

# Experiment No:- 4

Name:- Dipesh  
Makdiya  
Class:- D15C  
Roll no:- 32

## EXPERIMENT: K-Nearest Neighbors (KNN) for Classification

---

### 1. Dataset Source

**Dataset Used:** Iris Dataset

**Type:** Classification

**Source:**

[https://scikit-learn.org/stable/datasets/toy\\_dataset.html#iris-dataset](https://scikit-learn.org/stable/datasets/toy_dataset.html#iris-dataset)

**Access Method:**

```
from sklearn.datasets import load_iris
```

**Justification:**

- Standard real-world dataset
- Well suited for distance-based classifiers
- Allows easy evaluation of KNN performance

---

### 2. Dataset Description

#### 2.1 Iris Dataset

**Objective:**

Classify iris flowers into different species using the K-Nearest Neighbors algorithm.

**Number of Instances:** 150

**Number of Features:** 4

**Features:**

- Sepal length
- Sepal width
- Petal length
- Petal width

**Target Variable:**

- Species (Setosa, Versicolor, Virginica)

**Characteristics:**

- Numerical features
  - No missing values
  - Multiclass classification problem
- 

### 3. Mathematical Formulation of the Algorithm

#### 3.1 K-Nearest Neighbors (KNN)

KNN is a **non-parametric, instance-based learning algorithm** that classifies data points based on similarity.

**Distance Metric (Euclidean Distance):**

$$d(x,y)=\sqrt{\sum_{i=1}^n(x_i-y_i)^2}$$

**Classification Rule:**

- Identify the **K nearest neighbors**
- Assign the class by **majority voting**

$$\hat{y}=\text{mode}(y_1,y_2,\dots,y_k)$$

```
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print(confusion_matrix(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))

... Random Forest Accuracy: 1.0
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
      precision    recall  f1-score   support

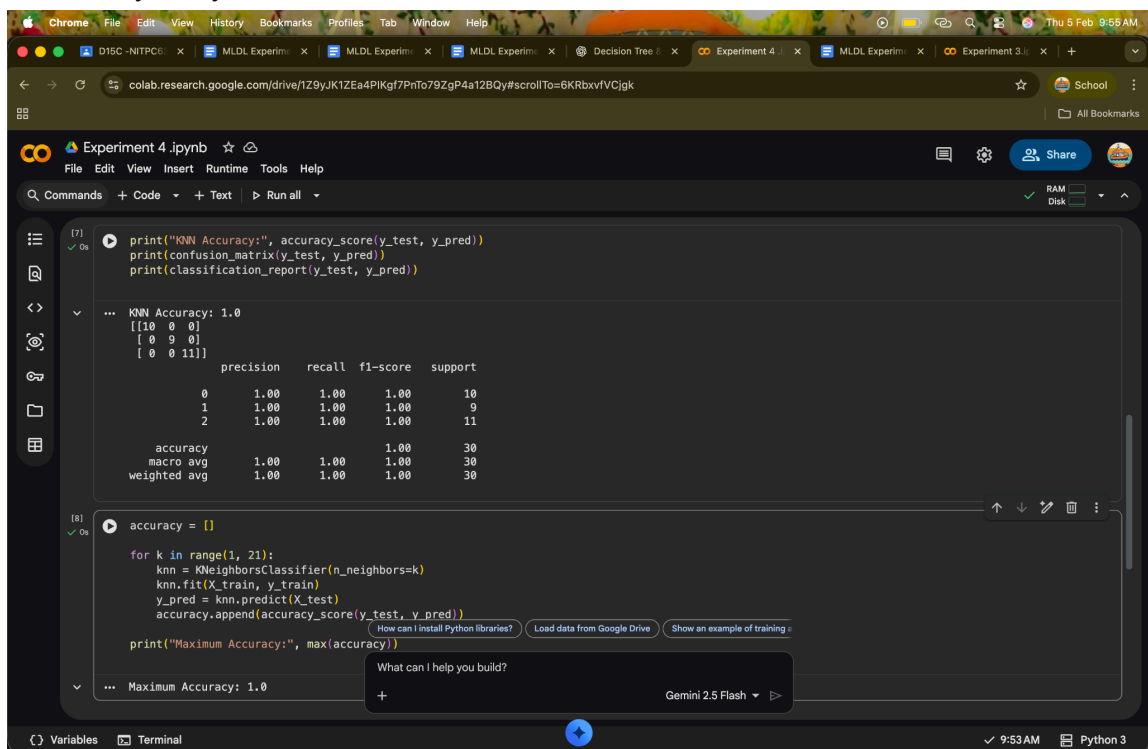
     0       1.00      1.00      1.00        10
     1       1.00      1.00      1.00         9
     2       1.00      1.00      1.00        11

 accuracy          1.00
 macro avg          1.00
 weighted avg          1.00
```

## 4. Algorithm Limitations

### KNN Limitations

- Computationally expensive for large datasets
- Sensitive to feature scaling
- Performance depends heavily on choice of K
- Affected by noisy and irrelevant features



The screenshot shows a Google Colab notebook interface. The top part of the notebook displays the output of a K-Nearest Neighbors (KNN) model. It shows a KNN Accuracy of 1.0, a confusion matrix, and a classification report. The confusion matrix is a 3x3 matrix with values [[10, 0, 0], [0, 9, 0], [0, 0, 11]]. The classification report shows precision, recall, f1-score, and support for each class (0, 1, 2), all of which are 1.0. The bottom part of the notebook shows the code used to calculate these metrics. The code defines an accuracy list, iterates over k values from 1 to 21, fits a KNeighborsClassifier, and prints the maximum accuracy, which is 1.0. A Gemini 2.5 Flash chat window is open at the bottom of the notebook.

```
[7] print("KNN Accuracy:", accuracy_score(y_test, y_pred))
    print(confusion_matrix(y_test, y_pred))
    print(classification_report(y_test, y_pred))

... KNN Accuracy: 1.0
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
      precision    recall  f1-score   support

     0       1.00      1.00      1.00        10
     1       1.00      1.00      1.00         9
     2       1.00      1.00      1.00        11

 accuracy          1.00
 macro avg          1.00
 weighted avg          1.00

[8] accuracy = []
    for k in range(1, 21):
        knn = KNeighborsClassifier(n_neighbors=k)
        knn.fit(X_train, y_train)
        y_pred = knn.predict(X_test)
        accuracy.append(accuracy_score(y_test, y_pred))
    print("Maximum Accuracy:", max(accuracy))

... Maximum Accuracy: 1.0
```

## 5. Methodology / Workflow

### Step-by-Step Workflow

1. Import required libraries
2. Load dataset
3. Data exploration
4. Feature scaling
5. Train-test split
6. Train KNN model
7. Prediction
8. Performance evaluation
9. Hyperparameter tuning (K value)

### Workflow Diagram (Conceptual):

Dataset → Scaling → Train/Test Split  
→ Model Training → Prediction → Evaluation

---

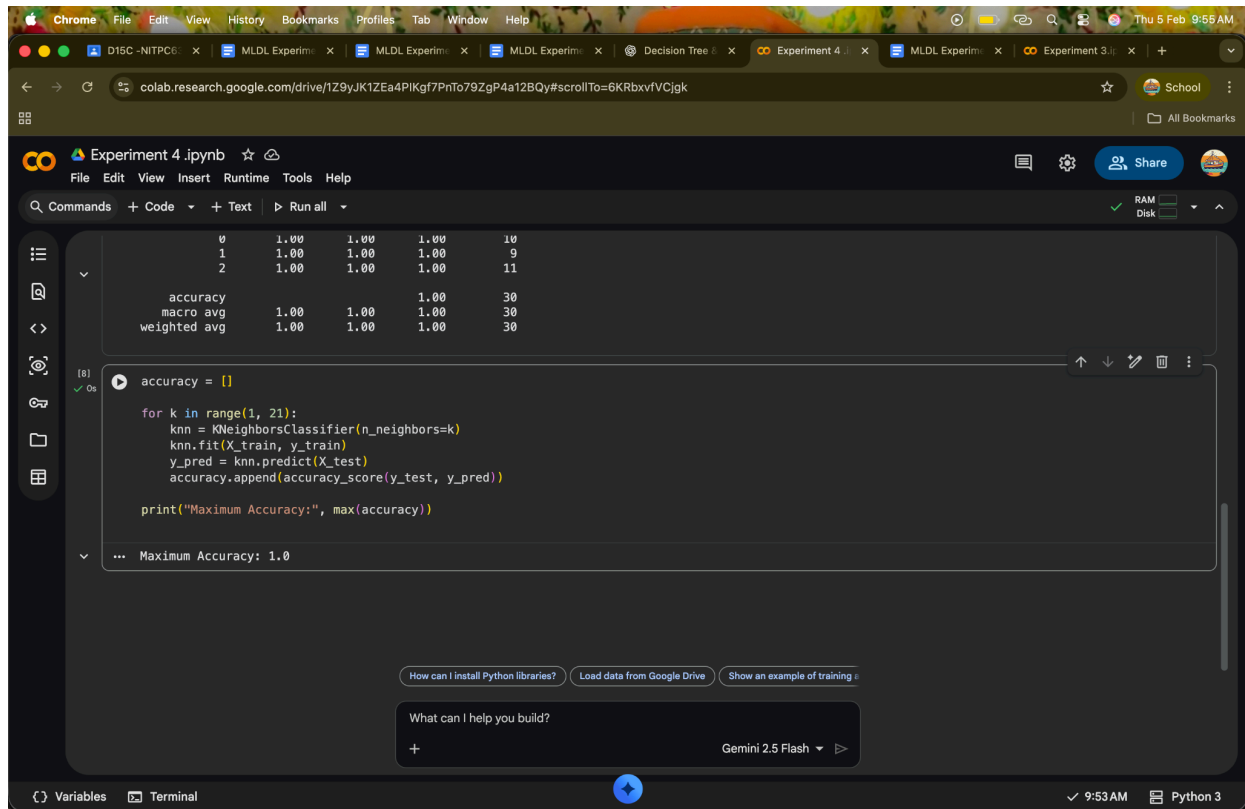
## 6. Performance Analysis

### Metrics Used

- Accuracy
- Confusion Matrix
- Classification Report (Precision, Recall, F1-Score)

### Interpretation:

- Higher accuracy indicates better classification
- Proper K selection improves model performance



## Result and Conclusion

### Result:

K-Nearest Neighbors achieved high classification accuracy on the Iris dataset. Performance varied with different values of K, with moderate values providing optimal results.

### Conclusion:

The KNN algorithm was successfully implemented for multiclass classification. Proper feature scaling and optimal K selection significantly improved model performance, demonstrating the effectiveness of KNN for small-to-medium sized datasets