Ankara University Computer Engineering

Com436B-Fuzzy Logic

Project-1

(2019-2020 Spring Semester)

- Instructor Name Dr. Murat OSMANOĞLU

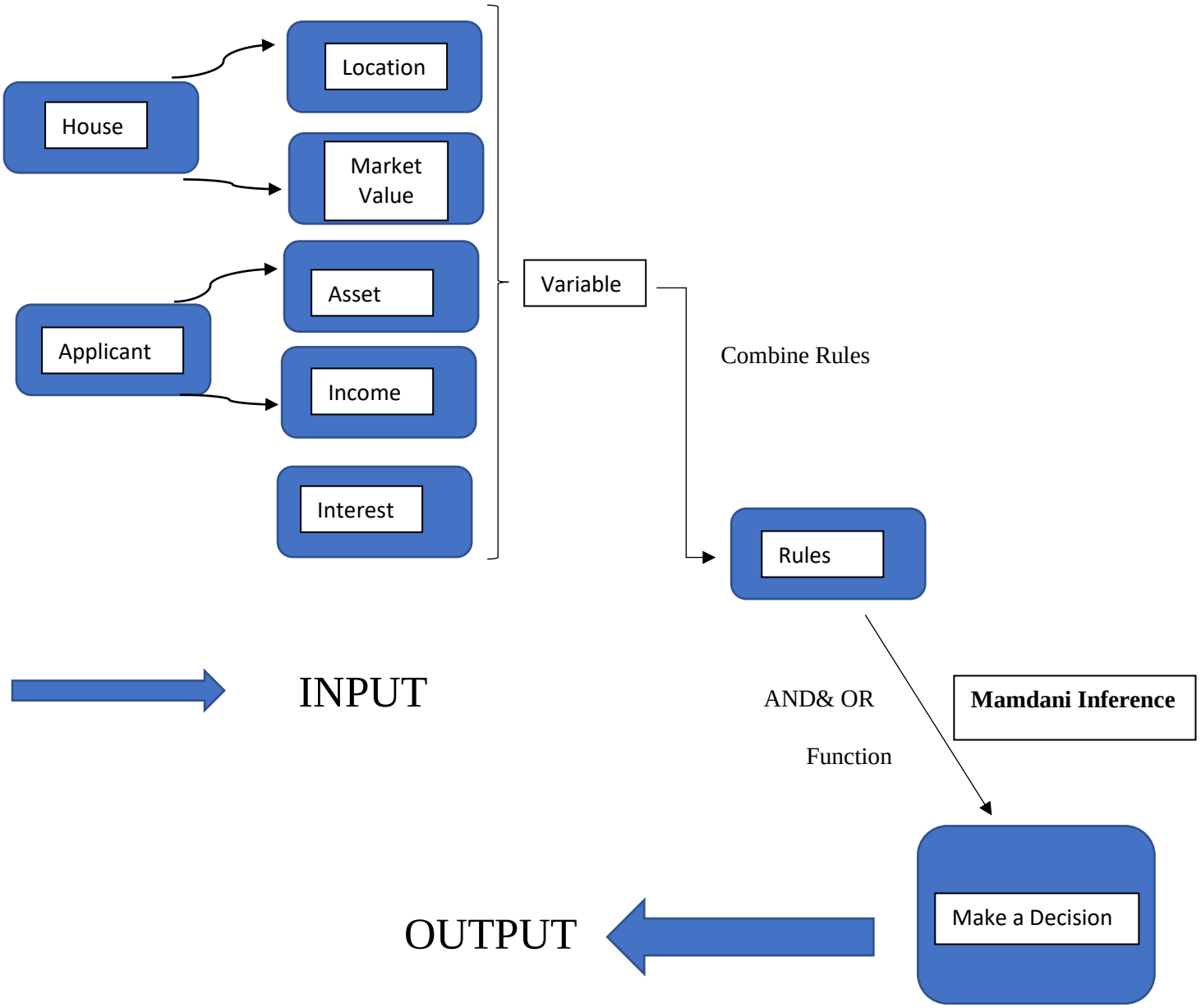
Student Name: Mehmet KEKEÇ

Student No:15290106

IDE used in homework: Visual Studio Code

While developing this project, **Mamdani inference method** was used. Inputs and outputs are fuzzy values in Mamdani inference Membership values are calculated according to the rules triggered by input values. Then the calculated values are given to the max or min operator according to their logical conjunctions and / or in the rules. There are certain definitions and rules in this project, and the execution of the program is done through the rules.

Credit System for Bank Officers



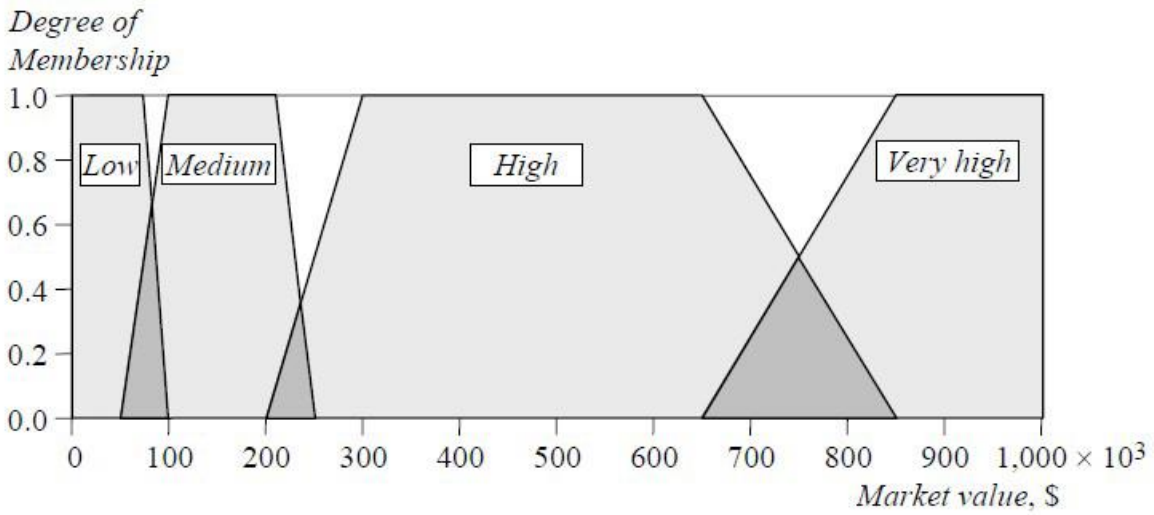
The libraries used are:

```
import numpy
import skfuzzy
import matplotlib.pyplot as plt
import warnings
import matplotlib.cbook
warnings.filterwarnings("ignore",category=matplotlib.cbook.mplDeprecation)
```

Inputs Range

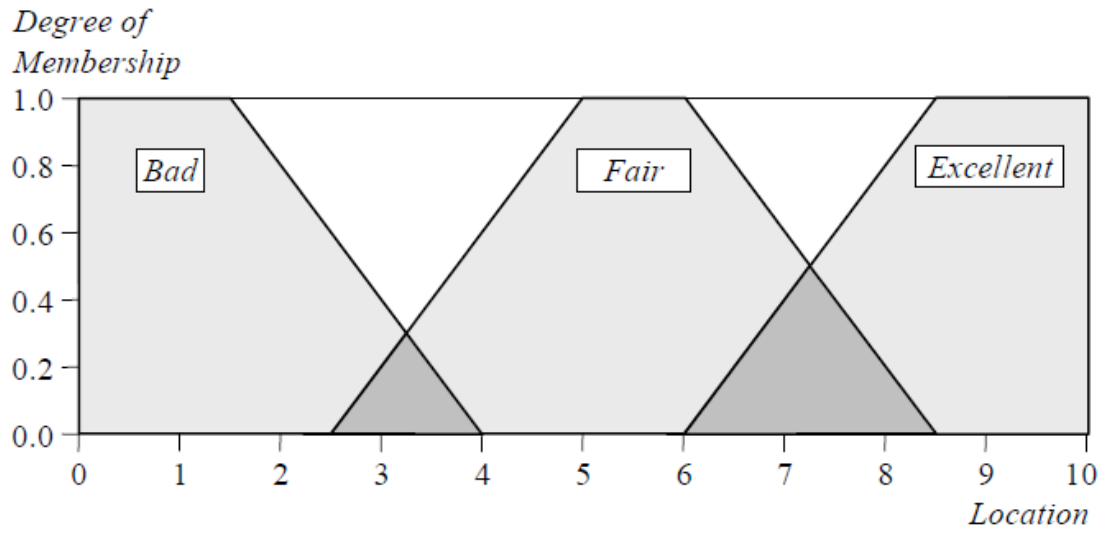
```
hmvalue = numpy.arange(0, 1000) # Market value*1000
hloca = numpy.arange(0, 10, .01) # house location
p_asset = numpy.arange(0,1000) # Asset *1000
p_income = numpy.arange(0,100, .1) # income *1000
interestvalue = numpy.arange(0, 10, .01) # Interest
```

1. Market Value of the House:



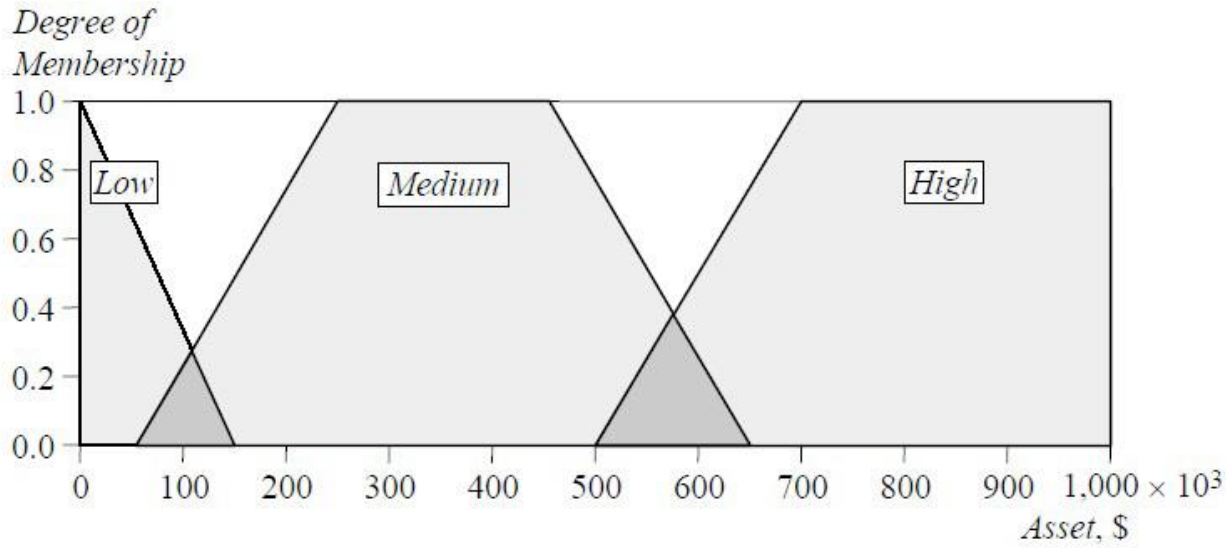

```
mlow = skfuzzy.trapmf(hmvalue, [0, 0, 50, 100])
mmedium = skfuzzy.trapmf(hmvalue, [50, 100, 200, 250])
mhigh = skfuzzy.trapmf(hmvalue, [200, 300, 650, 850])
mvery_high = skfuzzy.trapmf(hmvalue, [650, 850, 1000, 1000])
```

2. Location of the House:



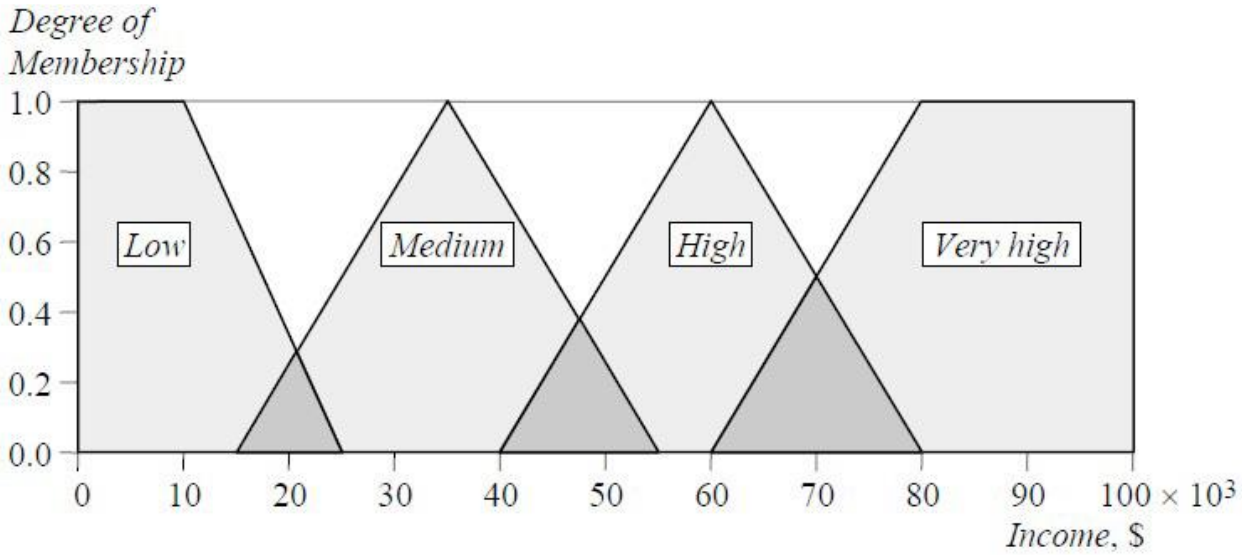

```
Lbad = skfuzzy.trapmf(hloca, [0, 0, 1.5, 4])
Lfair = skfuzzy.trapmf(hloca, [2.5, 5, 6, 8.5])
Lexcellent = skfuzzy.trapmf(hloca, [6, 8.5, 10, 10])
```

3. Asset of the Applicant:



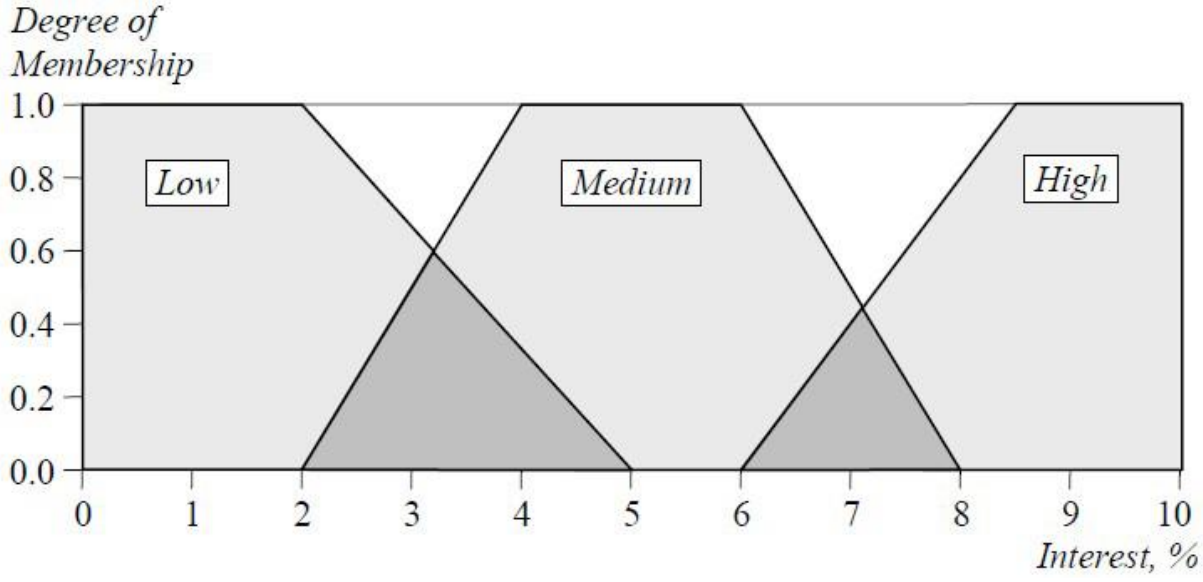

```
pa_low = skfuzzy.trimf(p_asset, [0, 0, 150])
pa_medium = skfuzzy.trapmf(p_asset, [50, 250, 500, 650])
pa_high = skfuzzy.trapmf(p_asset, [500, 700, 1000, 1000])
```

4. Income of the Applicant:

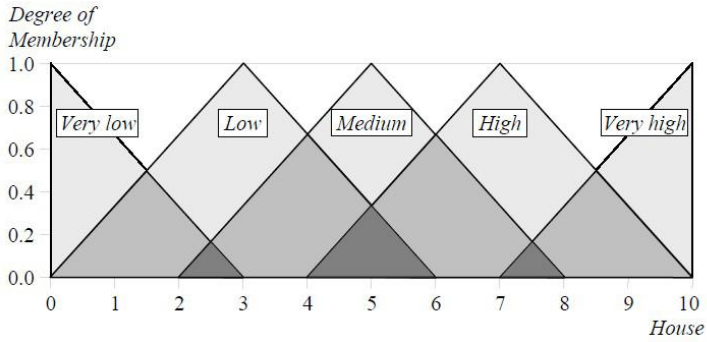



```
p_income_low = skfuzzy.trapmf(p_income, [0, 0, 10, 25])
p_income_medium = skfuzzy.trimf(p_income, [15, 35, 55])
p_income_high = skfuzzy.trimf(p_income, [40, 60, 80])
p_income_very_high = skfuzzy.trapmf(p_income, [60, 80, 100, 100])
```

5. Interest:

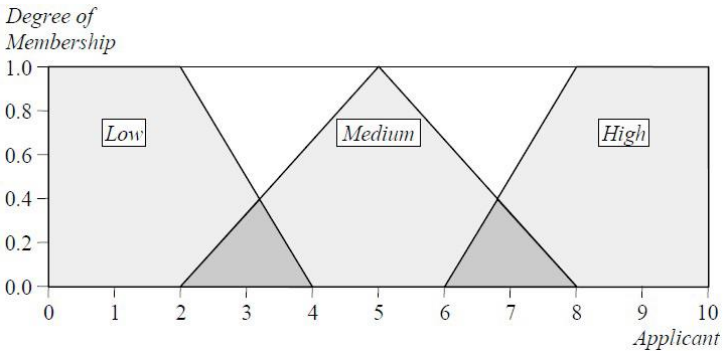



```
b_interest_low = skfuzzy.trapmf(interestvalue, [0, 0, 2, 5])
b_interest_medium = skfuzzy.trapmf(interestvalue, [2, 4, 6, 8])
b_interest_high = skfuzzy.trapmf(interestvalue, [6, 8.5, 10, 10])
```



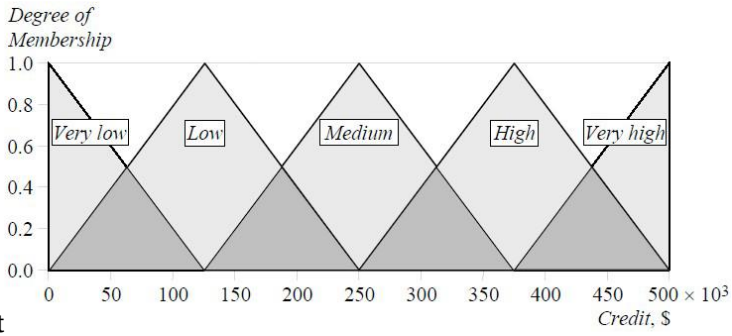
House:

```
house_very_low = skfuzzy.trimf(houvalue, [0, 0, 3])
house_low = skfuzzy.trimf(houvalue, [0, 3, 6])
house_medium = skfuzzy.trimf(houvalue, [2, 5, 8])
house_high = skfuzzy.trimf(houvalue, [4, 7, 10])
house_very_high = skfuzzy.trimf(houvalue, [7, 10, 10])
```



Applicant:

```
applicant_low = skfuzzy.trapmf(appvalue, [0, 0, 2, 4])
applicant_medium = skfuzzy.trimf(appvalue, [2, 5, 8])
applicant_high = skfuzzy.trapmf(appvalue, [6, 8, 10, 10])
```



Credit Amount

```
credit_very_low = skfuzzy.trimf(crevalue, [0, 0, 125])
credit_low = skfuzzy.trimf(crevalue, [0, 125, 250])
credit_medium = skfuzzy.trimf(crevalue, [125, 250, 375])
credit_high = skfuzzy.trimf(crevalue, [250, 375, 500])
credit_very_high = skfuzzy.trimf(crevalue, [375, 500, 500])
```

The rule set:

1. **House Evaluation**

(or)

1. If (Market_value is Low) then (House is Low)
2. If (Location is Bad) then (House is Low)
3. If (Location is Bad) and (Market_value is Low) then (House is Very_low)
4. If (Location is Bad) and (Market_value is Medium) then (House is Low)
5. If (Location is Bad) and (Market_value is High) then (House is Medium)
6. If (Location is Bad) and (Market_value is Very_high) then (House is High)
7. If (Location is Fair) and (Market_value is Low) then (House is Low)
8. If (Location is Fair) and (Market_value is Medium) then (House is Medium)
9. If (Location is Fair) and (Market_value is High) then (House is High)
10. If (Location is Fair) and (Market_value is Very_high) then (House is Very_high)
11. If (Location is Excellent) and (Market_value is Low) then (House is Medium)
12. If (Location is Excellent) and (Market_value is Medium) then (House is High)
13. If (Location is Excellent) and (Market_value is High) then (House is Very_high)
14. If (Location is Excellent) and (Market_value is Very_high) then (House is Very_high)

```
house_act_low1 = numpy.fmin(mlevel_low, house_low)
house_act_low2 = numpy.fmin(Llevel_bad, house_low)
house_act_very_low = and_rule(Llevel_bad, mlevel_low, house_very_low)
house_act_low3 = and_rule(Llevel_bad, mlevel_medium, house_low)
house_act_medium1 = and_rule(Llevel_bad, mlevel_high, house_medium)
house_act_high1 = and_rule(Llevel_bad, mlevel_very_high, house_high)
house_act_low4 = and_rule(Llevel_fair, mlevel_low, house_low)
house_act_medium2 = and_rule(Llevel_fair, mlevel_medium, house_medium)
house_act_high2 = and_rule(Llevel_fair, mlevel_high, house_high)
house_act_very_high1 = and_rule(Llevel_fair, mlevel_very_high, house_very_high)
house_act_medium3 = and_rule(Llevel_excellent, mlevel_low, house_medium)
house_act_high3 = and_rule(Llevel_excellent, mlevel_medium, house_high)
house_act_very_high2 = and_rule(Llevel_excellent, mlevel_high, house_very_high)
house_act_very_high3 = and_rule(Llevel_excellent, mlevel_very_high, house_very_hig
h)
```

2. Applicant Evaluation

(or)

1. If (Asset is Low) and (Income is Low) then (Applicant is Low)
2. If (Asset is Low) and (Income is Medium) then (Applicant is Low)
3. If (Asset is Low) and (Income is High) then (Applicant is Medium)

4. If (Asset is Low) and (Income is Very_high) then (Applicant is High)

5. If (Asset is Medium) and (Income is Low) then (Applicant is Low)

6. If (Asset is Medium) and (Income is Medium) then (Applicant is Medium)

7. If (Asset is Medium) and (Income is High) then (Applicant is High)

8. If (Asset is Medium) and (Income is Very_high) then (Applicant is High)

9. If (Asset is High) and (Income is Low) then (Applicant is Medium)

10.If (Asset is High) and (Income is Medium) then (Applicant is Medium)

11.If (Asset is High) and (Income is High) then (Applicant is High)

12.If (Asset is High) and (Income is Very_high) then (Applicant is High)

```
applicant_act_low1 = and_rule(pa_level_low, p_income_level_low, applicant_low)
applicant_act_low2 = and_rule(pa_level_low, p_income_level_medium, applicant_low)
applicant_act_medium1 = and_rule(pa_level_low, p_income_level_high, applicant_medi
um)
applicant_act_high1 = and_rule(pa_level_low, p_income_level_very_high, applicant_h
igh)
applicant_act_low3 = and_rule(pa_level_medium, p_income_level_low, applicant_low)
applicant_act_medium2 = and_rule(pa_level_medium, p_income_level_medium, applicant
_medium)
applicant_act_high2 = and_rule(pa_level_medium, p_income_level_high, applicant_hig
h)
applicant_act_high3 = and_rule(pa_level_medium, p_income_level_very_high, applican
t_high)
applicant_act_medium3 = and_rule(pa_level_high, p_income_level_low, applicant_medi
um)
applicant_act_medium4 = and_rule(pa_level_high, p_income_level_medium, applicant_m
edium)
applicant_act_high4 = and_rule(pa_level_high, p_income_level_high, applicant_high)
applicant_act_high5 = and_rule(pa_level_high, p_income_level_very_high, applicant_
high)
```

3. Evaluation of the Amount of Credit
(or)

1. If (Income is Low) and (Interest is Medium) then (Credit is Very_low)
2. If (Income is Low) and (Interest is High) then (Credit is Very_low)
3. If (Income is Medium) and (Interest is High) then (Credit is Low)
4. If (Applicant is Low) then (Credit is Very_low)
5. If (House is Very_low) then (Credit is Very_low)
6. If (Applicant is Medium) and (House is Very_low) then (Credit is Low)
7. If (Applicant is Medium) and (House is Low) then (Credit is Low)
8. If (Applicant is Medium) and (House is Medium) then (Credit is Medium)
9. If (Applicant is Medium) and (House is High) then (Credit is High)
10.If (Applicant is Medium) and (House is Very_high) then (Credit is High)
11.If (Applicant is High) and (House is Very_low) then (Credit is Low)
12.If (Applicant is High) and (House is Low) then (Credit is Medium)
13.If (Applicant is High) and (House is Medium) then (Credit is High)
14.If (Applicant is High) and (House is High) then (Credit is High)
15.If (Applicant is High) and (House is Very_high) then (Credit is Very_high)

```
credit_act_very_low1 = and_rule(p_income_level_low, b_interest_level_medium, c
redit_very_low)
credit_act_very_low2 = and_rule(p_income_level_low, b_interest_level_high, cre
dit_very_low)
credit_act_low1 = and_rule(p_income_level_medium, b_interest_level_high, credi
t_low)
credit_act_very_low3 = numpy.fmin(applicant_level_low, credit_very_low)
credit_act_very_low4 = numpy.fmin(house_level_very_low, credit_very_low)
credit_act_low2 = and_rule(applicant_level_medium, house_level_very_low, credi
t_low)
credit_act_low3 = and_rule(applicant_level_medium, house_level_low, credit_low
)
credit_act_medium1 = and_rule(applicant_level_medium, house_level_medium, cred
it_medium)
credit_act_high1 = and_rule(applicant_level_medium, house_level_high, credit_h
igh)
credit_act_high2 = and_rule(applicant_level_medium, house_level_very_high, cre
dit_high)
credit_act_low4 = and_rule(applicant_level_high, house_level_very_low, credit_
low)
credit_act_medium2 = and_rule(applicant_level_high, house_level_low, credit_me
dium)
credit_act_high3 = and_rule(applicant_level_high, house_level_medium, credit_h
igh)
credit_act_high4 = and_rule(applicant_level_high, house_level_high, credit_hig
h)
```

AND and OR Functions

Thanks to these functions, we can adapt the given rules to the Mamdani inference model**.**

For "and" we used minimum fonction to apply rules

For "or" we used maximum function to apply rules

```
def and_rule(x, y, z):
    rule = numpy.fmin(x, y)
    act = numpy.fmin(rule, z)
    return act

def or_rule(x, y, z):
    rule = numpy.fmax(x, y)
    act = numpy.fmax(rule, z)
    return act
```

House_Rule = C1 OR C2 OR C3 OR C4 OR C5 OR C6 OR C7 OR C8 OR C9 OR C10 OR C11 OR C12 OR C13 OR C14

Applicant_Rule=C1 OR C2 OR C3 OR C4 OR C5 OR C6 OR C7 OR C8 OR C9 OR C10 OR C11 OR C12

Credit_Rule== C1 OR C2 OR C3 OR C4 OR C5 OR C6 OR C7 OR C8 OR C9 OR C10 OR C11 OR C12 OR C13 OR C14 OR C15

The method written for the processing of these rules;

```
def apply_all_rules(market_value, location, assets, income, interest, verbose=
0):
    house = apply_house_rules(market_value, location, verbose)
    applicant = apply_applicant_rules(assets,income, verbose)
    credit = apply_credit_rules(house, income, interest, applicant)
    return credit
```

Making a Decision

- After all the rules applied, we defuzify the output of the rules with mean of maximum and we generate a single value as a decision of the system

```
def apply_all_rules(mvalue, location, assets, income, interest, verbose=0):
 house = apply_house_rules(mvalue, location, verbose)
 applicant = apply_applicant_rules(assets,income, verbose)
 credit = apply_credit_rules(house, income, interest, applicant)
 return credit
def make_decision(mvalue, location, assets, income, interest, verbose=0):
    credit = apply_all_rules(mvalue, location, assets, income, interest, verbo
se)
    # defuzzification with mean of maximum
    defuzz_credit = skfuzzy.defuzz(crevalue, credit,'mom')
    max_n = numpy.max(credit)
return defuzz_credit
```

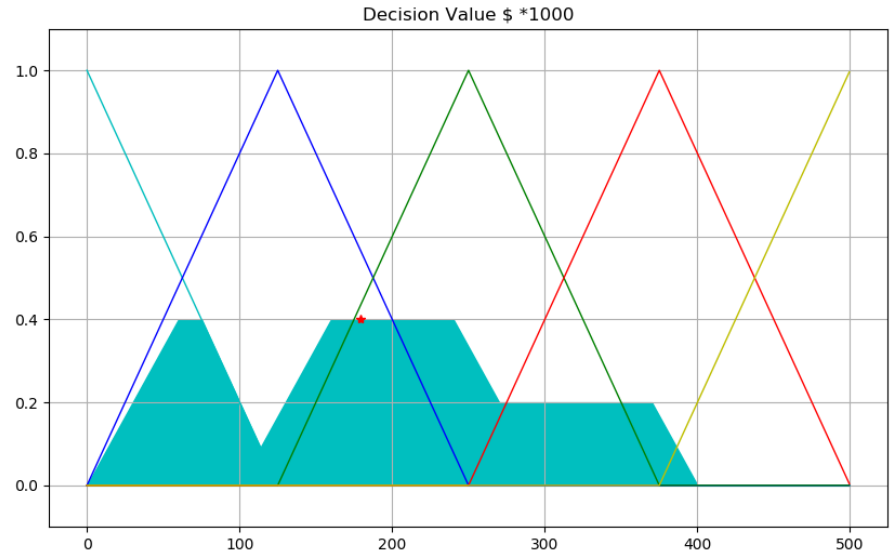
Result

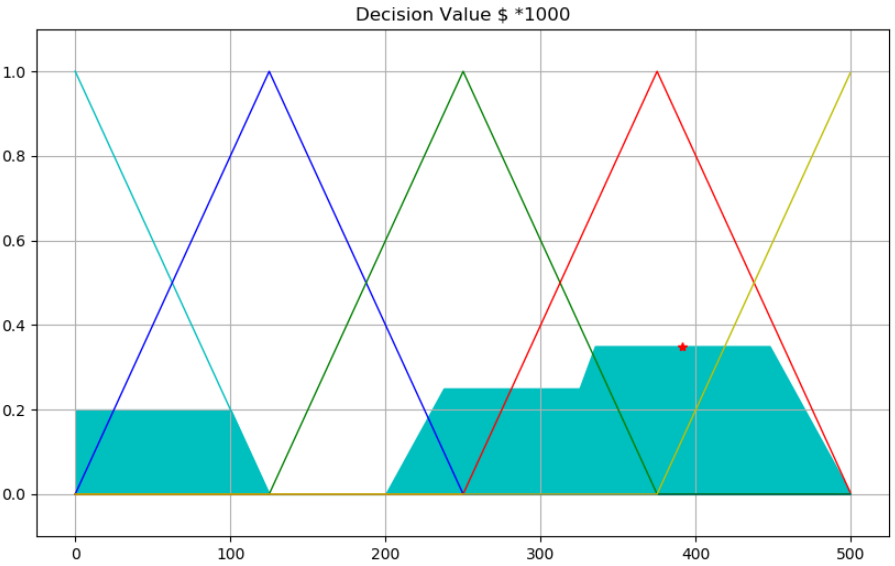
```
credit_decision = make_decision(420, 5, 120, 50, 2, verbose=1)#person 1
credit_decision = make_decision(260, 2, 100, 35, 1, verbose=1)#person 2
credit_decision = make_decision(180, 3, 550, 45, 6, verbose=1) #person 3
```

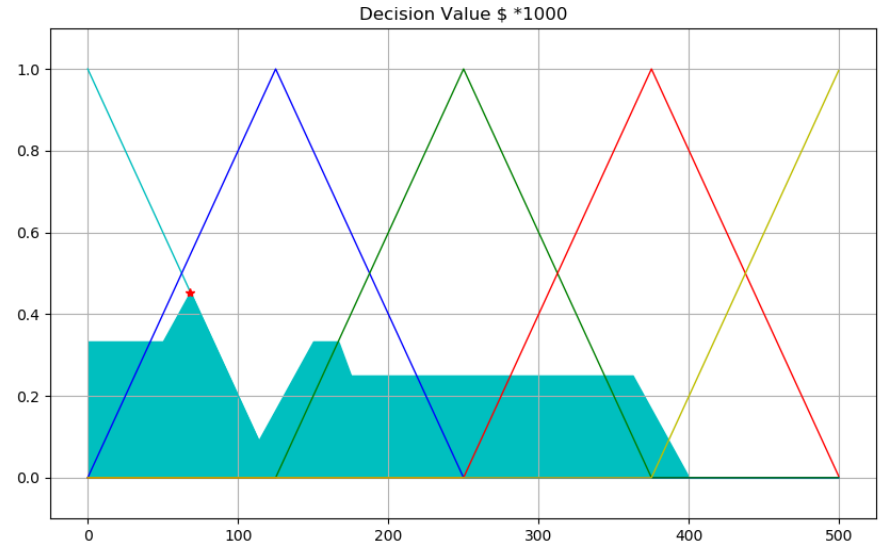
Output: 179227 $ for person 1

Output: 391000 $ for person 2

Output: 68000 $ for person 3







Results are marked in red on the charts.

Appendices

```
import numpy
import skfuzzy
import matplotlib.pyplot as plt
import warnings
import matplotlib.cbook
warnings.filterwarnings("ignore",category=matplotlib.cbook.mplDeprecation)

hmvalue = numpy.arange(0, 1000) # Market value*1000
hloca = numpy.arange(0, 10, .01) # house location
p_asset = numpy.arange(0,1000) # Asset *1000
p_income = numpy.arange(0,100, .1) # income *1000
interestvalue = numpy.arange(0, 10, .01) # Interest
# house market
mlow = skfuzzy.trapmf(hmvalue, [0, 0, 50, 100])
mmedium = skfuzzy.trapmf(hmvalue, [50, 100, 200, 250])
mhigh = skfuzzy.trapmf(hmvalue, [200, 300, 650, 850])
mvery_high = skfuzzy.trapmf(hmvalue, [650, 850, 1000, 1000])
# house location
Lbad = skfuzzy.trapmf(hloca, [0, 0, 1.5, 4])
Lfair = skfuzzy.trapmf(hloca, [2.5, 5, 6, 8.5])
Lexcellent = skfuzzy.trapmf(hloca, [6, 8.5, 10, 10])
# p asset
pa_low = skfuzzy.trimf(p_asset, [0, 0, 150])
pa_medium = skfuzzy.trapmf(p_asset, [50, 250, 500, 650])
pa_high = skfuzzy.trapmf(p_asset, [500, 700, 1000, 1000])
# p income
p_income_low = skfuzzy.trapmf(p_income, [0, 0, 10, 25])
p_income_medium = skfuzzy.trimf(p_income, [15, 35, 55])
p_income_high = skfuzzy.trimf(p_income, [40, 60, 80])
p_income_very_high = skfuzzy.trapmf(p_income, [60, 80, 100, 100])
# interest
b_interest_low = skfuzzy.trapmf(interestvalue, [0, 0, 2, 5])
b_interest_medium = skfuzzy.trapmf(interestvalue, [2, 4, 6, 8])
b_interest_high = skfuzzy.trapmf(interestvalue, [6, 8.5, 10, 10])

houvalue = numpy.arange(0, 10, .01) # House evaluation range
appvalue = numpy.arange(0, 10, .01) # applicant evalutaion range
crevalue = numpy.arange(0, 500, .5) # Credit evalutaion Range $ x10^3

# house
house_very_low = skfuzzy.trimf(houvalue, [0, 0, 3])
house_low = skfuzzy.trimf(houvalue, [0, 3, 6])
house_medium = skfuzzy.trimf(houvalue, [2, 5, 8])
house_high = skfuzzy.trimf(houvalue, [4, 7, 10])
house_very_high = skfuzzy.trimf(houvalue, [7, 10, 10])
#applicant
applicant_low = skfuzzy.trapmf(appvalue, [0, 0, 2, 4])
```

```
applicant_medium = skfuzzy.trimf(appvalue, [2, 5, 8])
applicant_high = skfuzzy.trapmf(appvalue, [6, 8, 10, 10])
# credit evalutation output fuzzy sets
credit_very_low = skfuzzy.trimf(crevalue, [0, 0, 125])
credit_low = skfuzzy.trimf(crevalue, [0, 125, 250])
credit_medium = skfuzzy.trimf(crevalue, [125, 250, 375])
credit_high = skfuzzy.trimf(crevalue, [250, 375, 500])
credit_very_high = skfuzzy.trimf(crevalue, [375, 500, 500])

def and_rule(x, y, z):
    rule = numpy.fmin(x, y)
    act = numpy.fmin(rule, z)
    return act

def or_rule(x, y, z):
    rule = numpy.fmax(x, y)
    act = numpy.fmax(rule, z)
    return act

def apply_house_rules(mvalue, location, verbose=0):
    # house market value functions
    mlevel_low = skfuzzy.interp_membership(hmvalue, mlow, mvalue)
    mlevel_medium = skfuzzy.interp_membership(hmvalue, mmedium, mvalue)
    mlevel_high = skfuzzy.interp_membership(hmvalue, mhigh, mvalue)
    mlevel_very_high = skfuzzy.interp_membership(hmvalue, mvery_high, mvalue)
    # house location
    Llevel_bad = skfuzzy.interp_membership(hloca, Lbad, location)
    Llevel_fair = skfuzzy.interp_membership(hloca, Lfair, location)
    Llevel_excellent = skfuzzy.interp_membership(hloca, Lexcellent, location)

    ### rules
    house_act_low1 = numpy.fmin(mlevel_low, house_low)
    house_act_low2 = numpy.fmin(Llevel_bad, house_low)
    house_act_very_low = and_rule(Llevel_bad, mlevel_low, house_very_low)
    house_act_low3 = and_rule(Llevel_bad, mlevel_medium, house_low)
    house_act_medium1 = and_rule(Llevel_bad, mlevel_high, house_medium)
    house_act_high1 = and_rule(Llevel_bad, mlevel_very_high, house_high)
    house_act_low4 = and_rule(Llevel_fair, mlevel_low, house_low)
    house_act_medium2 = and_rule(Llevel_fair, mlevel_medium, house_medium)
    house_act_high2 = and_rule(Llevel_fair, mlevel_high, house_high)
    house_act_very_high1 = and_rule(Llevel_fair, mlevel_very_high, house_very_high
)
    house_act_medium3 = and_rule(Llevel_excellent, mlevel_low, house_medium)
    house_act_high3 = and_rule(Llevel_excellent, mlevel_medium, house_high)
    house_act_very_high2 = and_rule(Llevel_excellent, mlevel_high, house_very_high
)
    house_act_very_high3 = and_rule(Llevel_excellent, mlevel_very_high, house_very
_high)
    # combine the rules
```

```
    step = or_rule(house_act_low1, house_act_low2, house_act_low3)
    house_act_low = numpy.fmax(step, house_act_low4)
    house_act_medium = or_rule(house_act_medium1, house_act_medium2, house_act_med
ium3)
    house_act_high = or_rule(house_act_high1, house_act_high2, house_act_high3)
    house_act_very_high = or_rule(house_act_very_high1, house_act_very_high2, hous
e_act_very_high3)
    step = or_rule(house_act_very_low, house_act_low, house_act_medium)
    house = or_rule(step, house_act_high, house_act_very_high)
    return house

def apply_applicant_rules(assets, income, verbose=0):
    # person asset
    pa_level_low = skfuzzy.interp_membership(p_asset, pa_low, assets)
    pa_level_medium = skfuzzy.interp_membership(p_asset, pa_medium, assets)
    pa_level_high = skfuzzy.interp_membership(p_asset, pa_high, assets)
    # person income
    p_income_level_low = skfuzzy.interp_membership(p_income, p_income_low, income)
    p_income_level_medium = skfuzzy.interp_membership(p_income, p_income_medium, i
ncome)
    p_income_level_high = skfuzzy.interp_membership(p_income, p_income_high, incom
e)
    p_income_level_very_high = skfuzzy.interp_membership(p_income, p_income_very_h
igh, income)

    applicant_act_low1 = and_rule(pa_level_low, p_income_level_low, applicant_low)
    applicant_act_low2 = and_rule(pa_level_low, p_income_level_medium, applicant_l
ow)
    applicant_act_medium1 = and_rule(pa_level_low, p_income_level_high, applicant_
medium)
    applicant_act_high1 = and_rule(pa_level_low, p_income_level_very_high, applica
nt_high)
    applicant_act_low3 = and_rule(pa_level_medium, p_income_level_low, applicant_l
ow)
    applicant_act_medium2 = and_rule(pa_level_medium, p_income_level_medium, appli
cant_medium)
    applicant_act_high2 = and_rule(pa_level_medium, p_income_level_high, applicant
_high)
    applicant_act_high3 = and_rule(pa_level_medium, p_income_level_very_high, appl
icant_high)
    applicant_act_medium3 = and_rule(pa_level_high, p_income_level_low, applicant_
medium)
    applicant_act_medium4 = and_rule(pa_level_high, p_income_level_medium, applica
nt_medium)
    applicant_act_high4 = and_rule(pa_level_high, p_income_level_high, applicant_h
igh)
    applicant_act_high5 = and_rule(pa_level_high, p_income_level_very_high, applic
ant_high)
```

```
    # combine the rules
    applicant_act_low = or_rule(applicant_act_low1, applicant_act_low2, applicant_
act_low3)

    step = or_rule(applicant_act_medium1, applicant_act_medium2, applicant_act_med
ium3)
    applicant_act_medium = numpy.fmax(step, applicant_act_medium4)
    step = or_rule(applicant_act_high1, applicant_act_high2, applicant_act_high3)
    applicant_act_high = or_rule(step, applicant_act_high4, applicant_act_high5)

    applicant = or_rule(applicant_act_low, applicant_act_medium, applicant_act_hig
h)
    return applicant

def apply_credit_rules(house, income, interest, applicant):
    # house
    house_level_very_low = numpy.fmin(house, house_low)
    house_level_low = numpy.fmin(house, house_low)
    house_level_medium = numpy.fmin(house, house_medium)
    house_level_high = numpy.fmin(house, house_high)
    house_level_very_high = numpy.fmin(house, house_very_high)
    # person income
    p_income_level_low = skfuzzy.interp_membership(p_income, p_income_low, income)
    p_income_level_medium = skfuzzy.interp_membership(p_income, p_income_medium, i
ncome)
    p_income_level_high = skfuzzy.interp_membership(p_income, p_income_high, incom
e)
    p_income_level_very_high = skfuzzy.interp_membership(p_income, p_income_very_h
igh, income)
    # interest
    b_interest_level_low = skfuzzy.interp_membership(interestvalue, b_interest_low
, interest)
    b_interest_level_medium = skfuzzy.interp_membership(interestvalue, b_interest_
medium, interest)
    b_interest_level_high = skfuzzy.interp_membership(interestvalue, b_interest_hi
gh, interest)
    # applicant
    applicant_level_low = numpy.fmin(applicant, applicant_low)
    applicant_level_medium = numpy.fmin(applicant, applicant_medium)
    applicant_level_high = numpy.fmin(applicant, applicant_high)

    credit_act_very_low1 = and_rule(p_income_level_low, b_interest_level_medium, c
redit_very_low)
    credit_act_very_low2 = and_rule(p_income_level_low, b_interest_level_high, cre
dit_very_low)
    credit_act_low1 = and_rule(p_income_level_medium, b_interest_level_high, credi
t_low)
    credit_act_very_low3 = numpy.fmin(applicant_level_low, credit_very_low)
    credit_act_very_low4 = numpy.fmin(house_level_very_low, credit_very_low)
```

```
    credit_act_low2 = and_rule(applicant_level_medium, house_level_very_low, credi
t_low)
    credit_act_low3 = and_rule(applicant_level_medium, house_level_low, credit_low
)
    credit_act_medium1 = and_rule(applicant_level_medium, house_level_medium, cred
it_medium)
    credit_act_high1 = and_rule(applicant_level_medium, house_level_high, credit_h
igh)
    credit_act_high2 = and_rule(applicant_level_medium, house_level_very_high, cre
dit_high)
    credit_act_low4 = and_rule(applicant_level_high, house_level_very_low, credit_
low)
    credit_act_medium2 = and_rule(applicant_level_high, house_level_low, credit_me
dium)
    credit_act_high3 = and_rule(applicant_level_high, house_level_medium, credit_h
igh)
    credit_act_high4 = and_rule(applicant_level_high, house_level_high, credit_hig
h)
    credit_act_very_high = and_rule(applicant_level_high, house_level_very_high, c
redit_very_high)


    step = or_rule(credit_act_very_low1, credit_act_very_low2, credit_act_very_low
3)
    credit_act_very_low = numpy.fmax(step, credit_act_very_low4)
    step = or_rule(credit_act_low1, credit_act_low2, credit_act_low3)
    credit_act_low = numpy.fmax(step, credit_act_low4)
    credit_act_medium = numpy.fmax(credit_act_medium1, credit_act_medium2)
    step = or_rule(credit_act_high1, credit_act_high2, credit_act_high3)
    credit_act_high = numpy.fmax(step, credit_act_high4)
    step = or_rule(credit_act_very_low, credit_act_low, credit_act_medium)
    credit = or_rule(step, credit_act_high, credit_act_very_high)

    return credit

def apply_all_rules(mvalue, location, assets, income, interest, verbose=0):
    house = apply_house_rules(mvalue, location, verbose)
    applicant = apply_applicant_rules(assets,income, verbose)
    credit = apply_credit_rules(house, income, interest, applicant)
    return credit
def make_decision(mvalue, location, assets, income, interest, verbose=0):
    credit = apply_all_rules(mvalue, location, assets, income, interest, verbose)
    # defuzzification with mean of maximum
    defuzz_credit = skfuzzy.defuzz(crevalue, credit,'mom')
    max_n = numpy.max(credit)

    if (verbose == 1):
        matplotlib.pyplot.rcParams["figure.figsize"] = 10, 6
```

```
        matplotlib.pyplot.plot(crevalue, credit_very_low, 'c', linestyle='-
', linewidth=1)
        matplotlib.pyplot.plot(crevalue, credit_low, 'b', linestyle='-
', linewidth=1)
        matplotlib.pyplot.plot(crevalue, credit_medium, 'g', linestyle='-
', linewidth=1)
        matplotlib.pyplot.plot(crevalue, credit_high, 'r', linestyle='-
', linewidth=1)
        matplotlib.pyplot.plot(crevalue, credit_very_high, 'y', linestyle='-
', linewidth=1),matplotlib.pyplot.title("Decision Value $ *1000" )

        matplotlib.pyplot.fill_between(crevalue, credit, color='c')
        matplotlib.pyplot.ylim(-0.1, 1.1)
        matplotlib.pyplot.grid(True)

        matplotlib.pyplot.plot(defuzz_credit, max_n, '*', color='r')
        matplotlib.pyplot.show()

    print ("Output: ", int(defuzz_credit*1000), "$")
    return defuzz_credit

credit_decision = make_decision(420, 5, 120, 50, 2, verbose=1)#person 1
credit_decision = make_decision(260, 2, 100, 35, 1, verbose=1)#person 2
credit_decision = make_decision(180, 3, 550, 45, 6, verbose=1) #person 3
```