# Racsaanth.B BL.EN.U4AIE23142

# ML Lab-3

## Python Imports

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.model\_selection import train\_test\_split  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.metrics import confusion\_matrix, classification\_report, accuracy\_score

## Dataset Loading

df = pd.read\_excel("/content/ML-3 dataset.xlsx")  # Load the dataset  
X = df.iloc[:, :-1].values # Extract feature columns  
y = df.iloc[:, -1].values # Extract class labels

## A1: Evaluate Intraclass Spread & Interclass Distance

class\_0 = X[y == 0]  
class\_1 = X[y == 1]  
  
centroid\_0 = np.mean(class\_0, axis=0)  
centroid\_1 = np.mean(class\_1, axis=0)  
  
spread\_0 = np.std(class\_0, axis=0)  
spread\_1 = np.std(class\_1, axis=0)  
  
distance\_between\_centroids = np.linalg.norm(centroid\_0 - centroid\_1)

## A2: Feature Density Analysis

feature\_index = 2 # Choose a feature  
plt.hist(X[:, feature\_index], bins=10, alpha=0.7, color='blue', edgecolor='black')  
plt.xlabel("Feature Value")  
plt.ylabel("Frequency")  
plt.title("Histogram of Feature")  
plt.show()

## A3: Minkowski Distance Calculation

minkowski\_distances = [np.linalg.norm(X[0] - X[1], ord=r) for r in range(1, 11)]  
plt.plot(range(1, 11), minkowski\_distances, marker='o')  
plt.xlabel("r value")  
plt.ylabel("Minkowski Distance")  
plt.title("Minkowski Distance for Different r values")  
plt.show()

## A4: Train-Test Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

## A5: Train k-NN Classifier (k=3)

neigh = KNeighborsClassifier(n\_neighbors=3)  
neigh.fit(X\_train, y\_train)

## A6: Evaluate k-NN Accuracy

accuracy = neigh.score(X\_test, y\_test)

## A7: Predict Using k-NN

y\_pred = neigh.predict(X\_test)

## A8: k-NN Accuracy with Varying k

k\_values = range(1, 12)  
accuracies = []  
for k in k\_values:  
 knn = KNeighborsClassifier(n\_neighbors=k)  
 knn.fit(X\_train, y\_train)  
 accuracies.append(knn.score(X\_test, y\_test))  
  
plt.plot(k\_values, accuracies, marker='o')  
plt.xlabel("k value")  
plt.ylabel("Accuracy")  
plt.title("k-NN Accuracy for Different k Values")  
plt.show()

## A9: Confusion Matrix & Performance Metrics

conf\_matrix = confusion\_matrix(y\_test, y\_pred)  
report = classification\_report(y\_test, y\_pred)  
print("Confusion Matrix:\n", conf\_matrix)  
print("Classification Report:\n", report)