نام خدا
هوم ورک ششم درس شبکه عصبی
مهدی فقهی
۴۰۱۷۲۲۱۳۶

سوال اول :

الف.به طور کلی مراحل طراحی یک کنترل کننده فازی را توضیح دهید.

(10 امتیاز)

مرحله اول) اول متغیرهای ورودی و خروجی را مشخص کنیم و ترم‌های زبانی را باید به صورت فازی مشخص کنیم .

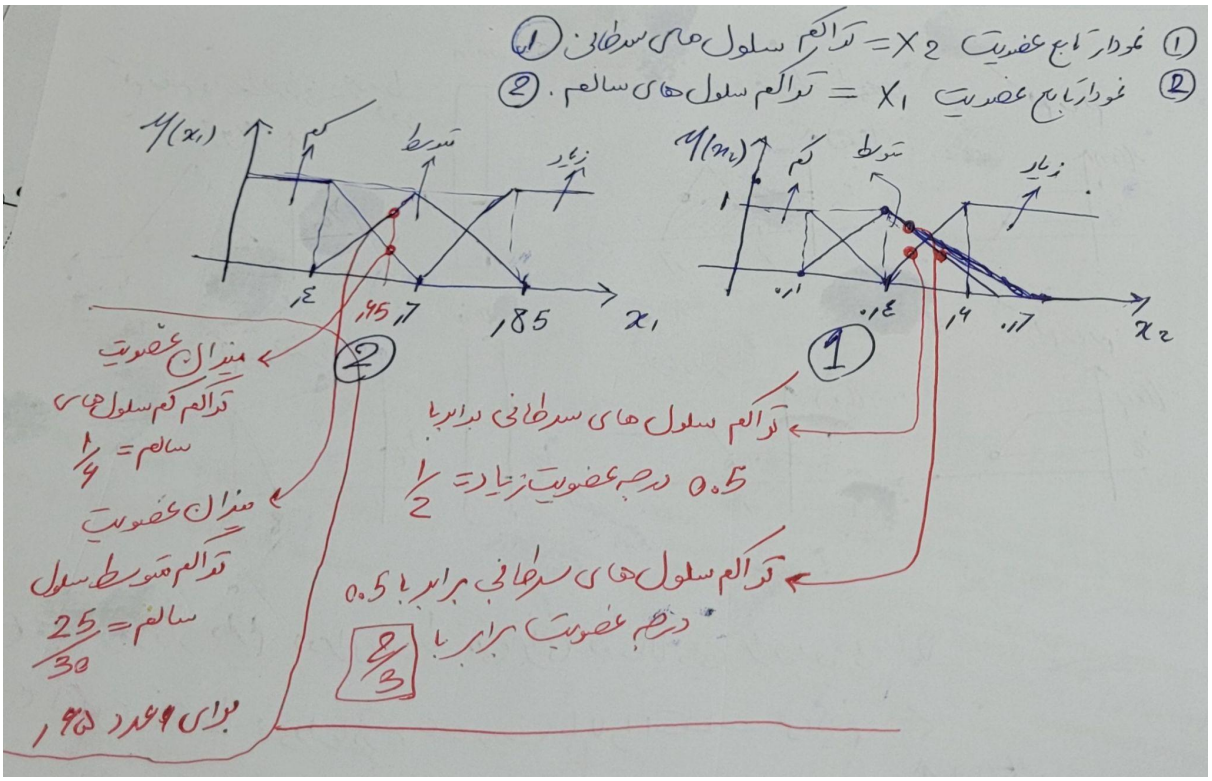
مرحله دوم) سپس در مرحله بعد باید fuzzification function را برای هر ورودی مشخص کنیم .

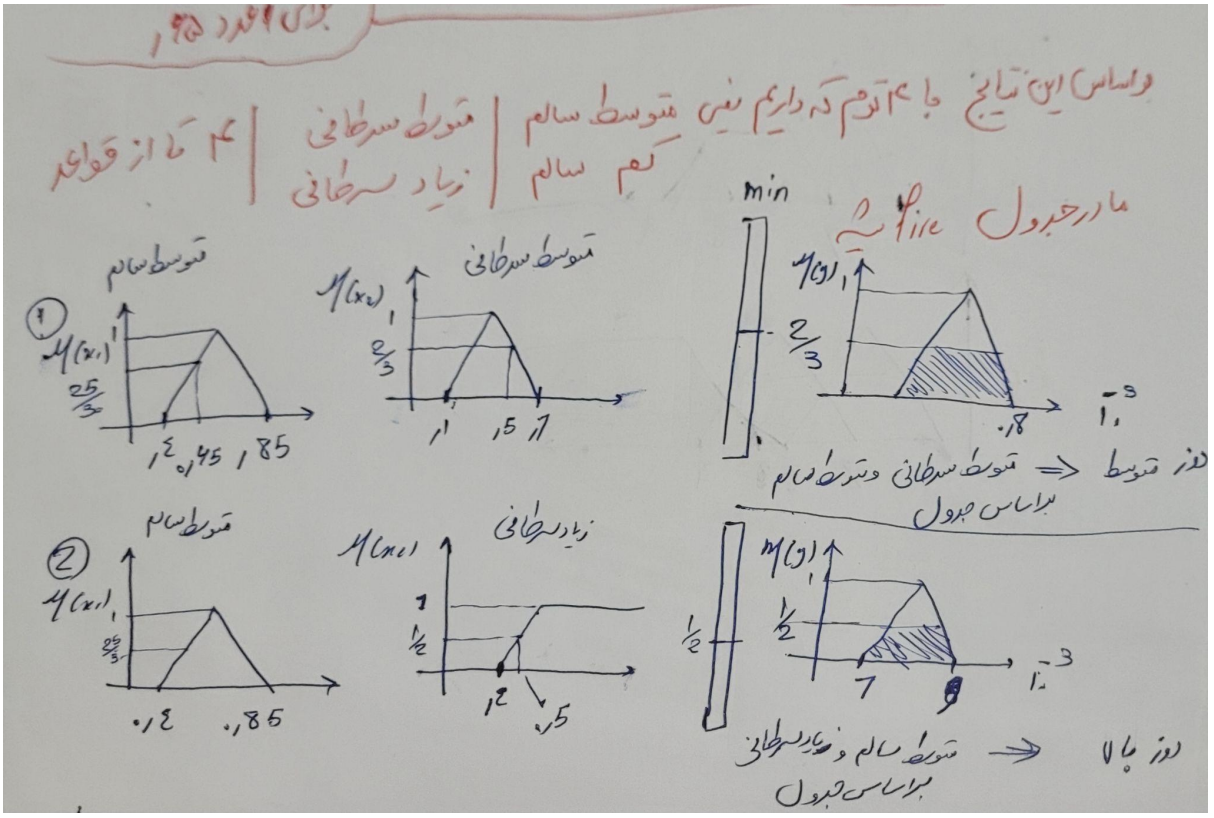
مرحله سوم) باید قواعد حاصل از ترم‌ها را به کمک متخصص امر استخراج کنیم .

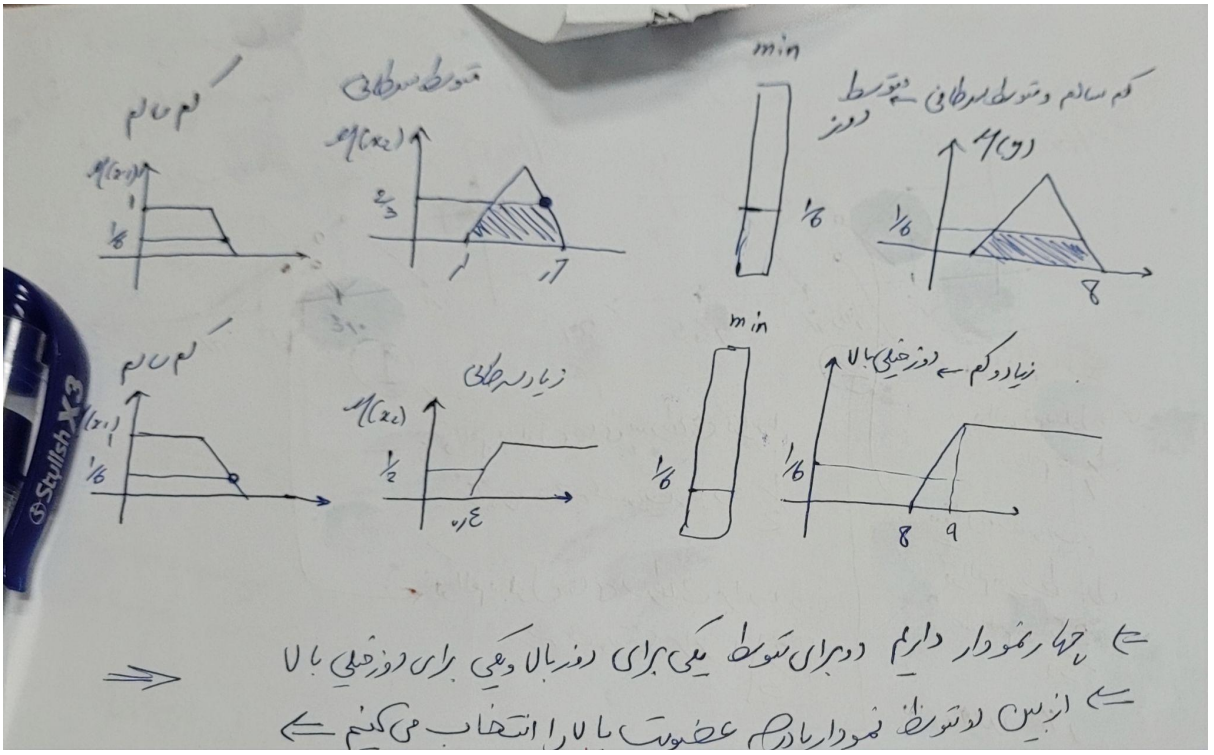
مرحله چهارم) باید inference engine خودمون را برای این مسائل مشخص کنیم و محاسبات و اعمالی چون اشتراک اجتماع و ... را به کمک آنان انجام دهیم.

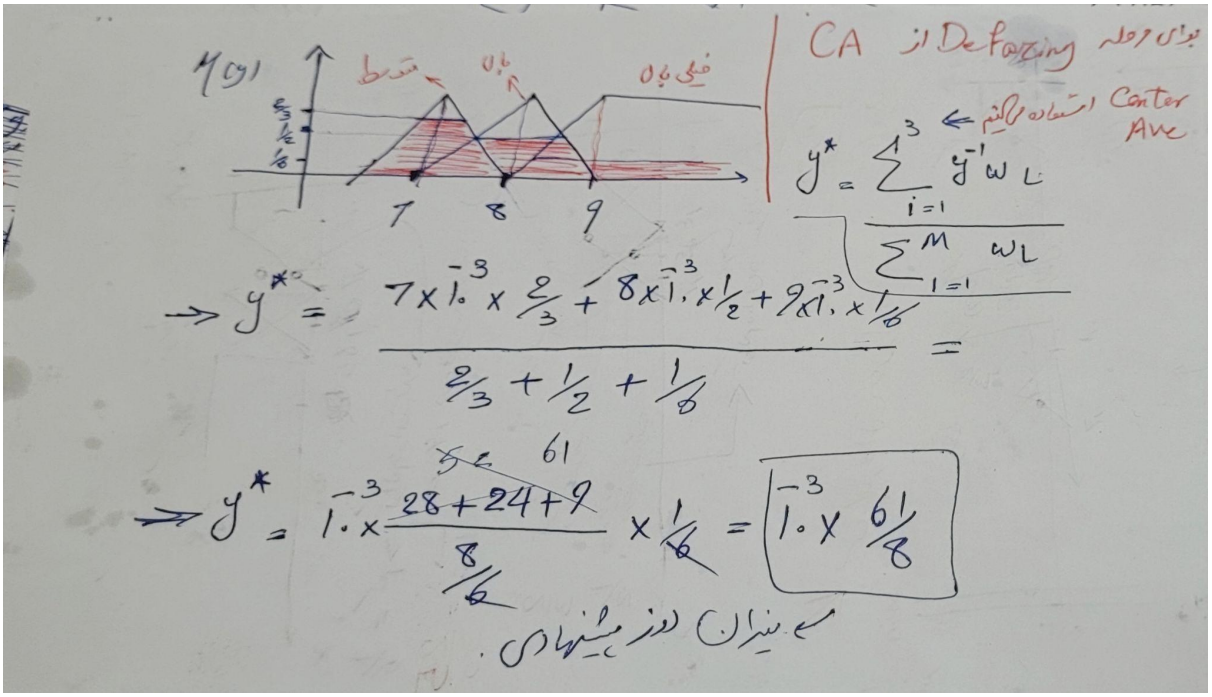
مرحله پنجم) باید defuzzification انتخاب کنیم که قاعده ذهنی و فازی بدست آمده را به یک عدد تبدیل کنیم .

قسمت ب و پ) درون عکس :




براساس اعداد ۱۹۵
و اساس این نمایخ با ۴ اتم که داریم نفی متوسط سالم
کم سالم
متوسط سرطانی
زیاد سرطانی
۴ تا از قواعد
min
متوسط سالم
متوسط سرطانی
متوسط سالم
زیاد سرطانی
دوز متوسط ⇒ متوسط سرطانی و متوسط سالم
براساس جدول
دوز بالا ⇒ متوسط سالم و زیاد سرطانی
براساس جدول


کم سالم
متوسط سرطانی
min
کم سالم و متوسط سرطانی ⇒ دوز متوسط
کم سالم
زیاد سرطانی
min
زیاد و کم ⇒ دوز خیلی بالا
⇐ چهار نمودار داریم دو برای متوسط یکی برای دوز بالا و یکی برای دوز خیلی بالا
⇐ از بین دو متوسط نمودار بادی عضویت بالا را انتخاب می کنیم ⇐



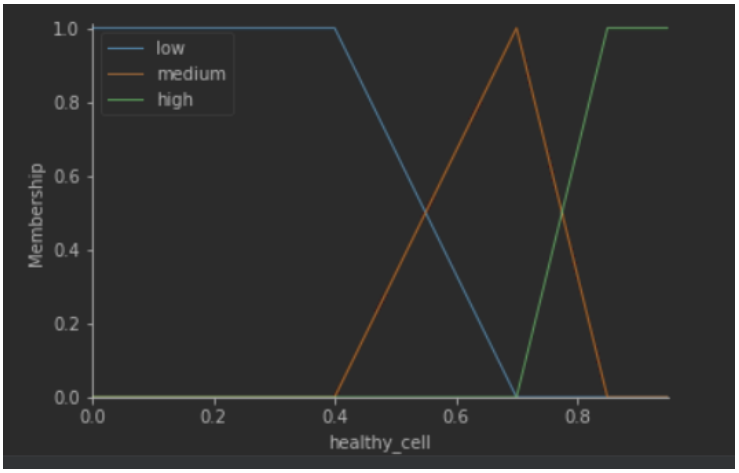
که حاصل تقریبا برابرا با 0.0074 است .

سوال دوم :

ابتدا یک ترم فازی ورودی به نام healthy cell تعریف می‌کنیم که همان ترم تراکم سلولی سالم است که با دقت 0.05 و تعریف شده در بازه 0 تا ۱ .

در قسمت بعد Inference fuzzy را برای قسمت low به صورت ذوزنقه‌ای و برای قسمت medium به صورت مثلی و برای high به صورت ذوزنقه‌ای باز تعریف می‌کنیم و به ترتیب رئوس این اشکال را در ورودی دوم به صورت لیست مشخص می‌کنیم .

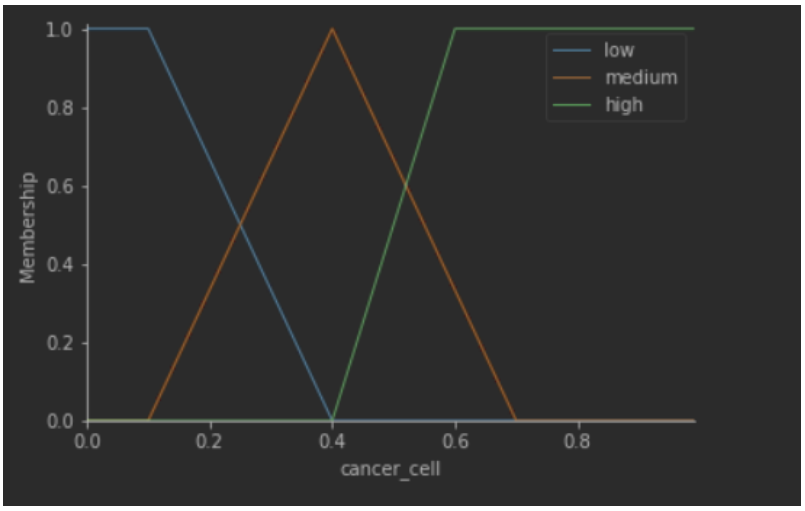
```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

healthy_cell = ctrl.Antecedent(np.arange(0, 1, 0.05), 'healthy_cell')

healthy_cell['low'] = fuzz.trapmf(healthy_cell.universe, [0,0, 0.4, 0.7])
healthy_cell['medium'] = fuzz.trimf(healthy_cell.universe, [0.4,0.7, 0.85])
healthy_cell['high'] = fuzz.trapmf(healthy_cell.universe, [0.7, 0.85,1,1])
healthy_cell.view()
```



شکل حاصل همانند بالا خواهد بود .

همین کار را برای ترم فازی بعدی که تراکم سلولی سرطانی هست را نیز انجام می‌دهیم .

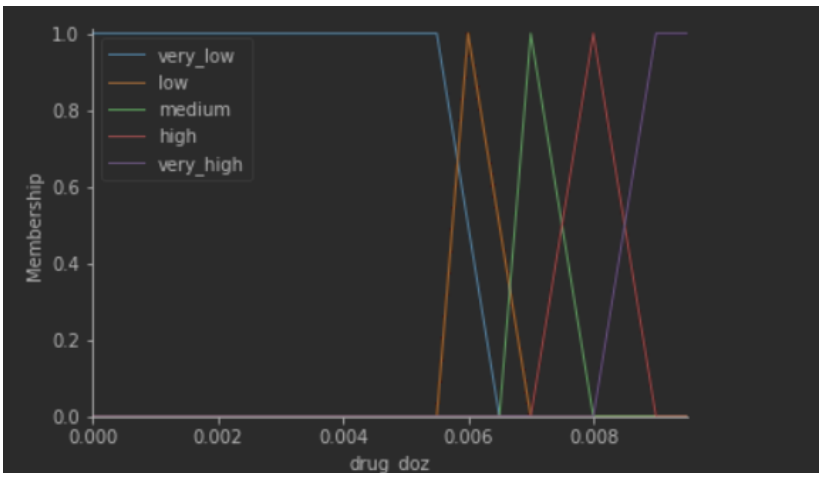
```
#In this part we want to define one of the entery word that name is healthy cell
#We set this item between lower band 0 and upper band 1 with Accuracy  of 0.01

cancer_cell = ctrl.Antecedent(np.arange(0, 1, 0.01), 'cancer_cell')

# Define fuzzy term low and medium and high with shape of trapezius and traingle and trapezius

cancer_cell['low'] = fuzz.trapmf(cancer_cell.universe, [0,0, 0.1, 0.4])
cancer_cell['medium'] = fuzz.trimf(cancer_cell.universe, [0.1,0.4, 0.7])
cancer_cell['high'] = fuzz.trapmf(cancer_cell.universe, [0.4,0.6, 1, 1])
cancer_cell.view()
```



شکل حاصل از تعریف بازه‌ها برای ترم فازی تراکم سلول‌های سرطانی

سپس ترم فازی خروجی را تعریف می‌کنم و همانند قبل شش بازه این ترم فازی را با اشکال ذوزنقه ،مثلث،مثلث،مثلث، ذوزنقه قرار می‌دهیم . بازه‌های بین عدد 0 تا 0.01 با دقت 0.0005 تعریف شده است .

```
#In this part we want to define our result term that name is drug doz
# it between 0,0.01 with accuracy of 0.0005
drug_doz = ctrl.Consequent(np.arange(0, 0.01, 0.0005), 'drug_doz')

# we have six situation for this term .
drug_doz['very_low'] = fuzz.trapmf(drug_doz.universe, [0,0, 0.0055, 0.0065])
drug_doz['low'] = fuzz.trimf(drug_doz.universe,[0.0055,0.006,0.007])
drug_doz['medium'] = fuzz.trimf(drug_doz.universe,[0.0065,0.007,0.008])
drug_doz['high'] = fuzz.trimf(drug_doz.universe,[0.007,0.008,0.009])
drug_doz['very_high'] = fuzz.trapmf(drug_doz.universe,[0.008,0.009,0.01,0.01])
drug_doz.view()
```



شکل ترم فازی خروجی برای تعیین مقدار دوز دارو

سپس قوانین را همانند جدول تعریف می کنیم .

```
# rules of this contrler with or entry and result fuzzy term
rule1 = ctrl.Rule(healthy_cell['low'] & cancer_cell['low'], drug_doz['low'])
rule2 = ctrl.Rule(healthy_cell['medium']& cancer_cell['low'], drug_doz['low'])
rule3 = ctrl.Rule(healthy_cell['high'] & cancer_cell['low'], drug_doz['very_low'])
rule4 = ctrl.Rule(healthy_cell['low'] & cancer_cell['medium'], drug_doz['medium'])
rule5 = ctrl.Rule(healthy_cell['medium']& cancer_cell['medium'], drug_doz['medium'])
rule6 = ctrl.Rule(healthy_cell['high'] & cancer_cell['medium'], drug_doz['low'])
rule7 = ctrl.Rule(healthy_cell['low'] & cancer_cell['high'], drug_doz['very_high'])
rule8 = ctrl.Rule(healthy_cell['medium']& cancer_cell['high'], drug_doz['high'])
rule9 = ctrl.Rule(healthy_cell['high'] & cancer_cell['high'], drug_doz['medium'])
```

سپس در انتها یک کنترلر فازی براساس قوانینی که بر حسب ترم های ورودی و خروجی تعریف کردیم را می‌سازیم .

```
duzz_healthy_ctrl = ctrl.ControlSystem([rule1, rule2, rule3,rule4, rule5, rule6,rule7, rule8, rule9])
```

+ Code + Text

```
duzz_healthy = ctrl.ControlSystemSimulation(duzz_healthy_ctrl)
```

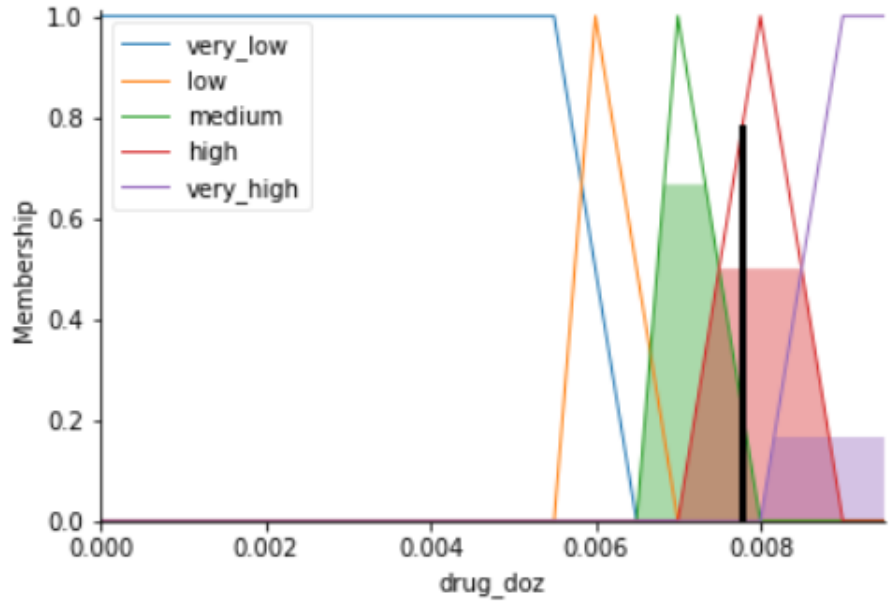
به عنوان ورودی اول تراکم 0.65 و 0.5 به ترتیب سالم و سرطانی را می‌دهیم به این کنترلر .

سپس result را به صورت تصویری می‌بینیم که چه قوانینی و با چه میزان تاثیری در خروجی مشاهده شده است .

```
duzz_healthy.input['healthy_cell'] = 0.65
duzz_healthy.input['cancer_cell'] = 0.5
duzz_healthy.compute()
```

```
"""
Once computed, we can view the result as well as visualize it.
"""
print(duzz_healthy.output['drug_doz'])
drug_doz.view(sim=duzz_healthy)
```

```
0.0077813620071684585
```



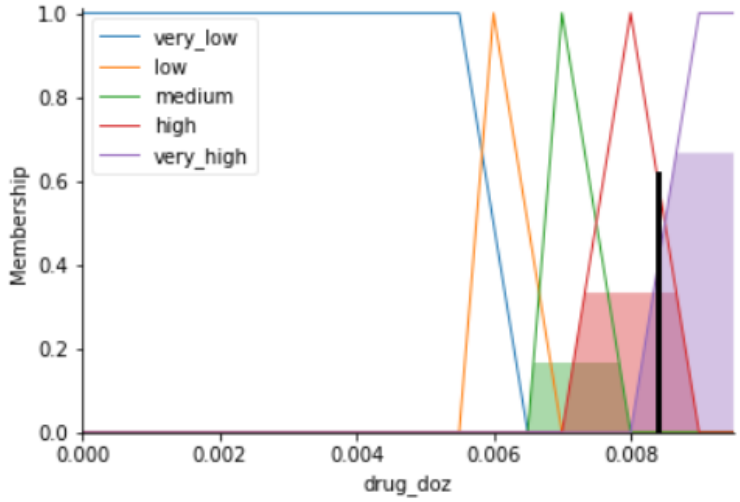
همانطور که مشاهده می‌کنیم میزان دوز بدست آمده از کنترلر فازی ما برابر با 0.0077 که برابر با میزان بدست آمده از سوال قبل نزدیک می‌باشد .

سپس براساس ورودی 0.5 و 0.65 که به ترتیب میزان تراکمی سلول‌های سالم و سرطانی را نشان می‌دهد خروجی سیستم را می‌سنجیم .

```
duzz_healthy.input['healthy_cell'] = 0.5
duzz_healthy.input['cancer_cell'] = 0.65
duzz_healthy.compute()
```

```
"""
Once computed, we can view the result as well as visualize it.
"""
print(duzz_healthy.output['drug_doz'])
drug_doz.view(sim=duzz_healthy)
```

```
0.008385009310986964
```



که همانطور که خروجی نشان می‌دهد باید میزان دوز دارو را در این مدل بالاتر ببریم .

سوال سوم :

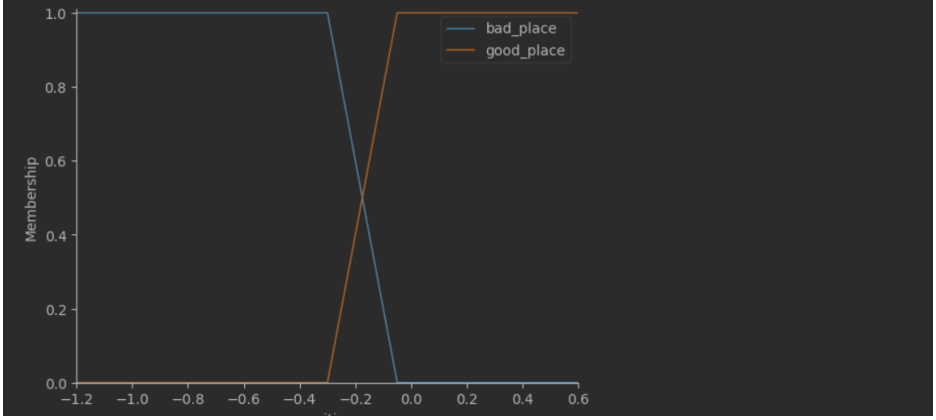
دو ترم ورودی به اسم position , speed برای ماشین دارم که متناسب با آن باید تصمیم بگیرم که نیروی به عقب یا جلوی ماشین وارد کنم . متناسب با بازه شروع و پایان گفته شده ترم‌های ورودی و خروجی خود را معرفی می‌نمایم.

```
position = ctrl.Antecedent(np.linspace(-1.2, 0.6, 10000), 'position') # 10000 points between -1.2 to 0.6 inclusive
speed = ctrl.Antecedent(np.linspace(-0.07, 0.07, 10000), 'speed') # 10000 points between -0.07 to +0.07 inclusive
force = ctrl.Consequent(np.linspace(-1, 1, 10000), 'force') # 10000 points between -1 to 1 inclusive
```

سپس ترم اول ورودی یعنی position را به شکل زیر تعریف می‌کنم که از دو بازه تشکیل شده به اسم good place و bad place ، هدف ما این هست که تا جایی ماشین می‌تواند به سمت good place جلو برود .

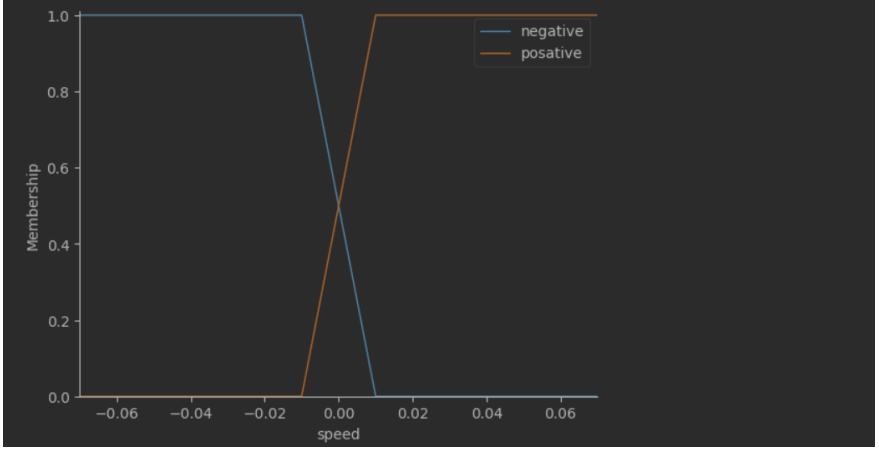
```
position['bad_place'] = fuzz.trapmf(position.universe, [-1.2, -1.2, -0.3, -0.05]) # trapezoidal membership
position['good_place'] = fuzz.trapmf(position.universe, [-0.3, -0.05, 0.6, 0.6]) # trapezoidal membership

position.view()
```

```
c:\Users\erfan\AppData\Local\Programs\Python\Python311\Lib\site-packages\skfuzzy\control\fuzzyvariable.py:122: U
module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```



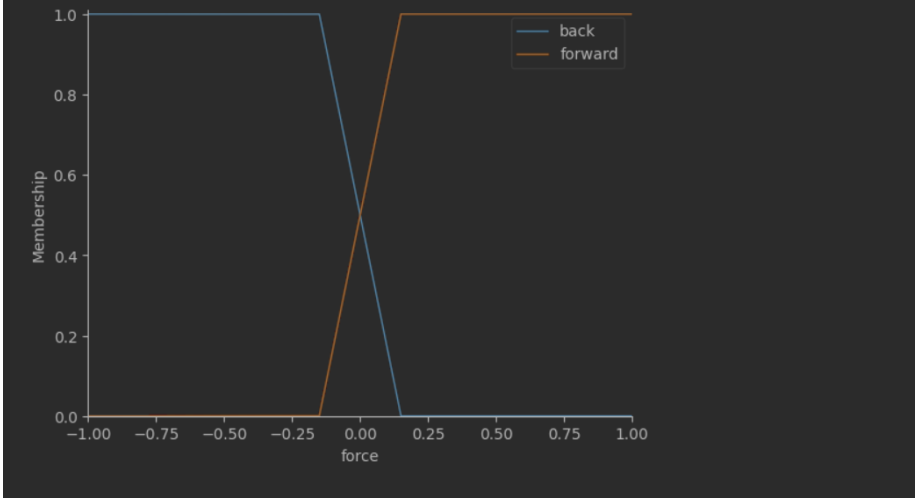
در گام بعد از آنجا که سرعت ماشین می‌تواند مثبت باشد یا منفی دو بازه مثبت بودن و منفی بود را برای آن تعریف می‌کنم .

```
speed['negative'] = fuzz.trapmf(speed.universe, [-0.07, -0.07, -0.01, 0.01]) # trapezoidal membership
speed['posative'] = fuzz.trapmf(speed.universe, [-0.01, 0.01, 0.07, 0.07]) # trapezoidal membership

speed.view()
```



در آخر نیز میزان نیرو به جلو و عقب را با بازه خودمان مشخص می کنیم .

```
force['back'] = fuzz.trapmf(force.universe, [-1, -1, -0.15, 0.15]) # trapezoidal membership
force['forward'] = fuzz.trapmf(force.universe, [-0.15, 0.15, 1, 1]) # trapezoidal membership

force.view()
```



سپس قوانین را به این شکل می‌نویسیم که اگر در مکان درست نبودیم و سرعت‌مان نیز منفی بود، تاجایی که امکان دارد در مکان بد خود پیش می‌رویم تا سرعتمان دوباره مثبت شود در واقع می‌خواهیم در هنگامی که سرعت‌مان مثبت شد در بالاترین مکان بد باشیم تا به کمک جاذبه بتوانیم به سمت جلو و مکان درست بهتر پیش رویم .

پس همین که احساس کردیم سرعتمان در حال مثبت شدن هست به سمت جلو حرکت می کنیم و این کار را یعنی به سمت جلو حرکت کردن را تا زمانی که سرعتمان مثبت است انجام می‌دهیم .

به محض آنکه سرعتمان منفی شد دوباره به عقب برمی گردیم .

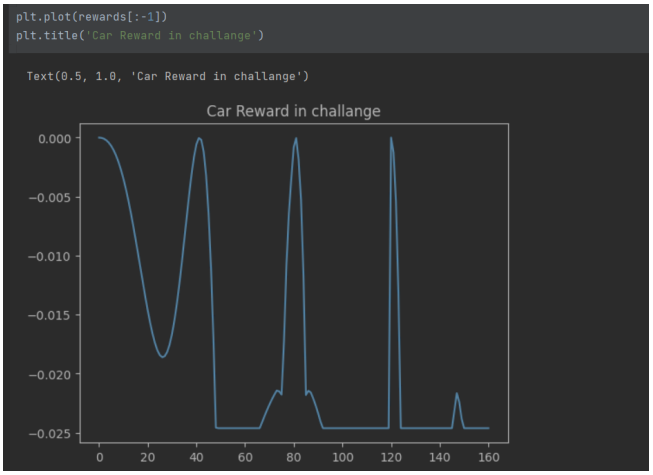
```
list_rules = []
list_rules.append(ctrl.Rule(position['bad_place'] & speed['negative'], force['back']))
list_rules.append(ctrl.Rule(position['bad_place'] & speed['posative'], force['forward']))
list_rules.append(ctrl.Rule(position['good_place'] & speed['negative'], force['back']))
list_rules.append(ctrl.Rule(position['good_place']& speed['posative'], force['forward']))

fuzzy_ctrl = ctrl.ControlSystem(list_rules)
car = ctrl.ControlSystemSimulation(fuzzy_ctrl)
```

به کمک کدهای زیر حداکثر به تعداد 500 بار تلاش برای رد کردن دره می‌کنیم و در هر مرحله نیز میزان پاداش را ذخیره می‌کنیم و براساس سرعت و جایگاهی که قرار گرفتیم براساس قوانینی که داریم میزان نیرو به سمت جلو یا عقب را مشخص می‌کنیم و در نهایت اگر از مکان موردنظر رد شدیم ، نتیجه که تعداد step ما برای رسیدن به این نقطه هست را مشخص می‌کنیم .

```
rewards = []
frames = []
env = gym.make("MountainCarContinuous-v0")
env.seed(101)
np.random.seed(101)

last_position,last_speed = env.reset()
frames.append(env.render('rgb_array'))
step = 0
while step < 500:

    step += 1
    car.input['position'] = last_position
    car.input['speed'] = last_speed
    car.compute()
    force_amount = np.array([car.output["force"]])


    last_position_and_last_speed, reward, ending, _ = env.step(force_amount)
    last_position = last_position_and_last_speed[0]
    last_speed = last_position_and_last_speed[1]
    frames.append(env.render('rgb_array'))
    rewards.append(reward)

    if ending:
        break
env.close()

print(f'my step: {step}')
print(f'prove that my car reached = {last_position}')
```

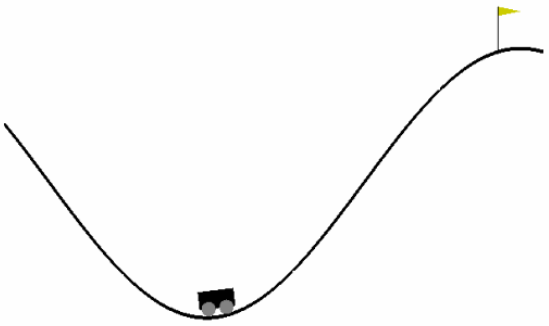
```
my step: 162
prove that my car reached = 0.48718892121010887
```

در نهایت براساس frame های ضبط شده یک گیف بر اساس حرکت ماشین فازی خود را می‌سازیم و خوب نمودار reward را هم در پایین مشاهده می‌نمایید .



همانطور که که می‌بینیم متناسب با حرکت ماشین به سمت قسمت‌های خوب زیاد و کم می‌شود، نمودار تا مرحله ۱۶۱ را نشان می‌دهد و در مرحله آخر یعنی ۱۶۲ باید با یک خط صاف مثل مرحله ۱۲۰ مواجه باشیم که نشان می‌دهد به بالاترین حد خود رسیده و پاداش به بیشترین مقدار خود می‌رسد و جایی که پاداش کمینه هست نشان دهنده این هست که ما در بدترین نقطه از شروع قرار داریم .

هرچند که ما سعی می کردیم در آن نقطه بیشتر قرار بگیریم تا با نیرو و سرعت بیشتری به سمت بالا حرکت کنیم و از نقطه مورد نظر عبور کنیم .



فایل گیف در فایل مربوطه آورده شده است .

سوال چهارم :

الف.در ابتدا این الگوریتم را مورد مطالعه قرار دهید و به طور مختصر شرح دهید. برای این منظور می توانید از این لینک کمک بگیرید.(20 امتیاز)

ب.در این قسمت قصد داریم تا با نحوهی کارکرد این الگوریتم آشنا شویم، برای این منظور لزومی به نوشتن از پایه این الگوریتم نداشته و از کتابخانه sk fuzzy استفاده میکنیم. (30 امتیاز)

1.در ابتدا فایل credit card.csv که در کنار این فایل قابل مشاهده است را با کتابخانه pandas بخوانید

```
import pandas as pd

data_fram = pd.read_csv('credit_card.csv',header=1)

data_fram.head()
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	default payment next month
0	1	20000	2	2	1	24	2	2	-1	-1	...	0	0	0	0	689	0	0	0	0	1
1	2	120000	2	2	2	26	-1	2	0	0	...	3272	3455	3261	0	1000	1000	1000	0	2000	1
2	3	90000	2	2	2	34	0	0	0	0	...	14331	14948	15549	1518	1500	1000	1000	1000	5000	0
3	4	50000	2	2	1	37	0	0	0	0	...	28314	28959	29547	2000	2019	1200	1100	1069	1000	0
4	5	50000	1	2	1	57	-1	0	-1	0	...	20940	19146	19131	2000	36681	10000	9000	689	679	0

5 rows × 25 columns

به کمک لایبری pandas و با قرار دادن سطر اول به عنوان ستون csv موجود را به شکل بالا می‌خوانیم .

```
data_fram.shape
```

```
(30000, 25)
```

```
list(data_fram.columns)
```

```
['ID',
 'LIMIT_BAL',
 'SEX',
 'EDUCATION',
 'MARRIAGE',
 'AGE',
 'PAY_0',
 'PAY_2',
 'PAY_3',
 'PAY_4',
 'PAY_5',
 'PAY_6',
 'BILL_AMT1',
 'BILL_AMT2',
 'BILL_AMT3',
 'BILL_AMT4',
 'BILL_AMT5',
 'BILL_AMT6',
 'PAY_AMT1',
 'PAY_AMT2',
 'PAY_AMT3',
 'PAY_AMT4',
 'PAY_AMT5',
 'PAY_AMT6',
 'default payment next month']
```

شکل ورودی آن برابر با۳۰۰۰۰ردیف و ۲۵ ستون است و نام ستون‌ها همانند بالا است .

لیست تمام آیتم‌های ستون که در آن‌ها BILL هست را در می‌آوریم .

```
list_of_columns = list(data_fram.columns)

list_of_col_have_BILL = [item for item in list_of_columns if "BILL" in item]
print(list_of_col_have_BILL)
```

```
['BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6']
```

```
data_fram['BILL_TOTAL'] = data_fram[list_of_col_have_BILL].sum(axis=1)

data_fram.head()
```

سپس ستونی به اسم BILL_TOTAL بدست می‌آوریم که حاصل جمع ستون‌هایی که BILL هست را بدست می‌آوریم .

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	default payment next month	BILL_TOTAL
0	1	20000	2	2	1	24	2	2	-1	-1	...	0	0	0	689	0	0	0	0	1	7704
1	2	120000	2	2	2	26	-1	2	0	0	...	3455	3261	0	1000	1000	1000	0	2000	1	17077
2	3	90000	2	2	2	34	0	0	0	0	...	14948	15549	1518	1500	1000	1000	1000	5000	0	101653
3	4	50000	2	2	1	37	0	0	0	0	...	28959	29547	2000	2019	1200	1100	1069	1000	0	231334
4	5	50000	1	2	1	57	-1	0	-1	0	...	19146	19131	2000	36681	10000	9000	689	679	0	109339

5 rows × 26 columns

BILL_TOTAL
7704
17077
101653
231334
109339

2.ویژگی های BILL_AMT1تا BILL_AMT6را برای هر داده با هم جمع کنید و ویژگی جدیدی به نام BILL_TOTALدر دیتاست اضافه کنید، در ادامه تنها دو ویژگی LIMIT_BALو BILL_TOTALرا مورد استفاده قرار خواهیم داد.

برای انجام کار انتهایی بالا :

```
new_data_fram = data_fram[["LIMIT_BAL","BILL_TOTAL"]]
new_data_fram.head()
```

	LIMIT_BAL	BILL_TOTAL
0	20000	7704
1	120000	17077
2	90000	101653
3	50000	231334
4	50000	109339

۳.دیتاست جدیدی تنها با دو ویژگی داده شده بسازید و برای این دو ویژگی ازعملیات نرمال سازی یا استانداردسازی استفاده کنید.

سپس ردیف بدست آمده را به کمک MinMaxScaler نرمال می کنیم.

```
from sklearn.preprocessing import MinMaxScaler
norm = MinMaxScaler().fit(new_data_fram)
new_data_fram_norm = norm.transform(new_data_fram)

new_data_fram_norm
```

```
array([[0.01010101, 0.06142041],
       [0.11111111, 0.06309411],
       [0.08080808, 0.07819659],
       ...,
       [0.02020202, 0.07263298],
       [0.07070707, 0.10765263],
       [0.04040404, 0.10127118]])
```

4.از الگوریتم FCM برای خوشه بندی دیتاست جدید استفاده کنید وcرا برابر 2تا10 قرار دهید و برای هر یک از مقادیرc ابتدا داده ها را بر حسب خوشه ای که بیشترین تعلق به آن خوشه دارد را به آن خوشه نسبت داده و روی نمودار با رنگ های مختلف نشان دهید.

برای انجام این کار ابتدا یک نمودار ۳ در ۳ که در هر کدام یک نمودار قرار می‌دهیم را می‌سازیم، همچنین ۱۰ رنگ برای بیشترین تعداد کلاس که ۱۰ مورد است را می‌سازیم .

```
import matplotlib.pyplot as plt
import skfuzzy as fuzz
import numpy as np
colors = ['b', 'orange', 'g', 'r', 'c', 'm', 'y', 'k', 'Brown', 'ForestGreen']

# Set up the loop and plot
fig1, axes1 = plt.subplots(3, 3, figsize=(8, 8))
fpcs = []
```

سپس تعداد مراکز را از ۲ تا ۱۰ در نظر می گیریم و برای هر یک مدل فازی بدست می‌آوریم . که دیتای نرمال شده را به آن می‌دهیم و تعداد مراکز را مشخص می کنیم و درکنار آن تعداد etaration و میزان error را مشخص می کنیم .

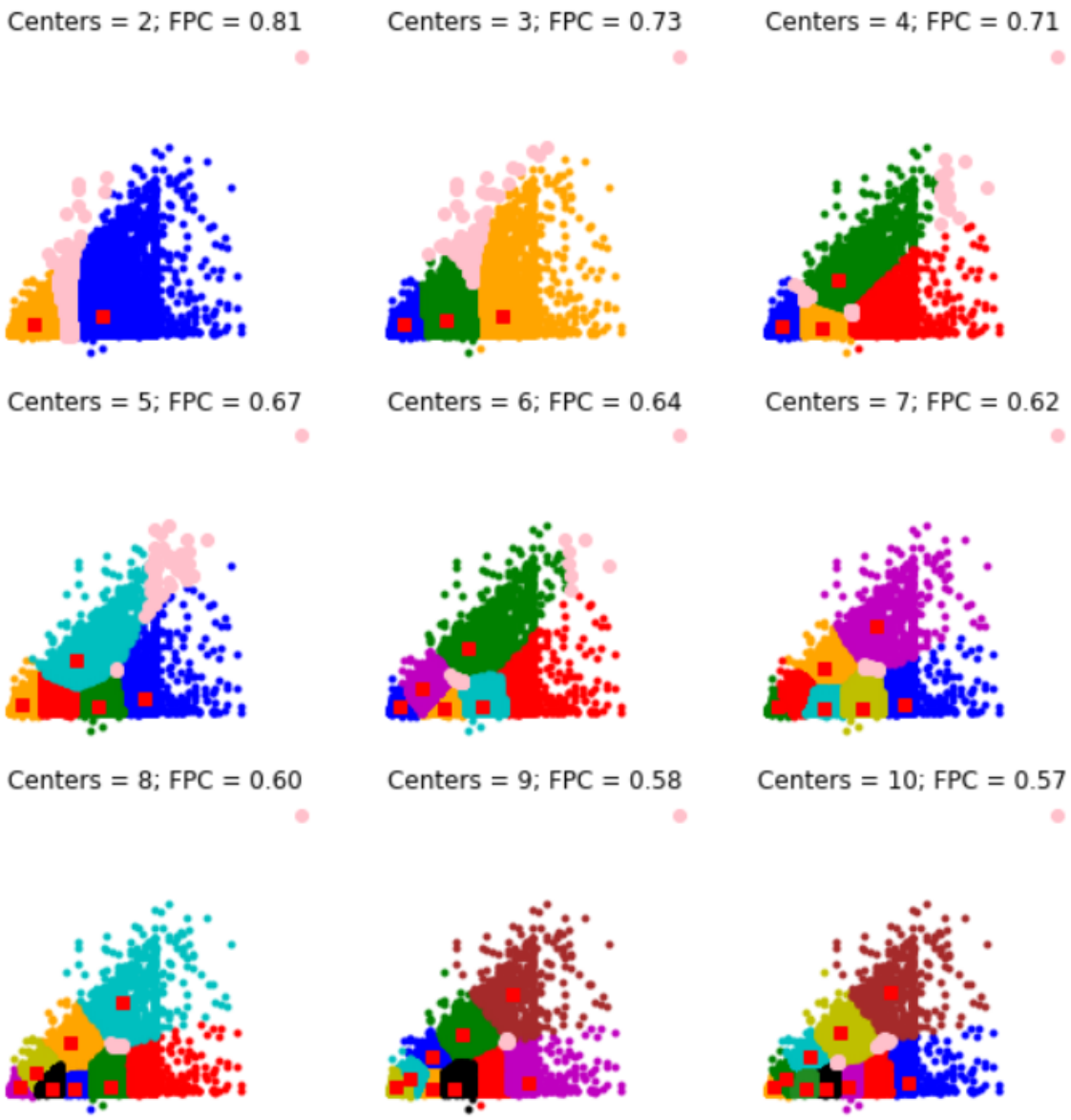
حاصل به ما مختصات مراکز ، میزان اینکه هر آیتم ما چقدر از لحاظ فازی به یک کلاس نزدیک هستند (u) , فاصله اقلیدسی هر آیتم از هر کدام از مراکز (d) که , میزان fpc که در آینده از آن صحبت خواهیم کرد را به ما می‌دهد .

```
for ncenters, ax in enumerate(axes1.reshape(-1), 2):
    cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(
       new_data_fram_norm.T , ncenters, 2, error=0.005, maxiter=10000, init=None)

    # Store fpc values for later
    fpcs.append(fpc)

    tershold = (1 / ncenters) + 0.1
    cluster_membership = np.argmax(u, axis=0)
    for j in range(ncenters):
        ax.plot(new_data_fram_norm[cluster_membership == j,0],
                new_data_fram_norm[cluster_membership == j,1], '.', color=colors[j])

    list_of_border_patterns = []
    index = 0
    for item in u.T:
        if max(item) < tershold:
            ax.plot(new_data_fram_norm[index][0],new_data_fram_norm[index][1],'o',color='pink')

        index += 1

    # Mark the center of each fuzzy cluster
    for pt in cntr:
        ax.plot(pt[0], pt[1], 'rs')

    ax.set_title('Centers = {0}; FPC = {1:.2f}'.format(ncenters, fpc))
    ax.axis('off')

fig1.tight_layout()
```

میزان fpc هر کدام از تعداد مراکز را نگه می‌داریم ، سپس یک میزان threshold در نظر می گیریم .
این میزان threshold میزان اطمینان از درستی کلاس مورد نظر یک مجموعه هست که برابر هست با حاصل جمع احتمال قرار گرفتن در هر کدام از کلاس و tolerance 0.1 یعنی اگر ما بخواهیم به دو کلاس صفحه را تقسیم کنیم، میزان آن برابر خواهد بود با ½ + 0.1 که برابر با 60 درصد خواهد شد .اگر ۳ کلاس باشد نیز با همین فرمول برابر با 40 درصد خواهد شد .
به زبان ساده اگر میزان درستی ماکس متعلق به یکی از کلاس‌ها از میزان انتخاب شانسی + میزان tolerance بیشتر باشد می‌توانیم با اطمینان بگوییم آن نمونه عضو آن کلاس هست وگزنه با اطمینان نمی‌توانیم درباره آن صحبت کنیم .

خوب در ادامه ماکزیمم میزان فازی بودن به ازای هر کلاس را برای هر pattern بدست می‌آوریم و کلاس مربوطه که می‌شود به آن pattern اعلام کرد را مشخص می کنیم و به ازای هر کلاس یک رنگ انتصاب می‌دهیم .

سپس به ازای نقاطی که با اطمینان نمی‌توانیم بگوییم به کدام کلاس متعلق هست براساس threshold آنان را صورتی می کنیم . مراکز هر کلاس را روی نمودار مشخص می کنیم و در بالای هر نمودار تعداد کلاس و fpc هر کدام از این تعداد کلاس را مشخص می کنیم .
براساس موارد بالا خواهیم داشت :



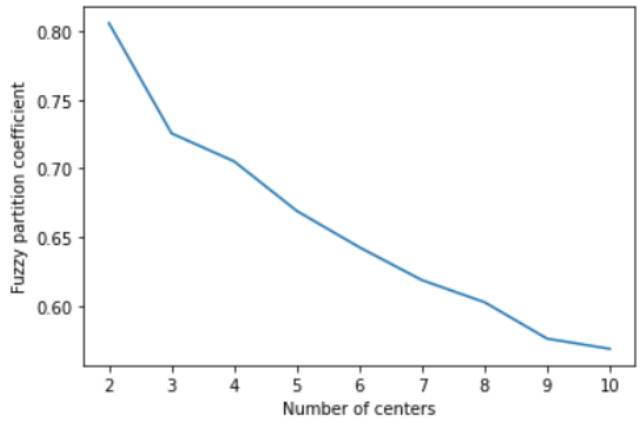
5.برخی از داده ها تعلقشان به یک خوشه زیاد نمی باشد و نمی توان به طور قطعی به آن خوشه نسبت داد. داده هایی که تعلقشان به یک خوشه زیاد(این معیار برای بررسی زیاد بودن و نبودن را توضیح دهید که بر چه اساس تعیین کردید) نیست را به عنوان داده های مشکوک در نظر گرفته و با رنگ های مختلف روی نمودار نشان دهید. (برای هر یک از مقادیرcیک نمودار مجزا می بایست رسم شود)

خوب مقادیر صورتی رنگ در نمودار بالا که در قسمت قبل توضیح دادیم برای این قسمت از سوال مشخص شده و معیار تعیین کردن آنان نیز به طور کامل توضیح داده شده است .

6.با استفاده از معیار FPC بهترین تعداد خوشه را مشخص کنید (ابتدا یک توضیح کوتاهی درباره این معیار دهید) و نحوه ی این انتخاب را توضیح دهید و برای بهترین مقدارcخوشه‌ی مربوط به هر داده را مشخص کنید و در ستونی در دیتاست اضافه کنید. (داده هایی را که می توان به چند خوشه نسبت داد نیز چند خوشه به عنوان خوشه مربوطه در نظر بگیرید و تمام نوع خوشه‌های نسبت داده شده را نمایش دهید. (برای مثال فرض کنید بهترین c برابر3 بوده است، برخی از داده ها به خوشه‌ی1، برخی به خوشه 2 و برخی به خوشه 3 نسبت داده شده اند، برخی نیز مابین خوشه 1و2 و برخی مابین خوشه 1و3 و برخی مابین خوشه 2و3 و برخی بین هر سه خوشه(برحسب درجه عضویتشان برای هر خوشه) و در این حالت 7 نوع خوشه به عنوان خروجی باید وجود داشته باشد))

```
fig2, ax2 = plt.subplots()
ax2.plot(np.r_[2:11], fpcs)
ax2.set_xlabel("Number of centers")
ax2.set_ylabel("Fuzzy partition coefficient")
```

```
Text(0, 0.5, 'Fuzzy partition coefficient')
```



براساس میزان fpcs بدست آمده از قسمت قبل می‌بینیم که بهترین آن به ازای تعداد کلاس ۲ و به میزان ۸۰ درصد است . البته به صورت چشمی هم از نمودار قبل قابل فهم بود این قضیه و هرچقدر تعداد کلاس‌ها بیشتر می‌شود میزان fpcs کاهش پیدا می کند .

FPC در محدوده 0 تا 1 تعریف شده است که 1 بهترین است. این معیاری است که به ما می گوید داده های ما با یک مدل خاص چقدر تمیز توصیف می شوند. در مرحله بعد مجموعه داده های خود را - که می دانیم دارای سه خوشه است - چندین بار با بین 2 تا 9 خوشه خوشه بندی می کنیم. سپس نتایج خوشه‌بندی را نشان می‌دهیم و ضریب تقسیم فازی را رسم می کنیم. هنگامی که FPC به حداکثر می رسد، داده های ما به بهترین شکل توصیف می شوند.

پس با این تفاسیر بهترین تعداد کلاس برای این نوع داده برابر با ۲ کلاس هست .

```
cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(
       new_data_fram_norm.T , 2, 2, error=0.005, maxiter=10000, init=None)

tershold = (1 / 2) + 0.1

list_of_class = []

for item in u.T:
        if max(item) > tershold:
          list_of_class.append([np.argmax(item)])
        else:
            # print("hi")
            index = 0
            list_of_class.append([0,1])
```

سپس دوباره مدل را برای یادگیری دو کلاس استفاده می‌کنیم و از میزان threshold برای انتصاب دادن هر آیتم به یک کلاس استفاده می‌کنیم به این صورت که اگر میزان بزرگترین مقدار فازی یک آیتم بیشتر از threshold بود به آن کلاس انتصابش می‌دهیم در غیر این صورت این کلاس به صورت مرزی هم برای 0 و هم برای کلاس ۱ قابل تعمیم هست پس .

```
data_fram['class_assigend'] = list_of_class
```

```
data_fram[['default payment next month','class_assigend']].head(300)
```

	default payment next month	class_assigend
0	1	[1]
1	1	[1]
2	0	[1]
3	0	[1]
4	0	[1]
...	...	...
295	0	[0, 1]
296	0	[1]
297	0	[0, 1]
298	0	[0]
299	1	[0]

300 rows × 2 columns

یک ستون به این شکل به dataframe خود اضافه می کنیم .

7.آیا رابطه ای بین خوشه های نسبت داده شده وdefault payment next monthوجود دارد؟ مورد بررسی قرار دهید و نتایج آن را تحلیل کنید.

خوب برای اینکه ببینیم چقدر این دو معیار شبیه یکدگیر هستند می‌توانیم ببینم در چند موارد در جایی که default payment next month صفر گفته کلاس فازی ما صفر و در جایی که یک گفته کلاس فازی ما نیز کلاس صفر است .

```
for_expriment = data_fram[['default payment next month','class_assigend']]
arr = for_expriment.to_numpy()
arr
```

```
array([[1, list([1])],
       [1, list([1])],
       [0, list([1])],
       ...,
       [1, list([1])],
       [1, list([1])],
       [1, list([1])]], dtype=object)
```

```
number_of_simmilar_item = 0
for item in arr :
  if len(item[1]) == 1 and item[1][0] == item[0]:
      number_of_simmilar_item +=1

print(number_of_simmilar_item)
print(number_of_simmilar_item/len(arr))
```

```
12244
0.40813333333333335
```

در چهل درصد موارد همانطور که می‌بینید با همدیگر شبیه هستند .

اگر کلاس‌های فازی را نیز در نقاط مرزی درست در نظر بگیریم خواهیم داشت :

```
number_of_simmilar_item = 0
for item in arr :
  if len(item[1]) == 1 and item[1][0] == item[0]:
      number_of_simmilar_item +=1
  elif len(item[1]) == 2:
      number_of_simmilar_item +=1

print(number_of_simmilar_item)
print(number_of_simmilar_item/len(arr))
```

```
15135
0.5045
```

با در نظر گرفتن نقاط مرزی 50 درصد موارد شبیه هم هستند .

این معیار تعریف شده با این میزان شباهت نمی‌تواند یک معیار خوب برای تشخیص ویژگی default payment next month باشد .