

Batch: HO-ML 1**Experiment Number: 04****Roll Number: 16010422234****Name: Chandana Galgali**

Aim of the Experiment: Implementation of KNN (K Nearest Neighbor) algorithm for classification

Program/ Steps:

1. Apply KNN to the dataset shown below and tabulate the results Calculate efficiency of KNN for this dataset.
2. Write the program for the same.

K - Nearest Neighbor Solved Example					
Sr	Type	Phy	Chem	Math	Class
1	Training	100	100	100	Admitted
2	Training	100	98	100	Admitted
3	Training	99	95	99	Admitted
4	Training	98	96	99	Admitted
5	Training	97	99	94	Admitted
6	Training	96	96	98	Admitted
7	Training	97	95	95	Admitted
8	Training	97	95	94	Admitted
9	Training	95	94	95	Admitted
10	Training	96	91	97	Rejected
11	Training	91	97	96	Rejected
12	Training	95	94	94	Admitted
13	Training	95	94	94	Rejected
14	Training	94	91	85	Rejected
15	Training	92	85	83	Rejected
16	Test	95	97	97	
17	Test	95	91	95	
18	Test	97	95	87	
19	Test	95	81	89	
20	Test	80	80	81	

Output/Result:

```

import numpy as np
import pandas as pd
from collections import Counter
from math import sqrt

# Training dataset
data = {
    'Type': ['Training']*15 + ['Test']*5,
    'Phy': [100, 100, 99, 98, 97, 96, 97, 97, 95, 96, 91, 95, 95, 94, 92,
95, 95, 97, 95, 80],
    'Chem': [100, 98, 95, 96, 99, 96, 95, 95, 94, 91, 97, 94, 94, 91, 85,
97, 91, 95, 81, 80],
    'Math': [100, 100, 99, 99, 94, 98, 95, 94, 95, 97, 96, 94, 94, 85, 83,
97, 95, 87, 89, 81],
    'Class': ['Admitted', 'Admitted', 'Admitted', 'Admitted', 'Admitted',
'Admitted', 'Admitted', 'Admitted', 'Admitted', 'Rejected',
'Rejected', 'Admitted', 'Rejected', 'Rejected', 'Rejected',
None, None, None, None, None]
}

# Convert data to a DataFrame
df = pd.DataFrame(data)

# Function to calculate Euclidean distance
def euclidean_distance(point1, point2):
    return sqrt(sum((x - y) ** 2 for x, y in zip(point1, point2)))

# KNN function
def knn(train_data, test_point, k):
    # Calculate distances between test_point and all training points
    distances = []
    for index, row in train_data.iterrows():
        distance = euclidean_distance([row['Phy'], row['Chem'],
row['Math']], test_point)
        distances.append((distance, row['Class']))

    # Sort by distance
    distances.sort(key=lambda x: x[0])

```

```

# Get the nearest k neighbors
k_nearest = distances[:k]

# Find the most common class among the neighbors
classes = [neighbor[1] for neighbor in k_nearest]
most_common_class = Counter(classes).most_common(1)[0][0]


return most_common_class

# Apply KNN for each test point
k = 3 # Choosing k=3
for i, row in df[df['Type'] == 'Test'].iterrows():
    test_point = [row['Phy'], row['Chem'], row['Math']]
    predicted_class = knn(df[df['Type'] == 'Training'], test_point, k)
    df.at[i, 'Class'] = predicted_class

# Output the updated DataFrame with predictions
print(df)

# Calculate efficiency
correct_predictions = sum(df[df['Type'] == 'Test']['Class'] ==
df[df['Type'] == 'Test']['Class'])
total_predictions = len(df[df['Type'] == 'Test'])
efficiency = (correct_predictions / total_predictions) * 100
print(f"Efficiency of KNN: {efficiency}%")

```



	Type	Phy	Chem	Math	Class
0	Training	100	100	100	Admitted
1	Training	100	98	100	Admitted
2	Training	99	95	99	Admitted
3	Training	98	96	99	Admitted
4	Training	97	99	94	Admitted
5	Training	96	96	98	Admitted
6	Training	97	95	95	Admitted
7	Training	97	95	94	Admitted
8	Training	95	94	95	Admitted
9	Training	96	91	97	Rejected
10	Training	91	97	96	Rejected
11	Training	95	94	94	Admitted
12	Training	95	94	94	Rejected
13	Training	94	91	85	Rejected
14	Training	92	85	83	Rejected
15	Test	95	97	97	Admitted
16	Test	95	91	95	Admitted
17	Test	97	95	87	Admitted
18	Test	95	81	89	Rejected
19	Test	80	80	81	Rejected

Efficiency of KNN: 100.0%

Post Lab Question-Answers:

1. What are the advantages and disadvantages of KNN?

Advantages:

- Simplicity: KNN is simple and easy to understand. There is no need for a complex training phase, and it is easy to implement.
- No Training Required: KNN is a lazy learning algorithm, meaning that it does not require an explicit training step. The model stores the training instances and classifies new instances by comparing them to the stored ones.
- Flexible to Feature Spaces: It can be applied to both classification and regression problems. KNN works well with multi-class classification.

- Adaptability: KNN is non-parametric, meaning it makes no assumptions about the data distribution, which makes it adaptable to various types of datasets.
- Interpretable: The decision process of KNN (majority voting from neighbors) is easy to interpret.

Disadvantages:

- Computationally Expensive: Since KNN stores all the training data, it can be computationally expensive when the dataset is large, especially during the prediction phase (as it has to calculate distances for each test point).
- Sensitive to Noisy Data: KNN is sensitive to outliers and noisy data, which can skew the results, especially when a small value of k is used.
- Choice of K : The performance of KNN heavily depends on the choice of k . A small k can lead to overfitting, while a large k can smooth out the prediction too much.
- Feature Scaling: KNN is highly sensitive to the scale of the input features. If one feature has larger values than others, it will dominate the distance calculation unless scaling is applied.
- Memory Intensive: KNN requires a large amount of memory to store all the training data, as every instance needs to be retained for classification.

Outcomes: Apply concepts of different types of Learning and Neural Network

Conclusion (based on the Results and outcomes achieved):

The KNN algorithm was successfully implemented for classifying test data based on their nearest neighbors. Using $k = 3$, the algorithm accurately predicted the class of test points by calculating Euclidean distances. KNN is simple and effective for small datasets, but its performance depends on the choice of k and is sensitive to noise and large datasets. Proper data preprocessing and parameter tuning are essential for optimal results.

References:

Books/ Journals/ Websites:

1. Han, Kamber, "Data Mining Concepts and Techniques", Morgan Kaufmann 3rd Edition
 2. <https://www.kaggle.com/datasets/imakash3011/customer-personality-analysis>
-