Decision Tree

Decision trees are a powerful and versatile technique widely used in machine learning, data mining, and statistics. They excel at identifying subtle correlations among numerous components in a dataset, providing a clear depiction that enables users to make data-driven decisions easily and transparently.

A decision tree resembles a flowchart used for making decisions or predictions. It comprises nodes representing attribute evaluations or tests, branches indicating the outcomes of these decisions, and leaf nodes representing final outcomes or predictions. Each internal node corresponds to an attribute test, each branch to the test result, and each leaf node to a class label or continuous value.

Structure of a Decision Tree:

i. **Root Node:** Represents the entire dataset and the initial decision to be made.
ii. **Internal Nodes:** Represent attribute-based decisions or tests. Each internal node has one or more branches.
iii. **Branches:** Represent the results of a decision or test, leading to another node.
iv. **Leaf Nodes:** Represent the final conclusion or prediction, with no further splits at these nodes.

Decision Tree ~ Algorithm

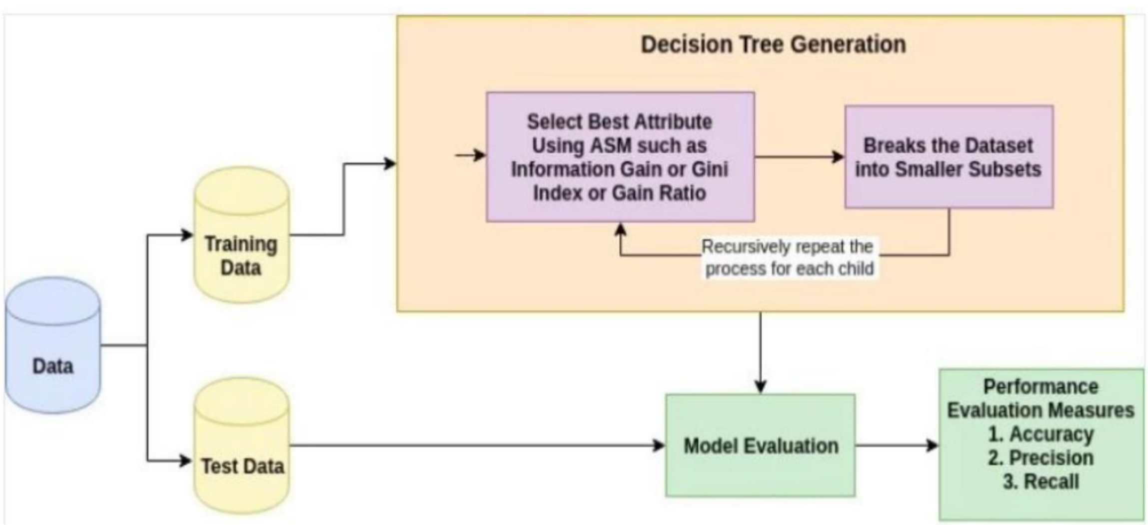
Step 1: Start the tree with the root node, denoted as S, which contains the complete dataset.

Step 2: Identify the best attribute in the dataset using an Attribute Selection Measure.

Step 3: Split S into subsets based on the possible values of the best attribute.

Step 4: Create a decision tree node representing the best attribute.

Step 5: Use the subsets of the dataset from step 3 to recursively construct new decision trees. Continue this process until further classification is not possible, at which point the final nodes are called leaf nodes.



Attribute Selection Measure of Decision Tree

The attribute selection measure in decision trees evaluates how effectively different attributes split the data at each node. Here are the key measures:

- **Gini Impurity:** Measures the probability of a new instance being incorrectly classified if it were classified randomly based on the dataset's class distribution.

$$\mathbf{Gini=1-\sum_{i=1}^{n} (p_i)^2}$$

Where p_i is the probability of an instance being classified into a particular class.

- **Entropy**: Determines the level of uncertainty or impurity in the dataset.

$$\mathbf{Entropy=-\sum_{i=1}^{n} p_i \log_2 (p_i)}$$

Where p_i is the probability of an instance being classified into a particular class.

- **Information Gain**: Measures the reduction in entropy or Gini impurity after splitting a dataset by an attribute.

$$\mathbf{Information\ Gain = Entropy_{parent} - \sum_{i=1}^{n} \left(\frac{|D_i|}{|D|} * Entropy(D_i)\right)}$$

Where D_i is the subset of D after splitting by an attribute.

CODE ON JUPYTER NOTEBOOK

In this lab, we'll utilize the dataset 'play_tennis.csv' to describe weather conditions and whether tennis was played on different days. First, we import all of the essential libraries for the decision tree particle lab.

```
[6]: import numpy as np
```

```
[7]: import pandas as pd
     import matplotlib.pyplot as plt
     from sklearn.tree import DecisionTreeClassifier,plot_tree
     from sklearn.model_selection import train_test_split
     from sklearn import tree
     from sklearn.preprocessing import LabelEncoder
```

```
[8]: data= pd.read_csv("play_tennis.csv")
```

```
[9]: data
```

Here, we have used this terms for specific reasons such as:

- Numpy and pandas for numerical operations and data manipulation.
- Matplotlib.pyplot for plotting and visualization.
- DecisionTreeClassifier and plot_tree from sklearn.tree for creating and visualizing decision trees.
- Train_test_split from sklearn.model_selection for splitting the data into training and test sets.
- LabelEncoder from sklearn.preprocessing for encoding categorical labels.
- Cell loads a dataset named play_tennis.csv into a pandas DataFrame called data.
- Displays the contents of the data DataFrame, which contains the play_tennis.csv dataset. The dataset has columns for day, outlook, temp, humidity, wind, and play.

[9]:

	day	outlook	temp	humidity	wind	play
0	D1	Sunny	Hot	High	Weak	No
1	D2	Sunny	Hot	High	Strong	No
2	D3	Overcast	Hot	High	Weak	Yes
3	D4	Rain	Mild	High	Weak	Yes
4	D5	Rain	Cool	Normal	Weak	Yes
5	D6	Rain	Cool	Normal	Strong	No
6	D7	Overcast	Cool	Normal	Strong	Yes
7	D8	Sunny	Mild	High	Weak	No
8	D9	Sunny	Cool	Normal	Weak	Yes
9	D10	Rain	Mild	Normal	Weak	Yes
10	D11	Sunny	Mild	Normal	Strong	Yes
11	D12	Overcast	Mild	High	Strong	Yes
12	D13	Overcast	Hot	Normal	Weak	Yes
13	D14	Rain	Mild	High	Strong	No

The dataset play_tennis.csv contains information about weather conditions and whether tennis was played on different days. Here's a brief explanation of the columns:

- day: Identifier for the day (e.g., D1, D2, etc.)
- outlook: The weather outlook (e.g., Sunny, Overcast, Rain)
- temp: Temperature (e.g., Hot, Mild, Cool)
- humidity: Humidity level (e.g., High, Normal)
- wind: Wind conditions (e.g., Weak, Strong)
- play: The target variable indicating whether tennis was played (Yes or No)

To proceed, we'll create a decision tree classifier to predict whether tennis will be played based on the weather conditions.

Decision Tree Classifier

- Preprocessing: Convert categorical variables to numeric values.
- Train-Test Split: Split the data into training and testing sets.
- Model Training: Train the decision tree classifier on the training data.
- Prediction and Evaluation: Evaluate the model's performance on the test data.

Let's begin with these steps in the Jupyter Notebook.

Preprocessing

We'll convert the categorical variables into numerical representations using LabelEncoder.

The categorical variables have been successfully encoded into numerical values:
outlook: Sunny = 2, Overcast = 0, Rain = 1
temp: Hot = 1, Mild = 2, Cool = 0
humidity: High = 0, Normal = 1
wind: Weak = 1, Strong = 0
play: No = 0, Yes = 1
Train-Test Split

Next, we split the data into training and testing sets.

Model Training

We'll train a decision tree classifier on the training data.

Prediction and Evaluation

We'll evaluate the model's performance on the test data. The decision tree classifier achieved perfect accuracy on the test set. Here's a summary of the evaluation:

[10]:
```
label_encoder =LabelEncoder()
```

[11]:
```
data.select_dtypes(include=['object']).columns.tolist()
```

[11]:
```
['day', 'outlook', 'temp', 'humidity', 'wind', 'play']
```

[12]:
```
categorical_cols=data.select_dtypes(include=['object']).columns.tolist()
```

[13]:
```
for col in categorical_cols:
   data[col]=label_encoder.fit_transform(data[col])
```

[14]:
```
data
```

```
[15]: X=data.drop(['play','day'],axis=1)
      y=data['play']
```

```
[17]: y
```

```
[17]: 0     0
      1     0
      2     1
      3     1
      4     1
      5     0
      6     1
      7     0
      8     1
      9     1
      10    1
      11    1
      12    1
      13    0
      Name: play, dtype: int32
```

- This cell prepares the feature matrix X by dropping the play and day columns from the dataset.
- The target vector y is assigned to the play column, which contains the labels for whether to play tennis.

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, stratify = y,random_state=2)
```

```
X_train
```

	outlook	temp	humidity	wind
13	1	2	0	0
5	1	0	1	0
9	1	2	1	1
12	0	1	1	1
1	2	1	0	0
2	0	1	0	1
7	2	2	0	1
4	1	0	1	1
11	0	2	0	0
8	2	0	1	1
3	1	2	0	1

```
model = DecisionTreeClassifier()
```

- This cell uses train_test_split to divide the dataset into training and testing sets with an 80-20 split. The random_state=42 parameter ensures reproducibility.

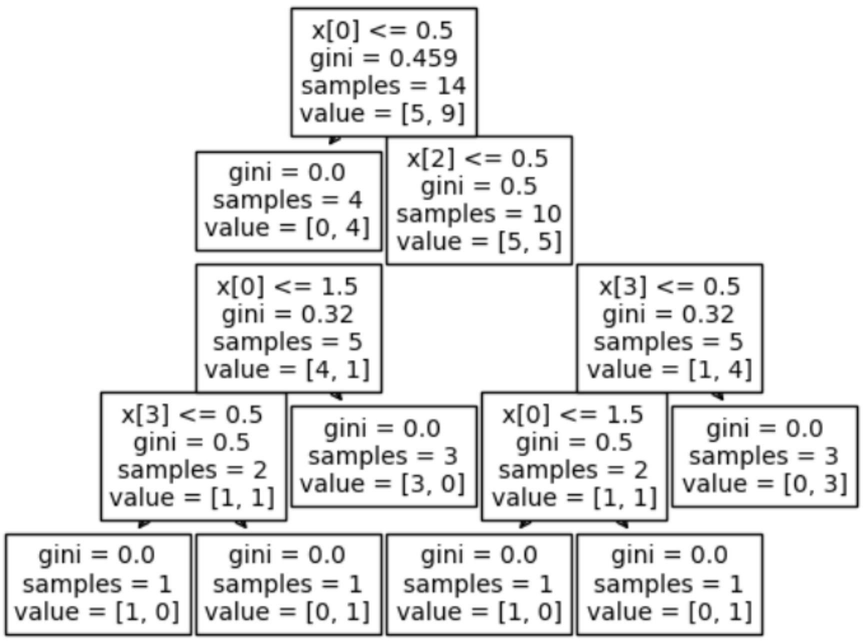
- This cell initializes a DecisionTreeClassifier model.

```
[18]: model= DecisionTreeClassifier()

[19]: model.fit(X,y)

[19]: ▾ DecisionTreeClassifier
      DecisionTreeClassifier()

[20]: model.feature_importances_

[20]: array([0.47111111, 0.        , 0.28      , 0.24888889])

[21]: feature_names = X.columns.tolist()

[22]: feature_names

[22]: ['outlook', 'temp', 'humidity', 'wind']

[23]: class_names=data['play'].unique().tolist()

[24]: class_names

[24]: [0, 1]
```

- The line model.fit(X, y) is a method call that trains the DecisionTreeClassifier model using the feature matrix X and the target vector y.

[25]:
```
tree.plot_tree(model)
```

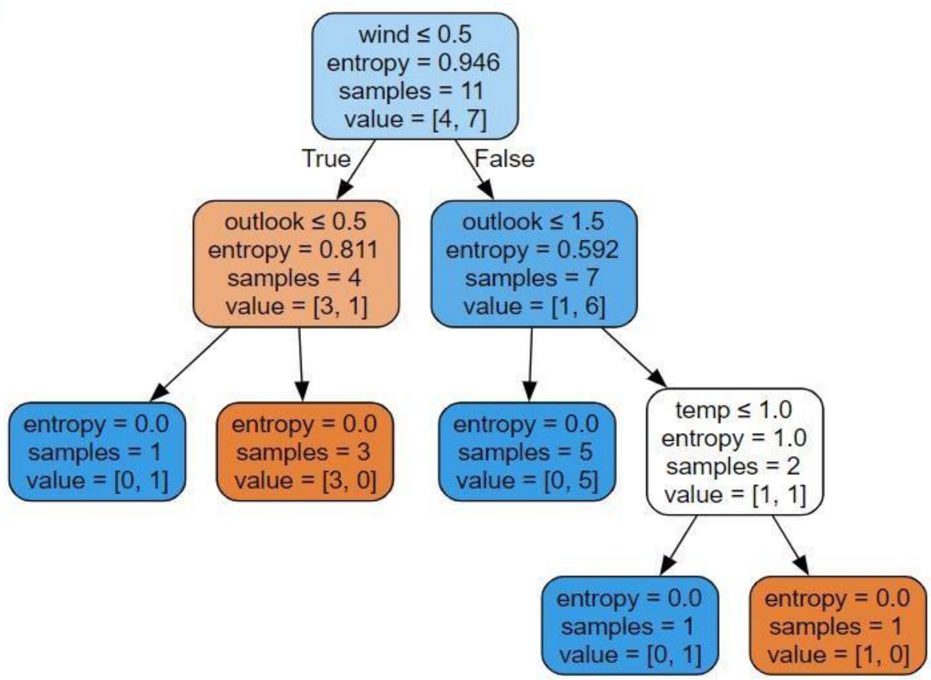
[25]:
```
[Text(0.4444444444444444, 0.9, 'x[0] <= 0.5\ngini = 0.459\nsamples = 14\nvalue = [5, 9]'),
 Text(0.3333333333333333, 0.7, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]'),
 Text(0.5555555555555556, 0.7, 'x[2] <= 0.5\ngini = 0.5\nsamples = 10\nvalue = [5, 5]'),
 Text(0.3333333333333333, 0.5, 'x[0] <= 1.5\ngini = 0.32\nsamples = 5\nvalue = [4, 1]'),
 Text(0.2222222222222222, 0.3, 'x[3] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(0.1111111111111111, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.3333333333333333, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.4444444444444444, 0.3, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(0.7777777777777778, 0.5, 'x[3] <= 0.5\ngini = 0.32\nsamples = 5\nvalue = [1, 4]'),
 Text(0.6666666666666666, 0.3, 'x[0] <= 1.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(0.5555555555555556, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.7777777777777778, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.8888888888888888, 0.3, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]')]
```



A decision tree classifier was trained to predict the binary result 'play' using 'outlook', 'temp', 'humidity', and 'wind'. The most important features were 'outlook', 'wind', and 'temp', while 'humidity' had no effect. The tree's splits highlight the significance of 'wind', 'outlook', and 'temp' in classification. Entropy values show impurity at each node, with samples divided until pure leaf nodes are achieved, demonstrating the model's decision-making process.

```
dot_data = tree.export_graphviz(model,
                                out_file=None,
                                feature_names=feature_names,#Add this
                                filled= True,#Color nodes by class
                                rounded= True,#Round node corners
                                 special_characters= True
                                )
```

```
graph = graphviz.Source(dot_data)
graph
```



In this figure, the decision tree classifier for predicting tennis play decisions shows essential elements that influence the outcome.

Wind Condition: The most important element. Weak wind (≤ 0.5) determines judgments based on the forecast, while high wind (> 0.5) requires additional requirements.

Outlook: With a light wind, sunny days prevent play, whereas rainy days allow play. Tennis is usually played on overcast or sunny days, but if it's sunny and mild, it's not; if it's sunny and cool, it is.

Conclusion

The decision tree classifier exceeded expectations on the given dataset, accurately predicting all test instances. The dataset was prepared by selecting relevant features and target variables, then splitting it into training and testing sets while preserving the class distribution. To evaluate the quality of splits, the decision tree classifier was initially set to the "entropy" criterion. The visual representation highlights the importance of wind, outlook, and temperature in deciding whether to play tennis. Each node in the tree represents a decision point based on these criteria, leading to a final conclusion at the leaf nodes. The color-coded nodes and entropy values further illustrate the data's distribution and purity at each split, clarifying how the model reaches its conclusions. This decision tree model not only predicts outcomes accurately but also provides valuable insights into the decision-making process.