EXPERIMENT 3

CODE

```
import math
from collections import Counter

# ---------------- Dataset (Play Tennis) ----------------
# Each record: [Outlook, Temperature, Humidity, Wind, PlayTennis]
dataset = [
    ['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']
]

attributes = ['Outlook', 'Temperature', 'Humidity', 'Wind']

# ---------------- Entropy ----------------
def entropy(data):
    labels = [row[-1] for row in data]
    total = len(labels)
    counts = Counter(labels)
    ent = 0
    for count in counts.values():
        p = count / total
        ent -= p * math.log2(p)
    return ent

# ---------------- Information Gain ----------------
def info_gain(data, attr_index):
    total_entropy = entropy(data)
    values = set(row[attr_index] for row in data)
    weighted_entropy = 0
    for value in values:
        subset = [row for row in data if row[attr_index] == value]
        weighted_entropy += (len(subset) / len(data)) * entropy(subset)
    return total_entropy - weighted_entropy

# ---------------- ID3 Algorithm ----------------
def id3(data, attrs):
    labels = [row[-1] for row in data]

    # If all examples have same label
    if labels.count(labels[0]) == len(labels):
        return labels[0]

    # If no attributes left
    if not attrs:
        return Counter(labels).most_common(1)[0][0]

    # Choose best attribute
    gains = [info_gain(data, i) for i in range(len(attrs))]
    best_attr_index = gains.index(max(gains))
    best_attr = attrs[best_attr_index]

    tree = {best_attr: {}}

    values = set(row[best_attr_index] for row in data)
    for value in values:
        subset = [row[:best_attr_index] + row[best_attr_index+1:]
                  for row in data if row[best_attr_index] == value]

        sub_attrs = attrs[:best_attr_index] + attrs[best_attr_index+1:]
        tree[best_attr][value] = id3(subset, sub_attrs)

    return tree

# ---------------- Classification ----------------
def classify(tree, attrs, sample):
    if not isinstance(tree, dict):
        return tree
    attr = next(iter(tree))
    attr_index = attrs.index(attr)
    value = sample[attr_index]
    return classify(tree[attr][value],
                    attrs[:attr_index] + attrs[attr_index+1:],
                    sample[:attr_index] + sample[attr_index+1:])
```

```
# ---------------- Build Decision Tree ----------------
decision_tree = id3(dataset, attributes)

print("Decision Tree:")
print(decision_tree)

# ---------------- Classify New Sample ----------------
new_sample = ['Sunny', 'Cool', 'High', 'Strong']
result = classify(decision_tree, attributes, new_sample)

print("\nNew Sample:", new_sample)
print("Classification Result:", result)
```

OUTPUT:-

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

New Sample: ['Sunny', 'Cool', 'High', 'Strong']
Classification Result: No
```