exp3.py

```python
'''
3) 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.
'''
import numpy as np
import pandas as pd
from collections import Counter

# Data Preparation
data_text = """
Outlook,Temperature,Humidity,Wind,PlayTennis
Sunny,Hot,High,Weak,No
Sunny,Hot,High,Strong,No
Overcast,Hot,High,Weak,Yes
Rain,Mild,High,Weak,Yes
Rain,Cool,Normal,Weak,Yes
Rain,Cool,Normal,Strong,No
Overcast,Cool,Normal,Strong,Yes
Sunny,Mild,High,Weak,No
Sunny,Cool,Normal,Weak,Yes
Rain,Mild,Normal,Weak,Yes
Sunny,Mild,Normal,Strong,Yes
Overcast,Mild,High,Strong,Yes
Overcast,Hot,Normal,Weak,Yes
Rain,Mild,High,Strong,No
"""

# This code snippet is preparing the data for further analysis. Here's what each line is doing:
data = [line.split(",") for line in data_text.strip().split("\n")]
df = pd.DataFrame(data[1:], columns=data[0])

def entropy(labels):
    total_count = len(labels)
    return -sum((count / total_count) * np.log2(count / total_count) for count in Counter(labels).values())

def information_gain(data, split_attribute, target_attribute):
    total_entropy = entropy(data[target_attribute])
    values, counts = np.unique(data[split_attribute], return_counts=True)
    weighted_entropy = sum((counts[i] / sum(counts)) * entropy(data[data[split_attribute] == values[i]][target_attribute])
                           for i in range(len(values)))
    return total_entropy - weighted_entropy

def id3(data, features, target_attribute):
    if len(np.unique(data[target_attribute])) == 1:
        return np.unique(data[target_attribute])[0]
    elif len(features) == 0:
        return Counter(data[target_attribute]).most_common(1)[0][0]
    else:
        best_feature = max(features, key=lambda feature: information_gain(data, feature, target_attribute))
        tree = {best_feature: {}}
        features = [feature for feature in features if feature != best_feature]
        for value in np.unique(data[best_feature]):
            subtree = id3(data[data[best_feature] == value], features, target_attribute)
            tree[best_feature][value] = subtree
        return tree

def classify(sample, tree):
    attribute = list(tree.keys())[0]
    if sample[attribute] in tree[attribute]:
        result = tree[attribute][sample[attribute]]
        if isinstance(result, dict):
            return classify(sample, result)
        else:
            return result
    else:
        return None

features = list(df.columns[:-1])
target_attribute = df.columns[-1]
decision_tree = id3(df, features, target_attribute)

new_sample = {'Outlook': 'Sunny', 'Temperature': 'Cool', 'Humidity': 'High', 'Wind': 'Strong'}
classification_result = classify(new_sample, decision_tree)

print("Constructed Decision Tree:")
print(decision_tree)
print("\nClassification Result for the New Sample:")
print(classification_result)

'''output:
Constructed Decision Tree:
{'Outlook': {'Overcast': 'Yes', 'Rain': {'Wind': {'Strong': 'No', 'Weak': 'Yes'}}, 'Sunny': {'Humidity': {'High': 'No', 'Normal': 'Yes'}}}}

Classification Result for the New Sample:
No
'''
```