Data Analytics III - Naive Bayes Classification (Practical 6)

Theory and Definitions

1. Naive Bayes Classifier:

- A probabilistic classifier based on Bayes' Theorem with strong (naive) independence assumptions.

- Formula: P(A|B) = [P(B|A) * P(A)] / P(B)

- Used for classification tasks and assumes that the presence of a feature is unrelated to the presence of any other.

2. Gaussian Naive Bayes:

- Assumes that the features follow a normal distribution.

- Commonly used for continuous data (like the iris dataset).

3. Confusion Matrix:

- A table used to evaluate the performance of a classification model.

- It provides the number of true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN).

4. Accuracy, Error Rate, Precision, Recall:

- Accuracy = (TP + TN) / Total

- Error Rate = 1 - Accuracy

- Precision = TP / (TP + FP)

- Recall = TP / (TP + FN)

5. Dataset: iris.csv

- A well-known dataset containing measurements of sepal and petal length/width for three iris

species.

Code with Comments

```
# Import necessary libraries
import pandas as pd
import numpy as np
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 dataset
df = pd.read_csv('iris.csv')

# Extract features and labels
x = df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
y = df['species']

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)

# Create and train the model
model = GaussianNB()
model.fit(X_train, y_train)

# Make predictions
y_predict = model.predict(X_test)

# Evaluate the model
model.score(X_train, y_train)  # Training Accuracy
model.score(X_test, y_test)    # Testing Accuracy

# Confusion matrix and metrics
cm = confusion_matrix(y_test, y_predict)
TP = np.diag(cm)
FP = cm.sum(axis=0) - TP
FN = cm.sum(axis=1) - TP
TN = cm.sum() - (TP + FP + FN)

# Print metrics
accuracy = accuracy_score(y_test, y_predict)
error_rate = 1 - accuracy
precision = precision_score(y_test, y_predict, average='macro')
recall = recall_score(y_test, y_predict, average='macro')
print(f"Accuracy: {accuracy:.2f}")
print(f"Error Rate: {error_rate:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
```