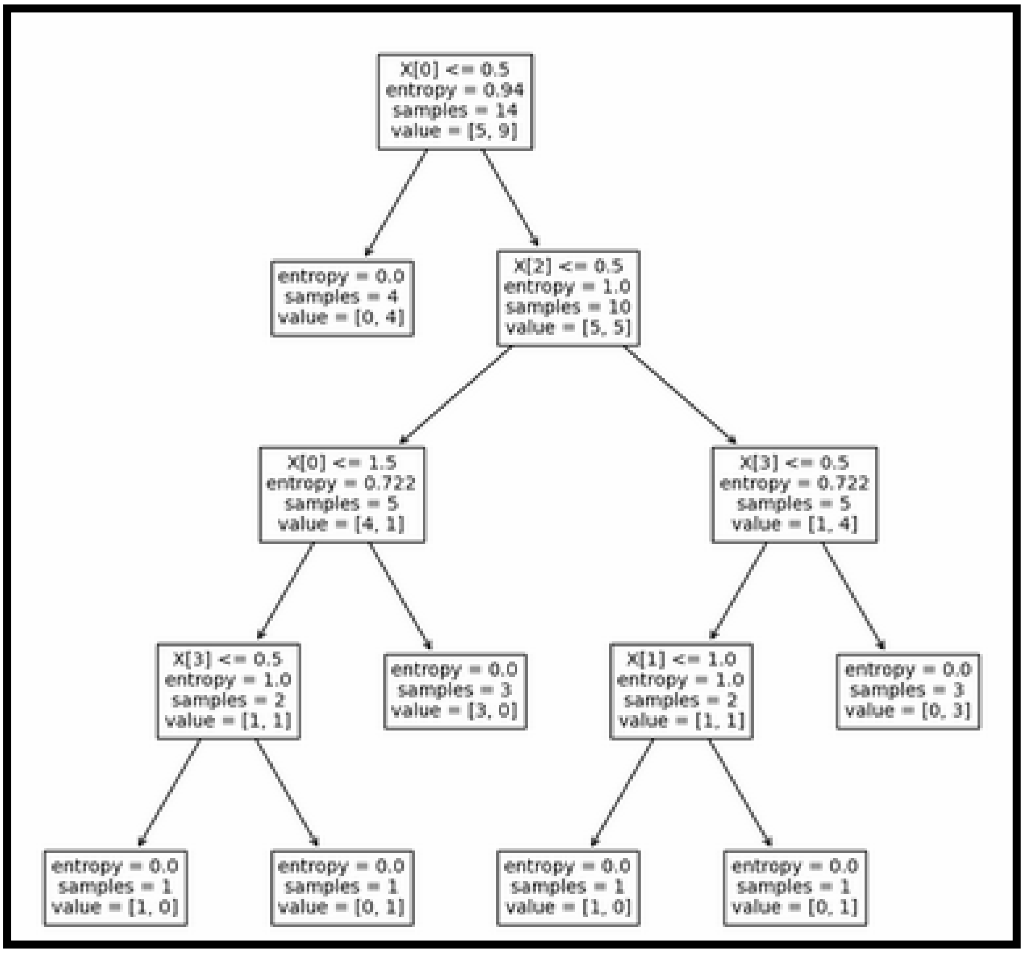
<u>7. Decision Tree Classification(Plot-Tree):</u>

The **plot_tree method** was added to sklearn**.** It requires **matplotlib** to be installed. It allows us to easily produce figure of the tree. The more information **about plot_tree arguments** are in the docs.

```
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(x_train, y_train)
clf.fit(x,y)
```

Here, we can also **visualize the size of the Tree** and also include the **Font Size.**

```
fig, ax = plt.subplots(figsize=(10, 10))
tree.plot_tree(clf, fontsize=10)
plt.show()
```



8. Fitting the Model & Predictions:

Here we can use **Python's sklearn library** holds tons of **modules that help to build predictive models.** It contains tools for **data splitting, pre-processing, feature selection, tuning and supervised – unsupervised learning algorithms.**

Now we fit Decision tree algorithm on training data, predicting labels for validation dataset and printing the accuracy of the model using various parameters.

DecisionTreeClassifier(): This is the classifier function for **DecisionTree.** It is the main function for implementing the algorithms. Here we can apply the Criteria is **" Entropy"**

```
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(x_train, y_train)
#predections
x_pred = clf.predict(x_test)
```

Classification on Decision Tree:

#classification report to check accuracy, precision, recall etc.

```
from sklearn.metrics import classification_report
print(classification_report(y_test, X_pred))
```

Import the ID3 Algorithm: **Here we can calculate the Accuracy Score.** The function **accuracy_score()** will be used to **print accuracy of Decision Tree algorithm**. By accuracy, we mean the **ratio of the correctly predicted data points to all the predicted data points**. **Accuracy as a metric** helps to understand the **effectiveness of our algorithm.**

```
# import the ID3 Estimator
from id3 import Id3Estimator
estimator = Id3Estimator()
estimator.fit (X_train, y_train)
X_pred = estimator.predict(X_test)
#showing classification report
print(classification_report(y_test, X_pred))
print(accuracy_score(y_test, X_pred))
```

PROGRAM:

```
#Importing lib for data pre-processing and algorithm building
!pip install decision-tree-id3
import matplotlib.pyplot as plt
import pandas as pd  # for reading data set
import six
import sys
sys.modules['sklearn.externals.six'] = six
from id3 import Id3Estimator
%matplotlib inline
#Reading tennis data set
tennis_data = pd.read_csv('PlayTennis.csv')
#showing first 5 records
tennis_data.head(5)
#converting data to numeric
# we have different approaches to convert to numeric values LabelEncoder is one of the most used technique
from sklearn.preprocessing import LabelEncoder
Le = LabelEncoder()
# converting each column
tennis_data['Outlook'] = Le.fit_transform(tennis_data['Outlook'])
tennis_data['Temperature'] = Le.fit_transform(tennis_data['Temperature'])
tennis_data['Humidity'] = Le.fit_transform(tennis_data['Humidity'])
tennis_data['Wind'] = Le.fit_transform(tennis_data['Wind'])
tennis_data['Play Tennis'] = Le.fit_transform(tennis_data['Play Tennis'])
#seprating target and features
y = tennis_data['Play Tennis']
X = tennis_data.drop(['Play Tennis'],axis=1)
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.9, random_state = 0)
```

 # Fitting the model

```
from sklearn import tree
print(tree.export_text(clf))
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(x_train, y_train)
clf.fit(x,y)
fig, ax = plt.subplots(figsize=(10, 10))
tree.plot_tree(clf, fontsize=10)
plt.show()
```

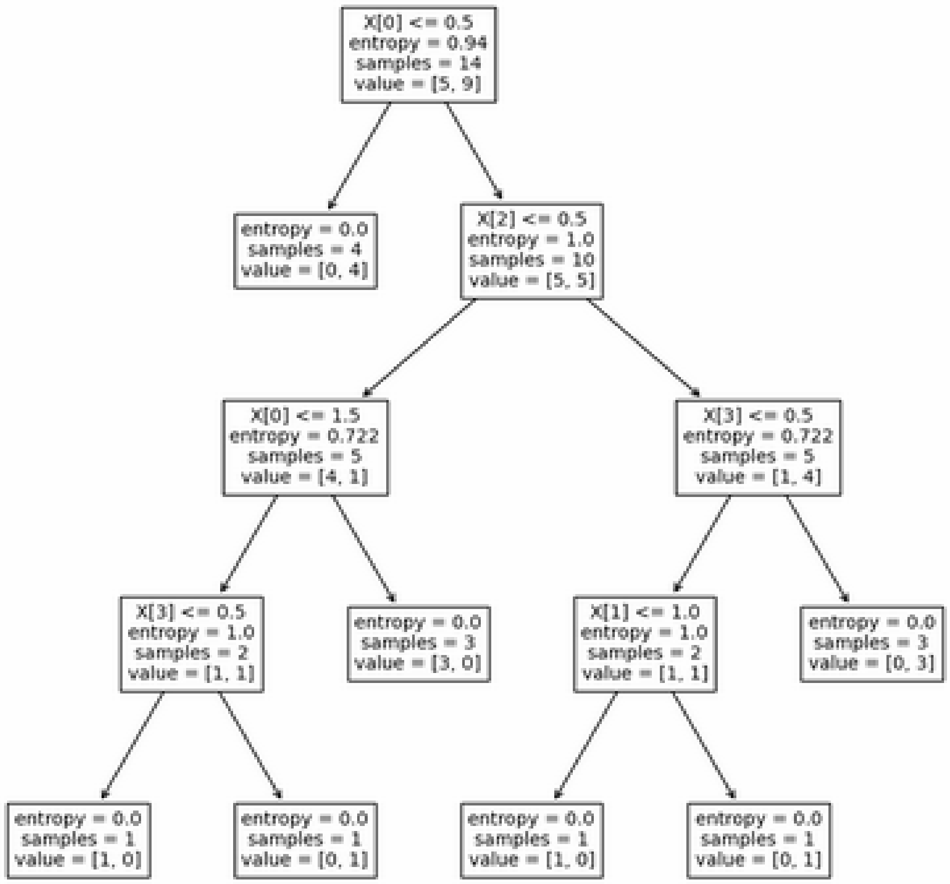
#predections

```
X_pred = clf.predict(X_test)
```

#classification report to check accuracy,precision,recall etc.

```
#importing ID3
from id3 import Id3Estimator
estimator = Id3Estimator()
estimator.fit(X_train, y_train)
X_pred = estimator.predict(X_test)
#showing classification report
print(classification_report(y_test, X_pred))
print(accuracy_score(y_test, X_pred))
```

INPUT/OUTPUT:




```
print(accuracy_score(y_test, X_pred))
```

```
0.6153846153846154
```

CONCLUSION: **Program is executed successfully without any error.**