```
import pandas as pd

# Load the dataset
file_path = '/content/smart_home_energy_usage_dataset.csv'
data = pd.read_csv(file_path)

# Display the first few rows to understand the data
data.head()
```

	timestamp	home_id	energy_consumption_kWh	temperature_setting_C	occupancy_status	appliance	usage_duration_minutes	season
0	01-01-2023 00:00	44	2.87	22.1	Occupied	Refrigerator	111	Spring
1	01-01-2023 01:00	81	0.56	15.4	Occupied	HVAC	103	Summer

```
data.describe()
```

	home_id	energy_consumption_kWh	temperature_setting_C	usage_duration_minutes	holiday
count	1000000.000000	1000000.000000	1000000.000000	1000000.000000	1000000.000000
mean	50.019812	2.548839	19.999284	59.505089	0.099588
std	28.605155	1.415527	2.887678	34.651890	0.299450
min	1.000000	0.100000	15.000000	0.000000	0.000000
25%	25.000000	1.320000	17.500000	30.000000	0.000000
50%	50.000000	2.550000	20.000000	59.000000	0.000000
75%	75.000000	3.780000	22.500000	90.000000	0.000000
max	99.000000	5.000000	25.000000	119.000000	1.000000

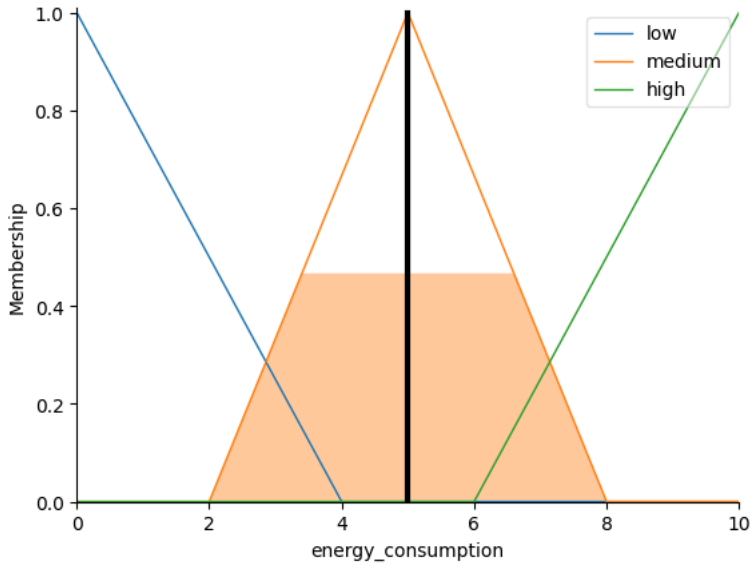
```
!pip install -U scikit-fuzzy
```

```
Requirement already satisfied: scikit-fuzzy in /usr/local/lib/python3.10/dist-packages (0.5.0)
```

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Define fuzzy variables
temperature = ctrl.Antecedent(np.arange(15, 31, 1), 'temperature')
occupancy = ctrl.Antecedent(np.arange(0, 2, 1), 'occupancy')  # 0 for Unoccupied, 1 for Occupied
usage_duration = ctrl.Antecedent(np.arange(0, 181, 1), 'usage_duration')
energy_consumption = ctrl.Consequent(np.arange(0, 11, 1), 'energy_consumption')

# Define membership functions for each variable
temperature['cold'] = fuzz.trapmf(temperature.universe, [15, 15, 18, 21])
temperature['moderate'] = fuzz.trimf(temperature.universe, [18, 22, 26])
temperature['hot'] = fuzz.trapmf(temperature.universe, [23, 26, 30, 30])

occupancy['unoccupied'] = fuzz.trimf(occupancy.universe, [0, 0, 1])
occupancy['occupied'] = fuzz.trimf(occupancy.universe, [0, 1, 1])

usage_duration['short'] = fuzz.trimf(usage_duration.universe, [0, 0, 60])
usage_duration['medium'] = fuzz.trimf(usage_duration.universe, [30, 90, 150])
usage_duration['long'] = fuzz.trimf(usage_duration.universe, [120, 180, 180])

energy_consumption['low'] = fuzz.trimf(energy_consumption.universe, [0, 0, 4])
energy_consumption['medium'] = fuzz.trimf(energy_consumption.universe, [2, 5, 8])
energy_consumption['high'] = fuzz.trimf(energy_consumption.universe, [6, 10, 10])

# Define rules
rule1 = ctrl.Rule(temperature['cold'] & occupancy['unoccupied'], energy_consumption['low'])
rule2 = ctrl.Rule(temperature['cold'] & occupancy['occupied'] & usage_duration['short'], energy_consumption['low'])
rule3 = ctrl.Rule(temperature['moderate'] & occupancy['occupied'] & usage_duration['medium'], energy_consumption['medium'])
rule4 = ctrl.Rule(temperature['hot'] & occupancy['occupied'] & usage_duration['long'], energy_consumption['high'])
rule5 = ctrl.Rule(temperature['hot'] & occupancy['unoccupied'], energy_consumption['medium'])

# Create control system and simulation
energy_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5])
```

```
energy_consumption_simulation = ctrl.ControlSystemSimulation(energy_ctrl)

# User inputs for simulation
user_temperature = float(input("Enter temperature setting (15-30°C): "))
user_occupancy = int(input("Enter occupancy status (0 for Unoccupied, 1 for Occupied): "))
user_usage_duration = float(input("Enter usage duration in minutes (0-180): "))

# Assign inputs to simulation
energy_consumption_simulation.input['temperature'] = user_temperature
energy_consumption_simulation.input['occupancy'] = user_occupancy
energy_consumption_simulation.input['usage_duration'] = user_usage_duration

# Perform computation
energy_consumption_simulation.compute()

# Output result
print(f"Predicted energy consumption (in kWh): {energy_consumption_simulation.output['energy_consumption']}")

# Visualize the result
energy_consumption.view(sim=energy_consumption_simulation)
plt.show()
```

```
Enter temperature setting (15-30°C): 20
Enter occupancy status (0 for Unoccupied, 1 for Occupied): 1
Enter usage duration in minutes (0-180): 122
Predicted energy consumption (in kWh): 5.000000000000001
```




```
# Temperature Membership Functions
temperature.view()
plt.title('Figure 2: Temperature Membership Functions')
plt.show()

# Occupancy Membership Functions
occupancy.view()
plt.title('Figure 3: Occupancy Membership Functions')
plt.show()

# Usage Duration Membership Functions
usage_duration.view()
plt.title('Figure 4: Usage Duration Membership Functions')
plt.show()
```

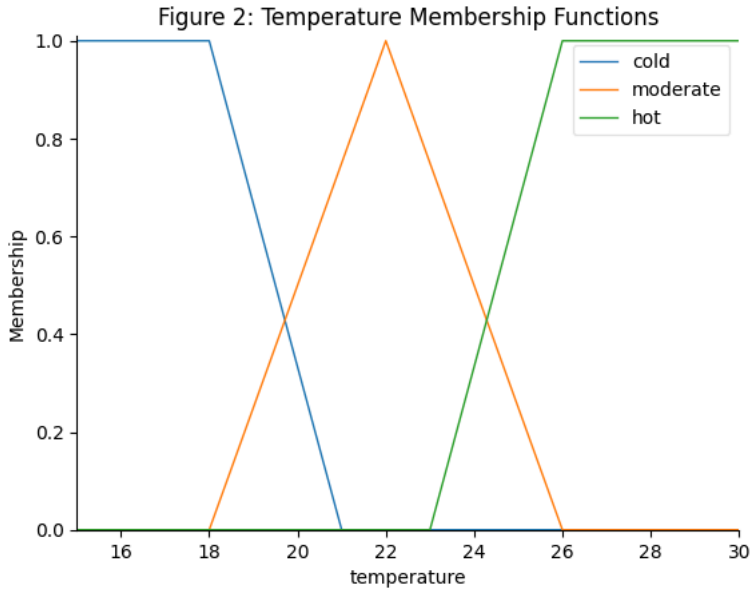


Figure 2: Temperature Membership Functions
cold
moderate
hot
Membership
temperature
1.0
0.8
0.6
0.4
0.2
0.0
16
18
20
22
24
26
28
30

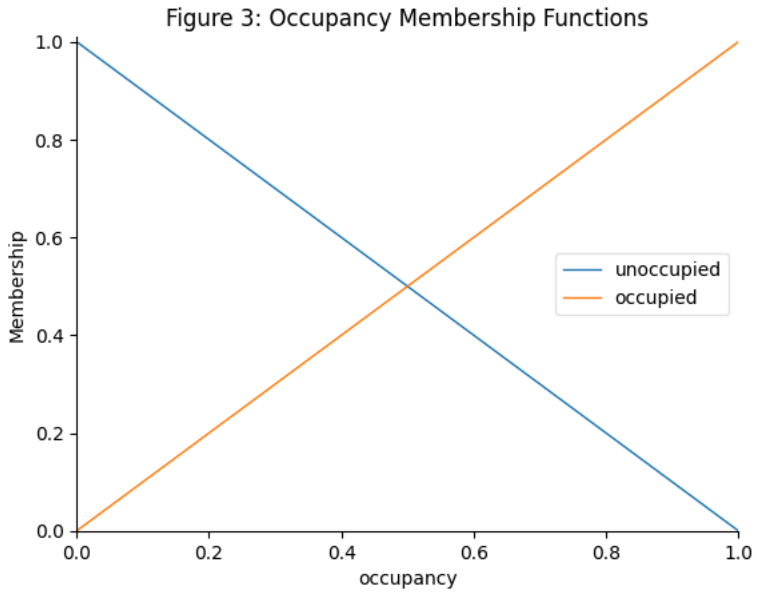


Figure 3: Occupancy Membership Functions
unoccupied
occupied
Membership
occupancy
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

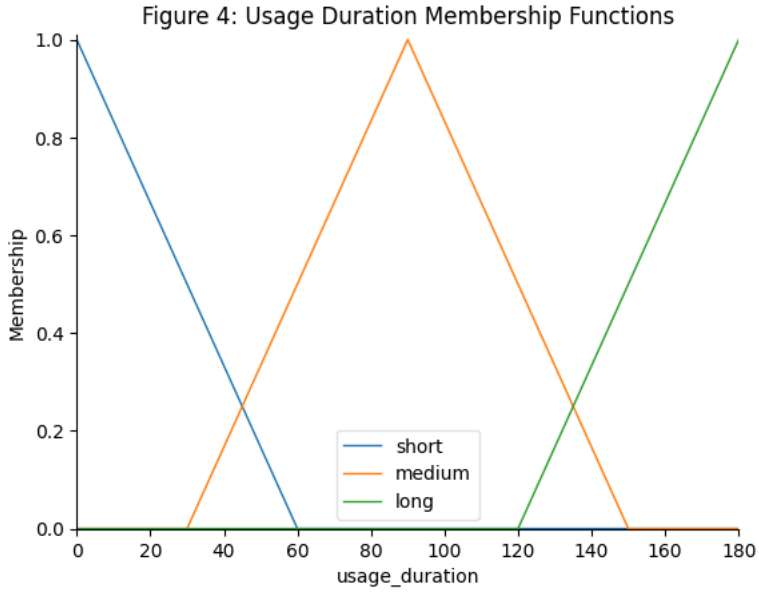


Figure 4: Usage Duration Membership Functions
short
medium
long
Membership
usage_duration
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
120
140
160
180