Soft Computing (CS3123)

Assignment-1

1.A) CODE:

```
import pandas as pd
import numpy as np
df = pd.read_csv('seed_data.csv')
print(df)
```

OUTPUT:

	A	P	C	LK	WK	A_Coef	LKG	target
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220	0
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	0
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825	0
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	0
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	0
...	...	...	...	...	...	...	...	...
205	12.19	13.20	0.8783	5.137	2.981	3.631	4.870	2
206	11.23	12.88	0.8511	5.140	2.795	4.325	5.003	2
207	13.20	13.66	0.8883	5.236	3.232	8.315	5.056	2
208	11.84	13.21	0.8521	5.175	2.836	3.598	5.044	2
209	12.30	13.34	0.8684	5.243	2.974	5.637	5.063	2

```
df = df.drop('target', axis=1)
print(df)
```

OUTPUT:

	A	P	C	LK	WK	A_Coef	LKG
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175
...	...	...	...	...	...	...	...
205	12.19	13.20	0.8783	5.137	2.981	3.631	4.870
206	11.23	12.88	0.8511	5.140	2.795	4.325	5.003
207	13.20	13.66	0.8883	5.236	3.232	8.315	5.056
208	11.84	13.21	0.8521	5.175	2.836	3.598	5.044
209	12.30	13.34	0.8684	5.243	2.974	5.637	5.063

1.C) CODE:

```
def normalized_data(data):
    col_len = len(data.columns)
    normalized_df = data.copy()
    for col in range(col_len):
            max_value = normalized_df.iloc[:,col].max()
            normalized_df.iloc[:,col] = normalized_df.iloc[:,col] /
max_value
    return normalized_df

normalized_df = normalized_data(df)
print(normalized_df)
```

OUTPUT:

	A	P	C	LK	WK	A_Coef	LKG
0	0.720491	0.860290	0.948492	0.863371	0.821225	0.262654	0.796947
1	0.702550	0.844638	0.959490	0.832060	0.826432	0.120388	0.756641
2	0.674693	0.816812	0.985517	0.792659	0.827424	0.319182	0.736641
3	0.653447	0.808116	0.975172	0.797603	0.837838	0.267148	0.733588
4	0.762040	0.868986	0.983774	0.847640	0.883213	0.160241	0.790076
...	...	...	...	...	...	...	...
205	0.575543	0.765217	0.956441	0.769588	0.739152	0.429399	0.743511
206	0.530217	0.746667	0.926821	0.770037	0.693032	0.511471	0.763817
207	0.623229	0.791884	0.967331	0.784419	0.801389	0.983325	0.771908
208	0.559018	0.765797	0.927910	0.775281	0.703199	0.425497	0.770076
209	0.580737	0.773333	0.945660	0.785468	0.737416	0.666627	0.772977

2.A) CODE:

```
def calculate_similarity_matrix(data):
    m = len(data)
    S = np.zeros((m,m))

    for i in range(m):
        for j in range(i,m):
            distance = np.sqrt(sum((data.iloc[i] - data.iloc[j])**2))
            S[i][j] = distance
            S[j][i] = distance
    return S

S = calculate_similarity_matrix(normalized_df)
print(S)
```

OUTPUT:

```
array([[0.       , 0.15349082, 0.13126062, ..., 0.73560553, 0.29044088,
        0.45162218],
       [0.15349082, 0.       , 0.20904546, ..., 0.87001563, 0.37349252,
        0.57345694],
       [0.13126062, 0.20904546, 0.       , ..., 0.66833906, 0.21784492,
        0.37748974],
       ...,
       [0.73560553, 0.87001563, 0.66833906, ..., 0.       , 0.57206529,
        0.32712662],
       [0.29044088, 0.37349252, 0.21784492, ..., 0.57206529, 0.       ,
        0.24550062],
       [0.45162218, 0.57345694, 0.37748974, ..., 0.32712662, 0.24550062 ,  0. ]])
```

2.B) CODE:

```
def clusters(S):
    clusters = []
    m = len(S)
    for i in range(m):
        avg_dissimilarity = np.mean(S[i])
        cluster_i = [j for j in range(m) if S[i][j] < avg_dissimilarity]
        clusters.append(cluster_i)

    print("Clusters:")
    for i, cluster in enumerate(clusters):
        print(f"Cluster C{i+1}: {cluster}")

print(clusters(S))
```

```
Clusters:
Cluster C1: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 2
6, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 40, 41, 42, 44, 45, 46, 47, 48, 49, 50, 52, 53, 5
4, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 74, 76, 79, 85, 86, 91, 92, 95, 98, 9
9, 100, 101, 105, 106, 107, 109, 110, 112, 115, 121, 122, 123, 124, 127, 130, 131, 132, 133, 1
35, 136, 137, 138, 139, 146, 148, 160, 165, 167, 179, 192, 198, 199, 201, 204, 205, 208]
:
Cluster C131: [0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 17, 18, 20, 21, 22, 24, 25, 28, 31, 32, 33,
34, 35, 36, 37, 38, 40, 41, 43, 44, 46, 47, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 66, 67, 68, 70,
71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 89, 90, 91, 92, 94, 95, 96, 97,
98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 114, 115, 116, 117, 1
18, 119, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137,
138, 139]
:
Cluster C210: [6, 10, 12, 13, 14, 15, 16, 19, 20, 26, 29, 31, 32, 38, 39, 43, 50, 51, 52, 53, 62, 6
3, 69, 71, 75, 80, 100, 122, 129, 132, 133, 134, 135, 140, 141, 142, 143, 144, 145, 147, 148, 1
49, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 166, 167, 168,
169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 180, 181, 182, 183, 184, 185, 186, 187, 188
, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 202, 203, 204, 205, 206, 207, 20
8, 209]
```

3.A) CODE:

```
def form_initial_clusters(similarity_matrix):
    m = len(similarity_matrix)
    clusters = []
    for i in range(m):
        avg_dissimilarity = np.mean(similarity_matrix[i])
        cluster_i = [j for j in range(m) if similarity_matrix[i][j] < avg_dissimilarity]
        is_subset = False
        for cluster in clusters:
            if set(cluster_i).issubset(set(cluster)):
                is_subset = True
                break
        if not is_subset:
            clusters.append(cluster_i)
    return clusters

initial_clusters = form_initial_clusters(S)
print(len(initial_clusters), initial_clusters)
```

OUTPUT:

164

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 40, 41, 42, 44, 45, 46, 47, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 74, 76, 79, 85, 86, 91, 92, 95, 98, 99, 100, 101, 105, 106, 107, 109, 110, 112, 115, 121, 122, 123, 124, 127, 130, 131, 132, 133, 135, 136, 137, 138, 139, 146, 148, 160, 165, 167, 179, 192, 198, 199, 201, 204, 205, 208]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 40, 41, 42, 44, 45, 46, 47, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 79, 83, 85, 86, 95, 97, 98, 99, 101, 105, 106, 109, 110, 112, 115, 123, 124, 127, 130, 131, 135, 136, 137, 138, 139, 146, 148, 153, 156, 160, 165, 167, 179, 192, 198, 199, 201, 204, 205, 208]
.
.
.
[6, 10, 12, 13, 14, 15, 16, 19, 20, 26, 29, 31, 32, 38, 39, 43, 50, 51, 52, 53, 62, 63, 69, 71, 75, 80, 100, 122, 129, 132, 133, 134, 135, 140, 141, 142, 143, 144, 145, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 202, 203, 204, 205, 206, 207, 208, 209]

3.B) CODE:

```
def calculate_cluster_similarity(clusters):
    p = len(clusters)
    C = np.zeros((p, p))

    def similarity_measure(ci, cj):
        intersection = len(set(ci).intersection(set(cj)))
        union = len(set(ci).union(set(cj)))
        return intersection / union

    for i in range(p):
        for j in range(i+1, p):
            similarity = similarity_measure(clusters[i], clusters[j])
            C[i][j] = similarity
            C[j][i] = similarity

    return C

similarity_matrix = calculate_cluster_similarity(initial_clusters);
print(similarity_matrix)
```

OUTPUT:

```
array([[0.       , 0.8018018 , 0.83333333, ..., 0.69026549, 0.74789916,
        0.58677686],
       [0.8018018 , 0.       , 0.82882883, ..., 0.81372549, 0.77192982,
        0.6460177 ],
       [0.83333333, 0.82882883, 0.       , ..., 0.67241379, 0.9009009 ,
        0.53543307],
       ...,
       [0.69026549, 0.81372549, 0.67241379, ..., 0.       , 0.62184874,
        0.75247525],
       [0.74789916, 0.77192982, 0.9009009 , ..., 0.62184874, 0.       ,
        0.49230769],
       [0.58677686, 0.6460177 , 0.53543307, ..., 0.75247525, 0.49230769,
        0.       ]])
```

3.C) CODE:

```
def merge_most_similar_clusters(cluster_similarity_matrix, clusters):
    max_val = np.max(cluster_similarity_matrix)
    k= np.argmax(cluster_similarity_matrix)//cluster_similarity_matrix.
shape[0]
    l = np.argmax(cluster_similarity_matrix)%cluster_similarity_matrix.
shape[0]

    new_cluster = list(set(clusters[k] + clusters[l]))
    a = [m for m , _ in enumerate(clusters) if m in [k,l]]
    clusters = [c for i, c in enumerate(clusters) if i not in [k, l]]
    clusters.append(new_cluster)

    return clusters
```

3.D) CODE:

```
K = 3
while len(initial_clusters) > K:
    cluster_similarity_matrix =
calculate_cluster_similarity(initial_clusters)
    initial_clusters =
merge_most_similar_clusters(cluster_similarity_matrix,
initial_clusters)

print("Final Clusters:")
for i, cluster in enumerate(initial_clusters):
    print(f"Cluster C{i+1}: {cluster}")
```

OUTPUT:

Final Clusters:
Cluster C1: [6, 10, 12, 13, 15, 16, 19, 20, 29, 31, 32, 36, 37, 38, 39, 43, 44, 47, 4
8, 50, 51, 52, 53, 56, 63, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 84, 85,
87, 88, 89, 91, 92, 93, 94, 95, 96, 98, 100, 102, 103, 104, 105, 106, 107, 108, 11
0, 111, 113, 114, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 12
9, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 147, 15
0, 151, 152, 157, 159, 161, 162, 163, 166, 181, 182, 183, 184, 186, 194, 195, 19
6, 197, 200, 203, 207, 209]
Cluster C2: [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, 39, 40, 41, 42,
43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 6
4, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85,
86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 10
5, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 12
1, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 13
7, 138, 139, 140, 141, 142, 146, 147, 148, 152, 153, 154, 156, 160, 161, 163, 16
5, 166, 167, 169, 179, 181, 184, 185, 191, 192, 194, 195, 197, 198, 199, 201, 20
2, 204, 205, 208]
Cluster C3: [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, 39, 40, 41, 42,
43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 6
4, 65, 66, 67, 68, 69, 70, 71, 72, 74, 75, 76, 78, 79, 80, 81, 82, 87, 91, 92, 93, 94,
95, 96, 98, 100, 101, 102, 103, 104, 107, 111, 113, 116, 118, 120, 121, 122, 123,
124, 125, 126, 129, 130, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158,
159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174,
175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190,
191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206,
207, 208, 209]