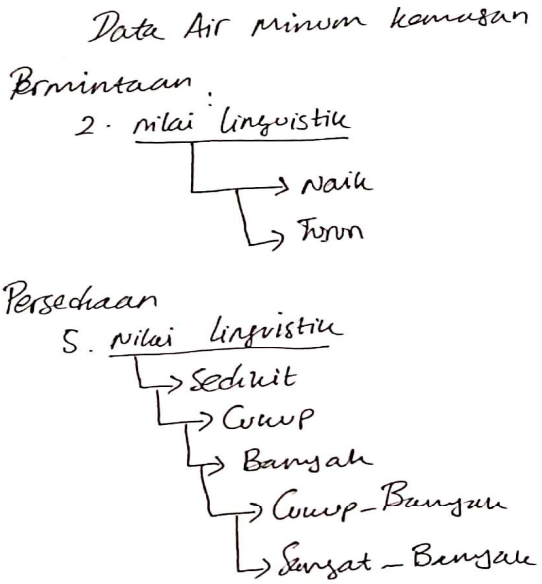
Nama : Esra Tarigan
NIM : 191011402557
Mata Kuliah : Kecerdasan Buatan
Dosen : Agung Perdananto, M. Kom
Shift : C-Sabtu



Hitung Jumlah Produksi Air minum kemasan dengan Metode Tsukamoto
Jika permintaan 2300 dan Persediaan 127

Variabel Permintaan

$$\text{Pmt Turun } [x] = \frac{x_{max} - x}{x_{max} - x_{min}}$$

$$= \frac{3500 - 2300}{3500 - 2100}$$

$$= {}^{1200}/_{1400}$$

$$= 0,8571$$

$$\text{Pmt Naik } [x] = \frac{x - x_{min}}{x_{max} - x_{min}}$$

$$= \frac{2300 - 2100}{3500 - 2100}$$

$$= {}^{200}/_{1400}$$

$$= 0,1428$$

Variabel Persediaan

Psd 127 Sedikit $= \frac{237-127}{237-118}$
$= 110/119$
$= 0,9243$

Psd 127 Cukup $= \frac{343-127}{343-237}$
$= 216/109$
$= 1,9816$

Psd 127 Banyak $= \frac{564-127}{564-343}$
$= 437/221$
$= 1,9773$

Psd 127 Cukup_banyak $= \frac{780-127}{780-564}$
$= 653/216$
$= 3,0231$

Psd 127 Sangat_banyak $= \frac{127-564}{780-564}$
$= 437/216$
$= 2,0231$

R_1 Pmt Turun [x] ∩ Psd Sedikit [Y] :

$\alpha_1 = \min([0{,}8571], [0{,}9243])$

$= 0{,}9243$

$z_1 = z_{max} - \alpha_1 (5000 - 1000)$

$= 5000 - 0{,}9243 (4000)$

$= 1302$

R_2 Pmt turun [x] ∩ Psd Cukup [Y]

$\alpha_2 = \min([0{,}8571], [1{,}9816])$

$= 1{,}9816$

$z_2 = 5000 - 1{,}9816 (4000)$

$= 2926$

R_3 Pmt turun [x] ∩ Psd banyak [Y]

$\alpha_3 = \min([0{,}8571], [1{,}9773])$

$= 1{,}9773$

$z_3 = 5000 - 1{,}9773 (4000)$

$= 2.909$

R_4 Pmt turun [x] ∩ Psd Cukup_banyak [Y]

$\alpha_4 = \min([0{,}8571], [3{,}0231])$

$= 3{,}0231$

$z_4 = 5000 - 3{,}0231 (4000)$

$= 7092$

R_5 Pmt turun [x] ∩ Psd Sangat_banyak [Y]

$\alpha_5 = \min([0{,}8571], [2{,}0231])$

$= 2{,}0231$

$z_5 = 5000 - 2{,}0231 (4000)$

$= 3092$

R6 Pmt naik [x] ∩ Psd Sedikit [Y]

$\alpha_6 = \min([0,1428], [0,9243])$

$= 0,9243$

$z_6 = \alpha(z_{max} - z_{min}) + z_{min}$

$= 0,9243\ (4000) + 1000$

$= 4697$

R7 Pmt naik [x] ∩ Psd Cukup [Y]

$\alpha_7 = \min([0,1428], [1,9816])$

$= 1,9816$

$z_7 = 1,9816\ (4000) + 1000$

$= 8926$

R8 Pmt naik [x] ∩ Psd banyak [Y]

$\alpha_8 = \min([0,1428], [1,9773])$

$= 1,9773$

$z_8 = 1,9773\ (4000) + 1000$

$= 8909$

R9 Pmt naik [x] ∩ Psd Cukup_banyak [Y]

$\alpha_9 = \min([0,1428], [3,0231])$

$= 3,0231$

$z_9 = 3,0231\ (4000) + 1000$

$= 13092$

R10 Pmt naik [x] ∩ Psd Sangat_banyak [Y]

$\alpha_{10} = \min([0,1428], [2,0231])$

$= 2,0231$

$z_{10} = 2,0231\ (4000) + 1000$

$= 9092$

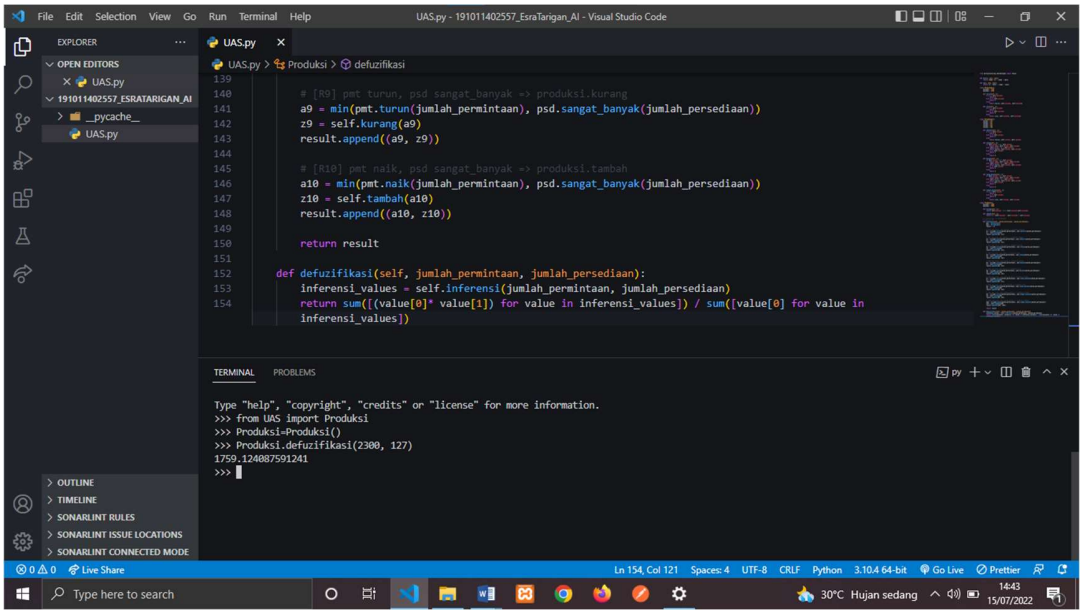
Defuzifikasi

$$Z = \frac{\alpha_1 * z_1 + \alpha_2 * z_2 + \alpha_3 * z_3 + \alpha_4 * z_4 + \alpha_5 * z_5 + \alpha_6 * z_6 + \alpha_7 * z_7 + \alpha_8 * z_8 + \alpha_9 * z_9 + \alpha_{10} z_{10}}{\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 + \alpha_7 + \alpha_8 + \alpha_9 + \alpha_{10}}$$

$$= \frac{1.203,4386 + 5,798,1616 + 5.751,9657 + 21.439,8252 + 6.255,4252 + 4.341,4371 + 17.687,7616 + 17.615,7657 + 39.578,4252 + 18.394,0252}{63.037}$$

$$= \frac{138.066,2311}{63.037}$$

$$= 2,1902$$



Source Code :

```
from multiprocessing.sharedctypes import Value


def down(x, xmin, xmax):
    return (xmax- x) / (xmax - xmin)

def up(x, xmin, xmax):
    return (x - xmin) / (xmax - xmin)

class Permintaan():
    minimum = 2100
    maximum = 3500

    def turun(self, x):
        if x >= self.maximum:
            return 0
        elif x <= self.minimum:
            return 1
        else:
            return down(x, self.minimum, self.maximum)
```

```
    def naik(self, x):
        if x >= self.maximum:
            return 1
        elif x <= self.minimum:
            return 0
        else:
            return up(x, self.minimum, self.maximum)

class Persediaan():
    value1 = 118
    value2 = 237
    value3 = 343
    value4 = 564
    value5 = 780

    def sedikit(self, x):
        if x >= self.value2:
            return 0
        elif x <= self.value1:
            return 1
        else:
            return down(x, self.value1, self.value2)

    def cukup(self, x):
        if self.value1 < x < self.value2:
            return up(x, self.value1, self.value2)
        elif self.value2 < x < self.value3:
            return down(x, self.value2, self.value3)
        elif x == self.value2:
            return 1
        else:
            return 0

    def banyak(self, x):
        if self.value2 < x < self.value3:
            return up(x, self.value2, self.value3)
        elif self.value3 < x < self.value4:
            return down(x, self.value3, self.value4)
        elif x == self.value3:
            return 1
        else:
```

```
            return 0

    def cukup_banyak(self, x):
        if self.value3 < x < self.value4:
            return up(x, self.value3, self.value4)
        elif self.value4 < x < self.value5:
            return down(x, self.value4, self.value5)
        elif x == self.value4:
            return 1
        else:
            return 0

    def sangat_banyak(self, x):
        if x >= self.value5:
            return 1
        elif x <= self.value4:
            return 0
        else:
            return up(x, self.value4, self.value5)

class Produksi():
    minimum = 1000
    maximum = 5000

    def kurang(self, x):
        return self.maximum - x * (self.maximum-self.minimum)

    def tambah(self, x):
        return x * (self.maximum - self.minimum) + self.minimum

    # 2 permintaan * 5 persediaan
    # rule === 10
    def inferensi(self, jumlah_permintaan, jumlah_persediaan):
        pmt = Permintaan()
        psd = Persediaan()
        result = []

        # [R1] pmt turun, psd sedikit => produksi.kurang
        a1 = min(pmt.turun(jumlah_permintaan),
psd.sedikit(jumlah_persediaan))
        z1 = self.kurang(a1)
```

```
        result.append((a1, z1))

        # [R2] pmt naik, psd sedikit => produksi.tambah
        a2 = min(pmt.naik(jumlah_permintaan),
psd.sedikit(jumlah_persediaan))
        z2 = self.tambah(a2)
        result.append((a2, z2))

        # [R3] pmt turun, psd cukup => produksi.kurang
        a3 = min(pmt.turun(jumlah_permintaan),
psd.cukup(jumlah_persediaan))
        z3 = self.kurang(a3)
        result.append((a3, z3))

        # [R4] pmt naik, psd cukup => produksi.naik
        a4 = min(pmt.naik(jumlah_permintaan),
psd.cukup(jumlah_persediaan))
        z4 = self.tambah(a4)
        result.append((a4, z4))

        # [R5] pmt turun, psd banyak => produksi.kurang
        a5 = min(pmt.turun(jumlah_permintaan),
psd.banyak(jumlah_persediaan))
        z5 = self.kurang(a5)
        result.append((a5, z5))

        # [R6] pmt naik, psd banyak => produksi.naik
        a6 = min(pmt.naik(jumlah_permintaan),
psd.banyak(jumlah_persediaan))
        z6 = self.tambah(a6)
        result.append((a6, z6))

        # [R7] pmt turun, psd cukup_banyak => produksi.kurang
        a7 = min(pmt.turun(jumlah_permintaan),
psd.cukup_banyak(jumlah_persediaan))
        z7 = self.kurang(a7)
        result.append((a7, z7))

        # [R8] pmt naik, psd cukup_banyak => produksi.naik
        a8 = min(pmt.naik(jumlah_permintaan),
psd.cukup_banyak(jumlah_persediaan))
```

```
        z8 = self.tambah(a8)
        result.append((a8, z8))

        # [R9] pmt turun, psd sangat_banyak => produksi.kurang
        a9 = min(pmt.turun(jumlah_permintaan),
psd.sangat_banyak(jumlah_persediaan))
        z9 = self.kurang(a9)
        result.append((a9, z9))

        # [R10] pmt naik, psd sangat_banyak => produksi.tambah
        a10 = min(pmt.naik(jumlah_permintaan),
psd.sangat_banyak(jumlah_persediaan))
        z10 = self.tambah(a10)
        result.append((a10, z10))

        return result

    def defuzifikasi(self, jumlah_permintaan, jumlah_persediaan):
        inferensi_values = self.inferensi(jumlah_permintaan,
jumlah_persediaan)
        return sum([(value[0]* value[1]) for value in
inferensi_values]) / sum([value[0] for value in inferensi_values])
```