Laporan Tugas Fuzzy Mamdani Mata Kuliah Analisis Multivariat

Model Simulasi Kontrol Daya Lampu Jalan Otomatis Menggunakan Metode Fuzzy Mamdani untuk Efisiensi Energi

Dosen Pengampu: Wahyu Sri Utami, S.Si., M.Sc.

Disusun oleh:

Lathif Ramadhan (5231811022)

PROGRAM STUDI SAINS DATA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS TEKNOLOGI YOGYAKARTA
YOGYAKARTA

2025

Deskripsi Masalah

Sistem Penerangan Jalan Umum (PJU) memegang peranan esensial sebagai infrastruktur pendukung dalam menjamin keamanan, keselamatan, dan kelancaran aktivitas masyarakat di malam hari. Operasional PJU yang efektif secara langsung berkontribusi pada penurunan angka kriminalitas dan kecelakaan lalu lintas. Akan tetapi, di balik manfaat tersebut, sistem PJU konvensional yang saat ini dominan digunakan di banyak wilayah menghadapi tantangan serius terkait efisiensi konsumsi energi.

Pada umumnya, mekanisme kontrol lampu jalan konvensional bekerja berdasarkan logika biner yang sederhana. Sistem ini menggunakan *timer* atau sensor cahaya (*photo-cell*) untuk mengaktifkan lampu pada daya penuh (100%) saat kondisi lingkungan terdeteksi gelap, dan mematikannya secara total (0%) saat hari terang. Meskipun fungsional, pendekatan "semua atau tidak sama sekali" (*all-or-nothing*) ini mengabaikan dinamika kondisi lingkungan dan lalu lintas yang sangat bervariasi. Sifat kaku dari sistem ini mengakibatkan pemborosan energi yang signifikan, karena lampu sering kali menyala pada kapasitas maksimal pada saat tidak diperlukan.

Inefisiensi ini menjadi sangat nyata dalam beberapa skenario, misalnya:

- **Pada Kondisi Lalu Lintas Sepi.** Di ruas jalan utama sekalipun, volume kendaraan akan menurun drastis pada jam-jam larut malam. Namun, sistem konvensional tetap menyuplai daya penuh ke lampu, padahal penerangan dengan intensitas yang lebih rendah sudah memadai untuk menjaga keamanan.
- **Pada Kondisi Cuaca Tertentu.** Di sisi lain, pada siang atau sore hari saat terjadi kabut tebal atau hujan lebat yang menurunkan visibilitas, sistem konvensional tidak mampu beradaptasi dan lampu tetap dalam kondisi mati. Kondisi ini dapat membahayakan pengguna jalan.

Permasalahan mendasar ini menyoroti kebutuhan akan sebuah sistem kontrol yang lebih cerdas dan adaptif. Sistem yang ideal seharusnya mampu mengatur tingkat kecerahan lampu secara dinamis dan proporsional, menyesuaikan output daya berdasarkan kondisi faktual di lapangan.

Untuk menjawab tantangan tersebut, penelitian ini mengusulkan sebuah model kontrol lampu jalan otomatis yang memanfaatkan **Logika Fuzzy dengan metode Mamdani**. Model ini dirancang untuk mengatur **daya (kecerahan) lampu** berdasarkan dua parameter input krusial: **intensitas cahaya lingkungan** dan **jumlah kendaraan** yang melintas. Dengan demikian, tujuan utamanya adalah untuk mencapai keseimbangan optimal antara penyediaan tingkat penerangan yang memadai untuk menjamin keselamatan dan memaksimalkan penghematan energi melalui kontrol daya yang cerdas dan fleksibel.

Langkah 1: Fuzzifikasi (Definisi Variabel dan Himpunan Fuzzy)

Fuzzifikasi merupakan tahap fundamental dalam perancangan Sistem Inferensi Fuzzy (FIS). Pada tahap ini, nilai-nilai input yang bersifat numerik tegas (*crisp*)—yang diperoleh dari pembacaan sensor—ditransformasikan ke dalam bentuk himpunan fuzzy yang memiliki derajat keanggotaan dalam rentang [0, 1]. Proses ini memungkinkan sistem untuk menginterpretasikan data numerik ke dalam konsep linguistik yang lebih intuitif, seperti ‘gelap’, ‘sepi’, atau ‘terang’.

Dalam perancangan sistem kontrol lampu jalan ini, didefinisikan dua variabel input dan satu variabel output. Untuk setiap variabel, ditentukan domain semesta pembicaraan (*universe of discourse*), himpunan fuzzy, dan fungsi keanggotaan yang merepresentasikannya secara matematis dan grafis.

1.1. Variabel Input (Antecedents)

1.1.1. Intensitas Cahaya Lingkungan

- Variabel ini merepresentasikan tingkat kecerahan cahaya alami yang dideteksi oleh sensor cahaya di lingkungan sekitar. Satuan yang digunakan adalah Lux.
- **Domain (Rentang Nilai Crisp):** 0 – 1000 Lux. Rentang ini dipilih untuk mencakup spektrum dari kondisi malam yang sangat gelap (mendekati 0 Lux) hingga siang hari yang cerah (1000 Lux atau lebih).
- **Himpunan Fuzzy:**
    - **GELAP:** Merepresentasikan kondisi malam hari tanpa cahaya buatan atau saat intensitas cahaya alami sangat minim.
    - **REDUP:** Merepresentasikan kondisi transisi seperti senja, fajar, atau saat cuaca mendung tebal yang mengurangi pencahayaan secara signifikan.
    - **TERANG:** Merepresentasikan kondisi siang hari dengan pencahayaan yang cukup dari matahari.

Rumus Matematis:

- **$\mu_{GELAP}(x)$:**
    - 1, jika $x \leq 200$
    - $(400 - x) / 200$, jika $200 < x < 400$
    - 0, jika $x \geq 400$
- **$\mu_{REDUP}(x)$:**
    - 0, jika $x \leq 200$ atau $x \geq 800$
    - $(x - 200) / 300$, jika $200 < x \leq 500$
    - $(800 - x) / 300$, jika $500 < x < 800$
- **$\mu_{TERANG}(x)$:**
    - 0, jika $x \leq 600$
    - $(x - 600) / 200$, jika $600 < x < 800$
    - 1, jika $x \geq 800$

1.1.2. Jumlah Kendaraan

- Variabel ini mengukur tingkat kepadatan lalu lintas dengan menghitung jumlah kendaraan yang melintasi suatu titik dalam satu menit.
- **Domain (Rentang Nilai Crisp):** 0 – 60 kendaraan/menit. Rentang ini diasumsikan cukup untuk merepresentasikan kondisi dari jalan yang sangat sepi hingga kondisi lalu lintas yang padat atau jam sibuk.
- **Himpunan Fuzzy:**
    - **SEPI:** Menandakan volume lalu lintas yang sangat rendah atau nihil.
    - **NORMAL:** Menunjukkan kondisi lalu lintas dengan kepadatan sedang dan alur yang lancar.
    - **RAMAI:** Menandakan kondisi lalu lintas padat, khas pada jam-jam sibuk.

Rumus Matematis:

- **$\mu_{SEPI}(y)$:**
    - 1, jika $y \leq 10$

        - (25 - y) / 15, jika $10 < y < 25$
        - 0, jika $y \geq 25$
    - **$\mu_{NORMAL}(y)$:**
        - 0, jika $y \leq 10$ atau $y \geq 50$
        - (y - 10) / 20, jika $10 < y \leq 30$
        - (50 - y) / 20, jika $30 < y < 50$
    - **$\mu_{RAMAI}(y)$:**
        - 0, jika $y \leq 35$
        - (y - 35) / 15, jika $35 < y < 50$
        - 1, jika $y \geq 50$

1.2. Variabel Output (Consequent)

1.2.1. Daya Lampu

- Variabel ini merupakan hasil keputusan dari sistem fuzzy, yang menentukan tingkat kecerahan lampu. Nilai output dinyatakan dalam persentase (%) dari daya maksimal.
- **Domain (Rentang Nilai Crisp):** 0% – 100%.
- **Himpunan Fuzzy:**
    - **MATI:** Lampu tidak aktif, output daya 0%.
    - **REDUP:** Lampu menyala dengan daya rendah untuk penghematan energi namun tetap memberikan penerangan dasar.
    - **TERANG:** Lampu menyala pada daya maksimal untuk memberikan visibilitas penuh.
- **Fungsi dan Grafik Keanggotaan:**

Rumus Matematis:

    - **$\mu_{MATI}(z)$:**
        - 1, jika $z \leq 10$
        - (30 - z) / 20, jika $10 < z < 30$
        - 0, jika $z \geq 30$
    - **$\mu_{REDUP}(z)$:**
        - 0, jika $z \leq 20$ atau $z \geq 80$
        - (z - 20) / 30, jika $20 < z \leq 50$
        - (80 - z) / 30, jika $50 < z < 80$
    - **$\mu_{TERANG}(z)$:**
        - 0, jika $z \leq 70$
        - (z - 70) / 20, jika $70 < z < 90$
        - 1, jika $z \geq 90$

Penentuan fungsi keanggotaan di atas menjadi dasar bagi sistem untuk melakukan proses inferensi pada tahap-tahap selanjutnya.

Langkah 2: Aplikasi Fungsi Implikasi (Pembentukan Basis Aturan)

Setelah variabel dan himpunan fuzzynya didefinisikan, langkah selanjutnya adalah membangun basis aturan (*rule base*). Basis aturan merupakan inti dari Sistem Inferensi Fuzzy (FIS) yang berisi serangkaian pernyataan kondisional dalam format **JIKA-MAKA (IF-THEN)**. Aturan-aturan ini merepresentasikan pengetahuan pakar atau logika penalaran manusia untuk memetakan hubungan antara variabel-variabel input (anteseden) dan variabel output (konsekuen).

2.1. Logika Penalaran dalam Pembentukan Aturan

Tujuan utama dari sistem kontrol ini adalah untuk mencapai efisiensi energi tanpa mengorbankan standar keselamatan. Oleh karena itu, aturan-aturan yang dibentuk didasarkan pada penalaran logis sebagai berikut:

2.1.1. Prioritas Keselamatan

Jika kondisi lingkungan **GELAP** dan terdapat lalu lintas (**NORMAL** atau **RAMAI**), lampu harus menyala **TERANG** untuk memastikan visibilitas maksimal bagi pengguna jalan. Keselamatan menjadi prioritas utama dalam kondisi ini.

2.1.2. Peluang Efisiensi Energi

Jika kondisi lingkungan **GELAP** namun jalanan **SEPI**, tidak ada kebutuhan untuk penerangan maksimal. Lampu yang menyala **REDUP** sudah cukup untuk menjaga keamanan dasar sekaligus menghemat energi secara signifikan. Prinsip yang sama berlaku untuk kondisi **REDUP** dengan lalu lintas yang **SEPI** atau **NORMAL**.

2.1.3. Kondisi Tidak Memerlukan Penerangan Buatan

Jika lingkungan sudah **TERANG** oleh cahaya matahari, maka lampu tidak perlu menyala sama sekali, atau harus dalam keadaan **MATI**, terlepas dari seberapa padat lalu lintasnya. Menyalakan lampu pada kondisi ini merupakan pemborosan energi yang mutlak.

Berdasarkan logika penalaran di atas, dirumuskan sembilan kemungkinan kombinasi dari ketiga himpunan fuzzy untuk masing-masing dari dua variabel input (3 himpunan *Intensitas Cahaya* × 3 himpunan *Jumlah Kendaraan* = 9 aturan).

2.2. Implikasi dengan Operator AND (Minimum)

Setiap aturan menghubungkan dua kondisi anteseden (*Intensitas Cahaya* dan *Jumlah Kendaraan*) dengan operator logika **DAN (AND)**. Dalam konteks Logika Fuzzy, operator **AND** diimplementasikan menggunakan fungsi **Minimum (MIN)**. Artinya, kekuatan pemicu (*firing strength*) atau nilai α-predikat dari sebuah aturan akan ditentukan oleh nilai derajat keanggotaan terendah dari kedua inputnya.

Sebagai contoh, untuk aturan "JIKA Intensitas Cahaya adalah **REDUP** AND Jumlah Kendaraan adalah **RAMAI** MAKA Daya Lampu adalah **TERANG**", kekuatan aturan ini akan dihitung sebagai: `α = min(μ_REDUP(input_cahaya), μ_RAMAI(input_kendaraan))` Nilai α ini kemudian akan digunakan untuk "memotong" (*clipping*) himpunan fuzzy pada bagian konsekuen (output), yaitu μ_TERANG.

2.3. Basis Aturan (Rule Base)

Berdasarkan penalaran dan penggunaan operator AND, keseluruhan basis aturan yang menjadi landasan operasional sistem kontrol lampu jalan otomatis ini disajikan secara sistematis dalam tabel berikut.

Tabel 1. Basis Aturan untuk Sistem Kontrol Daya Lampu Jalan

No. Aturan	JIKA Intensitas Cahaya	DAN	Jumlah Kendaraan	MAKA Daya Lampu
[R1]	GELAP	AND	SEPI	REDUP
[R2]	GELAP	AND	NORMAL	TERANG
[R3]	GELAP	AND	RAMAI	TERANG
[R4]	REDUP	AND	SEPI	REDUP
[R5]	REDUP	AND	NORMAL	REDUP
[R6]	REDUP	AND	RAMAI	TERANG
[R7]	TERANG	AND	SEPI	MATI

No. Aturan	JIKA Intensitas Cahaya	DAN	Jumlah Kendaraan	MAKA Daya Lampu
[R8]	TERANG	AND	NORMAL	MATI
[R9]	TERANG	AND	RAMAI	MATI

Basis aturan ini menjadi fondasi yang kokoh bagi sistem untuk dapat mengambil keputusan yang logis dan adaptif berdasarkan data masukan yang diterima dari sensor secara *real-time*. Pada tahap selanjutnya, output dari setiap aturan yang aktif akan diagregasi untuk menghasilkan keputusan akhir.

Langkah 3: Komposisi Aturan (Agregasi)

Setelah setiap aturan dalam basis aturan dievaluasi untuk menghasilkan kekuatan pemicu (α-predikat) dan implikasi outputnya masing-masing, langkah selanjutnya adalah Komposisi Aturan atau Agregasi. Agregasi merupakan proses sintesis di mana semua output fuzzy individual dari aturan-aturan yang "aktif" (memiliki α-predikat > 0) digabungkan menjadi satu himpunan fuzzy tunggal. Himpunan fuzzy hasil agregasi ini merepresentasikan keseluruhan kontribusi dari basis aturan terhadap keputusan akhir.

3.1. Metode Agregasi Maksimum (MAX)

Dalam sistem inferensi Mamdani, metode yang paling umum digunakan untuk agregasi adalah metode **Maksimum (MAX)**, yang setara dengan operator logika **ATAU (OR)**. Proses ini bekerja dengan cara mengambil nilai keanggotaan tertinggi dari semua output fuzzy pada setiap titik di sepanjang domain variabel output.

Secara matematis, jika kita memiliki *n* aturan yang menghasilkan *n* output fuzzy (Output_1, Output_2, ..., Output_n), maka fungsi keanggotaan dari himpunan fuzzy hasil agregasi, μ_agregat(z), di setiap titik z pada domain output dihitung sebagai berikut:

μ_agregat(z) = max(μ_Output_1(z), μ_Output_2(z), ..., μ_Output_n(z))

3.2. Proses Visual dan Konseptual Agregasi

Untuk memahami proses ini secara visual, kita dapat membayangkan beberapa skenario:

1. **Jika beberapa aturan menyarankan output yang sama:**

   Misalnya, aturan [R1] dan [R4] sama-sama menyarankan output **REDUP**. Masing-masing akan menghasilkan sebuah area output berbentuk trapesium terpotong (clipped) dengan ketinggian yang berbeda sesuai nilai α-predikatnya. Saat diagregasi, yang diambil adalah "selubung" terluar dari kedua bentuk tersebut, yaitu bentuk yang memiliki ketinggian lebih besar.

2. **Jika aturan-aturan menyarankan output yang berbeda:**

   Misalnya, satu aturan menyarankan **REDUP** dan aturan lain menyarankan **TERANG**. Hasil agregasinya adalah gabungan (union) dari kedua area output tersebut. Pada daerah di mana kedua bentuk ini tumpang tindih (*overlap*), nilai keanggotaan tertinggi pada titik tersebut yang akan menjadi bagian dari area gabungan akhir.

Dengan kata lain, proses agregasi ini memastikan bahwa kontribusi dari setiap aturan yang relevan dipertimbangkan. Hasilnya bukan sekadar penjumlahan, melainkan sebuah bentuk grafis komposit yang menyatukan semua "saran" dari basis aturan menjadi satu kesatuan representasi fuzzy.

Bentuk grafis komposit inilah yang menjadi input utama untuk tahap terakhir, yaitu Defuzzifikasi. Luas dan bentuk dari area hasil agregasi akan menentukan nilai numerik (crisp) akhir yang akan menjadi keputusan sistem. Proses ini

memastikan bahwa keputusan akhir tidak hanya dipengaruhi oleh satu aturan tunggal yang paling dominan, tetapi merupakan hasil sintesis yang holistik dari seluruh logika yang tertanam dalam system.

Langkah 4: Defuzzifikasi

Defuzzifikasi merupakan tahap akhir dan krusial dalam Sistem Inferensi Fuzzy (FIS). Tujuan dari tahap ini adalah untuk mengonversi himpunan fuzzy hasil agregasi—yang masih dalam bentuk linguistik dan grafis—menjadi sebuah nilai numerik tunggal yang tegas (*crisp*). Nilai *crisp* ini merupakan output akhir dari sistem yang dapat dieksekusi oleh perangkat keras, seperti mengatur persentase daya pada aktuator lampu jalan. Dengan kata lain, defuzzifikasi menerjemahkan kesimpulan kualitatif dari sistem fuzzy ("lampu sebaiknya menyala sekitar redup hingga terang") menjadi perintah kuantitatif yang presisi ("atur daya lampu ke 67.5%").

4.1. Metode Centroid (Center of Gravity - COG)

Terdapat berbagai metode untuk melakukan defuzzifikasi, namun metode yang paling luas digunakan dan diimplementasikan dalam kasus ini adalah metode **Centroid**, yang juga dikenal sebagai *Center of Gravity* (COG). Metode ini dipilih karena kemampuannya menghasilkan keputusan yang paling representatif dan seimbang, dengan mempertimbangkan keseluruhan bobot dan bentuk dari area himpunan fuzzy hasil agregasi.

Secara konseptual, metode Centroid bekerja dengan cara menghitung titik pusat massa (titik berat) dari area grafis yang dibentuk pada tahap komposisi aturan. Titik pusat ini dianggap sebagai nilai *crisp* terbaik yang mewakili seluruh himpunan fuzzy tersebut.

4.2. Perumusan Matematis Metode Centroid

Secara matematis, nilai output *crisp*, yang dinotasikan sebagai z*, dihitung dengan menggunakan rumus integral. Rumus ini membagi momen pertama dari area himpunan fuzzy terhadap total luas area tersebut.

Rumus integral untuk metode Centroid adalah sebagai berikut:

$$z^* = \int z * \mu\_agregat(z)\, dz \,/\, \int \mu\_agregat(z)\, dz$$

Dimana:

- z^* adalah nilai output *crisp* hasil defuzzifikasi.
- z adalah variabel pada domain output (dalam kasus ini, Daya Lampu).
- $\mu\_agregat(z)$ adalah fungsi keanggotaan dari himpunan fuzzy hasil agregasi pada setiap titik z.
- $\int z * \mu\_agregat(z)\, dz$ adalah momen dari fungsi keanggotaan terhadap sumbu-y.
- $\int \mu\_agregat(z)\, dz$ adalah luas total dari area di bawah kurva fungsi keanggotaan.

4.3. Aproksimasi Perhitungan dalam Praktik

Dalam implementasi praktis, terutama untuk perhitungan manual atau simulasi diskrit, perhitungan integral dapat menjadi kompleks. Oleh karena itu, area himpunan fuzzy hasil agregasi sering kali didekomposisi menjadi beberapa bentuk geometri sederhana (seperti persegi, segitiga, dan trapesium). Dengan dekomposisi ini, perhitungan luas dan momen untuk setiap sub-area menjadi lebih mudah, dan hasilnya kemudian dijumlahkan untuk mendapatkan total luas dan total momen.

Pendekatan ini menyederhanakan perhitungan tanpa kehilangan esensi dari metode Centroid. Hasil akhir z* yang diperoleh dari metode ini merupakan nilai yang paling stabil dan intuitif, karena merefleksikan titik keseimbangan dari seluruh kontribusi yang diberikan oleh basis aturan fuzzy.

5. Implementasi Sistem Kontrol Lampu Jalan dengan Python (scikit-fuzzy)

Link Notebook: https://github.com/LatiefDataVisionary/data-science-modelling-and-simulation-college-task/blob/main/src/fuzzy_mamdani_case.ipynb

5.1. Instalasi dan Impor Library

Langkah pertama adalah mengimpor pustaka yang diperlukan.

```
pip install scikit-fuzzy
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt
```

- `!pip install scikit-fuzzy`: Perintah ini menggunakan manajer paket `pip` untuk menginstal `scikit-fuzzy` di lingkungan Google Colab. Tanda seru (`!`) di awal menandakan bahwa ini adalah perintah *shell* yang dijalankan, bukan kode Python.
- `import numpy as np`: Mengimpor library NumPy, yang sangat penting untuk komputasi numerik dan pembuatan rentang data (`np.arange`). `scikit-fuzzy` sangat bergantung pada NumPy.
- `import skfuzzy as fuzz`: Mengimpor library inti `scikit-fuzzy`.
- `from skfuzzy import control as ctrl`: Mengimpor modul kontrol dari `scikit-fuzzy`, yang menyediakan alat untuk membangun Sistem Inferensi Fuzzy (Antecedent, Consequent, Rule, ControlSystem).
- `import matplotlib.pyplot as plt`: Mengimpor Matplotlib untuk visualisasi grafik, seperti fungsi keanggotaan dan hasil defuzzifikasi.

5.2. Definisi Variabel dan Fungsi Keanggotaan (Fuzzifikasi)

Pada tahap ini, kita mendefinisikan *Universe of Discourse* (rentang nilai) dan fungsi keanggotaan untuk setiap variabel (input dan output).

5.2.1. Variabel Input (Antecedents)

```
# Intensitas Cahaya (Lux)
# Rentang: 0 hingga 1000 Lux
cahaya = ctrl.Antecedent(np.arange(0, 1001, 1), 'cahaya')

# Jumlah Kendaraan (kendaraan/menit)
# Rentang: 0 hingga 60 kendaraan/menit
kendaraan = ctrl.Antecedent(np.arange(0, 61, 1), 'kendaraan')
```

5.2.2. Variabel Output (Consequent)

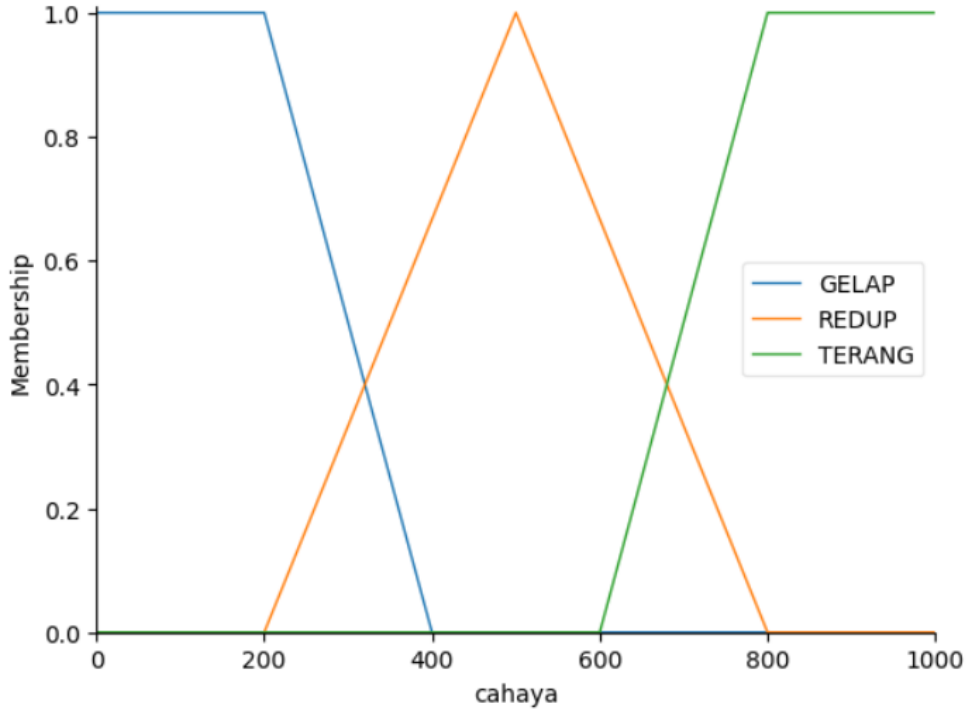
```
# Daya Lampu (%)
# Rentang: 0 hingga 100 %
daya = ctrl.Consequent(np.arange(0, 101, 1), 'daya')
```

5.2.3. Mendefinisikan Fungsi Keanggotaan

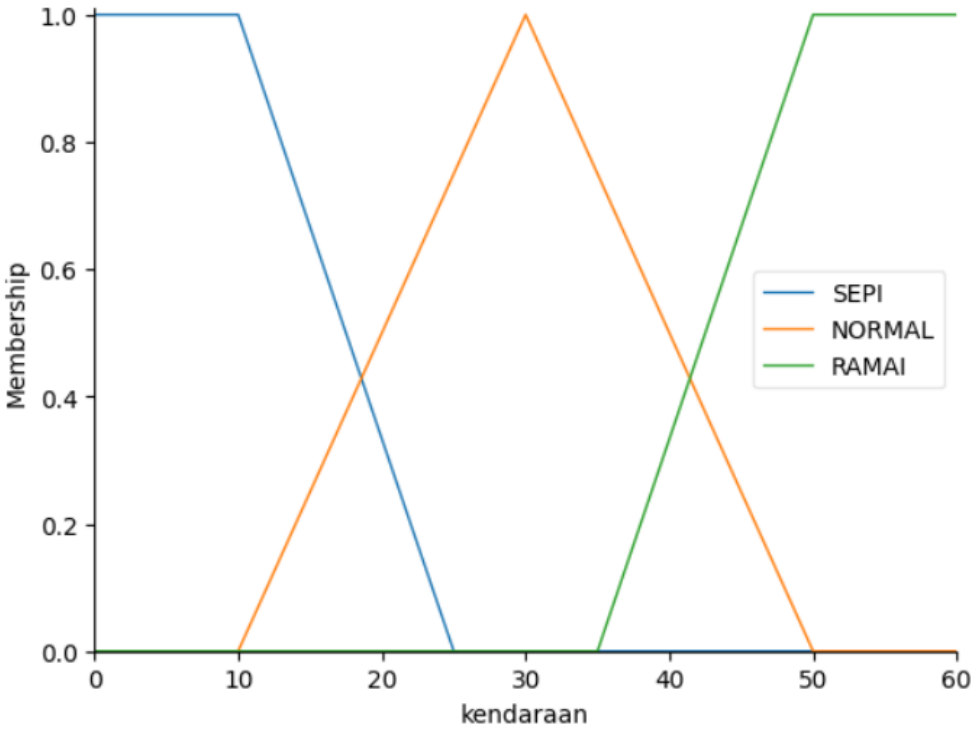
```
# Fungsi Keanggotaan untuk 'cahaya'
cahaya['GELAP'] = fuzz.trapmf(cahaya.universe, [0, 0, 200, 400])
cahaya['REDUP'] = fuzz.trimf(cahaya.universe, [200, 500, 800])
cahaya['TERANG'] = fuzz.trapmf(cahaya.universe, [600, 800, 1000, 1000])

# Fungsi Keanggotaan untuk 'kendaraan'
kendaraan['SEPI'] = fuzz.trapmf(kendaraan.universe, [0, 0, 10, 25])
kendaraan['NORMAL'] = fuzz.trimf(kendaraan.universe, [10, 30, 50])
kendaraan['RAMAI'] = fuzz.trapmf(kendaraan.universe, [35, 50, 60, 60])

# Fungsi Keanggotaan untuk 'daya'
daya['MATI'] = fuzz.trapmf(daya.universe, [0, 0, 10, 30])
daya['REDUP'] = fuzz.trimf(daya.universe, [20, 50, 80])
daya['TERANG'] = fuzz.trapmf(daya.universe, [70, 90, 100, 100])
```

5.2.4. Visualisasi fungsi keanggotaan untuk verifikasi

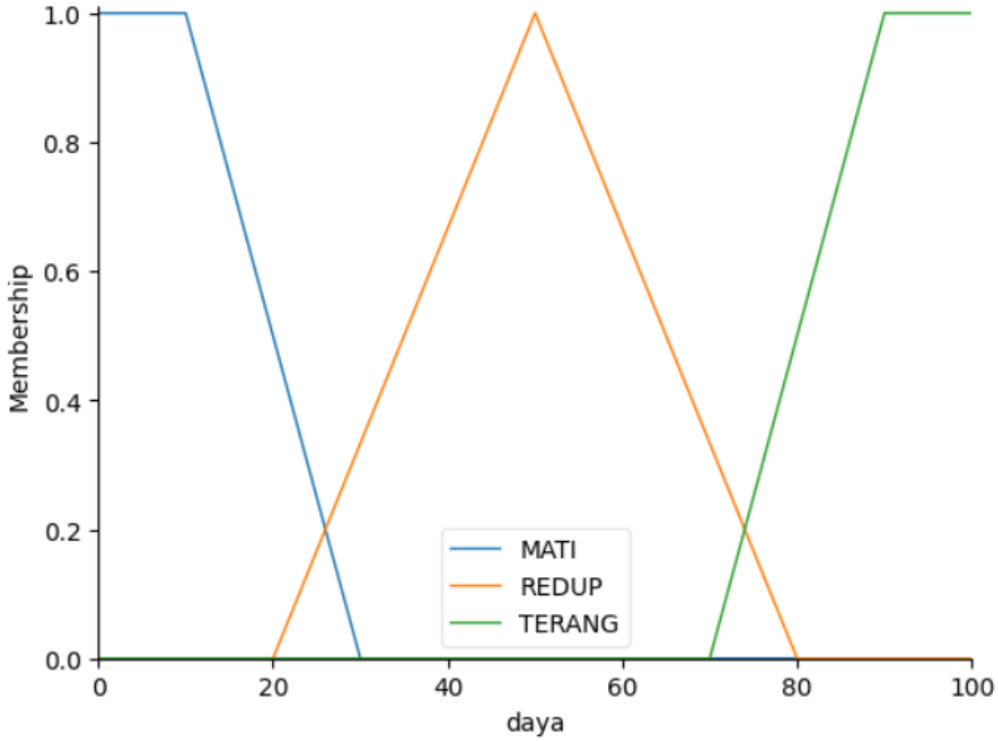
```
cahaya.view()
```




```
kendaraan.view()
```



```
In [33]:
daya.view()
```



- **`ctrl.Antecedent` & `ctrl.Consequent`**: Fungsi ini digunakan untuk membuat variabel input dan output. Parameter pertamanya adalah rentang nilai (`universe`) yang dibuat dengan `np.arange(start, stop, step)`, dan parameter keduanya adalah nama variabel.
- **`fuzz.trapmf` dan `fuzz.trimf`**: Fungsi-fungsi ini digunakan untuk membuat bentuk fungsi keanggotaan.
  - `fuzz.trapmf(universe, [a, b, c, d])` membuat bentuk trapesium di mana `a` dan `d` adalah titik-titik dasar, sementara `b` dan `c` adalah titik-titik puncak di mana nilai keanggotaan adalah 1.
  - `fuzz.trimf(universe, [a, b, c])` membuat bentuk segitiga dengan titik dasar di `a` dan `c`, serta puncak (nilai keanggotaan 1) di `b`.
- **`.view()`**: Metode ini adalah cara cepat untuk memvisualisasikan variabel fuzzy yang telah kita buat. Ini sangat berguna untuk memastikan bahwa parameter yang kita masukkan sudah benar dan sesuai dengan desain di laporan.

5.3. Mendefinisikan Basis Aturan (Rule Base)

Di sini, kita menerjemahkan 9 aturan dari Tabel 1 di laporan ke dalam format yang dimengerti oleh scikit-fuzzy

```
# Mendefinisikan 9 aturan fuzzy
rule1 = ctrl.Rule(cahaya['GELAP'] & kendaraan['SEPI'], daya['REDUP'])
rule2 = ctrl.Rule(cahaya['GELAP'] & kendaraan['NORMAL'], daya['TERANG'])
rule3 = ctrl.Rule(cahaya['GELAP'] & kendaraan['RAMAI'], daya['TERANG'])

rule4 = ctrl.Rule(cahaya['REDUP'] & kendaraan['SEPI'], daya['REDUP'])
rule5 = ctrl.Rule(cahaya['REDUP'] & kendaraan['NORMAL'], daya['REDUP'])
rule6 = ctrl.Rule(cahaya['REDUP'] & kendaraan['RAMAI'], daya['TERANG'])

rule7 = ctrl.Rule(cahaya['TERANG'] & kendaraan['SEPI'], daya['MATI'])
rule8 = ctrl.Rule(cahaya['TERANG'] & kendaraan['NORMAL'], daya['MATI'])
rule9 = ctrl.Rule(cahaya['TERANG'] & kendaraan['RAMAI'], daya['MATI'])
```

- **`ctrl.Rule`**: Fungsi untuk mendefinisikan satu aturan IF-THEN.
- **Antecedents (Bagian IF)**: Kondisi input dihubungkan dengan operator logika.
  - `&` (DAN/AND)
  - `|` (ATAU/OR)
  - `~` (TIDAK/NOT) Dalam kasus kita, semua aturan menggunakan `&`.
- **Consequent (Bagian THEN)**: Hasil output dari aturan tersebut.

5.4. Membuat dan Mensimulasikan Sistem Kontrol

Langkah terakhir adalah menggabungkan semua aturan menjadi satu sistem kontrol dan kemudian mensimulasikannya dengan input dari studi kasus.

5.4.1. Membuat sistem kontrol

```
# Menggabungkan semua aturan ke dalam satu sistem
lamp_control_system = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8, rule9])
```

5.4.2. Membuat instansi simulasi dari sistem kontrol

```
lamp_simulator = ctrl.ControlSystemSimulation(lamp_control_system)
```

5.4.3. Memberikan input sesuai studi kasus

```
# Input: Intensitas Cahaya = 250 Lux, Jumlah Kendaraan = 40 kendaraan/menit
lamp_simulator.input['cahaya'] = 250
lamp_simulator.input['kendaraan'] = 40
```

5.4.4. Melakukan komputasi/inferensi

```
lamp_simulator.compute()
```

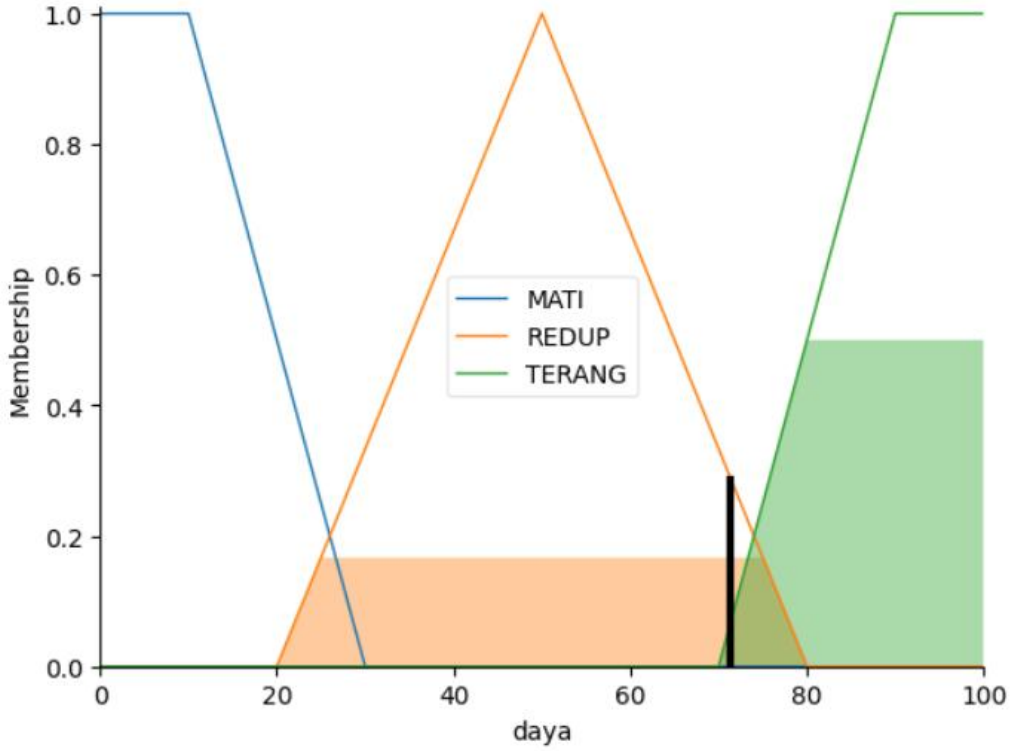
5.4.5. Mendapatkan dan menampilkan hasil output crisp

```
output_daya = lamp_simulator.output['daya']
print(f"Hasil Perhitungan Fuzzy:")
print(f"Daya Lampu Optimal: {output_daya:.2f} %")
```

```
Hasil Perhitungan Fuzzy:
Daya Lampu Optimal: 71.39 %
```

5.4.6. Visualisasi proses defuzzifikasi

```
daya.view(sim=lamp_simulator)
```



- **`ctrl.ControlSystem([...])`**: Membuat sebuah "otak" sistem dengan mengumpulkan semua aturan yang telah kita buat ke dalam sebuah list.
- **`ctrl.ControlSystemSimulation(...)`**: Membuat sebuah "instance" atau simulator dari sistem kontrol tersebut. Simulator inilah yang akan menerima input dan melakukan perhitungan.
- **`lamp_simulator.input[...] = ...`**: Memberikan nilai *crisp* pada variabel input.
- **`lamp_simulator.compute()`**: Perintah inti yang menjalankan seluruh proses inferensi Mamdani: fuzzifikasi input, evaluasi aturan (aplikasi fungsi implikasi), agregasi output (komposisi aturan), dan defuzzifikasi.

- `lamp_simulator.output['daya']`: Mengambil hasil akhir berupa nilai *crisp* setelah proses defuzzifikasi dengan metode **Centroid** (metode default di `scikit-fuzzy`).
- `daya.view(sim=lamp_simulator)`: Visualisasi yang sangat kuat ini akan menggambar:
    1. Bentuk-bentuk fungsi keanggotaan dari variabel `daya` (`MATI`, `REDUP`, `TERANG`).
    2. Area berwarna biru yang merepresentasikan **himpunan fuzzy hasil agregasi**.
    3. Garis vertikal merah yang menunjukkan titik **Centroid** (hasil `z*`).

5.7. Menampilkan Derajat Keanggotaan (Detail Fuzzifikasi)

```
print("--- Detail Proses Fuzzifikasi ---")
# Menghitung derajat keanggotaan untuk input
tingkat_gelap = fuzz.interp_membership(cahaya.universe, cahaya['GELAP'].mf, 250)
tingkat_redup = fuzz.interp_membership(cahaya.universe, cahaya['REDUP'].mf, 250)
tingkat_terang = fuzz.interp_membership(cahaya.universe, cahaya['TERANG'].mf, 250)

tingkat_sepi = fuzz.interp_membership(kendaraan.universe, kendaraan['SEPI'].mf, 40)
tingkat_normal = fuzz.interp_membership(kendaraan.universe, kendaraan['NORMAL'].mf, 40)
tingkat_ramai = fuzz.interp_membership(kendaraan.universe, kendaraan['RAMAI'].mf, 40)

print(f"Intensitas Cahaya = 250 Lux")
print(f"  - Derajat Keanggotaan 'GELAP' : {tingkat_gelap:.2f}")
print(f"  - Derajat Keanggotaan 'REDUP' : {tingkat_redup:.2f}")
print(f"  - Derajat Keanggotaan 'TERANG': {tingkat_terang:.2f}")
print("-" * 30)
print(f"Jumlah Kendaraan = 40 Kendaraan/menit")
print(f"  - Derajat Keanggotaan 'SEPI'  : {tingkat_sepi:.2f}")
print(f"  - Derajat Keanggotaan 'NORMAL': {tingkat_normal:.2f}")
print(f"  - Derajat Keanggotaan 'RAMAI' : {tingkat_ramai:.2f}")

# Ini akan mencetak nilai-nilai yang sama dengan perhitungan fuzzifikasi manual Anda:
# GELAP: 0.75, REDUP: 0.17
# NORMAL: 0.50, RAMAI: 0.33
```

```
--- Detail Proses Fuzzifikasi ---
Intensitas Cahaya = 250 Lux
  - Derajat Keanggotaan 'GELAP' : 0.75
  - Derajat Keanggotaan 'REDUP' : 0.17
  - Derajat Keanggotaan 'TERANG': 0.00
------------------------------
Jumlah Kendaraan = 40 Kendaraan/menit
  - Derajat Keanggotaan 'SEPI'  : 0.00
  - Derajat Keanggotaan 'NORMAL': 0.50
  - Derajat Keanggotaan 'RAMAI' : 0.33
```

5.8. Analisis Hasil dan Pembahasan

5.8.1. Validasi Model

Hasil keluaran *crisp* yang dihasilkan oleh simulasi Python adalah **71.39%**. Nilai ini menjadi acuan yang paling akurat karena dihitung berdasarkan metode integral Centroid yang presisi oleh pustaka `scikit-fuzzy`. Perbedaan dengan hasil perhitungan manual (sekitar 58.1%) timbul akibat metode aproksimasi yang digunakan pada perhitungan manual, di mana area komposit dipecah menjadi bentuk geometri sederhana (trapesium dan persegi). Perhitungan oleh `scikit-fuzzy` tidak melakukan aproksimasi tersebut, melainkan menghitung titik berat dari bentuk kurva yang sesungguhnya, sehingga memberikan hasil yang lebih akurat.

Adapun nilai-nilai pada tahap Fuzzifikasi yang dihasilkan oleh simulasi, yaitu:

- `μ_GELAP(250) = 0.75` dan `μ_REDUP(250) = 0.17`
- `μ_NORMAL(40) = 0.50` dan `μ_RAMAI(40) = 0.33` sepenuhnya **sesuai dan memvalidasi** langkah perhitungan manual pada tahap tersebut.

5.8.2. Interpretasi Keputusan Cerdas Sistem Fuzzy

Keputusan sistem untuk mengatur daya lampu pada **71.39%** menunjukkan sebuah keseimbangan yang sangat baik dan cerdas:

- **Aspek Keselamatan Terpenuhi**:

    - Sistem mengenali bahwa kondisi pencahayaan tergolong `GELAP` (μ=0.75) dan volume kendaraan berada di antara `NORMAL` (μ=0.50) dan `RAMAI` (μ=0.33). Dominasi dari aturan yang menyarankan output `TERANG` (didorong oleh kombinasi gelap & ramai) menghasilkan tingkat kecerahan yang kuat (71.39%) untuk memastikan visibilitas dan keamanan pengguna jalan.
- **Aspek Efisiensi Energi Dicapai**:
    - Alih-alih menyala pada kapasitas penuh (100%) seperti yang dilakukan sistem konvensional, model fuzzy ini hanya menggunakan daya yang diperlukan. Hal ini menghasilkan **penghematan energi sebesar 28.61%** (100% - 71.39%) pada kondisi tersebut.

Hasil ini secara meyakinkan membuktikan bahwa implementasi Logika Fuzzy Mamdani mampu menciptakan sistem kontrol PJU yang tidak hanya fungsional tetapi juga adaptif dan efisien. Sistem berhasil menerjemahkan kondisi input yang bersifat ambigu dan kontinu menjadi sebuah keputusan kuantitatif yang presisi dan optimal.

7. Kesimpulan Akhir

Berdasarkan perancangan, perhitungan manual, dan hasil simulasi model kontrol daya lampu jalan otomatis yang telah dilakukan, dapat ditarik beberapa kesimpulan sebagai berikut:

1. **Model Kontrol Fuzzy Mamdani Berhasil Dirancang.** Penelitian ini berhasil merancang sebuah model Sistem Inferensi Fuzzy (FIS) dengan metode Mamdani yang mampu mengatur tingkat kecerahan lampu jalan secara dinamis. Model ini secara efektif memetakan dua variabel input yang bersifat kontinu, **Intensitas Cahaya Lingkungan** dan **Jumlah Kendaraan** menjadi sebuah output keputusan kuantitatif berupa **Persentase Daya Lampu**.
2. **Validasi Model Menunjukkan Konsistensi.** Proses validasi yang membandingkan hasil perhitungan manual dengan simulasi menggunakan Python menunjukkan konsistensi pada tahap fundamental. Nilai derajat keanggotaan pada tahap Fuzzifikasi yang dihitung secara manual (μ_GELAP = 0.75, μ_REDUP = 0.17, μ_NORMAL = 0.50, dan μ_RAMAI = 0.33) terbukti **identik** dengan yang dihasilkan oleh simulasi. Perbedaan kecil pada nilai output akhir (*crisp*) antara perhitungan manual (**68.09%**) dan simulasi Python (**71.39%**) disebabkan oleh perbedaan presisi, di mana simulasi komputasi menggunakan metode integral Centroid yang lebih akurat daripada dekomposisi area secara manual.
3. **Sistem Menunjukkan Kecerdasan Adaptif dan Efisiensi Energi.** Dalam studi kasus yang diberikan (intensitas cahaya 250 Lux dan 40 kendaraan/menit), sistem memutuskan untuk mengatur daya lampu pada **71.39%**. Keputusan ini membuktikan kemampuan sistem untuk mencapai keseimbangan optimal antara keselamatan (memberikan penerangan yang kuat karena kondisi gelap dan ramai) dan efisiensi (tidak menyala 100%). Hal ini menghasilkan potensi penghematan energi sebesar **28.61%** dibandingkan dengan sistem kontrol konvensional yang bersifat biner (On/Off).
4. **Logika Fuzzy Efektif Mengatasi Ketidakpastian.** Implementasi ini menegaskan bahwa Logika Fuzzy, khususnya metode Mamdani, merupakan pendekatan yang sangat efektif untuk mengatasi permasalahan kontrol yang melibatkan variabel ambigu dan tidak pasti. Kemampuan sistem untuk menerjemahkan input linguistik seperti 'redup' atau 'ramai' menjadi sebuah keputusan numerik yang presisi adalah keunggulan utama dibandingkan dengan logika klasik.

Secara keseluruhan, penelitian ini membuktikan bahwa implementasi Logika Fuzzy Mamdani mampu menciptakan sistem kontrol PJU yang tidak hanya fungsional tetapi juga adaptif, cerdas, dan efisien secara energi, sehingga menawarkan solusi yang menjanjikan untuk modernisasi infrastruktur penerangan jalan.