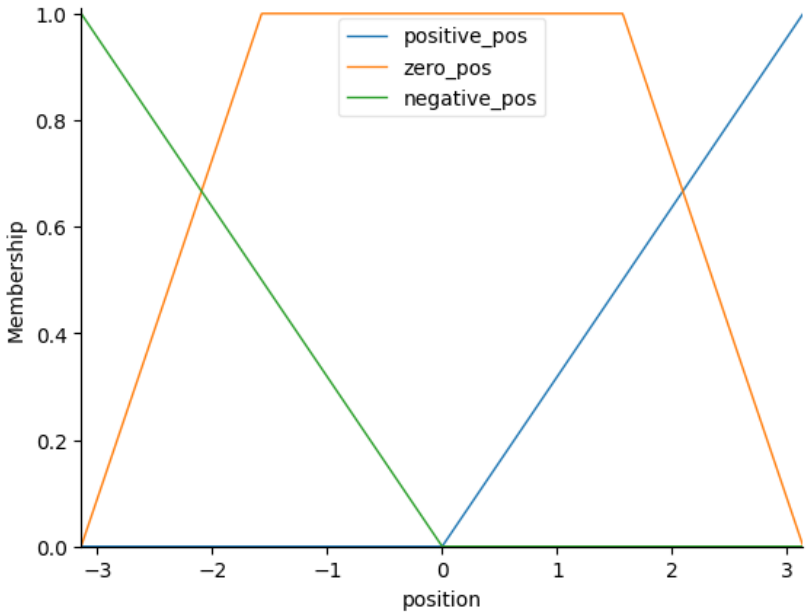
به نام خدا

هانیه اسعدی 99521055

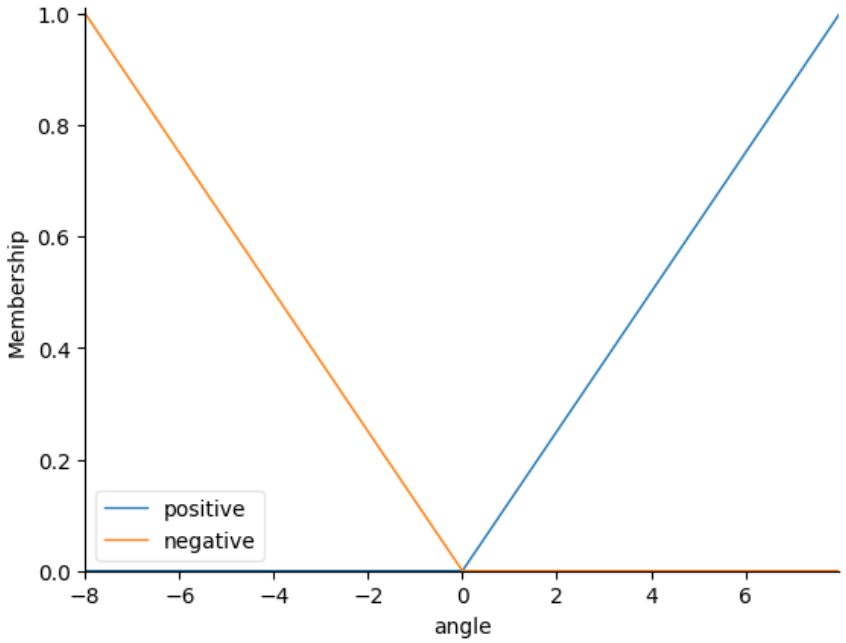
تمرین پنجم هوش محاسباتی

1- متغیر position را برای موقعیت میله تعریف می‌کنیم. که در بازه π- تا π با گام‌های 0.01 تعریف شده است.

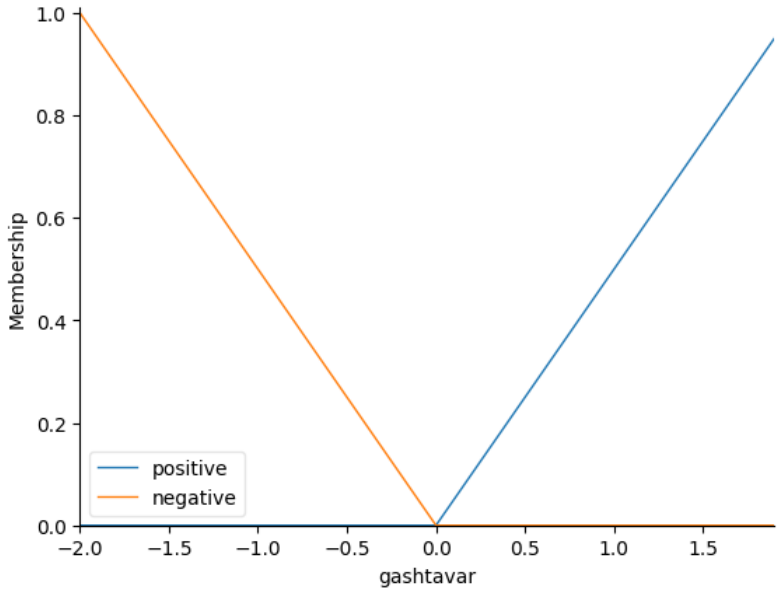
موقعیت‌های مثبت یک تابع مثلثی با نقاط 0 و π است. موقعیت صفر، یک تابع مستطیلی با نقاط π- ، π/2-، π/2وπ است. موقعیت منفی نیز تابع مثلثی با نقاط π- و صفر است.



متغیر angle را برای سرعت زاویه‌ای بین -8 تا 8 و با گام‌های 0.01 تعریف می‌کنیم. سرعت زاویه‌ای مثبت، یک تابع مثلثی بین صفر تا 8 و سرعت زاویه‌ای منفی یک تابع مثلثی بین -8 تا صفر خواهد بود.



متغیر گشتاور را نیز -2 تا 2 و با گام‌های 0.1 تعریف می‌کنیم. گشتاور مثبت یک تابع مثلثی با نقاط 0 و 2 و گشتاور منفی نیز یک تابع مثلثی با نقاط -2 و صفر.



خواسته مسئله این است که میله بتواند با سرعت کمی، تقریبا به صورت ثابت در حالت عمودی بایستد. به این منظور، باید وقتی میله نزدیک حالت عمودی است و گشتاور به سمت موافق حرکت می‌کند که باعث میشود میله از حالت ثابت در بیاید، گشتاور را به سمت مخالف عوض کنیم تا سرعت کم شود. حال قوانین را این گونه تعریف می‌کنیم:

I. اگر موقعیت مثبت و سرعت زاویه‌ای مثبت باشد، گشتاور مثبت اعمال شود.

II. اگر موقعیت مثبت و سرعت زاویه‌ای منفی باشد، گشتاور منفی اعمال شود.
III. اگر موقعیت صفر و سرعت زاویه‌ای مثبت باشد، گشتاور منفی اعمال شود.
IV. اگر موقعیت صفر و سرعت زاویه‌ای منفی باشد، گشتاور مثبت اعمال شود.
V. اگر موقعیت منفی و سرعت زاویه‌ای مثبت باشد، گشتاور مثبت اعمال شود.
VI. اگر موقعیت منفی و سرعت زاویه‌ای منفی باشد، گشتاور منفی اعمال شود.

کنترلر را با قوانین تعریف شده می‌سازیم.

در این بازی، observation ما، سه عضو دارد که شامل جدول زیر است:

Num	Observation	Min	Max
0	x = cos(theta)	-1.0	1.0
1	y = sin(theta)	-1.0	1.0
2	Angular Velocity	-8.0	8.0

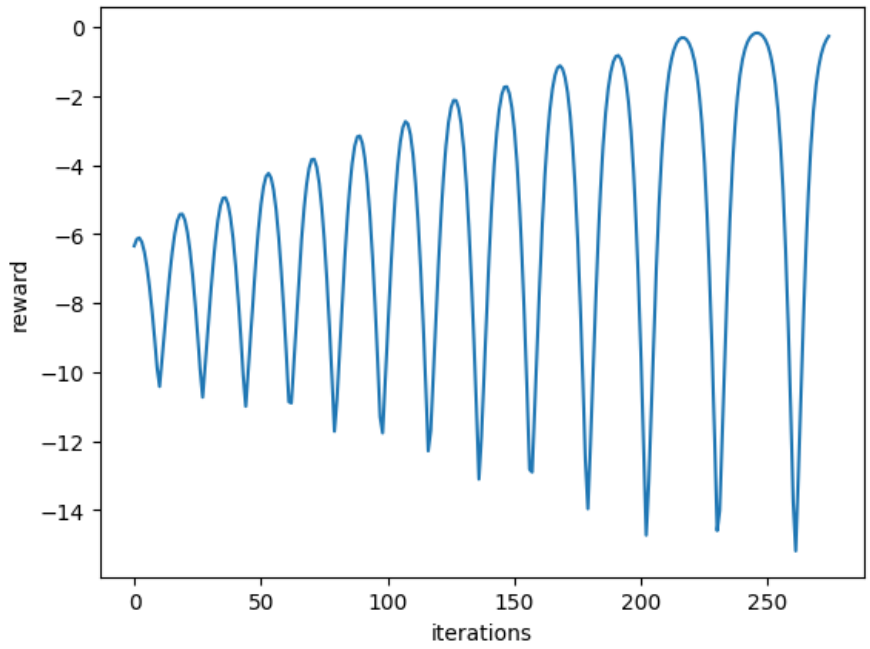
با استفاده از x و y این قسمت، موقعیت پاندول را حساب می‌کنیم. ابتدا با استفاده از تابع atan2 درجه رادیانی را از نقطه (0,0) به نقطه‌ی (x, y) می‌رسانیم. سپس با بازه بندی مناسب، اگر مقدار رادیان بین π- تا π/2- باشد، آن را با 3π/2 جمع می‌کنیم. و اگر از π/2- بزرگتر باشد، آن را با π/2- جمع می‌کنیم.

حال در هر iteration مقدار reward را نیز ذخیره می‌کنیم تا در نهایت بتوانیم نمودار آن را بکشیم.

به منظور سهولت در حل مسئله، اگر موقعیت میله به بیشتر از 0.99 برسد و گشتاور نیز کمتر از 1.5 باشد، مسئله را حل شده در نظر می‌گیریم.

با توجه به نمودار rewards دیده می‌شود که در ابتدا که از نقطه ایده آل خیلی فاصله داریم، مقدار پاداش هم کم است. اما به مرور زمان که گشتاور و موقعیت به جالت ایده آل نزدیک می‌شوند، مقدار reward نیز افزایش می‌یابد.

با توجه به اینکه مقدار اولیه به صورت رندوم تولید می‌شود، ممکن است در بعضی اجراهای کد، در کمتر از 10 iteration اول، به حالت ایده آل رسیده باشیم و پنجره نمایش پاندول سریع بسته شود. بهتر است چندین بار اجرا کنیم تا درستی کد دیده شود.



2- الف) الگوریتم C-Means یک الگوریتم خوشه بندی بدون ناظر است که به پارامتر m بستگی دارد که با درجه مبهم بودن جواب مطابقت دارد. اگر مقدار پارامتر m را خیلی بزرگ انتخاب کنیم، باعث محو شدن کلاس ها می‌شود و انگار همه دیتاها به همه کلاسها تعلق دارند. راه حل های مشکل بهینه سازی به پارامتر m بستگی دارد. یعنی مقادیر مختلف m منجر به خوشه بندی های متفاوت خواهد شد. این الگوریتم نسبت به نسخه الگوریتم خوشه بندی hard-treshold که به هر نقطه دقیقا یک لیبل می‌دهد، قوی‌تر است. در این الگوریتم به هر نقطه از دیتا، یک مقدار membership به هر مرکز خوشه اختصاص داده می‌شود که بر اساس فاصله بین نقطه و مرکز خوشه است. پس هر چه نقطه به مرکز خوشه نزدیک باشد، مقدار عضویت بیشتر می‌شود. مجموع عضویت هر نقطه داده نیز برابر یک می‌شود.

تفاوت C-Means با K-Means :

I. انتساب به یک خوشه: در خوشه بندی فازی، هر نقطه از دیتا، یک درصد تعلق به هر خوشه به خود می‌گیرد. اما در خوشه بندی K-Means هر نقطه به صورت مطلق به یک خوشه تعلق دارد. در خوشه بندی فازی، هر نقطه دارای وزنی مرتبط با یک خوشه خاص است فلذا یک نقطه به اندازه ارتباط ضعیف یا قوی با خوشه که با فاصله معکوس تعیین می‌شود، در یک خوشه قرار نمی‌گیرد.
II. سرعت: خوشه بندی فازی نسبت به K-Means با توجه به اینکه عملیات بیشتری انجام می‌دهد، سرعت کمتری دارد. هر نقطه نسبت به هر خوشه ارزیابی می‌شود و این ارزیابی عملیات بیشتری در پی دارد. در حالی که در K-Means فقط کافی ست فاصله را محاسبه کنیم اما در C-Means باید وزن دهی کامل با فاصله معکوس انجام دهیم.
III. خوشه بندی فازی نسبت به K-Means کمتر احمقانه است که به خوشه‌های کشیده می‌رسد.

ب) 1)

```
data = pd.read_csv('data1.csv')
✓ 0.0s
```

2) ستون‌های اول و دوم را برای نرمال سازی انتخاب می‌کنیم. سپس با استفاده از تابع fit_transform از StandardScaler دو ستون را استاندارد کرده و در دیتای ابتدایی جایگزین می‌کنیم.

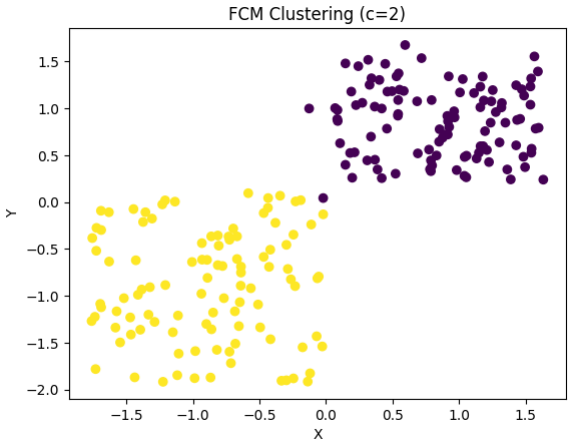
```
X_y = data[['X', 'Y', 'Class']]
X = X_y[['X', 'Y']]

scaler = StandardScaler()
X_normalized = scaler.fit_transform(X)

data[['X', 'Y']] = X_normalized
print(data)
✓ 0.0s
```

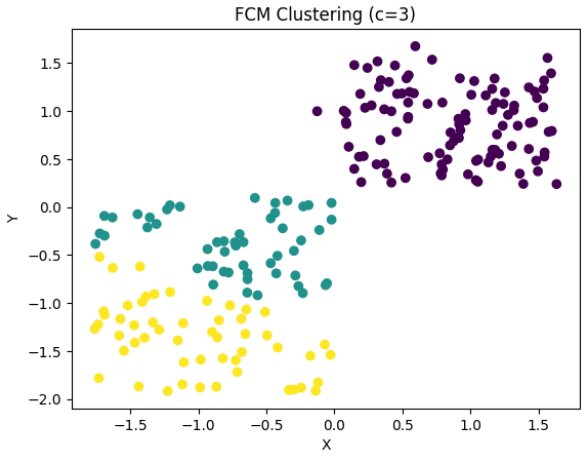
دیتای ستون‌ها پس از تغییرات:

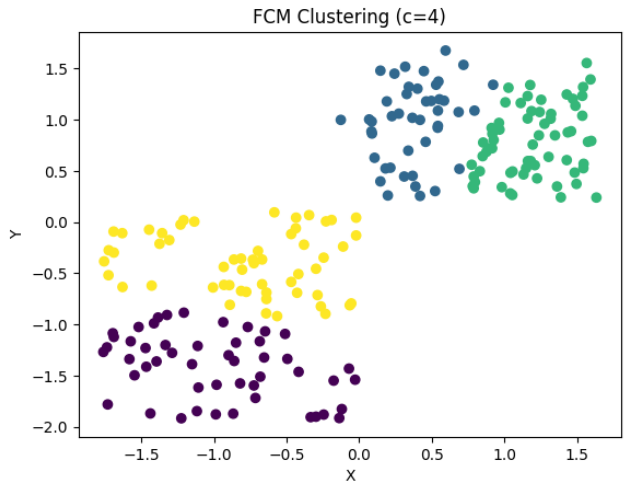
```
            X         Y  Class
0   -0.813747 -0.357511      1
1   -0.126986  0.998022      1
2   -0.725701 -0.402696      1
3    0.418901  0.998022      0
4   -0.813747 -0.673802      1
..        ...       ...    ...
207  0.457641  0.783396      0
208  0.197024  0.259256      0
209  0.797500  0.390291      0
210  0.830958  0.496475      0
211  1.243014  1.074835      0

[212 rows x 3 columns]
```

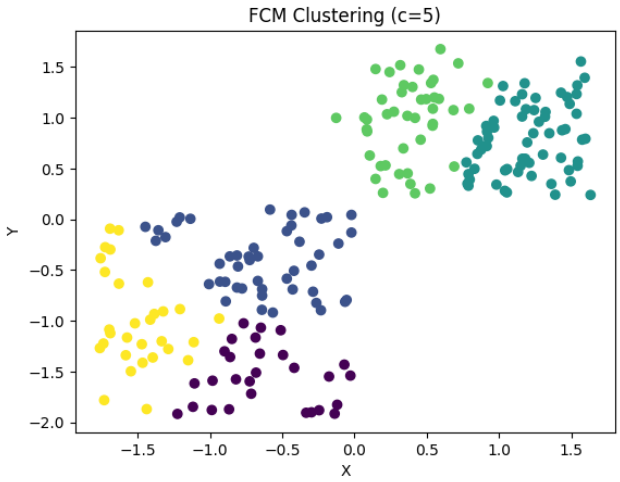
3) مقدار FPC ها را در هر کلاس بندی ذخیره میکنیم تا در قسمت بعدی نمودار آن را رسم کنیم. به ازای c های مختلف، خوشه بندی را انجام می‌دهیم و در هر مرحله، بهترین خوشه بندی ای که تا این مرحله بدست آورده ایم را ذخیره می‌کنیم. سپس نمودار حاصل از خوشه بندی را برای cهای مختلف رسم می‌کنیم.

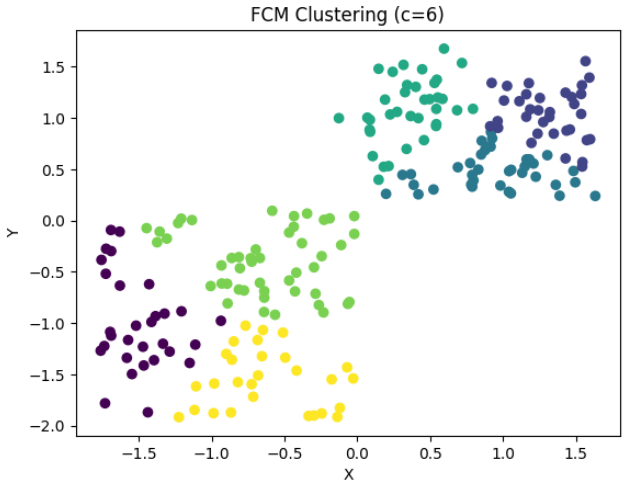
```python
fpc_values = []
best_c = 0
best_fpc = -1

def run_fcm_and_plot(X, c):
    global best_c, best_fpc
    cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(X.T, c, 2, error=0.005, maxiter=1000, init=None)
    cluster_membership = np.argmax(u, axis=0)

    fpc_values.append(fpc)

    if fpc > best_fpc:
        best_fpc = fpc
        best_c = c

    plt.scatter(X['X'], X['Y'], c=cluster_membership, cmap='viridis')

    plt.title(f'FCM Clustering (c={c})')
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.show()

for c in range(2, 11):
    run_fcm_and_plot(data[['X', 'Y']], c)

c_values = list(range(2, 11))
plt.plot(c_values, fpc_values, marker="o")
plt.title('Fuzzy partition coefficient (FPC) for different cluster counts')
plt.xlabel('Number of clusters(c)')
plt.ylabel('FPC')
plt.show()

print(f'The best number of clusters (c) based on FPC is: {best_c}')
```

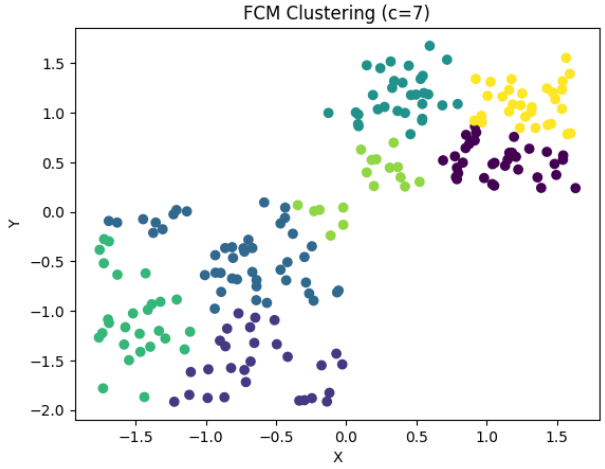


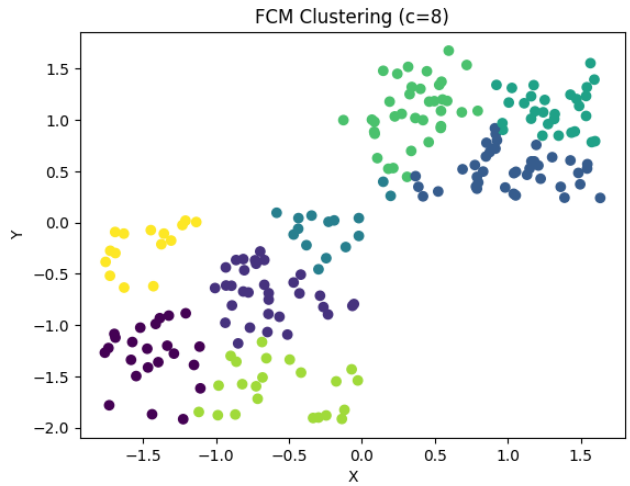


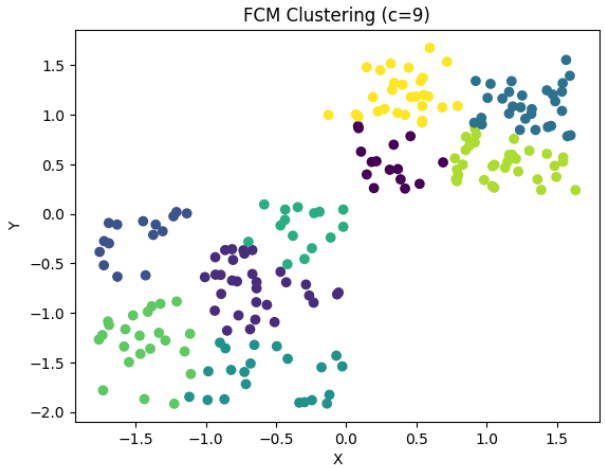

FCM Clustering (c=4)
X
Y

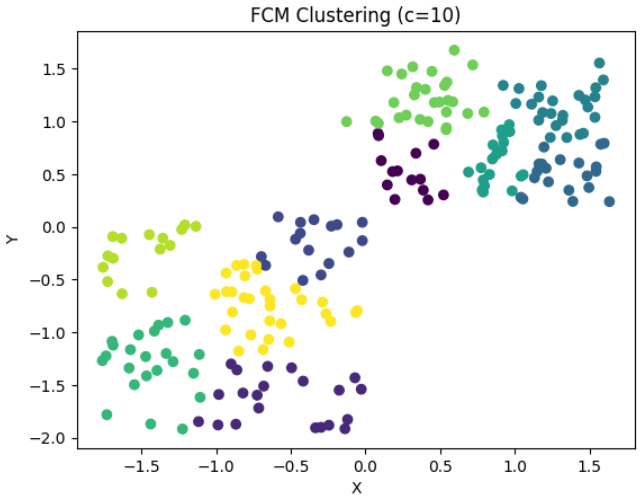

FCM Clustering (c=5)
X
Y


FCM Clustering (c=6)
X
Y

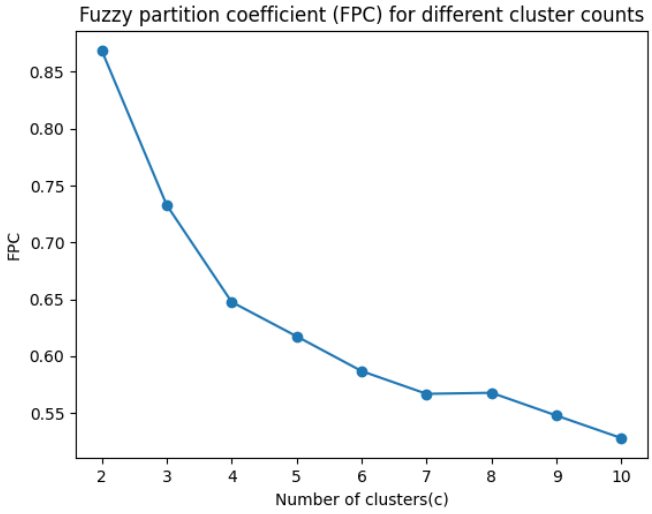

FCM Clustering (c=7)
X
Y


FCM Clustering (c=8)
X
Y


FCM Clustering (c=9)
X
Y



نمودار FPC بر اساس تعداد خوشه به صورت روبرو است:



4) معیار FPC یک معیار اندازه‌گیری کیفیت خوشه‌بندی در الگوریتم‌های خوشه‌بندی فازی است. این معیار به دنبال ارزیابی درصد عضویت داده‌ها در خوشه‌ها و همچنین درجه فازی می‌باشد. هدف اصلی این معیار نشان دادن درصد بیشتری از داده‌ها که به یک خوشه تعلق دارند و درجه کمتری از ابهام است.

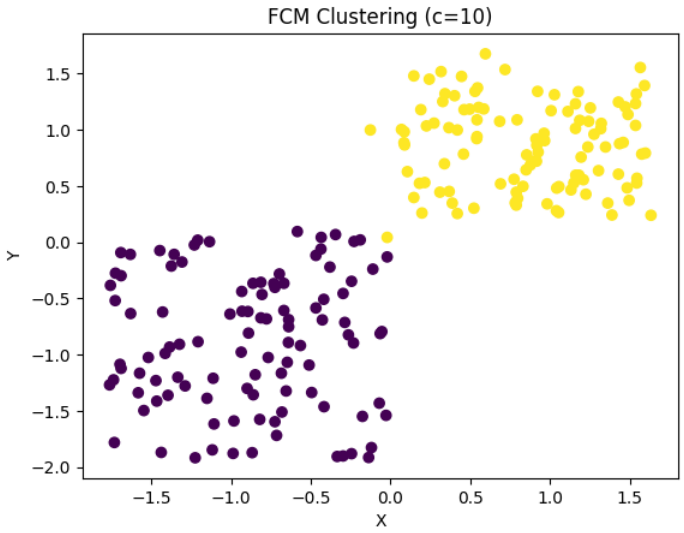
$$FPC = \frac{\sum_{i=1}^{c}\left(\sum_{j=1}^{N} u_{ij}^{m}\right)^{2}}{\sum_{i=1}^{c}\sum_{j=1}^{N}\left(u_{ij}^{m}\right)^{2}}$$

فرمول معیار FPC به صورت روبرو است که در آن، N تعداد داده ها، c تعداد خوشه ها، m پارامتر فازی سازی، u_{ij} درصد عضویت داده زام در خوشه iام است. با استفاده از این فرمول، فرمول FPC از مربع مجموع درصدهای عضویت در هر خوشه برای هر داده استفاده می‌کند. سپس مقدار حاصل این مربعات را برای هر خوشه جمع می‌کند. در نهایت، مقدار FPC نسبت این مجموع مربعات به

مجموع مربعات کلی از درصدهای عضویت را نشان می‌دهد. این مقدار نزدیک به 1 نشان‌دهنده یک خوشه‌بندی دقیق و با کمترین ابهام است.

به عبارت دیگر، FPCتلاش می‌کند اطلاعات درصدهای عضویت را در خوشه‌بندی در نظر گرفته و به صورت خلاصه نشان دهد که چقدر داده‌ها به طور قطعی در یک خوشه تعلق دارند و درجه فازی آنها چقدر است.

در دیتاهای اول، بهترین مقدار FPC با تعداد خوشه بندی 2 است که نمودار خوشه بندی آن به صورت زیر است:



حال همه مراحل بالا را بار دیگر برای data2 طی میکنیم:

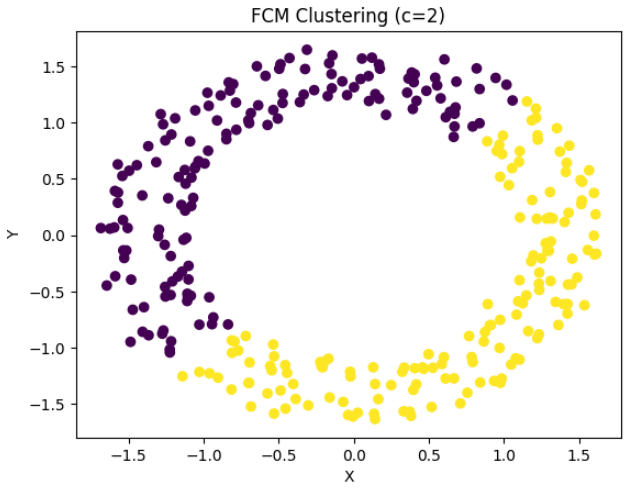
خواندن دیتا و نرمال سازی آن و نشان دادن دیتا پس از نرمال سازی:

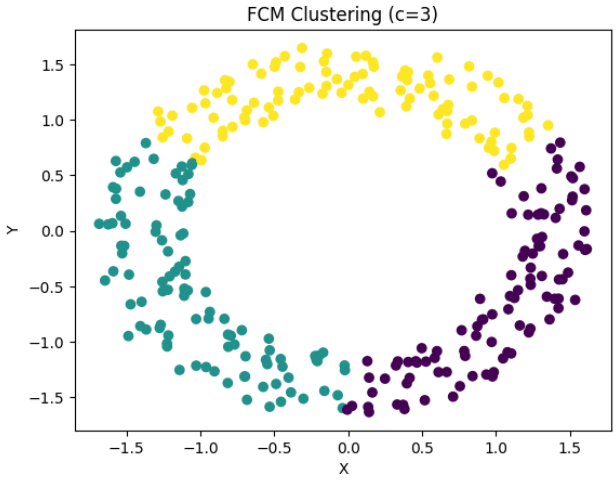
```
data2 = pd.read_csv('data2.csv')

X_y = data2[['X', 'Y', 'Class']]
X = X_y[['X', 'Y']]

scaler = StandardScaler()
X_normalized = scaler.fit_transform(X)

data2[['X', 'Y']] = X_normalized
print(data2)
```

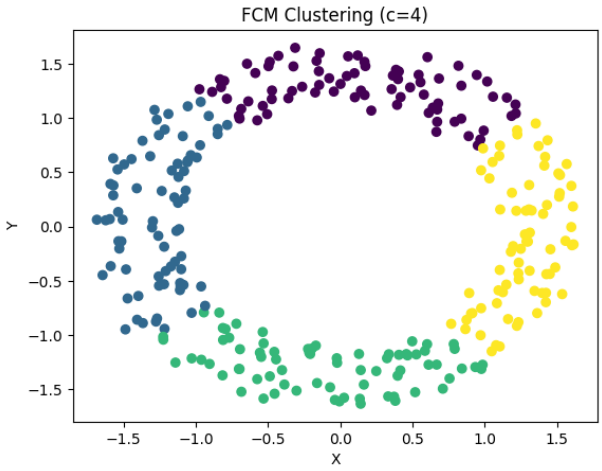
✓ 0.0s

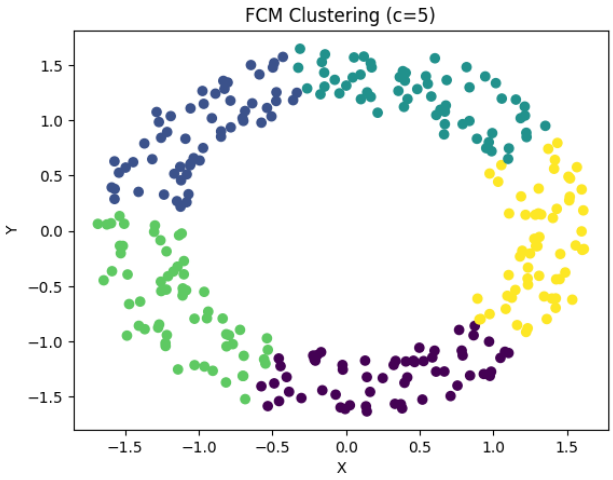
```
           X         Y  Class
0   -1.315522  0.649213      0
1    0.141025 -1.323862      1
2   -1.506091  0.064147      0
3   -0.264216  1.289819      1
4    0.417045  1.194334      1
..        ...       ...    ...
295 -0.825294  1.286212      0
296  1.284232  0.146129      1
297 -1.406355 -0.859190      0
298 -0.837684 -0.793684      1
299 -1.482871 -0.394832      0

[300 rows x 3 columns]
```

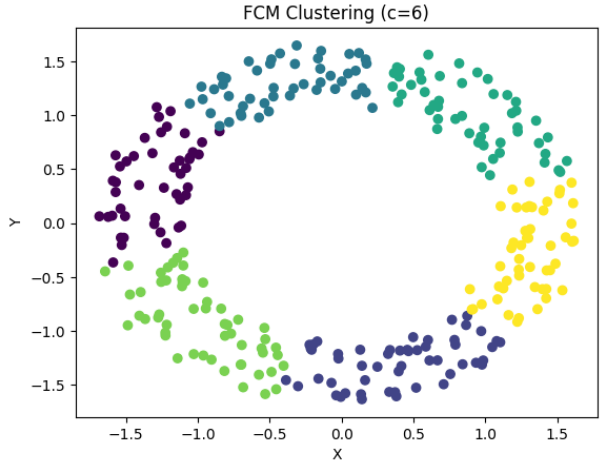
اعمال الگوریتم C-Means به ازای مقادیر مختلف c از بازه‌ی 2 تا 10:

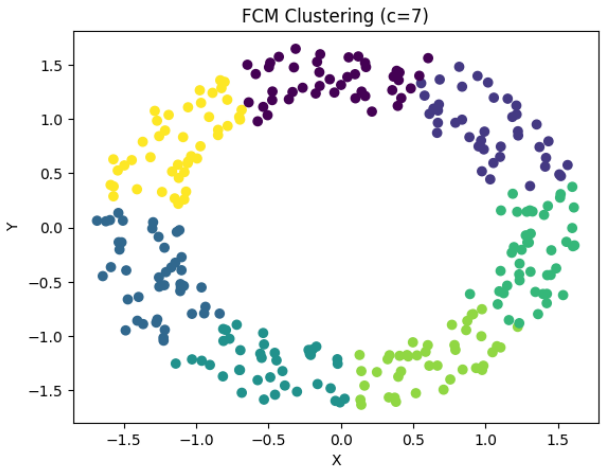


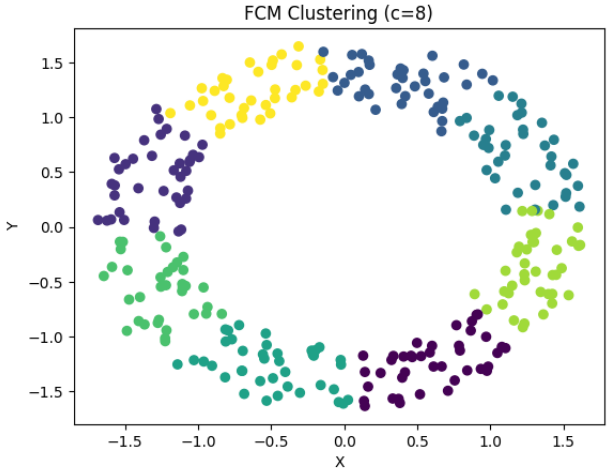


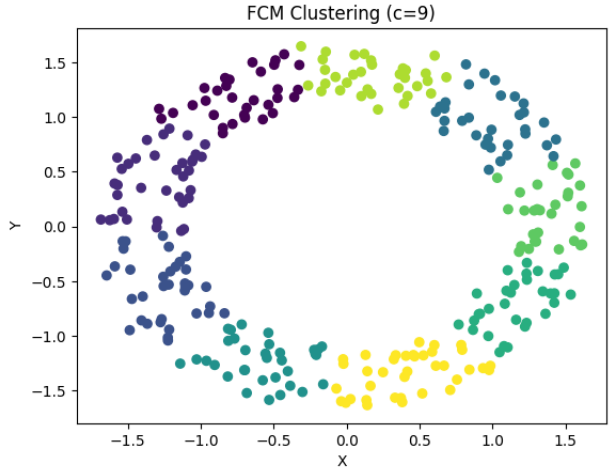

FCM Clustering (c=4)
X
Y

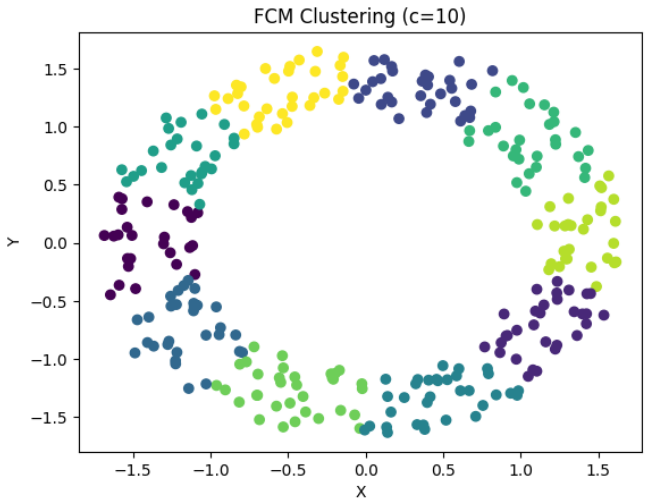

FCM Clustering (c=5)
X
Y

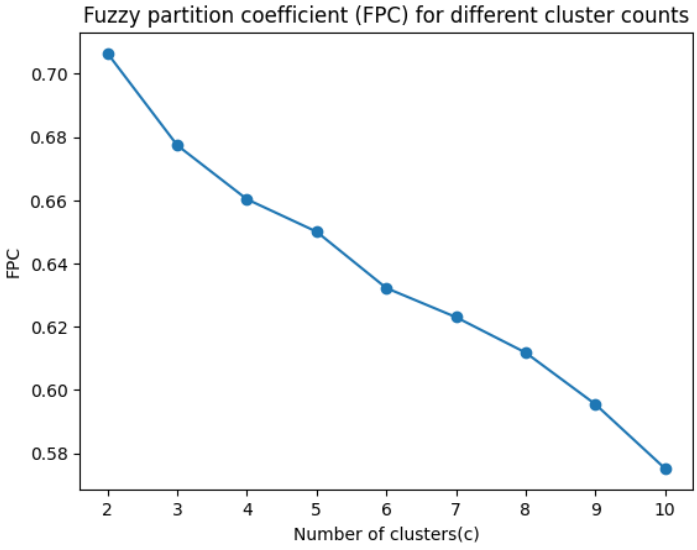

FCM Clustering (c=6)
X
Y


FCM Clustering (c=7)
X
Y


FCM Clustering (c=8)
X
Y

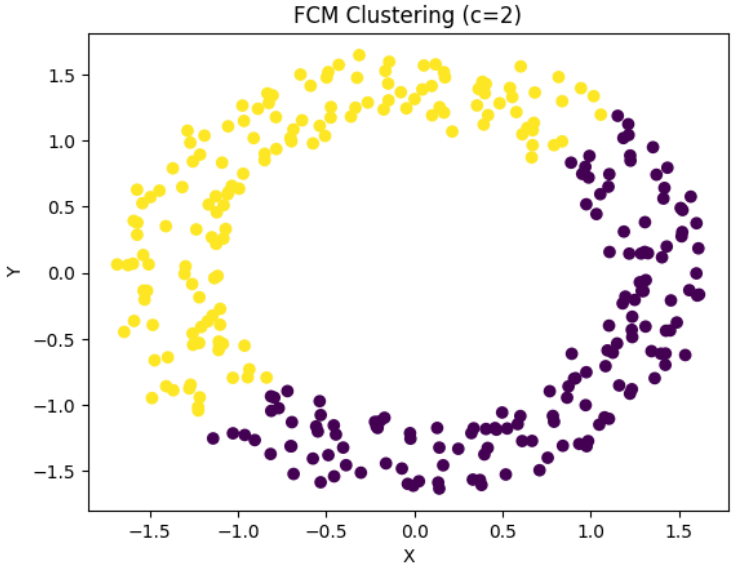

FCM Clustering (c=9)
X
Y





نمودار FPC بر اساس تعداد خوشه:

و نمودار بهترین خوشه بندی به دست آمده:



دو نفر با وزن های 55 و 95 و سن 45 و 60 سال:

الف) نفر دوم نسبتا چاق تر و جوان تر از نفر اول است.

این عبارت حاصل دو عبارت And است.

هر عبارت را جداگانه بدست می آوریم.

نفر اول: $\mu_{fat} = 1-\left(\frac{55-150}{100}\right)^2 = 1-(0/95)^2 = /975$

نفر دوم: $\mu_{fat} = 1-\left(\frac{95-150}{100}\right)^2 = /6975$

برای ترم نسبتا، از هر مقدار جذر آن را میگیریم.

نفر اول نسبتا چاق: /312   نفر دوم نسبتا چاق: /835

نفر اول: $\mu_{young} = \left(1+\left(\frac{45-25}{5}\right)^2\right)^{-1} = 0/588$

نفر دوم: $\mu_{young} = \left(1+\left(\frac{60-25}{5}\right)^2\right)^{-1} = /02$

برای 8 به and، از تفاضل موجود حساب ضرایب همینهم حساب می‌کنیم.

$$\mathrm{Min}(0.835 - 0.312, (0.2 - 0.588)) =$$
$$\mathrm{Min}(0.523, -0.388) = -0.388$$

که چون این مقدار کوچکتر از صفر است پس این عبارت صحیح نیست.

ب) اگر فرد اول خیلی لاغر باشد، آنگاه نفر دوم نسبتا جوان است.

با استفاده از قاعده modus pones، به عبارت زیر می‌رسیم:

نفر اول خیلی لاغر نیست و نفر دوم نسبتا جوان است.

نفر اول: $\mu_{thin} = \left(1 + \left(\frac{55-25}{5}\right)^2\right)^{-1} = 0.027$

نفر دوم: $\mu_{thin} = \left(1 + \left(\frac{95-25}{5}\right)^2\right)^{-1} = 0.005$

برای نفر دوم خیلی لاغر، مقادیر را به توان ۲ می‌رسانیم:

نفر اول خیلی لاغر: 0.00729

برای نفر دوم نسبتا جوان هم، ابتدا مقدار μ_{young} را برای نفر دوم به دست می‌آوریم که طبق قسمت قبل 0.02 است پس از آن جذر می‌گیریم.

نفر دهم سینا جوان : ۰/۱۴۱

همانند قسمت قبل چون فقط And در عبارت داریم
از بین آنها مینیمم میگیریم.

$Min(1 - 0/00729, 0/141) = 0/141$

۰/۹۹۲۷۱

پس این عبارت درست است.