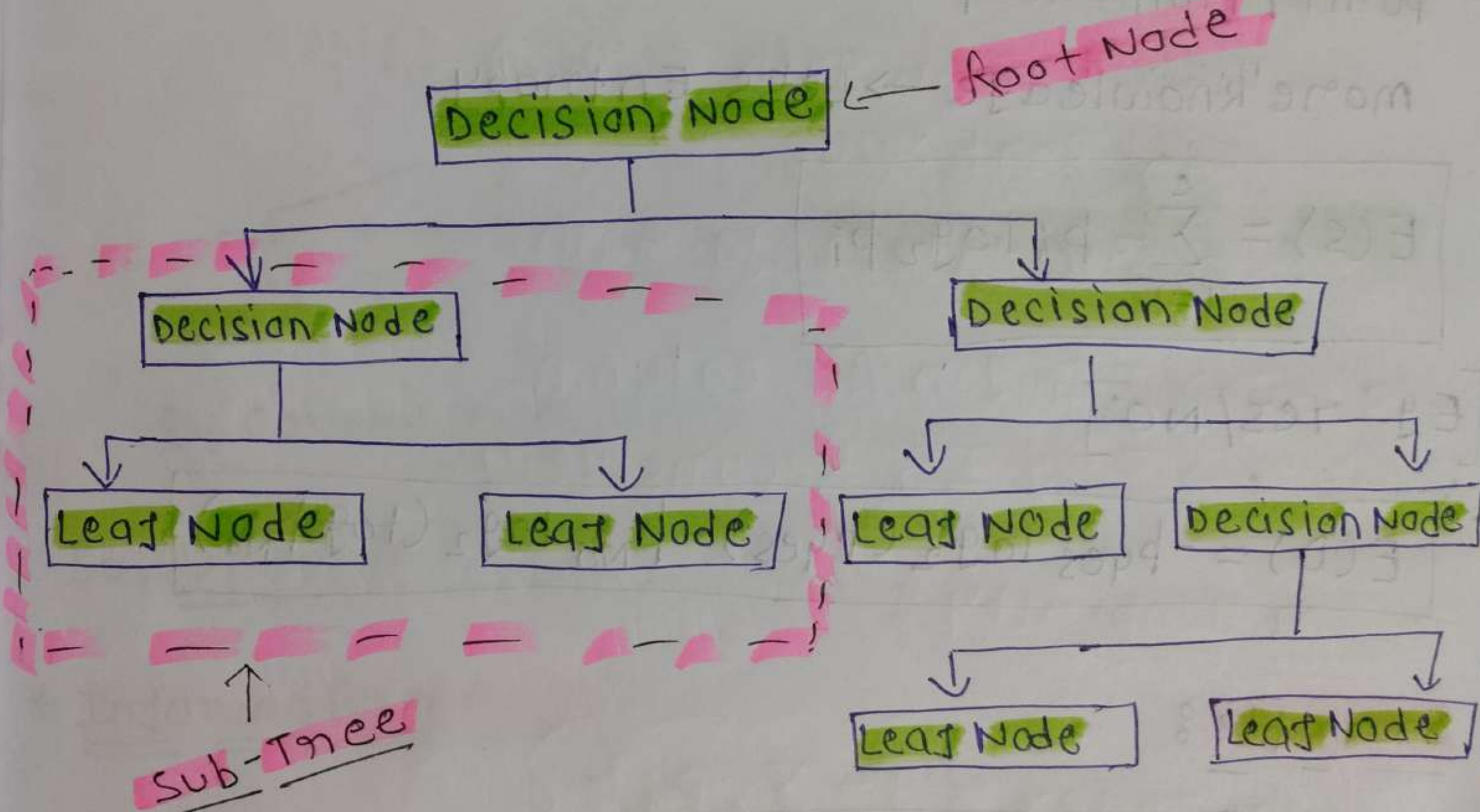
* Decision Trees Algorithms :

↳ Supervised Learning Technique.
↳ For Both Classification & Regression.
↳ Tree Structured / Flow chart Structure Classifier.

↳ Internal node → Features of data
Branches → Decision Rule
Leaf Node → outcomes



↳ It's easy to understand.
↳ Pruning : Process of Removing unwanted Branches from tree.
↳ we find best attribute in dataset by using ASM (Attribute selection measure)
↳ we use CART Algorithm.

* Advantages

i) Intutive
ii) minimal data Processing
iii) logarithimic.

Disadvantages

i) overfitting
ii) Prone to error for imbalanced data.

* Entropy :

measure of disorder. or measure of purity / impurity.

more knowledge → Less Entropy.

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

[Eg. Yes/NO]

$$\boxed{E(D) = -p_{yes} \log_2 (p_{yes}) - p_{NO} \log_2 (\cancel{tog}\, p_{NO})}$$

Example :

Salary	Age	Purchase
20000	21	Yes
10000	45	NO
6000	27	Yes
8000	31	NO
12000	18	NO

Yes → 2
NO → 3

$$E = -\frac{2}{5} \log_2 \left(\frac{2}{5}\right) - \frac{3}{5} \log_2 \left(\frac{3}{5}\right)$$

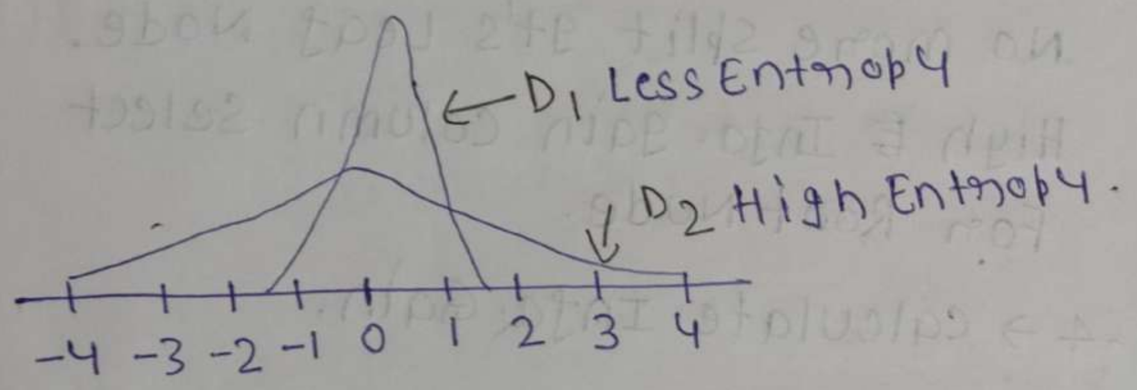
$$\boxed{E = 0.97}$$

↳ For 2 class problem min entropy is 0. & max is 1

↳ For more than 2 class min entropy is 0 & max can be greater than 1.

↳ Both $\log_e$ & $\log_2$ can be used.

* Entropy for continous variables :



D_1 cover moradata points in small area

So, D_1 have less Entropy

* Information Gain :

$$I.G = E(Parent) - \{weighted\ Avg.\} * E(Child)$$

$$\Sigma(P) = \sum_{i=1}^{c} -p_i \log_2 p_i \leftarrow \text{Parent Entropy}$$

Step-2 → ~~eta~~ calculate Entropy of all children after Root node split.

Step-3 → calculated weighted Entropy of Children.

[Entropy = 0 → Leaf Node]

NOTE: when Entropy is zero 0 then No more split it's Leaf Node. High ~~E~~ Info. gain column select For Root Node.

Step-4 → calculate Info. Gain.

Step-5 → IG for All column.

Step-6 → Find IG Recursively.

* Gini Impurity:

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

Eg. Yes | No

↓

$$G = 1 - (P_y^2 + P_n^2)$$

In [1]:
```python
import pyforest
```

In [2]:
```python
data= pd.read_csv(r'E:\IT Learning\My Projects\Python Projects\Datasets\PlayTennis.csv')
```

In [3]:
```python
data
```

Out[3]:

	Outlook	Temperature	Humidity	Wind	Play Tennis
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

In [4]:
```python
data.shape
```

Out[4]:
```
(14, 5)
```

In [5]:
```
y= data['Play Tennis']
```

In [6]:
```
y.value_counts()
```

Out[6]:
```
Play Tennis
Yes    9
No     5
Name: count, dtype: int64
```

In [7]:
```
py= 9/14
pn= 5/14
```

In [8]:
```
# Parent Entropy
E_P= -(9/14)*np.log2(9/14)-(5/14)*np.log2(5/14)
E_P
```

Out[8]:
```
0.9402859586706311
```

Features Entropy And Info Gain

For Outlook Feature

In [9]:
```
data.groupby('Outlook')['Play Tennis'].value_counts()
```

Out[9]:
```
Outlook   Play Tennis
Overcast  Yes            4
Rain      Yes            3
          No             2
Sunny     No             3
          Yes            2
Name: count, dtype: int64
```

In [10]:
```
e_overcast= 0
e_rain= -3/5*np.log2(3/5)-2/5*np.log2(2/5)
e_sunny= -2/5*np.log2(2/5)-3/5*np.log2(3/5)
```

In [11]:
```
weighted_avg_O= 4/14*e_overcast+5/15*e_rain+5/14*e_sunny
```

In [12]:
```
weighted_avg_O
```

Out[12]:
```
0.6704182675996521
```

In [13]:
```
IG_O= E_P-weighted_avg_O
```

In [14]:
```
IG_O
```

Out[14]:
```
0.2698676910709791
```

For Temperature Feature

In [15]:
```
data.groupby('Temperature')['Play Tennis'].value_counts()
```

Out[15]:
```
Temperature  Play Tennis
Cool         Yes            3
             No             1
Hot          No             2
             Yes            2
Mild         Yes            4
             No             2
Name: count, dtype: int64
```

In [16]:
```
e_cool= -3/4*np.log2(3/4)-1/4*np.log2(1/4)
e_hot= -2/4*np.log2(2/4)-2/4*np.log2(2/4)
e_mild= -4/6*np.log2(4/6)-2/6*np.log2(2/6)
```

In [17]:
```
weighted_avg_T= 4/14*e_cool+4/14*e_hot+6/14*e_mild
```

In [18]:
```
IG_T= E_P-weighted_avg_T
```

In [19]:
```
IG_T
```

Out[19]:
```
0.02922256565895487
```

For Humidity

In [20]:
```
data.groupby('Humidity')['Play Tennis'].value_counts()
```

Out[20]:
```
Humidity  Play Tennis
High      No             4
          Yes            3
Normal    Yes            6
          No             1
Name: count, dtype: int64
```

In [21]:
```
e_high= -3/7*np.log2(3/7)-4/7*np.log2(4/7)
e_normal= -6/7*np.log2(6/7)-1/7*np.log2(1/7)
```

In [22]:
```
weighted_avg_H= 7/14*e_high+7/14*e_normal
```

In [23]:
```
IG_H= E_P-weighted_avg_H
IG_H
```

Out[23]:
```
0.15183550136234159
```

For Wind

In [24]:
```
data.groupby('Wind')['Play Tennis'].value_counts()
```

Out[24]:
```
Wind    Play Tennis
Strong  No             3
        Yes            3
Weak    Yes            6
        No             2
Name: count, dtype: int64
```

In [25]:
```
e_strong= -3/6*np.log2(3/6)-3/6*np.log2(3/6)
e_weak= -6/8*np.log2(6/8)-2/8*np.log2(2/8)
```

In [26]:
```
weighted_avg_W= 6/14*e_strong+8/14*e_weak
```

In [27]:
```
IG_W= E_P-weighted_avg_W
IG_W
```

Out[27]:
```
0.04812703040826949
```

Information Gain Values For All Features

In [28]:
```
Information_Gain= [IG_O,IG_H,IG_T,IG_W]
```

In [29]:
```
Information_Gain.sort(reverse= True)
```

In [30]:
```
Information_Gain
```

Out[30]:
```
[0.2698676910709791,
 0.15183550136234159,
 0.04812703040826949,
 0.02922256565895487]
```

In [31]:
```
feature_names = ['Outlook', 'Humidity', 'Temperature', 'Windy']
```

In [32]:
```
for feature, ig in zip(feature_names, Information_Gain):
    print(f'{feature}: {ig}')
```

```
Outlook: 0.2698676910709791
Humidity: 0.15183550136234159
Temperature: 0.04812703040826949
Windy: 0.02922256565895487
```

Final Summary:

when constructing a decision tree for this dataset, you should start by splitting on "Outlook" to capture the most significant source of information gain. "Humidity" and "Temperature" can be used for further refinement if needed, while "Windy" is likely to have a minimal impact on the decision-making process. This conclusion aligns with the principles of decision tree construction, where features with higher information gain are given priority in the splitting process.

In [ ]:
```

```