AAYUSH PATIL - 56

MLDL EXPERIMENT 3

AIM -

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

DATASET SOURCE -

https://www.kaggle.com/datasets/uciml/iris

DATASET DESCRIPTION -

The Iris dataset was used in R.A. Fisher's classic 1936 paper, The Use of Multiple Measurements in Taxonomic Problems, can also be found on the UCI Machine Learning Repository.

It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

The columns in this dataset are:

- Id
- SepalLengthCm
- SepalWidthCm
- PetalLengthCm
- PetalWidthCm
- Species

THEORY -

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a supervised machine learning algorithm used for classification and regression problems. It is a non-parametric and instance-based learning algorithm.

In classification tasks, KNN assigns a class label to a new data point based on the majority class among its K nearest neighbors.

KNN is called a **lazy learning algorithm** because it does not build a mathematical model during training. Instead, it stores the dataset and performs computation only at the time of prediction.

1. Objective of the Experiment

The objective of this experiment is:

- To classify Iris flowers into three categories:

    a. Iris-setosa
    b. Iris-versicolor
    c. Iris-virginica

- To use K = 3 for nearest neighbor classification.
- To evaluate how accurately the model performs.

2. Working Principle of KNN

KNN works on the principle of distance measurement.

Step-by-step working:

1. Choose the value of K.
2. Take a new input data point.
3. Calculate distance between the new point and all training points.
4. Select K nearest neighbors.
5. Perform majority voting.
6. Assign the class with the highest votes.

3. Distance Formula Used

KNN commonly uses Euclidean Distance:

$$\text{Distance} = \sqrt{[(SL_1 - SL_2)^2 + (SW_1 - SW_2)^2 + (PL_1 - PL_2)^2 + (PW_1 - PW_2)^2]}$$

Where:

- SL = Sepal Length
- SW = Sepal Width
- PL = Petal Length
- PW = Petal Width

The smaller the distance, the more similar the points.

4. Choice of K = 3

In this experiment:

- K = 3
- The algorithm selects the 3 closest flowers.
- The class appearing most among them is assigned to the new flower.

Reason for choosing small K:

- Prevents overfitting (K=1 is too sensitive).
- Maintains local pattern learning.

5. Evaluation of Model Performance

Model performance is evaluated using:

- Accuracy → Percentage of correct predictions
- Confusion Matrix → Correct vs incorrect classifications
- Classification Report → Precision, Recall, F1-score

High accuracy indicates that nearest neighbor classification is effective for this dataset.

6. Advantages of KNN

- Simple and easy to implement
- No training phase required
- Works well for small datasets
- Effective for multi-class classification

7. Disadvantages of KNN

- Slow for large datasets
- Sensitive to noise
- Requires feature scaling
- Performance depends on choice of K

8. Application in This Experiment

In this experiment:

- The entire dataset was stored.
- For a new flower input, distances were calculated.
- Three nearest neighbors were selected.
- Majority voting was performed
- Final classification was assigned.
- Graphs were plotted to visualize neighbors and distances.

CODE -

```
# ============================================
# KNN (K=3) WITH FULL CALCULATION DISPLAY
# ============================================

import pandas as pd
import numpy as np

# ============================================
# Step 1: Load Dataset
# ============================================

df = pd.read_csv('/content/Iris.csv')

# Remove Id column if present
if 'Id' in df.columns:
    df = df.drop('Id', axis=1)


# ============================================
# Step 2: Separate Features and Target
# ============================================

X = df.drop('Species', axis=1).values
y = df['Species'].values

print("\nDataset contains 3 categories:")
print(set(y))

# ============================================
# Step 3: KNN with K = 3
# ============================================

k = 3

# ============================================
# Step 4: Take User Input
# ============================================

print("\nEnter Flower Measurements:")

sepal_length = float(input("Sepal Length (4.3 - 7.9): "))
```

```
sepal_width = float(input("Sepal Width (2.0 - 4.4): "))
petal_length = float(input("Petal Length (1.0 - 6.9): "))
petal_width = float(input("Petal Width (0.1 - 2.5): "))

new_point = np.array([sepal_length, sepal_width, petal_length, petal_width])

# ==========================================
# Step 5: Calculate Distances
# ==========================================

distances = []

print("\n---- Distance Calculations ----")

for i in range(len(X)):
    distance = np.sqrt(np.sum((X[i] - new_point)**2))
    distances.append((distance, y[i]))
    print(f"Distance to sample {i} ({y[i]}): {distance:.4f}")

# ==========================================
# Step 6: Sort and Select K Nearest
# ==========================================

distances.sort(key=lambda x: x[0])

print("\n---- 3 Nearest Neighbors ----")
for i in range(k):
    print(f"Neighbor {i+1}: Class = {distances[i][1]}, Distance = {distances[i][0]:.4f}")

# ==========================================
# Step 7: Majority Voting
# ==========================================

nearest_classes = [distances[i][1] for i in range(k)]

final_prediction = max(set(nearest_classes), key=nearest_classes.count)

print("\n---- Final Classification ----")
print("Nearest Classes:", nearest_classes)
print("Final Predicted Category:", final_prediction)

import matplotlib.pyplot as plt
import numpy as np

# Use first 2 features for 2D visualization
plt.figure()

# Plot all dataset points
for species in np.unique(y):
    subset = df[df["Species"] == species]
    plt.scatter(subset.iloc[:,0], subset.iloc[:,1], label=species)

# Highlight new input
plt.scatter(new_point[0], new_point[1], marker='X', s=200)

# Highlight 3 nearest neighbors
for i in range(k):
    idx = distances.index(distances[i])
    plt.scatter(X[idx][0], X[idx][1], s=200)
```

```
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.title("KNN Visualization (K=3)")
plt.legend()
plt.show()

all_distances = [d[0] for d in distances]

plt.figure()
plt.hist(all_distances, bins=20)
plt.title("Distance Distribution from New Point")
plt.xlabel("Distance")
plt.ylabel("Frequency")
plt.show()

nearest_distances = [distances[i][0] for i in range(k)]
nearest_labels = [distances[i][1] for i in range(k)]

plt.figure()
plt.bar(range(1, k+1), nearest_distances)
plt.title("Distances of 3 Nearest Neighbors")
plt.xlabel("Neighbor Rank")
plt.ylabel("Distance")
plt.show()

from collections import Counter

vote_count = Counter([distances[i][1] for i in range(k)])

plt.figure()
plt.bar(vote_count.keys(), vote_count.values())
plt.title("Majority Voting Result (K=3)")
plt.xlabel("Species")
plt.ylabel("Votes")
plt.show()

from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

for species in np.unique(y):
    subset = df[df["Species"] == species]
    ax.scatter(subset.iloc[:,0],
               subset.iloc[:,1],
               subset.iloc[:,2],
               label=species)

# Plot new point
ax.scatter(new_point[0], new_point[1], new_point[2], s=200)

ax.set_xlabel("Sepal Length")
ax.set_ylabel("Sepal Width")
ax.set_zlabel("Petal Length")
ax.set_title("3D KNN Visualization")
plt.legend()
plt.show()
sorted_distances = sorted(all_distances)
plt.figure()
plt.plot(sorted_distances)
```

```
plt.axvline(x=k, linestyle='--')
plt.title("Sorted Distances (K Cutoff)")
plt.xlabel("Samples (Sorted by Distance)")
plt.ylabel("Distance")
plt.show()
```

OUTPUT -

```
Dataset contains 3 categories:
{'Iris-setosa', 'Iris-virginica', 'Iris-versicolor'}

Enter Flower Measurements:
Sepal Length (4.3 - 7.9): 4
Sepal Width (2.0 - 4.4): 3
Petal Length (1.0 - 6.9): 3.4
Petal Width (0.1 - 2.5): 1
```

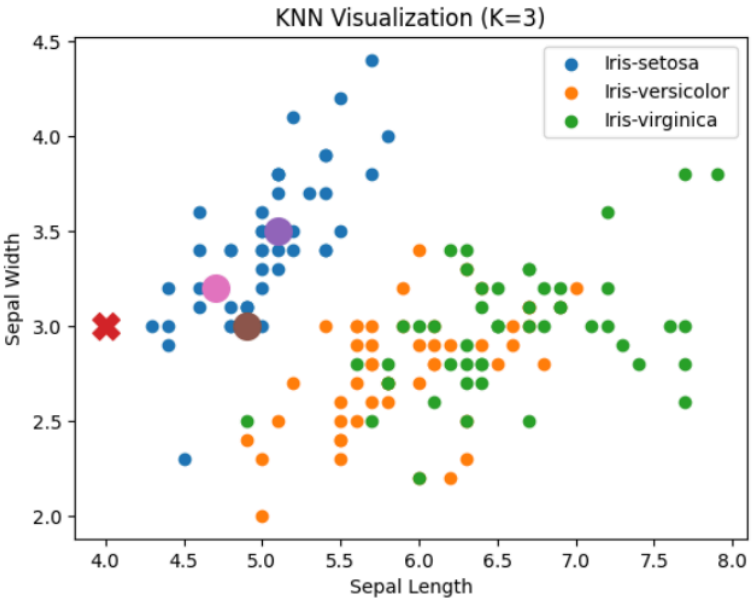
```
Distance to sample 0 (Iris-setosa): 2.4698
Distance to sample 1 (Iris-setosa): 2.3345
Distance to sample 2 (Iris-setosa): 2.3622
Distance to sample 3 (Iris-setosa): 2.1494
Distance to sample 4 (Iris-setosa): 2.4495
Distance to sample 5 (Iris-setosa): 2.4536
Distance to sample 6 (Iris-setosa): 2.2383
Distance to sample 7 (Iris-setosa): 2.3259
Distance to sample 8 (Iris-setosa): 2.1932
Distance to sample 9 (Iris-setosa): 2.2891
Distance to sample 10 (Iris-setosa): 2.5884
Distance to sample 11 (Iris-setosa): 2.1633
Distance to sample 12 (Iris-setosa): 2.3345
Distance to sample 13 (Iris-setosa): 2.4880
Distance to sample 14 (Iris-setosa): 3.1177
Distance to sample 15 (Iris-setosa): 2.9698
Distance to sample 16 (Iris-setosa): 2.7459
Distance to sample 17 (Iris-setosa): 2.4393
Distance to sample 18 (Iris-setosa): 2.6287
Distance to sample 19 (Iris-setosa): 2.4393
Distance to sample 20 (Iris-setosa): 2.3770
Distance to sample 21 (Iris-setosa): 2.3812
Distance to sample 22 (Iris-setosa): 2.6683
Distance to sample 23 (Iris-setosa): 2.1071
Distance to sample 24 (Iris-setosa): 1.9209
Distance to sample 25 (Iris-setosa): 2.2091
Distance to sample 26 (Iris-setosa): 2.1817
```

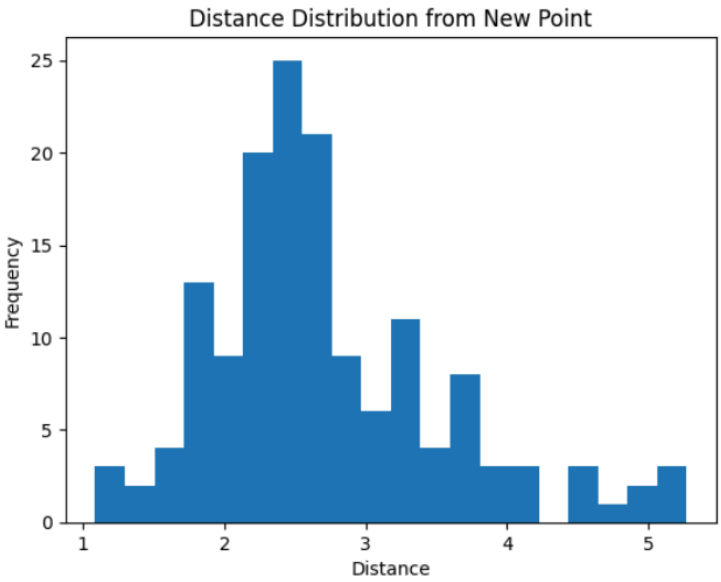
```
Distance to sample 51 (Iris-versicolor): 2.6944
Distance to sample 52 (Iris-versicolor): 3.3045
Distance to sample 53 (Iris-versicolor): 1.7861
Distance to sample 54 (Iris-versicolor): 2.8249
Distance to sample 55 (Iris-versicolor): 2.0567
Distance to sample 56 (Iris-versicolor): 2.7258
Distance to sample 57 (Iris-versicolor): 1.0863
Distance to sample 58 (Iris-versicolor): 2.8810
Distance to sample 59 (Iris-versicolor): 1.3928
Distance to sample 60 (Iris-versicolor): 1.4177
Distance to sample 61 (Iris-versicolor): 2.1213
Distance to sample 62 (Iris-versicolor): 2.2361
Distance to sample 63 (Iris-versicolor): 2.5040
Distance to sample 64 (Iris-versicolor): 1.6432
Distance to sample 65 (Iris-versicolor): 2.9086
Distance to sample 66 (Iris-versicolor): 2.0050
Distance to sample 67 (Iris-versicolor): 1.9545
Distance to sample 68 (Iris-versicolor): 2.6344
Distance to sample 69 (Iris-versicolor): 1.7521
Distance to sample 70 (Iris-versicolor): 2.5000
Distance to sample 71 (Iris-versicolor): 2.2136
Distance to sample 72 (Iris-versicolor): 2.8355
Distance to sample 73 (Iris-versicolor): 2.4860
Distance to sample 74 (Iris-versicolor): 2.5826
Distance to sample 75 (Iris-versicolor): 2.8142
Distance to sample 76 (Iris-versicolor): 3.1623
Distance to sample 77 (Iris-versicolor): 3.2156
Distance to sample 78 (Iris-versicolor): 2.3388
```

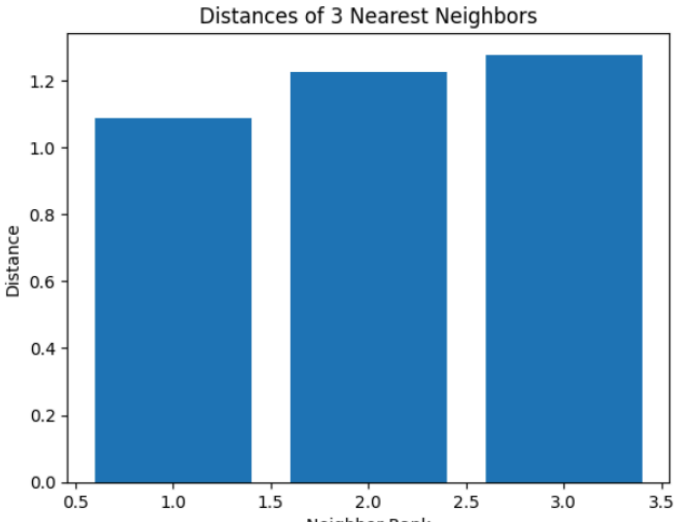
```
Distance to sample 91 (Iris-versicolor): 2.4515
Distance to sample 92 (Iris-versicolor): 1.9494
Distance to sample 93 (Iris-versicolor): 1.2247
Distance to sample 94 (Iris-versicolor): 1.8385
Distance to sample 95 (Iris-versicolor): 1.8894
Distance to sample 96 (Iris-versicolor): 1.9053
Distance to sample 97 (Iris-versicolor): 2.3979
Distance to sample 98 (Iris-versicolor): 1.2767
Distance to sample 99 (Iris-versicolor): 1.8735
Distance to sample 100 (Iris-virginica): 3.7934
Distance to sample 101 (Iris-virginica): 2.6514
Distance to sample 102 (Iris-virginica): 4.1316
Distance to sample 103 (Iris-virginica): 3.2833
Distance to sample 104 (Iris-virginica): 3.6674
Distance to sample 105 (Iris-virginica): 4.9406
Distance to sample 106 (Iris-virginica): 1.6613
Distance to sample 107 (Iris-virginica): 4.4665
Distance to sample 108 (Iris-virginica): 3.7336
Distance to sample 109 (Iris-virginica): 4.4878
Distance to sample 110 (Iris-virginica): 3.1906
Distance to sample 111 (Iris-virginica): 3.2047
Distance to sample 112 (Iris-virginica): 3.6688
Distance to sample 113 (Iris-virginica): 2.5884
Distance to sample 114 (Iris-virginica): 2.8513
Distance to sample 115 (Iris-virginica): 3.3317
Distance to sample 116 (Iris-virginica): 3.3615
Distance to sample 117 (Iris-virginica): 5.1633
Distance to sample 118 (Iris-virginica): 5.2716
```

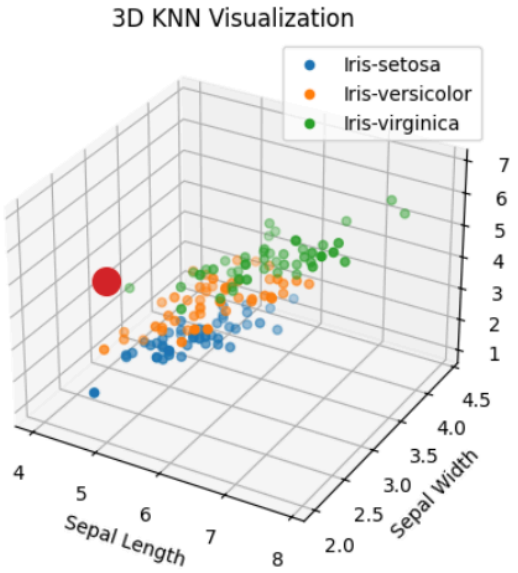
```
---- 3 Nearest Neighbors ----
Neighbor 1: Class = Iris-versicolor, Distance = 1.0863
Neighbor 2: Class = Iris-versicolor, Distance = 1.2247
Neighbor 3: Class = Iris-versicolor, Distance = 1.2767

---- Final Classification ----
Nearest Classes: ['Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor']
Final Predicted Category: Iris-versicolor
```









CONCLUSION -

The KNN algorithm was successfully implemented using K = 3 to classify Iris flowers into three categories.The model classified data based on similarity using Euclidean distance. The experiment demonstrated that KNN is an effective and simple classification algorithm for structured datasets like Iris.