# MLDL

## PRACTICAL 5

Name: Janhavi Mandhan
Class:D15C
Roll No:33
Batch B

---

# AIM:-

To implement K-Nearest Neighbors (KNN) algorithm for classification tasks and evaluate its performance using standard evaluation metrics.

---

# 1. Introduction

K-Nearest Neighbors (KNN) is a supervised machine learning algorithm used for classification and regression problems. It is a non-parametric, instance-based learning algorithm that classifies new data points based on similarity with training data.

KNN works on the principle that similar data points exist close to each other in feature space.

---

# 2. Dataset Description

Dataset Used: Breast Cancer Dataset / (Dataset used in notebook)

Source: UCI Machine Learning Repository / sklearn dataset

### Dataset Details:

- Number of Instances: 569
- Number of Features: 30
- Target Variable: Diagnosis

Target Encoding:

- 0 → Benign
- 1 → Malignant

### Dataset Characteristics:

- Binary classification problem
- Medical dataset
- Continuous numerical features
- No missing values

---

# 3. Mathematical Formulation of KNN

KNN calculates distance between test data and training data points.

### Euclidean Distance Formula:

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

Where:

- xxx = test data point
- yyy = training data point
- n = number of features

---

### Classification Rule:

1. Choose value of K
2. Calculate distance from all training points
3. Select K nearest neighbors
4. Assign class based on majority voting

y^=mode(y1,y2,...,yk)\hat{y} = mode(y_1, y_2, ..., y_k)y^=mode(y1,y2,...,yk)

---

# 4. Methodology / Workflow

### Steps Followed:

1. Import required libraries
2. Load dataset
3. Split dataset into training and testing sets
4. Apply feature scaling
5. Choose value of K
6. Train KNN model
7. Make predictions
8. Evaluate model performance

---

### Workflow Diagram (Textual)

Dataset
↓
Preprocessing
↓
Feature Scaling
↓
Train-Test Split
↓
KNN Training
↓
Prediction
↓
Evaluation

---

# 5. Performance Evaluation Metrics

## Accuracy

Accuracy=TP+TNTP+TN+FP+FNAccuracy = \frac{TP + TN}{TP + TN + FP + FN}Accuracy=TP+TN+FP+FNTP+TN

## Precision

Precision=TPTP+FPPrecision = \frac{TP}{TP + FP}Precision=TP+FPTP

## Recall

Recall=TPTP+FNRecall = \frac{TP}{TP + FN}Recall=TP+FNTP

## F1-Score

F1=2×Precision×RecallPrecision+RecallF1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}F1=2×Precision+RecallPrecision×Recall

## Confusion Matrix

| | Predicted Positive | Predicted Negative |
|---|---|---|
| Actual Positive | TP | FN |
| Actual Negative | FP | TN |

---

# 6. Hyperparameter Tuning

Important Parameter:

| Parameter | Description |
|---|---|
| n_neighbors (K) | Number of nearest neighbors |
| metric | Distance metric |
| weights | Uniform or distance |

### Impact:

- Small K → Overfitting
- Large K → Underfitting
- Proper K → Better generalization

---

# 7. Advantages of KNN

- Simple and easy to implement
- No training phase (lazy learning)

- Works well for small datasets
- Flexible decision boundaries

---

# 8. Limitations of KNN

- Computationally expensive for large datasets
- Sensitive to scaling
- Performance depends on value of K
- Memory intensive

---

# FINAL CONCLUSION

In this practical, the K-Nearest Neighbors (KNN) algorithm was successfully implemented for classification tasks. The model classifies data points based on similarity and distance from neighboring points. Proper feature scaling and selection of optimal K value significantly improve model performance.

Experimental results indicate that KNN provides reliable classification for structured datasets. However, it becomes computationally expensive for large datasets. Overall, KNN is an effective and simple algorithm for small to medium-sized classification problems.