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.

```
# Step 1: Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


# Step 2: Import the Iris dataset using URL
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"


# Step 3: Initialize the DataFrame
columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
df = pd.read_csv(url, header=None, names=columns)


# Step 4: Data Preprocessing

# Convert Categorical to Numerical Values (Label Encoding)
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['species'] = le.fit_transform(df['species'])  # Convert species names to integers



# Check for Null Values
print("Null values:\n", df.isnull().sum())
```

```
Null values:
 sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

```
# Step 4 (continued): Divide dataset into Independent (X) and Dependent (y) variables
X = df.drop('species', axis=1)
y = df['species']


# Split the dataset into Training and Testing datasets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Scale the features if necessary
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


# Step 5: Use Naive Bayes Algorithm
from sklearn.naive_bayes import GaussianNB
gaussian = GaussianNB()
gaussian.fit(X_train, y_train)
```

```
GaussianNB
GaussianNB()
```

```
# Step 6: Predict Y_pred
y_pred = gaussian.predict(X_test)



# Step 7: Evaluate Model Performance
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix


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(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
```

```
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
```

```
# Step 8: Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
```

```
Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Start coding or generate with AI.