PART A

(PART A : TO BE REFFERED BY STUDENTS)

Experiment No.07

A.1 Aim:

Implementation of Naïve Bayes Algorithm using any programming language like Python

A.2 Prerequisite:

Familiarity with the programming languages

A.3 Outcome:

After successful completion of this experiment students will be able to □ Use classification and clustering algorithms of data mining.

A.4 Theory:

THEORY:

Introduction to Bayesian Classification

The Bayesian Classification represents a supervised learning method as well as a statistical method for classification. Assumes an underlying probabilistic model and it allows us to capture uncertainty about the model in a principled way by determining probabilities of the outcomes. It can solve diagnostic and predictive problems. This Classification is named after Thomas Bayes (1702-1761), who proposed the Bayes Theorem. Bayesian classification provides practical learning algorithms and prior knowledge and observed data can be combined. Bayesian Classification provides a useful perspective for understanding and evaluating many learning algorithms. It calculates explicit probabilities for hypothesis and it is robust to noise in input data. Uses of Naive Bayes classification:

1. Naive Bayes text classification (http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html) The Bayesian classification is used as a probabilistic learning method (Naive Bayes text classification). Naive Bayes classifiers are among the most successful known algorithms for learning to classify text documents.

2. Spam filtering (http://en.wikipedia.org/wiki/Bayesian_spam_filtering) Spam filtering is the best known use of Naive Bayesian text classification. It makes use of a naive Bayes classifier to identify spam e-mail. Bayesian spam filtering has become a popular mechanism to distinguish illegitimate spam email from legitimate email (sometimes called "ham" or "bacn")[4] Many modern mail clients implement Bayesian spam filtering. Users can also install separate email filtering programs. Server-side email filters, such as DSPAM, SpamAssassin, SpamBayes, Bogofilter and ASSP, make use of Bayesian spam filtering techniques, and the functionality is sometimes embedded within mail server software itself.

3. Hybrid Recommender System Using Naive Bayes Classifier and Collaborative Filtering (http://eprints.ecs.soton.ac.uk/18483/) Recommender Systems apply machine learning and data mining techniques for filtering unseen information and can predict whether a user would like a given resource.

It is proposed a unique switching hybrid recommendation approach by combining a Naive Bayes classification approach with the collaborative filtering. Experimental results on two different data sets, show that the proposed algorithm is scalable and provide better performance–in terms of accuracy and coverage–than other algorithms while at the same time eliminates some recorded problems with the recommender systems.

1. Online applications (http://www.convo.co.uk/x02/) .This online application has been set up as a simple example of supervised machine learning and affective computing. Using a training set of examples which reflect nice, nasty or neutral sentiments, we're training Ditto to distinguish between them. Simple Emotion Modelling, combines a statistically based classifier with a dynamical model. The Naive Bayes classifier employs single words and word pairs as features. It allocates user utterances into nice, nasty and neutral classes, labelled +1, -1 and 0 respectively. This numerical output drives a simple first-order dynamical system, whose state represents the simulated emotional state of the experiment's personification, Ditto the donkey.

Naïve Bayesian Classification

It is based on the Bayesian theorem. It is particularly suited when the dimensionality of the inputs is high. Parameter estimation for naive Bayes models uses the method of maximum likelihood. In spite over-simplified assumptions, it often performs better in many complex real world situations. Advantage: Requires a small amount of training data to estimate the parameters

DATASET : https://raw.githubusercontent.com/jbrownlee/Dataset/master/iris.csv

For Example:

- **Given training data X, *posteriori probability of a hypothesis* H, P(H|X), follows the Bayes' theorem**

$$P(H \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid H)P(H)}{P(\mathbf{X})} = P(\mathbf{X} \mid H) \times P(H) / P(\mathbf{X})$$

- **Informally, this can be viewed as posteriori = likelihood x prior/evidence**
- **Predicts X belongs to C_i iff the probability $P(C_i|X)$ is the highest among all the $P(C_k|X)$ for all the *k* classes**
- **Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost Given:**

Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no' Data to be classified:

X = (age <=30,

Income = medium,

Student = yes

Credit_rating = Fair)

age	income	student	credit_rating	buys_computer	
<=30	high	no	fair	no	
<=30	high	no	excellent	no	
31…40	high	no	fair	yes	
>40	medium	no	fair	yes	
>40	low	yes	fair	yes	
>40	low	yes	excellent	no	
31…40	low	yes	excellent	yes	
<=30	medium	no	fair	no	
<=30	low	yes	fair	yes	
>40	medium	yes	fair	yes	
<=30	medium	yes	excellent	yes	
31…40	medium	no	excellent	yes	
31…40	high	yes	fair	yes	
>40	medium	no	excellent	no	

- **P(C_i): P(buys_computer = "yes") = 9/14 = 0.643 P(buys_computer = "no") = 5/14= 0.357**
- **Compute P(X|C_i) for each class**

P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222
P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6
P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

P(X|C_i) : P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019
P(X|C_i)*P(C_i) : P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007
Therefore, X belongs to class ("buys_computer = yes")

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

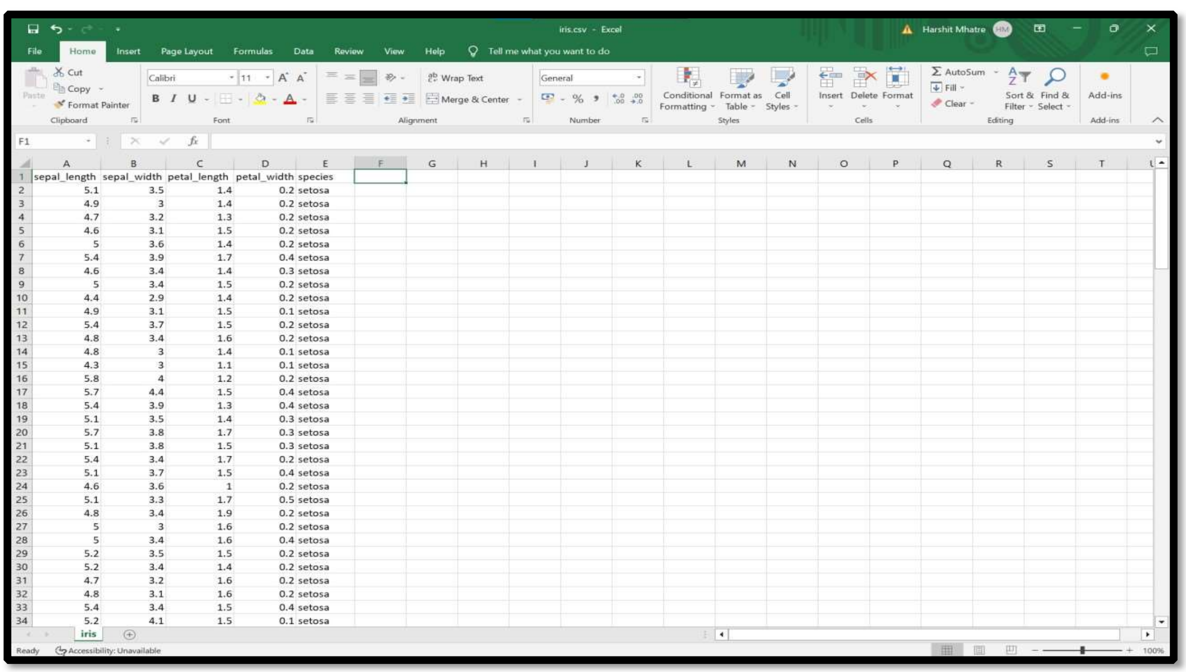
Roll. No.: A17	Name: Ritesh Yadav
Class: TE (AI&DS)	Batch: A1
Date of Experiment: 30-Sep-2024	Date of Submission: 07-Oct-2024
Grade:	

B.1 Software Code written by student:

```
import pandas as pd from sklearn.model_selection import
train_test_split from sklearn.naive_bayes import
GaussianNB from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt import numpy as np from
sklearn.datasets import load_iris

# Load the Iris dataset iris = load_iris() data =
pd.DataFrame(data=iris.data, columns=iris.feature_names) data['target']
= iris.target

# Selecting features and target variable
X = data[['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']] y =
data['target']  # Use target as the target variable

# 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 a Gaussian Naive Bayes classifier classifier =
GaussianNB()

# Train the classifier on the training data classifier.fit(X_train,
y_train)

# Make predictions on the test data using both Gaussian probability and Naive Bayes
probability y_prob_naive_bayes = classifier.predict_proba(X_test)
# Calculate accuracy using Gaussian probability y_pred_gaussian =
classifier.predict(X_test) accuracy_gaussian = accuracy_score(y_test,
y_pred_gaussian) print("Accuracy using Gaussian Probability:",
accuracy_gaussian)

# Print the probabilities for the top 5 records
top_5_probabilities = y_prob_naive_bayes[:5] for i,
probabilities in enumerate(top_5_probabilities):
  print(f"\nRecord {i + 1} - True Class: {y_test.iloc[i]}, Probabilities:", probabilities)
```

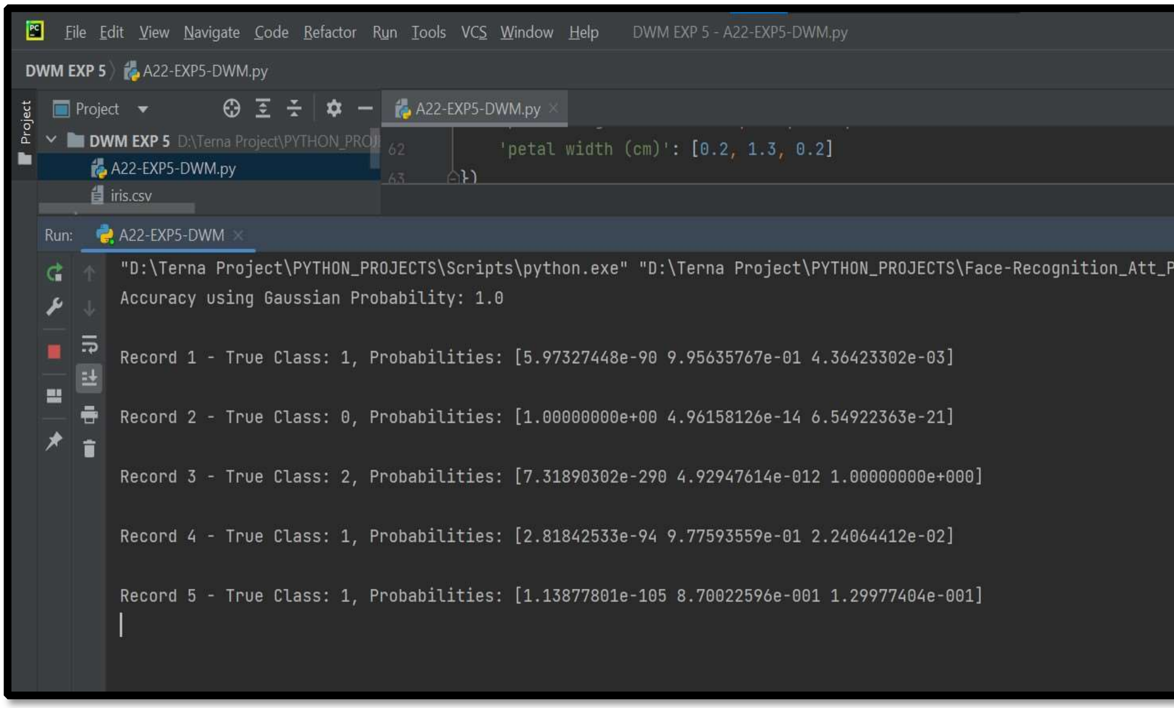
```
# Plot the probability distribution for each class classes =
np.unique(y_train)

plt.figure(figsize=(12, 6))

for class_name in classes:
    prob = y_prob_naive_bayes[:, class_name]     plt.hist(prob,
bins=20, label=f'Class {class_name}', alpha=0.5)

plt.xlabel('Probability') plt.ylabel('Frequency') plt.title('Naive Bayes
Probability Distribution for Iris Classes') plt.legend(loc='best')
plt.savefig('Probability Distribution') plt.show()

# You can now use the trained classifier to predict class labels for new data # For
example:
new_data = pd.DataFrame({
    'sepal length (cm)': [5.1, 6.2, 4.8],
    'sepal width (cm)': [3.5, 2.9, 3.4],
    'petal length (cm)': [1.4, 4.3, 1.9],
    'petal width (cm)': [0.2, 1.3, 0.2]
})
predicted_class = classifier.predict(new_data) print("\nPredicted Classes using Gaussian
    Probability:", predicted_class)
```

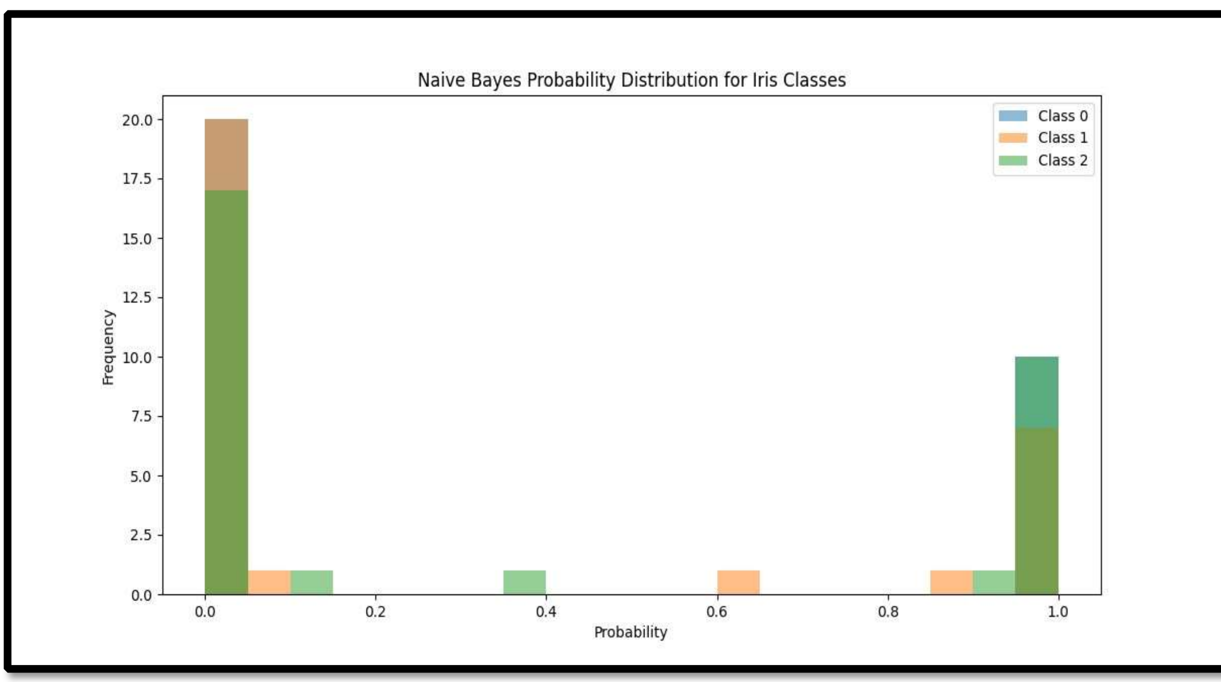
B.2 Input and Output:



sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3	1.4	0.1	setosa
4.3	3	1.1	0.1	setosa
5.8	4	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa
5.4	3.4	1.7	0.2	setosa
5.1	3.7	1.5	0.4	setosa
4.6	3.6	1	0.2	setosa
5.1	3.3	1.7	0.5	setosa
4.8	3.4	1.9	0.2	setosa
5	3	1.6	0.2	setosa
5	3.4	1.6	0.4	setosa
5.2	3.5	1.5	0.2	setosa
5.2	3.4	1.4	0.2	setosa
4.7	3.2	1.6	0.2	setosa
4.8	3.1	1.6	0.2	setosa
5.4	3.4	1.5	0.4	setosa
5.2	4.1	1.5	0.1	setosa

iris.csv dataset





Probability distribution

B.3 Observations and learning:

Observations:
In this experiment, the Iris dataset was used to implement a Gaussian Naive Bayes classifier for classification. The data was split into training and testing sets, and the classifier's probabilities were calculated. Visualizing the probability distribution helped understand prediction confidence.

Learnings:
This experiment highlighted the principles of supervised learning with Naive Bayes. Gaussian Naive Bayes suits datasets with continuous features. Data splitting for evaluation and using accuracy as a metric were crucial. Additionally, interpreting probability distributions enhances understanding of the model's confidence in predictions.

B.4 Conclusion:

In this experiment, we successfully applied the Gaussian Naive Bayes classifier to the Iris dataset, showcasing its effectiveness in handling continuous feature data. Through this practical exercise, we learned the significance of essential concepts like data splitting, model evaluation, and probability analysis. This experience has enriched our understanding of supervised learning and its practical application in real-world classification tasks.

B.5 Question of Curiosity

Q1: How many instances and attributes are in the data set?
The Iris dataset comprises 150 instances and 4 attributes. Each instance represents a unique iris flower sample, and the dataset's attributes include measurements for sepal length, sepal width, petal length, and petal width of these iris flowers.

Q2: What is the minimum, maximum and mean values of that attributes?
The minimum, maximum, and mean values for each of the four attributes in the Iris dataset:

Sepal Length:
Minimum: 4.3 cm
Maximum: 7.9 cm
Mean: Approximately 5.84 cm

Sepal Width:
Minimum: 2.0 cm
Maximum: 4.4 cm
Mean: Approximately 3.05 cm

Petal Length:
Minimum: 1.0 cm
Maximum: 6.9 cm
Mean: Approximately 3.76 cm

Petal Width:
Minimum: 0.1 cm
Maximum: 2.5 cm
Mean: Approximately 1.20 cm

Q3: What is the accuracy of the classifier?

Accuracy using Gaussian Probability: 1.0

Record 1 - True Class: 1, Probabilities: [5.97327448e-90 9.95635767e-01 4.36423302e-03]
Record 2 - True Class: 0, Probabilities: [1.00000000e+00 4.96158126e-14 6.54922363e-21]
Record 3 - True Class: 2, Probabilities: [7.31890302e-290 4.92947614e-012 1.00000000e+000]
Record 4 - True Class: 1, Probabilities: [2.81842533e-94 9.77593559e-01 2.24064412e-02]
Record 5 - True Class: 1, Probabilities: [1.13877801e-105 8.70022596e-001 1.29977404e-001]

Predicted Classes using Gaussian Probability: [0 1 0]