**Experiment NO: 6** **Date:**

**Aim:** To implement Naïve Bayes Classifier.

**Theory:** The Naïve Bayes classifier is a simple probabilistic classifier based on Bayes' theorem with an assumption of independence among features. Here's a theoretical overview of the Naïve Bayes classifier:

**Bayes' Theorem:**

P(A|B) = (P(B|A).P(A)) / P(B)

Where:

P(A|B) is the probability of event A given that event B has occurred.

P(B|A) is the probability of event B given that event A has occurred.

P(A) and P(B) are the probabilities of events A and B, respectively.

**Naive Bayes Assumption:**

The Naïve Bayes classifier makes the assumption that the features used to describe an observation are conditionally independent given the class label. Mathematically, this can be expressed as:

P(X1, X2, . . . , Xn | C) = P(X1|C).P(X2|C) . . . . .P(Xn|C)

Where:

X1, X2, . . . Xn are the features describing an observation.

C is the class label.

**Classification Rule:**

The goal of the Naïve Bayes classifier is to predict the class label (C) for a given set of features (X1, X2, . . . , Xn). The classifier assigns the class label that maximizes the posterior probability:

Predicted Class = arg maxc P(C| X1, X2, . . . , Xn

Using Bayes' theorem and the Naïve Bayes assumption, this can be simplified to:

Predicted Class = arg maxc P(C) . pi(from i=1 to n) P(Xi|C)

Training the Naïve Bayes Classifier:

1. Prior Probability P(C):

- Estimate the prior probability of each class based on the training data.

2. Likelihood P(Xi | C):

- For each feature Xi and each class C, estimate the likelihood of Xi given C from the training data.

3. Prediction:

- Given a new observation with features X1, X2, . . . , Xn, calculate the posterior probability for each class and predict the class with the highest probability.

Pros and Cons:

Pros:

- Simple and easy to implement.

- Works well in practice, especially for text classification.

Cons:

- Assumes independence among features, which may not hold in real-world scenarios.

- Sensitive to irrelevant features.

- Requires a sufficient amount of training data.

Despite its simplicity and assumptions, Naïve Bayes often performs surprisingly well in various applications, especially when the independence assumption is approximately met or when there is limited training data.

**Code:**

11/14/23, 06:22 PM bayes file:///C:/P Jeevesh Naidu/college/honours/sem 5/bayes.html 1/2

In [ ]:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

In [ ]:

# Load the dataset

diabetes = pd.read\_csv('Diabetes\_RF.csv')

col\_names = list(diabetes.columns)

predictors = col\_names[0:8]

target = col\_names[8]

print(diabetes.head())

Out[ ]:

Number of times pregnant Plasma glucose concentration

0 6 148

1 1 85

2 8 183

3 1 89

4 0 137

Diastolic blood pressure Triceps skin fold thickness

0 72 35

1 66 29

2 64 0

3 66 23

4 40 35

2-Hour serum insulin Body mass index Diabetes pedigree function

0 0 33.6 0.627

1 0 26.6 0.351

2 0 23.3 0.672

3 94 28.1 0.167

4 168 43.1 2.288

Age (years) Class variable

0 50 YES

1 31 NO

2 32 YES

3 21 NO

4 33 YES

In [ ]:

# Split the dataset into training and testing sets

from sklearn.model\_selection import train\_test\_split

train, test = train\_test\_split(diabetes, test\_size=0.3, random\_state=0)

In [ ]:

# Gaussian Naive Bayes

from sklearn.naive\_bayes import GaussianNB

Gmodel = GaussianNB()

train\_pred\_gau = Gmodel.fit(train[predictors], train[target]).predict(train[predictors])

test\_pred\_gau = Gmodel.fit(train[predictors], train[target]).predict(test[predictors])

In [ ]:

train\_acc\_gau = np.mean(train\_pred\_gau == train[target])

test\_acc\_gau = np.mean(test\_pred\_gau == test[target])

In [ ]:

print("Gaussian Naive Bayes:")

print(f"Training Accuracy: {train\_acc\_gau:.3f}")

print(f"Testing Accuracy: {test\_acc\_gau:.3f}")

print()

Out[ ]:

Gaussian Naive Bayes:

Training Accuracy: 0.767

Testing Accuracy: 0.762

In [ ]:

# Display results using Matplotlib

fig, ax = plt.subplots()

# Set axis properties to hide the axes

ax.set\_axis\_off()

# Create a table to display results

table\_data = [

["", "Training Accuracy", "Testing Accuracy"],

["Gaussian Naive Bayes", f"{train\_acc\_gau:.3f}", f"{test\_acc\_gau:.3f}"]

]

table = ax.table(cellText=table\_data, loc='center', cellLoc='center', colLabels=None, colColours=None)

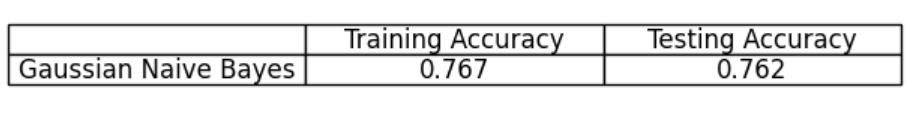
table.auto\_set\_font\_size(False)

table.set\_fontsize(12)

table.scale(1.2, 1.2)

plt.show()

Out[ ]:



**Conclusion:**

Naïve Bayes Classifier was studied and successfully implemented.