In [ ]:

```python
import pandas as pd
import numpy as np
df_tennis = pd.read_csv('play_tennis.csv')
```

In [ ]:

```python
from collections import Counter
def entropy_list(a_list):
    cnt = Counter(x for x in a_list)
    num_instance = len(a_list)*1.0
    probs = [x/num_instance for x in cnt.values()]
    return entropy(probs)
```

In [ ]:

```python
import math
def entropy(probs):
    return sum([-prob*math.log(prob,2) for prob in probs])
```

In [ ]:

```python
def info_gain(df,split,target,trace=0):
    df_split = df.groupby(split)
    nobs = len(df.index)*1.0
    df_agg_ent = df_split.agg({ target:[entropy_list, lambda x: len(x)/nobs] })
    df_agg_ent.columns = ['Entropy','PropObserved']
    new_entropy = sum( df_agg_ent['Entropy'] * df_agg_ent["PropObserved"])
    old_entropy = entropy_list(df[target])
    return old_entropy - new_entropy
```

In [ ]:

```python
def id3(df,target,attribute_name,default_class = None):
    cnt = Counter(x for x in df[target])
    if len(cnt)==1:
        return next(iter(cnt))
    elif df.empty or (not attribute_name):
        return default_class
    else:
        default_class = max(cnt.keys())
        gains = [info_gain(df,attr,target) for attr in attribute_name]
        index_max = gains.index(max(gains))
        best_attr = attribute_name[index_max]
        tree = { best_attr:{ } }
        remaining_attr = [x for x in attribute_name if x!=best_attr]
        for attr_val, data_subset in df.groupby(best_attr):
            subtree = id3(data_subset,target,remaining_attr,default_class)
            tree[best_attr][attr_val] = subtree
        return tree
```

In [ ]:

```python
def classify(instance,tree,default = None):
    attribute = next(iter(tree))
    if instance[attribute] in tree[attribute].keys():
        result = tree[attribute][instance[attribute]]
        if isinstance(result,dict):
            return classify(instance,result)
        else:
            return result
    else:
        return default
```

In [ ]:

```python
attribute_names = list(df_tennis.columns)
print(attribute_names)
attribute_names.remove('PlayTennis') #Remove the class attribute
tree = id3(df_tennis,'PlayTennis',attribute_names)
print("\n\nThe Resultant Decision Tree is :\n")
print(tree)
```

```
['Outlook', 'Temperature', 'Humidity', 'Windy', 'PlayTennis']


The Resultant Decision Tree is :

{'Outlook': {'Overcast': 'Yes', 'Rainy': {'Windy': {'Strong': 'No', 'Wea
k': 'Yes'}}, 'Sunny': {'Humidity': {'High': 'No', 'Normal': 'Yes'}}}}
```

In [ ]:

```
training_data = df_tennis.iloc[1:-4] # all but last thousand instances
test_data = df_tennis.iloc[-4:] # just the last thousand
train_tree = id3(training_data, 'PlayTennis', attribute_names)
print("\n\nThe Resultant Decision train_tree is :\n")
print(train_tree)
test_data['predicted2'] = test_data.apply(classify,axis=1,args=(train_tree,'Yes') )
print ('\n\n Training the model for a few samples, and again predicting \'PlayTennis\'
 for remaining attribute')
print('The Accuracy for new trained data is : ' + str( sum(test_data['PlayTennis']==tes
t_data['predicted2'] ) / (1.0*len(test_data.index)) ))
```

```


The Resultant Decision train_tree is :

{'Outlook': {'Overcast': 'Yes', 'Rainy': {'Windy': {'Strong': 'No', 'Wea
k': 'Yes'}}, 'Sunny': {'Temperature': {'Cool': 'Yes', 'Hot': 'No', 'Mild':
'No'}}}}


 Training the model for a few samples, and again predicting 'PlayTennis' f
or remaining attribute
The Accuracy for new trained data is : 0.75

C:\Users\asus\AppData\Local\Temp\ipykernel_3004\3713865083.py:6: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_data['predicted2'] = test_data.apply(classify,axis=1,args=(train_tr
ee,'Yes') )
```

In [ ]: