Pratical- 8 :

Aim:- Implement the K-NN algorithm for classification or regression

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import roc\_auc\_score

plt.style.use('ggplot')

# Load the data

df = pd.read\_csv("D:/Ai practical/diabetes.csv")

print(df.head(),"\n")

print(df.shape,"\n")

print(df.dtypes,"\n")

# Prepare the data

x = df.drop('Outcome', axis=1).values

y = df['Outcome'].values

# Split the data into training and testing sets

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.4, random\_state=42)

# KNN model training and evaluation

neighbors = np.arange(1, 9)

train\_accuracy = np.empty(len(neighbors))

test\_accuracy = np.empty(len(neighbors))

for i, k in enumerate(neighbors):

knn = KNeighborsClassifier(n\_neighbors=k)

knn.fit(x\_train, y\_train)

train\_accuracy[i] = knn.score(x\_train, y\_train)

test\_accuracy[i] = knn.score(x\_test, y\_test)

# Plotting the accuracy for different values of k

plt.title('k-NN Varying number of neighbors')

plt.plot(neighbors, test\_accuracy, label='Testing Accuracy')

plt.plot(neighbors, train\_accuracy, label='Training accuracy')

plt.legend()

plt.xlabel('Number of neighbors')

plt.ylabel('Accuracy')

plt.show()

# Predict probabilities for ROC AUC score calculation

knn = KNeighborsClassifier(n\_neighbors=5)

knn.fit(x\_train, y\_train)

y\_pred\_prob = knn.predict\_proba(x\_test)[:, 1]

# Grid Search for optimal hyperparameters

param\_grid = {'n\_neighbors': np.arange(1, 50)}

knn = KNeighborsClassifier()

knn\_cv = GridSearchCV(knn, param\_grid, cv=5)

print("\n\n",knn\_cv.fit(x, y))

print("\n\n",knn\_cv.best\_score\_)

print("\n\n",knn\_cv.best\_params\_)

output:-



