.

Assignment No: 6

Title :

1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset.

2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

Objective:

- Students should be able to data analysis using Naïve Bayes classification algorithm using Python for any open-source dataset.

Prerequisite:

1. Basic of Python Programming

2.Concept of Join and Marginal Probability.

Contents for Theory:

1. Concepts used in Naïve Bayes classifier.
2. Naive Bayes Example
3. Confusion Matrix Evaluation Metrics

1. Concepts used in Naïve Bayes classifier.

- Naïve Bayes Classifier can be used for Classification of categorical data.
    - Let there be a 'j' number of classes. C={1,2,….j}
    - Let, input observation is specified by 'P' features. Therefore input observation x is given , x = {F1,F2,…..Fp}
    - The Naïve Bayes classifier depends on Bayes' rule from probability theory.
- Prior probabilities: Probabilities which are calculated for some event based on no other information are called Prior probabilities.

For example, P(A), P(B), P(C) are prior probabilities because while calculating P(A), occurrences of event B or C are not concerned i.e. no information about occurrence of any other event is used.

Conditional Probabilities:

$$P\left(\frac{A}{B}\right) = \frac{P(A \cap B)}{P(B)} \qquad if\ P(B) \neq 0 \qquad \ldots \ldots .(1)$$

$$P\left(\frac{B}{A}\right) = \frac{P(B \cap A)}{P(A)} \qquad \ldots \ldots \ldots (2)$$

From equation (1) and (2) ,

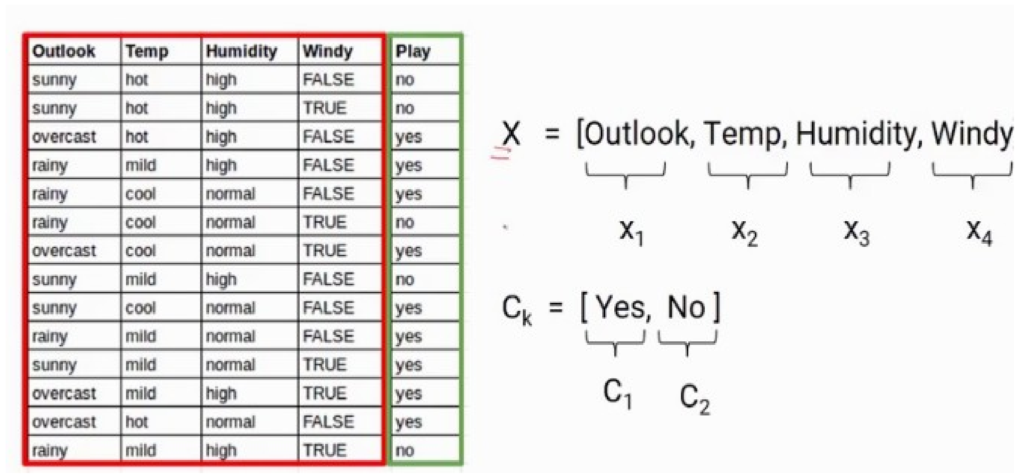
$$P(A \cap B) = P\left(\frac{A}{B}\right).P(B) = P\left(\frac{B}{A}\right).P(A)$$

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

Is called the Bayes Rule.

2. **Example of Naive Bayes**

We have a dataset with some features Outlook, Temp, Humidity, and Windy, and the target here is to predict whether a person or team will play tennis or not.



Outlook	Temp	Humidity	Windy	Play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

$$X = [\underbrace{\text{Outlook}}_{X_1}, \underbrace{\text{Temp}}_{X_2}, \underbrace{\text{Humidity}}_{X_3}, \underbrace{\text{Windy}}_{X_4}$$

$$C_k = [\underbrace{\text{Yes}}_{C_1}, \underbrace{\text{No}}_{C_2}]$$

Conditional Probability

Here, we are predicting the probability of class1 and class2 based on the given condition. If I try to write the same formula in terms of classes and features, we will get the following equation

$$P(C_k \mid X) = \frac{P(X \mid C_k) * P(C_k)}{P(X)}$$

Now we have two classes and four features, so if we write this formula for class C1, it will be something like this.

$$P(C_1 \mid x_1 \cap x_2 \cap x_3 \cap x_4) = \frac{P(x_1 \cap x_2 \cap x_3 \cap x_4 \mid C_1) * P(C_1)}{P(x_1 \cap x_2 \cap x_3 \cap x_4)}$$

Here, we replaced Ck with C1 and X with the intersection of X1, X2, X3, X4. You might have a question, It's because we are taking the situation when all these features are present at the same time.

The Naive Bayes algorithm assumes that all the features are independent of each other or in other words all the features are unrelated. With that assumption, we can further simplify the above formula and write it in this form

$$\boxed{P(C_1 \mid x_1 \cap x_2 \cap x_3 \cap x_4) = \frac{P(x_1 \mid C_1) * P(x_2 \mid C_1) * P(x_3 \mid C_1) * P(x_4 \mid C_1) * P(C_1)}{P(x_1) * P(x_2) * P(x_3) * P(x_4)}}$$

This is the final equation of the Naive Bayes and we have to calculate the probability of both C1 and C2.For this particular example.

Outlook	Temp	Humidity	Windy	Play
Rainy	Cool	High	True	?

$$P(Yes \mid X) = P(Rainy \mid Yes) \times P(Cool \mid Yes) \times P(High \mid Yes) \times P(True \mid Yes) \times P(Yes)$$

$$P(Yes \mid X) = 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = \boxed{0.00529} \rightarrow 0.2 = \frac{0.00529}{0.02057 + 0.00529}$$

$$P(No \mid X) = P(Rainy \mid No) \times P(Cool \mid No) \times P(High \mid No) \times P(True \mid No) \times P(No)$$

$$P(No \mid X) = 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = \boxed{0.02057} \rightarrow 0.8 = \frac{0.02057}{0.02057 + 0.00529}$$

P (N0 | Today) > P (Yes | Today) So, the prediction that golf would be played is ‘No’.

1) **Algorithm (Iris Dataset)**:

Step 1: Import libraries and create alias for Pandas, Numpy and Matplotlib

Step 2: Import the Iris dataset by calling URL.

Step 3: Initialize the data frame

Step 4: Perform Data Preprocessing

- Convert Categorical to Numerical Values if applicable
- Check for Null Value
- Divide the dataset into Independent (X) and Dependent (Y) variables.
- Split the dataset into training and testing datasets.
- Scale the Features if necessary.

2) **Step 5: Use Naive Bayes algorithm(Train the Machine) to Create Model**

```
# import the class
from sklearn.naive_bayes import GaussianNBgaussian =
GaussianNB() gaussian.fit(X_train, y_train)
```

3) **Step 6: Predict the y_pred for all values of train_x and test_x**

```
Y_pred=gaussian.predict(X_test)
```

4) **Step 7:Evaluate the performance of Model for train_y and test_y**

```
accuracy=accuracy_score(y_test,Y_pred)
```

```
precision =precision_score(y_test, Y_pred,average='micro')recall =
recall_score(y_test, Y_pred,average='micro')
```

5) **Step 8: Calculate the required evaluation parameters**

```
from sklearn.metrics import precision_score,confusion_matrix,accuracy_score,recall_score
cm = confusion_matrix(y_test, Y_pred)
```

Conclusion:

In this way we have done data analysis using Naive Bayes Algorithm for Iris dataset and evaluated the performance of the model.

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score,
```

In [5]:
```
iris = load_iris()
X = iris.data
y = iris.target
target_names = iris.target_names
```

In [6]:
```
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['target'] = iris.target
print("First 5 rows of the dataset:")
print(df.head())
```

```
First 5 rows of the dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
target               0
dtype: int64
```

In [8]:
```
df.head()
```

Out[8]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [9]:
```
X = df.drop('target', axis=1)
y = df['target']
```

In [11]:
```
target_names = iris.target_names
target_names
```

Out[11]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

Split the dataset into training and testing datasets

In [13]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Data Transformation

In [14]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Use Naive Bayes algorithm

In [15]:
```
gaussian = GaussianNB()
gaussian.fit(X_train, y_train)
```

Out[15]:
```
▾ GaussianNB
GaussianNB()
```

Predicting Values

In [21]:
```
y_pred=gaussian.predict(X_test)
```

In [22]:
```
y_pred
```

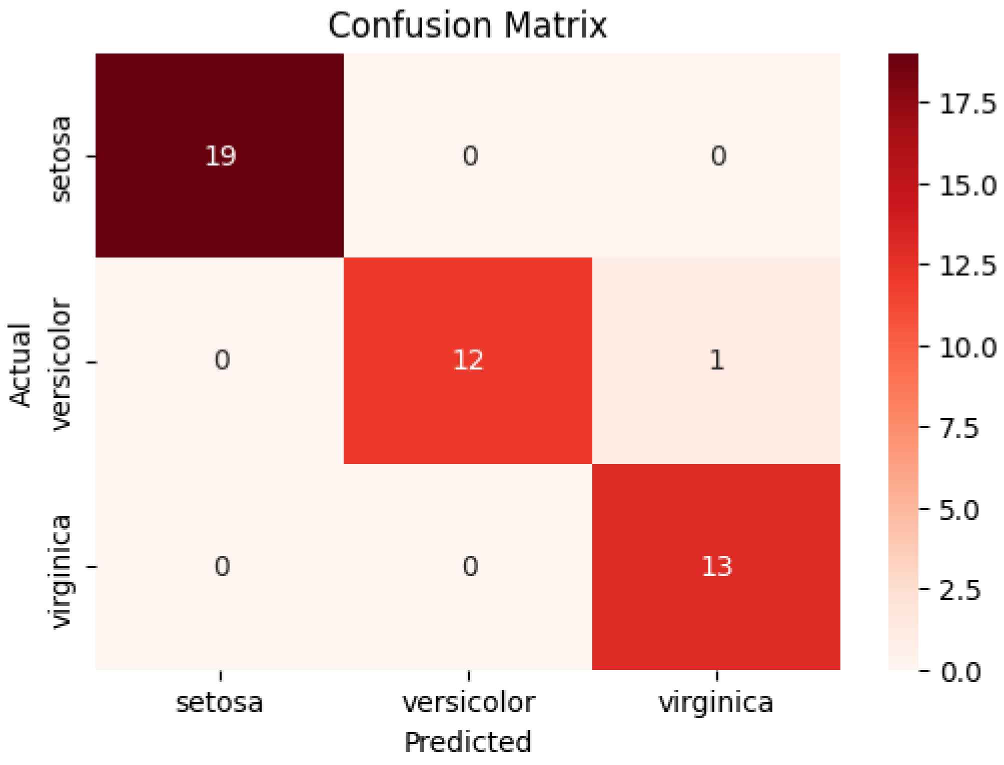
Out[22]:
```
array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 2, 2, 1, 1, 2, 0, 2,
       0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 2, 1, 1, 0,
       0])
```

Evaluting Modal

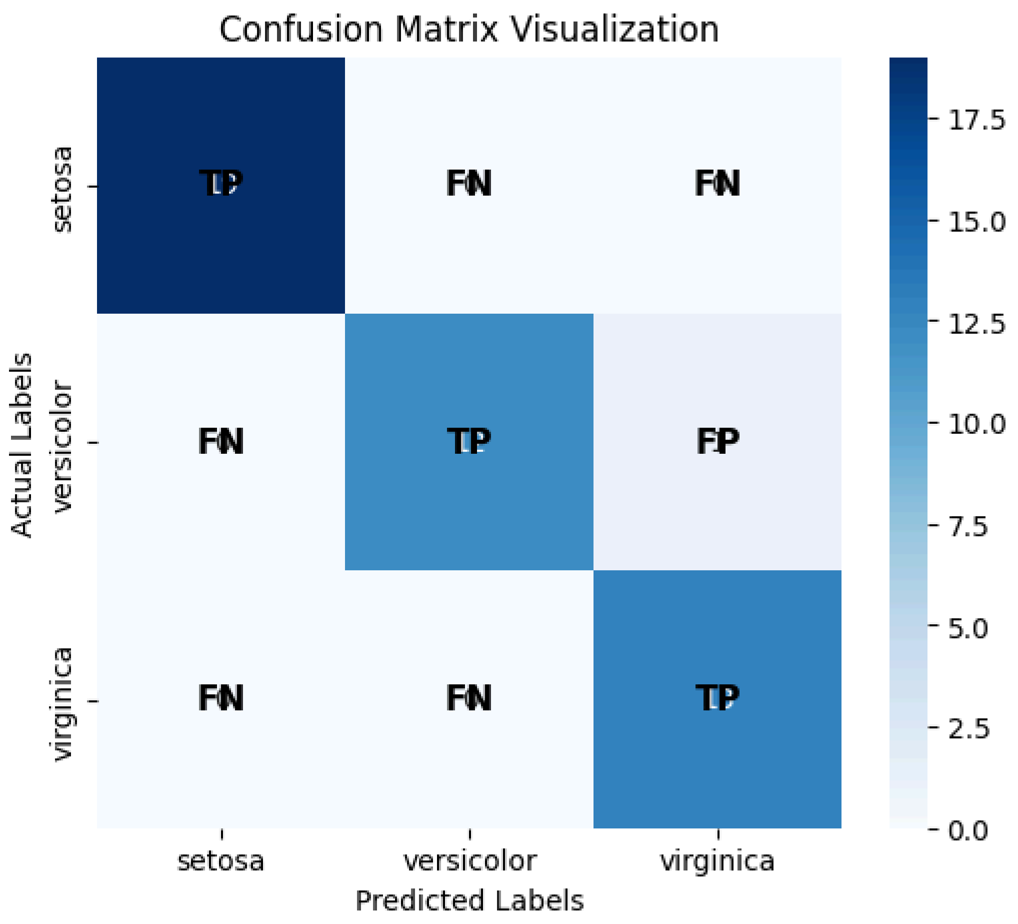
In [26]:
```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred,average='micro' )
recall = recall_score(y_test, y_pred, average='micro')

print("\nModel Performance on Test Set:")
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision (micro-average): {precision:.4f}")
print(f"Recall (micro-average): {recall:.4f}")

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(cm)
```

```
Model Performance on Test Set:
Accuracy: 0.9778
Precision (micro-average): 0.9778
Recall (micro-average): 0.9778

Confusion Matrix:
[[19  0  0]
 [ 0 12  1]
 [ 0  0 13]]
```

In [30]:
```python
import seaborn as sns
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Reds",
            xticklabels=iris.target_names,
            yticklabels=iris.target_names)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



In [31]:
```python
import seaborn as sns
fig, ax = plt.subplots(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=iris.target_names, yticklabels

plt.xlabel('Predicted Labels')
plt.ylabel('Actual Labels')
plt.title('Confusion Matrix Visualization')


for i in range(len(cm)):
    for j in range(len(cm)):
        plt.text(j + 0.5, i + 0.5, f"TP" if i == j else "FP" if cm[i, j] > 0 and i != j else
                 ha='center', va='center', color='black', fontsize=12, fontweight='bold')

plt.show()
```


Confusion Matrix Visualization
TP
FN
FN
FN
TP
FP
FN
FN
TP
setosa
versicolor
virginica
Actual Labels
setosa
versicolor
virginica
Predicted Labels
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0

```
In [ ]:
```