**[K-NN]**

**CSE 303: Machine Learning**

Submitted by

Name:Nuthalapati Charanteja

Roll No:AP22110010491

Section: CSE M

Lab Date: 17/09/24

Submission Date: 24/09/24

**A picture containing text

Description automatically generated**

**Department Computer Science and Engineering**

**School of Engineering and Sciences**

**SRM University–AP**

**Amaravati, Andhra Pradesh – 522 240, India**

1. **Question**

Implement K-NN Classifier for classification of any dataset of your choice.

a. Load an existing data set

b. Split the data set to train and test sets

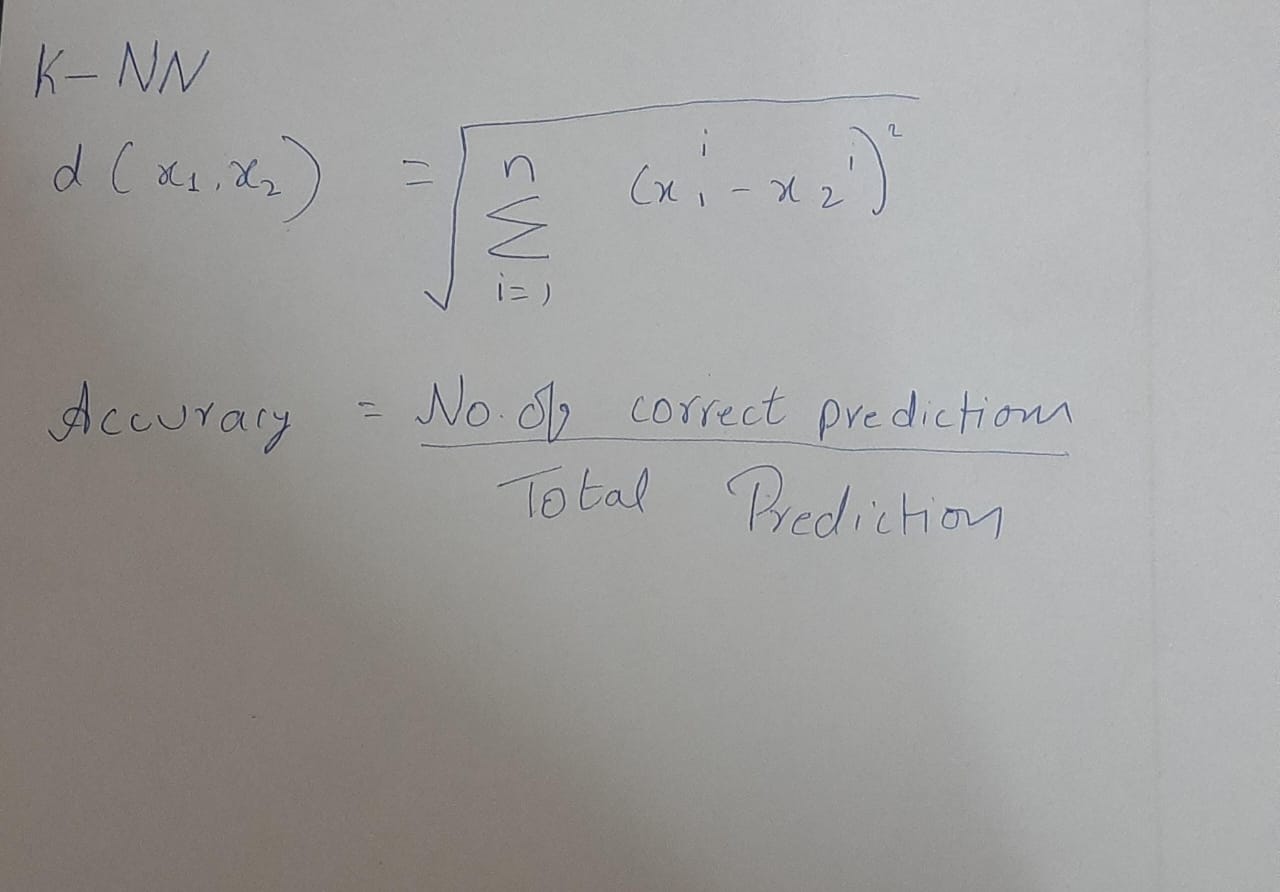
c. Test your model using test set. Find accuracy and confusion Matrix.

d. Examine the effect of the value of K on accuracy/performance. Plot the curve “k vs

accuracy” and find out the value of k for maximum accuracy for the test samples.

NOTE: Don’t use any library. Develop a generalised function to implement K-NN Classifier.

1. **Algorithm Description**



1. **Solution**

Dataset used:  
User\_Data.csv

import numpy as np

import matplotlib.pyplot as plt

from collections import Counter

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix, accuracy\_score

import pandas as pd

from sklearn.preprocessing import LabelEncoder

data = pd.read\_csv('User\_Data.csv')

print(data)

encoders = {}

for col in data.columns:

if data[col].dtype == 'object':

encoder = LabelEncoder()

data[col] = encoder.fit\_transform(data[col])

encoders[col] = encoder

X = data.iloc[:, :-1].values

y = data.iloc[:, -1].values

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

def euclidean\_dist(x, y):

return np.linalg.norm(x - y)

def knn\_classifier(train\_data, train\_labels, test\_data, k):

results = []

for test\_point in test\_data:

dist\_list = [euclidean\_dist(test\_point, train\_point) for train\_point in train\_data]

k\_closest = np.argsort(dist\_list)[:k]

k\_labels = [train\_labels[i] for i in k\_closest]

majority\_label = Counter(k\_labels).most\_common(1)[0][0]

results.append(majority\_label)

return results

def evaluate\_knn(k\_values):

accuracies = []

for k in k\_values:

predictions = knn\_classifier(X\_train, y\_train, X\_test, k)

score = accuracy\_score(y\_test, predictions)

accuracies.append(score)

print(f"For K={k}, Accuracy={score \* 100:.2f}%")

return accuracies

k\_range = range(1, 21)

accuracy\_scores = evaluate\_knn(k\_range)

plt.figure(figsize=(10, 6))

plt.plot(k\_range, accuracy\_scores, marker='o')

plt.title("K vs Accuracy")

plt.xlabel("K")

plt.ylabel("Accuracy")

plt.xticks(k\_range)

plt.grid(True)

plt.show()

best\_k = k\_range[np.argmax(accuracy\_scores)]

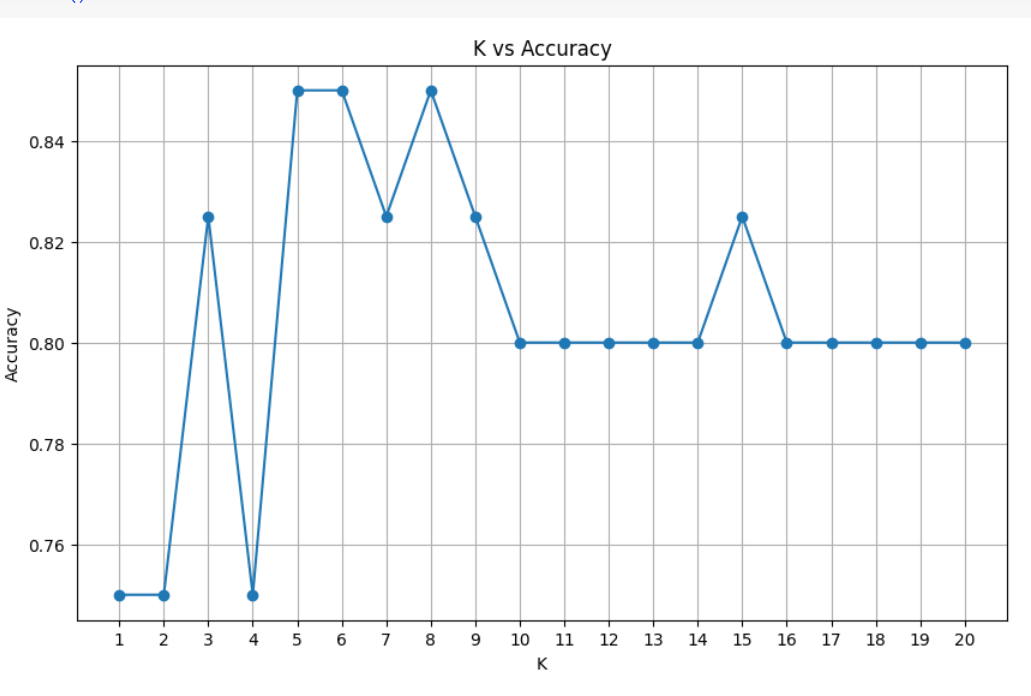
optimal\_predictions = knn\_classifier(X\_train, y\_train, X\_test, best\_k)

conf\_matrix = confusion\_matrix(y\_test, optimal\_predictions)

optimal\_accuracy = accuracy\_score(y\_test, optimal\_predictions)

conf\_matrix, optimal\_accuracy, best\_k

OUTPUT:



(array([[26, 1],

[ 5, 8]]),

0.85,

5)

1. **Code Repository:**

https://github.com/sivanilahari/ML-Assignment/blob/main/KNN.ipynb