4) Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample

Understanding the ID3 Decision Tree Algorithm

Think of the ID3 algorithm as a game of 20 Questions. It's like trying to guess something by asking yes/no questions. The goal is to build a tree of questions that helps us make the best guess.

The Data

Imagine we have a list of objects (like fruits) and for each object, we have some information (attributes) about them. For example, for fruits, the attributes could be color, size, and taste. Each fruit is either "good" or "bad."

Building the Tree

We start with a big question at the top of the tree, like "Is it red?" This is our first question.

- If the answer is "Yes," we ask another question, like "Is it small?" If "No," we might guess "Apple."
- If the answer is "No" to the first question, we might guess "Banana."

We keep asking questions and making guesses until we're sure about the answer (like guessing the fruit correctly).

Testing with a New Object

Now, imagine you have a new fruit that you've never seen before. You don't know what it is, but you have some information about it (its color, size, and taste).

- You start at the top of your tree with the first question, like "Is it red?" If it's "Yes," you move to the next question, and so on.
- You follow the tree until you reach a guess, like "Apple."

Understanding the Program

The program reads a spreadsheet with the information about fruits and their "good" or "bad" classification.

- It uses the ID3 algorithm to build the tree of questions and guesses.

- Then, it defines a new, unknown fruit and uses the tree to guess what it is (whether it's "good" or "bad").

The Result

The program tells you what it guessed for the new fruit based on the questions and answers in the tree. This is like a computer making a guess about something it's never seen before.

In a nutshell: The program uses a tree of questions to guess what something is, just like playing 20 Questions. It uses information about known things to make the best guess about something new.

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

eps = np.finfo(float).eps
from numpy import log2 as log

def find_entropy(df):
    Class = df.keys()[-1]
    entropy = 0
    values = df[Class].unique()
    for value in values:
        fraction = df[Class].value_counts()[value] / len(df[Class])
        entropy += -fraction * np.log2(fraction)
    return entropy

def find_entropy_attribute(df, attribute):
    Class = df.keys()[-1]
    target_variables = df[Class].unique()
    variables = df[attribute].unique()
    entropy2 = 0
    for variable in variables:
        entropy = 0
        for target_variable in target_variables:
            num = len(df[attribute][df[attribute] == variable][df[Class] == target_variable]
            den = len(df[attribute][df[attribute] == variable])
            fraction = num / (den + eps)
            entropy += -fraction * log(fraction + eps)
        fraction2 = den / len(df)
        entropy2 += -fraction2 * entropy
    return abs(entropy2)

def find_winner(df):
    Entropy_att = []
    IG = []
    for key in df.keys()[:-1]:
        Entropy_att.append(find_entropy_attribute(df, key))
        IG.append(find_entropy(df) - find_entropy_attribute(df, key))
    return df.keys()[:-1][np.argmax(IG)]

def get_subtable(df, node, value):
    return df[df[node] == value].reset_index(drop=True)

def buildTree(df, tree=None):
    Class = df.keys()[-1]
    node = find_winner(df)
    attValue = np.unique(df[node])
    if tree is None:
        tree = {}
        tree[node] = {}
    for value in attValue:
        subtable = get_subtable(df, node, value)
        clValue, counts = np.unique(subtable[Class], return_counts=True)
        if len(counts) == 1:
            tree[node][value] = clValue[0]
        else:
            tree[node][value] = buildTree(subtable)
    return tree

df = pd.read_csv('playtennis.csv')  # Make sure 'playtennis.csv' contains your dataset

print("\nGiven Play Tennis Data Set:\n\n", df)

tree = buildTree(df)

print('The resultant decision tree is:')
pprint.pprint(tree)

test = {'Outlook': 'Sunny', 'Temperature': 'Hot', 'Humidity': 'High', 'Wind': 'Weak'}

def func(test, tree, default=None):
    attribute = next(iter(tree))
    print(attribute)
    if test[attribute] in tree[attribute].keys():
        print(tree[attribute].keys())
        print(test[attribute])
        result = tree[attribute][test[attribute]]
        if isinstance(result, dict):
            return func(test, result)
        else:
            return result
    else:
        return default
ans = func(test, tree)
print(ans)
```

```
Given Play Tennis Data Set:
     Outlook Temperature Humidity    Wind Play Tennis
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
```

```
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

The resultant decision tree is:
{'Outlook': {'Overcast': 'Yes',
             'Rain': {'Wind': {'Strong': 'No', 'Weak': 'Yes'}},
             'Sunny': {'Humidity': {'High': 'No', 'Normal': 'Yes'}}}}


Classification for the test sample: No
```