Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Summer, Year:2025), B.Sc. in CSE (Day)

Lab Report NO #4

Course Title: Artificial Intelligent

Course Code: CSE 316 Section: 221 D12

Lab Experiment Name: kMeansClustering

Student Details

	Name	ID
1.	Al Ekram Hossain	221002535

Lab Date : 15 April 2025

Submission Date : 26 April 2025

Course Teacher's Name : Md. Sabbir Hosen Mamun

Lab Report Status

Marks: …… **Signature:.....**

Comments:..... **Date:.....**

Title : kMeansClustering

Objectives :

- Implement a **modified K-Means clustering algorithm** using **Python**.
- Replace Euclidean distance with **Manhattan distance** for clustering.
- Generate **100 random points** and **10 cluster centers**.
- Visualize the clustered data on a **2D grid** using only the `print()` function.

Problem Analysis :

By substituting the Manhattan distance metric for the conventional Euclidean distance, the usual K-Means clustering algorithm must be modified.

Clustering 100 randomly generated points into ten groups according to how close they are to the original cluster centers is the aim.

A 2D grid must be created, where each cell represents a coordinate on the plane and uses different symbols to depict either a data point or a cluster center, as only the print() function may be used for visualization.

Accurate distance calculation, proper cluster assignment, iterative center updating, and efficient visualization within constrained console output capabilities are the primary obstacles.

Implementation :

```
import random

WIDTH = 30
HEIGHT = 15
NUM_POINTS = 100
NUM_CLUSTERS = 10

POINT_SYMBOL = '.'
CLUSTER_SYMBOLS = ['A','B','C','D','E','F','G','H','I','J']

def manhattan_distance(p1, p2):
```

```
    return abs(p1[0] - p2[0]) + abs(p1[1] - p2[1])

points = [(random.randint(0, WIDTH-1), random.randint(0, HEIGHT-1)) for _ in
range(NUM_POINTS)]
clusters = [(random.randint(0, WIDTH-1), random.randint(0, HEIGHT-1)) for _
in range(NUM_CLUSTERS)]

def assign_clusters(points, clusters):
    assignments = []
    for p in points:
        distances = [manhattan_distance(p, c) for c in clusters]
        cluster_index = distances.index(min(distances))
        assignments.append(cluster_index)
    return assignments

def update_clusters(points, assignments, num_clusters):
    new_clusters = []
    for i in range(num_clusters):
        cluster_points = [p for p, a in zip(points, assignments) if a == i]
        if cluster_points:
            avg_x = sum(p[0] for p in cluster_points) // len(cluster_points)
            avg_y = sum(p[1] for p in cluster_points) // len(cluster_points)
            new_clusters.append((avg_x, avg_y))
        else:
              new_clusters.append((random.randint(0, WIDTH-1), random.randint(0,
HEIGHT-1)))
    return new_clusters

def k_means(points, clusters, iterations=5):
    for _ in range(iterations):
        assignments = assign_clusters(points, clusters)
        clusters = update_clusters(points, assignments, NUM_CLUSTERS)
    return assignments, clusters

assignments, final_clusters = k_means(points, clusters)
```

```
grid = [[' ' for _ in range(WIDTH)] for _ in range(HEIGHT)]

for (x, y), cluster_idx in zip(points, assignments):
    grid[y][x] = CLUSTER_SYMBOLS[cluster_idx]

for idx, (x, y) in enumerate(final_clusters):
    grid[y][x] = CLUSTER_SYMBOLS[idx]

for row in grid:
    print(' '.join(row))
```

Output :

```
assignments = assign_clusters(points, clusters)
clusters = update_clusters(points, assignments, NUM CLUSTERS)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
    F F   E                 J         B     B      B
F F   F           H H     J   J      B   B     B B B B
    F           H H       J J   J           B
F   F           H           J     J      G
        D                 A J
I               D      A               G           G
    I        D        A    A        G G G G G   G G
I I            D    A                             G
  I              D        A A      A             G
                             A   A    G    G
          C     C     A           A               G
I     C                A A     A        G
PS E:\Ai>
```

Conclusion :

Using the Manhattan distance metric, the modified K-Means clustering technique was effectively applied in this experiment.

With just the print() method, we were able to efficiently show the findings on a 2D grid and cluster 100 random points into 10 different groups.

This experiment showed how simple algorithms may be modified to meet particular needs and clarified the effects of various distance measurements on clustering outcomes.

Additionally, the project reinforced the ideas of data display in a limited setting, iterative improvement, and distance computation.