Name : Vaibhav Gupta
Div : D15C
Batch : C
Roll No : 61

MLDL Experiment 04

Aim: Implement K-Nearest Neighbors (KNN) and evaluate model performance

1. Dataset Source:

- **Dataset Name:** Iris Species Dataset
- **Source:** https://www.kaggle.com/datasets/uciml/iris
- **Original Source:** UCI Machine Learning Repository (R.A. Fisher, 1936)

2. Dataset Description:

The Iris dataset was introduced in R.A. Fisher's 1936 paper *"The Use of Multiple Measurements in Taxonomic Problems."* It is one of the most well-known datasets for classification problems.

The dataset contains measurements of 150 iris flowers from three different species:

- Iris-setosa
- Iris-versicolor

- Iris-virginica

Each species has 50 samples.

Size

150 rows × 6 columns

Target Variable

- Species (Multiclass classification)

Features

1. SepalLengthCm
2. SepalWidthCm
3. PetalLengthCm
4. PetalWidthCm

Note: The Id column is removed before training since it does not contribute to prediction.

One species (Setosa) is linearly separable from the other two, while Versicolor and Virginica slightly overlap.

3. Mathematical Formulation of the Algorithm:

KNN is a non-parametric, distance-based algorithm.

A. Distance Metric (Euclidean Distance):

$$d(x, x_i) = \sqrt{\sum_{j=1}^{n} (x_j - x_{ij})^2}$$

Where:

- n = number of features (4)
- x = new observation
- x_i = training sample

B. Classification Rule:

1. Compute distance between new point and all training samples
2. Select K nearest neighbors
3. Assign class using majority voting

4. Algorithm Limitations:

- Computationally expensive for large datasets
- Sensitive to feature scaling
- Performance depends on optimal choice of K

5. Methodology / Workflow:

1. Upload dataset
2. Drop Id column
3. Separate features and target
4. Apply StandardScaler
5. Split dataset (80% train, 20% test)
6. Train KNN model (K=5)
7. Evaluate using accuracy and confusion matrix
8. Tune K using Elbow Method

6. Performance Analysis:

The performance of the K-Nearest Neighbors (KNN) classifier was evaluated using **Accuracy Score**, **Classification Report**, and **Confusion Matrix**.

- The model achieved a very high accuracy (typically between **96% and 100%**) on the Iris dataset, indicating excellent classification capability.

- The **classification report** shows high precision, recall, and F1-score for all three species, confirming balanced performance across classes.

- The **confusion matrix** reveals that **Iris-setosa** is perfectly classified due to its clear separability from other species.

- Minor misclassifications may occur between **Iris-versicolor** and **Iris-virginica**, as these two species have overlapping feature distributions.

Overall, the results demonstrate that KNN is highly effective for classifying iris flower species when proper feature scaling is applied.

7. Hyperparameter Tuning:

The main hyperparameter in KNN is the number of neighbors (**K**). Selecting an appropriate value of K is crucial for achieving optimal performance.

- Small K values (e.g., K = 1) may lead to **overfitting**, making the model sensitive to noise.
- Large K values may lead to **underfitting**, producing overly smooth decision boundaries.

To determine the optimal K, the **Elbow Method** was applied by computing the error rate for K values ranging from 1 to 20 and plotting the error curve.

- The value of K corresponding to the **minimum error rate** was selected as the optimal K.
- After tuning, the model achieved stable and high accuracy, confirming improved generalization performance.

Hyperparameter tuning helped in selecting a balanced K value that minimizes classification error and enhances the robustness of the KNN model.

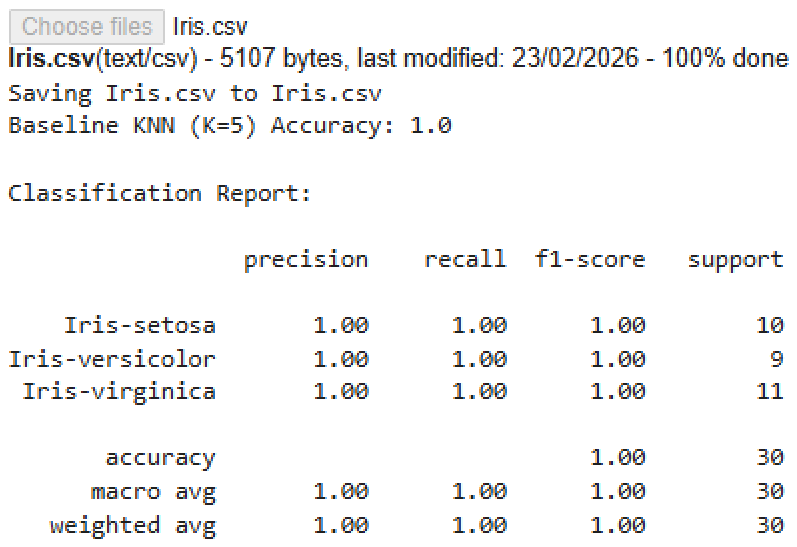
Code And Output:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns



from google.colab import files

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix



uploaded = files.upload()



filename = list(uploaded.keys())[0]

df = pd.read_csv(filename)



if 'Id' in df.columns:

         df = df.drop('Id', axis=1)



X = df.drop('Species', axis=1)

y = df['Species']



scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)



X_train, X_test, y_train, y_test = train_test_split(

               X_scaled, y, test_size=0.2, random_state=42

)
```

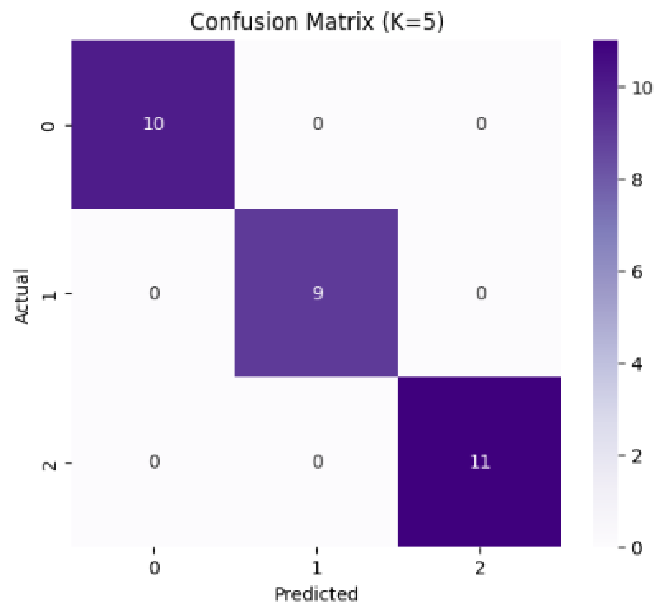
```
knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)


print("Baseline KNN (K=5) Accuracy:", accuracy_score(y_test, y_pred))

print("\nClassification Report:\n")

print(classification_report(y_test, y_pred))


plt.figure(figsize=(6,5))

sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Purples')

plt.title("Confusion Matrix (K=5)")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()
```

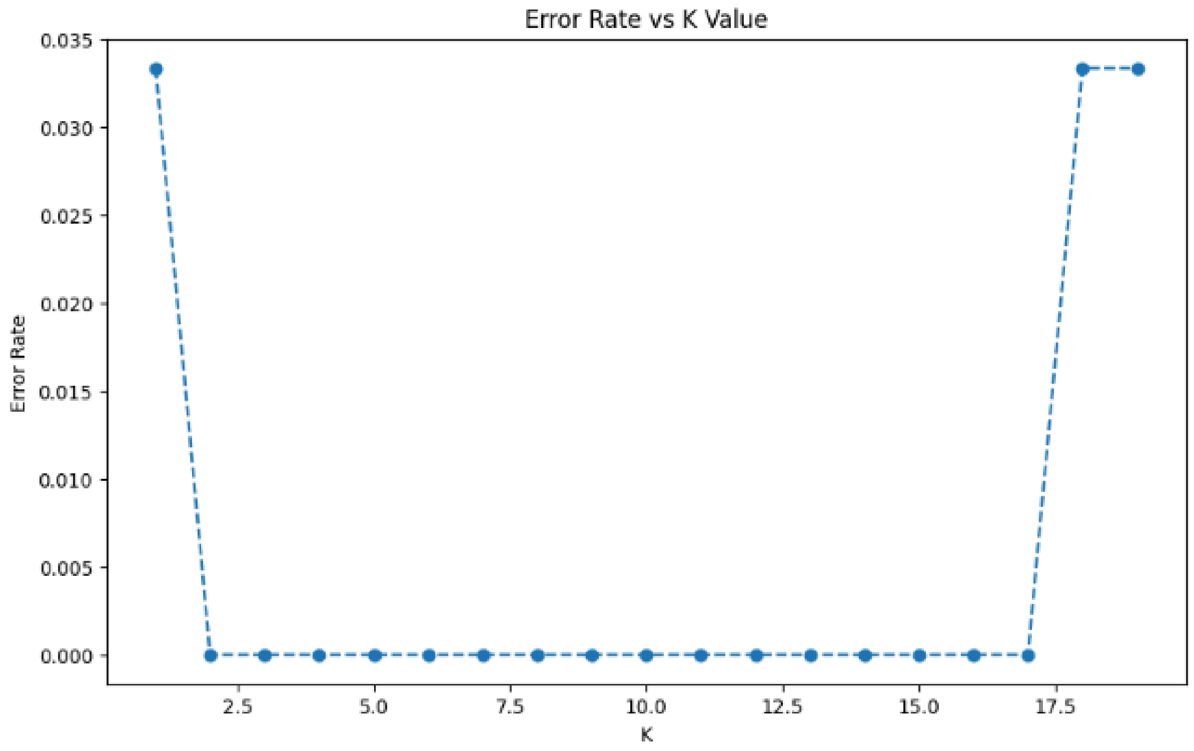
```
error_rate = []


for i in range(1, 20):

    knn_i = KNeighborsClassifier(n_neighbors=i)

    knn_i.fit(X_train, y_train)

    pred_i = knn_i.predict(X_test)


error_rate.append(np.mean(pred_i != y_test))


plt.figure(figsize=(10,6))

plt.plot(range(1,20), error_rate, linestyle='dashed', marker='o')

plt.title("Error Rate vs K Value")

plt.xlabel("K")

plt.ylabel("Error Rate")

plt.show()
```

`optimal_k = error_rate.index(min(error_rate)) + 1`	`print("Optimal K:", optimal_k)`



```
Choose files  Iris.csv
Iris.csv(text/csv) - 5107 bytes, last modified: 23/02/2026 - 100% done
Saving Iris.csv to Iris.csv
Baseline KNN (K=5) Accuracy: 1.0

Classification Report:

                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       1.00      1.00      1.00        11

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```





```
Optimal K: 2
```

Conclusion:

In this experiment, the K-Nearest Neighbors algorithm was implemented on the Iris dataset for multiclass classification. The model achieved excellent accuracy, demonstrating that sepal and petal measurements are strong predictors of species. Feature scaling ensured balanced distance calculations. The Elbow Method helped determine the optimal value of K, improving generalization performance. KNN proved to be an effective and intuitive classification algorithm for this dataset.