Smt. Indira Gandhi College of Engineering
Computer Engineering Department
Ghansoli – Navi Mumbai
Academic Year 2023-24 (Even Sem)

Student Name: Khyati Garude **Roll No.:** 13 **Class:** BE **Sem:** VIII

Course Name: Applied Data Science Lab

Course Code: CSL8023

Experiment No. 07

Experiment Title: Anomaly / Outlier Detection Using Distance Based Method

Date of Performance	Date of Submission	Marks (10)					Sign / Remark
		A	B	C	D	E	
		2	3	2	2	1	
14/3/24	21/3/24	2	3	2	2	1	12/4/24
		Total Marks					
		10					

A: Prerequisite Knowledge
B: Implementation
C: Oral
D: Content
E: Punctuality & Discipline

Khyati Garude
BE COMPS
ADS

Date:

DATE	EXPERIMENT-7	SIGN
21/3/24	Anomaly / outlier detection using distance based method.	21/4/24.

AIM: Anomaly / outlier detection using distance based method.

THEORY:

Anomaly or outlier detection using unsupervised k-Nearest Neighbours (KNN) involves using the distance-based KNN algorithm to identify data points that deviate significantly from their neighbors.

→ At the core of the KNN algorithm lies the concept of nearest neighbour search. Given a dataset, each data point's proximity to its neighbors is computed based on a chosen distance metric, such as Euclidean distance or Manhattan distance.

→ In the unsupervised setting, the KNN algorithm does not require labeled data. For each data point in the dataset, it identifies its 'k' nearest neighbours based on their distances.

Date:_____

The value of 'K' is a hyperparameter that needs to be specified beforehand.

→ Once the nearest neighbors for each data point are identified, an outlier score is computed for each data point. This score is typically based on the distance to its 'k' th nearest neighbor. The farther a data point is from its Kth nearest neighbour, the higher its outlier score.

→ Advantage of KNN based anomaly detection is its interpretability. Since anomalies are identified based on their distances to neighbors, it's often easier to understand why a particular data point is flagged as an outlier compared to more complex anomaly detection algorithms.

→ One challange with KNN - based anomaly detection is scalability, especially as the size of dataset increases. Computing distances to all data points can become computationally expensive, particularly in high - dimensional spaces. Approximate nearest neighbor search techniques or dimensionality reduction methods can be employed to address this challenge.

Date:

ALGORITHM:

(i) Input:
The function 'detect_anomaly' takes a data array and an optional parameter 'k' (default set to 2).

(ii) Initialization:
Initialize empty lists for storing data points ('datapoint') and anomalies ('anomaly').

(iii) Loop through data points:
For each data point in the array:
- → Compute distances to all other data points except itself.
- → find the 'k' smallest distances.
- → Calculate the mean of these 'k' distances.

(iv) Anomaly detection:
If the mean of the 'k' smallest distances is greater than zero, classify the data point as an anomaly and append it to the 'anomaly' list. Otherwise, classify it as a regular data point and append it to the 'datapoint' list.

(v) Output:
Return the lists containing regular data points and anomalies.

Date:

(vi) Visualization:

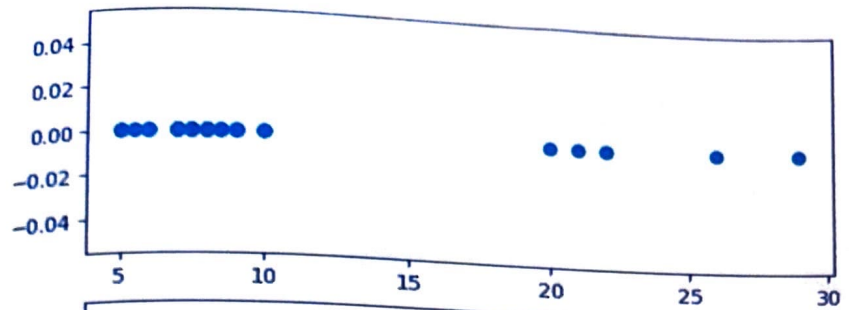
Plot the original data points along with anomalies to visualize the outliers detected by the algorithm.

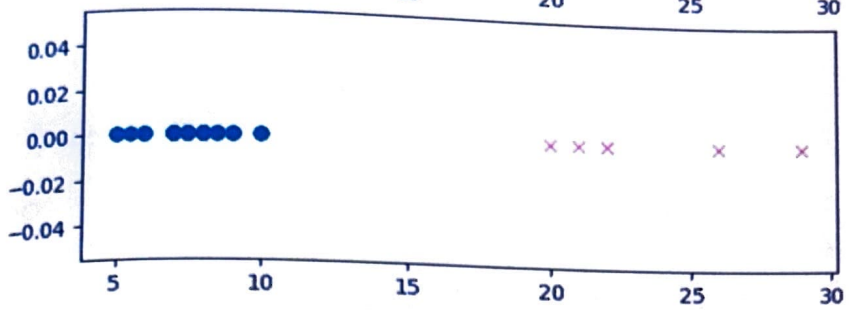
CONCLUSION:

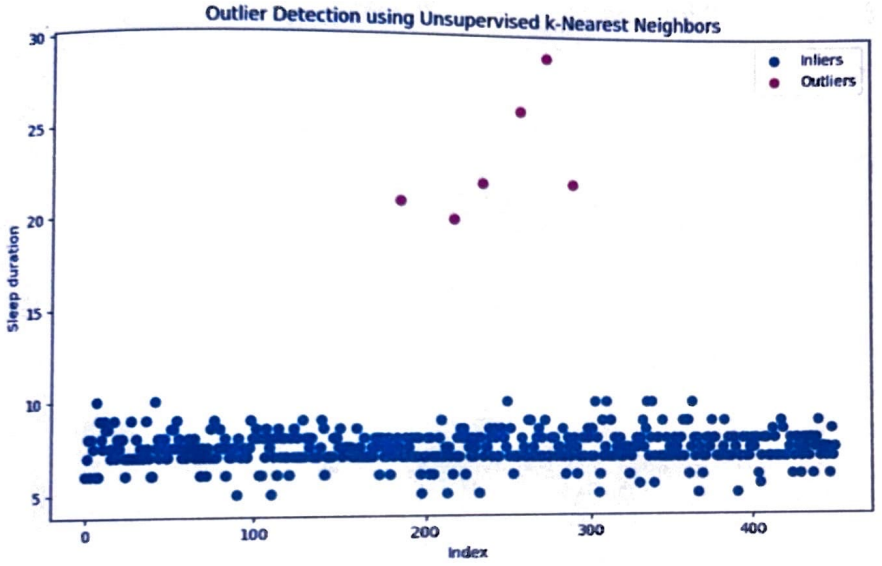
By leveraging the unsupervised KNN algorithm, anomaly detection can effectively identify outliers in datasets without requiring labeled data, making it suitable for detecting anomalies in real-time or in situations where labeled data is scarce. Overall, anomaly detection using distance based method leverages the concept of nearest neighbors to identify outliers in datasets, making it suitable for various applications across different domains.

12/4/24

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

def detect_anomaly(data_array, k=2):
    #k = k
    datapoint = []
    anomaly = []
    for i in range(len(data_array)):
        mean = 0
        distances = []
        for j in range(len(data_array)):
            if i == j: continue
            dist = data_array[j] - data_array[i]
            distances.append(abs(dist))

        distances.sort()
        kmin = distances[0:k]

        for j in range(len(kmin)):
            mean += kmin[j]
        mean = mean / len(kmin)
        #print(mean)
        anomaly.append(data_array[i]) if mean > 0 else datapoint.append(data_array[i])

    return datapoint, anomaly

# Load data from Excel file
data = pd.read_csv("/content/outliers.csv")

d = list(data["Sleep duration"])
datapoint, anomaly = detect_anomaly(d)
val = 0

plt.figure(1)
plt.subplot(211)
plt.plot(d, np.zeros_like(d) + val, 'o')  # plotting the original data
plt.subplot(212)
plt.plot(datapoint, np.zeros_like(datapoint) + val, 'o') # plotting the original data
plt.plot(anomaly, np.zeros_like(anomaly) + val, 'x')  # plotting anomalies
plt.show()

plt.figure(figsize=(10, 6))
plt.scatter(range(len(d)), d, c='b', label='Inliers')
plt.scatter(outlier_indices, outliers, c='r', label='Outliers')
plt.xlabel('Index')
plt.ylabel('Sleep duration')
plt.title('Outlier Detection using Unsupervised k-Nearest Neighbors')
plt.legend()
plt.show()
```


0.04
0.02
0.00
−0.02
−0.04
5
10
15
20
25
30


0.04
0.02
0.00
−0.02
−0.04
5
10
15
20
25
30


Outlier Detection using Unsupervised k-Nearest Neighbors
Inliers
Outliers
Sleep duration
Index
30
25
20
15
10
5
0
100
200
300
400

12/4/24.