**Ex 3: K – Nearest Neighbour Algorithm**

**Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions.**

**Aim:**

To implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions.

**Algorithm:**

**1. Import Libraries**

Import numpy, pandas, and relevant modules from sklearn**.**

**2. Load and Inspect Dataset**

Read the Iris dataset from a CSV file and view the first few rows to understand its structure.

**3. Prepare Data**

Separate features (X) and target (y) by dropping the species column from features.

**4. Split Dataset**

Divide data into training and testing sets using train\_test\_split().

**5. Initialize and Train Model**

Create a KNN classifier with k=3 and train it using the training data.

**6. Make Predictions**

Use the trained model to predict target labels for the test data.

**7. Evaluate Model**

Compare predictions with actual labels, print correct/wrong predictions, and compute accuracy.

**Program:**

# Importing necessary libraries

import numpy as np

import pandas as pd

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

# Step 1: Manually load the dataset

# Assuming you have the Iris dataset as a CSV file (Iris.csv)

# Replace the path with the location of your file

file\_path = 'C:/Users/HDC0422236/Desktop/iris dataset/Iris.csv'

data = pd.read\_csv(file\_path)

# Step 2: Check the first few rows to understand the structure of the data

print(data.head())

# Step 3: Separate features and target

X = data.drop('species', axis=1) # Drop the target column 'Species'

y = data['species'] # Target column

# Step 4: Split the dataset into training and testing sets

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

# Step 5: Initialize the KNN classifier

k = 3 # You can change k to experiment with different values

knn = KNeighborsClassifier(n\_neighbors=k)

# Step 6: Fit the model to the training data

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

# Step 7: Make predictions on the test set

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

# Step 8: Evaluate the predictions

correct\_predictions = np.where(y\_pred == y\_test)[0]

wrong\_predictions = np.where(y\_pred != y\_test)[0]

# Print out correct and wrong predictions

print(f"Correct predictions (indices): {correct\_predictions}")

print(f"Wrong predictions (indices): {wrong\_predictions}")

# Print the accuracy score

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy \* 100:.2f}%")

# Optionally, display the predicted vs true labels for clarity

print("\nPredictions vs Actuals:")

for i in range(len(y\_test)):

print(f"Predicted: {y\_pred[i]} | Actual: {y\_test.iloc[i]}")

**OUTPUT:**

**Correct predictions (indices):** [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 40 41 43 44]

**Wrong predictions (indices):** [39 42]

**Accuracy: 95.56%**

**Predictions vs Actuals:**

Predicted: setosa | Actual: setosa

Predicted: versicolor | Actual: versicolor

Predicted: setosa | Actual: setosa

Predicted: virginica | Actual: virginica

Predicted: setosa | Actual: setosa

Predicted: versicolor | Actual: versicolor

Predicted: virginica | Actual: virginica

Predicted: virginica | Actual: virginica

Predicted: versicolor | Actual: versicolor

Predicted: versicolor | Actual: versicolor

Predicted: setosa | Actual: setosa

Predicted: setosa | Actual: setosa

Predicted: setosa | Actual: setosa

Predicted: virginica | Actual: virginica

Predicted: versicolor | Actual: versicolor

Predicted: setosa | Actual: setosa

Predicted: versicolor | Actual: versicolor

Predicted: virginica | Actual: virginica

Predicted: virginica | Actual: virginica

Predicted: virginica | Actual: virginica

Predicted: versicolor | Actual: versicolor

Predicted: setosa | Actual: setosa

Predicted: virginica | Actual: virginica

Predicted: setosa | Actual: setosa

Predicted: virginica | Actual: virginica

Predicted: virginica | Actual: virginica

Predicted: versicolor | Actual: versicolor

Predicted: setosa | Actual: setosa

Predicted: virginica | Actual: virginica

Predicted: virginica | Actual: virginica

Predicted: setosa | Actual: setosa

Predicted: virginica | Actual: virginica

Predicted: versicolor | Actual: versicolor

Predicted: setosa | Actual: setosa

Predicted: virginica | Actual: virginica

Predicted: setosa | Actual: setosa

Predicted: virginica | Actual: virginica

Predicted: setosa | Actual: setosa

Predicted: setosa | Actual: setosa

Predicted: setosa | Actual: versicolor

Predicted: virginica | Actual: virginica

Predicted: versicolor | Actual: versicolor

Predicted: virginica | Actual: versicolor

Predicted: virginica | Actual: virginica

Predicted: versicolor | Actual: versicolor