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.

```
data =
pd.read_csv('https://raw.githubusercontent.com/justmarkham/DAT8/master
/data/chipotle.tsv',sep='\t')
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]
```

import Pandas, Numpy

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

Create Following Data

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

Now Define Function to Calculate Entropy

```
def entropy(y):
    values , counts = np.unique(y , return_counts = True)
    print(values)
    print(counts)
    probabilities = counts / counts.sum()
    print(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

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

```
['No' 'Yes']
[1 3]
[0.25 0.75]
0.8112781244591328
```

Define function to Calculate Information Gain

```
def information_gain(data, split_attribute, target):
    total_entropy = entropy(data[target])
    print("total_entropy",total_entropy)
    values , counts = np.unique(data[split_attribute],return_counts =
True)
    print("counts",values)
    print("counts",counts)

    weighted_entropy = 0
    for i in range (len(values)):
        subset = data[data[split_attribute] == values[i]]
        print("subset",subset)
        weighted_entropy += (counts[i] / counts.sum()) *
entropy(subset[target])
        print("weighted_entropy",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 - 0.31127812445913283

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

['No' 'Yes']
[1 3]
[0.25 0.75]
total_entropy 0.8112781244591328
counts ['Rain' 'Sunny']
counts [2 2]
subset   Weather Play
2    Rain  Yes
3    Rain  Yes
['Yes']
[2]
[1.]
weighted_entropy 0.0
subset   Weather Play
0   Sunny  Yes
1   Sunny   No
['No' 'Yes']
[1 1]
```

```
[0.5 0.5]
weighted_entropy 0.5

0.31127812445913283
```

Implement ID3 Algo

```
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], target)
        tree[best_feature][value] = subtree

    return tree
```

Use ID3

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

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

```
['No' 'Yes']
[5 9]
[0.35714286 0.64285714]
total_entropy 0.9402859586706311
counts ['Overcast' 'Rain' 'Sunny']
counts [4 5 5]
subset      Outlook Temperature Humidity    Wind PlayTennis
2   Overcast         Hot     High    Weak        Yes
6   Overcast        Cool   Normal  Strong        Yes
```

```
11  Overcast        Mild     High  Strong        Yes
12  Overcast         Hot   Normal    Weak        Yes
['Yes']
[4]
[1.]
weighted_entropy 0.0
subset    Outlook Temperature Humidity    Wind PlayTennis
3     Rain        Mild     High    Weak        Yes
4     Rain        Cool   Normal    Weak        Yes
5     Rain        Cool   Normal  Strong         No
9     Rain        Mild   Normal    Weak        Yes
13    Rain        Mild     High  Strong         No
['No' 'Yes']
[2 3]
[0.4 0.6]
weighted_entropy 0.3467680694480959
subset    Outlook Temperature Humidity    Wind PlayTennis
0    Sunny         Hot     High    Weak         No
1    Sunny         Hot     High  Strong         No
7    Sunny        Mild     High    Weak         No
8    Sunny        Cool   Normal    Weak        Yes
10   Sunny        Mild   Normal  Strong        Yes
['No' 'Yes']
[3 2]
[0.6 0.4]
weighted_entropy 0.6935361388961918
['No' 'Yes']
[5 9]
[0.35714286 0.64285714]
total_entropy 0.9402859586706311
counts ['Cool' 'Hot' 'Mild']
counts [4 4 6]
subset     Outlook Temperature Humidity    Wind PlayTennis
4      Rain        Cool   Normal    Weak        Yes
5      Rain        Cool   Normal  Strong         No
6  Overcast        Cool   Normal  Strong        Yes
8     Sunny        Cool   Normal    Weak        Yes
['No' 'Yes']
[1 3]
[0.25 0.75]
weighted_entropy 0.23179374984546652
subset      Outlook Temperature Humidity    Wind PlayTennis
0      Sunny         Hot     High    Weak         No
1      Sunny         Hot     High  Strong         No
2   Overcast         Hot     High    Weak        Yes
12  Overcast         Hot   Normal    Weak        Yes
['No' 'Yes']
[2 2]
[0.5 0.5]
```

```
weighted_entropy 0.5175080355597522
subset      Outlook Temperature Humidity    Wind PlayTennis
3       Rain        Mild     High    Weak        Yes
7      Sunny        Mild     High    Weak         No
9       Rain        Mild   Normal    Weak        Yes
10     Sunny        Mild   Normal  Strong        Yes
11  Overcast        Mild     High  Strong        Yes
13      Rain        Mild     High  Strong         No
['No' 'Yes']
[2 4]
[0.33333333 0.66666667]
weighted_entropy 0.9110633930116763
['No' 'Yes']
[5 9]
[0.35714286 0.64285714]
total_entropy 0.9402859586706311
counts ['High' 'Normal']
counts [7 7]
subset      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
7      Sunny        Mild     High    Weak         No
11  Overcast        Mild     High  Strong        Yes
13      Rain        Mild     High  Strong         No
['No' 'Yes']
[4 3]
[0.57142857 0.42857143]
weighted_entropy 0.49261406801712576
subset      Outlook Temperature Humidity    Wind PlayTennis
4       Rain        Cool   Normal    Weak        Yes
5       Rain        Cool   Normal  Strong         No
6   Overcast        Cool   Normal  Strong        Yes
8      Sunny        Cool   Normal    Weak        Yes
9       Rain        Mild   Normal    Weak        Yes
10     Sunny        Mild   Normal  Strong        Yes
12  Overcast         Hot   Normal    Weak        Yes
['No' 'Yes']
[1 6]
[0.14285714 0.85714286]
weighted_entropy 0.7884504573082896
['No' 'Yes']
[5 9]
[0.35714286 0.64285714]
total_entropy 0.9402859586706311
counts ['Strong' 'Weak']
counts [6 8]
subset      Outlook Temperature Humidity    Wind PlayTennis
```

```
1      Sunny         Hot     High  Strong         No
5       Rain        Cool   Normal  Strong         No
6   Overcast        Cool   Normal  Strong        Yes
10     Sunny        Mild   Normal  Strong        Yes
11  Overcast        Mild     High  Strong        Yes
13      Rain        Mild     High  Strong         No
['No' 'Yes']
[3 3]
[0.5 0.5]
weighted_entropy 0.42857142857142855
subset      Outlook Temperature Humidity  Wind PlayTennis
0      Sunny         Hot     High  Weak         No
2   Overcast         Hot     High  Weak        Yes
3       Rain        Mild     High  Weak        Yes
4       Rain        Cool   Normal  Weak        Yes
7      Sunny        Mild     High  Weak         No
8      Sunny        Cool   Normal  Weak        Yes
9       Rain        Mild   Normal  Weak        Yes
12  Overcast         Hot   Normal  Weak        Yes
['No' 'Yes']
[2 6]
[0.25 0.75]
weighted_entropy 0.8921589282623617
['No' 'Yes']
[2 3]
[0.4 0.6]
total_entropy 0.9709505944546686
counts ['Cool' 'Mild']
counts [2 3]
subset   Temperature Humidity    Wind PlayTennis
4        Cool   Normal    Weak        Yes
5        Cool   Normal  Strong         No
['No' 'Yes']
[1 1]
[0.5 0.5]
weighted_entropy 0.4
subset    Temperature Humidity    Wind PlayTennis
3         Mild     High    Weak        Yes
9         Mild   Normal    Weak        Yes
13        Mild     High  Strong         No
['No' 'Yes']
[1 2]
[0.33333333 0.66666667]
weighted_entropy 0.9509775004326937
['No' 'Yes']
[2 3]
[0.4 0.6]
total_entropy 0.9709505944546686
counts ['High' 'Normal']
```

```
counts [2 3]
subset    Temperature Humidity    Wind PlayTennis
3          Mild     High    Weak         Yes
13         Mild     High  Strong          No
['No' 'Yes']
[1 1]
[0.5 0.5]
weighted_entropy 0.4
subset   Temperature Humidity    Wind PlayTennis
4         Cool   Normal    Weak         Yes
5         Cool   Normal  Strong          No
9         Mild   Normal    Weak         Yes
['No' 'Yes']
[1 2]
[0.33333333 0.66666667]
weighted_entropy 0.9509775004326937
['No' 'Yes']
[2 3]
[0.4 0.6]
total_entropy 0.9709505944546686
counts ['Strong' 'Weak']
counts [2 3]
subset    Temperature Humidity    Wind PlayTennis
5          Cool   Normal  Strong          No
13         Mild     High  Strong          No
['No']
[2]
[1.]
weighted_entropy 0.0
subset   Temperature Humidity  Wind PlayTennis
3         Mild     High  Weak         Yes
4         Cool   Normal  Weak         Yes
9         Mild   Normal  Weak         Yes
['Yes']
[3]
[1.]
weighted_entropy 0.0
['No' 'Yes']
[3 2]
[0.6 0.4]
total_entropy 0.9709505944546686
counts ['Cool' 'Hot' 'Mild']
counts [1 2 2]
subset   Temperature Humidity  Wind PlayTennis
8         Cool   Normal  Weak         Yes
['Yes']
[1]
[1.]
weighted_entropy 0.0
```

```
subset   Temperature Humidity    Wind PlayTennis
0         Hot     High    Weak         No
1         Hot     High  Strong         No
['No']
[2]
[1.]
weighted_entropy 0.0
subset    Temperature Humidity    Wind PlayTennis
7         Mild     High    Weak         No
10        Mild   Normal  Strong        Yes
['No' 'Yes']
[1 1]
[0.5 0.5]
weighted_entropy 0.4
['No' 'Yes']
[3 2]
[0.6 0.4]
total_entropy 0.9709505944546686
counts ['High' 'Normal']
counts [3 2]
subset   Temperature Humidity    Wind PlayTennis
0         Hot     High    Weak         No
1         Hot     High  Strong         No
7        Mild     High    Weak         No
['No']
[3]
[1.]
weighted_entropy 0.0
subset    Temperature Humidity    Wind PlayTennis
8         Cool   Normal    Weak        Yes
10        Mild   Normal  Strong        Yes
['Yes']
[2]
[1.]
weighted_entropy 0.0
['No' 'Yes']
[3 2]
[0.6 0.4]
total_entropy 0.9709505944546686
counts ['Strong' 'Weak']
counts [2 3]
subset    Temperature Humidity    Wind PlayTennis
1          Hot     High  Strong         No
10        Mild   Normal  Strong        Yes
['No' 'Yes']
[1 1]
[0.5 0.5]
weighted_entropy 0.4
subset   Temperature Humidity  Wind PlayTennis
```

```
0         Hot     High  Weak         No
7        Mild     High  Weak         No
8        Cool   Normal  Weak        Yes
['No' 'Yes']
[2 1]
[0.66666667 0.33333333]
weighted_entropy 0.9509775004326937
```

Print Tree

```
print(tree)
```

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

Extra: Create Predict Function

```
def predict(tree, sample):

```

Extra: Predict for a sample

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

Your Answer ?

```


Prediction: No

```