Green University of Bangladesh
Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester: (Spring, Year:2024), B.Sc. in CSE (Day)

Lab Report NO 02
Course code: CSE412 section: 222D3

Lab Experiment Name: KNN from Scratch: Flower and News Classification with Custom Evaluation Metrics

Student Details

	Name	ID
1.	Anjumand Binte Mahmud	222902005

Submission Date : 7-8-2025
Course Teacher's Name : Md. Sabbir Hosen Mamun

Lab Report Status

Marks: …… **Signature:.....**
Comments:..... **Date:.....**

1. # Report Title:

KNN from Scratch: Flower and News Classification with Custom Evaluation Metrics.

2. # Implement the K-Nearest Neighbors (KNN) algorithm from scratch using Python:

```
 import numpy as np
from collections import Counter
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Step 1: Distance Function
def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))

# Step 2: KNN Class
class KNN:
    def __init__(self, k=3):
        self.k = k

    def fit(self, X, y):
        self.X_train = X
        self.y_train = y

    def predict(self, X):
        predictions = [self._predict(x) for x in X]
        return np.array(predictions)

    def _predict(self, x):
        # Compute distances to all training samples
        distances = [euclidean_distance(x, x_train) for x_train in self.X_train]

        # Get indices of k nearest neighbors
        k_indices = np.argsort(distances)[:self.k]
        k_nearest_labels = [self.y_train[i] for i in k_indices]

        # Majority vote
        most_common = Counter(k_nearest_labels).most_common(1)
        return most_common[0][0]

# Step 3: Accuracy Metric
def accuracy(y_true, y_pred):
    return np.sum(y_true == y_pred) / len(y_true)
```

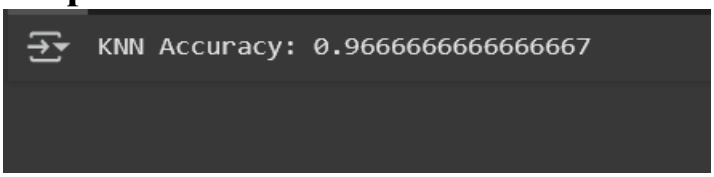
```
# Step 4: Load Dataset and Split
iris = load_iris()
X, y = iris.data, iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)

# Step 5: Train and Evaluate Model
model = KNN(k=3)
model.fit(X_train, y_train)
predictions = model.predict(X_test)

# Step 6: Output Accuracy
acc = accuracy(y_test, predictions)
print("KNN Accuracy:", acc)
```

3. Output:

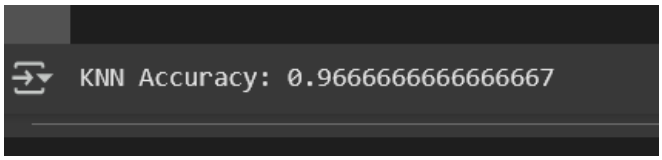


4. Evulation matrix :

```
def accuracy_score(y_true, y_pred):
    return np.sum(y_true == y_pred) / len(y_true)
def confusion_matrix(y_true, y_pred, labels=None):
    if labels is None:
        labels = np.unique(np.concatenate((y_true, y_pred)))
    n = len(labels)
    label_map = {label: i for i, label in enumerate(labels)}
    cm = np.zeros((n, n), dtype=int)
    for t, p in zip(y_true, y_pred):
        cm[label_map[t]][label_map[p]] += 1
    return cm, labels
def precision_recall_f1(cm):
    precisions, recalls, f1s = [], [], []
    for i in range(len(cm)):
        TP = cm[i, i]
        FP = np.sum(cm[:, i]) - TP
        FN = np.sum(cm[i, :]) - TP
        prec = TP / (TP + FP) if TP + FP != 0 else 0
        rec = TP / (TP + FN) if TP + FN != 0 else 0
```

```
    f1 = 2 * prec * rec / (prec + rec) if prec + rec != 0 else 0
    precisions.append(prec)
    recalls.append(rec)
    f1s.append(f1)
  return np.mean(precisions), np.mean(recalls), np.mean(f1s)
```

output:



5. Classify the Iris Flower and your own News datasets using the implemented KNN :

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
iris = load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = KNN(k=3)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

acc = accuracy_score(y_test, y_pred)
cm, labels = confusion_matrix(y_test, y_pred)
prec, rec, f1 = precision_recall_f1(cm)

print("Iris Dataset:")
print("Accuracy:", acc)
print("Confusion Matrix:\n", cm)
print("Precision:", prec, "Recall:", rec, "F1-Score:", f1)
```

output:

```
Iris Dataset:
Accuracy: 1.0
Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
Precision: 1.0 Recall: 1.0 F1-Score: 1.0
```

6. Find the optimal k value and split ratio for each dataset:

```
from sklearn.model_selection import train_test_split
k_values = [1, 3, 5, 7, 9]
splits = [0.2, 0.3, 0.4]
best_acc = 0

for split in splits:
    for k in k_values:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=split)
        model = KNN(k=k)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        acc = accuracy_score(y_test, y_pred)
        if acc > best_acc:
            best_acc = acc
            best_k = k
            best_split = split

      print(" Best k:", best_k, "Best split ratio:", 1-best_split, "train /", best_split, "test with
accuracy:", best_acc)
```

OUTPUT:

```
✅ Best k: 7 Best split ratio: 0.7 train / 0.3 test with accuracy: 1.0
```

7. ## Compare the custom KNN with scikit-learn's KNN:

```
from sklearn.neighbors import KNeighborsClassifier
sk_model = KNeighborsClassifier(n_neighbors=best_k)
sk_model.fit(X_train, y_train)
sk_pred = sk_model.predict(X_test)
my_model = KNN(k=best_k)
my_model.fit(X_train, y_train)
my_pred = my_model.predict(X_test)
print("Custom KNN Accuracy:", accuracy_score(y_test, my_pred))
print("Sklearn KNN Accuracy:", accuracy_score(y_test, sk_pred))
```

output: