

Problem Statement: Implement the K-Nearest Neighbor Classification algorithm.

Introduction: K-Nearest Neighbor algorithm falls in the category of numeric pattern classification technique. Using this algorithm, we can classify an object as a member of a class by determining the distances of that object from all the given classes. The class with the lowest distance from the object will inherit the object. The K here is an integer which defines the number of nearest neighbor to be taken to perform the classification. For K=1, it will simply be the Nearest Neighbor classification where we will take the closest value and assigned the new value to that class. For K>1, we will take that number (suppose k=3, so we will take 3 neighbors) of closest neighbors and then assigned the new object to the class with most frequent labels.

There are various methods to calculate the distance to the neighbors. Among them, Euclidean distance measure technique is the most practiced one.

Euclidean Distance measure: Suppose, in a rectangular coordinate system we have two vectors (X and Y) and we want to find the distance between them. According to the Euclidean distance measure technique the formula for calculating the distance is:

$$d(X,Y)_{euc} = \sqrt{\sum_{k=1}^n (X_k - Y_k)^2}$$

Where, n defines the dimensionality of the vector.

Now, let's take a two dimensional example. Then the formula will be:

$$d(X,Y)_{euc} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

In this report, the K-Nearest Neighbor is calculated with the value of K = 1 and 3 adopting the Euclidean Distance measure technique for a given dataset.

Dataset Description: The dataset used for K-Nearest Neighbor classification is the "Social Network Ads" dataset from Kaggle. It consists 400 entries, each with 5 columns. The dataset represents the details of 400 persons about their User ID, Gender, Age, EstimatedSalary, and whether they have purchased products from social network ad or not. The first 5 entries of the dataset are demonstrated below for better understanding.

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

Data Preprocessing: The dataset was processed before utilizing for classification. First two columns of the dataset were dropped due to less significance in distance calculating. Original shape of the dataset was (400, 5). After dropping two columns, it became (400, 3). After that, the entire dataset was split into train and test dataset with a ratio of 3:1. The new shape for train and test dataset was (300, 3) and (100, 3) respectively. Later on, the final processing was performed on test dataset where the last column containing information about purchasing was dropped leaving the test dataset with a shape of (100, 2). Now, the dataset is ready for K- Nearest Neighbor classification.

K-Nearest Neighbor Classification (K=1): Here, the classification is discussed with the value of K=1. This is also known as “Nearest Neighbor Classification” as only one neighbor is used to predict the class.

Methodology: To predict the class of each value in test dataset, the distances of them from each value of train dataset was calculated using the Euclidean Distance measuring technique. First, a minimum distance value and a selected node value was initialized. Later, while calculating the distances, if a minimum distance was found compared to present value, the minimum distance was updated with the new value and the selected node was updated with that class label. Finally, checking through all the elements, each data in test dataset was predicted as one of the two classes.

Code:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

plt.rcParams['figure.figsize'] = (20.0, 10.0)

%matplotlib inline


df = pd.read_csv('Social_Network_Ads.csv')

df.head(5)


Y = df.iloc[:,4].values

Y = Y[300:]

X = df.iloc[:,2:5].values

from sklearn.model_selection import train_test_split

X_train,X_test = train_test_split(X,test_size=0.25, random_state=0)

print('Train data shape'+str(X_train.shape))


X_test = np.delete(X_test, -1, 1)
```

```

print('Test data shape'+str(X_test.shape))

import math

accurate = 0

inaccurate = 0

k = 0

# Calculating Euclidean distance

print('Euclidean Distance:')

for i in X_test:

    minimum_distance = 9999999999999999

    selected_node = 0

    for j in X_train:

        distance = math.sqrt((j[0]-i[0])*(j[0]-i[0])+(j[1]-i[1])*(j[1]-i[1]))

        if distance < minimum_distance:

            minimum_distance = distance

            selected_node = j[2]

    if selected_node == Y[k]:

        accurate += 1

    else:

        inaccurate += 1

    k += 1

print('Input: ' + str(i) + ', Predicted class: ' + str(selected_node))

print('Accuracy: ' + str(accurate/(accurate+inaccurate)))

```

Output sample:

```

Euclidean Distance:
Input: [ 30 87000], Predicted class: 0
Input: [ 38 50000], Predicted class: 0
Input: [ 35 75000], Predicted class: 0
Input: [ 30 79000], Predicted class: 0
Input: [ 35 50000], Predicted class: 0
Input: [ 27 20000], Predicted class: 1
Input: [ 31 15000], Predicted class: 0
Input: [ 36 144000], Predicted class: 1
Input: [ 18 68000], Predicted class: 0
Input: [ 47 43000], Predicted class: 0

```

Performance Analysis & Result: Prediction of first 10 sample is demonstrated. An accuracy of 57% was achieved through this classification technique. The main drawback of this method is that it can't handle the outlier data. Because of this, a lot false prediction was performed. This drawback can be solved using K-Nearest Neighbor algorithm with a value of $K > 1$.

K-Nearest Neighbor Classification (K=3): Here, the classification is discussed with the value of $K=3$. This is known as "K-Nearest Neighbor Classification" as K number of neighbors are used to predict the class.

Methodology: In this classification the same dataset with same data distribution was used. The main difference was that, here the value of $K=3$, that is 3 nearest neighbors were identified and then calculating the occurrence of most frequent class, prediction was performed. Nearest Neighbors were identified by first calculating the minimum distance from all train data, then sorting was performed in an ascending order of distances. Then the first 3 entries were selected as the nearest neighbors.

Code:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

plt.rcParams['figure.figsize'] = (20.0, 10.0)

%matplotlib inline

df = pd.read_csv('Social_Network_Ads.csv')

df.head(5)

Y = df.iloc[:,4].values

Y = Y[300:]

X = df.iloc[:,2:5].values

from sklearn.model_selection import train_test_split

X_train,X_test = train_test_split(X,test_size=0.25, random_state=0)

print('Train data shape'+str(X_train.shape))

X_test = np.delete(X_test, -1, 1)

print('Test data shape'+str(X_test.shape))
```

```

# Example of making predictions

from math import sqrt

# calculate the Euclidean distance between two vectors
def euclidean_distance(row1, row2):
    distance = 0.0
    for i in range(len(row1)-1):
        distance += (row1[i] - row2[i])**2
    return sqrt(distance)

# Locate the most similar neighbors
def get_neighbors(train, test_row, num_neighbors):
    distances = list()
    for train_row in train:
        dist = euclidean_distance(test_row, train_row)
        distances.append((train_row, dist))
    distances.sort(key=lambda tup: tup[1])
    neighbors = list()
    for i in range(num_neighbors):
        neighbors.append(distances[i][0])
    return neighbors

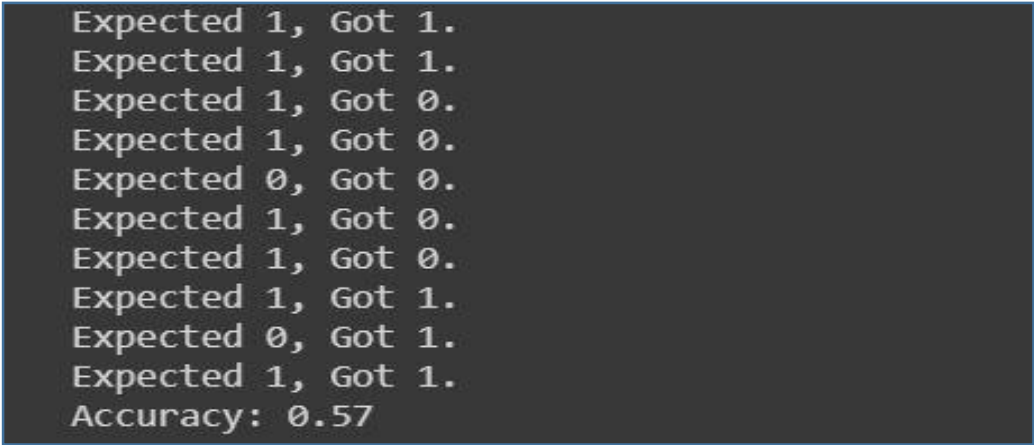
# Make a classification prediction with neighbors
def predict_classification(train, test_row, num_neighbors):
    neighbors = get_neighbors(train, test_row, num_neighbors)
    output_values = [row[-1] for row in neighbors]
    prediction = max(set(output_values), key=output_values.count)
    return prediction

accurate = 0
inaccurate = 0
i = 0
for i in range(100):
    prediction = predict_classification(X_train, X_test[i], 3)
    print('Expected %d, Got %d.' % (Y[i], prediction))

```

```
if prediction == Y[i]:  
    accurate += 1  
else:  
    inaccurate += 1  
i += 1  
print('Accuracy: ' + str(accurate/(accurate+inaccurate)))
```

Output Sample:



```
Expected 1, Got 1.  
Expected 1, Got 1.  
Expected 1, Got 0.  
Expected 1, Got 0.  
Expected 0, Got 0.  
Expected 1, Got 0.  
Expected 1, Got 0.  
Expected 1, Got 1.  
Expected 0, Got 1.  
Expected 1, Got 1.  
Accuracy: 0.57
```

Performance Analysis & Result: Prediction of last 10 sample is demonstrated. An accuracy of 57% was achieved through this classification technique.

Conclusion: In comparison between Nearest Neighbor and K-Nearest Neighbor classification, K-Nearest Neighbor classification is better but, in this case with the given dataset, same accuracy was achieved.