**AI and ML Laboratory (CS2672)**

**Lab 9:** Classification with k-NN

### Objective:

Understand and implement k-Nearest Neighbours (k-NN).

### Task:

- Load the Iris dataset using Scikit-learn.

- Write a k-NN classifier from scratch.

- Evaluate the model's accuracy using train-test split.

### Introduction to k-Nearest Neighbours (k-NN):

k-NN is a simple, non-parametric, and lazy learning algorithm used for classification and regression. It classifies a data point based on how its neighbors are classified.

### Step-by-Step Implementation:

#### Step 1: Load the Iris Dataset

The Iris dataset is a standard dataset included in Scikit-learn, containing three classes of iris flowers (Setosa, Versicolor, and Virginica) with four features each. Use `sklearn.datasets.load\_iris()` to load the dataset.

```python  
from sklearn import datasets  
import numpy as np  
  
data = datasets.load\_iris()  
X = data.data  
y = data.target  
```

#### Step 2: Implement k-NN from Scratch

Compute the Euclidean distance between test points and training points. Identify the `k` nearest neighbors. Assign the most common class among the `k` neighbors to the test point.

```python  
from collections import Counter  
from scipy.spatial import distance  
  
def euclidean\_distance(x1, x2):  
 return np.sqrt(np.sum((x1 - x2) \*\* 2))  
  
class KNN:  
 def \_\_init\_\_(self, k=3):  
 self.k = k  
   
 def fit(self, X\_train, y\_train):  
 self.X\_train = X\_train  
 self.y\_train = y\_train  
   
 def predict(self, X\_test):  
 predictions = [self.\_predict(x) for x in X\_test]  
 return np.array(predictions)  
   
 def \_predict(self, x):  
 distances = [euclidean\_distance(x, x\_train) for x\_train in self.X\_train]  
 k\_indices = np.argsort(distances)[:self.k]  
 k\_nearest\_labels = [self.y\_train[i] for i in k\_indices]  
 most\_common = Counter(k\_nearest\_labels).most\_common(1)  
 return most\_common[0][0]  
```

#### Step 3: Evaluate the Model

Split the dataset into training and testing sets. Train the model and predict the test set labels. Compute the accuracy.

```python  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
knn = KNN(k=5)  
knn.fit(X\_train, y\_train)  
y\_pred = knn.predict(X\_test)  
  
accuracy = accuracy\_score(y\_test, y\_pred)  
print(f'Accuracy: {accuracy \* 100:.2f}%')  
```

**Assignment:**

1. What is the impact of choosing different values of k on the model's accuracy?
2. How does k-NN handle multi-class classification problems?
3. How does feature scaling affect the performance of k-NN?
4. What other distance metrics can be used instead of Euclidean distance, and how do they impact classification?
5. How does the choice of train-test split ratio affect the model’s performance?
6. What happens when k is too small or too large?