

## K-Means Clustering

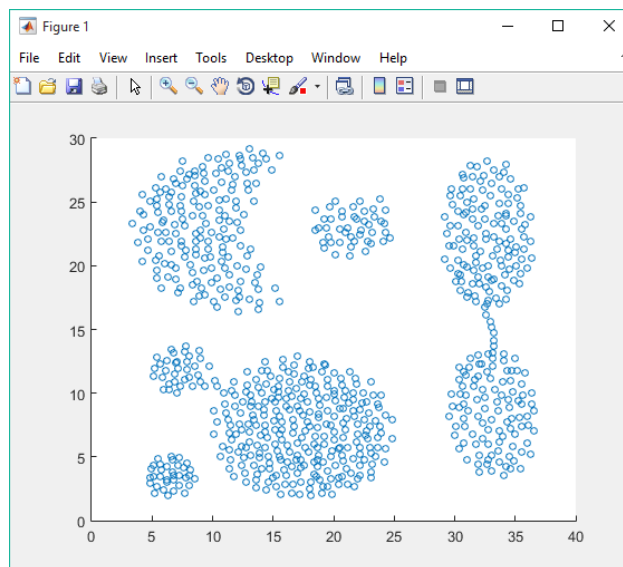
### 1. Partitional Clustering

#### a. Load Data & Visualize

- kMeansMain.m

```
Editor - C:\Users\Zero-Inside\Documents\Machine Learning\Assignment 4.1 K-Means\kMeansMain.m
kMeansMain.m
1 %%Load data
2 aggregation = csvread('Clustering_DataSets/K-means/Aggregation.csv');
3 X = aggregation(:,1:2);
4
5 %%Visualize
6 scatter(X(:,1),X(:,2),20);
```

- Output



#### Analisis Program :

Pada tugas ini saya menggunakan dataset aggregation dengan nama file Aggregation.csv. kemudian meload data tersebut ke dalam variable X untuk ditampilkan. Menggunakan fungsi scatter menghasilkan visualisasi seperti output tanpa label class.

#### b. Apply K-Means

##### i. kMeans Function

- kMeans.m

```
1 function [ finalCentroids result ] = kMeans( X , startCentroids )
2 % this function to do clustering with kMeans
3 % X is data training without label
4 % startCentroids is the variable to defined random position
5 % finalCentroids is the final center of data on every cluster
6 % result is the data training with the cluster label on it.
7
8 [ numRows numCol ] = size(X);
9 [ numK numkCol ] = size(startCentroids);
10 change = true;
11 i = 0;
12 tempCentroids = startCentroids; %save the centroids temporary
13 result = X;
14 finalCentroids = startCentroids;
15 listSSE = [];
16
17 while (change)
18     tempCentroids = finalCentroids;
19     % Form K clusters by assigning each point to its closest centroid.
20     for i = 1:numRow
21         whoMin = [];
22         for j = 1:numK
23             %Calculate the distance every single data to centroids
24             sub = X(i,:) - finalCentroids(j,:);
25             euclidean = sqrt(sum(sub.^2));
26             % save euclidean to list
27             whoMin = [whoMin; euclidean];
28         end
29         % find the smallest index
30         [~, idx] = min(whoMin);
31
32         % assign label to result
33         result(i,3) = idx;
34     end
35
36     % calculate the new centroids
37     for i = 1:numK
38         % find same label in result
39         condition = result(:, 3) == i;
40         % calculate mean every cluster
41         finalCentroids(i,:) = mean(result(condition,1:2));
42         % disp(finalCentroids);
43     end
44
45     % calculate SSE
46     listSSE = [listSSE ; SSE( result , finalCentroids )];
47
48     % check centroids dont change
49     if ((tempCentroids == finalCentroids))
50         change = false;
51     end
52 end
53
54 %plot(listSSE);
55 end
```

### Analisis Program

Function kMeans dengan parameter input X sebagai data training tanpa label dan startCentroids dimana berisi nilai acak dari centroids pada masing-masing cluster. Dengan langkah berikut menglompokkan data sehingga terbentuk K buah cluster dengan titik centroid dari setiap cluster merupakan titik centroid yang telah dipilih sebelumnya dengan menghitung jarak terdekat setiap data point ke setiap centroid, kemudian memberi label datapoint tersebut dengan label yang paling kecil,

Perbaharui nilai titik centroid dengan menghitung rata-rata dari setiap cluster. Ulangi langkah tersebut sampai centroid tidak lagi berubah.

ii. SSE Function

- SSE.m

```
n*
1 function resultSSE = SSE( result , finalCentroids )
2 % calculate the SSE of the final Centroids to every data points
3 % result is data testing
4 % finalCentroids is final centroids after calculate the means
5 % resultSSE is the measure total of distance, smaller is better
6
7 listSSE = [];
8 for i = 1:length(finalCentroids)
9 % find same label in result
10 condition = result(:, 3) == i;
11 dataCondition = result(condition,1:2);
12
13 for j=1:length(dataCondition)
14 sub = dataCondition(j,:) - finalCentroids(i,:);
15 euclidean = sqrt(sum(sub.^2));
16 end
17 listSSE = [listSSE ; euclidean];
18 end
19 % calculate total of every SSE
20 resultSSE = sum(listSSE);
21
22
23 end
24
```

- Analisis Program

Fungsi SSE dengan parameter input berupa result sebagai data yang akan dihitung SSEnya dan finalcentroid sebagai centroid baru hasil proses k-means. Perhitungan SSE dilakukan pada setiap class dan pada class tersebut semua data dihitung jarak euclidean dari centroid itu class itu sendiri. Setelah mendapatkan jarak setiap class antar masing-masing centroidnya kemudian dijumlahkan dengan fungsi sum. Fungsi ini akan mengembalikan nilai SSE yang akan digunakan untuk analisis performa k-means.

c. Run K-Means Algorithm

- kMeansMain.m

```
8 %% makeRandomCentroid
9 [numRow , numCol] = size(X);
10 k = 7; % number of cluster according to number of class
11 startCentroids = zeros(k,2);
12 % Randomly select K examples to be initial centroids
13 startCentroids = X(randperm(numRow, k), :);
14
15 %% do kMeans
16 [ finalCentroids result ] = kMeans( X , startCentroids);
17
```

### Analisis Program

Untuk menjalankan fungsi kMeans, terlebih dahulu menentukan centroids awal berupa random data points dari dataset dan banyak cluster yaitu K.

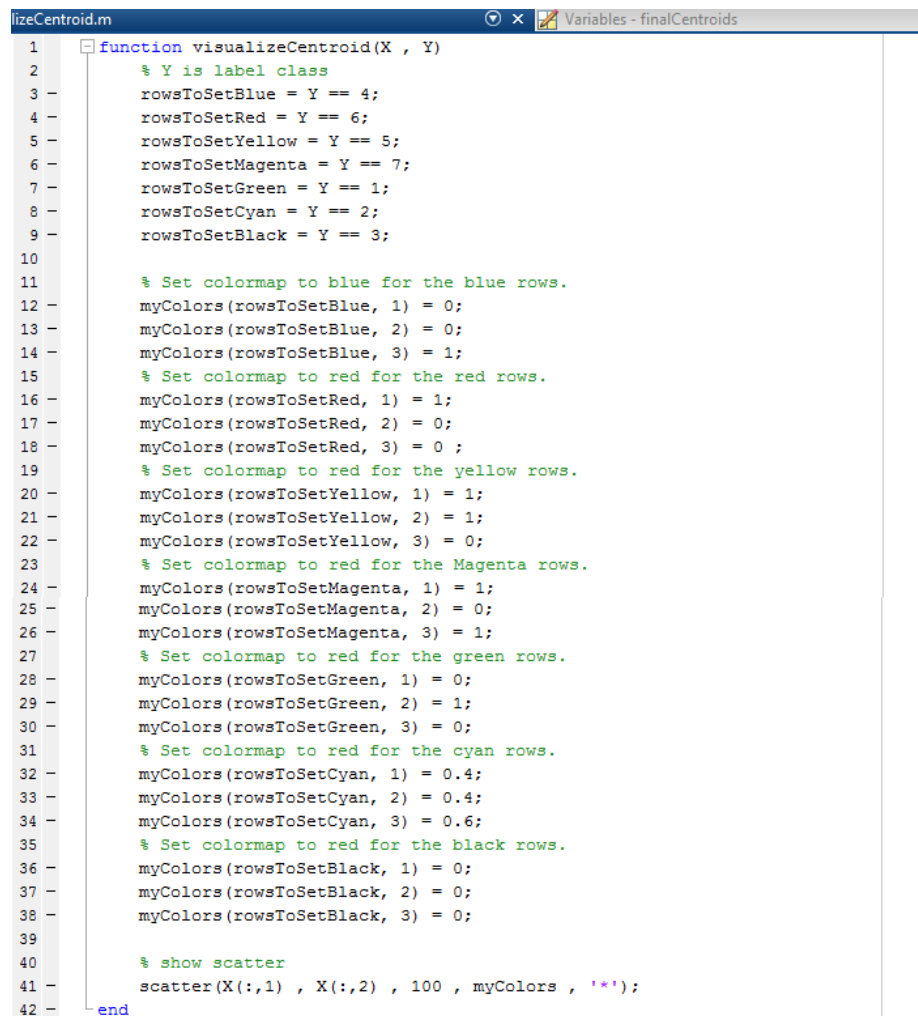
- kMeansMain.m

```
18 %% Visualize the centroids and Result
19 % Visualize the centroids
20 label = [1:length(finalCentroids)]';
21 visualizeCentroid(finalCentroids , label)
22 hold on;
23 % Visualize Result
24 visualize(result, result(:,3));
```

### Analisis Program

Baris ke-20 untuk memberi label dari 1 sampai 7 (sesuai dengan banyak cluster). Baris 21 memvisualisasikan final centroids dengan scatter. Pada baris 24 memvisualisasikan result hasil dari kMeans.

- visualizeCentroid.m



```
function visualizeCentroid(X , Y)
% Y is label class
rowsToSetBlue = Y == 4;
rowsToSetRed = Y == 6;
rowsToSetYellow = Y == 5;
rowsToSetMagenta = Y == 7;
rowsToSetGreen = Y == 1;
rowsToSetCyan = Y == 2;
rowsToSetBlack = Y == 3;

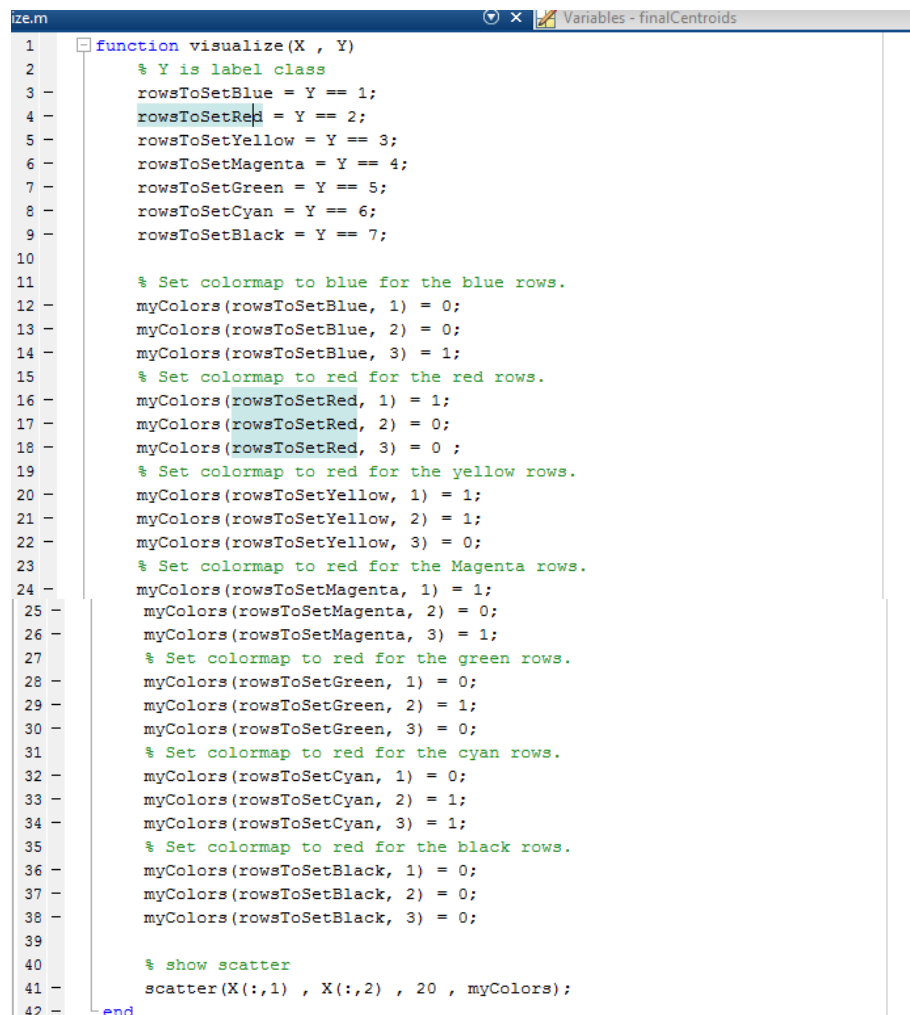
% Set colormap to blue for the blue rows.
myColors(rowsToSetBlue, 1) = 0;
myColors(rowsToSetBlue, 2) = 0;
myColors(rowsToSetBlue, 3) = 1;
% Set colormap to red for the red rows.
myColors(rowsToSetRed, 1) = 1;
myColors(rowsToSetRed, 2) = 0;
myColors(rowsToSetRed, 3) = 0;
% Set colormap to red for the yellow rows.
myColors(rowsToSetYellow, 1) = 1;
myColors(rowsToSetYellow, 2) = 1;
myColors(rowsToSetYellow, 3) = 0;
% Set colormap to red for the Magenta rows.
myColors(rowsToSetMagenta, 1) = 1;
myColors(rowsToSetMagenta, 2) = 0;
myColors(rowsToSetMagenta, 3) = 1;
% Set colormap to red for the green rows.
myColors(rowsToSetGreen, 1) = 0;
myColors(rowsToSetGreen, 2) = 1;
myColors(rowsToSetGreen, 3) = 0;
% Set colormap to red for the cyan rows.
myColors(rowsToSetCyan, 1) = 0.4;
myColors(rowsToSetCyan, 2) = 0.4;
myColors(rowsToSetCyan, 3) = 0.6;
% Set colormap to red for the black rows.
myColors(rowsToSetBlack, 1) = 0;
myColors(rowsToSetBlack, 2) = 0;
myColors(rowsToSetBlack, 3) = 0;

% show scatter
scatter(X(:,1) , X(:,2) , 100 , myColors , '*');
end
```

## Analisis Program

Dengan menggunakan scatter, memvisualisasikan dataset lebih mudah. Terlebih dahulu meload dataset dan membagi data train dan data class label. Kemudian memanggil fungsi visualize dengan parameter data dan label. Pada fungsi visualize(x,y) masing-masing data label class disesuaikan dengan data warna RGB. Kemudian memanggil scatter(param1,param2,param3,param4) dengan param1 dan param2 adalah kolom pertama dari data train dan kolom kedua dari data train, kemudian param3 adalah size dari ukuran plot, dan param4 adalah data color.

- visualize.m

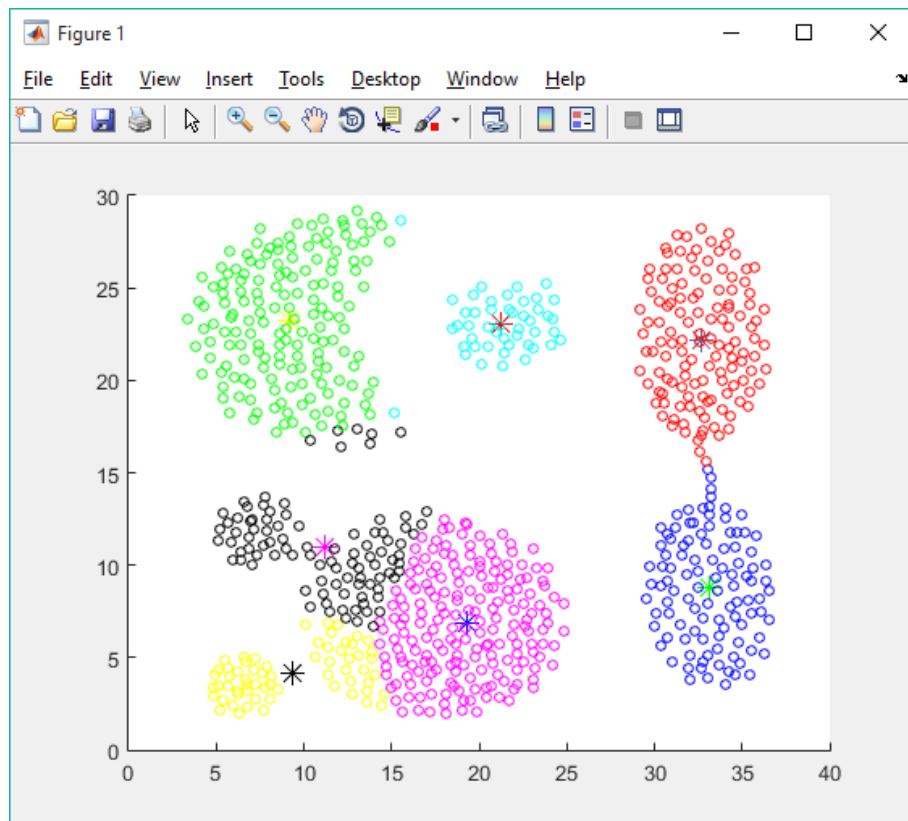


```
1 function visualize(X , Y)
2     % Y is label class
3     rowsToSetBlue = Y == 1;
4     rowsToSetRed = Y == 2;
5     rowsToSetYellow = Y == 3;
6     rowsToSetMagenta = Y == 4;
7     rowsToSetGreen = Y == 5;
8     rowsToSetCyan = Y == 6;
9     rowsToSetBlack = Y == 7;
10
11     % Set colormap to blue for the blue rows.
12     myColors(rowsToSetBlue, 1) = 0;
13     myColors(rowsToSetBlue, 2) = 0;
14     myColors(rowsToSetBlue, 3) = 1;
15     % Set colormap to red for the red rows.
16     myColors(rowsToSetRed, 1) = 1;
17     myColors(rowsToSetRed, 2) = 0;
18     myColors(rowsToSetRed, 3) = 0;
19     % Set colormap to red for the yellow rows.
20     myColors(rowsToSetYellow, 1) = 1;
21     myColors(rowsToSetYellow, 2) = 1;
22     myColors(rowsToSetYellow, 3) = 0;
23     % Set colormap to red for the Magenta rows.
24     myColors(rowsToSetMagenta, 1) = 1;
25     myColors(rowsToSetMagenta, 2) = 0;
26     myColors(rowsToSetMagenta, 3) = 1;
27     % Set colormap to red for the green rows.
28     myColors(rowsToSetGreen, 1) = 0;
29     myColors(rowsToSetGreen, 2) = 1;
30     myColors(rowsToSetGreen, 3) = 0;
31     % Set colormap to red for the cyan rows.
32     myColors(rowsToSetCyan, 1) = 0;
33     myColors(rowsToSetCyan, 2) = 1;
34     myColors(rowsToSetCyan, 3) = 1;
35     % Set colormap to red for the black rows.
36     myColors(rowsToSetBlack, 1) = 0;
37     myColors(rowsToSetBlack, 2) = 0;
38     myColors(rowsToSetBlack, 3) = 0;
39
40     % show scatter
41     scatter(X(:,1) , X(:,2) , 20 , myColors);
42 end
```

## Analisis Program

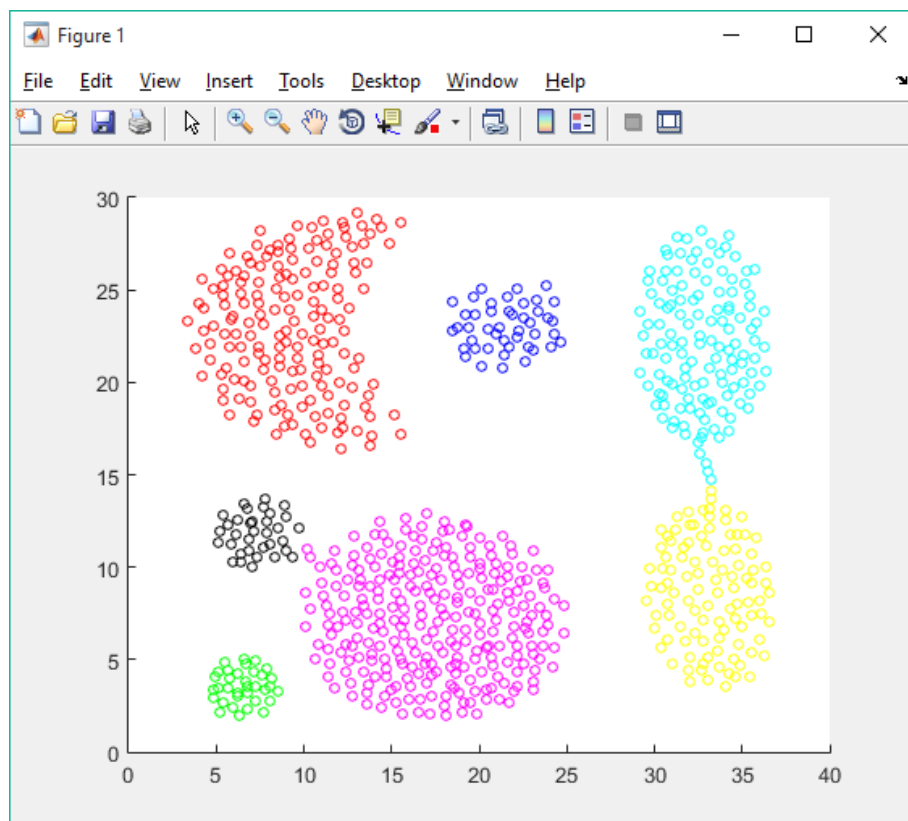
Sama seperti visualizeCentroids.

- Output



d. Visualize from training data.

- Output

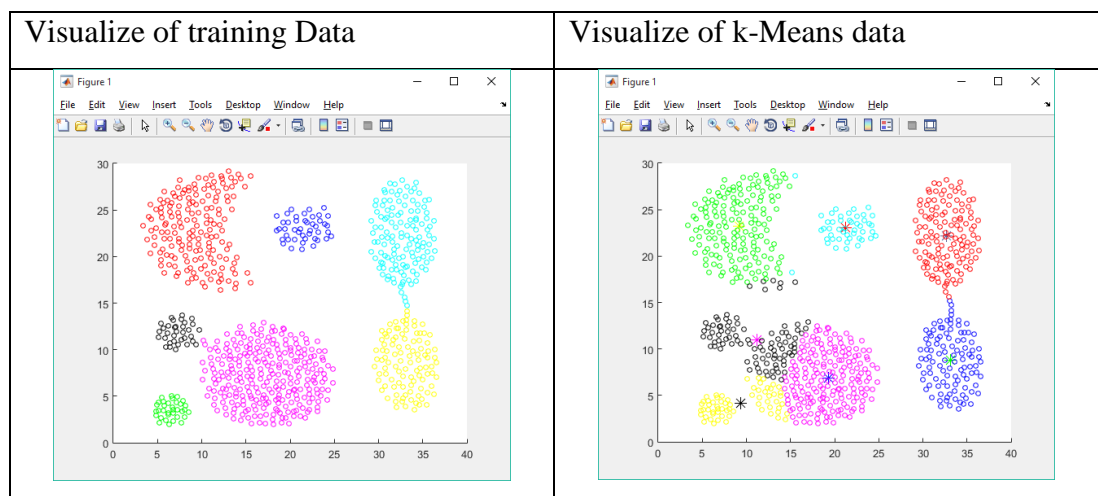


```
Command Window
>> visualize(aggregation(:,1:2),aggregation(:,3))
fx >> |
```

### Analisis Program

Dengan menjalankan visualize ditampilkan scatter plot terhadap data training dengan memasukkan label data asli.

#### e. Compare Result



### Analisis

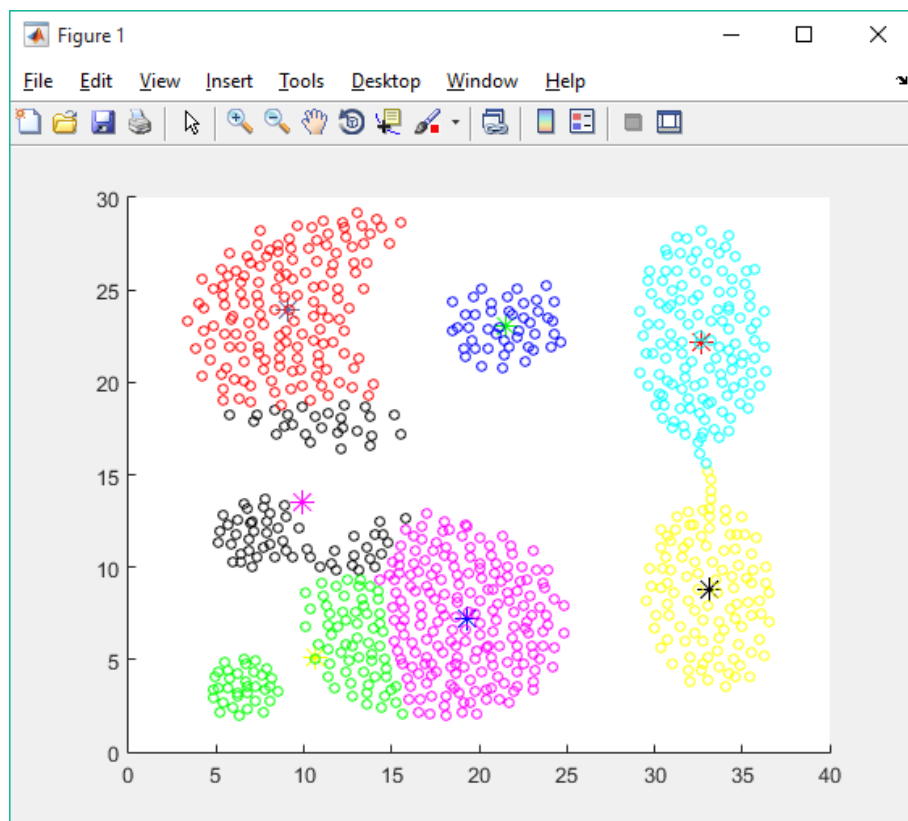
Hasil dari algoritma k-Means yang dijalankan terhadap data training tanpa label akan menghasilkan cluster(kelompok) data seperti table pada kolom ke 2. Terlihat bahwa ada beberapa data yang dikelompokkan melewati class yang seharusnya. Kemudian beberapa warna class juga ada yang berbeda. Karena initial centroids acak yang mungkin bernilai dengan class terdekat sehingga warna class berubah.

#### f. Do K-Means with initial centroids on each class and visualize it.

- kMeansMain.m

```
27 %% makeRandomCentroid from class
28 k = 7; % number of cluster according to number of class
29 startCentroids = zeros(k,2);
30 % Randomly select K examples to be initial centroids in the class
31 for i = 1 : k
32     condition = aggregation(:,3) == i;
33     filterX = X(condition,:);
34     [numRow , numCol] = size(filterX);
35     startCentroids(i,:) = filterX(randi([1 numRow]), :);
36 end
37
38 %% do kMeans
39 [ finalCentroids result ] = kMeans( X , startCentroids);
40
41 %% Visualize the centroids and Result
42 % Visualize the centroids
43 label = [1:length(finalCentroids)]';
44 visualizeCentroid(finalCentroids , label)
45 hold on;
46 % Visualize Result
47 visualize(result, result(:,3));
48
```

- Output

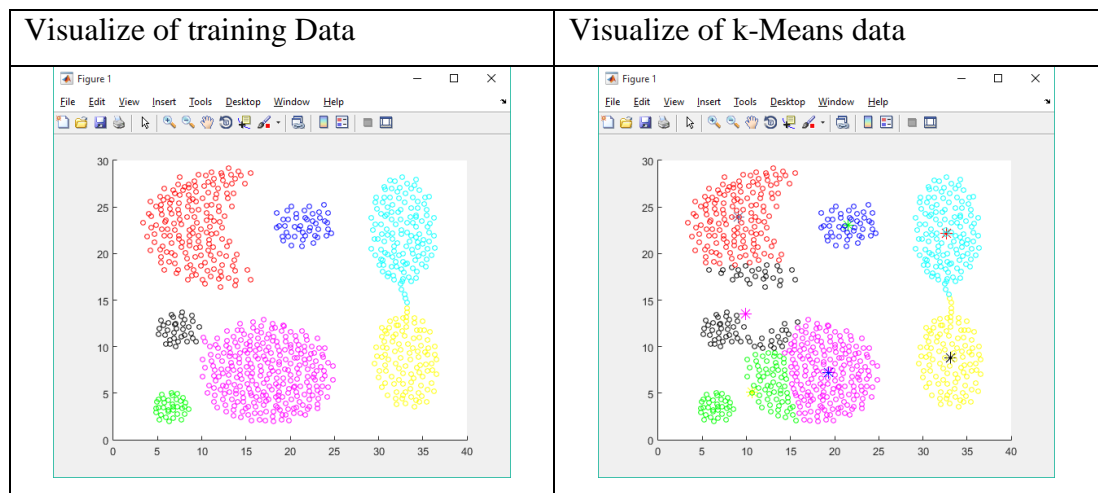


#### Analisis Program

Baris 31 sampai 36 akan menginisialisasi start centroids dengan masing-masing class. Untuk setiap class dipilih secara random datapoint yang ada. Kemudian melakukan kMeans dan menampilkannya dengan scatter.

#### g. Compare result

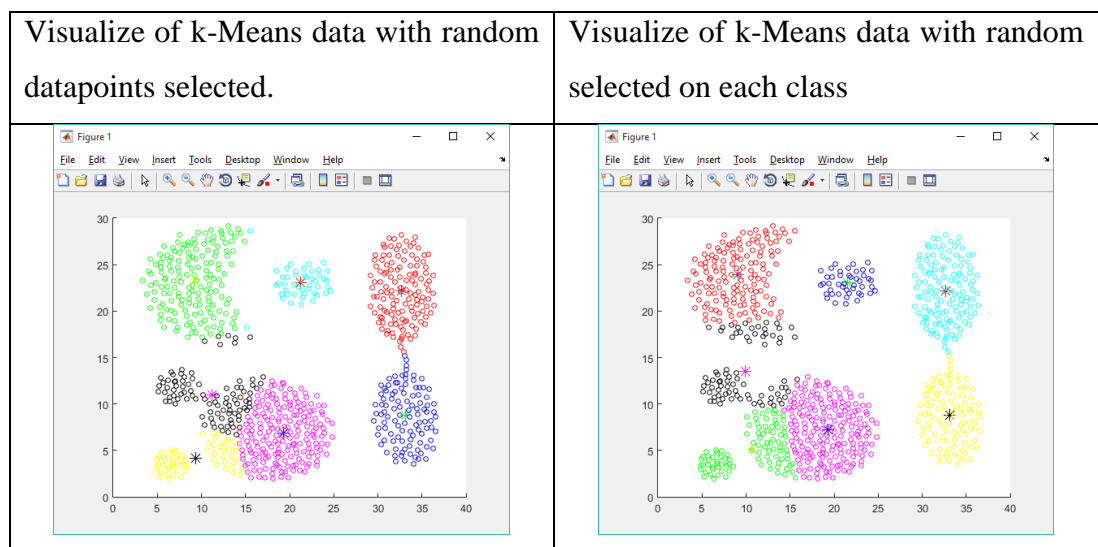




### Analisis

Dengan menjalankan visualize dari data yang sudah dilakukan algoritma K-Means didapat hasil sesuai kolom ke-2 pada table. Diperoleh data class yang sesuai dengan yang asli tetapi ada missclassification pada kelompok data yang kecil. Seperti warna hitam dan warna hijau. Karena data yang ungu dan merah tersebar luas, maka titik centroids pada warna hitam dan warna hijau akan menganggap data-points terdekat dari mereka adalah bagian dari class yang tersebar luas karena melihat jarak terdekat dari antar cluster.

- h. Compare result with random datapoints selected and random selected on each class as initial centroids.



### Analisis

Hasil yang diperoleh terlihat perbedaan warna antar cluster, cluster yang menggunakan initial centroids pada random data sesuai dengan classnya akan sesuai dengan hasil warna pada training data dan jika initial centroids menggunakan random datapoints dari seluruh data training maka tidak menjamin warna yang akan dihasilkan sama dengan training data beserta labelnya karena ada unsur acak tersebut. Selain warna, lebih akurat memilih initial centroids dari setiap class karena initial centroids menentukan performansi K-Means untuk mengclusterkan dengan tepat setiap datapoints terhadap centroidsnya lebih dekat dengan kelompok classnya maka dengan mudah dikelompokkan sesuai classnya.