```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

DEFINICION DE SUGENO

Se requiere 2 conjuntos difusos

Para este ejemplo seran:

- Servicio
- Comida

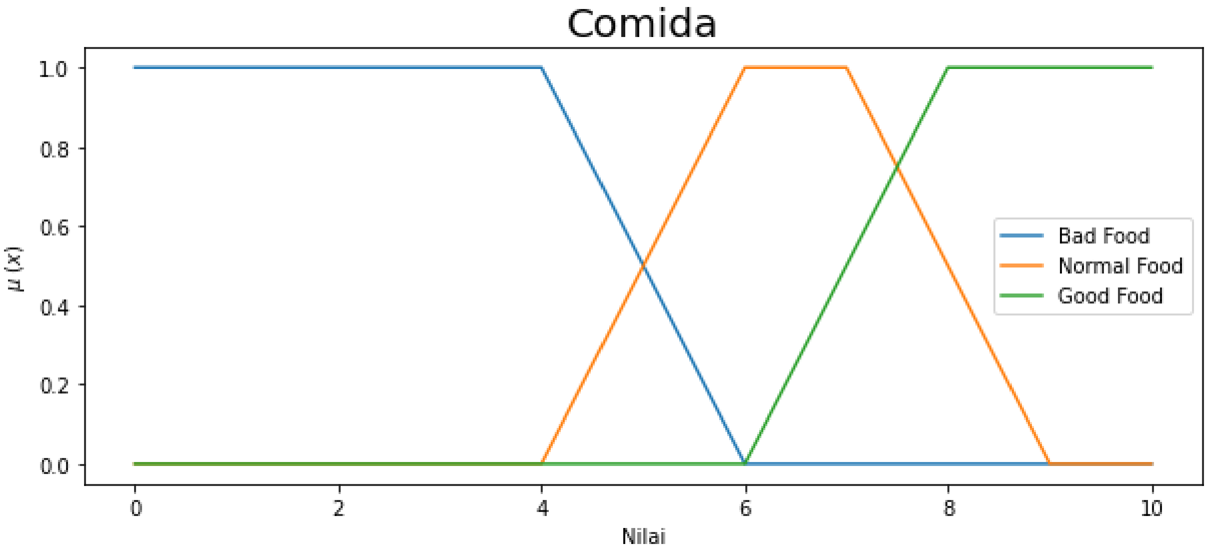
```
data = pd.read_excel("https://github.com/tigerofmurder/FuzzySegeno/blob/main/test.
data
```

	id	servicio	comida
0	1	58	7
1	2	54	1
2	3	98	2
3	4	52	4
4	5	11	4
...	...	...	...
95	96	30	1
96	97	25	3
97	98	27	10
98	99	8	6
99	100	11	8

100 rows × 3 columns

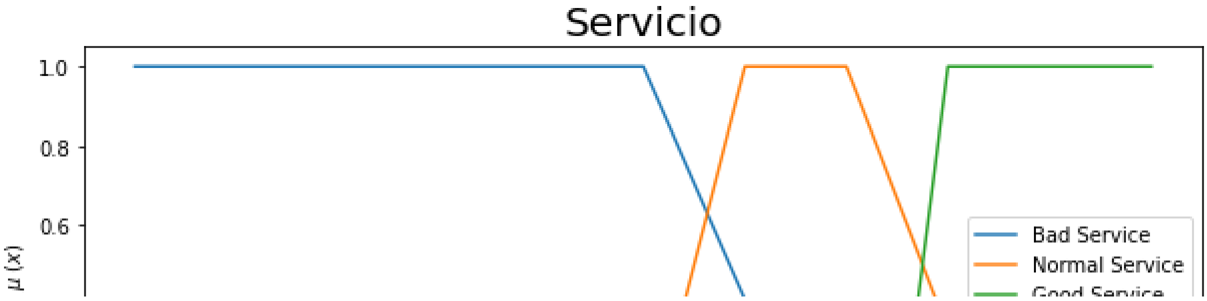
```
def BadFood(f):
    BadFood, notBadFood = 4, 6
    if f <= BadFood:
        return 1
    elif f > notBadFood:
        return 0
    elif f > BadFood and f <= notBadFood:
        return ((notBadFood - f) / (notBadFood - BadFood))

def NormalFood(f):
    notNormalFood1, NormalFood1, NormalFood2, notNormalFood2 = 4, 6, 7, 9
    if f > NormalFood1 and f <= NormalFood2:
        return 1
    elif f <= notNormalFood1 or f > notNormalFood2:
        return 0
    elif f > notNormalFood1 and f <= NormalFood1:
        return ((f - notNormalFood1) / (NormalFood1 - notNormalFood1))
    elif f > NormalFood2 and f <= notNormalFood2:
        return ((notNormalFood2 - f) / (notNormalFood2 - NormalFood2))

def GoodFood(f):
    notGoodFood, GoodFood = 6, 8
    if f > GoodFood:
        return 1
    elif f <= notGoodFood:
        return 0
    elif f > notGoodFood and f <= GoodFood:
        return ((f - notGoodFood) / (GoodFood - notGoodFood))

x = [i for i in range(11)]

ybadF = [BadFood(i) for i in x]
ynormalF = [NormalFood(i) for i in x]
ygoodF = [GoodFood(i) for i in x]

plt.figure(figsize=(10,4))
plt.title('Comida',fontsize = 20)
plt.plot(x, ybadF, label = 'Bad Food')
plt.plot(x, ynormalF, label = 'Normal Food')
plt.plot(x, ygoodF, label = 'Good Food')
plt.xlabel('Nilai')
plt.ylabel(r'$\mu\ (x)$')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fa55109eb10>
```



```
def BadService(x):
  badService = 50
  notbadService = 67
  if x <= badService:
    return 1;
  elif x > notbadService:
    return 0
  elif x > badService and x <= notbadService:
    return (notbadService - x) / (notbadService - badService)

def NormalService(x):
  notnormalService1 = 50
  normalService1 = 60
  normalService2 = 70
  notnormalService2 = 85

  if x > normalService1 and x <= normalService2:
    return 1
  elif x <= notnormalService1 or x > notnormalService2:
    return 0
  elif x > notnormalService1 and x <= normalService1:
    return (x - notnormalService1) / (normalService1 - notnormalService1)
  elif x > normalService2 and x <= notnormalService2:
    return (notnormalService2 - x) / (notnormalService2 - normalService2)

def GoodService(x):
  notgoodService = 75
  goodService = 80

  if x > goodService:
    return 1
  elif x <= notgoodService:
    return 0
  elif x > notgoodService and x <= goodService:
    return (x - notgoodService) / (goodService - notgoodService)

x = [i for i in range(101)]
ybadS = [BadService(i) for i in x]
ynormalS = [NormalService(i) for i in x]
ygoodS = [GoodService(i) for i in x]

plt.figure(figsize=(10,4))
plt.title('Servicio',fontsize = 20)
plt.plot(x, ybadS, label = 'Bad Service')
plt.plot(x, ynormalS, label = 'Normal Service')
plt.plot(x, ygoodS, label = 'Good Service')
plt.xlabel('Nilai')
plt.ylabel(r'$\mu\ (x)$')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fa550fdab10>
```



METODOS

- Fuzzificacion
- Inferencia
- Defuzzificacion

```
def fuzzificationFood(foodValue):
  foodSet = []
  foodSet.append(BadFood(foodValue))
  foodSet.append(NormalFood(foodValue))
  foodSet.append(GoodFood(foodValue))
  return foodSet

def fuzzificationService(serviceValue):
  serviceSet = []
  serviceSet.append(BadService(serviceValue))
  serviceSet.append(NormalService(serviceValue))
  serviceSet.append(GoodService(serviceValue))
  return serviceSet
```

```
def inference(serviceSet, foodSet):
  inferenceSet = []
  recommendedSet, moderatelySet, notRecommendedSet = [], [], []

  recommendedSet.append(min(foodSet[2], serviceSet[2]))
  recommendedSet.append(min(foodSet[2], serviceSet[1]))
  recommendedSet.append(min(foodSet[1], serviceSet[2]))


  moderatelySet.append(min(foodSet[1], serviceSet[1]))
  moderatelySet.append(min(foodSet[2], serviceSet[0]))

  notRecommendedSet.append(min(foodSet[0], serviceSet[0]))
  notRecommendedSet.append(min(foodSet[1], serviceSet[0]))
  notRecommendedSet.append(min(foodSet[0], serviceSet[1]))
  notRecommendedSet.append(min(foodSet[0], serviceSet[2]))

  inferenceSet.append(max(recommendedSet))
  inferenceSet.append(max(moderatelySet))
  inferenceSet.append(max(notRecommendedSet))
  return inferenceSet

def defuzzification(inferenceSet):
  multiplier = (inferenceSet[0]*100) + (inferenceSet[1]*80) + (inferenceSet[2]*50)
  divider = inferenceSet[0] + inferenceSet[1] + inferenceSet[2]
  return multiplier/divider
```

```
Result = []
for row in range(100):
  fuzziService = fuzzificationService(data['servicio'][row])
  fuzziFood = fuzzificationFood(data['comida'][row])
  inferensi = inference(fuzziService,fuzziFood)
  Result.extend([defuzzification(inferensi)])

data['fuzzy'] = Result
data
```

1 to 25 of 100 entries Filter

index	id	servicio	comida	fuzzy
0	1	58	7	76.7845659163987
1	2	54	1	50.0
2	3	98	2	50.0
3	4	52	4	50.0
4	5	11	4	50.0
5	6	59	10	93.13304721030043
6	7	61	8	85.07936507936508
7	8	30	10	80.0
8	9	45	1	50.0
9	10	36	9	80.0
10	11	10	5	50.0
11	12	38	7	60.0
12	13	80	3	50.0
13	14	31	8	70.0
14	15	78	5	76.5909090909091
15	16	82	6	96.66666666666667
16	17	70	3	50.0
17	18	3	9	80.0
18	19	42	3	50.0
19	20	49	10	80.0
20	21	48	2	50.0
21	22	79	9	100.0
22	23	18	4	50.0
23	24	100	9	100.0
24	25	61	10	94.78260869565217

Show 25 per page 1 2 3 4

Like what you see? Visit the data table notebook to learn more about interactive tables.