LAPORAN OBSERVASI *FUZZY LOGIC*

Pendahuluan

Fuzzy Logic merupakan cabang ilmu pengetahuan yang membuat komputer mampu menirukan kecerdasan manusia sehingga dapat melakukan hal-hal yang apabila dikerjakan manusia memerlukan kecerdasan.

Deskripsi Kegiatan

Kegiatan yang akan dilakukan dalam observasi ini adalah melakukan analisis, desain, dan mengimplementasikan logika *fuzzy* (*Fuzzy Logic*) ke dalam suatu program komputer untuk menentukan 20 (dua puluh) dari 100 (seratus) mahasiswa yang layak mendapatkan bantuan biaya registrasi sebesar 50%.

Metode Penyelesaian

Bahasa pemrograman yang digunakan adalah Python dengan menggunakan Jupyter dan Notepad++ sebagai *text editor*-nya. Metode penyelesaian yang digunakan untuk melakukan kegiatan terbagi menjadi beberapa tahap, yaitu:

A. Jumlah dan Nama Linguistik Setiap *Input*
   Masukan *(input)* merupakan 2 (dua) data dari *file* "Mahasiswa.xls" yaitu 'Penghasilan' dan ' Pengeluaran' yang bertipe bilangan *real*. Sedangkan, 'Id' hanya data, bukan masukan *(input)*.

B. Bentuk dan Batas Fungsi Keanggotaan
   Bentuk dan batas fungsi keanggotaan menggunakan fungsi Trapesium, Setengah Trapesium Menanjak, Setengah Trapesium Menurun, Sigmoid Naik, dan Sigmoid Turun.

```python
#fungsi keanggotaan penghasilan

#sigmoid turun
def sigmoid_turun_penghasilan_kecil(n):
    if (1<n<=3):
        return 1-(2*((n-1)/(5-1)**2))
    elif (3<n<5):
        return 2*((n-1)/(5-1)**2)
    elif (n<=1):
        return 1
    else:
        return 0

#trapesium
def trapesium_penghasilan_sedang(n):
    if (6<=n<=8):
        return 1
    elif (4<n<6):
        return (n-4)/(6-4)
    elif 8<n<=10:
        return -(n-10)/(10-8)
    else:
        return 0

#setengah trapesium menanjak
def trapesium_penghasilan_besar(n):
    if (11<=n<=13):
        return 1
    elif (8<n<11):
        return (n-4)/(6-4)
    elif 13<n<=15:
        return -(n-10)/(10-8)
    else:
        return 0

#sigmoid naik
def sigmoid_naik_penghasilan_sangat_besar(n):
    if (n>=18):
        return 1
    elif (14<n<=16):
        return 2*(((n-14)/(18-14))*((n-14)/(18-14)))
    elif (16<n<18):
        return 1-(2*((18-n)/(18-14)*(18-n)/(18-14)))
    else:
        return 0
```

```python
#fungsi keanggotaan pengeluaran

#setengah trapesium menurun
def trapesium_pengeluaran_kecil(n):
    if (1<=n<=3):
        return 1
    elif (3<n<=5):
        return -(n-5)/(5-3)
    else:
        return 0

def trapesium_pengeluaran_sedang(n):
    if (3<n<5):
        return (n-3)/(5-3)
    elif (5<=n<=7):
        return 1
    elif (7<n<=9):
        return -(n-9)/(9-7)
    else:
        return 0

def trapesium_pengeluaran_besar(n):
    if (7<n<9):
        return (n-7)/(9-7)
    elif (9<=n<=12):
        return 1
    else:
        return 0
```

C. Aturan Inferensi

```
##aturan inferensi

def aturan_inferensi(penghasilan_sangat_besar,penghasilan_besar, penghasilan
    rules =[ [min(penghasilan_kecil,pengeluaran_kecil), 'besar'], [min(pengh
             [min(penghasilan_sedang,pengeluaran_kecil), 'kecil'], [min(pengh
             [min(penghasilan_besar,pengeluaran_kecil), 'kecil'], [min(pengha
             [min(penghasilan_sangat_besar,pengeluaran_kecil),'kecil'], [min(
             'besar']
             ]

    besar = []
    kecil = []

    for i in range(len(rules)):
        if rules[i][1] == 'besar':
            besar.append(rules[i][0])
        elif rules[i][1] == 'kecil':
            kecil.append(rules[i][0])
    return max(besar) , max(kecil)

def nk_kecil(n):
    if (0<n<=40):
        return 1
    elif (40<n<=60):
        return -(n-60)/(60-40)
    else:
        return 0

def nk_besar(n,a=60,b=80,c=100):
    if (60<n<80):
        return (n-60)/(80-60)
    elif (80<=n<=100):
        return 1
    else:
        return 0
```

D. Metode Defuzzifikasi

Metodo defuzzifikasi yang digunakan yaitu Mean of Max (MoM).

```
#defuzzifikasi
#defuzzifikasi dengan Mean of Max (MoM)
def mom(x, mfx):
    return np.mean(x[mfx == max(mfx)])

def defuzz(besar,kecil):
    dx_kecil = np.array([nk_kecil(n) for n in range(0,60)])
    dx_besar = np.array([nk_besar(n) for n in range(60,101)])
    x_kecil = np.array([i for i in range(0,60)])
    x_besar = np.array([j for j in range(60,101)])
    mom_kecil = mom(x_kecil,dx_kecil)*kecil
    mom_besar = mom(x_besar,dx_besar)*besar
    total = (mom_besar + mom_kecil)/(besar+kecil)
    return total
```

```
data_mhsw = {}
data_kemungkinan = {}

for i in range(len(id_mhsw)):
    id_mahasiswa = id_mhsw[i]
    penghasilan_mahasiswa = float(penghasilan_mhsw[i])
    pengeluaran_mahasiswa = float(pengeluaran_mhsw[i])
    data_mhsw[id_mahasiswa] = [penghasilan_mahasiswa,pengeluaran_mahasiswa]

for index in data_mhsw:
    penghasilan_kecil = sigmoid_turun_penghasilan_kecil(data_mhsw[index][0])
    penghasilan_sedang = trapesium_penghasilan_sedang(data_mhsw[index][0])
    penghasilan_besar= trapesium_penghasilan_besar(data_mhsw[index][0])
    penghasilan_sangat_besar = sigmoid_naik_penghasilan_sangat_besar(data_mhsw[index][0])

    pengeluaran_kecil = trapesium_pengeluaran_kecil(data_mhsw[index][1])
    pengeluaran_sedang = trapesium_pengeluaran_sedang(data_mhsw[index][1])
    pengeluaran_besar = trapesium_pengeluaran_besar(data_mhsw[index][1])

    hasil = aturan_inferensi(penghasilan_sangat_besar,
            penghasilan_besar,penghasilan_sedang,
            penghasilan_kecil,pengeluaran_kecil,
            pengeluaran_sedang,pengeluaran_besar)

    hasil_defuzz = defuzz(hasil[1], hasil[0])
    data_kemungkinan[index] = hasil_defuzz

hasil_sort = sorted(data_kemungkinan.items(), key = lambda x: x[1], reverse=True)
print(hasil_sort)
```

Kesimpulan dan Output

Kesimpulan dari *Fuzzy Logic* ini adalah logika ini dapat digunakan untuk mencari 20 (dua puluh) mahasiswa yang layak mendapatkan bantuan registrasi sebesar 50%.

	id
2	2
3	5
4	11
5	14
6	19
7	26
8	28
9	34
10	36
11	37
12	39
13	48
14	13
15	8
16	27
17	35
18	6
19	10
20	25
21	12