Assignment 4

Name: Vaishnavi Gosavi
Roll No: 71
Class: TY CSAI-A
Batch: 2

Title: Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

Description:-

- **Naive Bayes Classifiers**
  Naive Bayes classifiers are supervised machine learning algorithms used for classification tasks, based on Bayes' Theorem to find probabilities. This article will give you an overview as well as more advanced use and implementation of Naive Bayes in machine learning. It is named as "Naive" because it assumes the presence of one feature does not affect other features. The "Bayes" part of the name refers to for the basis in Bayes' Theorem.

- **Key Features of Naive Bayes Classifiers**
  - The main idea behind the Naive Bayes classifier is to use Bayes' Theorem to classify data based on the probabilities of different classes given the features of the data. It is used mostly in high-dimensional text classification
  - The Naive Bayes Classifier is a simple probabilistic classifier and it has very few number of parameters which are used to build the ML models that can predict at a faster speed than other classification algorithms.
  - It is a probabilistic classifier because it assumes that one feature in the model is independent of existence of another feature. In other words, each feature contributes to the predictions with no relation between each other.
  - Naïve Bayes Algorithm is used in spam filtration, Sentimental analysis, classifying articles and many more.

- **Assumption of Naive Bayes**
  - The fundamental Naive Bayes assumption is that each feature makes an:
  - Feature independence: This means that when we are trying to classify something, we assume that each feature (or piece of information) in the data does not affect any other feature.
  - Continuous features are normally distributed: If a feature is continuous, then it is assumed to be normally distributed within each class.
  - Discrete features have multinomial distributions: If a feature is discrete, then it is assumed to have a multinomial distribution within each class.
  - Features are equally important: All features are assumed to contribute equally to the prediction of the class label.
  - No missing data: The data should not contain any missing values.

- **Understanding Bayes' Theorem for naive bayes**

  Bayes' Theorem gives the probability of a class CCC given the features (or attributes) X={x1,x2,...,xn} as:

  P(C|X)=P(X|C)/P(C)

  Where:

  - P(C|X) is the posterior probability of class C given the features X.
  - P(X|C) is the likelihood of observing the features X given the class CCC.
  - P(C)is the prior probability of the class C.
  - P(X) is the evidence or the total probability of observing the features X.

- **Applications of Naive Bayes Classifier**
  - *Spam Email Filtering:* Classifies emails as spam or non-spam based on features.
  - *Text Classification:* Used in sentiment analysis, document categorization, and topic classification.
  - *Medical Diagnosis*: Helps in predicting the likelihood of a disease based on symptoms.
  - *Credit Scoring:* Evaluates creditworthiness of individuals for loan approval.
  - *Weather Prediction*: Classifies weather conditions based on various factors.

- **ALGORITHM**

1. Preprocess the data (handle missing values, normalize if needed)
2. For each class C:
   - Compute prior probability P(C) = count(C) / total instances
3. For each feature x_i and each class C:
   - Compute mean (mu_C) and standard deviation (sigma_C) for feature x_i in class C
4. For a new instance X = {x_1, x_2, ..., x_n}:
   - For each class C:
   - Calculate the likelihood P(x_i | C) for each feature x_i using Gaussian distribution
   - Calculate the posterior probability P(C | X) = P(C) * product(P(x_i | C))
5. Predict the class with the highest posterior probability:
   - Predicted class: argmax(P(C | X))

Date: / /

NAIVE BAYES

→ Predict if a person buys a laptop based on age & income
Classify if a middle-aged person with medium income will buy or not

Dataset

Age	Income	Buys Laptop
Young	Low	No
Young	Medium	No
Young	High	Yes
Middle	Low	No
Middle	Medium	Yes
Middle	High	Yes
Senior	Low	Yes
Senior	Medium	Yes
Senior	High	No

Total no of instances = 9
Total no of yes = 5
Total no. of no = 4

$$P(\text{Yes}) = \frac{5}{9} = 0.55 \qquad P(\text{No}) = \frac{4}{9} = 0.44$$

Conditional Probabilities

For Yes –

Age = Middle (2 Yes)

$$\therefore P(\text{Middle} \mid \text{Yes}) = \frac{2}{5} = 0.4$$

$$P(\text{Medium} \mid \text{Yes}) = \frac{2}{5} = 0.4$$

Date: / /

For No

P Age (Middle | No) = $\frac{1}{4}$ = (0.25)

P Age (Medium | No) = $\frac{1}{4}$ (0.25)

Step 3: NAIVE BAYES

for yes

P(Yes | Medium) x P(Yes | Middle) x P(Yes)

= $\frac{2}{5} \times \frac{2}{8} \times \frac{8}{9}$

= $\frac{4}{45}$

= 0.0896

for no

P(No | Medium) x P(No | Middle) x P(No)

= $\frac{1}{4} \times \frac{1}{4} \times \frac{4}{9}$

= $\frac{1}{36}$

= 0.0275

P(Yes) > P(No)

∴ Yes

Date: / /

★ NAIVE BAYES CLASSIFIER

- supervised, based on conditional probability

$$P\left(\frac{A}{B}\right) = \frac{P\left(\frac{B}{A}\right) P(A)}{P(B)}$$

$$P\left(\frac{Yes}{x}\right) = R\left(\frac{y}{x_1 x_2 \ldots x_n}\right) = \frac{P\left(\frac{x_1}{y}\right) P\left(\frac{x_2}{y}\right) \ldots P(y)}{P(x_1)\, P(x_2) \ldots P(x_n)}$$

fruit = {yellow, sweet, long}

fruit	yellow	sweet	long	
orange	350	450	0	650
banana	400	300	350	400
other	50	100	50	150
	800	850	400	

$$P\left(\frac{f}{orange}\right) = P\left(\frac{yellow}{orange}\right) + P\left(\frac{sweet}{orange}\right) + P\left(\frac{long}{orange}\right)$$

$$P\left(\frac{yellow}{orange}\right) = \frac{P\left(\frac{orange}{yellow}\right) + P(yellow)}{P(orange)}$$

$$= \frac{\frac{350}{800} \times \frac{800}{1200}}{\frac{650}{1200}} = 0.5$$

$$P\left(\frac{Sweet}{orange}\right) = \frac{P\left(\frac{orange}{sweet}\right) + P(sweet)}{P(Sweet)}$$

$$= \frac{\frac{450}{850} \times \frac{850}{1200}}{\frac{650}{1200}}$$

Date: / /

$$P\left(\frac{long}{orange}\right) = \frac{P\left(\frac{orange}{long}\right) * P(long)}{P(orange)}$$

$$= 0$$

$$P\left(\frac{fruit}{banana}\right) = P\left(\frac{yellow}{banana}\right) * P\left(\frac{sweet}{banana}\right) * P\left(\frac{long}{banana}\right)$$

$$P\left(\frac{f}{others}\right) = P\left(\frac{yellow}{others}\right) * P\left(\frac{sweet}{others}\right) * P\left(\frac{long}{others}\right)$$

original question

n-dimensional plane represent in 2D plane

Decision tree

$$\text{Entropy} = \left(\frac{P}{P+N} \log \frac{P}{P+N}\right) - \left(\frac{N}{P+N} \log \frac{N}{P+N}\right)$$

$P = Yes = 9 \qquad N = No = 5 \qquad P+N = 14$

$$= \frac{9}{14} \log \frac{9}{14} - \frac{5}{14} \log \frac{5}{14}$$

$$= 0.94$$

Info. Gain (node) = entropy of whole data − entropy of (node) − entropy of (node)

Naives Bayes using predefined functions

```
# @title Naives Bayes using predefined functions
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create a Gaussian Naïve Bayes classifier
nb_classifier = GaussianNB()

# Train the classifier
nb_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = nb_classifier.predict(X_test)

# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.9777777777777777
```

Naives Bayes using user-defined functions

```
# @title Naives Bayes using user-defined functions
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('/content/User_Data.csv')
x = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

```
dataset
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

Next steps: Generate code with dataset | View recommended plots | New interactive sheet

```
y
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 0, 1])
```

```
# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)
```

```
GaussianNB
GaussianNB()
```

```
# Predicting the Test set results
y_pred = classifier.predict(x_test)
```

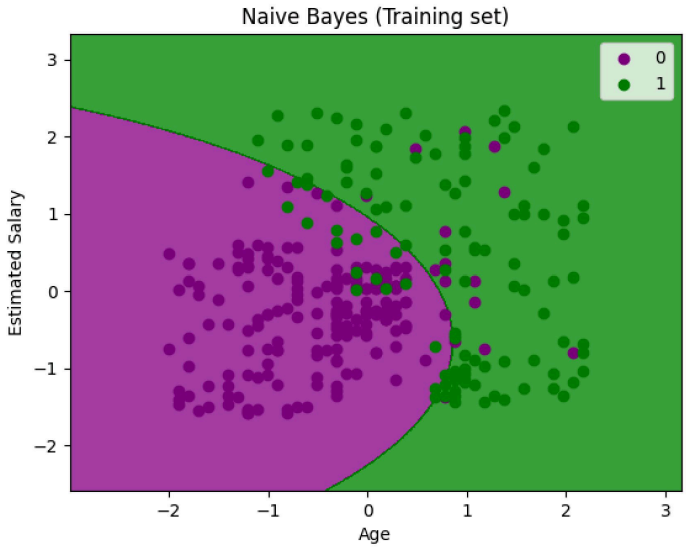
```
# Evaluate the classifier using additional metrics
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[65  3]
 [ 7 25]]
              precision    recall  f1-score   support

           0       0.90      0.96      0.93        68
           1       0.89      0.78      0.83        32

    accuracy                           0.90       100
   macro avg       0.90      0.87      0.88       100
weighted avg       0.90      0.90      0.90       100
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
X1, X2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step = 0.01),
                     nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(X1, X2, classifier.predict(nm.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('purple', 'green')))
mtp.xlim(X1.min(), X1.max())
mtp.ylim(X2.min(), X2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
                c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Naive Bayes (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

```
<ipython-input-19-ef2bc75a4d1e>:11: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value
  mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
```




```
# Visualising the Test set results
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
X1, X2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step = 0.01),
                     nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(X1, X2, classifier.predict(nm.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('purple', 'green')))
mtp.xlim(X1.min(), X1.max())
mtp.ylim(X2.min(), X2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
                c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Naive Bayes (test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

```
<ipython-input-20-5e1ce6421104>:11: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value
  mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
```