Practical No.7

Title: Write a program to implement k-Nearest Neighbor algorithm to classify the iris data set.

Objective:

The objective of this practical is to implement the **k-Nearest Neighbors (k-NN) algorithm** to classify the **Iris dataset**. The program will classify test samples, display both correct and incorrect predictions, and evaluate the accuracy of the model.

Introduction to k-Nearest Neighbors (k-NN)

The **k-Nearest Neighbors (k-NN) algorithm** is a **non-parametric, supervised learning algorithm** used for classification and regression. It classifies a new data point based on the **majority class** of its nearest neighbors in the feature space.

Working Principle

1. Select a value for **k** (number of neighbors).
2. Compute the **Euclidean distance** between the test point and all training points.
3. Select the **k-nearest neighbors** with the shortest distances.
4. Assign the most common class label among the k neighbors to the test point.

Distance Calculation (Euclidean Distance Formula)

The most common distance metric used in k-NN is **Euclidean Distance**, given by:

d(A,B)=(x1−x2)2+(y1−y2)2+...+(n1−n2)2d(A, B) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + ... + (n_1 - n_2)^2}d(A,B)=(x1−x2)2+(y1−y2)2+...+(n1−n2)2

where (x1,x2,...,n)(x_1, x_2, ..., n)(x1,x2,...,n) are the feature values of two points A and B.

Dataset Description: Iris Dataset

The **Iris dataset** is a well-known dataset in machine learning and consists of **150 samples** of iris flowers, each belonging to one of three species:

- **Setosa**
- **Versicolor**
- **Virginica**

Each sample has four features:

1. **Sepal Length**
2. **Sepal Width**
3. **Petal Length**
4. **Petal Width**

The dataset is **evenly distributed**, with 50 samples per class.

Implementation Steps

Step 1: Load and Explore the Dataset

- Load the **Iris dataset** from a CSV file or directly from a machine learning library.
- Display dataset information, including feature names, data distribution, and class labels.
- Check for missing values and handle them if necessary.

Step 2: Data Preprocessing

- Shuffle the dataset to ensure random distribution.
- Normalize the feature values to bring them to the same scale.
- Split the dataset into **training data (80%)** and **testing data (20%)** for evaluation.

Step 3: Implement the k-NN Algorithm

- Choose an appropriate value of **k** (typically an odd number to avoid ties).
- Compute the Euclidean distance between test samples and all training samples.
- Select the **k-nearest neighbors** and assign the most common class label to the test sample.

Step 4: Make Predictions

- Predict the class labels for the test dataset.

- Compare predicted labels with actual labels.

Step 5: Display Correct and Incorrect Predictions

- Print both correctly classified and misclassified test samples.

Step 6: Compute Model Accuracy

The accuracy of the model is computed using the formula:

Accuracy=Correct PredictionsTotal Predictions×100Accuracy=\frac{Correct\ Predictions}{Total\Predictions}\times 100Accuracy=Total PredictionsCorrect Predictions×100

Additionally, a **confusion matrix** will be generated to analyze correct and incorrect classifications.

Expected Output

The program will output:

- **Predicted vs. Actual Labels** for test samples.
- **List of correctly classified samples** and **list of misclassified samples**.
- **Model accuracy score** in percentage.
- **Confusion matrix** summarizing correct and incorrect classifications.

Conclusion

This practical demonstrates the implementation of the **k-NN algorithm** for classifying the **Iris dataset**. The model was trained, tested, and evaluated for accuracy.