assignment-1

August 8, 2023

[1]:
```python
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from scipy.spatial import distance_matrix as dm
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
```

[2]:
```python
df = pd.read_csv('iris.csv')
df
```

[2]:
```
     sepal_length  sepal_width  petal_length  petal_width           class
0             5.1          3.5           1.4          0.2     Iris-setosa
1             4.9          3.0           1.4          0.2     Iris-setosa
2             4.7          3.2           1.3          0.2     Iris-setosa
3             4.6          3.1           1.5          0.2     Iris-setosa
4             5.0          3.6           1.4          0.2     Iris-setosa
..            …            …             …            …             …
145           6.7          3.0           5.2          2.3  Iris-virginica
146           6.3          2.5           5.0          1.9  Iris-virginica
147           6.5          3.0           5.2          2.0  Iris-virginica
148           6.2          3.4           5.4          2.3  Iris-virginica
149           5.9          3.0           5.1          1.8  Iris-virginica

[150 rows x 5 columns]
```

[3]:
```python
df.drop(columns='class', axis=0, inplace=True)
df
```

[3]:
```
     sepal_length  sepal_width  petal_length  petal_width
0             5.1          3.5           1.4          0.2
1             4.9          3.0           1.4          0.2
2             4.7          3.2           1.3          0.2
3             4.6          3.1           1.5          0.2
4             5.0          3.6           1.4          0.2
..            …            …             …            …
145           6.7          3.0           5.2          2.3
146           6.3          2.5           5.0          1.9
```

```
147           6.5          3.0           5.2          2.0
148           6.2          3.4           5.4          2.3
149           5.9          3.0           5.1          1.8

[150 rows x 4 columns]
```

[4]:
```python
scaler = MinMaxScaler()
scaler.fit(df)
df2 = scaler.transform(df)
df2
```

[4]:
```
array([[0.22222222, 0.625     , 0.06779661, 0.04166667],
       [0.16666667, 0.41666667, 0.06779661, 0.04166667],
       [0.11111111, 0.5       , 0.05084746, 0.04166667],
       [0.08333333, 0.45833333, 0.08474576, 0.04166667],
       [0.19444444, 0.66666667, 0.06779661, 0.04166667],
       [0.30555556, 0.79166667, 0.11864407, 0.125     ],
       [0.08333333, 0.58333333, 0.06779661, 0.08333333],
       [0.19444444, 0.58333333, 0.08474576, 0.04166667],
       [0.02777778, 0.375     , 0.06779661, 0.04166667],
       [0.16666667, 0.45833333, 0.08474576, 0.        ],
       [0.30555556, 0.70833333, 0.08474576, 0.04166667],
       [0.13888889, 0.58333333, 0.10169492, 0.04166667],
       [0.13888889, 0.41666667, 0.06779661, 0.        ],
       [0.        , 0.41666667, 0.01694915, 0.        ],
       [0.41666667, 0.83333333, 0.03389831, 0.04166667],
       [0.38888889, 1.        , 0.08474576, 0.125     ],
       [0.30555556, 0.79166667, 0.05084746, 0.125     ],
       [0.22222222, 0.625     , 0.06779661, 0.08333333],
       [0.38888889, 0.75      , 0.11864407, 0.08333333],
       [0.22222222, 0.75      , 0.08474576, 0.08333333],
       [0.30555556, 0.58333333, 0.11864407, 0.04166667],
       [0.22222222, 0.70833333, 0.08474576, 0.125     ],
       [0.08333333, 0.66666667, 0.        , 0.04166667],
       [0.22222222, 0.54166667, 0.11864407, 0.16666667],
       [0.13888889, 0.58333333, 0.15254237, 0.04166667],
       [0.19444444, 0.41666667, 0.10169492, 0.04166667],
       [0.19444444, 0.58333333, 0.10169492, 0.125     ],
       [0.25      , 0.625     , 0.08474576, 0.04166667],
       [0.25      , 0.58333333, 0.06779661, 0.04166667],
       [0.11111111, 0.5       , 0.10169492, 0.04166667],
       [0.13888889, 0.45833333, 0.10169492, 0.04166667],
       [0.30555556, 0.58333333, 0.08474576, 0.125     ],
       [0.25      , 0.875     , 0.08474576, 0.        ],
       [0.33333333, 0.91666667, 0.06779661, 0.04166667],
       [0.16666667, 0.45833333, 0.08474576, 0.        ],
       [0.19444444, 0.5       , 0.03389831, 0.04166667],
```

```
[0.33333333, 0.625     , 0.05084746, 0.04166667],
[0.16666667, 0.45833333, 0.08474576, 0.        ],
[0.02777778, 0.41666667, 0.05084746, 0.04166667],
[0.22222222, 0.58333333, 0.08474576, 0.04166667],
[0.19444444, 0.625     , 0.05084746, 0.08333333],
[0.05555556, 0.125     , 0.05084746, 0.08333333],
[0.02777778, 0.5       , 0.05084746, 0.04166667],
[0.19444444, 0.625     , 0.10169492, 0.20833333],
[0.22222222, 0.75      , 0.15254237, 0.125     ],
[0.13888889, 0.41666667, 0.06779661, 0.08333333],
[0.22222222, 0.75      , 0.10169492, 0.04166667],
[0.08333333, 0.5       , 0.06779661, 0.04166667],
[0.27777778, 0.70833333, 0.08474576, 0.04166667],
[0.19444444, 0.54166667, 0.06779661, 0.04166667],
[0.75      , 0.5       , 0.62711864, 0.54166667],
[0.58333333, 0.5       , 0.59322034, 0.58333333],
[0.72222222, 0.45833333, 0.66101695, 0.58333333],
[0.33333333, 0.125     , 0.50847458, 0.5       ],
[0.61111111, 0.33333333, 0.61016949, 0.58333333],
[0.38888889, 0.33333333, 0.59322034, 0.5       ],
[0.55555556, 0.54166667, 0.62711864, 0.625     ],
[0.16666667, 0.16666667, 0.38983051, 0.375     ],
[0.63888889, 0.375     , 0.61016949, 0.5       ],
[0.25      , 0.29166667, 0.49152542, 0.54166667],
[0.19444444, 0.        , 0.42372881, 0.375     ],
[0.44444444, 0.41666667, 0.54237288, 0.58333333],
[0.47222222, 0.08333333, 0.50847458, 0.375     ],
[0.5       , 0.375     , 0.62711864, 0.54166667],
[0.36111111, 0.375     , 0.44067797, 0.5       ],
[0.66666667, 0.45833333, 0.57627119, 0.54166667],
[0.36111111, 0.41666667, 0.59322034, 0.58333333],
[0.41666667, 0.29166667, 0.52542373, 0.375     ],
[0.52777778, 0.08333333, 0.59322034, 0.58333333],
[0.36111111, 0.20833333, 0.49152542, 0.41666667],
[0.44444444, 0.5       , 0.6440678 , 0.70833333],
[0.5       , 0.33333333, 0.50847458, 0.5       ],
[0.55555556, 0.20833333, 0.66101695, 0.58333333],
[0.5       , 0.33333333, 0.62711864, 0.45833333],
[0.58333333, 0.375     , 0.55932203, 0.5       ],
[0.63888889, 0.41666667, 0.57627119, 0.54166667],
[0.69444444, 0.33333333, 0.6440678 , 0.54166667],
[0.66666667, 0.41666667, 0.6779661 , 0.66666667],
[0.47222222, 0.375     , 0.59322034, 0.58333333],
[0.38888889, 0.25      , 0.42372881, 0.375     ],
[0.33333333, 0.16666667, 0.47457627, 0.41666667],
[0.33333333, 0.16666667, 0.45762712, 0.375     ],
[0.41666667, 0.29166667, 0.49152542, 0.45833333],
```

```
[0.47222222, 0.29166667, 0.69491525, 0.625     ],
[0.30555556, 0.41666667, 0.59322034, 0.58333333],
[0.47222222, 0.58333333, 0.59322034, 0.625     ],
[0.66666667, 0.45833333, 0.62711864, 0.58333333],
[0.55555556, 0.125     , 0.57627119, 0.5       ],
[0.36111111, 0.41666667, 0.52542373, 0.5       ],
[0.33333333, 0.20833333, 0.50847458, 0.5       ],
[0.33333333, 0.25      , 0.57627119, 0.45833333],
[0.5       , 0.41666667, 0.61016949, 0.54166667],
[0.41666667, 0.25      , 0.50847458, 0.45833333],
[0.19444444, 0.125     , 0.38983051, 0.375     ],
[0.36111111, 0.29166667, 0.54237288, 0.5       ],
[0.38888889, 0.41666667, 0.54237288, 0.45833333],
[0.38888889, 0.375     , 0.54237288, 0.5       ],
[0.52777778, 0.375     , 0.55932203, 0.5       ],
[0.22222222, 0.20833333, 0.33898305, 0.41666667],
[0.38888889, 0.33333333, 0.52542373, 0.5       ],
[0.55555556, 0.54166667, 0.84745763, 1.        ],
[0.41666667, 0.29166667, 0.69491525, 0.75      ],
[0.77777778, 0.41666667, 0.83050847, 0.83333333],
[0.55555556, 0.375     , 0.77966102, 0.70833333],
[0.61111111, 0.41666667, 0.81355932, 0.875     ],
[0.91666667, 0.41666667, 0.94915254, 0.83333333],
[0.16666667, 0.20833333, 0.59322034, 0.66666667],
[0.83333333, 0.375     , 0.89830508, 0.70833333],
[0.66666667, 0.20833333, 0.81355932, 0.70833333],
[0.80555556, 0.66666667, 0.86440678, 1.        ],
[0.61111111, 0.5       , 0.69491525, 0.79166667],
[0.58333333, 0.29166667, 0.72881356, 0.75      ],
[0.69444444, 0.41666667, 0.76271186, 0.83333333],
[0.38888889, 0.20833333, 0.6779661 , 0.79166667],
[0.41666667, 0.33333333, 0.69491525, 0.95833333],
[0.58333333, 0.5       , 0.72881356, 0.91666667],
[0.61111111, 0.41666667, 0.76271186, 0.70833333],
[0.94444444, 0.75      , 0.96610169, 0.875     ],
[0.94444444, 0.25      , 1.        , 0.91666667],
[0.47222222, 0.08333333, 0.6779661 , 0.58333333],
[0.72222222, 0.5       , 0.79661017, 0.91666667],
[0.36111111, 0.33333333, 0.66101695, 0.79166667],
[0.94444444, 0.33333333, 0.96610169, 0.79166667],
[0.55555556, 0.29166667, 0.66101695, 0.70833333],
[0.66666667, 0.54166667, 0.79661017, 0.83333333],
[0.80555556, 0.5       , 0.84745763, 0.70833333],
[0.52777778, 0.33333333, 0.6440678 , 0.70833333],
[0.5       , 0.41666667, 0.66101695, 0.70833333],
[0.58333333, 0.33333333, 0.77966102, 0.83333333],
[0.80555556, 0.41666667, 0.81355932, 0.625     ],
```

```
       [0.86111111, 0.33333333, 0.86440678, 0.75      ],
       [1.        , 0.75      , 0.91525424, 0.79166667],
       [0.58333333, 0.33333333, 0.77966102, 0.875     ],
       [0.55555556, 0.33333333, 0.69491525, 0.58333333],
       [0.5       , 0.25      , 0.77966102, 0.54166667],
       [0.94444444, 0.41666667, 0.86440678, 0.91666667],
       [0.55555556, 0.58333333, 0.77966102, 0.95833333],
       [0.58333333, 0.45833333, 0.76271186, 0.70833333],
       [0.47222222, 0.41666667, 0.6440678 , 0.70833333],
       [0.72222222, 0.45833333, 0.74576271, 0.83333333],
       [0.66666667, 0.45833333, 0.77966102, 0.95833333],
       [0.72222222, 0.45833333, 0.69491525, 0.91666667],
       [0.41666667, 0.29166667, 0.69491525, 0.75      ],
       [0.69444444, 0.5       , 0.83050847, 0.91666667],
       [0.66666667, 0.54166667, 0.79661017, 1.        ],
       [0.66666667, 0.41666667, 0.71186441, 0.91666667],
       [0.55555556, 0.20833333, 0.6779661 , 0.75      ],
       [0.61111111, 0.41666667, 0.71186441, 0.79166667],
       [0.52777778, 0.58333333, 0.74576271, 0.91666667],
       [0.44444444, 0.41666667, 0.69491525, 0.70833333]])
```

[5]:

```
k_rng = range(1, 10)
sse = []
for k in k_rng:
    km = KMeans(n_clusters=k, n_init=10)
    km.fit(df2)
    sse.append(km.inertia_)
sse
```

[5]:

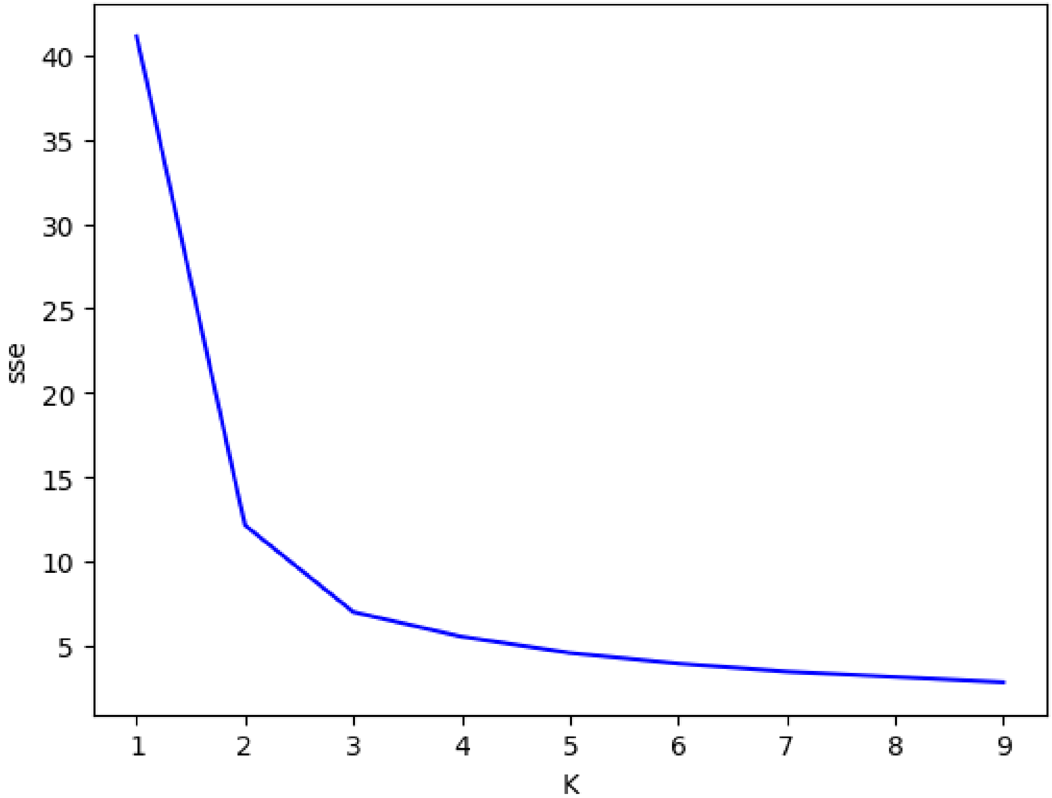
```
[41.13817202297777,
 12.14368828157972,
 6.998114004826761,
 5.532831003081897,
 4.571211374951953,
 3.940719646486447,
 3.461683375476735,
 3.14413280606377,
 2.828268052983513]
```

[6]:

```
plt.xlabel('K')
plt.ylabel('sse')
plt.plot(k_rng, sse, color='blue')
```

[6]:

```
[<matplotlib.lines.Line2D at 0x7ff4e3ddff40>]
```




```
[7]: km = KMeans(n_clusters=3, n_init=10)
     y_predicted = km.fit_predict(df2)
     df['cluster'] = y_predicted
     df
```

```
[7]:      sepal_length  sepal_width  petal_length  petal_width  cluster
     0             5.1          3.5           1.4          0.2        0
     1             4.9          3.0           1.4          0.2        0
     2             4.7          3.2           1.3          0.2        0
     3             4.6          3.1           1.5          0.2        0
     4             5.0          3.6           1.4          0.2        0
     ..            …            …             …            …          …
     145           6.7          3.0           5.2          2.3        1
     146           6.3          2.5           5.0          1.9        2
     147           6.5          3.0           5.2          2.0        1
     148           6.2          3.4           5.4          2.3        1
     149           5.9          3.0           5.1          1.8        2

     [150 rows x 5 columns]
```

1 Part 2

[8]:
```
d = dm(df2, df2, p=2)
d
```

[8]:
```
array([[0.        , 0.21561354, 0.16810102, …, 1.08257132, 1.14907064,
        0.96462829],
       [0.21561354, 0.        , 0.10157824, …, 1.08390691, 1.17619813,
        0.95649502],
       [0.16810102, 0.10157824, 0.        , …, 1.12088708, 1.19544459,
        0.98859665],
       …,
       [1.08257132, 1.08390691, 1.12088708, …, 0.        , 0.226928  ,
        0.18710825],
       [1.14907064, 1.17619813, 1.19544459, …, 0.226928  , 0.        ,
        0.28409587],
       [0.96462829, 0.95649502, 0.98859665, …, 0.18710825, 0.28409587,
        0.        ]])
```

[9]:
```
avg_dissimilarity = []
m = len(d)
for i in range(m):
    total = sum(d[i])
    avg_dissimilarity.append(total/m)
avg_dissimilarity
```

[9]:
```
[0.710406198236202,
 0.7138986862151165,
 0.7331428280458493,
 0.7318745290001425,
 0.7289888779248361,
 0.7203631947720547,
 0.7313492260044282,
 0.7000819128985032,
 0.7773679398645925,
 0.7218920520551985,
 0.7242977407104872,
 0.7108807042218028,
 0.7421556847047821,
 0.8265103558500383,
 0.8113556377135483,
 0.8621134352439145,
 0.7471221384958912,
 0.6937607139759608,
 0.7173906445985772,
 0.7341393002871166,
 0.6777652639033831,
```

0.7028127916485095,
0.8007733089648635,
0.6376674587099815,
0.6972992617841202,
0.6942360366953089,
0.6646808902046466,
0.6998320643973082,
0.69842017906821,
0.7127923086027982,
0.7035681576344529,
0.6588257203026473,
0.8330355365237214,
0.8401710685040565,
0.7218920520551985,
0.7168876373517267,
0.7143195814831513,
0.7218920520551985,
0.7775240673440745,
0.6946481410279595,
0.7075068333928404,
0.8412607268061166,
0.7734553660826652,
0.6593644480603104,
0.6998566739515535,
0.7059922866531654,
0.7446607116184024,
0.7376189867152552,
0.7251362751532665,
0.7019299008674348,
0.5775363990533194,
0.5081651104741776,
0.5618296038078799,
0.5672350542624158,
0.5101743755490761,
0.4881653596292917,
0.5235061188022729,
0.614917644621685,
0.5169578792647075,
0.5323247583159872,
0.6908064978853429,
0.4799689509951395,
0.5967365659085894,
0.4843626165245724,
0.4908164897765076,
0.5258182377843336,
0.4957212615948518,
0.506479383154484,

0.5950950934330866,
0.5273445824603482,
0.5323710372926517,
0.482715716632698,
0.5433187263116123,
0.4967193468125985,
0.49429634163402153,
0.5109766759018295,
0.5481053830603957,
0.5469211782174017,
0.48070737296390953,
0.5235592108878986,
0.5505402336262812,
0.5582939588595737,
0.49061645885825506,
0.521669144791335,
0.5126582535846168,
0.5291579338979644,
0.5306825469606864,
0.5710226770691181,
0.4845111725580885,
0.5271638476701082,
0.5189020174413874,
0.48185406522274443,
0.5016105613691123,
0.6227483740668026,
0.49501481909315215,
0.4830656428945352,
0.47897723026765626,
0.481488095077632,
0.583266200733983,
0.48198977839350915,
0.7567439483679103,
0.5637159769626263,
0.6984866745869153,
0.5638530000802006,
0.6551052450212803,
0.8324837286960383,
0.6268366388451895,
0.7239895277849292,
0.6468246627652733,
0.8623649887727045,
0.5840781334376847,
0.575453313499803,
0.6364550641817576,
0.6103231078029823,
0.6769925761573858,

```
 0.6541499193921613,
 0.5650963250651216,
 0.9465001780727916,
 0.9309819244845031,
 0.6064221325208713,
 0.7091305772862274,
 0.5796567633675715,
 0.850884158982752,
 0.5386455610537654,
 0.6574550942048741,
 0.688914302380367,
 0.5233275370838055,
 0.5199201529019587,
 0.6215288231468222,
 0.655331630602641 9,
 0.7369480863175698,
 0.9287612059115986,
 0.6441644084339043,
 0.5145008916706876,
 0.5651677134853442,
 0.8455145710401315,
 0.7135845757649312,
 0.5630746907820452,
 0.5173992289641393,
 0.6443354936684232,
 0.7067663022500497,
 0.6786485848976934,
 0.5637159769626263,
 0.7129123387068474,
 0.753332366116419,
 0.660506881120318,
 0.5869741584015645,
 0.577807201128253,
 0.676248540889586,
 0.5317698552238226]
```

[10]:
```
first_clusters = []
for i in range(m):
    temp_arr = []
    for j in range(m):
        if d[i][j]<avg_dissimilarity[i]:
            temp_arr.append(j)
    first_clusters.append(temp_arr)
```

[11]:
```
df3 = pd.DataFrame(first_clusters)
df3.to_csv('op.csv', index=False)
```

[12]:
```
def cluster_reducton(clusters):
    final_clusters = []
    for cluster in clusters:
        is_subset = False
        for existing_cluster in final_clusters:
            if set(cluster).issubset(existing_cluster):
                is_subset = True
                break;
        if not is_subset:
            final_clusters.append(cluster)
    for first_cluster in final_clusters:
        for second_cluster in final_clusters:
            if set(second_cluster).issubset(first_cluster):
                final_clusters.remove(second_cluster)
    return final_clusters
```

[13]:
```
def similarity(clusters):
    p = len(clusters)
    similarity_matrix = [[0.0] *p for _ in range(p)]
    for i in range(p):
        for j in range(i, p):
            intersection = len(set(clusters[i]) & set(clusters[j]))
            union = len(set(clusters[i]) | set(clusters[j]))
            similarity_matrix[i][j] = intersection/union
            similarity_matrix[j][i] = intersection/union
    return similarity_matrix
```

[14]:
```
def get_joining_clusters(similarity_matrix):
    p = len(similarity_matrix)
    max = 0.0
    l_index = 0
    r_index = 0
    for i in range(p):
        for j in range(i, p):
            value = similarity_matrix[i][j]
            if i != j and value > max:
                max = similarity_matrix[i][j]
                l_index = i
                r_index = j
    return l_index, r_index
```

[15]:
```
def unite(clusters, i, j):
    temp_clusters = clusters
    temp_cluster_i = clusters[i]
    temp_cluster_j = clusters[i]
    temp_clusters.append(list(set(temp_cluster_i).union(temp_cluster_j)))
    del temp_clusters[i]
```

```
    del temp_clusters[j]
    return temp_clusters
```

[16]:
```
def final_test(first_clusters):
    temp_clusters = first_clusters
    for i in range(2):
        clusters = cluster_reducton(temp_clusters)
        similarity_matrix = similarity(clusters)
        i, j= get_joining_clusters(similarity_matrix)
        final_clusters = unite(clusters, i, j)
        temp_clusters = final_clusters
    return temp_clusters


semi_final_clusters = final_test(first_clusters)

df5 = pd.DataFrame(semi_final_clusters)
df5
```

[16]:
```
    0   1   2   3   4   5   6   7   8   9  ...     84     85     86     87  \
0   0   1   2   3   4   5   6   7   8   9  ...    NaN    NaN    NaN    NaN
1  23  51  53  54  55  56  57  58  59  60  ...    NaN    NaN    NaN    NaN
2  50  51  52  53  54  55  56  57  58  59  ...  139.0  140.0  141.0  142.0
3  50  51  52  53  54  55  56  58  59  61  ...  140.0  141.0  142.0  143.0
4  23  50  51  52  53  54  55  56  57  58  ...    NaN    NaN    NaN    NaN

      88     89     90     91     92     93
0    NaN    NaN    NaN    NaN    NaN    NaN
1    NaN    NaN    NaN    NaN    NaN    NaN
2  143.0  145.0  146.0  147.0  148.0  149.0
3  144.0  145.0  146.0  147.0  148.0  149.0
4    NaN    NaN    NaN    NaN    NaN    NaN

[5 rows x 94 columns]
```

[17]:
```
def remove_overlap(clusters):
    for  i in range(len(clusters)):
        for obj in clusters[i]:
            current_sim = 1/len(clusters[i])
            temp_sim = [0.0] * len(clusters)
            j = (i+1)%len(clusters)
            while j != i:
                if obj in clusters[j]:
                    temp_sim[j] = 1/len(clusters[j])
                    clusters[j].remove(obj);
                j = (j+1)%len(clusters)
            max_sim = max(temp_sim)
```

```
            index = temp_sim.index(max_sim)
            if max_sim>current_sim:
                clusters[index].append(obj)
                clusters[i].remove(obj)
    return clusters
final_clusters = remove_overlap(semi_final_clusters)

df6 = pd.DataFrame(semi_final_clusters)
df6
```

[17]:

```
    0    1    2    3    4    5    6    7    8    9  …    40    41    42  \
0   0    1    2    3    4    5    6    7    8    9  …  41.0  42.0  43.0
1  51   56   60   64   68   72   75   78   83   86  …   NaN   NaN   NaN
2  88  100  102  104  105  110  112  114  120  124  …   NaN   NaN   NaN
3  52   74   88   98  112  116  118  130  135  140  …   NaN   NaN   NaN
4  88   59   67   80   89   94   99   54   57   61  …   NaN   NaN   NaN

     43    44    45    46    47    48    49
0  44.0  45.0  46.0  47.0  48.0  49.0  88.0
1   NaN   NaN   NaN   NaN   NaN   NaN   NaN
2   NaN   NaN   NaN   NaN   NaN   NaN   NaN
3   NaN   NaN   NaN   NaN   NaN   NaN   NaN
4   NaN   NaN   NaN   NaN   NaN   NaN   NaN

[5 rows x 50 columns]
```

[18]:

```
df6.to_csv('op.csv', index=False)
```

[19]:

```
count=0
for cluster in final_clusters:
    for obj in cluster:
        count+=1
print(count)
```

```
153
```

[ ]: