1BM22CS241

SANJEET PRAJWAL PANDIT

```
Entropy and Information Gain for each feature:
Feature: Outlook | Entropy: 0.9403 | Information Gain: 0.2467
Feature: Temperature | Entropy: 0.9403 | Information Gain: 0.0292
Feature: Humidity | Entropy: 0.9403 | Information Gain: 0.1518
Feature: Wind | Entropy: 0.9403 | Information Gain: 0.0481


Decision Tree:
    Outlook:
      Sunny ->          Humidity:
          High ->               No
          Normal ->               Yes
      Overcast ->         Yes
      Rainy ->          Wind:
          Weak ->               Yes
          Strong ->               No
```

12/3/25 DATE: PAGE:

lab2:

```
import pandas as pd
import numpy as np
// data = {  'Day': [...] , ... }
// df = pd.read_csv("weather.csv")  // Day, Outlook, Temp, Humidity, Wind, Decision
data = pd.read_csv("weather data.csv") , df = pd.DataFrame(data)
//function to calculate entropy
def entropy(target):
    class_counts = target.value_counts()
    probabilities = class_counts / len(target)
    return -np.sum(probabilities * np.log2(probabilities))

// function to calculate info gain
def information_gain(data, feature, target):
    entropy_before = entropy(target)
    feature_values = data[feature].unique()

    weighted_entropy = 0
    for value in feature_values:
        subset = target[data[feature] == value]
        weighted_entropy += (len(subset) / len(target)) * entropy(subset)

    return entropy_before - weighted_entropy

def print_entropy_and_gain(data, features, target):
    print("\n Entropy and information gain for each feature:")
    for feature in features:
        gain = information_gain(data, feature, target)
        ent = entropy(target)
        print(f"Feature: {feature} | Entropy: {ent:.4f} |
                                Information gain: {gain:.4f}")
```

DATE: PAGE

```
//function to build decision tree recursively
def build_tree(data, target, features):
    if len(target.unique()) == 1:
        return target.iloc[0]

    if len(features) == 0:
        return target.mode()[0]

    gains = {feature: information_gain(data, feature, target) for feature in
             features}

    best_feature = max(gains, key=gains.get)

    tree = {best_feature: {}}
    feature_values = data[best_feature].unique()

    for value in feature_values:
        subset_data = data[data[best_feature] == value]
        subset_target = target[data[best_feature] == value]

        remaining_features = [f for f in features if f != best_feature]

        subtree = build_tree(subset_data, subset_target, remaining_features)
        tree[best_feature][value] = subtree

    return tree
```

DATE: PAGE:

```
def print_tree(tree, indent=""):
    if isinstance(tree, dict):
        for feature, branches in tree.items():
            print(f"{indent}{feature}:")
            for value, subtree in branches.items():
                print(f"{indent} {value} → ", end="")
                print_tree(subtree, indent+"  ")
    else:
        print(f"{indent}{tree}")

target = df['Decision']
features = ['Outlook', 'Temperature', 'Humidity', 'Wind']

print_entropy_and_gain(df, features, target)

tree = build_tree(df, target, features)

print("\n Decision Tree:")
print_tree(tree, indent="  ")
```

Output:

Entropy and Information gain for each feature:

Feature: Outlook | Entropy: 0.9403 | Information gain: 0.2467

Feature: Temperature | Entropy: 0.9403 | Information gain: 0.0292

Feature: Humidity | Entropy: 0.9403 | Information gain: 0.1518

Feature: Wind | Entropy: 0.9403 | Information gain: 0.048

Decision Tree:

Outlook:

Sunny → Humidity:

High → No

Normal → Yes

Overcast → Yes

Rainy → Wind:

Weak → Yes

Strong → No

Tue 12/3/25