Tehran Polytechnic University

Computer Engineering Department

Data Mining

Assignment Two

Name: Mohammad Hossein Badiei

Student ID: 9531701

Majors: Artificial Intelligence and Robotics (Amirkabir) | Electrical Engineering (Tehran)

Instructor: Dr. Ehsan Nazerafard

Spring 2021

پاسخ سوال 1

الف)

مکان داده‌ها طبق صورت سوال به صورت زیر است.

	x	*y*
A_1	1	2
A_2	6	3
A_3	8	4
A_4	2	5
A_5	7	5
A_6	4	6
A_7	5	7
A_8	2	8

سوال سه خوشه را در حالت اولیه به صورت زیر در نظر گرفته است:

خوشه اول	A_3 , A_4 , A_8
خوشه دوم	A_2 , A_5 , A_7
خوشه سوم	A_1 , A_6

حال الگوریتم k-means را که سودو کد آن به صورت زیر است در نظر می‌گیریم و سپس مراکز خوشه ها را بدست می‌اوریم.
دقت بفرمایید که k تعداد خوشه‌ها می‌باشد که برابر با 3 است و D نیز برابر با $\{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\}$ می‌باشد.

```
Input:
    D= {t1, t2, …. Tn  }   // Set of elements
    K                      // Number of desired clusters
Output:
    K                      // Set of clusters
K-Means algorithm:
   Assign initial values for m1, m2,…. mk
   repeat
     assign each item tj    to the clusters which has the closest mean;
     calculate new mean for each cluster;
   until convergence criteria is met;
```

حال مراکز خوشه‌ها را تعیین کرده و خوشه‌ی جدید را بدست می‌آوریم. پاسخ به صورت زیر می‌باشد.

مرحله اول

$$C_1 = \left(\frac{x_{A_3} + x_{A_4} + x_{A_8}}{3}, \frac{y_{A_3} + y_{A_4} + y_{A_8}}{3}\right) = \left(\frac{8+2+2}{3}, \frac{4+5+8}{3}\right) = \left(4, \frac{17}{3}\right)$$

$$C_2 = \left(\frac{x_{A_2} + x_{A_5} + x_{A_7}}{3}, \frac{y_{A_2} + y_{A_5} + y_{A_7}}{3}\right) = \left(\frac{6+7+5}{3}, \frac{3+5+7}{3}\right) = (6, 5)$$

$$C_3 = \left(\frac{x_{A_1} + x_{A_6}}{2}, \frac{y_{A_1} + y_{A_6}}{2}\right) = \left(\frac{1+4}{2}, \frac{2+6}{2}\right) = \left(\frac{5}{2}, 4\right)$$

حال فاصله هر یک از نقاط را از هر یک از مراکز خوشه‌ها بدست می‌آوریم و خوشه های جدید را یافته و مراکز جدید را می یابیم.

$$|A_1 - C_1| = \sqrt{(1-4)^2 + (2-\frac{17}{3})^2} = \frac{\sqrt{202}}{3} \simeq 4.73$$

$$|A_1 - C_2| = \sqrt{(1-6)^2 + (2-5)^2} = \sqrt{34} \simeq 5.83$$

$$|A_1 - C_3| = \sqrt{(1-\frac{5}{2})^2 + (2-4)^2} = \frac{5}{2} \simeq 2.5$$

A_1 به C_3 نزدیکتر است پس آن را در خوشه سوم قرار می‌دهیم.

$$|A_2 - C_1| = \sqrt{(6-4)^2 + (3-\frac{17}{3})^2} = \frac{10}{3} \simeq 3.33$$

$$|A_2 - C_2| = \sqrt{(6-6)^2 + (3-5)^2} = 2$$

$$|A_2 - C_3| = \sqrt{(6-\frac{5}{2})^2 + (3-4)^2} = \frac{\sqrt{53}}{2} \simeq 3.64$$

A_2 به C_2 نزدیکتر است پس آن را در خوشه دوم قرار می‌دهیم.

$$|A_3 - C_1| = \sqrt{(8-4)^2 + (4-\frac{17}{3})^2} = \frac{13}{3} \simeq 4.33$$

$$|A_3 - C_2| = \sqrt{(8-6)^2 + (4-5)^2} = \sqrt{5} \simeq 2.23$$

$$|A_3 - C_3| = \sqrt{(8 - \frac{5}{2})^2 + (4 - 4)^2} = \frac{11}{2} = 5.5$$

A_3 به C_2 نزدیکتر است پس آن را در خوشه دوم قرار می‌دهیم.

$$|A_4 - C_1| = \sqrt{(2 - 4)^2 + (5 - \frac{17}{3})^2} = \frac{2\sqrt{10}}{3} \simeq 2.1$$

$$|A_4 - C_2| = \sqrt{(2 - 6)^2 + (5 - 5)^2} = 4$$

$$|A_4 - C_3| = \sqrt{(2 - \frac{5}{2})^2 + (5 - 4)^2} = \frac{\sqrt{5}}{2} \simeq 1.12$$

A_4 به C_3 نزدیکتر است پس آن را در خوشه سوم قرار می‌دهیم.

$$|A_5 - C_1| = \sqrt{(7 - 4)^2 + (5 - \frac{17}{3})^2} = \frac{\sqrt{85}}{3} \simeq 3.07$$

$$|A_5 - C_2| = \sqrt{(7 - 6)^2 + (5 - 5)^2} = 1$$

$$|A_5 - C_3| = \sqrt{(7 - \frac{5}{2})^2 + (5 - 4)^2} = \frac{\sqrt{85}}{2} \simeq 4.62$$

A_5 به C_2 نزدیکتر است پس آن را در خوشه دوم قرار می‌دهیم.

$$|A_6 - C_1| = \sqrt{(4 - 4)^2 + (6 - \frac{17}{3})^2} = \frac{1}{3} \simeq 0.33$$

$$|A_6 - C_2| = \sqrt{(4 - 6)^2 + (6 - 5)^2} = \sqrt{5} \simeq 2.23$$

$$|A_6 - C_3| = \sqrt{(4 - \frac{5}{2})^2 + (6 - 4)^2} = \frac{5}{2} = 2.5$$

A_6 به C_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

$$|A_7 - C_1| = \sqrt{(5-4)^2 + (7-\frac{17}{3})^2} = \frac{5}{3} \simeq 1.66$$

$$|A_7 - C_2| = \sqrt{(5-6)^2 + (7-5)^2} = \sqrt{5} \simeq 2.23$$

$$|A_7 - C_3| = \sqrt{(5-\frac{5}{2})^2 + (7-4)^2} = \frac{\sqrt{61}}{2} \simeq 3.9$$

A_7 به C_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

$$|A_8 - C_1| = \sqrt{(2-4)^2 + (8-\frac{17}{3})^2} = \frac{\sqrt{85}}{3} \simeq 3.07$$

$$|A_8 - C_2| = \sqrt{(2-6)^2 + (8-5)^2} = 5$$

$$|A_8 - C_3| = \sqrt{(2-\frac{5}{2})^2 + (8-4)^2} = \frac{\sqrt{65}}{2} \simeq 4.03$$

A_8 به C_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

در نهایت در مرحله اول، جدول خوشه ها به صورت زیر در می اید.

خوشه اول	A_6, A_7, A_8
خوشه دوم	A_2, A_3, A_5
خوشه سوم	A_1, A_4

مرحله دوم

مراکز خوشه های جدید را می‌یابیم.

$$C_1 = \left(\frac{x_{A_6} + x_{A_7} + x_{A_8}}{3}, \frac{y_{A_6} + y_{A_7} + y_{A_8}}{3}\right) = \left(\frac{4+5+2}{3}, \frac{6+7+8}{3}\right) = \left(\frac{11}{3}, 7\right)$$

$$C_2 = \left(\frac{x_{A_2} + x_{A_3} + x_{A_5}}{3}, \frac{y_{A_2} + y_{A_3} + y_{A_5}}{3}\right) = \left(\frac{6+8+7}{3}, \frac{3+4+5}{3}\right) = (7,4)$$

$$C_3 = \left(\frac{x_{A_1} + x_{A_4}}{2}, \frac{y_{A_1} + y_{A_4}}{2}\right) = \left(\frac{1+2}{2}, \frac{2+5}{2}\right) = (\frac{3}{2}, \frac{7}{2})$$

حال فاصله هر یک از نقاط را از هر یک از مراکز خوشه‌ها بدست می‌آوریم و دسته های جدید را یافته و مراکز جدید را می یابیم.

$$|A_1 - C_1| = \sqrt{(1 - \frac{11}{3})^2 + (2-7)^2} = \frac{17}{3} \simeq 5.66$$

$$|A_1 - C_2| = \sqrt{(1-7)^2 + (2-4)^2} = 2\sqrt{10}$$

$$|A_1 - C_3| = \sqrt{(1 - \frac{3}{2})^2 + (2 - \frac{7}{2})^2} = \frac{\sqrt{10}}{2} = 1.58$$

A_1 به C_3 نزدیکتر است پس آن را در خوشه سوم قرار می‌دهیم.

$$|A_2 - C_1| = \sqrt{(6 - \frac{11}{3})^2 + (3-7)^2} = \frac{\sqrt{193}}{3} \simeq 4.63$$

$$|A_2 - C_2| = \sqrt{(6-7)^2 + (3-4)^2} = \sqrt{2} \simeq 1.41$$

$$|A_2 - C_3| = \sqrt{(6 - \frac{3}{2})^2 + (3 - \frac{7}{2})^2} = \frac{\sqrt{82}}{2} \simeq 4.53$$

A_2 به C_2 نزدیکتر است پس آن را در خوشه دوم قرار می‌دهیم.

$$|A_3 - C_1| = \sqrt{(8 - \frac{11}{3})^2 + (4-7)^2} = \frac{5\sqrt{10}}{3} \simeq 5.27$$

$$|A_3 - C_2| = \sqrt{(8-7)^2 + (4-4)^2} = 1$$

$$|A_3 - C_3| = \sqrt{(8 - \frac{3}{2})^2 + (4 - \frac{7}{2})^2} = \frac{\sqrt{170}}{2} \simeq 6.52$$

A_3 به C_2 نزدیکتر است پس آن را در خوشه دوم قرار می‌دهیم.

$$|A_4 - C_1| = \sqrt{(2 - \frac{11}{3})^2 + (5-7)^2} = \frac{\sqrt{61}}{3} \simeq 2.6$$

$$|A_4 - C_2| = \sqrt{(2-7)^2 + (5-4)^2} = \sqrt{26} \simeq 5.1$$

$$|A_4 - C_3| = \sqrt{(2-\frac{3}{2})^2 + (5-\frac{7}{2})^2} = \frac{\sqrt{10}}{2} \simeq 1.58$$

A_4 به C_3 نزدیکتر است پس آن را در خوشه سوم قرار می‌دهیم.

$$|A_5 - C_1| = \sqrt{(7-\frac{11}{3})^2 + (5-7)^2} = \frac{2\sqrt{34}}{3} \simeq 3.89$$

$$|A_5 - C_2| = \sqrt{(7-7)^2 + (5-4)^2} = 1$$

$$|A_5 - C_3| = \sqrt{(7-\frac{3}{2})^2 + (5-\frac{7}{2})^2} = \frac{\sqrt{130}}{2} \simeq 5.7$$

A_5 به C_2 نزدیکتر است پس آن را در خوشه دوم قرار می‌دهیم.

$$|A_6 - C_1| = \sqrt{(4-\frac{11}{3})^2 + (6-7)^2} = \frac{1}{3} \simeq 1.05$$

$$|A_6 - C_2| = \sqrt{(4-7)^2 + (6-4)^2} = \sqrt{5} \simeq 3.6$$

$$|A_6 - C_3| = \sqrt{(4-\frac{3}{2})^2 + (6-\frac{7}{2})^2} = \frac{5\sqrt{2}}{2} = 3.53$$

A_6 به C_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

$$|A_7 - C_1| = \sqrt{(5-\frac{11}{3})^2 + (7-7)^2} = \frac{4}{3} \simeq 1.33$$

$$|A_7 - C_2| = \sqrt{(5-7)^2 + (7-4)^2} = \sqrt{13} \simeq 3.6$$

$$|A_7 - C_3| = \sqrt{(5-\frac{3}{2})^2 + (7-\frac{7}{2})^2} = \frac{7\sqrt{2}}{2} \simeq 4.95$$

A_7 به C_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

$$|A_8 - C_1| = \sqrt{(2 - \frac{11}{3})^2 + (8-7)^2} = \frac{\sqrt{34}}{3} \simeq 1.94$$

$$|A_8 - C_2| = \sqrt{(2-7)^2 + (8-4)^2} = \sqrt{41} \simeq 6.4$$

$$|A_8 - C_3| = \sqrt{(2 - \frac{3}{2})^2 + (8 - \frac{7}{2})^2} = \frac{\sqrt{82}}{2} \simeq 4.53$$

A_8 به C_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

در نهایت در مرحله دوم، جدول خوشه ها به صورت زیر در می اید.

خوشه اول	A_6, A_7, A_8
خوشه دوم	A_2, A_3, A_5
خوشه سوم	A_1, A_4

مرحله سوم

با توجه به اینکه داده های هر یک از این خوشه ها به خوشه ی دیگری منتقل نشده اند لذا نتایج این مرحله مشابه با مرحله دوم خواهد بود.

مراکز خوشه های جدید را می‌یابیم.

$$C_1 = \left(\frac{x_{A_6} + x_{A_7} + x_{A_8}}{3}, \frac{y_{A_6} + y_{A_7} + y_{A_8}}{3} \right) = \left(\frac{4+5+2}{3}, \frac{6+7+8}{3} \right) = \left(\frac{11}{3}, 7 \right)$$

$$C_2 = \left(\frac{x_{A_2} + x_{A_3} + x_{A_5}}{3}, \frac{y_{A_2} + y_{A_3} + y_{A_5}}{3} \right) = \left(\frac{6+8+7}{3}, \frac{3+4+5}{3} \right) = (7,4)$$

$$C_3 = \left(\frac{x_{A_1} + x_{A_4}}{2}, \frac{y_{A_1} + y_{A_4}}{2} \right) = \left(\frac{1+2}{2}, \frac{2+5}{2} \right) = (\frac{3}{2}, \frac{7}{2})$$

حال فاصله هر یک از نقاط را از هر یک از مراکز خوشه‌ها بدست می‌آوریم و دسته های جدید را یافته و مراکز جدید را می یابیم.

$$|A_1 - C_1| = \sqrt{(1 - \frac{11}{3})^2 + (2-7)^2} = \frac{17}{3} \simeq 5.66$$

$$|A_1 - C_2| = \sqrt{(1-7)^2 + (2-4)^2} = 2\sqrt{10}$$

$$|A_1 - C_3| = \sqrt{(1-\frac{3}{2})^2 + (2-\frac{7}{2})^2} = \frac{\sqrt{10}}{2} = 1.58$$

A_1 به C_3 نزدیکتر است پس آن را در خوشه سوم قرار می‌دهیم.

$$|A_2 - C_1| = \sqrt{(6-\frac{11}{3})^2 + (3-7)^2} = \frac{\sqrt{193}}{3} \simeq 4.63$$

$$|A_2 - C_2| = \sqrt{(6-7)^2 + (3-4)^2} = \sqrt{2} \simeq 1.41$$

$$|A_2 - C_3| = \sqrt{(6-\frac{3}{2})^2 + (3-\frac{7}{2})^2} = \frac{\sqrt{82}}{2} \simeq 4.53$$

A_2 به C_2 نزدیکتر است پس آن را در خوشه دوم قرار می‌دهیم.

$$|A_3 - C_1| = \sqrt{(8-\frac{11}{3})^2 + (4-7)^2} = \frac{5\sqrt{10}}{3} \simeq 5.27$$

$$|A_3 - C_2| = \sqrt{(8-7)^2 + (4-4)^2} = 1$$

$$|A_3 - C_3| = \sqrt{(8-\frac{3}{2})^2 + (4-\frac{7}{2})^2} = \frac{\sqrt{170}}{2} \simeq 6.52$$

A_3 به C_2 نزدیکتر است پس آن را در خوشه دوم قرار می‌دهیم.

$$|A_4 - C_1| = \sqrt{(2-\frac{11}{3})^2 + (5-7)^2} = \frac{\sqrt{61}}{3} \simeq 2.6$$

$$|A_4 - C_2| = \sqrt{(2-7)^2 + (5-4)^2} = \sqrt{26} \simeq 5.1$$

$$|A_4 - C_3| = \sqrt{(2-\frac{3}{2})^2 + (5-\frac{7}{2})^2} = \frac{\sqrt{10}}{2} \simeq 1.58$$

A_4 به C_3 نزدیکتر است پس آن را در خوشه سوم قرار می‌دهیم.

$$|A_5 - C_1| = \sqrt{(7 - \frac{11}{3})^2 + (5-7)^2} = \frac{2\sqrt{34}}{3} \simeq 3.89$$

$$|A_5 - C_2| = \sqrt{(7-7)^2 + (5-4)^2} = 1$$

$$|A_5 - C_3| = \sqrt{(7 - \frac{3}{2})^2 + (5 - \frac{7}{2})^2} = \frac{\sqrt{130}}{2} \simeq 5.7$$

A_5 به C_2 نزدیکتر است پس آن را در خوشه دوم قرار می‌دهیم.

$$|A_6 - C_1| = \sqrt{(4 - \frac{11}{3})^2 + (6-7)^2} = \frac{1}{3} \simeq 1.05$$

$$|A_6 - C_2| = \sqrt{(4-7)^2 + (6-4)^2} = \sqrt{5} \simeq 3.6$$

$$|A_6 - C_3| = \sqrt{(4 - \frac{3}{2})^2 + (6 - \frac{7}{2})^2} = \frac{5\sqrt{2}}{2} = 3.53$$

A_6 به C_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

$$|A_7 - C_1| = \sqrt{(5 - \frac{11}{3})^2 + (7-7)^2} = \frac{4}{3} \simeq 1.33$$

$$|A_7 - C_2| = \sqrt{(5-7)^2 + (7-4)^2} = \sqrt{13} \simeq 3.6$$

$$|A_7 - C_3| = \sqrt{(5 - \frac{3}{2})^2 + (7 - \frac{7}{2})^2} = \frac{7\sqrt{2}}{2} \simeq 4.95$$

A_7 به C_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

$$|A_8 - C_1| = \sqrt{(2 - \frac{11}{3})^2 + (8-7)^2} = \frac{\sqrt{34}}{3} \simeq 1.94$$

$$|A_8 - C_2| = \sqrt{(2-7)^2 + (8-4)^2} = \sqrt{41} \simeq 6.4$$

$$|A_8 - C_3| = \sqrt{(2 - \frac{3}{2})^2 + (8 - \frac{7}{2})^2} = \frac{\sqrt{82}}{2} \simeq 4.53$$

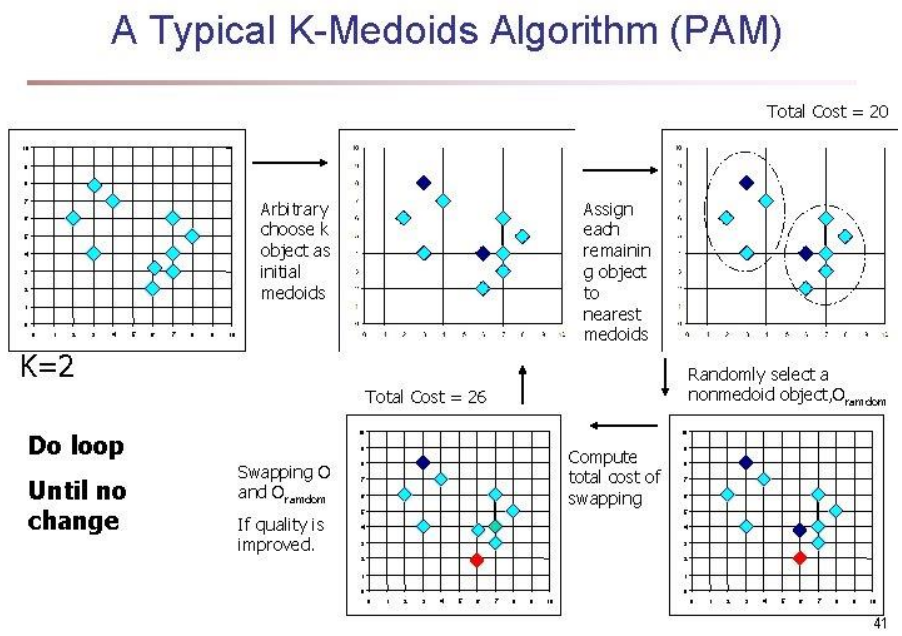
A_8 به C_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

در نهایت در مرحله سوم، جدول خوشه ها به صورت زیر در می اید.

خوشه اول	A_6, A_7, A_8
خوشه دوم	A_2, A_3, A_5
خوشه سوم	A_1, A_4

ب)

قبل از شروع به حل سوال ابتدا الگوریتم را در قالبِ یک شکل نمایش می‌دهیم.



مرحله اول

طبق فرض سوال برای انتخاب medoid ها عمل می کنیم. (medoid خوشه‌ی i ام را با M_i نمایش می‌دهیم)

$M_1 = A_1$

$M_2 = A_2$

$M_3 = A_3$

حال فاصله هر یک از نقاط را از هر یک از medoid خوشه‌ها بدست می‌آوریم و خوشه های جدید را یافته و medoid های جدید را می یابیم سپس با توجه به cost ای که دارند، تصمیم می‌گیریم که خوشه‌ی جدید جایگزین قبلی شود یا خیر.

$$|A_1 - M_1| = \sqrt{(1-1)^2 + (2-2)^2} = 0$$
$$|A_1 - M_2| = \sqrt{(1-6)^2 + (2-3)^2} = \sqrt{26} \simeq 5.1$$
$$|A_1 - M_3| = \sqrt{(1-8)^2 + (2-4)^2} = \sqrt{53} \simeq 7.28$$

A_1 به M_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

$$|A_2 - M_1| = \sqrt{(6-1)^2 + (3-2)^2} = \sqrt{26} \simeq 5.1$$
$$|A_2 - M_2| = \sqrt{(6-6)^2 + (3-3)^2} = 0$$
$$|A_2 - M_3| = \sqrt{(6-8)^2 + (3-4)^2} = \sqrt{5} \simeq 2.23$$

A_2 به M_2 نزدیکتر است پس آن را در خوشه دوم قرار می‌دهیم.

$$|A_3 - M_1| = \sqrt{(8-1)^2 + (4-2)^2} = \sqrt{53} \simeq 7.28$$
$$|A_3 - M_2| = \sqrt{(8-6)^2 + (4-3)^2} = \sqrt{5} \simeq 2.23$$
$$|A_3 - M_3| = \sqrt{(8-8)^2 + (4-4)^2} = 0$$

A_3 به M_3 نزدیکتر است پس آن را در خوشه سوم قرار می‌دهیم.

$$|A_4 - M_1| = \sqrt{(2-1)^2 + (5-2)^2} = \sqrt{10} \simeq 3.16$$
$$|A_4 - M_2| = \sqrt{(2-6)^2 + (5-3)^2} = 2\sqrt{5} \simeq 4.47$$
$$|A_4 - M_3| = \sqrt{(2-8)^2 + (5-4)^2} = \sqrt{37} \simeq 6.08$$

A_4 به M_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

$$|A_5 - M_1| = \sqrt{(7-1)^2 + (5-2)^2} = 3\sqrt{5} \simeq 6.71$$
$$|A_5 - M_2| = \sqrt{(7-6)^2 + (5-3)^2} = \sqrt{5} \simeq 2.24$$
$$|A_5 - M_3| = \sqrt{(7-8)^2 + (5-4)^2} = \sqrt{2} \simeq 1.41$$

A_5 به M_3 نزدیکتر است پس آن را در خوشه سوم قرار می‌دهیم.

$$|A_6 - M_1| = \sqrt{(4-1)^2 + (6-2)^2} = 5$$
$$|A_6 - M_2| = \sqrt{(4-6)^2 + (6-3)^2} = \sqrt{13} \simeq 3.61$$
$$|A_6 - M_3| = \sqrt{(4-8)^2 + (6-4)^2} = 2\sqrt{5} \simeq 4.47$$

A_6 به M_2 نزدیکتر است پس آن را در خوشه دوم قرار می‌دهیم.

$$|A_7 - M_1| = \sqrt{(5-1)^2 + (7-2)^2} = \sqrt{41} \simeq 6.4$$
$$|A_7 - M_2| = \sqrt{(5-6)^2 + (7-3)^2} = \sqrt{17} \simeq 4.12$$
$$|A_7 - M_3| = \sqrt{(5-8)^2 + (7-4)^2} = 3\sqrt{2} \simeq 4.24$$

A_7 به M_2 نزدیکتر است پس آن را در خوشه دوم قرار می‌دهیم.

$$|A_8 - M_1| = \sqrt{(2-1)^2 + (8-2)^2} = \sqrt{37} \simeq 6.08$$
$$|A_8 - M_2| = \sqrt{(2-6)^2 + (8-3)^2} = \sqrt{41} \simeq 6.4$$
$$|A_8 - M_3| = \sqrt{(2-8)^2 + (8-4)^2} = 2\sqrt{13} \simeq 7.21$$

A_8 به M_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

در نهایت در مرحله اول، جدول خوشه ها به صورت زیر در می اید.

خوشه اول	A_1, A_4, A_8
خوشه دوم	A_2, A_6, A_7
خوشه سوم	A_3, A_5

حال cost را بر اساس معیارِ خطایِ SSE در مرحله اول حساب می‌کنیم. (در واقع معیار WSS یعنی جمع SSE های هر خوشه را حساب میکنیم.)

Cost = $SSE(X) = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$

$$\text{Cost} = 0 + 0 + 0 + 10 + 2 + 13 + 17 + 37 = 79$$

پس cost در مرحله ی اول بر اساس معیارِ خطایِ SSE برابر با 79 شد. به سراغِ مرحله ی بعد می‌رویم و در صورتی که cost در مرحله‌ی بعد کمتر از مرحله‌ی فعلی باشد، کلاستر را به کلاسترِ بدست آمده در مرحله‌ی بعد تغییر میدهیم و در غیر اینصورت همین کلاستر را حفظ خواهیم نمود.

مرحله دوم

حال یک نقطه تصادفی دیگر از یکی از کلاسترها را به عنوانِ medoid در نظر گرفته و عملیات مرحله قبل را روی آن اجرا می‌کنیم.

طبق فرضی که در سوال گفته شده است، این نقطه تصادفی را A_4 که از کلاسترِ اول است به عنوان medoid بجای A_1 در نظر می‌گیریم.

$$M_1 = A_4$$

$$M_2 = A_2$$

$$M_3 = A_3$$

حال فاصله هر یک از نقاط را از هر یک از medoid خوشه‌ها بدست می‌آوریم و خوشه های جدید را یافته و medoid های جدید را می یابیم سپس با توجه به cost ای که دارند، تصمیم می‌گیریم که خوشه‌ی جدید جایگزین قبلی شود یا خیر.

$$|A_1 - M_1| = \sqrt{(1-2)^2 + (2-5)^2} = \sqrt{10}$$

$$|A_1 - M_2| = \sqrt{(1-6)^2 + (2-3)^2} = \sqrt{26} \simeq 5.1$$

$$|A_1 - M_3| = \sqrt{(1-8)^2 + (2-4)^2} = \sqrt{53} \simeq 7.28$$

A_1 به M_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

$$|A_2 - M_1| = \sqrt{(6-2)^2 + (3-5)^2} = 2\sqrt{5} \simeq 4.47$$

$$|A_2 - M_2| = \sqrt{(6-6)^2 + (3-3)^2} = 0$$

$$|A_2 - M_3| = \sqrt{(6-8)^2 + (3-4)^2} = \sqrt{5} \simeq 2.23$$

A_2 به M_2 نزدیکتر است پس آن را در خوشه دوم قرار می‌دهیم.

$$|A_3 - M_1| = \sqrt{(8-2)^2 + (4-5)^2} = \sqrt{37} \simeq 6.08$$

$$|A_3 - M_2| = \sqrt{(8-6)^2 + (4-3)^2} = \sqrt{5} \simeq 2.23$$

$$|A_3 - M_3| = \sqrt{(8-8)^2 + (4-4)^2} = 0$$

A_3 به M_3 نزدیکتر است پس آن را در خوشه سوم قرار می‌دهیم.

$$|A_4 - M_1| = \sqrt{(2-2)^2 + (5-5)^2} = 0$$

$$|A_4 - M_2| = \sqrt{(2-6)^2 + (5-3)^2} = 2\sqrt{5} \simeq 4.47$$

$$|A_4 - M_3| = \sqrt{(2-8)^2 + (5-4)^2} = \sqrt{37} \simeq 6.08$$

A_4 به M_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

$$|A_5 - M_1| = \sqrt{(7-2)^2 + (5-5)^2} = 5$$
$$|A_5 - M_2| = \sqrt{(7-6)^2 + (5-3)^2} = \sqrt{5} \simeq 2.24$$
$$|A_5 - M_3| = \sqrt{(7-8)^2 + (5-4)^2} = \sqrt{2} \simeq 1.41$$

A_5 به M_3 نزدیکتر است پس آن را در خوشه سوم قرار می‌دهیم.

$$|A_6 - M_1| = \sqrt{(4-2)^2 + (6-5)^2} = \sqrt{5} \simeq 2.23$$
$$|A_6 - M_2| = \sqrt{(4-6)^2 + (6-3)^2} = \sqrt{13} \simeq 3.61$$
$$|A_6 - M_3| = \sqrt{(4-8)^2 + (6-4)^2} = 2\sqrt{5} \simeq 4.47$$

A_6 به M_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

$$|A_7 - M_1| = \sqrt{(5-2)^2 + (7-5)^2} = \sqrt{13} \simeq 3.61$$
$$|A_7 - M_2| = \sqrt{(5-6)^2 + (7-3)^2} = \sqrt{17} \simeq 4.12$$
$$|A_7 - M_3| = \sqrt{(5-8)^2 + (7-4)^2} = 3\sqrt{2} \simeq 4.24$$

A_7 به M_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

$$|A_8 - M_1| = \sqrt{(2-2)^2 + (8-5)^2} = 3$$
$$|A_8 - M_2| = \sqrt{(2-6)^2 + (8-3)^2} = \sqrt{41} \simeq 6.4$$
$$|A_8 - M_3| = \sqrt{(2-8)^2 + (8-4)^2} = 2\sqrt{13} \simeq 7.21$$

A_8 به M_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

در مرحله دوم، جدول خوشه ها به صورت زیر در می اید.

خوشه اول	A_1, A_4, A_6, A_7, A_8
خوشه دوم	A_2
خوشه سوم	A_3, A_5

حال بررسی می‌کنیم که آیا cost کمتر شده است یا بیشتر؟

Cost =10 + 0 + 0 + 0 + 2 + 5 + 13 + 9 = 39

همانطور که مشاهده می‌کنیم، cost از 79 به 39 کاهش یافته و این کلاستر، بهبودی بیشتری را در خوشه بندی نسبت به مرحله‌ی قبل (مرحله اول) ایجاد می‌کند. لذا این کلاستر را به عنوانِ کلاسترِ جدید انتخاب می‌کنیم.

کلاستر جدید:

خوشه اول	A_1, A_4, A_6, A_7, A_8
خوشه دوم	A_2
خوشه سوم	A_3, A_5

مرحله سوم

حال مجددا یک نقطه تصادفی دیگر از یکی از کلاسترها را به عنوانِ medoid در نظر گرفته و عملیات مراحل قبل را روی آن اجرا می‌کنیم. طبق فرضی که در سوال گفته شده است، این نقطه تصادفی را A_5 که از کلاسترِ سوم است به عنوان medoid بجای A_3 در نظر می‌گیریم.

$M_1 = A_4$

$M_2 = A_2$

$M_3 = A_5$

حال فاصله هر یک از نقاط را از هر یک از medoid خوشه‌ها بدست می‌آوریم و خوشه های جدید را یافته و medoid های جدید را می یابیم سپس با توجه به cost ای که دارند، تصمیم می‌گیریم که خوشه‌ی جدید جایگزین قبلی شود یا خیر.

$|A_1 - M_1| = \sqrt{(1-2)^2 + (2-5)^2} = \sqrt{10}$

$|A_1 - M_2| = \sqrt{(1-6)^2 + (2-3)^2} = \sqrt{26} \simeq 5.1$

$|A_1 - M_3| = \sqrt{(1-7)^2 + (2-5)^2} = 3\sqrt{5} \simeq 6.71$

A_1 به M_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

$|A_2 - M_1| = \sqrt{(6-2)^2 + (3-5)^2} = 2\sqrt{5} \simeq 4.47$

$|A_2 - M_2| = \sqrt{(6-6)^2 + (3-3)^2} = 0$

$|A_2 - M_3| = \sqrt{(6-7)^2 + (3-5)^2} = \sqrt{5} \simeq 2.23$

A_2 به M_2 نزدیکتر است پس آن را در خوشه دوم قرار می‌دهیم.

$$|A_3 - M_1| = \sqrt{(8-2)^2 + (4-5)^2} = \sqrt{37} \simeq 6.08$$

$$|A_3 - M_2| = \sqrt{(8-6)^2 + (4-3)^2} = \sqrt{5} \simeq 2.23$$

$$|A_3 - M_3| = \sqrt{(8-7)^2 + (4-5)^2} = \sqrt{2} \simeq 1.41$$

A_3 به M_3 نزدیکتر است پس آن را در خوشه سوم قرار می‌دهیم.

$$|A_4 - M_1| = \sqrt{(2-2)^2 + (5-5)^2} = 0$$

$$|A_4 - M_2| = \sqrt{(2-6)^2 + (5-3)^2} = 2\sqrt{5} \simeq 4.47$$

$$|A_4 - M_3| = \sqrt{(2-7)^2 + (5-5)^2} = 5$$

A_4 به M_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

$$|A_5 - M_1| = \sqrt{(7-2)^2 + (5-5)^2} = 5$$

$$|A_5 - M_2| = \sqrt{(7-6)^2 + (5-3)^2} = \sqrt{5} \simeq 2.24$$

$$|A_5 - M_3| = \sqrt{(7-7)^2 + (5-5)^2} = 0$$

A_5 به M_3 نزدیکتر است پس آن را در خوشه سوم قرار می‌دهیم.

$$|A_6 - M_1| = \sqrt{(4-2)^2 + (6-5)^2} = \sqrt{5} \simeq 2.23$$

$$|A_6 - M_2| = \sqrt{(4-6)^2 + (6-3)^2} = \sqrt{13} \simeq 3.61$$

$$|A_6 - M_3| = \sqrt{(4-7)^2 + (6-5)^2} = \sqrt{10}$$

A_6 به M_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

$$|A_7 - M_1| = \sqrt{(5-2)^2 + (7-5)^2} = \sqrt{13} \simeq 3.61$$

$$|A_7 - M_2| = \sqrt{(5-6)^2 + (7-3)^2} = \sqrt{17} \simeq 4.12$$

$$|A_7 - M_3| = \sqrt{(5-7)^2 + (7-5)^2} = 2\sqrt{2} \simeq 2.83$$

A_7 به M_3 نزدیکتر است پس آن را در خوشه سوم قرار می‌دهیم.

$$|A_8 - M_1| = \sqrt{(2-2)^2 + (8-5)^2} = 3$$

$$|A_8 - M_2| = \sqrt{(2-6)^2 + (8-3)^2} = \sqrt{41} \simeq 6.4$$

$$|A_8 - M_3| = \sqrt{(2-7)^2 + (8-5)^2} = \sqrt{34} \simeq 5.83$$

A_8 به M_1 نزدیکتر است پس آن را در خوشه اول قرار می‌دهیم.

مرحله سوم، جدول خوشه ها به صورت زیر در می اید.

خوشه اول	A_1, A_4, A_6, A_8
خوشه دوم	A_2
خوشه سوم	A_3, A_5, A_7

حال بررسی می‌کنیم که آیا cost کمتر شده است یا بیشتر؟

Cost = 10 + 0 + 2 + 0 + 0 + 5 + 8 + 9 = 34

همانطور که مشاهده می‌کنیم، cost از 39 به 34 کاهش یافته و این کلاستر، بهبودی بیشتری را در خوشه بندی نسبت به مرحله‌ی قبل (مرحله دوم) ایجاد می‌کند. لذا این کلاستر را به عنوانِ کلاسترِ جدید انتخاب می‌کنیم.

کلاستر جدید:

خوشه اول	A_1, A_4, A_6, A_8
خوشه دوم	A_2
خوشه سوم	A_3, A_5, A_7

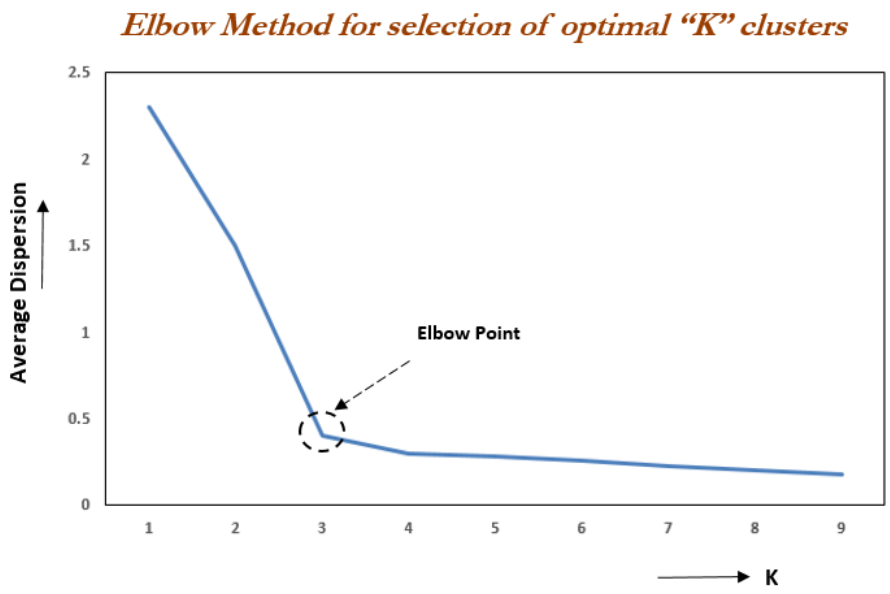
پاسخ سوال 2

ما در اینجا به روشی که مشهور به Elbow method است و برای یافتنِ تعداد بهینه‌ی خوشه‌ها در الگوریتم‌های خوشه‌بندی که k-means هم از جمله‌ی آن است، بکار می‌رود، اشاره می‌کنیم.

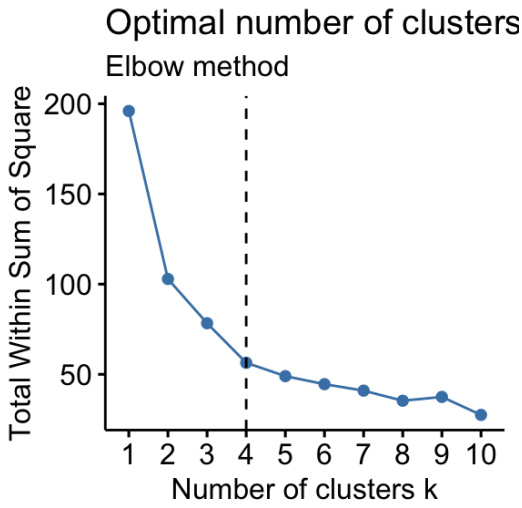
Elbow method

در این متد در ابتدا الگوریتمِ خوشه‌بندی را برای مقادیرِ مختلفِ k (مثلا k از 1 تا 10) را اعمال می‌کنیم. سپس برای هر مقدارِ k (تعداد خوشه) معیار خطای wss را محاسبه کرده. سپس منحنی معیار خطای wss را به ازای مقادیر مختلف از k رسم می‌نماییم و نقطه‌ی زانوی (شکستگی) منحنی را مشخص کرده و k متناظر با این نقطه‌ی زانویی برابر با مقدار k بهینه برای تعداد خوشه‌ها می‌باشد. (دقت کنید که پس از این نقطه نمودار به یک حالتِ نسبتاِ *stable* ای خواهد رسید و شیب و تغییرات آن کم خواهد شد)

در شکل زیر مفهوم دقیقِ نقطه‌ی زانویی (شکستگی) را در این متد نشان دادیم.



شکل زیر هم نمونه‌ای دیگر از اعمالِ این متد است.



پاسخ سوال 3

❖ K-medoids

در این الگوریتم به اندازه‌ی مقدارِ k (تعداد خوشه ها)، از میانِ داده‌ها medoid را که در ابتدا به صورتِ اختیاری است، انتخاب می‌کنیم. سپس فاصله‌ی هر داده را از این medoid ها محاسبه کرده و بر اساسِ نزدیکیِ فواصلِ داده‌ها به این medoid ها

هر یک در خوشه‌ی مربوط به خود قرار می‌گیرید. این روند مجددا تکرار شده و هر بار یک medoid جدید را (که کمترین فاصله نسبت به medoid خوشه خود دارد و در حالتِ pam، اختیاری است) در یکی از خوشه‌ها را جایگزینِ medoid قبلی کرده و در هر مرحله خطای SSE فواصل داده‌ها از medoid در هر خوشه را حساب کرده و با هم جمع می‌کنیم (در واقع داریم خطای WSS را محاسبه می‌کنیم). هر بار که این معیارِ خطا هزینه‌ی کمتری را برآورد کرد مسلما مدلِ بهتری خواهد بود و جایگزینِ مرحله‌ی قبلش خواهد شد و اگر که هزینه‌ی بیشتری را برآورد کرد، آنگاه مدل قبلی را حفظ کرده و medoid را مجددا تغییر می‌دهیم. شرطِ خاتمه نیز عدمِ تغییرِ داده‌ای تخصیص یافته به کلاسترها بعد از دو یا سه مرحله است.

مزایا :

- ✓ برای دیتاست‌های کوچک بسیار الگوریتمِ مناسبی است.
- ✓ مدیریت داده‌های داینامیک (Handling dynamic data)

معایب:

- ✓ برای دیتاست‌های بزرگ کارا نیست.
- ✓ مصونیت از نویز به طور کامل ندارد.
- ✓ مدیریت داده‌های از دست رفته را ندارد (No Missing Values Handling)

❖ CLARA

روشِ کارِ این الگوریتم بدین صورت است که از دیتاستِ اصلی چندین زیر مجموعه با اندازه ثابت ایجاد می‌کنیم. سپس الگوریتمِ PAM را با در نظر داشتنِ پارامترِ k (تعداد خوشه) روی هر یک از زیرمجموعه‌ها اجرا می‌کنیم و هر داده از کل داده ها را متناسب با نزدیکیِ آن به هر یک از medoid ها به آن medoid نسبت داده و در یک خوشه قرار می‌دهیم. و هر بار خطای SSE را مشابه با آنچه توضیح دادیم برای هر یک از این زیر مجموعه‌ها حساب کرده و به عنوانِ معیاری برای goodness آن کلاسترینگ تلقی می‌کنیم. این روند را چند بار تکرار کرده تا به خطای SSE مطلوبی برسیم (معمولا در بعضی مراجع تعداد 5 بار برای اجرای این روند را کافی می‌دانند). در نهایت آن کلاسترینگی را که دارای هزینه‌ی SSE کمتری باشد از goodness بالاتری برخوردار بوده و به عنوانِ کلاسترینگ مناسب برای دیتاست انتخاب می‌کنیم.

مزایا :

- ✓ برای دیتاست‌های کوچک و حتی دیتاست‌های بزرگ پاسخگو است.
- ✓ آن را می‌توان بر روی داده هایی با تمامی attribute ها اعمال کرد.

معایب:

- ✓ مصونیت از نویز به طور کامل ندارد.

❖ DBSCAN

DCSCAN در واقع مخفف شده‌ی Density Based Spatial Clustering of Applications with Noise می باشد و در واقع بدین صورت است که هیچ نیازی نیست که تعداد خوشه‌ها در ابتدا تعیین شود. در الگوریتم DBSCAN دو پارامتر min_samples و eps وجود دارد. هر نقطه از داده با نقاط دیگر فاصله‌ای دارد. هر نقطه‌ای که فاصله اش با یک نقطه مفروض کمتر از ε باشد به عنوان همسایه آن نقطه حساب می‌شود. هر نقطه مفروض که μ همسایه داشته باشد، یک نقطه مرکزی است. حال روند به این صورت است که در ابتدا نقطه مرکزی به صورت اختیاری انتخاب می‌شود که قبلاً بازدید نشده‌است. همسایگی این نقطه به شعاع ε بررسی می‌شود و در صورتی که حداقل تعداد نقاط همسایگی لازم را داشت. خوشه ایجاد می‌شود وگرنه نقطه نویزی برچسب می‌خورد. نکته قابل این است که در ادامه ممکن است این نقطه در همسایگی دیگر نقاط قرار گیردو قسمتی از خوشه دیگر شود.

. مزایا :

- ✓ عملیات کلاسترینگ برای داده‌هایی که بعد کمی داشته باشند بسیار سریع صورت می‌گیرد.
- ✓ نقاط نویز در این الگوریتم به خوبی تشخیص داده می‌شوند.
- ✓ مقیاس پذیری بالا (scalability)
- ✓ Outlier handling
- ✓ در این الگوریتم نیازی به تعریف تعداد خوشه‌ها از قبل نیست.

معایب:

- ✓ نقاط مرزی (نقاطی که در دو خوشه می‌توانند باشند) ممکن است به هریک از خوشه‌ها تعلق گیرند.
- ✓ مدیریت داده‌های از دست رفته را ندارد (No Missing Values Handling)
- ✓ این روش متناسب با تغییرات در μ و ε می‌تواند یک رفتار غیرقابل پیش بینی از خود نشان دهد.
- ✓ برای داده‌هایی با بعد بالا مناسب نیست.
- ✓ قبل از شروع به اجرای الگوریتم نیاز به تعیین دو پارامتر ورودی (εوμ) می‌باشد

❖ OPTICS

این الگوریتم در واقع راه حلی برای یکی از نقاط ضعف روش خوشه بندی مبتنی بر چگالی ارائه می کند آن هم وابستگی به پارامتر های ورودی آن می باشد. ورودی این الگوریتم اپسیلون و MinPts می باشد. اپسیلون همان بیشترین فاصله ای است که برای ساخت کلاستر باید در نظر گرفت. روش اپتیکس بر اساس فاصله کسینوسی بین نمونه ها کار میکنه و برای اینکه یک خوشه جدید تشکیل شود باید تعداد نمونه هایی که دور هم جمع شده اند از عدد خاصی بیشتر باشد که این عدد همان Minptr می باشد. طبیعتا هرچه این عدد کمتر باشد خوشه های بیشتری تشکیل میشود. در این روش همیشه خوشه‌ی آخر نمونه های باقیمانده از بقیه خوشه ها را در خود ذخیره می کند. به همین دلیل در اکثر موارد خوشه آخر اندازه بیشتری نسبت به بقیه‌ی خوشه ها دارد. روش کار این الگوریتم بدین صورت است که ابتدا تمام نقاط را بر اساس میزان چگالیِ آن‌ها مرتب کرده (برای اینکار می‌تواند از توزیع آماری مانند توزیع گوسی استفاده کند.) و سپس یک نقطه دلخواه را انتخاب نموده و همه‌ی نقاطی را که فاصله آنها از این نقطه کمتر یا مساوی با اپسیلون است را حساب می کند و اگر تعداد آنها بیشتر یا مساوی با Minptr باشد یک خوشه جدید را در این ایتریشن می‌سازد و نقطه‌ی مذکور را به عنوانِ مرکزِ این خوشه در نظر می‌گیرد. و در غیر اینصورت به سراغ نقطه‌ی بعدی می‌رویم.

مزایا :

- ✓ مقیاس پذیری بالا (scalability)
- ✓ Outlier handling
- ✓ نقاط نویز در این الگوریتم به خوبی تشخیص داده می‌شوند.

- ✓ بر خلاف الگوریتمِ قبلی، اگر داده ها دارای تراکم قابل تغییر باشند، این الگوریتم خوب عمل می‌کند.

معایب:

- ✓ نقاط مرزی (نقاطی که در دو خوشه می‌توانند باشند) ممکن است به هریک از خوشه‌ها تعلق گیرند.
- ✓ نسبت به داده های اشتباه حساسیت کمتری دارد.

❖ BIRCH

بیرچ مخفف عبارتِ **Balanced Iterative Reducing and Clustering using Hierarchies** است که در واقع این الگوریتم می‌تواند یک دیتاستِ بزرگ را خوشه بندی کند بدین صورت که یک مجموعه‌ی جمع و جور و کوچکتر که حداکثر مقدار اطلاعاتِ ممکن را از مجموعه‌ی اصلی دارد تولید کرده و عملیاتِ خوشه بندی را روی این مجموعه انجام می‌دهد و در نهایت این کلاسترینگِ بدست آمده به عنوانِ کلاسترینگِ کل دیتاست قرار می‌گیرد.

. مزایا :

- ✓ این الگوریتم یکی از بهترین الگوریتم‌ها برای دیتاست‌ها بزرگ است (از جنبه‌های running time و فضای مورد نیازو کیفیت و تعداد IO اعمالی به آن)
- ✓ این الگوریتم با توجه به افزایشِ تعداد آبجکت‌ها، یک مقیاس پدیری یا scalability خطی از خود نشان می‌دهد.

معایب:

- ✓ این الگوریتم تنها برای داده های spherical بسیار عالی عمل می‌کند.

❖ CHAMELEON

این الگوریتم، یک الگوریتم سلسله مراتبی تجمیعی و یا به عبارتِ بهتر تجمیعیِ دومرحله ای است که مبتنی برنمودار نزدیک ترین همسایه k می‌باشد بدین صورت که اگر هر دو رأس داخل نزدیک ترین همسایگی k خود نباشند، یک لبه حذف خواهد شد. در اولین مرحله ، Chameleon نمودار اتصال را داخل یک مجموعه زیر خوشه با برش کمینه لبه تقسیم می‌کند و این کار با یک الگوریتم

جزء بندی نمودار به نام hMetis انجام می‌شود. هر زیر نمودار باید شامل گره های کافی باشد و یک خوشه‌بندی سلسله مراتبی تجمیعی به کار گرفته می‌شود تا این زیرخوشه‌ها، خوشه‌های نهایی را تشکیل دهند.

مزایا:

- ✓ تشخیص خوشه‌هایی با شکل غیر کروی و با سایزهای متنوع
- ✓ در عملیاتِ ادغام ملاک ما فاصله بین کلاسترها و نزدیکی درون کلاسترها است.

معایب:

- ✓ عدمِ مدیریتِ داده‌های دور افتاده outliers
- ✓ به پارامترها خیلی حساس است.
- ✓ گراف باید متناسب با حافظه باشد.

پاسخ سوال 4

هر یک از عبارات را قرار داده و توضیحات مربوط به آنها را ارائه می‌کنیم.

الف) برای اینکه نقاط داده در یک خوشه قرار گیرند، باید در فاصله‌ی آستانه‌ای از یک نقطه هسته (core point) باشند.

پاسخ : این عبارت صحیح است ولی توجه شود که عبارت شفاف‌تر بدین صورت است که در واقع در الگوریتمِ DBSCAN برای آنکه نقاط داده در یک خوشه قرار گیرند باید دو شرط همزمان برقرار باشد. یکی اینکه نقاط در فاصله‌ی آستانه از یک نقطه‌ی هسته قرار گیرند و دوم اینکه تعدادشان از یک حد آستانه (min_samples) بیشتر باشد که البته چون نقطه‌ی هسته از قبل داریم لذا شرط دوم در اینجا برقرار بوده و لذا عبارت صحیح است. در واقع اگر از لفظ نقطه‌ی هسته استفاده نمی کرد و یک نقطه را ذکر میکرد و نیز تنها از لفظِ فاصله‌ی آستانه استفاده کنیم، تنها می‌توانیم بگوییم که این داده در همسایگی آن نقطه‌ی قرار دارد. ولی با توجه به اینکه در ایجا از لفظ نقطه‌ی هسته استفاده شده است، لذا این عبارت صحیح است.

ب) این الگوریتم نسبت به داده‌های پرت (outliers) مقاوم است.

این عبارت صحیح است. دلیلِ آن این است که DBSCAN یک الگوریتمی مبتنی بر چگالیِ داده‌هاست و لذا در هنگامِ کلاسترینگ تراکمِ داده‌ها ها را می‌تواند توسطِ دو پارامتر μوε بررسی کرده و سپس کلاسترینگ را انجام می‌دهد. حال با توجه به این نکته که ابتدا فاصله‌ی داده را تا نقطه‌ی هسته بررسی می‌کند و داده‌های outlier در فاصله‌ی دوری نسبت به بقیه‌ی داده‌ها طبق تعریفی که دارند قرار می‌گیرند، لذا به راحتی قابلِ تشخیص هستند و این الگوریتم این داده‌ها را تشخیص داده و به آن برچسبِ 1- می‌زند.

ج) پیچیدگی زمانی این الگوریتم از مرتبه $O(n^3)$ است.

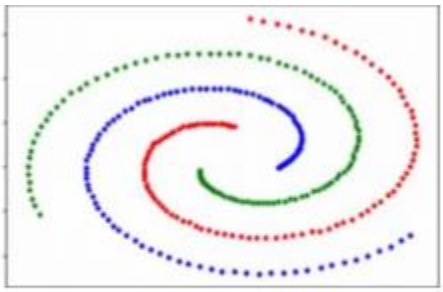
این عبارت غلط است. زیرا بدترین حالت پیچیدگیِ زمانیِ اجرای الگوریتم یا همان worth case run time complexity در این الگوریتم برابر با $O(n^2)$ است که در این صورت الگوریتم برای هر نقطه این بررسی را انجام می‌دهد که آیا نقطه‌ی هسته است یا خیر. البته این پیچیدگی زمانی می‌تواند به $O(n*log(n))$ نیز کاهش یابد که برای ابعادِ پایین‌تر و توسطِ ساختمان داده‌های efficient امکان‌پذیر است.

د) این الگوریتم نیازی به دانستن تعداد خوشه‌ها پیش از انجام خوشه‌بندی ندارد.

این عبارت کاملا صحیح است. در واقع این الگوریتم پارامترهای دیگری را به عنوان ورودی دارد که این پارامترها بر اساسِ تراکمِ موجود در در داده‌ها هود عملیاتِ کلاسترینگ را انجام داده و به مرور تعداد کلاسترها مشخص می‌شود. این پارامترها که ذکر شد پارامترهای min_samples و eps هستند. می‌دانیم هر نقطه از داده با نقاط دیگر فاصله‌ای دارد. هر نقطه‌ای که فاصله اش با یک نقطه مفروض کمتر از ε باشد به عنوان همسایه آن نقطه حساب می‌شود. هر نقطه مفروض که μ همسایه داشته باشد، یک نقطه مرکزی است. حال روند به این صورت است که در ابتدا نقطه مرکزی به صورت اختیاری انتخاب می‌شود که قبلاً بازدید نشده‌است. همسایگی این نقطه به شعاع ε بررسی می‌شود و در صورتی که حداقل تعداد نقاط همسایگی لازم را داشت. خوشه ایجاد می‌شود وگرنه نقطه نویزی برچسب می‌خورد. لذا بدین صورت خوشه‌ها پیدا می‌شود و هیچ نیازی نیست که تعداد خوشه‌ها در ابتدا تعیین شود.

پاسخ سوال 5

ابتدا **شکل الف** را بررسی می‌کنیم.



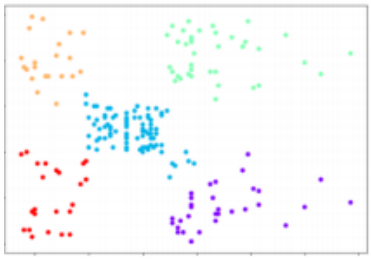
الف

روش DBSCAN

ابتدا باید به این نکته‌ی ظریف و کلیدی توجه کنیم که DBSCAN بر اساس چگالی و در واقع تراکمِ داده‌ها، کلاسترینگ را انجام می‌دهد ولی در k means این عملیاتِ خوشه‌بندی به گونه‌ای می‌تواند به طورِ مناسب صورت گیرد که بتوان حداکثر با تعداد خطِ صاف (یعنی حداکثر به تعداد مرزهای دو به دو یک خط صاف) خوشه‌ها را تفکیک کرد. لذا به راحتی با توجه به این نکته می‌توانیم در موردِ این چهار موارد تصمیم بگیریم که کدام الگوریتم بهتر است.

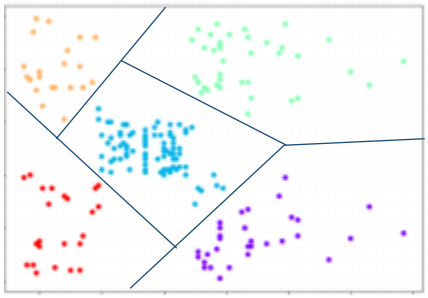
و اما برای شکل الف، طبق نکته‌ی فوق یقینا تنها الگوریتمِ مناسب بین این دو الگوریتم DBSCAN است. زیرا DBSCAN نقاط هسته را مشخص کرده و با تعیینِ درستِ فاصله‌ی همسایگی به راحتی خوشه را گسترش داده و سه خوشه‌ی مجزا بدلیلِ تراکمی که مشاهده می‌شود را ایجاد می‌کند و عملیات کلاسترینگ را با تفکیکِ دیتاستِ فوق به سه خوشه، به درستی انجام خواهد داد. ولی در kmeans با توجه به فرضِ سوال که از تعداد خوشه‌ها که سه تا هستند اطلاع داشته باشیم، هیچگاه نخواهیم توانست شکلِ فوق را که از سه خطِ مارپیچیِ متراکم تشکیل شده است، با سه خطِ صاف از یکدیگر تفیک کرد. در واقع هیچ سه خطِ صافی نمی‌تواند خوشه‌های فوق را از یکدیگر تفکیک نماید و لذا به هیچ عنوان kmeans مناسب نیست.

شکل ب



ب

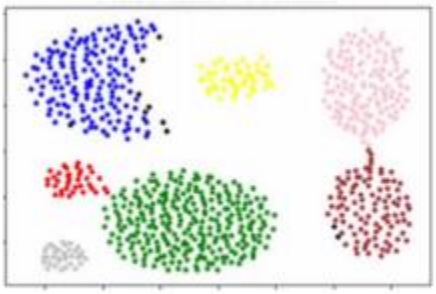
برای شکل ب kmeans الگوریتمِ مناسب تری است زیرا اولا می‌توانیم خوشه‌ها را با 5 خط که برابر با تعداد خوشه‌هاست از یکدیگر تفکیک کرده و دقیقا به شکل بالا رسید. به مرز بندی‌ای که در زیر کرده‌ایم توجه نمایید.



ب

لذا k means الگوریتمِ مناسبی برای کلاسترینگِ شکل ب است ولی توجه شود که الگوریتمِ DBSCAN آنچنان مناسب نیست و شاید نتواند به درستی عملیاتِ کلاسترینگ را انجام دهد زیرا داده‌های موجود در خوشه‌ها به گونه‌ای متراکم نیستند که DBSCAN بتواند درست خوشه بندی کند به عنوان مثال به فاصله خوشه قرمز و آبی توجه کنید(منظور فاصله‌ی نزدیکترین داده هاست) همانطور که میبینید هم این فواصلِ بینِ خوشه ها کم است و هم داده‌های موجود در خوشه‌ها متراکم نیستند و لذا این دو عامل باعث می‌شود که DBSCAN برای شکل ب مناسب نباشد.

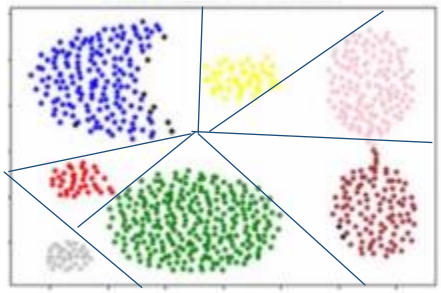
شکل ج



ج

هم DBSCAN و هم k means برای شکلِ ج مناسب هستندو اولا DBSCAN با توجه به تراکمی که بین داده های شکل فوق مشاهده می‌شود کاملا مناسب است زیرا اگر قرار باشد که خوشه بندی مطابقِ رنگ‌های شکلِ فوق صورت گیرد، مشاهده می‌کنید که تراکم در هر یک از این خوشه‌ها بالا و با توجه به فاصله‌ی همسایگیِ بینِ آنها که عاملی متمایز کننده برای خوشه‌ها است، می‌توان

نتیجه گرفت که با تنظیمِ درستِ پارامترهای εوμ به 7 کلاستر مشابه با شکلِ فوق رسید. از طرفی الگوریتمِ k means هم الگوریتمی مناسب خواهد بود زیرا می‌دانیم که k means خوشه‌ها را به گونه‌ای که بتوان بینِ مرزهای آنها دو به دو خط متمایز کننده‌ی صافی گذاشت، تفکیک می‌نماید (دلیلِ این امر بررسی فاصله‌ی نقاط تا مرکز است که در صورتِ برابر بودنِ فاصله‌ها یک خط را تشکیل می‌دهند و به همین دلیل گفتیم خطوط صاف جداکننده و هر قسمتِ تفکیک شده را می‌توان یک کلاستر در نظر گرفت)



ج

البته به هر حال اگرچه هر دو الگوریتم در صورتِ انتخابِ پارامترهای ورودیِ مناسب برای شکل ج کارا خواهند بود، ولی بدلیل تراکم‌هایی که در خوشه‌ها مشاهده می‌شود شاید DBSCAN از کارایی بالاتری برای خوشه‌بندی برخوردار باشد.

شکل د

د

بی‌شک بینِ این دو الگوریتم، تنها DBSCAN عملکرد مناسبی خواهد داشت. اولا با توجه به تراکمی که در خوشه‌های بیضی‌شکل مشاهده می‌شود و فاصله‌ای که بینِ نقاط این دو بیضی شکل وجود دارد یقینا DBSCAN مناسب است ولی دلیل اینکه k mean مناسب نیست چون اصلا نمی توان به گونه‌ای با چند خط صاف این دو تراکمِ داده را از یکدیگر تفکیک کرد. به نظرم اگر به این

دیتاست، k means را اعمال کنیم حداکثر بتواند از وسط به دو نیم تشسیم کرده و هر نیم را در یک خوشه قرار دهد ولی اصلا نخواهد تنوانست خوشه‌بندی‌ای مطابق با شکل فوق انجام دهد و k means برای شکل د اصلا مناسب نیست.

پاسخ سوال 6

	A	B	C	D	E
A		1.23	2.44	0.85	2.04
B	1.23		0.74	1.2	0.98
C	2.44	0.74		1.34	1.4
D	0.85	1.2	1.34		0.87
E	2.04	0.98	1.4	0.87	

با توجه به اینکه ماتریس فاصله تقارن دارد لذا ما محاسبات را بر روی نیمه پایین انجام داده و می‌دانیم بر حسبِ تقارن، نیمه ی بالایی نیز مشابه با نیمه‌ی پایینیِ ماتریس خواهد بود.

همچنین در هر مرحله نمودار dendrogram هم رسم خواهیم نمود تا پیشرفتِ این نمودار متناسب با هر مرحله مشخص شود.

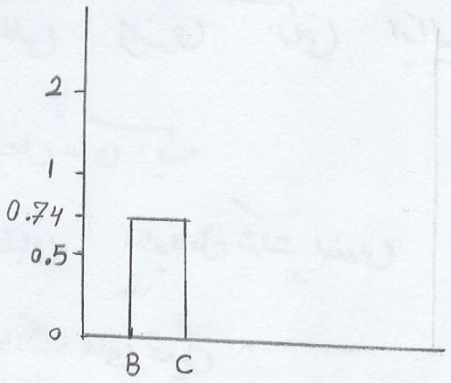
Single link

همانطور که می‌دانید در هر مرحله با توجه به ماتریسِ فاصله‌ی آپدیت شده که در single link کوچکترین فاصله را بین نودها (همچنین در نودهای merge شده) نشان می‌دهد، باید مقدار مینیمم را بیابیم و نودهای متناظرش را merge کنیم.

به محاسبات صفحه بعد توجه کنید.

مرحله ①

Dendogram



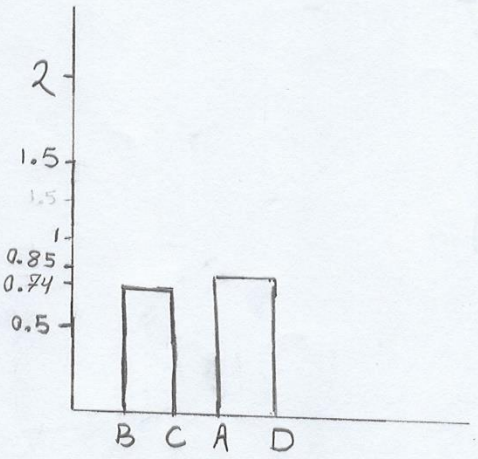
	A	B	C	D	E
A	0				
B	1.23	0			
C	2.44	0.74	0		
D	0.85	1.2	1.34	0	
E	2.04	0.98	1.4	0.87	0

min value : 0.74

merge → B , C

مرحله ②

Dendogram



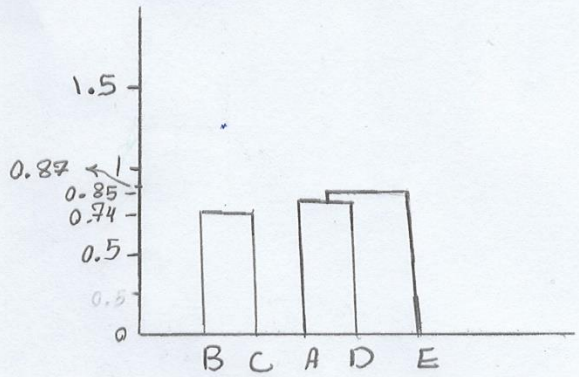
	A	(B,C)	D	E
A	0			
(B,C)	1.23	0		
D	0.85	1.2	0	
E	2.04	0.98	0.87	0

min value : 0.85

merge → A , D

مرحله ③

Dendogram

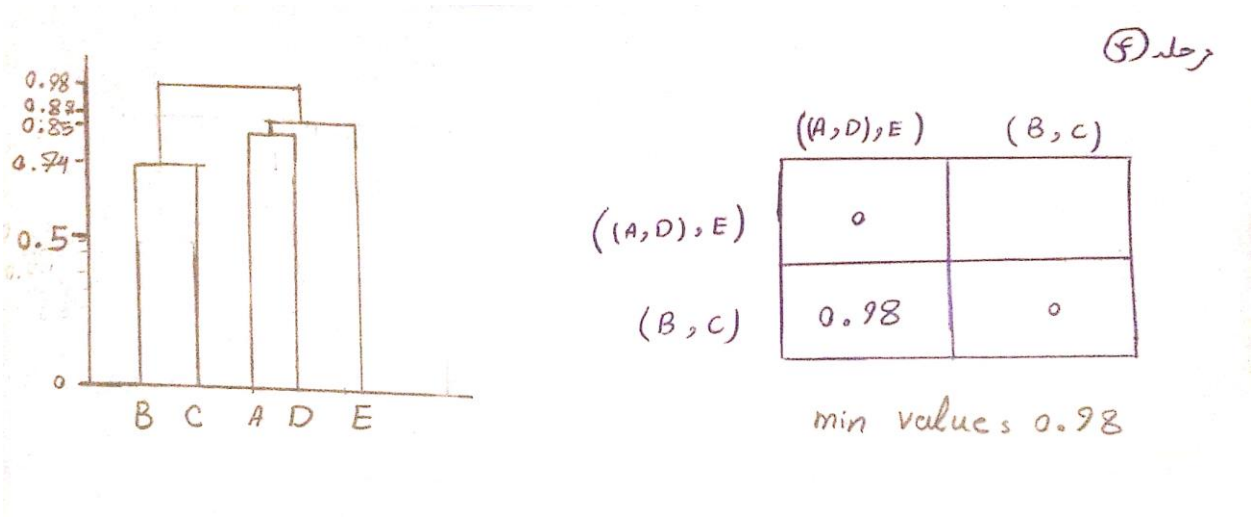


	(A,D)	(B,C)	E
(A,D)	0		
(B,C)	1.2	0	
E	0.87	0.98	0

min value : 0.87

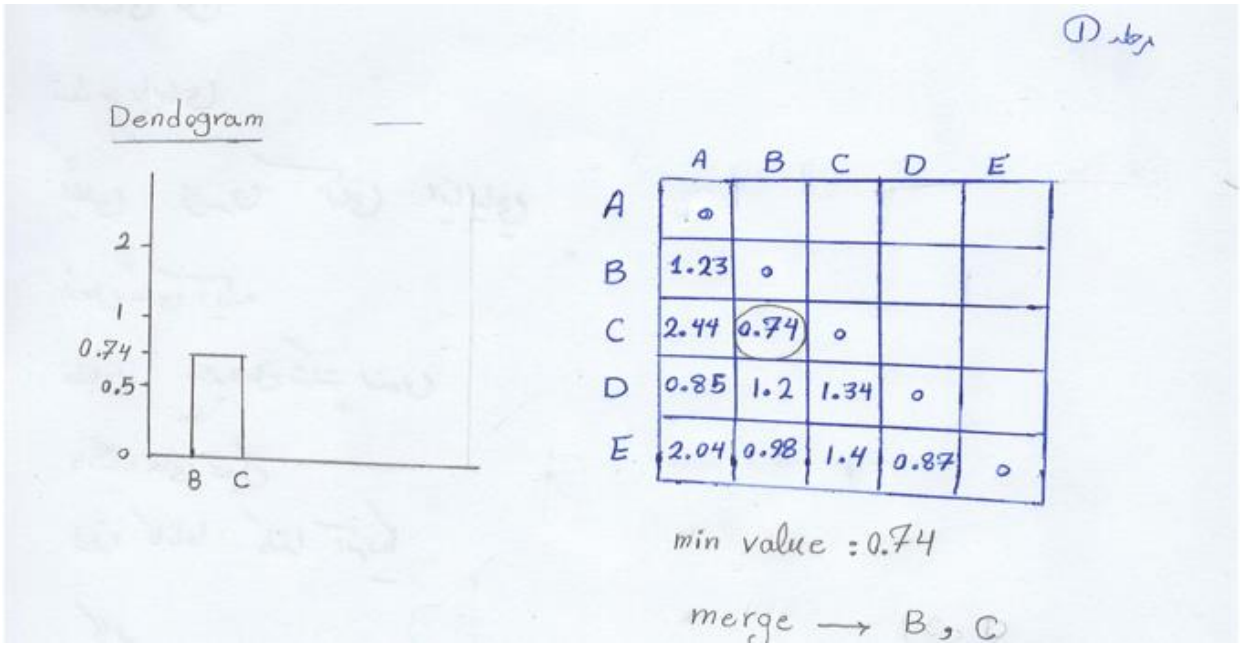
merge → (A,D) , E

و در مرحله آخر نیز که درختِ dendrogram به صورت زیر می‌شود.

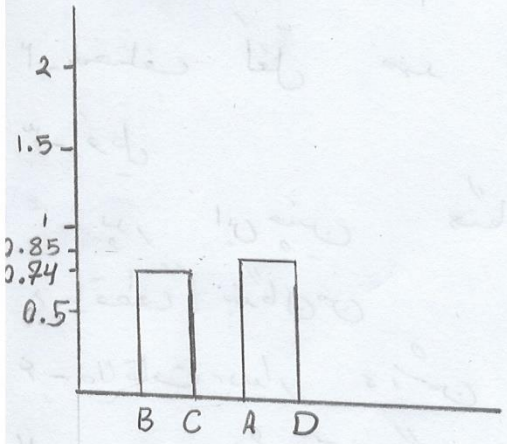


Complete link

تفاوت این نوع خوشه بندی با خوشه بندی single link در آپدیت کردنِ ماتریس است. بدین صورت که در هنگامِ ادغام کردنِ نود ها، وقتی می خواهیم فاصله بقیه نود ها تا این نودهای مرج شده را آپدیت کنیم باید، ماکسیمم فاصله را بین ترکیبی از نود با نودهای ادغام شده حساب کنیم. و در حالتی که دو نود هر دو ادغام شده باشند دو به دو فاصله ها را حساب کرده و ماکسیمم آن ها را قرار می‌دهیم بر خلافِ single link که مینیمم فاصله را در ماتریس آپدیت شده برای نودهای ادغام شده قرار می‌داد.



Dendogram



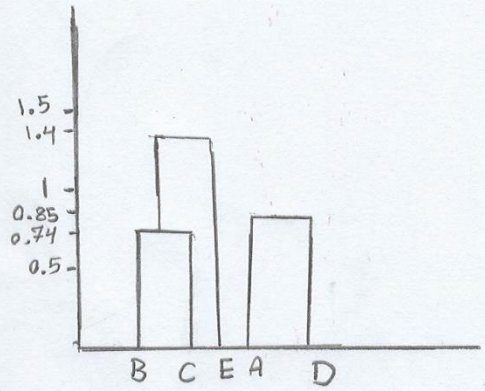
مرحله (۲)

	A	(B,C)	D	E
A	0			
(B,C)	2.44	0		
D	0.85	1.34	0	
E	2.04	1.4	0.87	0

min value : 0.85

merge ⟶ A, D

مرحله (۳)

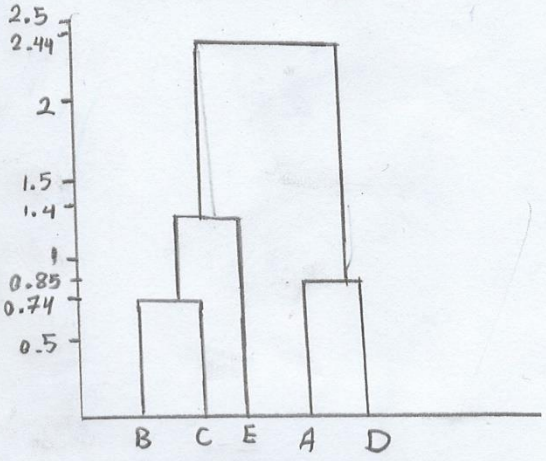


	(A,D)	(B,C)	E
(A,D)	0		
(B,C)	2.44	0	
E	2.04	1.4	0

min value : 1.4

merge ⟶ (B,C), E

مرحله (۴)



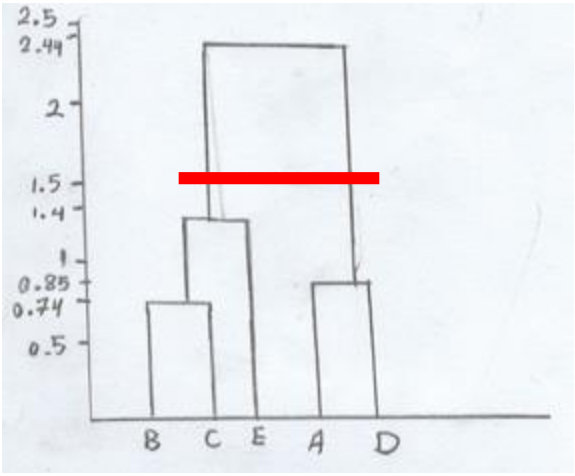
	(A,D)	((B,C),E)
(A,D)	0	
((B,C),E)	2.44	0

min value : 2.44

merge ⟶ ((B,C),E), (A,D)

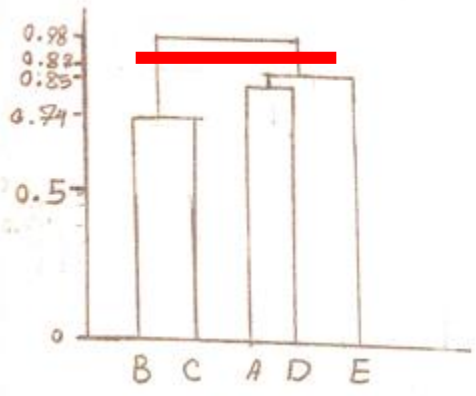
حال تمامی مراحل انجام شده است و کافی است که خط cut ای را رسم نماییم و دیتاست را به مجموعه‌ای از کلاسترهای مختلف و singleton هایی تقسیم نماییم. یک نمونه از کلاسترینگ را انجام می‌دهیم و چون سوال چیز بیشتری نگفته است لذا هر نوع تقسیمِ دستاست به کلاسترهای مختلف به مشابهِ زیر خواهد بود.

یک نمونه برای complete link



به عنوان یک نمونه در کلاسترینگِ complete link خط برش را روی 1 قرار دادیم و مشاهده می‌کنید که دیتاستِ ما به دو کلاستر تقسیم شده است.

یک نمونه برای single link



به عنوان یک نمونه در کلاسترینگِ single link خط برش را روی 0.9 قرار دادیم و مشاهده می‌کنید که در اینجا نیز دیتاستِ ما به دو کلاستر تقسیم شده است.

پاسخ مربوط به سوالات عملی

پاسخ 1

در ابتدا از ما خواسته شده است که الگوریتمِ Kmeans را پیاده سازی کنیم.

الگوریتم را مرحله به مرحله به همراه کد توضیح میدهیم و سپس تست‌های گرفته شده و درستی نتایج را نشان می‌دهیم.

همچنین طبق خواسته‌ی سوال، کد زده شده برای تمامی داده‌ها با هر ابعادی قابلِ اعمال است.

در ابتدا یک کلاس با نام Kmean را به صورت زیر ساخته ایم.

```python
import csv
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns, matplotlib.pyplot as plt, operator as op


class Kmean:
    def __init__(self ,file, cluster_num , max_itr):
        with open(file, newline='') as csvfile:
            data = np.asarray(list(csv.reader(csvfile)))
        self.data = data
        self.output = []
        self.cdata = self.data[np.array([x for x in range(1,self.data.shape[0])]),:]
        self.pure_data = self.cdata[:,np.array([x for x in range(0,self.data.shape[1]-1)])]
        self.C =[]

        #for i in range(1,data.shape[0]):
        #    if  self.data[i,self.data.shape[1]-1] not in self.data_class:
        #        self.data_class[self.data[i,self.data.shape[1]-1]] = self.pure_data[i-1,:]
        #    self.data_class[self.data[i,self.data.shape[1]-1]].append(self.pure_data[i-1,:])

        self.cluster_num = cluster_num
        self.max_itr = max_itr
        self.clusters = []

```

همانطور که مشاهده می‌کنید ورودی‌های ما فایلِ csv به عنوان داده‌های ورودی است و همچنین تعداد کلاسترها (k) که آن را با cluster_num نشان دادیم و پارامترِ ورودی آخر نیز، ماکسیمم ایترِیشِن است که آن را با max_itr نمایش داده ایم.

در اولین گامِ kmeans ، k نقطه‌ی تصادفی را به عنوانِ مرکزِ اولیه‌ی خوشه‌ها تعریف کرده‌ایم (این k نقطه کاملا تصادفی هستند ولی ما برای اینکه یک کدِ قابلِ تعمیم برای تمامیِ دیتاست‌ها بزنیم، مراکز اولیه را از بازه‌های داده‌ها به صورتِ رندوم انتخاب نمودیم که مراکز خیلی پرت نشوند و یا به بازه‌ی خاصی محدود نشوند.)

```
26      def setRand(self):
27          for i in range(self.cluster_num):
28              self.C.append(self.pure_data[np.random.randint(self.pure_data.shape[0])].astype(float))
29          #print(self.C)
30
```

پس از تعیین نقاط اولیه، دیتاست را به اندازه‌ی تعداد maximum iteration ، train می‌کنیم.

```
52      def train(self):
53          self.setRand()
54          for i in range(self.max_itr):
55              self.train_()
56
```

تابع train بدین صورت است که ابتدا فاصله‌ی هر داده را تا هر یک از مراکز حساب کرده و سپس به هر مرکزی نزدیکتر بود، آن را توسطِ یک لیست با نام clusters به آن متناظر می‌کند.

```
57
58      def train_(self):
59          self.lastClusters = self.clusters.copy
60          self.clusters = [[] for i in range(self.cluster_num)];
61          self.output_clusters = [[] for i in range(self.cluster_num)];
62          for p_data in self.cdata:
63              #print(p_data.shape)
64              pure_data = p_data[np.array([x for x in range(0,p_data.shape[0]-1)])]
65              pure_data = pure_data.astype(float)
66              d , last_d = 0 , 100000;
67              nearest_ci = 0;
68              for ci in range(self.cluster_num):
69                  d = self.dist(pure_data, self.C[ci])
70                  if d < last_d:
71                      last_d = d ;
72                      nearest_ci = ci ;
73              self.clusters[nearest_ci].append(p_data)
74              #print(np.asarray(p_data[p_data.shape[0]-1]))
75              self.output_clusters[nearest_ci].append(p_data[p_data.shape[0]-1])
76          self.clusters = np.array(self.clusters)
77          for ci in range(self.cluster_num):
78              if len(self.clusters[ci]) > 0:
79                  self.C[ci] = self.centerOfMass(self.clusters[ci])
80
```

همانطور که مشاهده می‌کنید در کلاسِ خود یک تابع با نامِ dist() درست کردیم که فاصله‌ی دو داده با هر ابعادی را در آن محاسبه نموده‌ایم. البته در این الگوریتم فاصله‌ی داده با هر یک از مراکز را توسطِ این تابع محاسبه می‌کنیم. در اینجا یک output_clusters نیز تعریف کرده ایم که جز پارامترهای اصلی کلاس است و در واقع خروجی خواسته شده در سوال را که نسبت دادنِ اسمِ گل به خوشه‌ی متناظرش است را درآن می‌ریزیم ولی در clusters همه‌ی داده را که مختصاتِ هر داده را نیز

شامل می‌شود در آن قرار می‌دهیم. در نهایت خروجی چاپ شده طبق خواسته‌ی سوال output_clusters است که تابع آن به صورت زیر می‌باشد.

```
80
81      def printClusters(self):
82          for i in range(self.cluster_num):
83              print('cluster No.' + str(i+1) + ' : ')
84              print(self.output_clusters[i])
85              print('\n')
86
```

همچنین تابع محاسبه‌ی distance برای دو نقطه با هر ابعادی نیز به صورت زیر است.

```
32      def dist(self,p1,p2):
33          return np.sqrt(np.sum((p2-p1)**2))
34
```

و همچنین خطای mean absolute error یا MAE را نیز به عنوانِ یک تابع در این کلاس تعریف نمودیم که در قسمتِ دوم توضیح می‌دهیم.

یک خروجی برای صحتِ پیاده‌سازی گرفته ایم. تعداد کلاستر را سه عدد و تعداد ماکسیمم iteration را ده عدد و داده‌ی ورودی نیز فایل iris.csv می‌باشد. خروجی به صورت زیر است.

```
C:\Windows\System32\cmd.exe - python3  main.py
D:\dars\[illegible]\HW2_9531701\Codes>python3 main.py
cluster No.1 :
['versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versic
olor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', '
versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicol
or', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 've
rsicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor
', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'versicolor', 'virginica', 'virginica', 'virginica', 'virgini
ca', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica'
, 'virginica']


cluster No.2 :
['setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa',
 'setosa', 'setosa']


cluster No.3 :
['versicolor', 'versicolor', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica',
'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'vi
rginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virgi
nica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virginic
a', 'virginica']


MAE Error : 0.6480304904934434
```

سه خوشه‌ای ایجاد شده با خطای بسیار کمی داده‌ها را به خوبی خوشه بندی کرده اند. (دقت بفرمایید که ما نقلط را تصادفی تعریف کردیم ولی با این وجود خوشه بندی به خوبی توسطِ این الگوریتم انجام شده است.

پاسخ 2

همانطور که در خروجی شکل قبل دیدید، MAE را برای تعداد کلاستری برابر با 3 عدد و max iteration ای برابر با 10 عدد، حدود 0.65 شد.

البته موارد خواسته شده در صورت سوال سه که له صورت یک نمودار باید خطاهای MAE را به ازای خطاهای مختلف بررسی کنیم را در همان بخش آورده ایم. و اینجا صرفا به پیاده‌سازیِ تابع MAE می‌پردازیم.

اسم این تابع را در کلاسِ خود avgOfDistance() گذاشته‌ایم.

```
86
87      def avgOfDistance(self):
88          sum = 0;
89
90          for ci in range(self.cluster_num):
91              cal_avg_data = np.asarray(self.clusters[ci])
92              cal_avg_data_arr = cal_avg_data[:,np.array([x for x in range(0,cal_avg_data.shape[1]-1)])]
93              cal_avg_data_arr=cal_avg_data_arr.astype(float)
94
95              for data in cal_avg_data_arr:
96                  sum = sum + abs(self.dist(data, self.C[ci]))
97          return sum/self.pure_data.shape[0]
98
```

همانطور که مشاهده می‌کنیم از تابعِ محاسبه‌گرِ فاصله‌ی نقاط تا مرکزِ هر خوشه‌ی متناظرِ خود را که در قسمت قبل توضیح دادیم، استفاده نمودیم و فواصلِ خواسته شده را حساب کرده و قدرِ مطلقِ همگی را جمع زدیم و در نهایت به تعداد داده‌ها تقسیم کردیم.

اگر به شکل صفحه قبل توجه بفرمایید مشاهده می‌کنید که خطای MAE داده‌های iris.csv حدودا برابر 0.65 محاسبه شد.

```
ca', 'virginica', 'virginica', 'virginica', 'virginica', 'virginica
, 'virginica']


cluster No.2 :
['setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setos
 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setos
 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setos
 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setosa', 'setos
 'setosa', 'setosa']


cluster No.3 :
['versicolor', 'versicolor', 'virginica', 'virginica', 'virginica',
'virginica', 'virginica', 'virginica', 'virginica', 'virginica', 'vi
rginica', 'virginica', 'virginica', 'virginica', 'virginica', 'virgi
nica', 'virginica', 'virginica', 'virginica', 'virginica', 'virgini
a', 'virginica']

MAE Error : 0.6480304904934434
```

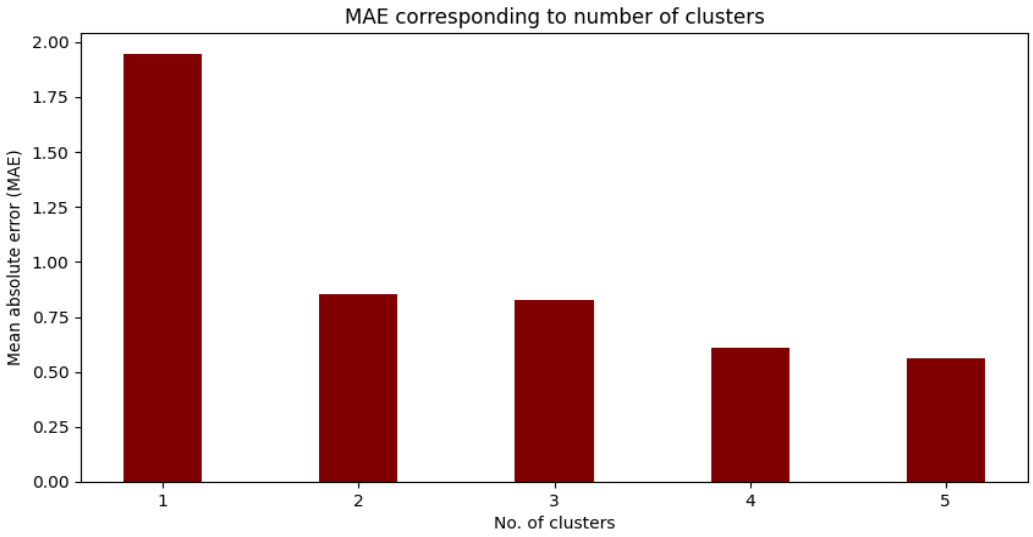
MAE

پاسخ 3

در این قسمت خواسته شده است که برای k در range(1,6) ، خطای MAE را محاسبه کرده و در یک نمودار ترسیم نماییم. به سادگی اینکار در قسمتِ main.py و با یک حلقه انجام دادیم. لذا کلاسِ خود را در این تابع فراخوانده و نتیجتا خروجی را در یک لیست ریخته با کتابخانه‌ی matplot ، آن را رسم نمودیم.

```python
import matplotlib.pyplot as plt
from Kmean import *

if __name__ == '__main__':

    max_iteration = 10;
    x_axis_name = []
    y_axis_value = []

    for k in range(1,6):
        model = Kmean('datasets/iris.csv',k,max_iteration)
        #model.train();
        #model.printClusters();
        #print("MAE Error : "+str(model.avgOfDistance())+"\n")

        x_axis_name.append(str(k))
        y_axis_value.append(model.avgOfDistance())

    fig = plt.figure(figsize = (10, 5))

    # creating the bar plot
    plt.bar(x_axis_name ,y_axis_value , color ='maroon',
            width = 0.4)

    plt.xlabel("No. of clusters")
    plt.ylabel("Mean absolute error (MAE)")
    plt.title("MAE corresponding to number of clusters")
    plt.show()

```

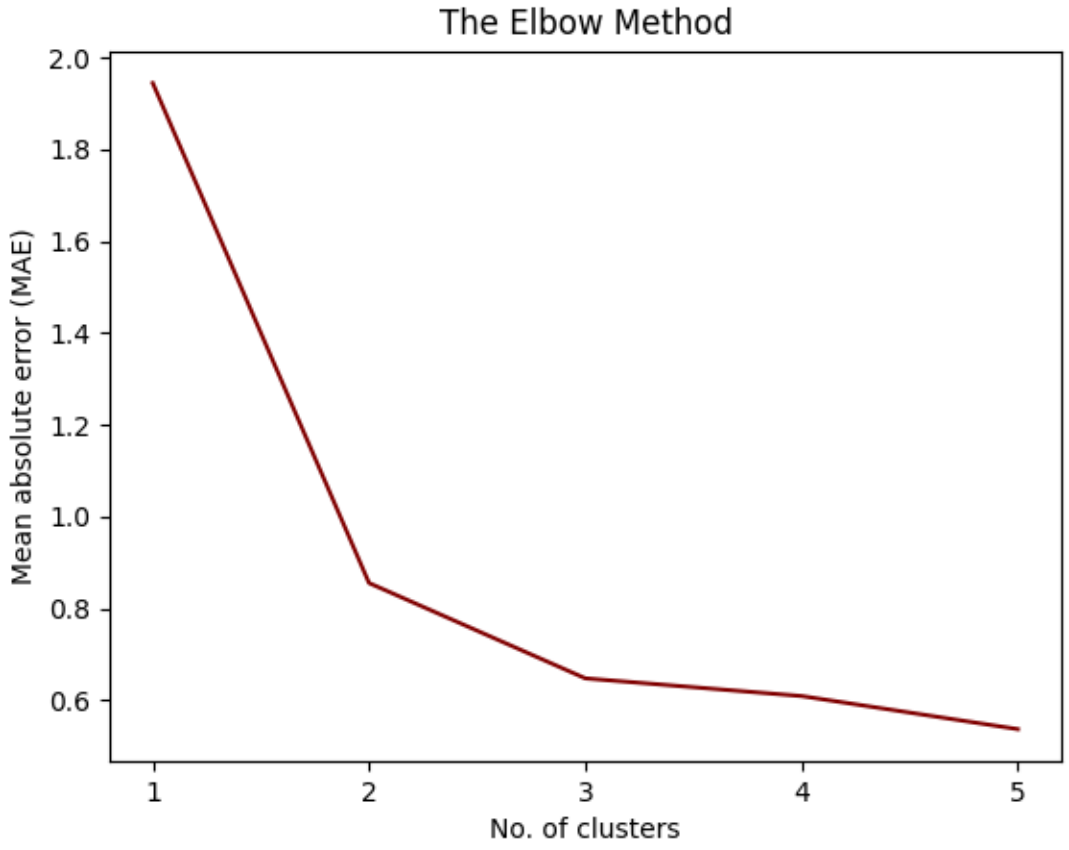
خروجی به صورت زیر است:



همانطوری که در شکل صفحه قبل مشاهده می‌کنید، با زیاد کردنِ تعداد کلاستر ها تا 5 کلاستر، خطا در حال کم شدن است. این برای این محدوده از تعداد کلاسترها قابلِ پیش‌بینی بود. لااقل تعداد سه کلاستر را نیاز داشتیم که هر گل با تعداد کمی خطای تشخیص، در دسته‌ی مخصوص به خود قرار گیرد. لذا شکل خواسته شده در قسمتِ 3 به صورت فوق می‌باشد.

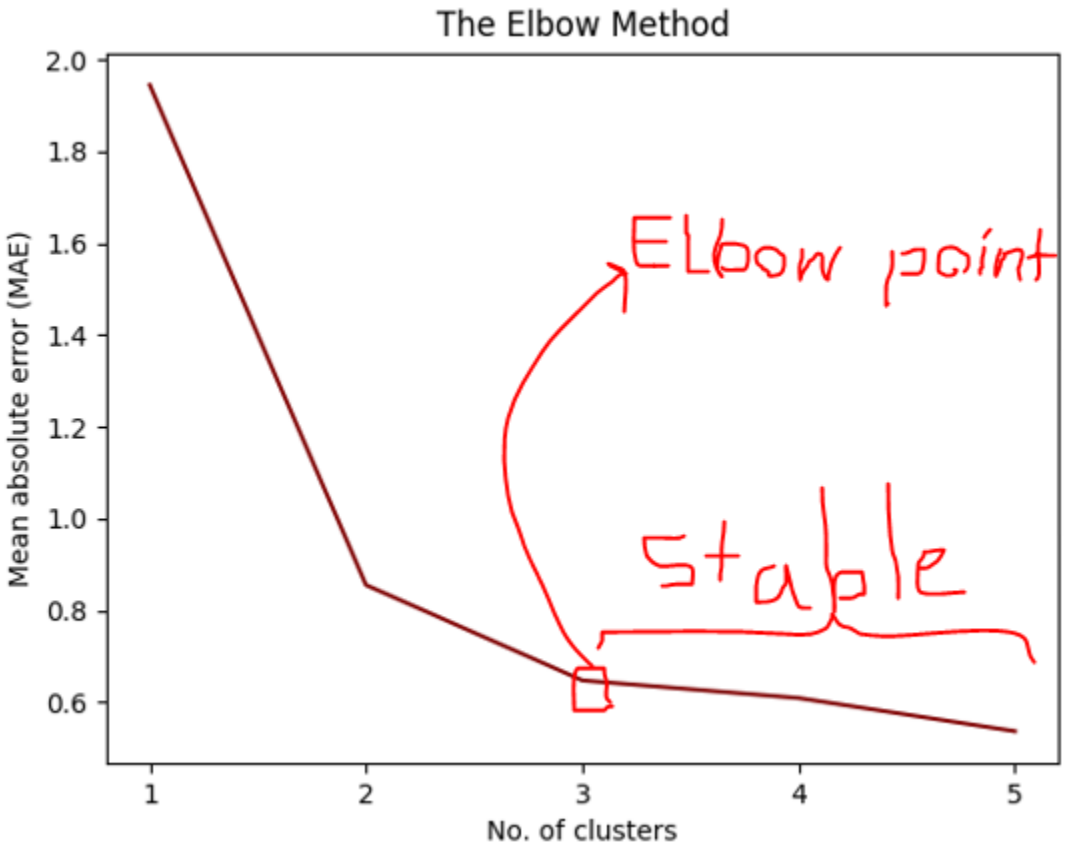
پاسخ 4

برای استفاده از روشِ elbow همانطور که در قسمتِ تئوری توضیح دادیم، نیاز داریم که منحنی را نه به صورتِ میله‌ای بلکه به صورتِ پیوسته رسم کرده و نقطه‌ی زانویی یا شکست را درآوریم.



به نظر می‌رسد که برای تعداد بیشتر از سه کلاستر به حالتِ stable ای می‌رسد و در واقع نقطه‌ی زانویی را تعداد 3 کلاستر در نظر می‌گیریم. لذا مقدار مناسب برای k طبقِ روشِ elbow برابر با 3 می‌باشد. (در این شکل k=2 یک شیب تند را هنوز دارا

می‌باشد و نمی‌توان دقیق گفت که این نقطه زانویی است و به نظرم باید طبقِ این روش stable بودن را نیز در نظر گرفت که احتمالا k=3 جوابِ این روش برای تعیین تعداد کلاسترها می‌باشد.

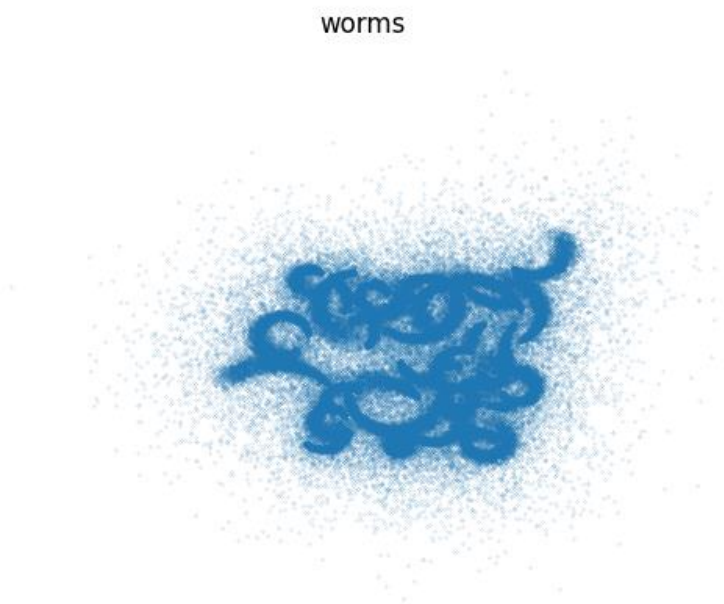


پاسخ 5

شکل را در کد kmeansWorms.py در قسمتِ تعریفِ پارامترهای کلاس رسم نمودیم.

```
class KmeansWorms:
    def __init__(self ,file, cluster_num , max_itr):
        with open(file, newline='') as csvfile:
            data = np.asarray(list(csv.reader(csvfile)))
        self.data = data
        self.output = []
        self.cdata = self.data[np.array([x for x in range(1,self.data.shape[0])]),:]
        self.pure_data = self.cdata[:,np.array([x for x in range(1,self.data.shape[1])])]
        self.C =[]
        self.cluster_num = cluster_num
        self.max_itr = max_itr
        self.clusters = []
        self.x = []
        self.y = []

        plt.figure()
        plt.scatter(self.pure_data[:,0].astype(float), self.pure_data[:,1].astype(float), s=0.01)
        plt.title('worms')
        plt.axis('off')
        plt.show()
```

در نهایت خروجی به شکل زیر در آمد (توجه کنید که برای جلوگیری از همگ کردنِ کامپیوتر بدلیلِ دیتاهای زیاد، axis را در مد off قرار دادم تا cpu به آن نپردازد)



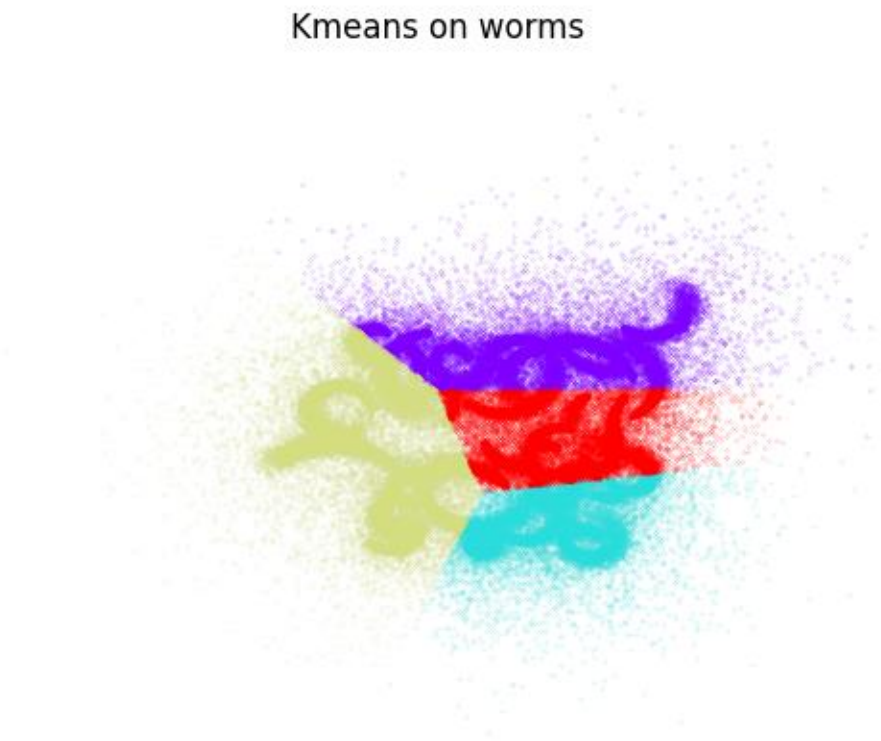
پاسخ 6

در این قسمت که کد این بخش و بخش قبل هر دو در KmeansWorms.py قرار دارد. آمدیم و تعداد کلاسترها را برابر با چهار و ماکسیمم iteration را برابر با دو قرار دادیم کد به صورت زیر است و از scatter برای رسم استفاده نمودیم.

```
84      def printClusters(self):
85          color=cm.rainbow(np.linspace(0,1,self.cluster_num))
86          plt.figure()
87          for i,c in zip(range(self.cluster_num),color):
88              #print('cluster No.' + str(i+1) + ' : ')
89              #print(self.output_clusters[i])
90              #print('\n')
91              plt.scatter(self.x[i],self.y[i], s=0.01, color=c);
92          plt.axis('off')
93          plt.title('Kmeans on worms')
94          plt.show()
```

نتیجه به صورت زیر در آمد:

چهار کلاستر و max_iteration = 2



مسلما بین مرزِ دو به دو تجمعِ داده ها نمیتوانیم خطوط صافی رسم کنیم و لذا کلاسترینگ با kmeans در این حالت بسیار بد عمل کرده و مطابق شکل فوق می‌شود.

پاسخ 7

در این قسمت که قسمتِ آخر است طبق صورت سوال از کتابخانه‌ی SKlearn بهره بردیم و اقدام به کلاسترینگِ داده‌های worms کردیم.

پارامتر های زیر را با سعی و خطا به صورت زیر در آوردیم:

eps=20, min_samples=36

کد بخش آخر به صورت زیر است که از کتابخانه‌ی SKlearn بهره گرفتیم و از دستور DBSCAN استفاده کردیم و آن را به تابع fit نمودیم و در آخر سر با scatter آن را plot کردیم.

```
import csv
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import cm
import seaborn as sns, matplotlib.pyplot as plt, operator as op
from sklearn.cluster import DBSCAN

class DBSCANworms:
    def __init__(self ,file, cluster_num , max_itr):
        with open(file, newline='') as csvfile:
            data = np.asarray(list(csv.reader(csvfile)))
        self.data = data
        self.cdata = self.data[np.array([x for x in range(1,self.data.shape[0])]),:]
        self.pure_data = self.cdata[:,np.array([x for x in range(1,self.data.shape[1])])]
        f = list(zip(self.pure_data[:,0].astype(float), self.pure_data[:,1].astype(float)));
        #print(f)

        clustering = DBSCAN(eps=20, min_samples=36).fit(f)
        y_pred = clustering.fit_predict(f)
        f = np.array(f)
        plt.scatter(f[:,0], f[:,1],c=y_pred, cmap='Paired',s=0.002)
        plt.title("DBSCAN")
        #print(clustering.labels_)

        plt.show()


def main():
    model = DBSCANworms('datasets/worms.csv',4,2)


main();
```

خروجی به صورت زیر در آمد: