ARTIFICIAL INTELLIGENCE PROGRAMMING ASSIGNMENT 02 REPORT

Report

Created to fulfill the assignment of the Artificial Intelligence course

By:

Muhammad Farhan Akbar	1301192246
Andi Achmad Adjie	1301194264
Muhammad Furqon Fahlevi	1301194214

INFORMATICS MAJOR

SCHOOL OF COMPUTING

TELKOM UNIVERSITY

BANDUNG

2021

1. INTRODUCTION

Fuzzy Logic is an approach to variable processing that allows for multiple values to be processed through the same variable. Fuzzy logic attempts to solve problems with an open, imprecise spectrum of data that makes it possible to obtain an array of accurate conclusions. Fuzzy logic is designed to solve problems by considering all available information and making the best possible decision given the input.

Fuzzy Logic in artificial intelligence is a generalized form of standard logic, where any concept might have a truth degree ranging between 0.0 and 1.0. Fuzzy Logic can be used for vague concepts.

2. OBSERVATION ANALYSIS

2.1. Amount of Linguistic Name of each entry

The inputs for the selections are 'Makanan' and 'Pelayanan', while the output is the acceptance score. We design the linguistic variable for each input into three (3) linguistics such as:

- Makanan: High, Mid, Low
- Pelayanan: High, Mid, Low

For the output, we also design three (3) linguistics:

- Score: Excellent, Average, Bad

2.2. Forms and Limits of Input Membership Functions

- Membership function for 'Makanan' (Tasty):
  [range: 0,10]

$$\begin{cases} 0, & n \le 6 \\ 1, & n > 8.5 \\ \dfrac{n-6}{8.5-6}, & 6 < n \le 8.5 \end{cases}$$

```
[9]  def membership_makanan_high(n):
       if n > 8.5: return 1
       if n <= 6: return 0
       return (n - 6) / (8.5 - 6)
```

- Membership function for 'Makanan' (Average):
  [range: 0,10]

$$\begin{cases} 0, & n \le 2,\ n > 8 \\ 1, & 4.5 < n \le 6.5 \\ \dfrac{n-2}{4.5-2}, & 2 < n \le 5 \\ \dfrac{8.5-n}{8.5-6.5}, & 6.5 < n \le 8.5 \end{cases}$$

```
[ ]  def membership_makanan_mid(n):
       if n > 8 or n <= 2: return 0
       if n > 2 and n <= 4.5: return (n - 2)/ (5 - 2)
       if n > 4.5 and n <= 6.5: return 1
       if n > 6.5 and n <= 8.5: return (8.5 - n) / (8.5 - 6.5)
```

- Membership function for 'Makanan' (Untasty):
  [range: 0,10]

$$\begin{cases} 0, & n > 4.5 \\ 1, & n \le 2 \end{cases}$$

$$\frac{4.5 - n}{4.5 - 2}, \qquad 2 < n \leq 4.5$$

```
[11] def membership_makanan_low(n):
       if n <= 2: return 1
       if n > 4.5: return 0
       return (4.5 - n) / (4.5 - 2)
```

- Membership function for 'Pelayanan' (Good):
  [range: 0,100]

$$\begin{cases} 0, & n \leq 60 \\ 1, & n > 80 \\ \dfrac{n - 60}{80 - 60}, & 60 < n \leq 80 \end{cases}$$

```
[ ]  def membership_pelayanan_high(n):
       if n > 80: return 1
       elif n <= 60: return 0
       else: return (n - 60) / (80-60)
```

- Membership function for 'Pelayanan' (Average):
  [range: 0,100]

$$\begin{cases} 0, & n \leq 20, n > 80 \\ 1, & 40 < n \leq 60 \\ \dfrac{n - 20}{40 - 20}, & 20 < n \leq 40 \\ \dfrac{80 - n}{80 - 60}, & 60 < n \leq 80 \end{cases}$$

```
[ ]  def membership_pelayanan_mid(n):
       if n > 80 or n <= 20: return 0
       elif n > 20 and n <= 40: return (n - 20) / (40 - 20)
       elif n > 40 and n <= 60: return 1
       elif n > 60 and n <= 80: return (80 - n) / (80 - 60)
```

- Membership function for 'Pelayanan' (Bad):
  [range: 0,100]

$$\begin{cases} 0, & n > 40 \\ 1, & n \leq 20 \\ \dfrac{40 - n}{40 - 20}, & 20 < n \leq 40 \end{cases}$$

```
[ ]  def membership_pelayanan_low(n):
       if n <= 20: return 1
       elif n > 40: return 0
       else: return (40 - n) / (40 - 20)
```

2.3. Inference Rules

We create this table for inference rules on fuzzy input to get fuzzy output, the formed inference rule is.

Makanan	Service	Score
Tasty	Good	Excellent
Tasty	Average	Excellent
Tasty	Bad	Average
Average	Good	Excellent
Average	Average	Average
Average	Bad	Average
Untasty	Good	Average
Untasty	Average	Bad
Untasty	Bad	Bad

We priority the score of 'Makanan' and after that the 'Service' score.

2.4. Defuzzification Method

For defuzzification method we use the *Sugeno* method, and we applied the fuzzy output to the equation:

$$z^* = \frac{\sum_{i=1}^{l} \mu B_i \cdot q}{\sum_{i=1}^{l} \mu B_i}$$

2.5. Forms and Limits of Output Membership Functions

Because we use *Sugeno* method, we determine the limits of output membership function from 'z*' formula of *Sugeno*:

- *excellent* = 100
- $average$ = 70
- bad = 50

$$z^* = \frac{(x * 100) + (y * 70) + (z * 50)}{(x + y + z)}$$

```
[ ]  def sugeno(x,y,z):
       return ((x * 100) + (y * 70) + (z * 50)) / (x + y + z)
```

Where:

- x is *Sugeno* result for excellent
- y is *Sugeno* result for average
- z is *Sugeno* result for bad

3. PROCESS BUILT

3.1. Reading File

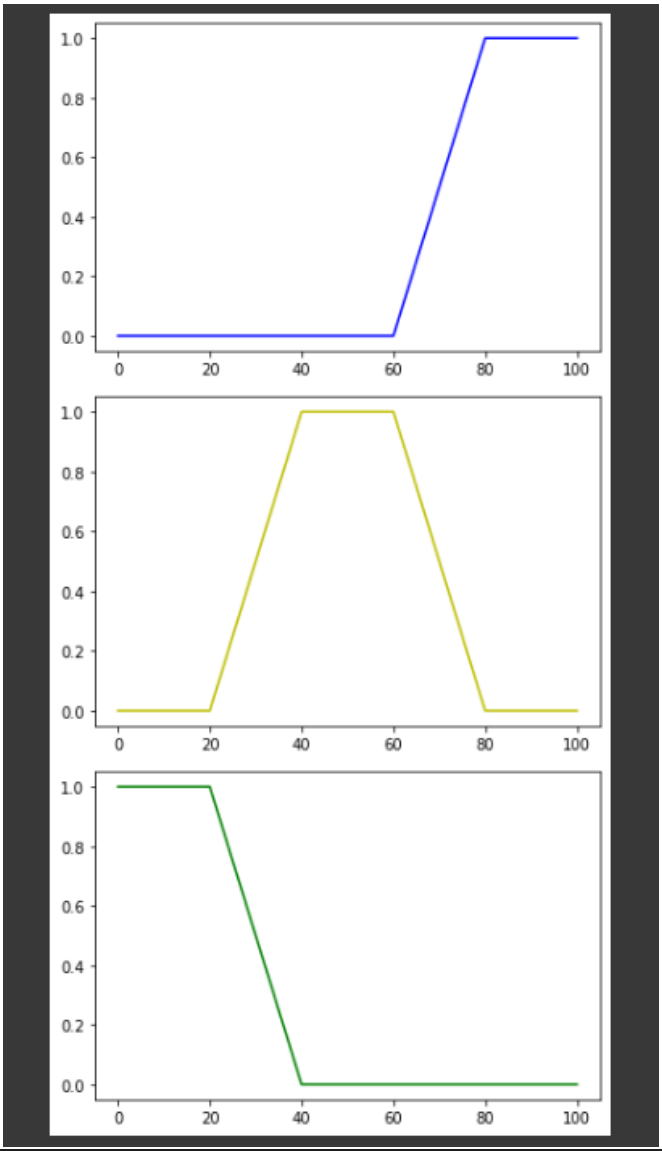
We write this code to reading a ‘xlsx’ file:

```
[ ]  data = pd.read_excel('https://github.com/farhanakbar8/fuzzylogic-AI/blob/main/restoran.xlsx?raw=true')
```

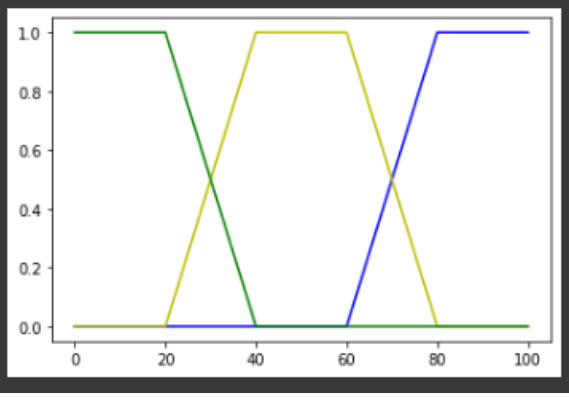
3.2. Fuzzification

Fuzzification is a process to change the input from crisp to fuzzy (linguistic variables) which is usually presented in the form of a fuzzy set with their respective membership functions. The crisp input value for ‘Makanan’ is 0 to 10 while the crisp input value for "Pelayanan" is 0 to 100 and the fuzzy values for ‘Makanan’ and ‘Pelayanan’ are 0,0 to 1,0.

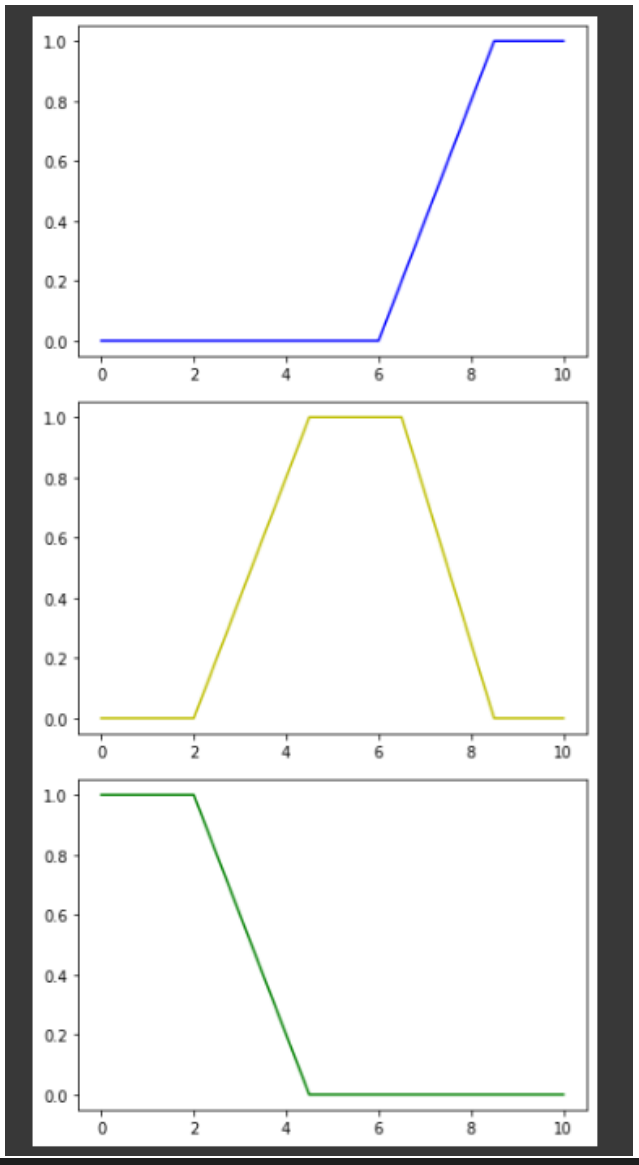
```
[3]  #Membership Function Plot Pelayanan

     # Pelayanan (Good)
     pelayanan_x_1 = [0, 60, 80, 100]
     pelayanan_y_1 = [0, 0, 1, 1]
     plt.plot(pelayanan_x_1, pelayanan_y_1, label='Good', color='b')
     plt.show()

     # Pelayanan (Average)
     pelayanan_x_2 = [0, 20, 40, 60, 80, 100]
     pelayanan_y_2 = [0, 0, 1, 1, 0, 0]
     plt.plot(pelayanan_x_2, pelayanan_y_2, label='Average', color='y')
     plt.show()

     # Pelayanan (Bad)
     pelayanan_x_3 = [0, 20, 40, 100]
     pelayanan_y_3 = [1, 1, 0, 0]
     plt.plot(pelayanan_x_3, pelayanan_y_3, label='Bad', color='g')
     plt.show()
```



[26]
```
# Combined Membership Function for Pelayanan
plt.plot(pelayanan_x_1, pelayanan_y_1, label='Good', color='b')
plt.plot(pelayanan_x_2, pelayanan_y_2, label='Average', color='y')
plt.plot(pelayanan_x_3, pelayanan_y_3, label='Bad', color='g')
plt.show()
```

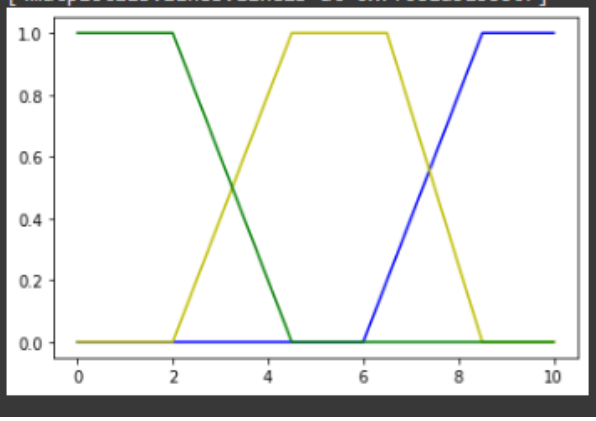


```
    membership pelayanan high(n):
  if n › 80: retd+i 1
  e?:- n ‹= 60! r-tu=r 0
  else: re-cr’ (n - 60) / (8B-60)


  nenbe rs h ip_pe1ay anan_m i d ( n ):
  if  n › 80 c - n<= 20:  et r 1 0
  < - n › 20 < nd n < = 40 : " I - iJ (n - 20) / ( 40 - 20)
  e - n › 46 a nd n < = 60 : "El - iJ 1
  e  - n › 66 a nd n < = 80 : "el  n (Be - n) / ( 80 - 66)


    membership pelayanan lo'w(n)!


    >  r-e r.r  (40 - n) / (40 - 20)


for i in nange(len(data)):
  print( "rsta ke (i} -- › {data.iloc[ij[ ps. .a’an ]}")
  print( "’is’ = {membership pelayanan high(data.Doc[i]['gels.a’en ])}")
  print( "niJ = {membership pelayanan mzd(data.iloc[i]['oela'a;er'])}")
  print( "Io ' = (membership_pelayanan_low(data.iloc[i]['oela',a sr'])}")
  print(" ”n'r')




makakanan_x_1 = [0, 6, 8.5, ]0]
makakanan _1 = [0, 0, ], l]
plt.plot(makakanan_x_1, makakanan  _1, label=  az-y', color='k')
plt.show()


nakanan_X_2 = [6,  2, 4. 5, 6. 5, 8. 5, IB ]
nakanan  _2 = [6,  B, 1, 1, B, B]
plt.plot(makakanan_x_2, makakanan  _2, label= Ave’:ge', color=''.’ )
plt.show()


makakanan x 3 = [0, 2, 4.5, ]M]
makakanan _3 = [1, 1, 0, 0]
plt.plot(makakanan x_3, makakanan  3, label= Jn?e=t,', color='g )
p1t . sha\v( )
```




```
[25] #Combined Membership Function for Makanan

     plt.plot(makanan_x_1, makanan_y_1, label='Tasty', color='b')
     plt.plot(makanan_x_2, makanan_y_2, label='Average', color='y')
     plt.plot(makanan_x_3, makanan_y_3, label='Untasty', color='g')
```

```
[<matplotlib.lines.Line2D at 0x7f681a515550>]
```



```
[9]  def membership_makanan_high(n):
       if n > 8.5: return 1
       if n <= 6: return 0
       return (n - 6) / (8.5 - 6)

[10] def membership_makanan_mid(n):
       if n > 8 or n <= 2: return 0
       if n > 2 and n <= 4.5: return (n - 2)/ (5 - 2)
       if n > 4.5 and n <= 6.5: return 1
       if n > 6.5 and n <= 8.5: return (8.5 - n) / (8.5 - 6.5)

[11] def membership_makanan_low(n):
       if n <= 2: return 1
       if n > 4.5: return 0
       return (4.5 - n) / (4.5 - 2)

[12] for i in range(len(data)):
       print(f"data ke {i} --> {data.iloc[i]['makanan']}")
       print(f"high = {membership_makanan_high(data.iloc[i]['makanan'])}")
       print(f"mid = {membership_makanan_mid(data.iloc[i]['makanan'])}")
       print(f"low = {membership_makanan_low(data.iloc[i]['makanan'])}")
       print('\n\n\n')
```

3.3. Inference

Using Disjunction rule, get the maximum value for each fuzzy output. Apply the inference rules to the fuzzy inputs to obtain fuzzy output.

```
[ ]  result = []
     for i in range(len(fuzzy)):
         max_excellent = 0
         max_average = 0
         max_bad = 0
         for j in range(len(fuzzy[i])):
           if fuzzy[i][j]['category'] == 'excellent':
             max_excellent = max(max_excellent, fuzzy[i][j]['score'])
             #print(fuzzy[i][j]['score'], max_excellent)
           elif fuzzy[i][j]['category'] == 'average':
             max_average = max(max_average, fuzzy[i][j]['score'])
             #print(fuzzy[i][j]['score'], max_average)
           elif fuzzy[i][j]['category'] == 'bad':
             max_bad = max(max_bad, fuzzy[i][j]['score'])
             #print(fuzzy[i][j]['score'], max_bad)
         result.append([max_excellent, max_average, max_bad])
```

3.4. Defuzzification

After we get the fuzzy output, we use the *Sugeno* method for defuzzification with value of 100 for good, 70 for average, and also 50 for bad. We define *Sugeno* and the fuzzy output to the equation:

```
[ ]  def sugeno(x,y,z):
       return ((x * 100) + (y * 70) + (z * 50)) / (x + y + z)

[ ]  sugeno_result = []
     for i in range(len(result)):
       print(f'excellent = {result[i][0]}\naverage = {result[i][1]}\nbad = {result[i][2]}')
       sugeno_result.append(sugeno(result[i][0], result[i][1], result[i][2]), )

     sugeno_result
```

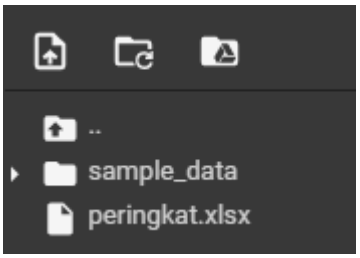
Here is the example result the equation of defuzzification:

```
[16] [80.43478260869566,
      50.0,
      70.0,
      65.38461538461537,
      65.38461538461537,
      100.0,
      92.85714285714285,
      85.0,
      50.0,
      94.0,
      70.0,
      80.43478260869566,
      80.71428571428571,
      86.5,
      97.0,
      100.0,
      70.0,
      70.0,
      57.14285714285714,
      100.0,
      50.0,
      100.0,
      65.38461538461537,
      100.0,
      100.0,
      50.0,
      92.85714285714285,
      57.14285714285714,
      70.0,
      70.0,
      100.0,
      57.14285714285714,
      89.5,
      93.07692307692307,
      50.0,
      70.0,
      64.66666666666667,
      50.0,
      80.43478260869566,
```

4. RESULT

After we get the result of defuzzification we stored the new data named ‘peringkat.xlsx’:

```
[ ]  datanew = data
     datanew['score'] = np.zeros(100)

     for i in range(len(datanew)):
       datanew['score'][i] = sugeno_result[i]

     datanew = datanew.sort_values(['score'], ascending=False)

[ ]  datanew2 = datanew.drop(['pelayanan', 'makanan', 'score'], axis=1)
     datanew2[:10].to_excel('peringkat.xlsx', index=False, header=False)
```



Here is the top ten restaurant ID:

51
16
31
54
25
24
22
69
20
79

5. CONCLUSION

Based on the fuzzy logic system that we made. We only choose the top ten restaurant ID with highest score.

	id	pelayanan	makanan	score
50	51	48	10	100.0
15	16	82	6	100.0
30	31	74	9	100.0
53	54	64	10	100.0
24	25	61	10	100.0
23	24	100	9	100.0
21	22	79	9	100.0
68	69	86	10	100.0
19	20	49	10	100.0
78	79	87	9	100.0

Video Presentation:
Muhammad Furqon Fahlevi:
https://drive.google.com/file/d/1cJ0RfInGa-3hs64UL1na2WdBoVQkTYHJ/view?usp=sharing
Muhammad Farhan Akbar:
https://drive.google.com/file/d/14TZNGaCGZih3OvAgPirOL2aHWRbXF73y/view?usp=sharing
Andi Achmad Adjie:
https://drive.google.com/file/d/11sF27PMUmPqck2TLd9Bv6onV8ZBsArbM/view?usp=sharing