```
import math
from collections import Counter

# Sample dataset
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']

# Helper functions
def entropy(examples):
    total = len(examples)
    label_counts = Counter(row[-1] for row in examples)
    return -sum((count / total) * math.log2(count / total) for count in
label_counts.values())

def info_gain(examples, attr_index):
    total_entropy = entropy(examples)
    subsets = {}
    for row in examples:
        key = row[attr_index]
        subsets.setdefault(key, []).append(row)
    subset_entropy = sum((len(subset) / len(examples)) * entropy(subset)
for subset in subsets.values())
    return total_entropy - subset_entropy

def majority_class(examples):
    labels = [row[-1] for row in examples]
    return Counter(labels).most_common(1)[0][0]

def id3(examples, attrs):
    labels = [row[-1] for row in examples]
    if labels.count(labels[0]) == len(labels):
        return labels[0]
    if not attrs:
        return majority_class(examples)

    gains = [info_gain(examples, attributes.index(attr)) for attr in
attrs]
    best_attr = attrs[gains.index(max(gains))]
```

```
    tree = {best_attr: {}}
    attr_index = attributes.index(best_attr)
    attr_values = set(row[attr_index] for row in examples)

    for val in attr_values:
        subset = [row for row in examples if row[attr_index] == val]
        if not subset:
            tree[best_attr][val] = majority_class(examples)
        else:
            new_attrs = [a for a in attrs if a != best_attr]
            tree[best_attr][val] = id3(subset, new_attrs)

    return tree

# Build the tree
decision_tree = id3(dataset, attributes)

# Function to classify a sample
def classify(tree, sample):
    if isinstance(tree, str):
        return tree
    attr = next(iter(tree))
    attr_index = attributes.index(attr)
    attr_value = sample[attr_index]
    subtree = tree[attr].get(attr_value)
    if not subtree:
        return "Unknown"
    return classify(subtree, sample)

# Print the tree
import pprint
print("Decision Tree:")
pprint.pprint(decision_tree)

# Classify a new sample
sample = ['Sunny', 'Cool', 'High', 'Strong']
print("\nClassifying sample:", sample)
result = classify(decision_tree, sample)
print("Prediction:", result)
```