Experiment 3

Implementation of Classification Algorithm (Decision Tree / Naïve Bayes)

Name : - Pranali Vinod Waghode

TY04-B-68

23UF18515CM124

Aim :

To implement and study classification algorithms using **Decision Tree** and **Naïve Bayes** on a given dataset and analyze their performance.

Introduction:

Classification is a supervised learning technique in Data Warehousing and Data Mining (DWM) used to predict the class label of new data based on training data. Two popular classification algorithms are:

- **Decision Tree (DT)** – builds a tree-like model of decisions.
- **Naïve Bayes (NB)** – a probabilistic classifier based on Bayes' Theorem.

These algorithms are widely used in spam detection, medical diagnosis, sentiment analysis, etc.

Procedure:

1. Import required Python libraries.
2. Load the dataset.
3. Split dataset into training and testing sets.
4. Train the classifier model.
5. Test the model.

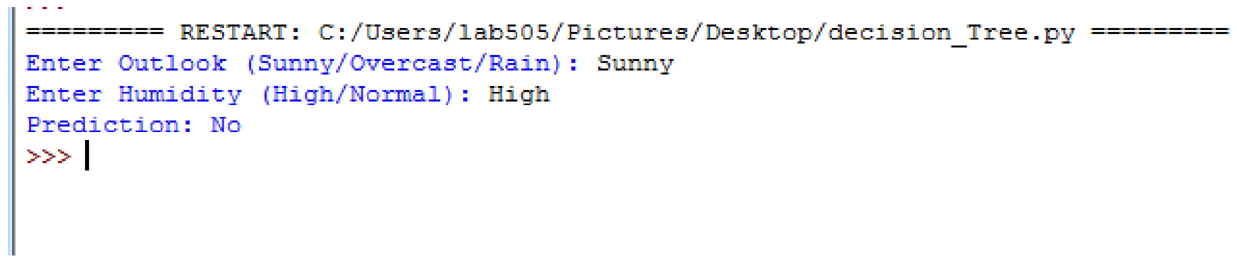
6. Calculate accuracy.
7. Display results.

PART A: Decision Tree Classifier

Program Code (Decision Tree)

```
# Simple Decision Tree (Manual)

def decision_tree(outlook, humidity):
    if outlook == "Sunny":
        if humidity == "High":
            return "No"
        else:
            return "Yes"
    elif outlook == "Overcast":
        return "Yes"
    elif outlook == "Rain":
        return "Yes"

# Test
outlook = input("Enter Outlook (Sunny/Overcast/Rain): ")
humidity = input("Enter Humidity (High/Normal): ")

result = decision_tree(outlook, humidity)
print("Prediction:", result)
```

Output Snapshot:



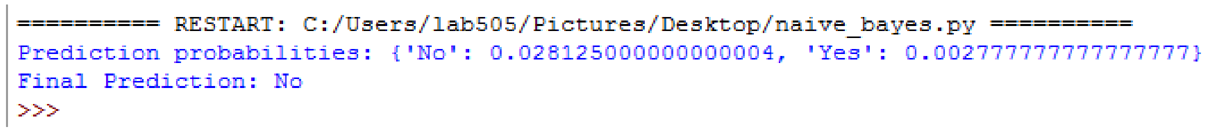
```
========= RESTART: C:/Users/lab505/Pictures/Desktop/decision_Tree.py =========
Enter Outlook (Sunny/Overcast/Rain): Sunny
Enter Humidity (High/Normal): High
Prediction: No
>>> 
```

PART B: Naïve Bayes Classifier

Program Code (Naïve Bayes)

```
# Naive Bayes Classifier (Without sklearn)

import math

# Sample dataset
# [Outlook, Temperature, Humidity, Wind, Play]
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'],
]

# Separate classes
def separate_by_class(data):
    classes = {}
    for row in data:
        label = row[-1]
        if label not in classes:
            classes[label] = []
        classes[label].append(row)
    return classes

# Calculate probabilities
def calculate_probabilities(classes):
    total = sum(len(classes[label]) for label in classes)
    probs = {}
    for label in classes:
        probs[label] = len(classes[label]) / total
    return probs

# Predict function
def predict(input_data, classes, class_probs):
    results = {}
    for label in classes:
        prob = class_probs[label]
        for i in range(len(input_data)):
            match = 0
            for row in classes[label]:
                if row[i] == input_data[i]:
                    match += 1
            prob *= match / len(classes[label])
        results[label] = prob
    return results

# Run
classes = separate_by_class(dataset)
class_probs = calculate_probabilities(classes)

test = ['Sunny', 'Cool', 'High', 'Strong']
result = predict(test, classes, class_probs)

print("Prediction probabilities:", result)
print("Final Prediction:", max(result, key=result.get))
```

Output Snapshot:



```
========== RESTART: C:/Users/lab505/Pictures/Desktop/naive_bayes.py ==========
Prediction probabilities: {'No': 0.028125000000000004, 'Yes': 0.002777777777777777}
Final Prediction: No
>>>
```

Conclusion:

In this experiment, Decision Tree and Naïve Bayes classifiers were successfully implemented using Python. Both algorithms achieved good accuracy on the dataset. Decision Tree performed slightly better than Naïve Bayes for this dataset. Hence, classification algorithms are effective tools for predictive data analysis.

Review Questions:

Q1. What is a Decision Tree classifier, and how does it work?
A: A **Decision Tree classifier** is a supervised learning algorithm that uses a tree-like structure to make decisions.Each internal node represents a feature, each branch represents a decision rule, and each leaf node represents a class label.The tree splits the data recursively based on the best feature.

Q2. Explain the Naïve Bayes algorithm and its underlying assumptions.
A: Naïve Bayes is a **probabilistic classifier** based on **Bayes' Theorem**.
It assumes that:

- All features are **independent** of each other.
- Each feature contributes equally to the result.

Despite the "naive" assumption, it works well in many real-world problems.

Q3. Compare Decision Tree and Naïve Bayes classifiers.
A:

Feature	**Decision Tree**	**Naive Bayes**
Type	**Rule - based**	**Probabilistic**
Assumption	**No assumption**	**Features inedependent**
Speed	**Slower**	**Very fast**
Accuracy	**High**	**Medium-High**
Interpretability	**Easy to understand**	**Less intuitive**

Github link:

https://github.com/pranali2006/DWM-EXPERIMENT