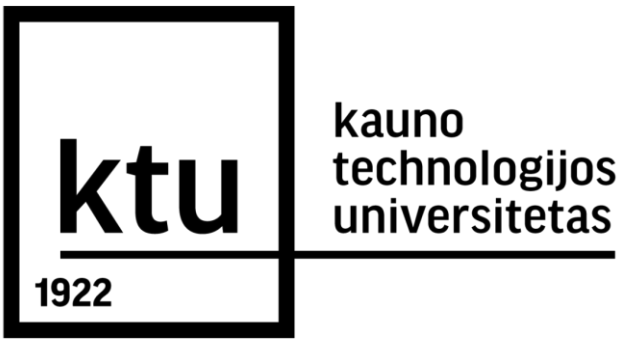
KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS



Intelektikos pagrindai (P176B101)

Laboratorinis darbas Nr.2

Atliko:

IFF-8/3 gr. studentas

Dovydas Zamas

2021 m. balandžio 16 d.

Priėmė:

Lekt. Paulauskaitė Tarasevičienė Agnė

KAUNAS 2021

Turinys

1. Užduotis

Užduotis - sukurkite sprendimo priėmimo sistemą remiantis miglotosios logikos teorija (rekomenduojama taikant Mamdani algoritmą, tačiau gali būti naudojamas ir Sugeno modelis). Duomenys gali būti naudojami realūs, iš atvirų šaltinių arba sugalvoti jūsų pačių (dažniausiai studentai sugalvoja savo duomenis ir patiems aktualią problemą – t.y. jūs tampate ekspertais). Sistemos programinė realizacija turi būti atlikta naudojant Python (arba C šeimos kalbomis).

Reikalavimai, kuriais remiantis bus vertinamas darbas pateikti žemiau:

1. Aiškus užduoties aprašas, t.y., koks uždavinys, pagal kokius duomenis ką reikia paskaičiuoti. Aprašomi kintamųjų matmenys, jie sugalvoti ar paimti iš išorinių šaltinių ir pan.;
2. Sistemos įvesčių kiekis ir fuzzy aibių skaičius: nuo 3 × 3 iki 4 × 4;
3. Sistemos išvesčių kiekis ir fuzzy aibių skaičius: nuo 1 × 3 iki 2 × 3;
4. Suformuotos ir pateiktos logiškos taisyklės naudojant du/tris skirtingus loginius kintamuosius (And, Or, Not).Visos taisyklės turi būti pateiktos ataskaitoje.
5. Pateikti metodai panaudoti implikacijai, agregacijai ir defuzifikacijai. Defuzifikacijai reikia panaudoti du skirtingus atsakymo skaičiavimo metodus: Centroid ir MOM(arba LOM).
6. Sudarius modelį reikia pateikti 3 testinių įvesčių reikšmių scenarijus ir gautus atsakymų rezultatus.
   Laboratorinio darbo vertinimas:
   - Ataskaita + programinis kodas - max 10 balai;
   - Individualus gynimas esant poreikiui (pvz., norite pasikelti balą).

2. Rezultatai

Užduotis buvo atlikta „Python“ kalba, PyCharm aplinkoje

Užduoties aprašas:

Sukurkite *fuzzy* sistemos modelį, kuris apskaičiuotų mašinos greitį atitinkamomis oro sąlygomis.

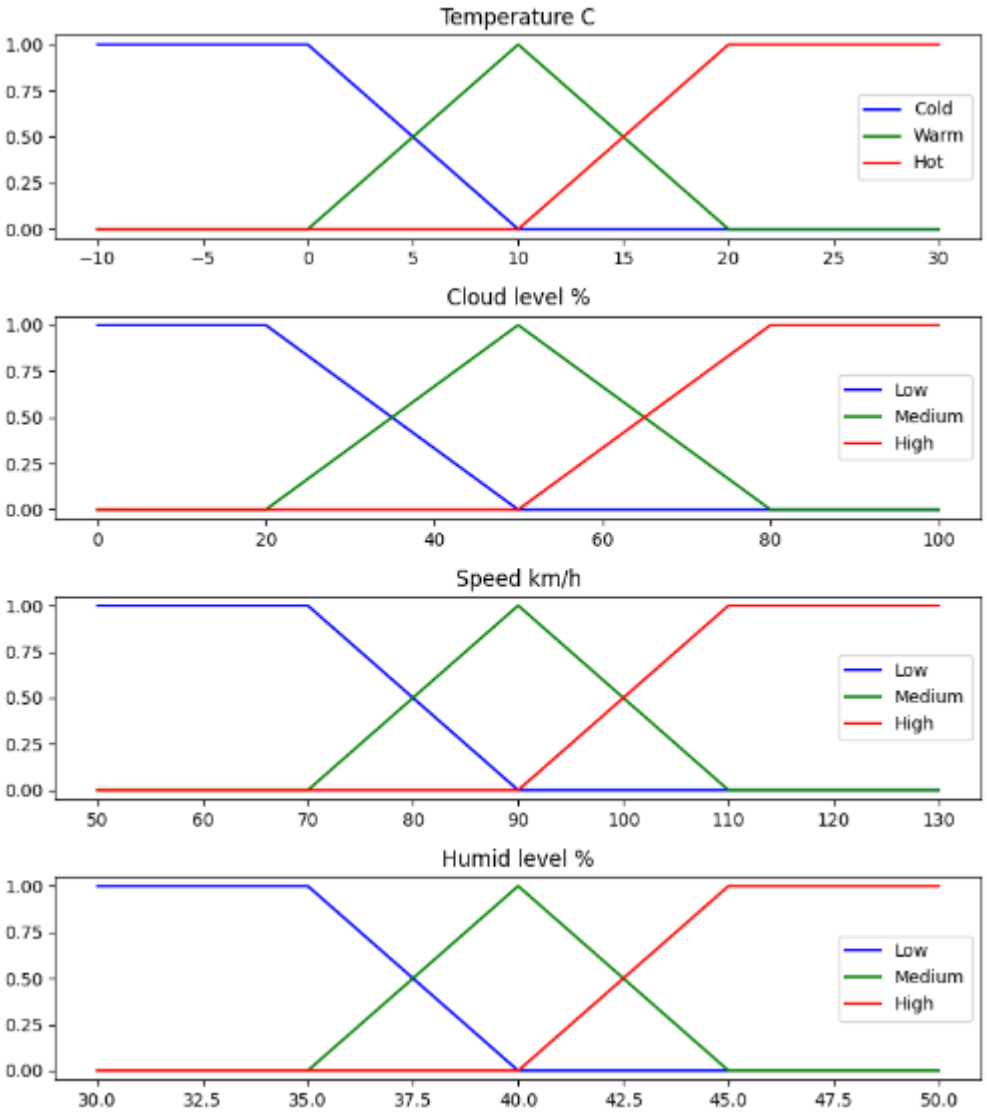
Įėjimo kintamieji:

- Oro temperatūra: [-10;30]; *fuzzy* aibės: šaltas, šiltas, karštas;
- Debesuotumo lygis: [0;100]; *fuzzy* aibės: žemas, vidutinis, aukšas;
- Mašinos greitis: [50;130]; *fuzzy* aibės: mažas, vidutinis, didelis;
- Oro drėgmės lygis: [30:50]; fuzzy aibės: mažas vidutinis, aukštas

Fuzzy taisyklės:

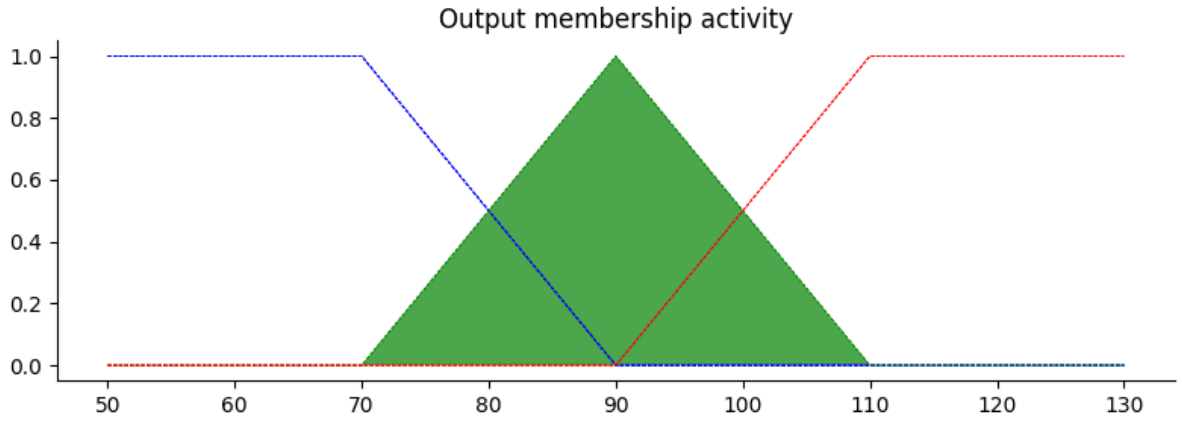
1. Jeigu debesuotumo lygis arba oro drėgmės lygis yra mažas ir oras yra karštas tai mašina galės važiuoti greitai
2. Jeigu debesuotumo lygis arba oro drėgmės lygis yra mažas ir oras yra šiltas tai mašina galės važiuoti greitai
3. Jeigu debesuotumo lygis arba oro drėgmės lygis yra mažas ir oras yra šaltas tai mašina galės važiuoti vidutiniškai
4. Jeigu debesuotumo arba drėgmės lygis yra vidutinis ir oras yra karštas tai mašina galės važiuoti greitai
5. Jeigu debesuotumo lygis arba drėmės lygis yra vidutinis ir oras yra šiltas tai mašina galės važiuoti vidutiniškai
6. Jeigu debesuotumo lygis arba oro drėmės lygis yra vidutinis ir oras yra šaltas tai mašina galės važiuoti lėtai
7. Jeigu debesuotumo lygis arba oro drėgmės lygis yra aukštas ir oras yra karštas tai mašina galės važiuoti lėtai
8. Jeigu debesuotumo lygis arba oro drėgmės lygis yra aukštas ir oras yra šiltas tai mašina galės važiuoti lėtai
9. Jeigu debesuotumo lygis arba oro drėgmės lygis yra aukštas ir oras yra šaltas tai mašina galės važiuoti lėtai

2.1. Įėjimo reikšmių atvaizdavimas

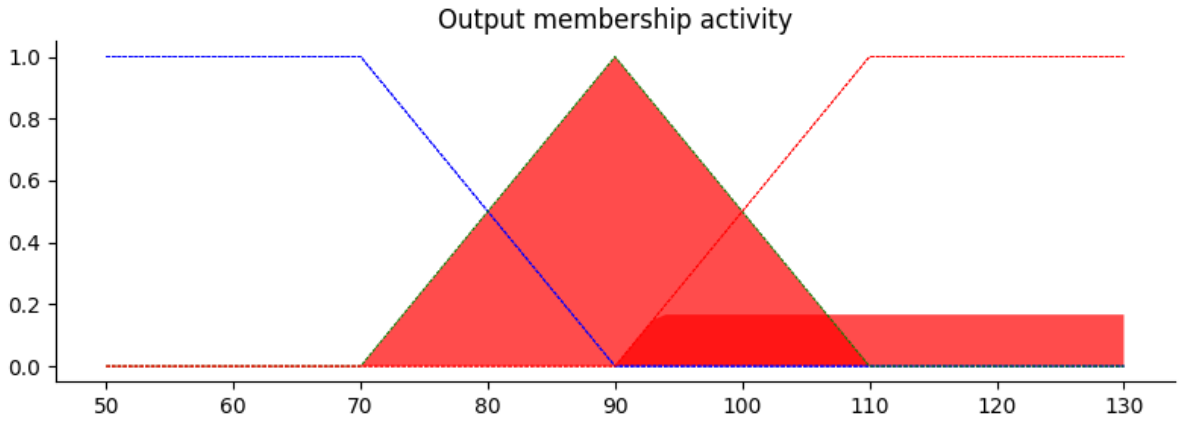


pav. 1 Įėjimo reikšmių grafikai

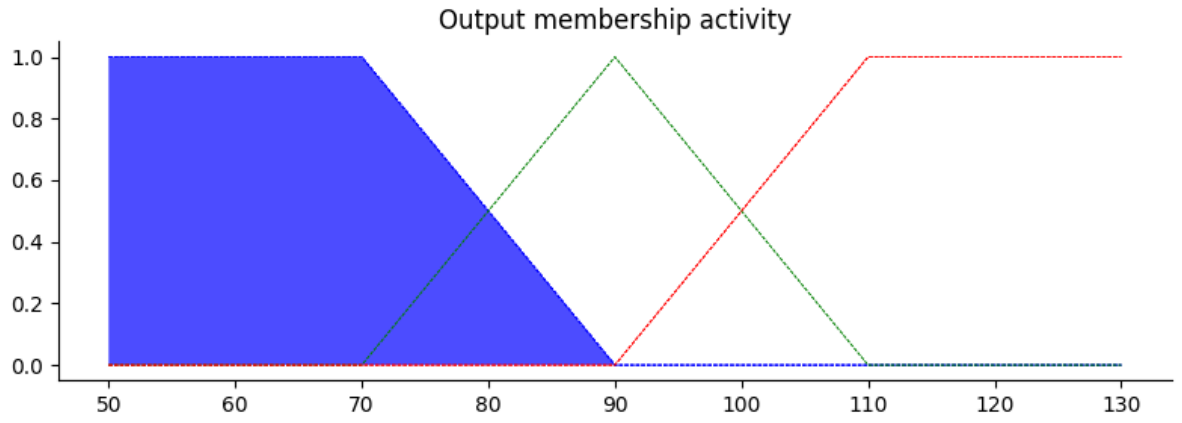
2.2. Taisyklių pritaikymas



pav. 2 Pritaikytų taisyklių, kai oro temp. -5C, debesuotumo lygis 20% ir drėgmės lygis 30% grafikas

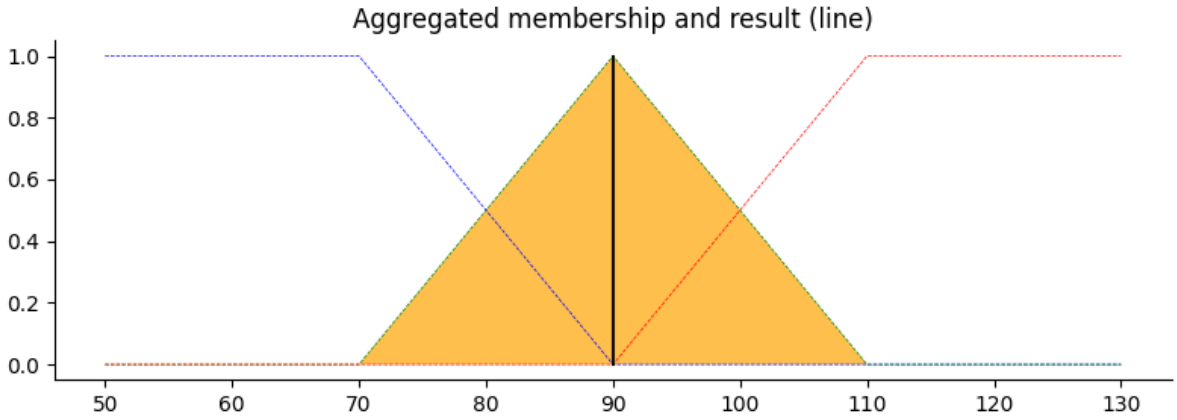


pav. 3 Pritaikytų taisyklių, kai oro temp. 10C, debesuotumo lygis 45% ir drėmės lygis 40 % grafikas

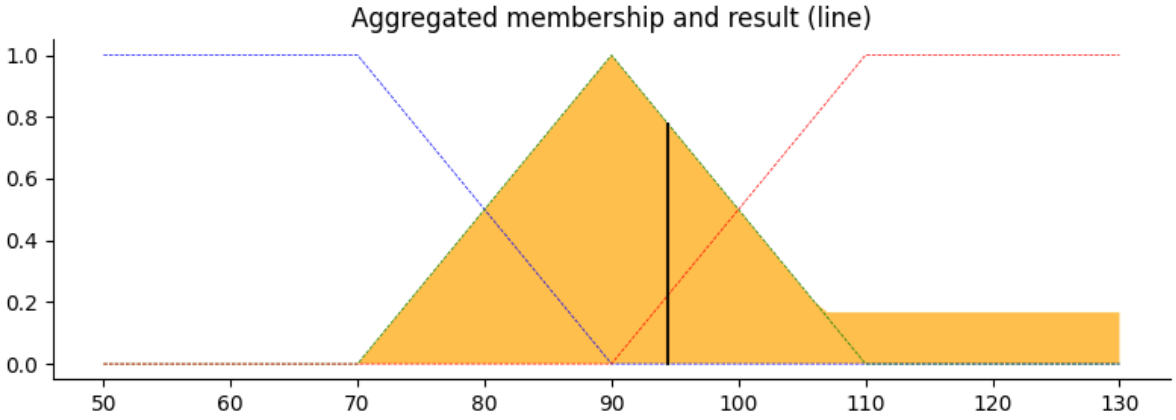


pav. 4 Pritaikytų taisyklių, kai oro temp. 25C, debesuotumo lygis 80% ir drėgmės lygis 50% grafikas

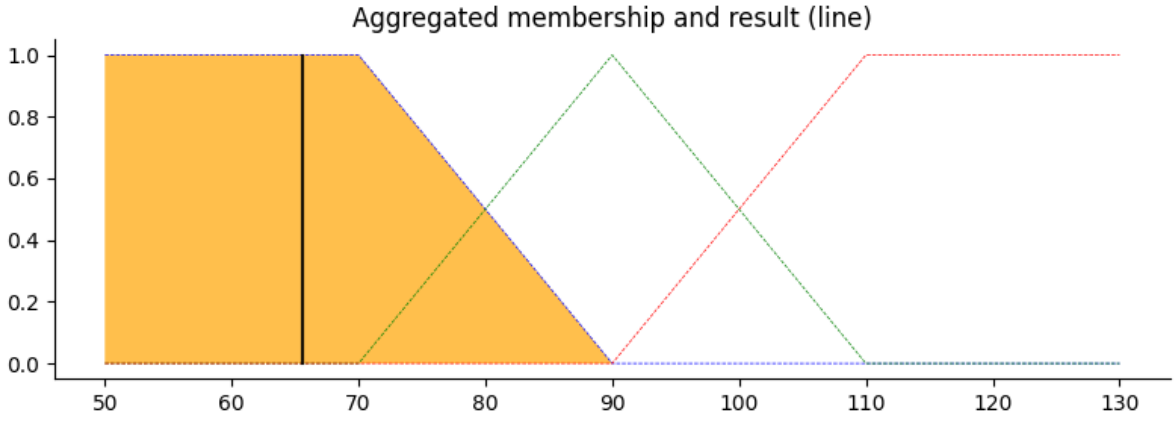
2.3. Defuzifikacija



pav. 5 Agregacijos ir atsakymo grafikas, kai oro temp. -5C, debesuotumo lygis 20% ir drėgmės lygis 30%



pav. 6 Agregacijos ir atsakymo grafikas, kai oro temp. 10C, debesuotumo lygis 45% ir drėgmės lygis 40%



pav. 7 Agregacijos ir atsakymo grafikas, kai oro temp. 25C, debesuotumo lygis 80% ir drėgmės lygis 50%

2.4. Atsakymai (defuzifikacija)

Debesuotumo lygis	Oro temperatūra	Oro drėgmės lygis	Mašinos greitis (Centroid) atsakymas	Mašinos greitis (MOM) atsakymas
20.00%	-5C	30.00%	90.0 km/h	90.0 km/h
45.00%	10C	40.00%	94.46 km/h	90.0 km/h
80.00%	25C	50.00%	65.55 km/h	60.0 km/h

3. Programos kodas

```
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt


def turn_off_top_right_axes(ax0, ax1, ax2):
    for ax in (ax0, ax1, ax2):
        ax.spines['top'].set_visible(False)
        ax.spines['right'].set_visible(False)
        ax.get_xaxis().tick_bottom()
        ax.get_yaxis().tick_left()


def plot_input_graphs(x_temp, x_cloud, x_speed, x_humid, temp_lo,
temp_md, temp_hi, cloud_lo, cloud_md, cloud_hi, speed_lo,
                      speed_md, speed_hi, humid_lo, humid_md, humid_hi
):
    fig, (ax0, ax1, ax2, ax3) = plt.subplots(nrows=4, figsize=(8, 9))

    ax0.plot(x_temp, temp_lo, 'b', linewidth=1.5, label='Cold')
    ax0.plot(x_temp, temp_md, 'g', linewidth=1.5, label='Warm')
    ax0.plot(x_temp, temp_hi, 'r', linewidth=1.5, label='Hot')
    ax0.set_title('Temperature C')
    ax0.legend()

    ax1.plot(x_cloud, cloud_lo, 'b', linewidth=1.5, label='Low')
    ax1.plot(x_cloud, cloud_md, 'g', linewidth=1.5, label='Medium')
    ax1.plot(x_cloud, cloud_hi, 'r', linewidth=1.5, label='High')
    ax1.set_title('Cloud level %')
    ax1.legend()

    ax2.plot(x_speed, speed_lo, 'b', linewidth=1.5, label='Low')
    ax2.plot(x_speed, speed_md, 'g', linewidth=1.5, label='Medium')
    ax2.plot(x_speed, speed_hi, 'r', linewidth=1.5, label='High')
    ax2.set_title('Speed km/h')
    ax2.legend()

    ax3.plot(x_humid, humid_lo, 'b', linewidth=1.5, label='Low')
    ax3.plot(x_humid, humid_md, 'g', linewidth=1.5, label='Medium')
    ax3.plot(x_humid, humid_hi, 'r', linewidth=1.5, label='High')
    ax3.set_title('Humid level %')
    ax3.legend()
    plt.tight_layout()
    plt.show()
```

```
def plot_applied_rules_graphs(x_speed, speed0, speed_lo, speed_md,
speed_hi, speed_activation_lo1, speed_activation_lo2,
                              speed_activation_lo3,
                              speed_activation_lo4,
speed_activation_md1, speed_activation_md2,
                              speed_activation_hi1,
speed_activation_hi2, speed_activation_hi3):
    fig, ax0 = plt.subplots(figsize=(8, 3))

    ax0.fill_between(x_speed, speed0, speed_activation_lo1,
facecolor='b', alpha=0.7)
    ax0.plot(x_speed, speed_lo, 'b', linewidth=0.5, linestyle='--', )
    ax0.fill_between(x_speed, speed0, speed_activation_lo2,
facecolor='b', alpha=0.7)
    ax0.plot(x_speed, speed_lo, 'b', linewidth=0.5, linestyle='--', )
    ax0.fill_between(x_speed, speed0, speed_activation_lo3,
facecolor='g', alpha=0.7)
    ax0.plot(x_speed, speed_lo, 'b', linewidth=0.5, linestyle='--')
    ax0.fill_between(x_speed, speed0, speed_activation_lo4,
facecolor='g', alpha=0.7)
    ax0.plot(x_speed, speed_lo, 'b', linewidth=0.5, linestyle='--')
    ax0.fill_between(x_speed, speed0, speed_activation_md1,
facecolor='g', alpha=0.7)
    ax0.plot(x_speed, speed_md, 'g', linewidth=0.5, linestyle='--')
    ax0.fill_between(x_speed, speed0, speed_activation_md2,
facecolor='r', alpha=0.7)
    ax0.plot(x_speed, speed_md, 'g', linewidth=0.5, linestyle='--')
    ax0.fill_between(x_speed, speed0, speed_activation_hi1,
facecolor='r', alpha=0.7)
    ax0.plot(x_speed, speed_hi, 'r', linewidth=0.5, linestyle='--')
    ax0.fill_between(x_speed, speed0, speed_activation_hi2,
facecolor='r', alpha=0.7)
    ax0.plot(x_speed, speed_hi, 'r', linewidth=0.5, linestyle='--')
    ax0.fill_between(x_speed, speed0, speed_activation_hi3,
facecolor='r', alpha=0.7)
    ax0.plot(x_speed, speed_hi, 'r', linewidth=0.5, linestyle='--')
    ax0.set_title('Output membership activity')
    for ax in (ax0,):
        ax.spines['top'].set_visible(False)
        ax.spines['right'].set_visible(False)
        ax.get_xaxis().tick_bottom()
        ax.get_yaxis().tick_left()

    plt.tight_layout()
    plt.show()


def apply_rules(x_speed, temp_level_lo, temp_level_md, temp_level_hi,
cloud_level_lo, cloud_level_md, cloud_level_hi,
                speed_lo, speed_md, speed_hi, humid_level_lo,
humid_level_md, humid_level_hi):
    # Jeigu debesuotumo lygis arba oro drėgmės lygis yra mažas ir oras
yra karštas tai mašina galės važiuoti greitai
    active_rule1 =
np.fmin(np.fmax(cloud_level_lo,humid_level_lo),temp_level_hi)
    speed_activation_hi1 = np.fmin(active_rule1, speed_hi)
    # Jeigu debesuotumo lygis arba oro drėgmės lygis yra mažas ir oras
yra šiltas tai  mašina galės važiuoti greitai
    active_rule2 =
np.fmin(np.fmax(cloud_level_lo,humid_level_lo),temp_level_md)
```

```
    speed_activation_hi2 = np.fmin(active_rule2, speed_hi)
    # Jeigu debesuotumo lygis arba oro drėgmės lygis yra mažas ir oras
yra šaltas tai mašina galės važiuoti vidutiniškai
    active_rule3 =
np.fmin(np.fmax(cloud_level_lo,humid_level_lo),temp_level_lo)
    speed_activation_md1 = np.fmin(active_rule3, speed_md)
    # Jeigu debesuotumo arba drėgmės lygis yra vidutinis ir oras yra
karštas tai mašina galės važiuoti greitai
    active_rule4 =
np.fmin(np.fmax(cloud_level_md,humid_level_md),temp_level_hi)
    speed_activation_hi3 = np.fmin(active_rule4, speed_hi)
    # Jeigu debesuotumo lygis arba drėmės lygis yra vidutinis ir oras
yra šiltas tai mašina galės važiuoti vidutiniškai
    active_rule5 =
np.fmin(np.fmax(cloud_level_md,humid_level_md),temp_level_md)
    speed_activation_md2 = np.fmin(active_rule5, speed_md)
    # Jeigu debesuotumo lygis arba oro drėmės lygis yra vidutinis ir
oras yra šaltas tai mašina galės važiuoti lėtai
    active_rule6 =
np.fmin(np.fmax(cloud_level_md,humid_level_md),temp_level_lo)
    speed_activation_lo1 = np.fmin(active_rule6, speed_lo)
    # Jeigu debesuotumo lygis arba oro drėgmės lygis yra aukštas ir oras
yra karštas tai mašina galės važiuoti lėtai
    active_rule7 =
np.fmin(np.fmax(cloud_level_hi,humid_level_hi),temp_level_hi)
    speed_activation_lo2 = np.fmin(active_rule7, speed_lo)
    # Jeigu debesuotumo lygis arba oro drėgmės lygis yra aukštas ir oras
yra šiltas tai mašina galės važiuoti lėtai
    active_rule8 =
np.fmin(np.fmax(cloud_level_hi,humid_level_hi),temp_level_md)
    speed_activation_lo3 = np.fmin(active_rule8, speed_lo)
    # Jeigu debesuotumo lygis arba oro drėgmės lygis yra aukštas ir oras
yra šaltas tai mašina galės važiuoti lėtai
    active_rule9 =
np.fmin(np.fmax(cloud_level_hi,humid_level_hi),temp_level_lo)
    speed_activation_lo4 = np.fmin(active_rule9, speed_lo)

    speed0 = np.zeros_like(x_speed)
    plot_applied_rules_graphs(x_speed, speed0, speed_lo, speed_md,
speed_hi, speed_activation_lo1, speed_activation_lo2,
                              speed_activation_lo3,
speed_activation_lo4, speed_activation_md1, speed_activation_md2,
                              speed_activation_hi1,
speed_activation_hi2, speed_activation_hi3)
    aggregated_lo = np.fmax(speed_activation_lo1,
                            np.fmax(speed_activation_lo2,

np.fmax(speed_activation_lo3,speed_activation_lo4)))

    aggregated_md = np.fmax(speed_activation_md1,speed_activation_md2)
    aggregated_hi = np.fmax(speed_activation_hi1,

np.fmax(speed_activation_hi2,speed_activation_hi3))

    aggregated = np.fmax(aggregated_lo,
                         np.fmax(aggregated_md, aggregated_hi))

    speed = fuzz.defuzz(x_speed, aggregated, 'centroid')
    speed_activation = fuzz.interp_membership(x_speed, aggregated,
speed)  # for plot
    print("Greitis = "+str(fuzz.defuzz(x_speed, aggregated,
```

```
'centroid'))+" km/h CENTROID")
    print("Greitis = "+str(fuzz.defuzz(x_speed, aggregated, 'mom'))+"
km/h MOM")

    fig, ax0 = plt.subplots(figsize=(8, 3))

    ax0.plot(x_speed, speed_lo, 'b', linewidth=0.5, linestyle='--', )
    ax0.plot(x_speed, speed_md, 'g', linewidth=0.5, linestyle='--')
    ax0.plot(x_speed, speed_hi, 'r', linewidth=0.5, linestyle='--')
    ax0.fill_between(x_speed, speed0, aggregated, facecolor='Orange',
alpha=0.7)
    ax0.plot([speed, speed], [0, speed_activation], 'k', linewidth=1.5,
alpha=0.9)
    ax0.set_title('Aggregated membership and result (line)')

    for ax in (ax0,):
        ax.spines['top'].set_visible(False)
        ax.spines['right'].set_visible(False)
        ax.get_xaxis().tick_bottom()
        ax.get_yaxis().tick_left()

    plt.tight_layout()
    plt.show()


def execute():
    # ----------DATA-------------#
    x_temp = np.arange(-10, 31, 1)
    x_cloud = np.arange(0, 101, 1)
    x_humid = np.arange(30,51,1)
    x_speed = np.arange(50, 131, 1)
    # ---------------------------#

    # -------GRAPHSDATA-----------#
    temp_lo = fuzz.trapmf(x_temp, [-10, -10, 0, 10])
    temp_md = fuzz.trimf(x_temp, [0, 10, 20])
    temp_hi = fuzz.trapmf(x_temp, [10, 20, 30, 30])
    cloud_lo = fuzz.trapmf(x_cloud, [0, 0, 20, 50])
    cloud_md = fuzz.trimf(x_cloud, [20, 50, 80])
    cloud_hi = fuzz.trapmf(x_cloud, [50, 80, 100, 100])
    speed_lo = fuzz.trapmf(x_speed, [50, 50, 70, 90])
    speed_md = fuzz.trimf(x_speed, [70, 90, 110])
    speed_hi = fuzz.trapmf(x_speed, [90, 110, 130, 130])
    humid_lo = fuzz.trapmf(x_humid,[30,30,35,40])
    humid_md = fuzz.trimf(x_humid,[35,40,45])
    humid_hi = fuzz.trapmf(x_humid,[40,45,50,50])
    # ---------------------------#

    plot_input_graphs(x_temp, x_cloud, x_speed, x_humid, temp_lo,
temp_md, temp_hi, cloud_lo, cloud_md,
                      cloud_hi, speed_lo, speed_md, speed_hi,humid_lo,
humid_md, humid_hi)

    temp_level_lo = fuzz.interp_membership(x_temp, temp_lo, 25)
    temp_level_md = fuzz.interp_membership(x_temp, temp_md, 25)
    temp_level_hi = fuzz.interp_membership(x_temp, temp_hi, 25)

    cloud_level_lo = fuzz.interp_membership(x_cloud, cloud_lo, 80)
    cloud_level_md = fuzz.interp_membership(x_cloud, cloud_md, 80)
    cloud_level_hi = fuzz.interp_membership(x_cloud, cloud_hi, 80)
```

```
    humid_level_lo = fuzz.interp_membership(x_humid, humid_lo, 50)
    humid_level_md = fuzz.interp_membership(x_humid, humid_md, 50)
    humid_level_hi = fuzz.interp_membership(x_humid, humid_hi, 50)
    apply_rules(x_speed, temp_level_lo, temp_level_md, temp_level_hi,
cloud_level_lo, cloud_level_md,
                cloud_level_hi, speed_lo, speed_md, speed_hi,
humid_level_lo,humid_level_md, humid_level_hi)


if __name__ == "__main__":
    execute()

    plt.tight_layout()
    plt.show()


def execute():
    # -----------DATA-------------#
    x_temp = np.arange(-10, 31, 1)
    x_cloud = np.arange(0, 101, 1)
    x_speed = np.arange(50, 131, 1)
    # ----------------------------#

    # -------GRAPHSDATA-----------#
    temp_lo = fuzz.trapmf(x_temp, [-10, -10, 0, 10])
    temp_md = fuzz.trimf(x_temp, [0, 10, 20])
    temp_hi = fuzz.trapmf(x_temp, [10, 20, 30, 30])
    cloud_lo = fuzz.trapmf(x_cloud, [0, 0, 20, 50])
    cloud_md = fuzz.trimf(x_cloud, [20, 50, 80])
    cloud_hi = fuzz.trapmf(x_cloud, [50, 80, 100, 100])
    speed_lo = fuzz.trapmf(x_speed, [50, 50, 70, 90])
    speed_md = fuzz.trimf(x_speed, [70, 90, 110])
    speed_hi = fuzz.trapmf(x_speed, [90, 110, 130, 130])
    # ----------------------------#

    plot_input_graphs(x_temp, x_cloud, x_speed, temp_lo, temp_md,
temp_hi, cloud_lo, cloud_md,
                      cloud_hi, speed_lo, speed_md, speed_hi)

    temp_level_lo = fuzz.interp_membership(x_temp, temp_lo, 25)
    temp_level_md = fuzz.interp_membership(x_temp, temp_md, 25)
    temp_level_hi = fuzz.interp_membership(x_temp, temp_hi, 25)

    cloud_level_lo = fuzz.interp_membership(x_cloud, cloud_lo, 80)
    cloud_level_md = fuzz.interp_membership(x_cloud, cloud_md, 80)
    cloud_level_hi = fuzz.interp_membership(x_cloud, cloud_hi, 80)
    apply_rules(x_speed, temp_level_lo, temp_level_md, temp_level_hi,
cloud_level_lo, cloud_level_md,
                cloud_level_hi, speed_lo, speed_md, speed_hi)


if __name__ == "__main__":
    execute()
```