Practical 6: Data Analytics III

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.

The dataset contains:

- sepal_length, sepal_width, petal_length, petal_width: Numeric variables
- species: Categorical variable with three categories: 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica'

```
import pandas as pd
import numpy as np
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

# Load the Iris dataset
iris = load_iris()
print(iris.keys())  # View available dataset keys
print(iris)

# Convert the dataset into a Pandas DataFrame
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
print(df)

# Add the target column to the DataFrame
df['target'] = iris.target

# Display the first few rows of the dataset
print(df.head())

y=df['target']
X=df.drop(['target'], axis=1)

# Split data into train and test sets (67% train, 33% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

# Initialize the Naïve Bayes classifier model
```

```
model = GaussianNB()

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

# Predict on the test data
y_pred = model.predict(X_test)

comparison_df = pd.DataFrame({
    "Actual": y_test,
    "Predicted": y_pred
})
comparison_df

# Compute the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
print(cm.sum())

# Extract TP, FP, TN, FN for each class
TP = np.diag(cm) #True Positives (TP): Diagonal elements of the confusion matrix
FP = cm.sum(axis=0) - TP  #False Positives (FP): Sum of the each column - TP
FN = cm.sum(axis=1) - TP # False Negatives (FN): Sum of each row - TP
TN = cm.sum() - (TP + FP + FN) # True Negatives (TN): total number of samples in the
confusion matrix -(TP+FP+FN )

print("\nTrue Positives (TP):", TP)
print("False Positives (FP):", FP)
print("False Negatives (FN):", FN)
print("True Negatives (TN):", TN)

accuracy = accuracy_score(y_test, y_pred)
error_rate = 1 - accuracy
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
```

average='macro' : Computes precision for each class and takes the average
average='micro' : Computes global precision considering all classes together
average='weighted' : Computes precision for each class and weights it by the number of samples in that class

```
    Predicted
     0   1   2
   ------------
0 | 10  0   0  (Setosa)
1 |  0   9   1  (Versicolor)
2 |  0   1   9  (Virginica)
```