Computação Inteligente para a Internet das Coisas

Mestrado Engenharia Electrónica

Projeto 2 – Fuzzy Systems

Autores:

João Bernardo Silva (97004) joao.garces.silva@tecnico.ulisboa.pt
Victor Macedo (1105095) victor.macedo@tecnico.ulisboa.pt

Grupo 4

2022/2023 – 2º Semestre, P4

Conteúdo

1 Introdução

O crescente avanço e a crescente capacidade computacional dos dispositivos IoT resultaram em volumes sem precedentes de dados. A quantidade de dados gerados por dispositivos no nosso mundo é maior do que nunca. No entanto, grande parte desses dados do IoT não é explorada ou utilizada, uma vez que o envio de todos esses dados gerados pelos dispositivos para um centro de dados centralizado ou para a nuvem causa problemas de largura de banda e latência.

O objetivo deste projeto é através de um sistema Fuzzy, criar um sistema inteligente de tarefas de computação Edge que não comprometa diversos fatores, como latência, congestão e largura de banda.

2 Metodologia

2.1 Parâmetros de entrada

O problema apresentado possui 6 possíveis entradas para o sistema fuzzy [1], porém é importante selecionar quais são realmente importantes, pois o numero de entrada aumenta a quantidade de regras de forma exponencial. Idealmente a escolha dos parâmetros deve ser feita em conjunto com um especialista do assunto, na falta de um deve-se utilizar o bom senso para o mesmo. Os parâmetros possíveis são:

- Memory usage (%)
- Processor load (%) – Note: Processor load is not the same as CLP
- Input network throughput
- Output network throughput
- Available output bandwidth
- Latency (mS)

Após analisar as opções acimas decidiu-se não utilizar os parâmetros Input network throughput e Available output bandwidth.

2.2 Rede Fuzzy

Como explicado na secção 2.1 a rede possui 4 parâmetros de entradas, considerando 3 termos linguísticos em cada seriam necessários $5^4 = 625$ regras para o sistema. Com objetivo de evitar a o crescimento exponencial o sistema deve ser dividido em 2 grupos menores, o primeiro chamado *load* que deve receber Memory usage e Processor load e *Network* que recebe Output network throughput e Latency.

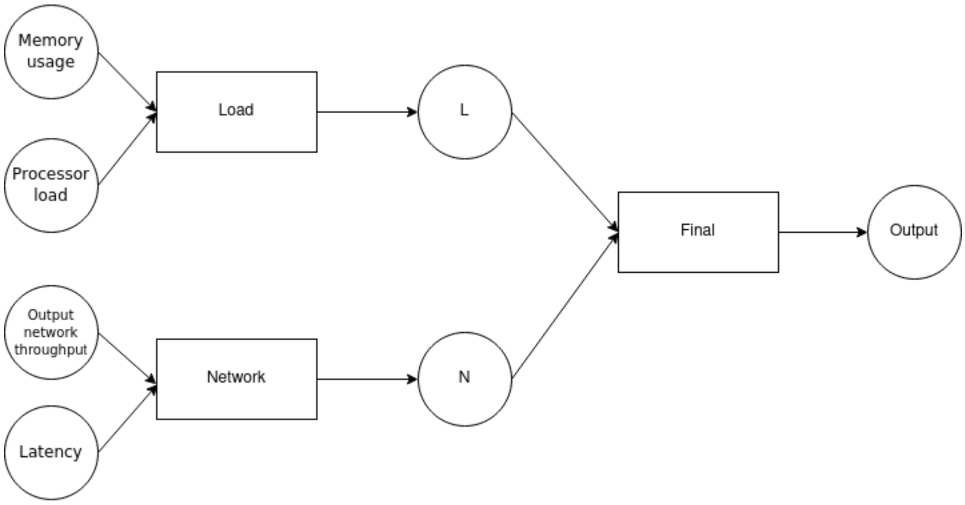

Figura 1: Diagrama da rede

Dessa forma são necessário apenas $5^2 + 5^2 + 5^2 = 75$ regras, uma redução superior a um oitavo no numero de regras.

2.3 Regras

Para as construções das regras do sistema foram utilizados 5 termos linguísticos em cada entrada resultando na seguinte seguem a tabelas 1, 2, 3

Sistema Load		Process				
		Very low	Low	Normal	High	Very High
Memory	Very low	Very High	Very High	High	High	High
	Low	Very High	Very High	High	High	Low
	Normal	High	High	Normal	Low	Low
	High	High	High	Low	Low	Very low
	Very High	High	Low	Low	Very low	Very low

Tabela 1: Regras do primeiro sistema Fuzzy

Sistema Network		Output network throughput				
		Very low	Low	Normal	High	Very High
Latency	Very low	Low	Low	Low	Very low	Very low
	Low	High	Low	Low	Low	Very low
	Normal	High	High	Normal	Low	Low
	High	Very High	High	High	Low	Low
	Very High	Very High	Very High	High	High	Low

Tabela 2: Regras do segundo sistema Fuzzy

Sistema Final		Network				
		Very low	Low	Normal	High	Very High
Load	Very low	Normal	High	Very High	Very High	Very High
	Low	Normal	Normal	High	Very High	Very High
	Normal	Very low	Normal	Normal	High	Very High
	High	Very low	Low	Low	Low	High
	Very High	Very low	Very low	Very low	Very low	Very low

Tabela 3: Regras do terceiro sistema Fuzzy

Os termos linguisticos foram definidos usando funções triangulares, como apresentado nas figuras 2, 3 e 4

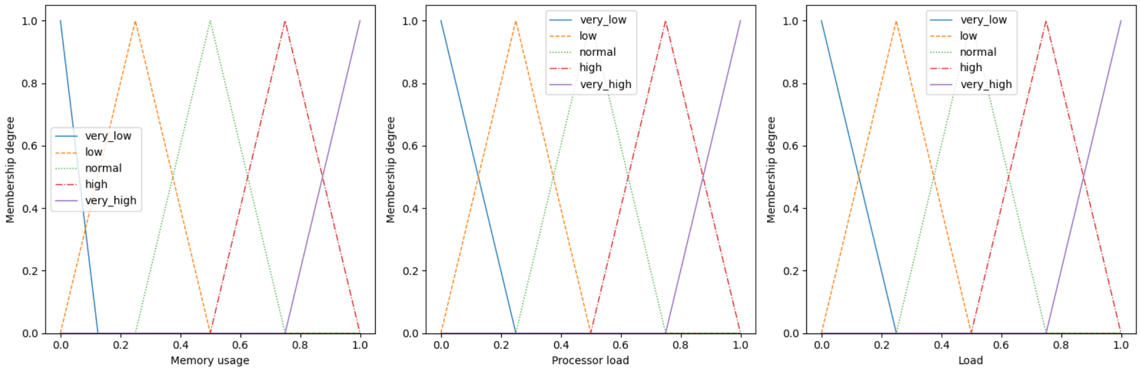

Figura 2: Regras termos linguísticos load

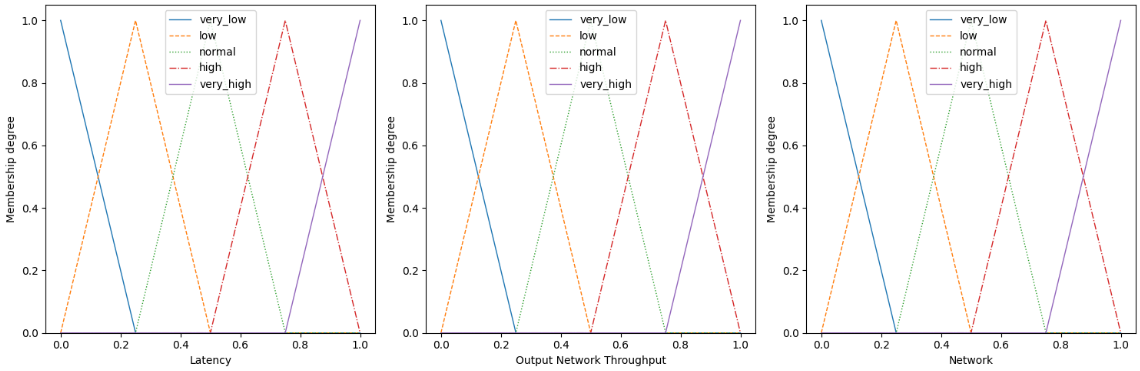

Figura 3: Regras termos linguísticos network

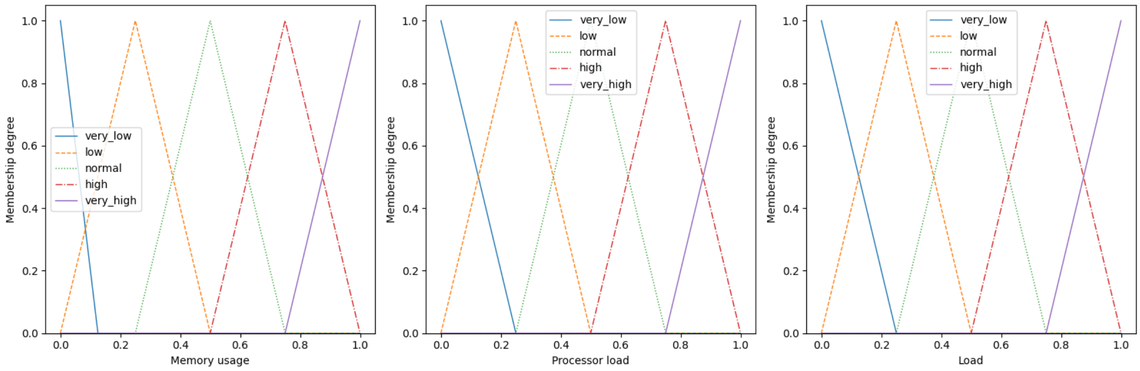

Figura 4: Regras termos linguísticos final

Por fim os sistemas descritos acima resultaram nos gráfico 3D das figuras 5, 6 e 7.

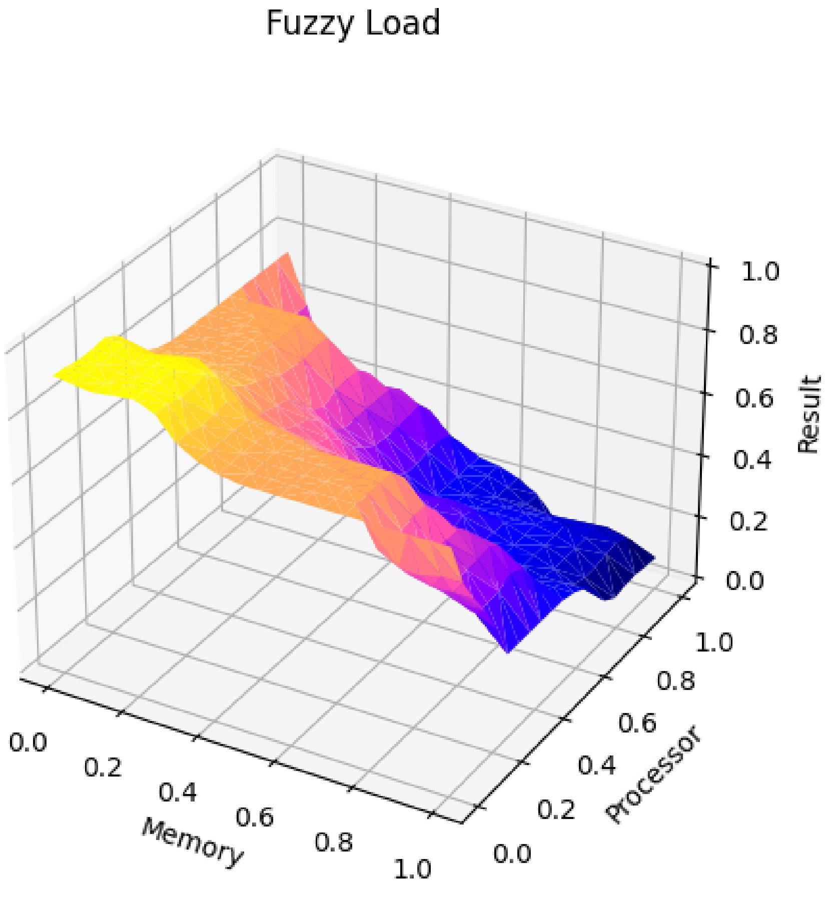

Figura 5: Two Input / Single Output Load

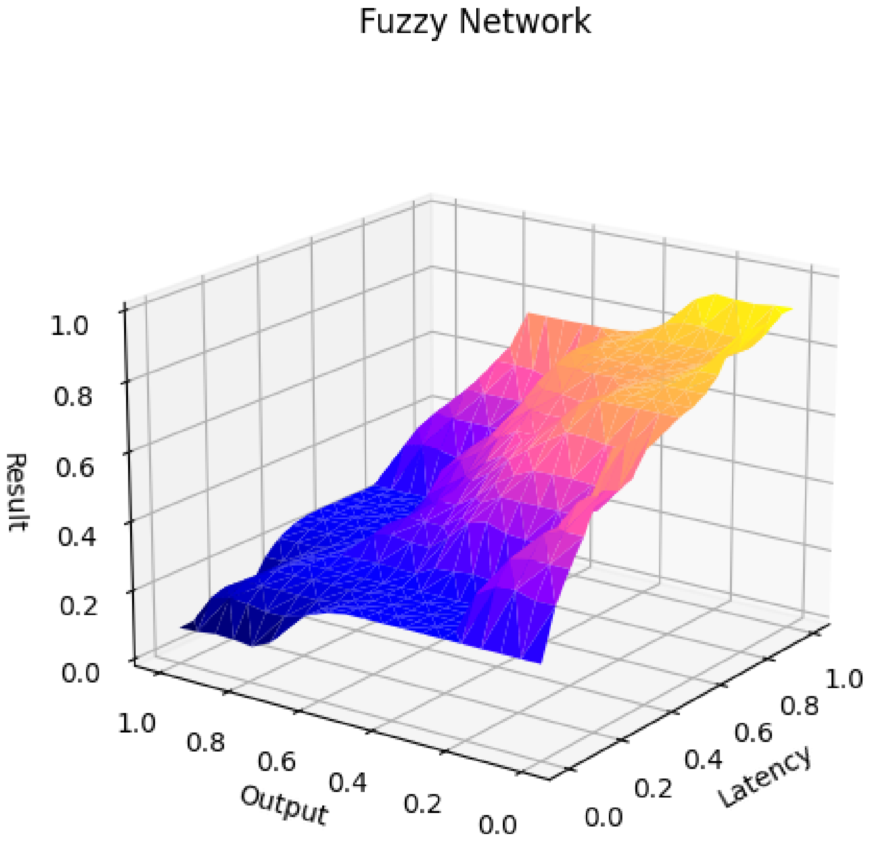

Figura 6: Two Input / Single Output Network

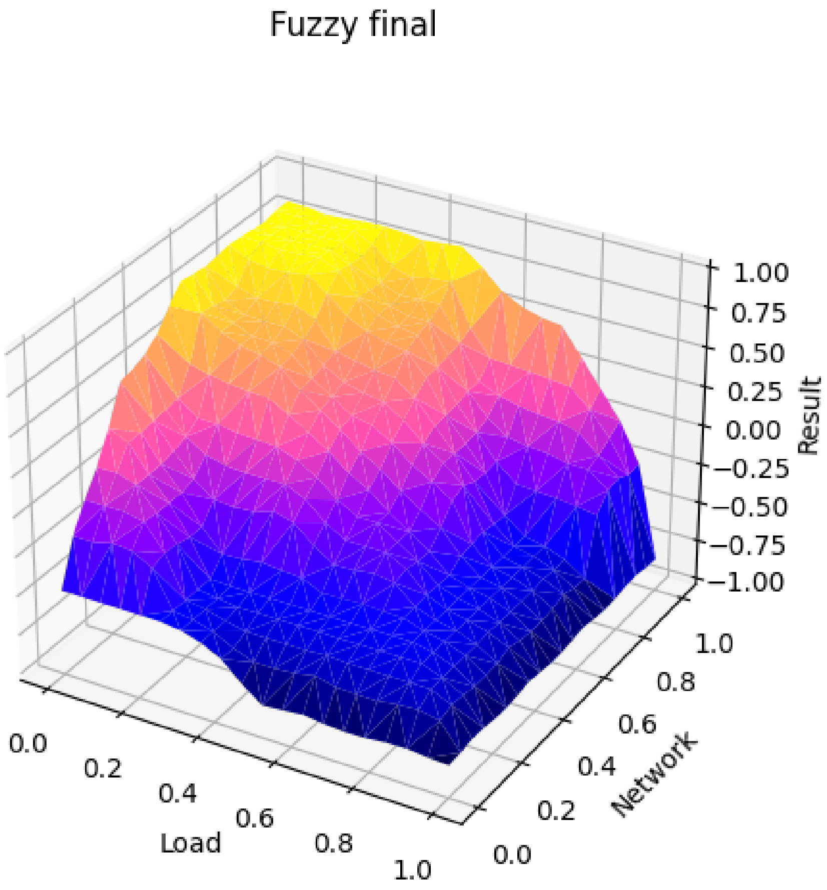

Figura 7: Two Input / Single Output finalç

2.4 Implementação

Para implementar o sistema descrito acima deve ser utilizado a linguagem de programação Python ([3]) utilizando o package simpful ([2]), o qual permite definir os termos linguísticos, regras e o sistema fuzzy de forma simples.

3 Resultados

Para avaliar o desempenho do sistema foi utilizado um dataset exemplo, os resultados obtido podem ser observado na tabela 4

Esperados	Obtidos
0.73	0.028
-0.82	-0.250
0.12	-0.050
-0.85	0.116
0.5	0.000
0.8	0.599
-0.31	-0.413
-0.65	-0.413
0.85	-0.254
0.85	-0.422

Tabela 4: Resultados utilizando o dataset

Durante a avaliação dos resultados foi necessário ajustar os parâmetros, como regras e termos linguísticos, do sistema diversas vezes para obter melhores respostas, dessas forma fica evidente que o auxilio de um profissional da área é capaz de agilizar e melhorar os resultados do sistema.

4 Conclusão

A construção desta rede Fuzzy apresentou desafios significativos, uma vez que os resultados obtidos variam consideravelmente em relação às expectativas estabelecidas. A implementação de tais redes é um processo complexo que requer experiência para alcançar resultados satisfatórios.

Apesar dos valores numéricos obtidos e esperados não serem tão próximos ao considerar o termo linguístico tem-se um resultado satisfatório para a maioria dos casos.

Referências

[1] Joao Paulo Carvalho. *Slides Computação Inteligente para a Internet das Coisas.* 2023.

[2] Cazzaniga P. Kaymak U. Besozzi D. Nobile M.S. Spolaor S., Fuchs C. Simpful: a user-friendly python library for fuzzy logic. *International Journal of Computational Intelligence Systems.*

[3] Guido Van Rossum and Fred L Drake Jr. *Python reference manual.* Centrum voor Wiskunde en Informatica Amsterdam, 1995.

Apêndices

A Código main

```python
import numpy as np
import pandas as pd
import matplotlib.pylab as plt

from simpful import *
from sklearn.neural_network import MLPRegressor



#df = pd.read_csv("/Users/mac/Documents/GitHub/CI4Iot/Projeto_2/Project2_SampleData.csv")
df = pd.read_csv("Projeto_2/Project2_SampleData.csv")

# Rede Fuzzy, input memory usage and processor load, output Load
# Create a fuzzy system object
FS_L = FuzzySystem()

# Define fuzzy sets and linguistic variables
S_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.125), term="very_low")
S_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
S_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
S_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
S_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
S_LV =  LinguisticVariable([S_1, S_2, S_3,S_4,S_5], concept="Memory usage", universe_of_discourse=[0,1])
FS_L.add_linguistic_variable("Memory",S_LV)


F_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.25), term="very_low")
F_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
F_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
F_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
F_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
F_LV = LinguisticVariable([F_1, F_2,F_3,F_4,F_5], concept="Processor load", universe_of_discourse=[0,1])
FS_L.add_linguistic_variable("Processor", F_LV)

# Define output fuzzy sets and linguistic variable
T_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.25), term="very_low")
T_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
T_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
T_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
T_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
T_LV = LinguisticVariable([T_1, T_2, T_3, T_4, T_5], universe_of_discourse=[0,1])
FS_L.add_linguistic_variable("Load", T_LV)
#FS_L.produce_figure()

# Define fuzzy rules
```

```
R_L1 = "IF (Memory IS very_low) AND (Processor IS very_low) THEN (Load IS very_high)"
R_L2 = "IF (Memory IS very_low) AND (Processor IS low) THEN (Load IS very_high)"
R_L3 = "IF (Memory IS very_low) AND (Processor IS normal) THEN (Load IS high)"
R_L4 = "IF (Memory IS very_low) AND (Processor IS high) THEN (Load IS high)"
R_L5 = "IF (Memory IS very_low) AND (Processor IS very_high) THEN (Load IS high)"
R_L6 = "IF (Memory IS low) AND (Processor IS very_low) THEN (Load IS very_high)"
R_L7 = "IF (Memory IS low) AND (Processor IS low) THEN (Load IS very_high)"
R_L8 = "IF (Memory IS low) AND (Processor IS normal) THEN (Load IS high)"
R_L9 = "IF (Memory IS low) AND (Processor IS high) THEN (Load IS high)"
R_L10 = "IF (Memory IS low) AND (Processor IS very_high) THEN (Load IS low)"
R_L11 = "IF (Memory IS normal) AND (Processor IS very_low) THEN (Load IS high)"
R_L12 = "IF (Memory IS normal) AND (Processor IS low) THEN (Load IS high)"
R_L13 = "IF (Memory IS normal) AND (Processor IS normal) THEN (Load IS normal)"
R_L14 = "IF (Memory IS normal) AND (Processor IS high) THEN (Load IS low)"
R_L15 = "IF (Memory IS normal) AND (Processor IS very_high) THEN (Load IS low)"
R_L16 = "IF (Memory IS high) AND (Processor IS very_low) THEN (Load IS high)"
R_L17 = "IF (Memory IS high) AND (Processor IS low) THEN (Load IS high)"
R_L18 = "IF (Memory IS high) AND (Processor IS normal) THEN (Load IS low)"
R_L19 = "IF (Memory IS high) AND (Processor IS high) THEN (Load IS low)"
R_L20 = "IF (Memory IS high) AND (Processor IS very_high) THEN (Load IS very_low)"
R_L21 = "IF (Memory IS very_high) AND (Processor IS very_low) THEN (Load IS high)"
R_L22 = "IF (Memory IS very_high) AND (Processor IS low) THEN (Load IS low)"
R_L23 = "IF (Memory IS very_high) AND (Processor IS normal) THEN (Load IS low)"
R_L24 = "IF (Memory IS very_high) AND (Processor IS high) THEN (Load IS very_low)"
R_L25 = "IF (Memory IS very_high) AND (Processor IS very_high) THEN (Load IS  very_low)"
FS_L.add_rules([ R_L1, R_L2, R_L3, R_L4, R_L5, R_L6, R_L7, R_L8, R_L9, R_L10, R_L11, R_L12, R_L13, R_L14, R_L15, R_L16, R_L17 ,R_L18, R_L19, R_L20, R_L21, R_L22, R_L23, R_L24, R_L25 ])

############ FIM Da REDE FS_L    ####################################


FS_N = FuzzySystem()
## REDE NETWORK
L_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.25), term="very_low")
L_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
L_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
L_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
L_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
L_LV = LinguisticVariable([L_1, L_2, L_3,L_4,L_5], concept="Latency", universe_of_discourse=[0, 1])
```

```python
FS_N.add_linguistic_variable("Latency", L_LV)

O_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.25), term="very_low")
O_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
O_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
O_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
O_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
O_LV = LinguisticVariable([O_1, O_2, O_3,O_4,O_5], concept="Output Network Throughput", universe_of_discourse=[0, 1])
FS_N.add_linguistic_variable("Output", O_LV)

# Define output fuzzy sets and linguistic variable
N_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.25), term="very_low")
N_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
N_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
N_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
N_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
N_LV = LinguisticVariable([N_1, N_2, N_3, N_4, N_5], universe_of_discourse=[0, 1])
FS_N.add_linguistic_variable("Network", N_LV)
#FS_N.produce_figure()
# Define fuzzy rules

R_N1 = "IF (Output IS very_low) AND (Latency IS very_low) THEN (Network IS low)"
R_N2 = "IF (Output IS very_low) AND (Latency IS low) THEN (Network IS high)"
R_N3 = "IF (Output IS very_low) AND (Latency IS normal) THEN (Network IS high)"
R_N4 = "IF (Output IS very_low) AND (Latency IS high) THEN (Network IS very_high)"
R_N5 = "IF (Output IS very_low) AND (Latency IS very_high) THEN (Network IS very_high)"
R_N6 = "IF (Output IS low) AND (Latency IS very_low) THEN (Network IS low"
R_N7 = "IF (Output IS low) AND (Latency IS low) THEN (Network IS low)"
R_N8 = "IF (Output IS low) AND (Latency IS normal) THEN (Network IS high)"
R_N9 = "IF (Output IS low) AND (Latency IS high) THEN (Network IS high)"
R_N10 = "IF (Output IS low) AND (Latency IS very_high) THEN (Network IS very_high)"
R_N11 = "IF (Output IS normal) AND (Latency IS very_low) THEN (Network IS low)"
R_N12 = "IF (Output IS normal) AND (Latency IS low) THEN (Network IS low)"
R_N13 = "IF (Output IS normal) AND (Latency IS normal) THEN (Network IS normal)"
R_N14 = "IF (Output IS normal) AND (Latency IS high) THEN (Network IS high)"
R_N15 = "IF (Output IS normal) AND (Latency IS very_high) THEN (Network IS high)"
R_N16 = "IF (Output IS high) AND (Latency IS low) THEN (Network IS low)"
R_N17 = "IF (Output IS high) AND (Latency IS very_low) THEN (Network IS very_low)"
R_N18 = "IF (Output IS high) AND (Latency IS normal) THEN (Network IS low)"
R_N19 = "IF (Output IS high) AND (Latency IS high) THEN (Network IS low)"
R_N20 = "IF (Output IS high) AND (Latency IS very_high) THEN (Network IS high)"
R_N21 = "IF (Output IS very_high) AND (Latency IS low) THEN (Network IS very_low)"
```

```python
R_N22 = "IF (Output IS very_high) AND (Latency IS very_low) THEN (Network IS very_low)"
R_N23 = "IF (Output IS very_high) AND (Latency IS normal) THEN (Network IS low)"
R_N24 = "IF (Output IS very_high) AND (Latency IS high) THEN (Network IS low)"
R_N25 = "IF (Output IS very_high) AND (Latency IS very_high) THEN (Network IS low)"
FS_N.add_rules([ R_N1, R_N2, R_N3, R_N4, R_N5, R_N6, R_N7, R_N8, R_N9, R_N10, R_N11, R_N12, R_N13, R_N14, R_N15, R_N16, R_N17 ,R_N18, R_N19, R_N20, R_N21, R_N22, R_N23, R_N24, R_N25 ])
########FIM REDE Network ###########################################

#Rede Final
FS_F = FuzzySystem()

G_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.25), term="very_low")
G_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
G_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
G_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
G_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
G_LV = LinguisticVariable([G_1, G_2, G_3,G_4,G_5], concept="Network", universe_of_discourse=[0, 1])
FS_F.add_linguistic_variable("Network", G_LV)

H_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.25), term="very_low")
H_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
H_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
H_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
H_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
H_LV = LinguisticVariable([H_1, H_2, H_3, H_4,H_5], concept="Load", universe_of_discourse=[0, 1])
FS_F.add_linguistic_variable("Load", H_LV)

# Define output fuzzy sets and linguistic variable
J_1 = FuzzySet(function=Triangular_MF(a=-1, b=-1, c=-0.5), term="very_low")
J_2 = FuzzySet(function=Triangular_MF(a=-1, b=-0.5, c=0.0), term="low")
J_3 = FuzzySet(function=Triangular_MF(a=-0.5, b=0.0, c=0.5), term="normal")
J_4 = FuzzySet(function=Triangular_MF(a=0.0, b=0.5, c=1), term="high")
J_5 = FuzzySet(function=Triangular_MF(a=0.5, b=1, c=1), term="very_high")
J_LV = LinguisticVariable([J_1, J_2, J_3,J_4,J_5], universe_of_discourse=[-1, 1])
FS_F.add_linguistic_variable("Result", J_LV)
#FS_L.produce_figure()
# Define fuzzy rules

R_F1 = "IF (Network IS very_low) AND (Load IS very_low) THEN (Result IS low)"
R_F2 = "IF (Network IS very_low) AND (Load IS low) THEN (Result IS low)"
R_F3 = "IF (Network IS very_low) AND (Load IS normal) THEN (Result IS very_low)"
R_F4 = "IF (Network IS very_low) AND (Load IS high) THEN (Result IS very_low)"
R_F5 = "IF (Network IS very_low) AND (Load IS very_high) THEN (Result IS very_low)"
```

```
R_F6 = "IF (Network IS low) AND (Load IS very_low) THEN (Result IS high"
R_F7 = "IF (Network IS low) AND (Load IS low) THEN (Result IS low)"
R_F8 = "IF (Network IS low) AND (Load IS normal) THEN (Result IS low)"
R_F9 = "IF (Network IS low) AND (Load IS high) THEN (Result IS low)"
R_F10 = "IF (Network IS low) AND (Load IS very_high) THEN (Result IS very_low)"
R_F11 = "IF (Network IS normal) AND (Load IS very_low) THEN (Result IS very_high)"
R_F12 = "IF (Network IS normal) AND (Load IS low) THEN (Result IS high)"
R_F13 = "IF (Network IS normal) AND (Load IS normal) THEN (Result IS normal)"
R_F14 = "IF (Network IS normal) AND (Load IS high) THEN (Result IS low)"
R_F15 = "IF (Network IS normal) AND (Load IS very_high) THEN (Result IS very_low)"
R_F16 = "IF (Network IS high) AND (Load IS low) THEN (Result IS very_high)"
R_F17 = "IF (Network IS high) AND (Load IS very_low) THEN (Result IS very_high)"
R_F18 = "IF (Network IS high) AND (Load IS normal) THEN (Result IS high)"
R_F19 = "IF (Network IS high) AND (Load IS high) THEN (Result IS low)"
R_F20 = "IF (Network IS high) AND (Load IS very_high) THEN (Result IS very_low)"
R_F21 = "IF (Network IS very_high) AND (Load IS low) THEN (Result IS very_high)"
R_F22 = "IF (Network IS very_high) AND (Load IS very_low) THEN (Result IS very_high)"
R_F23 = "IF (Network IS very_high) AND (Load IS normal) THEN (Result IS very_high)"
R_F24 = "IF (Network IS very_high) AND (Load IS high) THEN (Result IS high)"
R_F25 = "IF (Network IS very_high) AND (Load IS very_high) THEN (Result IS very_low)"
FS_F.add_rules([ R_F1, R_F2, R_F3, R_F4, R_F5, R_F6, R_F7, R_F8, R_F9, R_F10, R_F11, R_F12, R_F13, R_F14, R_F15, R_F16, R_F17 ,R_F18, R_F19, R_F20, R_F21, R_F22, R_F23, R_F24, R_F25 ])
# Set antecedents values

for n_teste in range(10):
    FS_N.set_variable("Latency", df['Latency'][n_teste])
    FS_N.set_variable("Output", df['OutBandwidth'][n_teste])

    FS_L.set_variable("Memory", df['ProcessorLoad'][n_teste])
    FS_L.set_variable("Processor", df['MemoryUsage'][n_teste])

    FS_F.set_variable("Load", float(FS_L.Mamdani_inference(["Load"])["Load"]))
    FS_F.set_variable("Network", float(FS_N.Mamdani_inference(["Network"])["Network"]))

    print("Teste n: ", n_teste )
    print(FS_F.Mamdani_inference(["Result"]))
    print("Resultado esperado: ", df['CLPVariation'][n_teste])

from mpl_toolkits.mplot3d import Axes3D
    # Plotting surface
xs = []
ys = []
```

```python
zs = []
DIVs = 20
for x in np.linspace(0,1,DIVs):
    for y in np.linspace(0,1,DIVs):
        FS_N.set_variable("Latency", x)
        FS_N.set_variable("Output", y)
        tip = FS_N.Mamdani_inference()["Network"]
        xs.append(x)
        ys.append(y)
        zs.append(tip)
xs = np.array(xs)
ys = np.array(ys)
zs = np.array(zs)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

xx, yy = plt.meshgrid(xs,ys)

ax.plot_trisurf(xs,ys,zs, vmin=0, vmax=1, cmap='gnuplot2')
ax.set_xlabel("Latency")
ax.set_ylabel("Output")
ax.set_zlabel("Result")
ax.set_title("Fuzzy Network", pad=20)
ax.set_zlim(0, 1)
plt.tight_layout()

    # Plotting surface
xs = []
ys = []
zs = []
for x in np.linspace(0,1,DIVs):
    for y in np.linspace(0,1,DIVs):
        FS_L.set_variable("Memory", x)
        FS_L.set_variable("Processor", y)
        tip = FS_L.Mamdani_inference()["Load"]
        xs.append(x)
        ys.append(y)
        zs.append(tip)
xs = np.array(xs)
ys = np.array(ys)
zs = np.array(zs)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

xx, yy = plt.meshgrid(xs,ys)

ax.plot_trisurf(xs,ys,zs, vmin=0, vmax=1, cmap='gnuplot2')
ax.set_xlabel("Memory")
ax.set_ylabel("Processor")
ax.set_zlabel("Result")
ax.set_title("Fuzzy Load", pad=20)
ax.set_zlim(0, 1)
plt.tight_layout()
```

```

    # Plotting surface
xs = []
ys = []
zs = []
for x in np.linspace(0,1,DIVs):
    for y in np.linspace(0,1,DIVs):
        FS_F.set_variable("Load", x)
        FS_F.set_variable("Network", y)
        tip = FS_F.Mamdani_inference()["Result"]
        xs.append(x)
        ys.append(y)
        zs.append(tip)
xs = np.array(xs)
ys = np.array(ys)
zs = np.array(zs)

from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection=’3d’)

xx, yy = plt.meshgrid(xs,ys)

ax.plot_trisurf(xs,ys,zs, vmin=-1, vmax=1, cmap=’gnuplot2’)
ax.set_xlabel("Load")
ax.set_ylabel("Network")
ax.set_zlabel("Result")
ax.set_title("Fuzzy final", pad=20)
ax.set_zlim(-1, 1)
plt.tight_layout()


plt.show()
```

B Código TestMe

```
    import pandas as pd
from simpful import *
from sklearn.neural_network import MLPRegressor


df = pd.read_csv("Projeto_2/Lab10-Proj2_TestS.csv")

# Rede Fuzzy, input memory usage and processor load, output Load
# Create a fuzzy system object
FS_L = FuzzySystem()

# Define fuzzy sets and linguistic variables
S_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.125), term="very_low")
S_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
S_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
S_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
S_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
```

```
S_LV =  LinguisticVariable([S_1, S_2, S_3,S_4,S_5], concept="Memory usage", universe_of_discourse=[0,1])
FS_L.add_linguistic_variable("Memory",S_LV)

F_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.25), term="very_low")
F_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
F_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
F_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
F_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
F_LV = LinguisticVariable([F_1, F_2,F_3,F_4,F_5], concept="Processor load", universe_of_discourse=[0,1])
FS_L.add_linguistic_variable("Processor", F_LV)

# Define output fuzzy sets and linguistic variable
T_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.25), term="very_low")
T_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
T_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
T_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
T_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
T_LV = LinguisticVariable([T_1, T_2, T_3, T_4, T_5], universe_of_discourse=[0,1])
FS_L.add_linguistic_variable("Load", T_LV)

# Define fuzzy rules
R_L1 = "IF (Memory IS very_low) AND (Processor IS very_low) THEN (Load IS very_high)"
R_L2 = "IF (Memory IS very_low) AND (Processor IS low) THEN (Load IS very_high)"
R_L3 = "IF (Memory IS very_low) AND (Processor IS normal) THEN (Load IS high)"
R_L4 = "IF (Memory IS very_low) AND (Processor IS high) THEN (Load IS high)"
R_L5 = "IF (Memory IS very_low) AND (Processor IS very_high) THEN (Load IS high)"
R_L6 = "IF (Memory IS low) AND (Processor IS very_low) THEN (Load IS very_high)"
R_L7 = "IF (Memory IS low) AND (Processor IS low) THEN (Load IS very_high)"
R_L8 = "IF (Memory IS low) AND (Processor IS normal) THEN (Load IS high)"
R_L9 = "IF (Memory IS low) AND (Processor IS high) THEN (Load IS high)"
R_L10 = "IF (Memory IS low) AND (Processor IS very_high) THEN (Load IS low)"
R_L11 = "IF (Memory IS normal) AND (Processor IS very_low) THEN (Load IS high)"
R_L12 = "IF (Memory IS normal) AND (Processor IS low) THEN (Load IS high)"
R_L13 = "IF (Memory IS normal) AND (Processor IS normal) THEN (Load IS normal)"
R_L14 = "IF (Memory IS normal) AND (Processor IS high) THEN (Load IS low)"
R_L15 = "IF (Memory IS normal) AND (Processor IS very_high) THEN (Load IS low)"
R_L16 = "IF (Memory IS high) AND (Processor IS very_low) THEN (Load IS high)"
R_L17 = "IF (Memory IS high) AND (Processor IS low) THEN (Load IS high)"
R_L18 = "IF (Memory IS high) AND (Processor IS normal) THEN (Load IS low)"
R_L19 = "IF (Memory IS high) AND (Processor IS high) THEN (Load IS low)"
R_L20 = "IF (Memory IS high) AND (Processor IS very_high) THEN (Load IS very_low)"
R_L21 = "IF (Memory IS very_high) AND (Processor IS very_low) THEN (Load IS
high)"
R_L22 = "IF (Memory IS very_high) AND (Processor IS low) THEN (Load IS low)"
R_L23 = "IF (Memory IS very_high) AND (Processor IS normal) THEN (Load IS low)"
R_L24 = "IF (Memory IS very_high) AND (Processor IS high) THEN (Load IS very_low)"
R_L25 = "IF (Memory IS very_high) AND (Processor IS very_high) THEN (Load IS  very_low)"
FS_L.add_rules([ R_L1, R_L2, R_L3, R_L4, R_L5, R_L6, R_L7, R_L8, R_L9, R_L10, R_L11, R_L12, R_L13, R_L14, R_L15, R_L16, R_L17 ,R_L18, R_L19, R_L20, R_L21, R_L22, R_L23, R_L24, R_L25 ])

############ FIM Da REDE FS_L    ####################################


FS_N = FuzzySystem()
## REDE NETWORK
L_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.25), term="very_low")
L_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
L_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
L_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
L_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
L_LV = LinguisticVariable([L_1, L_2, L_3,L_4,L_5], concept="Latency", universe_of_discourse=[0, 1])
FS_N.add_linguistic_variable("Latency", L_LV)

O_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.25), term="very_low")
O_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
O_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
O_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
O_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
O_LV = LinguisticVariable([O_1, O_2, O_3,O_4,O_5], concept="Output Network Throughput", universe_of_discourse=[0, 1])
FS_N.add_linguistic_variable("Output", O_LV)

# Define output fuzzy sets and linguistic variable
N_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.25), term="very_low")
N_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
N_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
N_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
N_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
N_LV = LinguisticVariable([N_1, N_2, N_3, N_4, N_5], universe_of_discourse=[0, 1])
FS_N.add_linguistic_variable("Network", N_LV)
# Define fuzzy rules

R_N1 = "IF (Output IS very_low) AND (Latency IS very_low) THEN (Network IS low)"
R_N2 = "IF (Output IS very_low) AND (Latency IS low) THEN (Network IS high)"
R_N3 = "IF (Output IS very_low) AND (Latency IS normal) THEN (Network IS high)"
R_N4 = "IF (Output IS very_low) AND (Latency IS high) THEN (Network IS very_high)"
R_N5 = "IF (Output IS very_low) AND (Latency IS very_high) THEN (Network IS very_high)"
```

```
R_N6 = "IF (Output IS low) AND (Latency IS very_low) THEN (Network IS low"
R_N7 = "IF (Output IS low) AND (Latency IS low) THEN (Network IS low)"
R_N8 = "IF (Output IS low) AND (Latency IS normal) THEN (Network IS high)"
R_N9 = "IF (Output IS low) AND (Latency IS high) THEN (Network IS high)"
R_N10 = "IF (Output IS low) AND (Latency IS very_high) THEN (Network IS very_high)"
R_N11 = "IF (Output IS normal) AND (Latency IS very_low) THEN (Network IS low)"
R_N12 = "IF (Output IS normal) AND (Latency IS low) THEN (Network IS low)"
R_N13 = "IF (Output IS normal) AND (Latency IS normal) THEN (Network IS normal)"
R_N14 = "IF (Output IS normal) AND (Latency IS high) THEN (Network IS high)"
R_N15 = "IF (Output IS normal) AND (Latency IS very_high) THEN (Network IS high)"
R_N16 = "IF (Output IS high) AND (Latency IS low) THEN (Network IS low)"
R_N17 = "IF (Output IS high) AND (Latency IS very_low) THEN (Network IS very_low)"
R_N18 = "IF (Output IS high) AND (Latency IS normal) THEN (Network IS low)"
R_N19 = "IF (Output IS high) AND (Latency IS high) THEN (Network IS low)"
R_N20 = "IF (Output IS high) AND (Latency IS very_high) THEN (Network IS high)"
R_N21 = "IF (Output IS very_high) AND (Latency IS low) THEN (Network IS very_low)"
R_N22 = "IF (Output IS very_high) AND (Latency IS very_low) THEN (Network IS very_low)"
R_N23 = "IF (Output IS very_high) AND (Latency IS normal) THEN (Network IS low)"
R_N24 = "IF (Output IS very_high) AND (Latency IS high) THEN (Network IS low)"
R_N25 = "IF (Output IS very_high) AND (Latency IS very_high) THEN (Network IS low)"
FS_N.add_rules([ R_N1, R_N2, R_N3, R_N4, R_N5, R_N6, R_N7, R_N8, R_N9, R_N10, R_N11, R_N12, R_N13, R_N14, R_N15, R_N16, R_N17 ,R_N18, R_N19, R_N20, R_N21, R_N22, R_N23, R_N24, R_N25 ])
#######FIM REDE Network ###########################################

#Rede Final
FS_F = FuzzySystem()

G_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.25), term="very_low")
G_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
G_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
G_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
G_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
G_LV = LinguisticVariable([G_1, G_2, G_3,G_4,G_5], concept="Network", universe_of_discourse=[0, 1])
FS_F.add_linguistic_variable("Network", G_LV)

H_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=0.25), term="very_low")
H_2 = FuzzySet(function=Triangular_MF(a=0.0, b=0.25, c=0.5), term="low")
H_3 = FuzzySet(function=Triangular_MF(a=0.25, b=0.5, c=0.75), term="normal")
H_4 = FuzzySet(function=Triangular_MF(a=0.5, b=0.75, c=1), term="high")
H_5 = FuzzySet(function=Triangular_MF(a=0.75, b=1, c=1), term="very_high")
H_LV = LinguisticVariable([H_1, H_2, H_3, H_4,H_5], concept="Load", universe_of_discourse=[0, 1])
```

```
FS_F.add_linguistic_variable("Load", H_LV)

# Define output fuzzy sets and linguistic variable
J_1 = FuzzySet(function=Triangular_MF(a=-1, b=-1, c=-0.5), term="very_low")
J_2 = FuzzySet(function=Triangular_MF(a=-1, b=-0.5, c=0.0), term="low")
J_3 = FuzzySet(function=Triangular_MF(a=-0.5, b=0.0, c=0.5), term="normal")
J_4 = FuzzySet(function=Triangular_MF(a=0.0, b=0.5, c=1), term="high")
J_5 = FuzzySet(function=Triangular_MF(a=0.5, b=1, c=1), term="very_high")
J_LV = LinguisticVariable([J_1, J_2, J_3,J_4,J_5], universe_of_discourse=[-1, 1])
FS_F.add_linguistic_variable("Result", J_LV)
# Define fuzzy rules

R_F1 = "IF (Network IS very_low) AND (Load IS very_low) THEN (Result IS low)"
R_F2 = "IF (Network IS very_low) AND (Load IS low) THEN (Result IS low)"
R_F3 = "IF (Network IS very_low) AND (Load IS normal) THEN (Result IS very_low)"
R_F4 = "IF (Network IS very_low) AND (Load IS high) THEN (Result IS very_low)"
R_F5 = "IF (Network IS very_low) AND (Load IS very_high) THEN (Result IS very_low)"
R_F6 = "IF (Network IS low) AND (Load IS very_low) THEN (Result IS high"
R_F7 = "IF (Network IS low) AND (Load IS low) THEN (Result IS low)"
R_F8 = "IF (Network IS low) AND (Load IS normal) THEN (Result IS low)"
R_F9 = "IF (Network IS low) AND (Load IS high) THEN (Result IS low)"
R_F10 = "IF (Network IS low) AND (Load IS very_high) THEN (Result IS very_low)"
R_F11 = "IF (Network IS normal) AND (Load IS very_low) THEN (Result IS very_high)"
R_F12 = "IF (Network IS normal) AND (Load IS low) THEN (Result IS high)"
R_F13 = "IF (Network IS normal) AND (Load IS normal) THEN (Result IS normal)"
R_F14 = "IF (Network IS normal) AND (Load IS high) THEN (Result IS low)"
R_F15 = "IF (Network IS normal) AND (Load IS very_high) THEN (Result IS very_low)"
R_F16 = "IF (Network IS high) AND (Load IS low) THEN (Result IS very_high)"
R_F17 = "IF (Network IS high) AND (Load IS very_low) THEN (Result IS very_high)"
R_F18 = "IF (Network IS high) AND (Load IS normal) THEN (Result IS high)"
R_F19 = "IF (Network IS high) AND (Load IS high) THEN (Result IS low)"
R_F20 = "IF (Network IS high) AND (Load IS very_high) THEN (Result IS very_low)"
R_F21 = "IF (Network IS very_high) AND (Load IS low) THEN (Result IS very_high)"
R_F22 = "IF (Network IS very_high) AND (Load IS very_low) THEN (Result IS very_high)"
R_F23 = "IF (Network IS very_high) AND (Load IS normal) THEN (Result IS very_high)"
R_F24 = "IF (Network IS very_high) AND (Load IS high) THEN (Result IS high)"
R_F25 = "IF (Network IS very_high) AND (Load IS very_high) THEN (Result IS very_low)"
FS_F.add_rules([ R_F1, R_F2, R_F3, R_F4, R_F5, R_F6, R_F7, R_F8, R_F9, R_F10, R_F11, R_F12, R_F13, R_F14, R_F15, R_F16, R_F17 ,R_F18, R_F19, R_F20, R_F21, R_F22, R_F23, R_F24, R_F25 ])
```

```python
# Set antecedents values

for n_teste in range(len(df)):
    FS_N.set_variable("Latency", df['Latency'][n_teste])
    FS_N.set_variable("Output", df['OutNetThroughput'][n_teste])

    FS_L.set_variable("Memory", df['ProcessorLoad'][n_teste])
    FS_L.set_variable("Processor", df['MemoryUsage'][n_teste])

    FS_F.set_variable("Load", float(FS_L.Mamdani_inference(["Load"])["Load"]))
    FS_F.set_variable("Network", float(FS_N.Mamdani_inference(["Network"])["Network"]))
    df["CLPVariation"][n_teste] =  FS_F.Mamdani_inference(["Result"])["Result"]
    df.to_csv("Projeto_2/Lab10-Proj2_TestS.csv")
```