DAY 18 **[15 - JULY - 2025]**

SUPERVISED MACHINE LEARNING

- *k-Nearest Neighbors (k-NN)*
- *Naive Bayes classifiers*

k-Nearest Neighbors (k-NN)

The k-Nearest Neighbors algorithm is a simple, instance-based learning method used for classification and regression. It works by finding the 'k' closest data points (neighbors) to a new data point and making a prediction based on the majority class (for classification) or average (for regression) of those neighbors. The algorithm is:

- **Non-parametric**: Makes no assumptions about the underlying data distribution
- **Distance-based**: Typically uses Euclidean distance to measure closeness
- **Simple but effective**: Works well for smaller datasets with clear class boundaries

Key parameters include the number of neighbors (k) and the distance metric.

Naive Bayes Classifier

Naive Bayes is a probabilistic classifier based on Bayes' Theorem, assuming independence between features. Despite this "naive" assumption, it often performs well, particularly for text classification. It calculates the probability of each class given the input features and selects the class with the highest probability. Key characteristics:

- **Fast and efficient**: Suitable for large datasets
- **Handles categorical and numerical data**: Different variants (e.g., GaussianNB for continuous data)
- **Robust to noise**: Performs well even with missing or noisy data

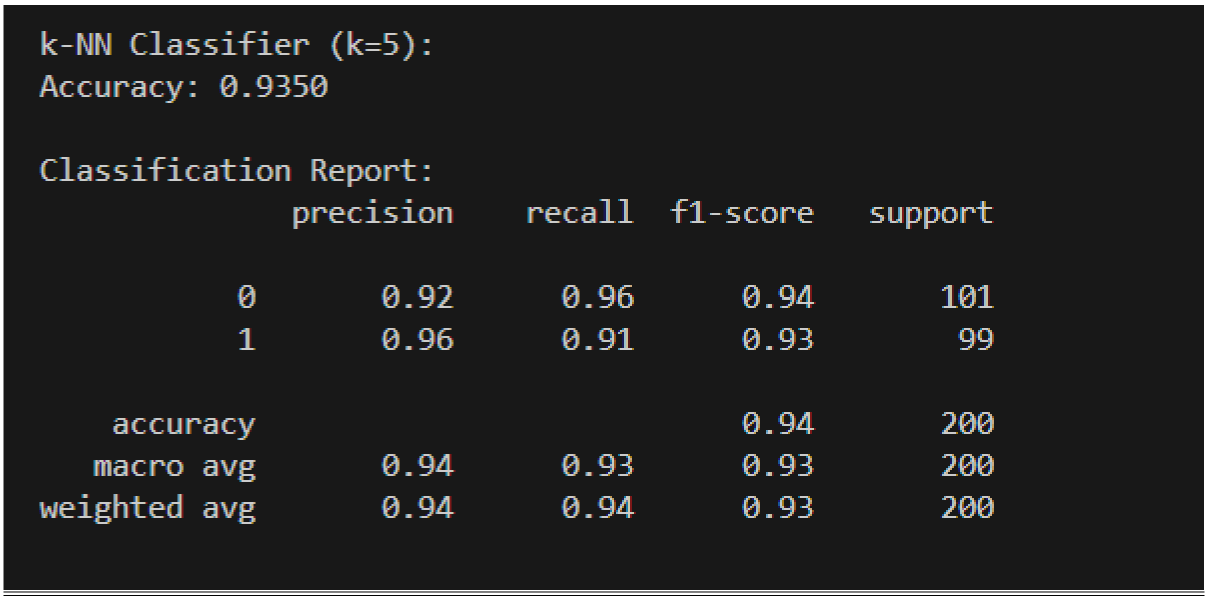
The following Python code demonstrates both classifiers using scikit-learn on a synthetic dataset.

Python Implementation

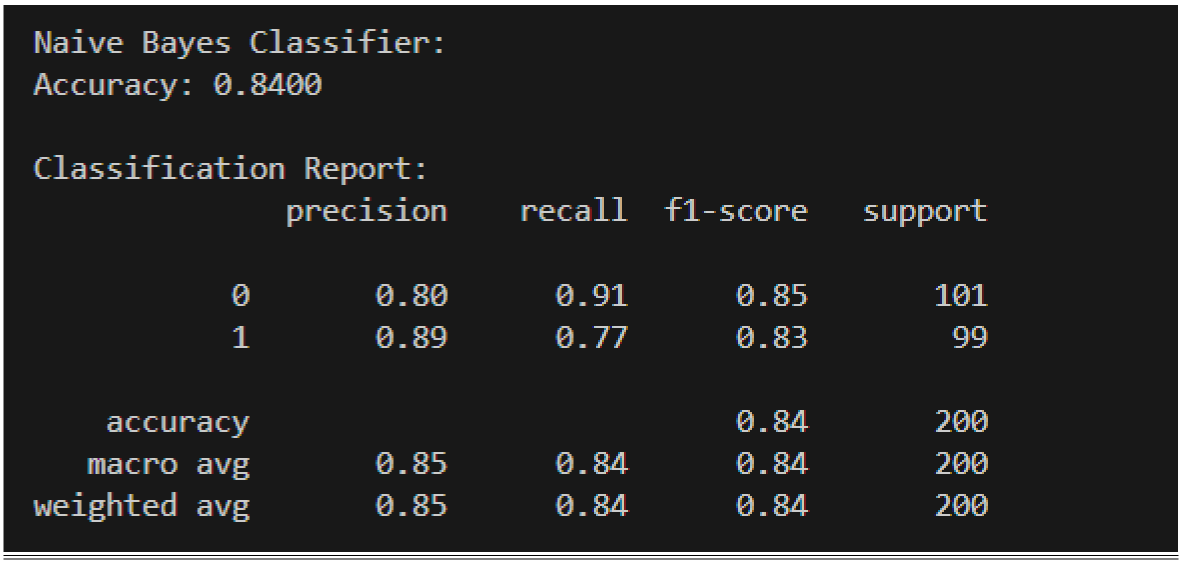
```
# Import necessary libraries
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
import numpy as np

# Generate synthetic dataset for classification
X, y = make_classification(n_samples=1000, n_features=4, n_classes=2,
random_state=42)

# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# k-Nearest Neighbors Classifier
def knn_classifier(X_train, X_test, y_train, y_test, k=5):
    # Initialize the k-NN classifier
    knn = KNeighborsClassifier(n_neighbors=k)

    # Train the model
    knn.fit(X_train, y_train)

    # Make predictions
    y_pred = knn.predict(X_test)

    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)

    print(f"\nk-NN Classifier (k={k}):")
    print(f"Accuracy: {accuracy:.4f}")
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))

    return knn

# Naive Bayes Classifier
def naive_bayes_classifier(X_train, X_test, y_train, y_test):
    # Initialize the Gaussian Naive Bayes classifier
    nb = GaussianNB()

    # Train the model
    nb.fit(X_train, y_train)

    # Make predictions
    y_pred = nb.predict(X_test)

    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)

    print("\nNaive Bayes Classifier:")
    print(f"Accuracy: {accuracy:.4f}")
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))

    return nb

# Run both classifiers
if __name__ == "__main__":
    # Run k-NN classifier with k=5
    knn_model = knn_classifier(X_train, X_test, y_train, y_test, k=5)

    # Run Naive Bayes classifier
    nb_model = naive_bayes_classifier(X_train, X_test, y_train, y_test)
```

Output:



```
k-NN Classifier (k=5):
Accuracy: 0.9350

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.96      0.94       101
           1       0.96      0.91      0.93        99

    accuracy                           0.94       200
   macro avg       0.94      0.93      0.93       200
weighted avg       0.94      0.94      0.93       200
```



```
Naive Bayes Classifier:
Accuracy: 0.8400

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.91      0.85       101
           1       0.89      0.77      0.83        99

    accuracy                           0.84       200
   macro avg       0.85      0.84      0.84       200
weighted avg       0.85      0.84      0.84       200
```