SCIKIT-FUZZY

Instalacija i importovanje scikit-fuzzy biblioteke

```
pip install scikit-fuzzy
```

Pored ove biblioteke potrebne su nam i biblioteke numpy i control:

```
import skfuzzy as fuzz
import numpy as np
from skfuzzy import control as ctrl
```

Kreiranje fazi skupova

Svi skupovi su na početku definisani kao univerzalni pomoću funkcije **np.arrange(start, stop, step)**

- **start** - početak intervala
- **stop** - kraj intervala
- **step** - korak, tj. razlika između dve susedne vrednosti

Input

- **hrana = ctrl.Antecedent(np.arange(0, 11, 1), 'hrana')**
- **usluga = ctrl.Antecedent(np.arange(0, 11, 1), 'usluga')**

Output

- **napojnica = ctrl.Consequent(np.arange(0, 26, 1), 'napojnica', defuzzify_method = mom)**

Koji način defazifikacije želimo, možemo da napišemo kao argument konsekventa (`**defuzzify_method**`), default-no je **centroid**, a ostale mogućnosti su - **bisector, mom** (mean of maximum), **som** (min of maximum), **lom** (max of maximum).

Lingvističke promenljive

Lingvističke promenljive se mogu definisati na dva načina:

- **bez definisanja intervala u okviru kog se nalaze elementi date lingvističke promenljive:**

```
input1.automf(names=['term1', 'term2', 'term3'])
```

- **sa definisanim intervalom u okviru kog se nalaze elementi date lingvističke promenljive, u okviru poziva funkcije pripadnosti fazi skupa:**

```
input1['term1'] = fuzz.trimf(input1.universe, [0, 0, 5])
input1['term2'] = fuzz.gaussmf(input1.universe, 5, 1)
input1['term3'] = fuzz.gaussmf(input1.universe, 10, 3)
```

Lingvističke promenljive

U primeru, lingvističke promenljive za hranu su:

- loša
- srednje ukusna
- veoma ukusna

za uslugu:

- loša
- dobra
- odlicna

i za napojnicu:

- niska
- srednja
- visoka

Za formiranje fazi skupova upotrebljeno je više vrsta **funkcija pripadnosti**: *trougaona, trapezna, gausova, s, z*...

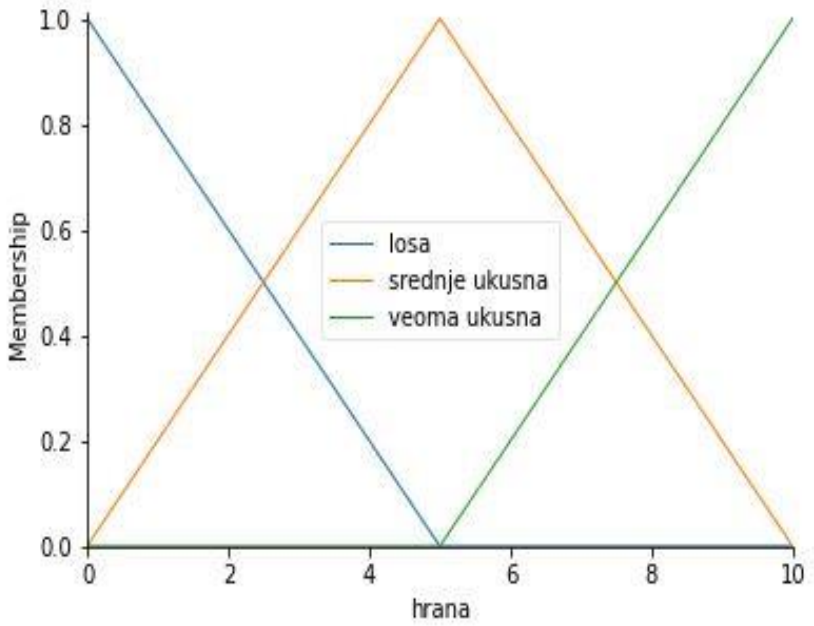
Metoda koja je korišćena u primeru je Mamdanijeva metoda zaključivanja (*Mamdani inference method*)

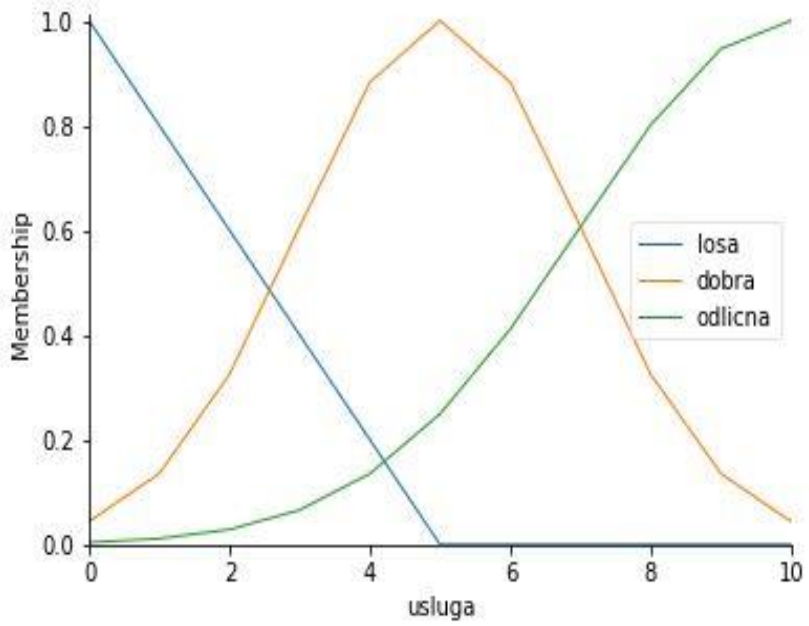
Lingvističke promenljive

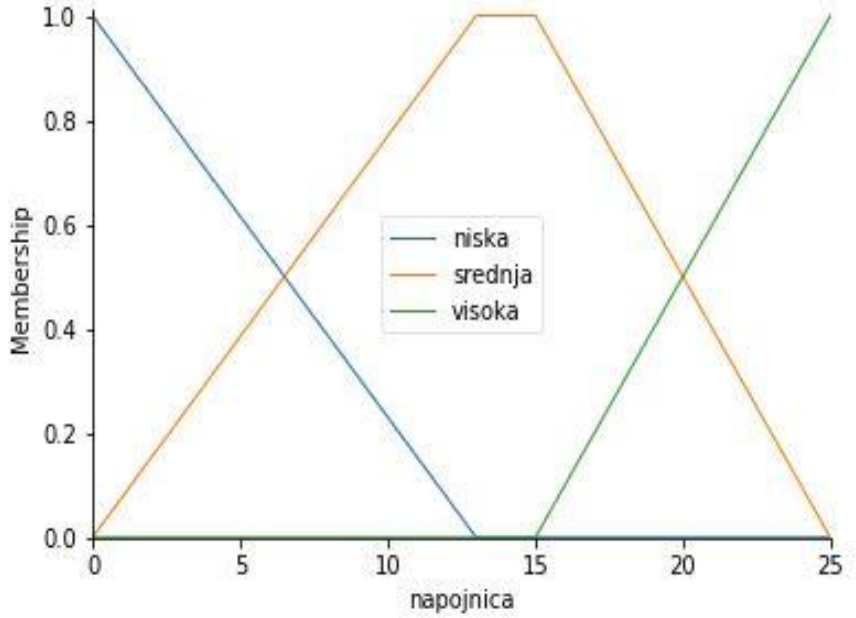
```
hrana.automf(names=['losa', 'srednje ukusna', 'veoma ukusna'])

usluga['losa'] = fuzz.trimf(usluga.universe, [0, 0, 5])
usluga['dobra'] = fuzz.gaussmf(usluga.universe, 5, 2)
usluga['odlicna'] = fuzz.gaussmf(usluga.universe, 10, 3)

#usluga['izvanredna'] = fuzz.trapmf(usluga.universe, [1, 4, 6, 8])
#usluga['izvanredna'] = fuzz.smf(usluga.universe, 4, 6)
#usluga['izvanredna'] = fuzz.zmf(usluga.universe, 4, 6)
#usluga['izvanredna'] = fuzz.gbellmf(usluga.universe, 2, 3, 6)

napojnica['niska'] = fuzz.trimf(napojnica.universe, [0, 0, 13])
napojnica['srednja'] = fuzz.trapmf(napojnica.universe, [0, 13,15, 25])
napojnica['visoka'] = fuzz.trimf(napojnica.universe, [15, 25, 25])
```

Grafički prikaz fazi skupova

```
hrana.view()
usluga.view()
napojnica.view()
```







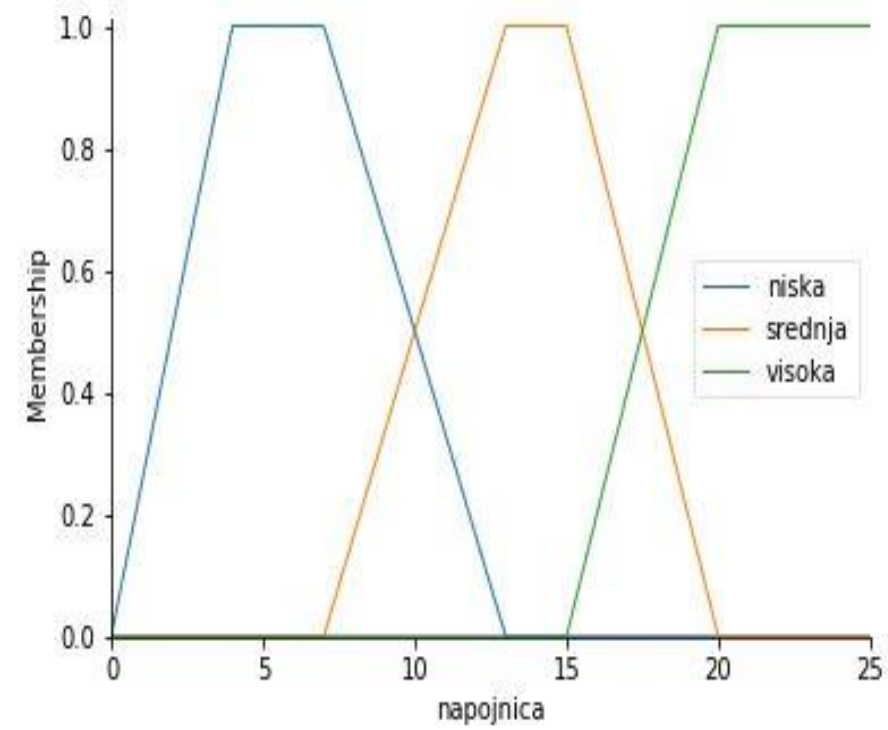
Različiti oblici funkcije pripadnosti

- Trapezna funkcija
- Trougaona funkcija
- S funkcija
- Z funkcija
- Zvonasta funkcija
- Gausova funkcija
- Pi funkcija(S i Z zajedno)

Trapezna funkcija

```
import skfuzzy as fuzz
y=fuzz.trapmf(x, [a,b,c,d])
```

- **x - univerzalni skup,**
- **vector [a,b,c,d] - sadrži 4 vrednosti za koje važi a<=b<=c<=d, i koje predstavljaju x- vrednosti za 4 temena trapeza**



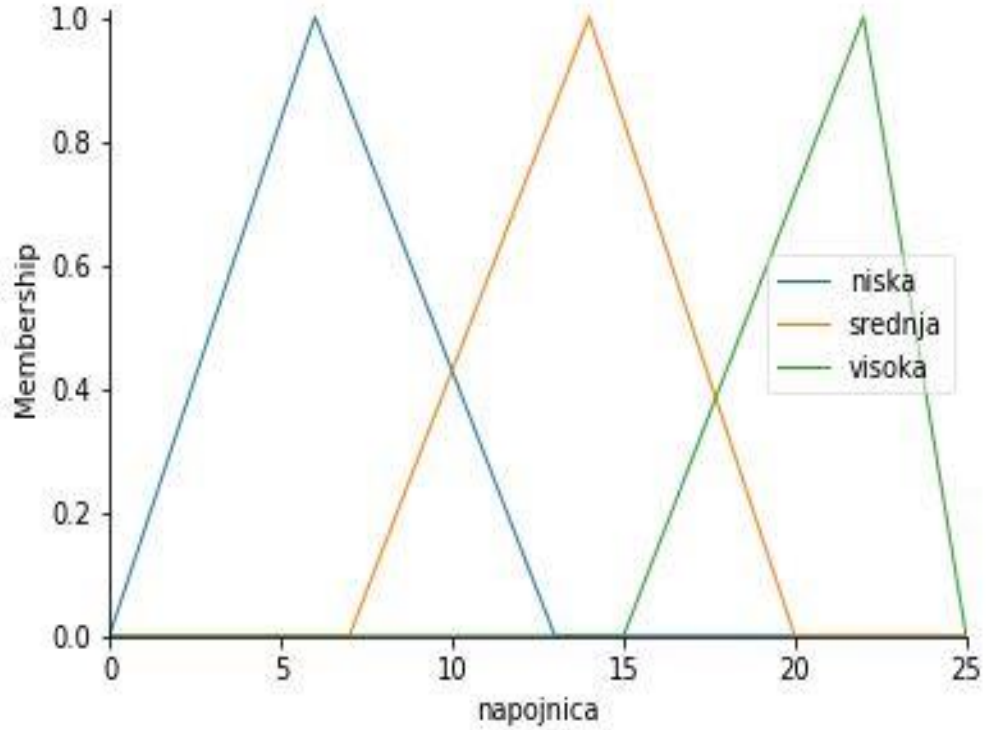

```
napojnica['niska'] = fuzz.trapmf(napojnica.universe,
[0, 4,7,13])
napojnica['srednja'] = fuzz.trapmf(napojnica.universe,
[7, 13,15, 20])
napojnica['visoka'] = fuzz.trapmf(napojnica.universe,
[15, 20,25, 25])

napojnica.view()
```

Trougaona funkcija

```
import skfuzzy as fuzz
y=fuzz.trimf(x, [a, b, c])
```

- **x - univerzalni skup,**
- **vektor [a,b,c] - sadrži 3 vrednosti za koje važi a<=b<=c, i koje predstavljaju 3 temena trougla**

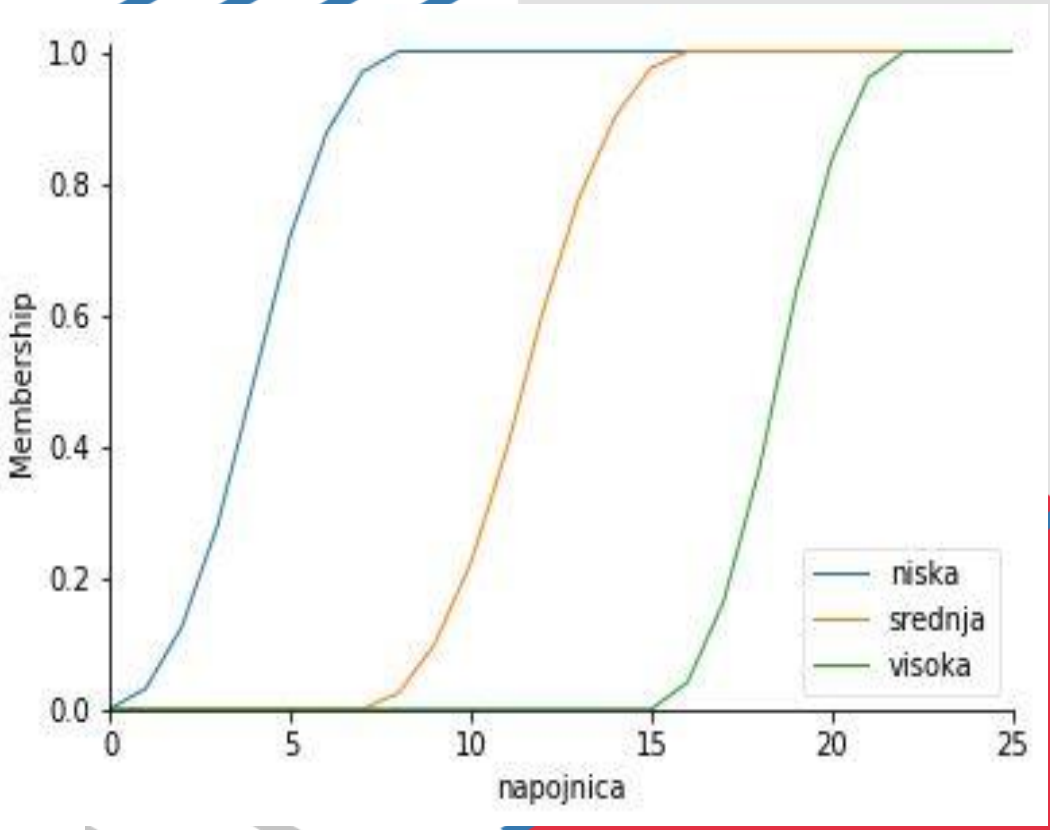
```
napojnica['niska'] = fuzz.trimf(napojnica.universe,
[0,6,13])
napojnica['srednja'] = fuzz.trimf(napojnica.universe
[7,14,20])
napojnica['visoka'] = fuzz.trimf(napojnica.universe,
[15,22,25])

napojnica.view()
```



S funkcija

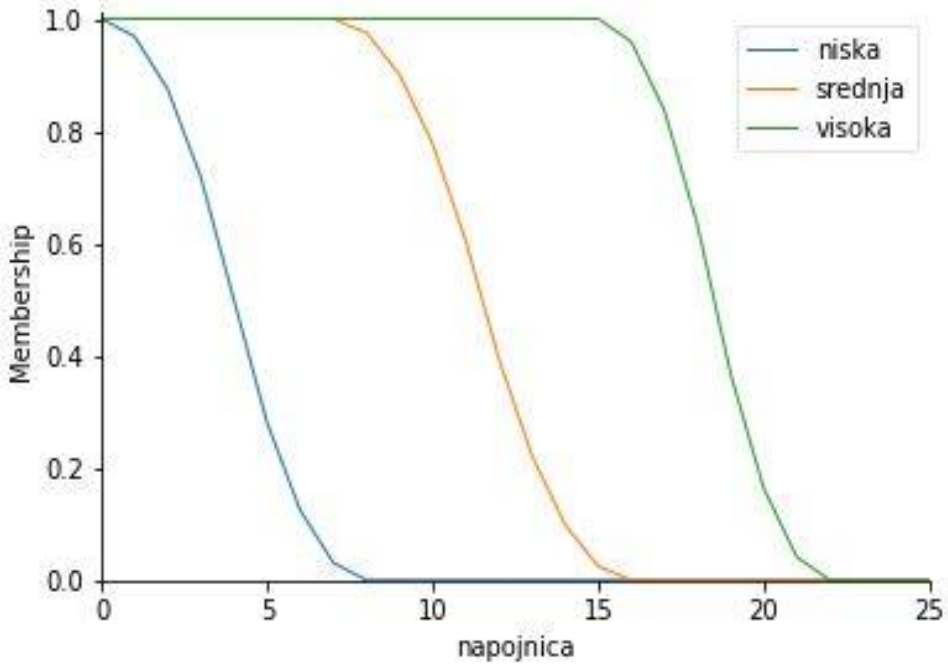
```
import skfuzzy as fuzz
y=fuzz.smf(x,a,b)
```

- **x - univerzalni skup**
- **a – tačka u kojoj funkcija počinje da raste na gore od 0**
- **b - tačka u kojoj funkcija dostiže vrednost 1**




```
napojnica['niska'] = fuzz.smf(napojnica.universe,0,8)
napojnica['srednja'] = fuzz.smf(napojnica.universe,7,16)
napojnica['visoka'] = fuzz.smf(napojnica.universe,15, 22)
napojnica.view()
```

Z funkcija

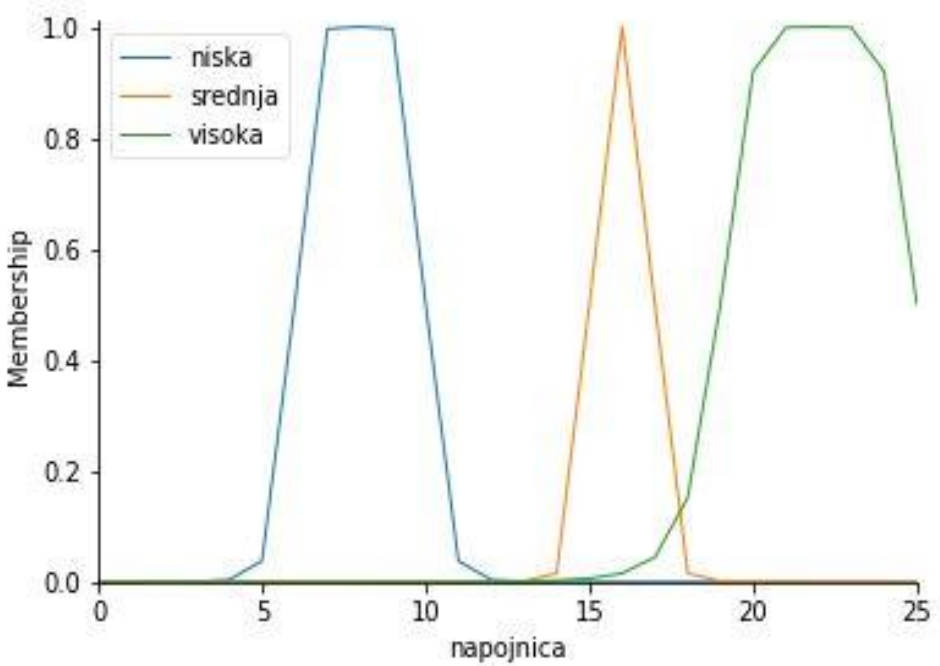



```
import skfuzzy as fuzz
y=fuzz.zmf(x,a,b)
```

- **x- univerzalni skup**
- **a – tačka u kojoj funkcija počinje da opada od 1**
- **b – tačka kad funkcija dostiže 0**

```
napojnica['niska'] = fuzz.zmf(napojnica.universe,0,8)
napojnica['srednja'] = fuzz.zmf(napojnica.universe,7,16)
napojnica['visoka'] = fuzz.zmf(napojnica.universe,15, 22)
napojnica.view()
```

Zvonasta funkcija




```
import skfuzzy as fuzz
y=fuzz.gbellmf(x,a,b,c)
```

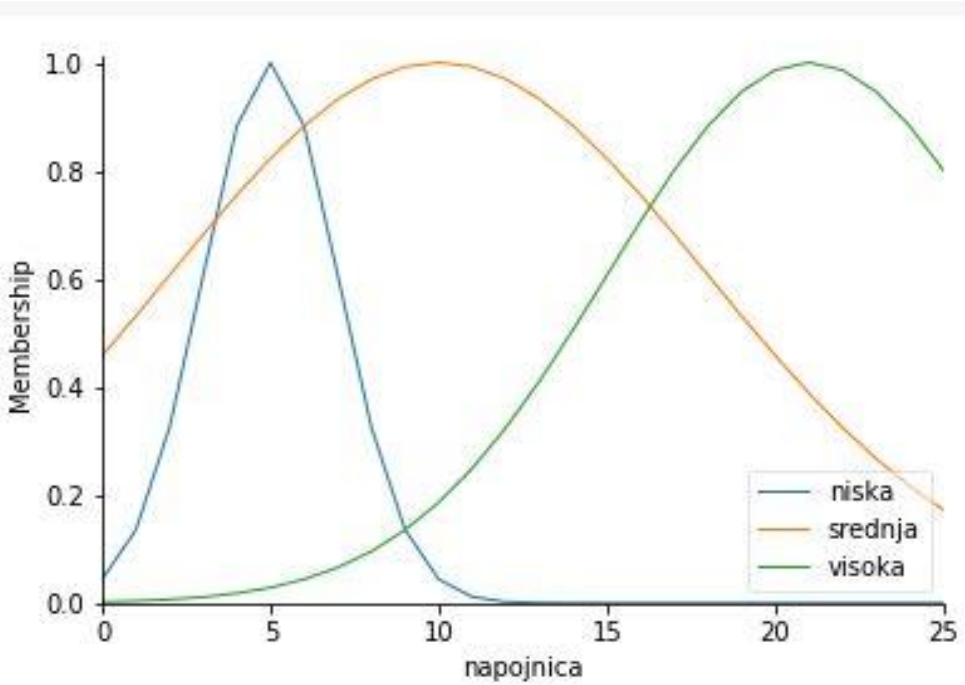
- **x - univerzalni skup**
- **a - parameter koji kontroliše širinu zvona**
- **b - parameter koji predstavlja nagib (što je nagib manji broj, onda je i ugao manji)**
- **c - parametar koji predstavlja centar funkcije (tačka u kojoj funkcija dostiže maksimalnu vrednost)**

```
napojnica['niska'] = fuzz.gbellmf(napojnica.universe,2,4,8)
napojnica['srednja'] = fuzz.gbellmf(napojnica.universe,1,3,16)
napojnica['visoka'] = fuzz.gbellmf(napojnica.universe,3,3,22)
napojnica.view()
```

Gausova funkcija

```
import skfuzzy as fuzz
y=fuzz.gaussmf(x,a,b)
```

- **x - univerzalni skup,**
- **a- srednja vrednost funkcije**
- **b - standardna devijacija**

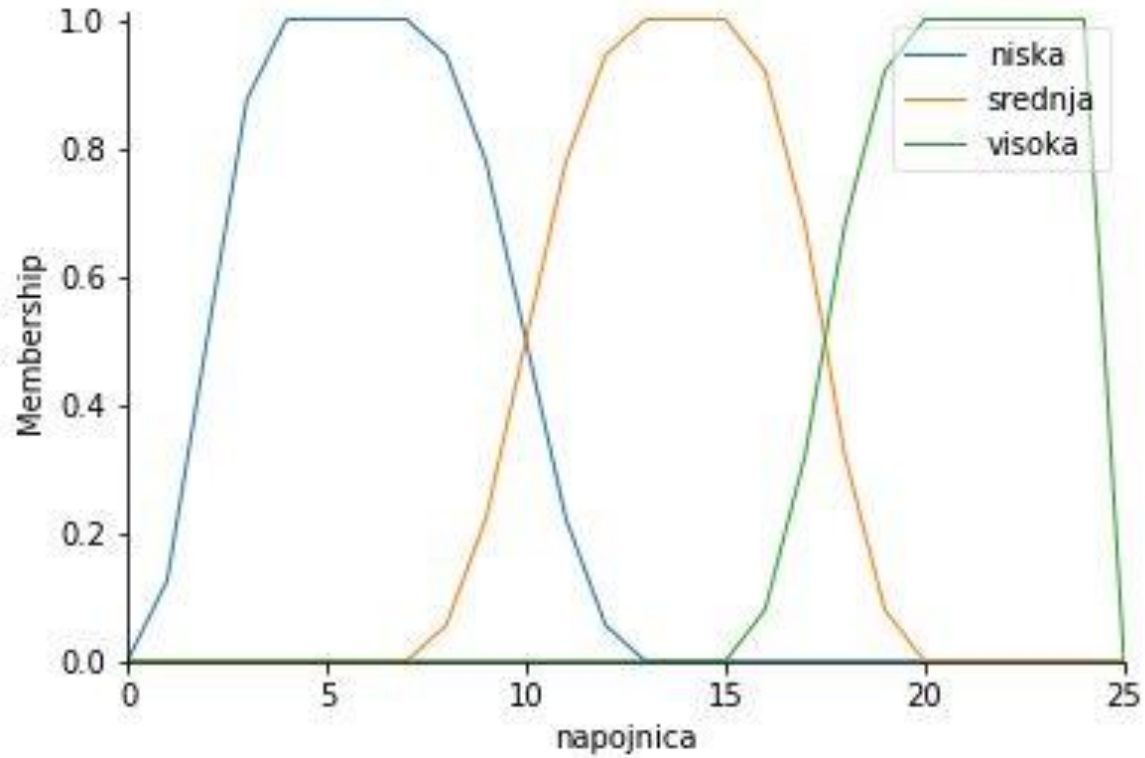



```
napojnica['niska'] = fuzz.gaussmf(napojnica.universe, 5,2)
napojnica['srednja'] = fuzz.gaussmf(napojnica.universe,10,8)
napojnica['visoka'] = fuzz.gaussmf(napojnica.universe,21,6)
napojnica.view()
```

Pi funkcija

```
import skfuzzy as fuzz
y=fuzz.pimf(x,a,b,c,d)
```

- **x – univerzalan skup,**
- **a - tačka u kojoj funkcija počinje da raste od 0**
- **b – tačka u kojoj funkcija dostiže 1**
- **c – tačka u kojoj funkcija počinje da opada od 1**
- **d – tačka u kojoj funkcija dostiže 0**




```
napojnica['niska'] = fuzz.pimf(napojnica.universe, 0, 4,7,13)
napojnica['srednja'] = fuzz.pimf(napojnica.universe, 7, 13,15, 20)
napojnica['visoka'] = fuzz.pimf(napojnica.universe, 15, 20,24, 25)
napojnica.view()
```

Kreiranje pravila odlučivanja

Pravila se definišu na sledeći način, pomoću funkcije **Rule()** iz paketa **control**, koji je sastavni deo biblioteke **skfuzzy**.
Prvi argument funkcije jeste ono što se nalazi sa leve strane pravila, a drugi argument ono što se nalazi sa desne strane pravila.

U primeru su definisana sledeća pravila:

- **1. Ako usluga nije loša ili je hrana veoma ukusna, napojnica će biti visoka.**
- **2. Ako je usluga dobra, napojnica će biti srednja.**
- **3. Ako je usluga loša i hrana je loša, napojnica će biti niska.**

```
rule1 = ctrl.Rule(~usluga['losa'] | hrana['veoma ukusna'], napojnica['visoka'])
rule2 = ctrl.Rule(usluga['dobra'], napojnica['srednja'])
rule3 = ctrl.Rule(usluga['losa'] & hrana['losa'], napojnica['niska'])
```

Kreiranje i simulacija fazi kontrolera

Simulacijom fazi kontrolera se vrši proces zaključivanja.

- Prvo se napravi kontroler (metoda **ControlSystem()**), koji kao argument prihvata listu pravila:

```
napojnica_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
```

- Nakon toga se napravi simulator (metoda **ControlSystemSimulation()**) koji kao argument prihvata kontroler.

Click to add text

```
napojnica_simulator = ctrl.ControlSystemSimulation(napojnica_ctrl)
```

- Potrebno je uneti određene numeričke vrednosti **za hranu i uslugu:**
- **napojnica_simulator.input['hrana'] = 9.5**
- **napojnica_simulator.input['usluga'] = 9.4**, na osnovu kojih će se dobiti vrednost za konsekvent, tj. napojnicu (`napojnica_simulator.compute()`). Taj proces se naziva defazifikacija.

```
napojnica_simulator.input['hrana'] = 9.5
napojnica_simulator.input['usluga'] = 9.4
napojnica_simulator.compute()
print(napojnica_simulator.output['napojnica'])
```

```
Output:
18.581174
```

Vizuelizacija rezultata

Grafički prikaz rezultata zaključivanja dobija se metodom **view()**, koja kao argument prihvata napravljeni simulator.

```
hrana.view(sim=napojnica_simulator)
usluga.view(sim=napojnica_simulator)
napojnica.view(sim=napojnica_simulator)
```