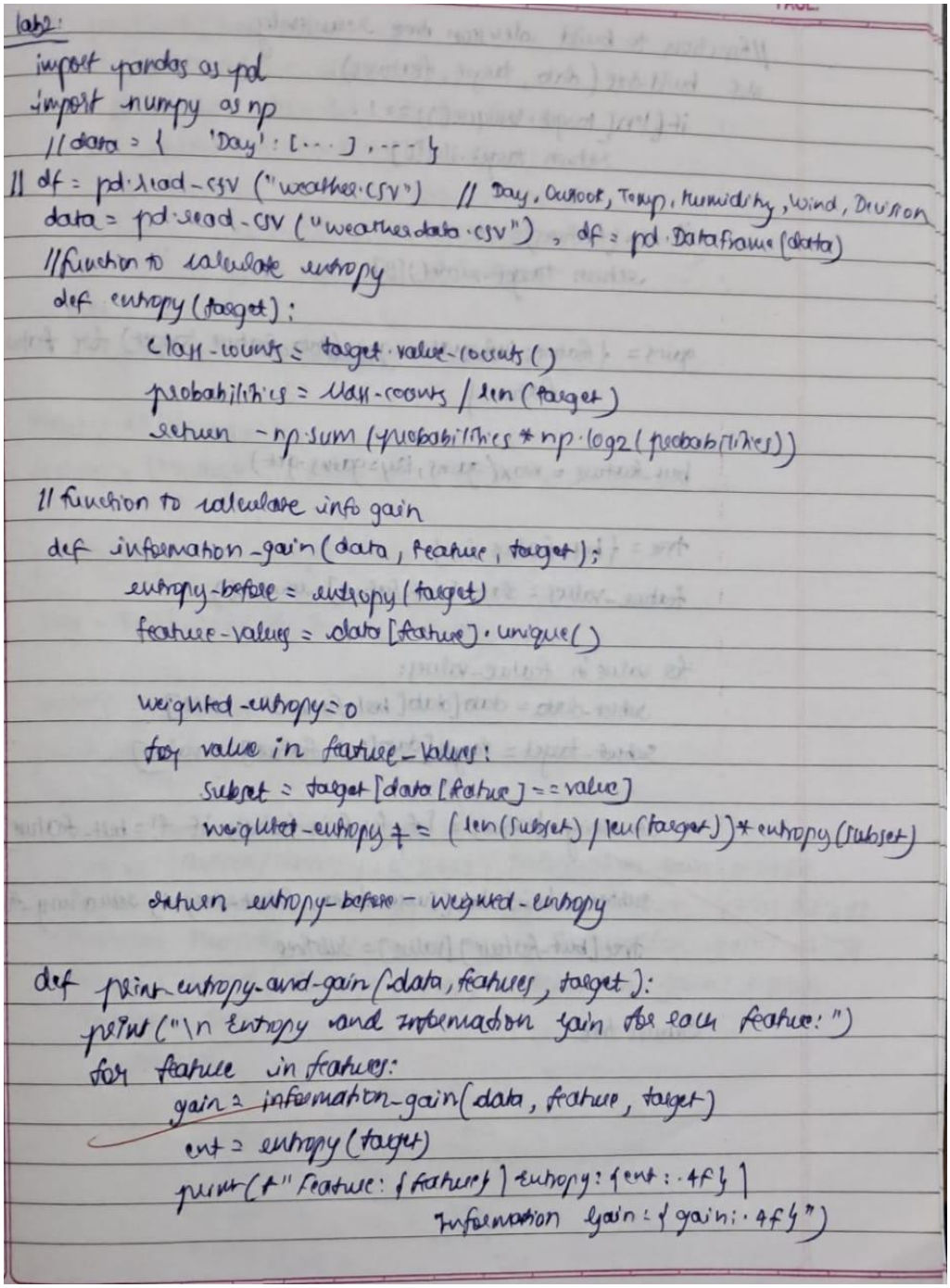
Program 5

Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new sample

Screenshot:



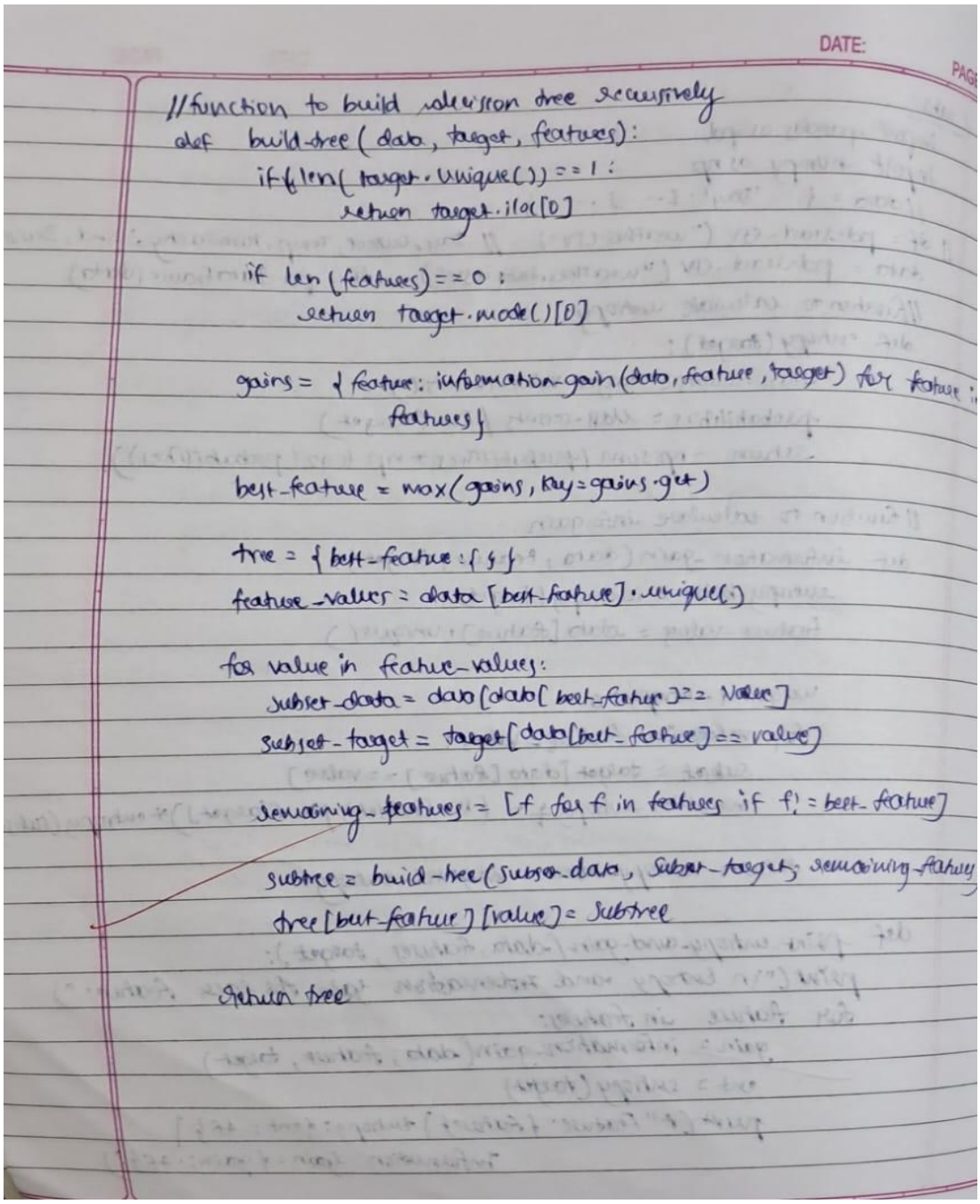
```
lab2:
import pandas as pd
import numpy as np
// data = {  'Day': [...] , ... }
// df = pd.read_csv("weather.csv")  // Day, Outlook, Temp, Humidity, Wind, Decision
data = pd.read_csv("weather_data.csv") , df = pd.DataFrame(data)
//function to calculate entropy
def entropy(target):
    class_counts = 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}")
```



```
//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
```

Code:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn import tree
import matplotlib.pyplot as plt

# Load the newly uploaded dataset
data = pd.read_csv('/content/seattle-weather.csv')

# Data Preparation
data_cleaned = data.drop('date', axis=1)
label_encoder = LabelEncoder()
data_cleaned['weather'] = label_encoder.fit_transform(data_cleaned['weather'])

# Split data into features and target
X = data_cleaned.drop('weather', axis=1)
y = data_cleaned['weather']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the Decision Tree Classifier with improved clarity
id3_classifier = DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state=42)
id3_classifier.fit(X_train, y_train)

# Visualize the Decision Tree with larger font size for clarity
plt.figure(figsize=(20, 12))
```

```
tree.plot_tree(id3_classifier, feature_names=X.columns, class_names=list(label_encoder.classes_),
filled=True, fontsize=10)

plt.title("ID3 Decision Tree Classifier (Enhanced Clarity)")

plt.show()
```