Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Data Mining

Lab - 10

Karan Songara

Roll no - 130 ---- Enrollment No - 24010101694

Implement Decision Tree(ID3) in python

Uses Information Gain to choose the best feature to split.

Recursively builds the tree until stopping conditions are met.

1. Calculate Entropy for the dataset.
2. Calculate Information Gain for each feature.
3. Choose the feature with maximum Information Gain.
4. Split dataset into subsets for that feature.
5. Repeat recursively until:

All samples in a node have the same label.
No features are left.
No data is left.

Step 2. Import the dataset from this address.

import Pandas, Numpy

In [43]:
```
import pandas as pd
import numpy as np
```

Create Following Data

In [45]:
```
data = pd.DataFrame({
    'Outlook': ['Sunny', 'Sunny', 'Overcast', 'Rain', 'Rain', 'Rain', 'Overcast'
    'Temperature': ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mild',
    'Humidity': ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal', '
    'Wind': ['Weak', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak
    'PlayTennis': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Y
})
```

In [47]:
```
data
```

Out[47]:

	Outlook	Temperature	Humidity	Wind	PlayTennis
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

Now Define Function to Calculate Entropy

In [29]:
```
def entropy(y):
    values, counts = np.unique(y, return_counts=True)
    print(values)
    print(counts)
    probabilities = counts / counts.sum()
    return -np.sum(probabilities * np.log2(probabilities))
```

Testing of Above Function -

y = np.array(['Yes', 'No', 'Yes', 'Yes'])

Function Call - > entropy(y))

output - 0.8112781244591328

In [31]:
```
y = np.array(['Yes', 'No', 'Yes', 'Yes'])
print(entropy(y))
```

```
['No' 'Yes']
[1 3]
0.8112781244591328
```

Define function to Calculate Information Gain

In [33]:
```
def information_gain(data, split_attribute, target):
    total_entropy = entropy(data[target])
    print(total_entropy)
    values, counts = np.unique(data[split_attribute], return_counts=True)
    print(values)
    print(counts)
    weighted_entropy = 0
    for v, c in zip(values, counts):
        subset = data[data[split_attribute] == v]
        weighted_entropy += (c / len(data)) * entropy(subset[target])
    return total_entropy - weighted_entropy
```

Testing of Above Function-

data = pd.DataFrame({ 'Weather': ['Sunny', 'Sunny', 'Rain', 'Rain'], 'Play': ['Yes', 'No', 'Yes', 'Yes'] })

Function Call - > information_gain(data, 'Weather', 'Play')

Output - 0.31127812445913283

In [35]:
```
data = pd.DataFrame({
    'Weather': ['Sunny', 'Sunny', 'Rain', 'Rain'],
    'Play':    ['Yes', 'No',   'Yes', 'Yes']
})

print(information_gain(data, 'Weather', 'Play'))
```

```
['No' 'Yes']
[1 3]
0.8112781244591328
['Rain' 'Sunny']
[2 2]
['Yes']
[2]
['No' 'Yes']
[1 1]
0.31127812445913283
```

Implement ID3 Algo

In [37]:
```
def id3(data, features, target):
    # If all labels are same → return the label
    if len(np.unique(data[target])) == 1:
```

```
        return np.unique(data[target])[0]

    # If no features left → return majority label
    if len(features) == 0:
        return data[target].mode()[0]

    # Choose best feature
    gains = [information_gain(data, feature, target) for feature in features]
    best_feature = features[np.argmax(gains)]

    tree = {best_feature: {}}

    # For each value of best feature → branch
    for value in np.unique(data[best_feature]):
        sub_data = data[data[best_feature] == value].drop(columns=[best_feature]
        subtree = id3(sub_data, [f for f in features if f != best_feature], targ
        tree[best_feature][value] = subtree
    return tree
```

Use ID3

In [58]:
```
features = list(data.columns[:-1])
target = 'PlayTennis'

tree = id3(data, features, target)
```

```
['No' 'Yes']
[5 9]
0.9402859586706311
['Overcast' 'Rain' 'Sunny']
[4 5 5]
['Yes']
[4]
['No' 'Yes']
[2 3]
['No' 'Yes']
[3 2]
['No' 'Yes']
[5 9]
0.9402859586706311
['Cool' 'Hot' 'Mild']
[4 4 6]
['No' 'Yes']
[1 3]
['No' 'Yes']
[2 2]
['No' 'Yes']
[2 4]
['No' 'Yes']
[5 9]
0.9402859586706311
['High' 'Normal']
[7 7]
['No' 'Yes']
[4 3]
['No' 'Yes']
[1 6]
['No' 'Yes']
[5 9]
0.9402859586706311
['Strong' 'Weak']
[6 8]
['No' 'Yes']
[3 3]
['No' 'Yes']
[2 6]
['No' 'Yes']
[2 3]
0.9709505944546686
['Cool' 'Mild']
[2 3]
['No' 'Yes']
[1 1]
['No' 'Yes']
[1 2]
['No' 'Yes']
[2 3]
0.9709505944546686
['High' 'Normal']
[2 3]
['No' 'Yes']
[1 1]
['No' 'Yes']
[1 2]
['No' 'Yes']
[2 3]
```

```
0.9709505944546686
['Strong' 'Weak']
[2 3]
['No']
[2]
['Yes']
[3]
['No' 'Yes']
[3 2]
0.9709505944546686
['Cool' 'Hot' 'Mild']
[1 2 2]
['Yes']
[1]
['No']
[2]
['No' 'Yes']
[1 1]
['No' 'Yes']
[3 2]
0.9709505944546686
['High' 'Normal']
[3 2]
['No']
[3]
['Yes']
[2]
['No' 'Yes']
[3 2]
0.9709505944546686
['Strong' 'Weak']
[2 3]
['No' 'Yes']
[1 1]
['No' 'Yes']
[2 1]
```

Print Tree

In [62]:
```
tree
```

Out[62]:
```
{'Outlook': {'Overcast': 'Yes',
  'Rain': {'Wind': {'Strong': 'No', 'Weak': 'Yes'}},
  'Sunny': {'Humidity': {'High': 'No', 'Normal': 'Yes'}}}}
```