Assignment: Gaussian Naive Bayes Classification

Aim

To implement a Gaussian Naive Bayes classifier on a sample 2D dataset, predict the class labels, evaluate the model's accuracy, and visualize the data.

Lab Setup

- Python 3.x environment (Google Colab, Jupyter Notebook, or local IDE)
- Required libraries:
    - numpy
    - matplotlib
    - scikit-learn
- Installation command (if needed):

```bash
pip install numpy matplotlib scikit-learn
```

- Dataset: Synthetic 2D points defined within the code.

Input

- Sample 2D data points and class labels:

```python
X = np.array([[1, 2], [2, 3], [3, 3], [6, 6], [7, 7], [8, 8]])
y = np.array([0, 0, 0, 1, 1, 1])
```

- Train-test split: 80% training, 20% test set.

Expected Output

- Accuracy of the Gaussian Naive Bayes model on the test set (printed value)
- Scatter plot displaying data points colored by their original labels.

Example output:

```text
Accuracy: 1.00
```

Theory / Algorithm

Gaussian Naive Bayes (GNB) is a probabilistic classifier based on applying Bayes' theorem with the assumption that features are conditionally independent given the class. For continuous features, GNB assumes feature values follow a Gaussian (normal) distribution.

The classification process involves:

1. Calculating prior probabilities of each class.
2. Calculating the likelihood of feature values under each class using the Gaussian probability density function.
3. Applying Bayes' theorem to compute the posterior probability for each class.
4. Assigning the class with the highest posterior probability to the sample.

Code

```
python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# Sample 2D data points and labels
X = np.array([[1, 2], [2, 3], [3, 3], [6, 6], [7, 7], [8, 8]])
y = np.array([0, 0, 0, 1, 1, 1])
# Split data into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Create and train Gaussian Naive Bayes model
gnb = GaussianNB()
gnb.fit(X_train, y_train)
# Predict on test set
y_pred = gnb.predict(X_test)
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
# Plot data points colored by true labels
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, edgecolors='k', cmap=plt.cm.Paired)
plt.title("Gaussian Naive Bayes Classification")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

Conclusion / Discussion

The Gaussian Naive Bayes classifier successfully learned from the small training dataset and achieved an accuracy score indicating perfect classification on the test set (dependent on random split). Visualization shows the original data points colored by their classes. GNB is computationally efficient and suitable for datasets where features approximately follow normal distributions and the independence assumption is reasonable.