20 August 2025

Data Mining : Lab - 10

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.

In [21]:

```
df = pd.read_csv('https://raw.githubusercontent.com/justmarkham/DAT8/master/
df
```

Out[21]:

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98
...	...	...	...	...	...
4617	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Sour ...	$11.75
4618	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...	$11.75
4619	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	$11.25
4620	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Lettu...	$8.75
4621	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	$8.75

4622 rows × 5 columns

import Pandas, Numpy

In [22]:

```
import pandas as pd
import numpy as np
```

Create Following Data

In [23]:

```
data = pd.DataFrame({
    'Outlook': ['Sunny', 'Sunny', 'Overcast', 'Rain', 'Rain', 'Rain', 'Overc
    'Temperature': ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mi
    'Humidity': ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal
    'Wind': ['Weak', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', '
```

```
    'PlayTennis': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes'
})
```

In [24]:
```
data
```

Out[24]:

	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 [25]:
```
def entropy(y):
    values, counts = np.unique(y, return_counts=True)
    # print("Values:", values)
    # print("Counts:", counts)
    probabilities = counts / counts.sum()
    # print("Probabilities:", probabilities)
    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 [26]:
```
y = np.array(['Yes', 'No', 'Yes', 'Yes'])
entropy(y)
```

Out[26]: 0.8112781244591328

In [27]:
```
y = np.array(['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Ye
entropy(y)
```

Out[27]: 0.9402859586706311

Define function to Calculate Information Gain

In [28]:
```
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 i in range(len(values)):
        subset = data[data[split_attribute] == values[i]]
        # print(subset)
        weighted_entropy += (counts[i] / counts.sum()) * entropy(subset[targ
        # print(weighted_entropy)

    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 - 0.311278124459132843

In [29]:
```
data = pd.DataFrame({ 'Weather': ['Sunny', 'Sunny', 'Rain', 'Rain'], 'Play':
information_gain(data, 'Weather', 'Play')
```

Out[29]: 0.31127812445913283

Implement ID3 Algo

In [33]:
```
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_fe
        subtree = id3(sub_data, [f for f in features if f != best_feature],
        tree[best_feature][value] = subtree

    return tree
```

Use ID3

In [34]:
```
print(data.columns)
```

```
Index(['Weather', 'Play'], dtype='object')
```

In [38]:
```
features = list(data.columns[:-1])
target = 'Play'
tree= id3(data, features, target)
```

Print Tree

In [39]:
```
tree
```

Out[39]:
```
{'Weather': {'Rain': 'Yes', 'Sunny': 'No'}}
```

Extra: Create Predict Function

In [40]:
```
def predict(tree, sample):
    for attr, branches in tree.items():  # get root attribute
        value = sample.get(attr)

        if value not in branches:
            possible_results = list(branches.values())
            while isinstance(possible_results[0], dict):
                possible_results = list(possible_results[0].values())
            return possible_results[0]

        result = branches[value]

        if isinstance(result, dict):  # subtree → recurse
            return predict(result, sample)
        else:  # leaf node
            return result
```

Extra: Predict for a sample

sample = {'Outlook': 'Sunny', 'Temperature': 'Cool', 'Humidity': 'High', 'Wind': 'Strong'}

Your Answer ?

In [41]:

```
sample = {'Outlook': 'Sunny', 'Temperature': 'Cool', 'Humidity': 'High', 'Wi
print(predict(tree, sample))
```

```
Yes
```