Data Mining

Mohil Parmar

23010101192

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.

import Pandas, Numpy

In [6]:
```
import pandas as pd
import numpy as np
```

In [8]:
```
url = "https://raw.githubusercontent.com/justmarkham/DAT8/master/data/chipotle.tsv"

csv_data = pd.read_csv(url,sep='\t')
print(csv_data)
```

```
      order_id  quantity                              item_name  \
0            1         1           Chips and Fresh Tomato Salsa
1            1         1                                   Izze
2            1         1                       Nantucket Nectar
3            1         1  Chips and Tomatillo-Green Chili Salsa
4            2         2                           Chicken Bowl
...        ...       ...                                    ...
4617      1833         1                          Steak Burrito
4618      1833         1                          Steak Burrito
4619      1834         1                     Chicken Salad Bowl
4620      1834         1                     Chicken Salad Bowl
4621      1834         1                     Chicken Salad Bowl

                                     choice_description item_price
0                                                   NaN     $2.39
1                                          [Clementine]     $3.39
2                                               [Apple]     $3.39
3                                                   NaN     $2.39
4     [Tomatillo-Red Chili Salsa (Hot), [Black Beans...    $16.98
...                                                 ...       ...
4617  [Fresh Tomato Salsa, [Rice, Black Beans, Sour ...    $11.75
4618  [Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...    $11.75
4619  [Fresh Tomato Salsa, [Fajita Vegetables, Pinto...    $11.25
4620  [Fresh Tomato Salsa, [Fajita Vegetables, Lettu...     $8.75
4621  [Fresh Tomato Salsa, [Fajita Vegetables, Pinto...     $8.75

[4622 rows x 5 columns]
```

Create Following Data

In [10]:
```python
data = pd.DataFrame({
    'Outlook': ['Sunny', 'Sunny', 'Overcast', 'Rain', 'Rain', 'Rain', 'Overcast', 'Sunny', 'Sunny', 'Rain', 'Sunny', 'Overcast
    'Temperature': ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mild', 'Cool', 'Mild', 'Mild', 'Mild', 'Hot', 'Mild'
    'Humidity': ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'Normal', 'Normal', 'High', '
    'Wind': ['Weak', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak',
    'PlayTennis': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
})

bookdata = pd.DataFrame({
```

```
    'a1':['t','t','f','f','f','t','t','t','f','f'],
    'a2':['h','h','h','c','c','c','h','h','c','c'],
    'a3':['hi','hi','hi','nor','nor','hi','hi','nor','nor','hi'],
    'class':['n','n','y','y','y','n','n','y','y','y']
})
```

In [ ]:

Now Define Function to Calculate Entropy

In [13]:

```
def entropy(y):

    unique_labels, counts = np.unique(y, return_counts=True)
    probabilities = counts / counts.sum()

    entropy_value = -np.sum(probabilities * np.log2(probabilities))
    return entropy_value
```

Testing of Above Function -

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

Function Call - > entropy(y))

output - 0.8112781244591328

In [15]:

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

```
0.9709505944546686
```

Define function to Calculate Information Gain

In [21]:

```
def information_gain(data, split_attribute, target):
    # Calculate total entropy of the target
    total_entropy = entropy(data[target])
```

```
    # Find unique values of the splitting attribute
    values, counts = np.unique(data[split_attribute], return_counts=True)

    # Calculate weighted entropy after splitting
    weighted_entropy = 0
    for value, count in zip(values, counts):
        subset = data[data[split_attribute] == value]
        weighted_entropy += (count / len(data)) * entropy(subset[target])

    # Information Gain formula
    info_gain = total_entropy - weighted_entropy
    return info_gain
```

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 [24]:
```
data = pd.DataFrame({
    'Weather': ['Sunny', 'Sunny', 'Rain', 'Rain'],
    'Play':    ['Yes', 'No',   'Yes', 'Yes']
})

print("Information Gain:", information_gain(data, 'Weather', 'Play'))
```

```
Information Gain: 0.31127812445913283
```

Implement ID3 Algo

In [27]:
```
def id3(data, target, features):
    # If all examples are same class → leaf node
    if len(np.unique(data[target])) == 1:
```

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

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

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

    # Create tree dict
    tree = {best_feature: {}}
    for value in np.unique(data[best_feature]):
        subset = data[data[best_feature] == value].drop(columns = [best_feature])
        subtree = id3(subset, target, [f for f in features if f != best_feature])
        tree[best_feature][value] = subtree

    return tree
```

Use ID3

In [30]:
```python
# Test dataset
# data = pd.DataFrame({
#     'Weather': ['Sunny', 'Sunny', 'Rain', 'Rain', 'Overcast'],
#     'Temperature': ['Hot', 'Mild', 'Hot', 'Mild', 'Hot'],
#     'Play': ['No', 'Yes', 'Yes', 'Yes', 'Yes']
# })

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

features = ['a1', 'a2','a3']
tree = id3(bookdata, 'class', features)
print(tree)
```

```
{'a1': {'f': 'y', 't': {'a3': {'hi': 'n', 'nor': 'y'}}}}
```

Print Tree

In [33]:
```python
print(tree)
```

```
{'a1': {'f': 'y', 't': {'a3': {'hi': 'n', 'nor': 'y'}}}}
```

Extra: Create Predict Function

In [36]:
```python
def predict(tree, sample):
```

```
  Cell In[36], line 2

    ^
SyntaxError: incomplete input
```

Extra: Predict for a sample

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

Your Answer ?