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

```
Collecting scikit-fuzzy
  Downloading https://files.pythonhosted.org/packages/6c/f0/5eb5dbe0fd8dfe7d4651a8f4e591a196623a22b9e533
     |████████████████████████████████| 1.0MB 2.7MB/s
Requirement already satisfied: numpy>=1.6.0 in /usr/local/lib/python3.6/dist-packages (from scikit-fuzzy
Requirement already satisfied: scipy>=0.9.0 in /usr/local/lib/python3.6/dist-packages (from scikit-fuzzy
Requirement already satisfied: networkx>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from scikit-fu
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from networkx
Building wheels for collected packages: scikit-fuzzy
  Building wheel for scikit-fuzzy (setup.py) ... done
  Created wheel for scikit-fuzzy: filename=scikit_fuzzy-0.4.2-cp36-none-any.whl size=894070 sha256=223c3
  Stored in directory: /root/.cache/pip/wheels/b9/4e/77/da79b16f64ef1738d95486e2731eea09d73e90a724650966
Successfully built scikit-fuzzy
Installing collected packages: scikit-fuzzy
Successfully installed scikit-fuzzy-0.4.2
```

Fuzzy Logic Project

Credit decision system

```
# needed libraries
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

Fuzzy Input Set Ranges

```
# ranges of the sets
x_house_market = np.arange(0, 1000, 1) # Market value $ x10^3
x_house_location = np.arange(0, 10, .01) # location of house
x_person_asset = np.arange(0,1000, 1) # Asset $ x10^3
x_person_income = np.arange(0,100, .1) # income $ x10^3
x_interest = np.arange(0, 10, .01) # Interest %
```

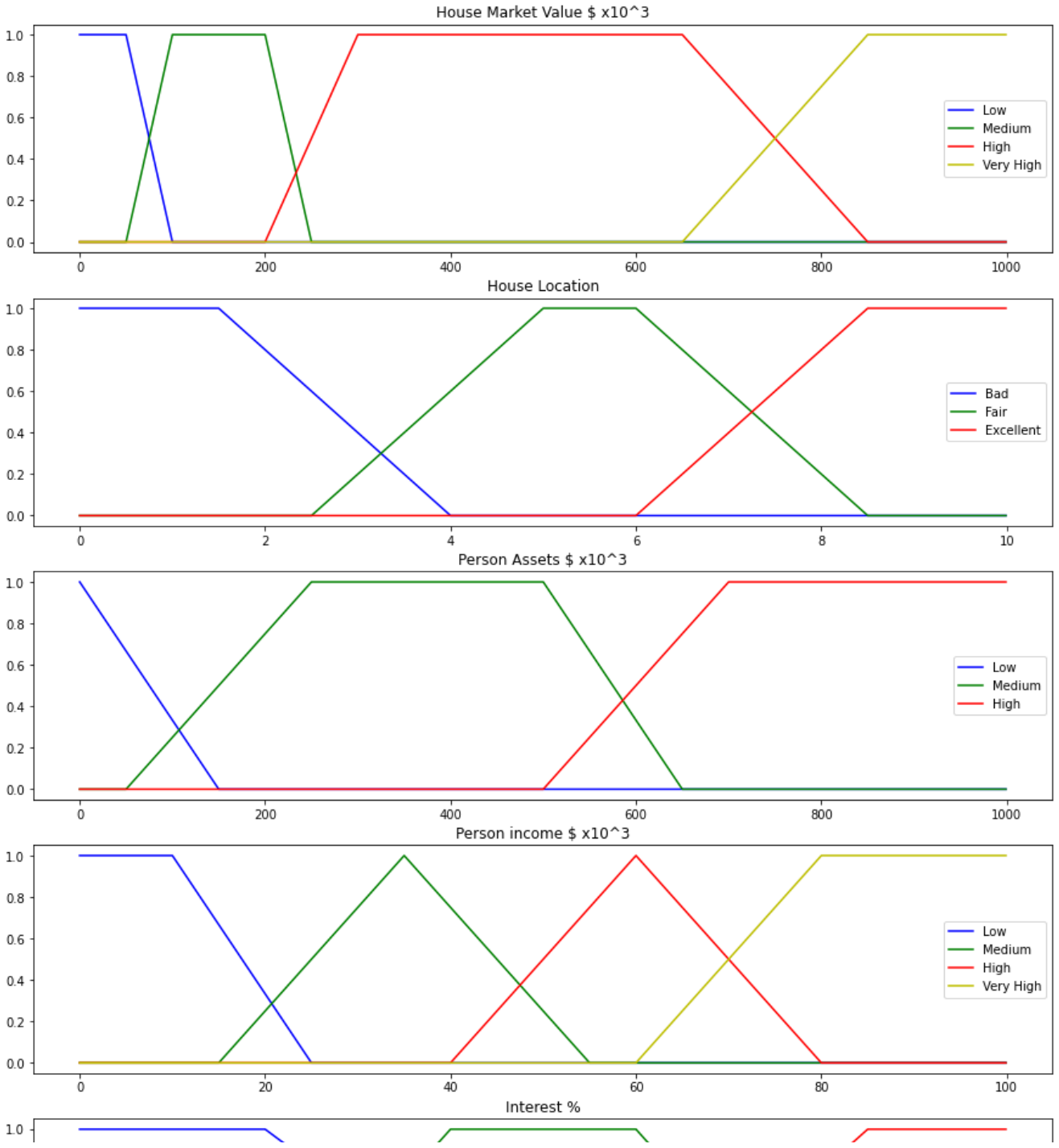
Defining the Fuzzy Inputs Sets

```
# house market value sets
market_low = fuzz.trapmf(x_house_market, [0, 0, 50, 100])
market_medium = fuzz.trapmf(x_house_market, [50, 100, 200, 250])
market_high = fuzz.trapmf(x_house_market, [200, 300, 650, 850])
market_very_high = fuzz.trapmf(x_house_market, [650, 850, 1000, 1000])

# house location sets
location_bad = fuzz.trapmf(x_house_location, [0, 0, 1.5, 4])
location_fair = fuzz.trapmf(x_house_location, [2.5, 5, 6, 8.5])
location_excellent = fuzz.trapmf(x_house_location, [6, 8.5, 10, 10])

# person asset sets
p_asset_low = fuzz.trimf(x_person_asset, [0, 0, 150])
p_asset_medium = fuzz.trapmf(x_person_asset, [50, 250, 500, 650])
p_asset_high = fuzz.trapmf(x_person_asset, [500, 700, 1000, 1000])

# person income sets
p_income_low = fuzz.trapmf(x_person_income, [0, 0, 10, 25])
```

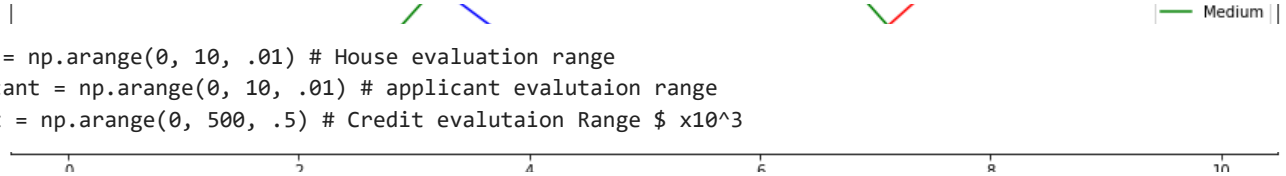
```
p_income_low = fuzz.trapmf(x_person_income, [0, 0, 10, 25])
p_income_medium = fuzz.trimf(x_person_income, [15, 35, 55])
p_income_high = fuzz.trimf(x_person_income, [40, 60, 80])
p_income_very_high = fuzz.trapmf(x_person_income, [60, 80, 100, 100])


# interest sets
b_interest_low = fuzz.trapmf(x_interest, [0, 0, 2, 5])
b_interest_medium = fuzz.trapmf(x_interest, [2, 4, 6, 8])
b_interest_high = fuzz.trapmf(x_interest, [6, 8.5, 10, 10])
```

Showing the Defined Fuzzy Inputs

```
plt.rcParams["figure.figsize"] = 15, 20
# house market value
plt.subplot(5,1,1), plt.plot(x_house_market, market_low, 'b', linewidth=1.5, label='Low')
plt.subplot(5,1,1), plt.plot(x_house_market, market_medium, 'g', linewidth=1.5, label='Medium')
plt.subplot(5,1,1), plt.plot(x_house_market, market_high, 'r', linewidth=1.5, label='High')
plt.subplot(5,1,1), plt.plot(x_house_market, market_very_high, 'y', linewidth=1.5, label='Very High'),plt.tit
plt.legend()

# house location
plt.subplot(5,1,2), plt.plot(x_house_location, location_bad, 'b', linewidth=1.5, label='Bad')
plt.subplot(5,1,2), plt.plot(x_house_location, location_fair, 'g', linewidth=1.5, label='Fair')
plt.subplot(5,1,2), plt.plot(x_house_location, location_excellent, 'r', linewidth=1.5, label='Excellent'),plt
plt.legend()

# person Assets
plt.subplot(5,1,3), plt.plot(x_person_asset, p_asset_low, 'b', linewidth=1.5, label='Low')
plt.subplot(5,1,3), plt.plot(x_person_asset, p_asset_medium, 'g', linewidth=1.5, label='Medium')
plt.subplot(5,1,3), plt.plot(x_person_asset, p_asset_high, 'r', linewidth=1.5, label='High'),plt.title("Perso
plt.legend()

# person income
plt.subplot(5,1,4), plt.plot(x_person_income, p_income_low, 'b', linewidth=1.5, label='Low')
plt.subplot(5,1,4), plt.plot(x_person_income, p_income_medium, 'g', linewidth=1.5, label='Medium')
plt.subplot(5,1,4), plt.plot(x_person_income, p_income_high, 'r', linewidth=1.5, label='High')
plt.subplot(5,1,4), plt.plot(x_person_income, p_income_very_high, 'y', linewidth=1.5, label='Very High'),plt.
plt.legend()

# Interest
plt.subplot(5,1,5), plt.plot(x_interest, b_interest_low, 'b', linewidth=1.5, label='Low')
plt.subplot(5,1,5), plt.plot(x_interest, b_interest_medium, 'g', linewidth=1.5, label='Medium')
plt.subplot(5,1,5), plt.plot(x_interest, b_interest_high, 'r', linewidth=1.5, label='High'),plt.title("Intere
plt.legend()

plt.show()
```



Defining Fuzzy Output Set Ranges




```
x_house = np.arange(0, 10, .01) # House evaluation range
x_applicant = np.arange(0, 10, .01) # applicant evalutaion range
x_credit = np.arange(0, 500, .5) # Credit evalutaion Range $ x10^3
```

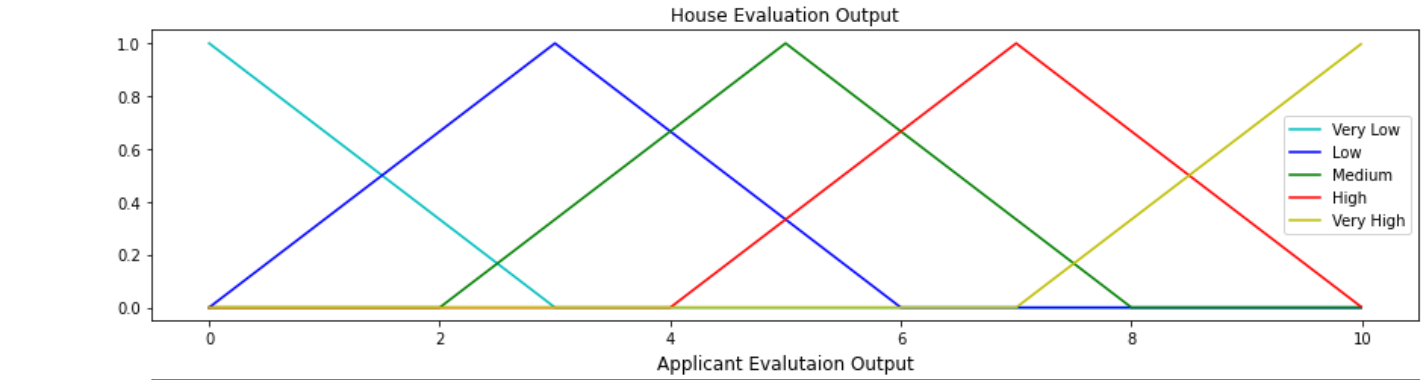
Defining the Fuzzy Output Sets

```
# house evalutation output fuzzy sets
house_very_low = fuzz.trimf(x_house, [0, 0, 3])
house_low = fuzz.trimf(x_house, [0, 3, 6])
house_medium = fuzz.trimf(x_house, [2, 5, 8])
house_high = fuzz.trimf(x_house, [4, 7, 10])
house_very_high = fuzz.trimf(x_house, [7, 10, 10])

# applicant evalutation output fuzzy sets
```

```
# applicant evaluation output fuzzy sets
applicant_low = fuzz.trapmf(x_applicant, [0, 0, 2, 4])
applicant_medium = fuzz.trimf(x_applicant, [2, 5, 8])
applicant_high = fuzz.trapmf(x_applicant, [6, 8, 10, 10])

# credit evalutation output fuzzy sets
credit_very_low = fuzz.trimf(x_credit, [0, 0, 125])
credit_low = fuzz.trimf(x_credit, [0, 125, 250])
credit_medium = fuzz.trimf(x_credit, [125, 250, 375])
credit_high = fuzz.trimf(x_credit, [250, 375, 500])
credit_very_high = fuzz.trimf(x_credit, [375, 500, 500])
```

Showing the Defined Fuzzy Outputs

```
plt.rcParams["figure.figsize"] = 15, 12
# house evaluation
plt.subplot(3,1,1), plt.plot(x_house, house_very_low, 'c', linewidth=1.5, label='Very Low')
plt.subplot(3,1,1), plt.plot(x_house, house_low, 'b', linewidth=1.5, label='Low')
plt.subplot(3,1,1), plt.plot(x_house, house_medium, 'g', linewidth=1.5, label='Medium')
plt.subplot(3,1,1), plt.plot(x_house, house_high, 'r', linewidth=1.5, label='High')
plt.subplot(3,1,1), plt.plot(x_house, house_very_high, 'y', linewidth=1.5, label='Very High'),plt.title("Hous
plt.legend()

# applicant evaluation
plt.subplot(3,1,2), plt.plot(x_applicant, applicant_low, 'b', linewidth=1.5, label='Low')
plt.subplot(3,1,2), plt.plot(x_applicant, applicant_medium, 'g', linewidth=1.5, label='Medium')
plt.subplot(3,1,2), plt.plot(x_applicant, applicant_high, 'r', linewidth=1.5, label='High'),plt.title("Applic
plt.legend()

# credit evaluation
plt.subplot(3,1,3), plt.plot(x_credit, credit_very_low, 'c', linewidth=1.5, label='Very Low')
plt.subplot(3,1,3), plt.plot(x_credit, credit_low, 'b', linewidth=1.5, label='Low')
plt.subplot(3,1,3), plt.plot(x_credit, credit_medium, 'g', linewidth=1.5, label='Medium')
plt.subplot(3,1,3), plt.plot(x_credit, credit_high, 'r', linewidth=1.5, label='High')
plt.subplot(3,1,3), plt.plot(x_credit, credit_very_high, 'y', linewidth=1.5, label='Very High'),plt.title("Cr
plt.legend()

plt.show()
```



AND and OR helper functions

- For "and" we used minimum fonction to apply rules
- For "or" we used maximum function to apply rules

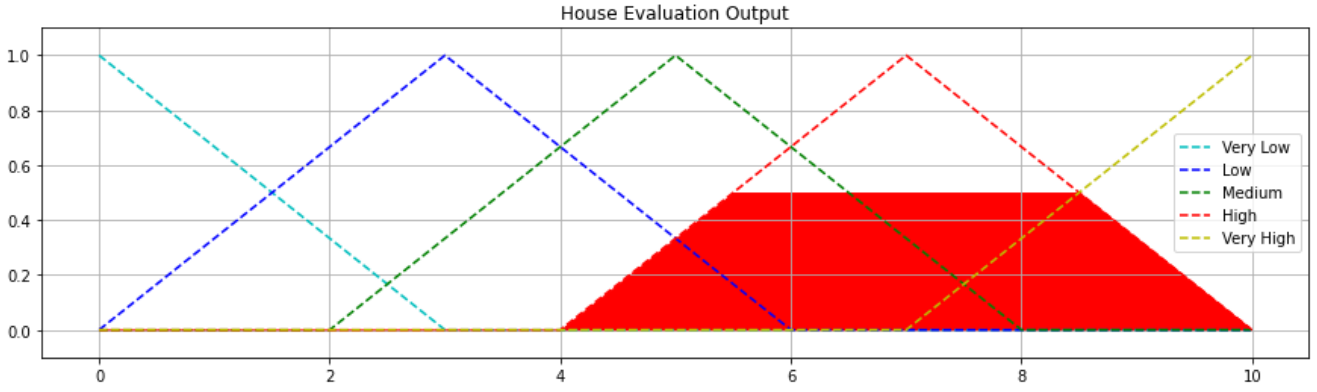
```
def and_rule(x, y, z):
    rule = np.fmin(x, y)
    act = np.fmin(rule, z)
    return act

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

```
def apply_house_rules(market_value, location, verbose=0):
    # house market value functions
    market_level_low = fuzz.interp_membership(x_house_market, market_low, market_value)
    market_level_medium = fuzz.interp_membership(x_house_market, market_medium, market_value)
    market_level_high = fuzz.interp_membership(x_house_market, market_high, market_value)
    market_level_very_high = fuzz.interp_membership(x_house_market, market_very_high, market_value)

    # house location
    location_level_bad = fuzz.interp_membership(x_house_location, location_bad, location)
    location_level_fair = fuzz.interp_membership(x_house_location, location_fair, location)
    location_level_excellent = fuzz.interp_membership(x_house_location, location_excellent, location)

    ### rules
    # 1. If (Market_value is Low) then (House is Low)
    house_act_low1 = np.fmin(market_level_low, house_low)
    # 2. If (Location is Bad) then (House is Low)
    house_act_low2 = np.fmin(location_level_bad, house_low)
    # 3. If (Location is Bad) and (Market_value is Low) then (House is Very_low)
    house_act_very_low = and_rule(location_level_bad, market_level_low, house_very_low)
    # 4. If (Location is Bad) and (Market_value is Medium) then (House is Low)
    house_act_low3 = and_rule(location_level_bad, market_level_medium, house_low)
    # 5. If (Location is Bad) and (Market_value is High) then (House is Medium)
    house_act_medium1 = and_rule(location_level_bad, market_level_high, house_medium)
    # 6. If (Location is Bad) and (Market_value is Very_high) then (House is High)
    house_act_high1 = and_rule(location_level_bad, market_level_very_high, house_high)
    # 7. If (Location is Fair) and (Market_value is Low) then (House is Low)
    house_act_low4 = and_rule(location_level_fair, market_level_low, house_low)
    # 8. If (Location is Fair) and (Market_value is Medium) then (House is Medium)
    house_act_medium2 = and_rule(location_level_fair, market_level_medium, house_medium)
    # 9. If (Location is Fair) and (Market_value is High) then (House is High)
    house_act_high2 = and_rule(location_level_fair, market_level_high, house_high)
    # 10. If (Location is Fair) and (Market_value is Very_high) then (House is Very_high)
    house_act_very_high1 = and_rule(location_level_fair, market_level_very_high, house_very_high)
    # 11. If (Location is Excellent) and (Market_value is Low) then (House is Medium)
    house_act_medium3 = and_rule(location_level_excellent, market_level_low, house_medium)
    # 12. If (Location is Excellent) and (Market_value is Medium) then (House is High)
```

```
    house_act_high3 = and_rule(location_level_excellent, market_level_medium, house_high)
    # 13. If (Location is Excellent) and (Market_value is High) then (House is Very_high)
    house_act_very_high2 = and_rule(location_level_excellent, market_level_high, house_very_high)
    # 14. If (Location is Excellent) and (Market_value is Very_high) then (House is Very_high)
    house_act_very_high3 = and_rule(location_level_excellent, market_level_very_high, house_very_high)

    # combine the rules
    step = or_rule(house_act_low1, house_act_low2, house_act_low3)
    house_act_low = np.fmax(step, house_act_low4)

    house_act_medium = or_rule(house_act_medium1, house_act_medium2, house_act_medium3)

    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, house_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)

    # if we want to see the graph of the output
    if verbose == 1:
        plt.rcParams["figure.figsize"] = 15, 4
        plt.plot(x_house, house_very_low, 'c', linestyle='--', linewidth=1.5, label='Very Low')
        plt.plot(x_house, house_low, 'b', linestyle='--', linewidth=1.5, label='Low')
        plt.plot(x_house, house_medium, 'g', linestyle='--', linewidth=1.5, label='Medium')
        plt.plot(x_house, house_high, 'r', linestyle='--', linewidth=1.5, label='High')
        plt.plot(x_house, house_very_high, 'y', linestyle='--', linewidth=1.5, label='Very High'),plt.title("H
        plt.legend()

        plt.fill_between(x_house, house, color='r')
        plt.ylim(-0.1, 1.1)
        plt.grid(True)
        plt.show()

    return house
```

Example Output For Rule Base 1

```
## triying the function with example
h_eval = apply_house_rules(250, 4, verbose=1)
```

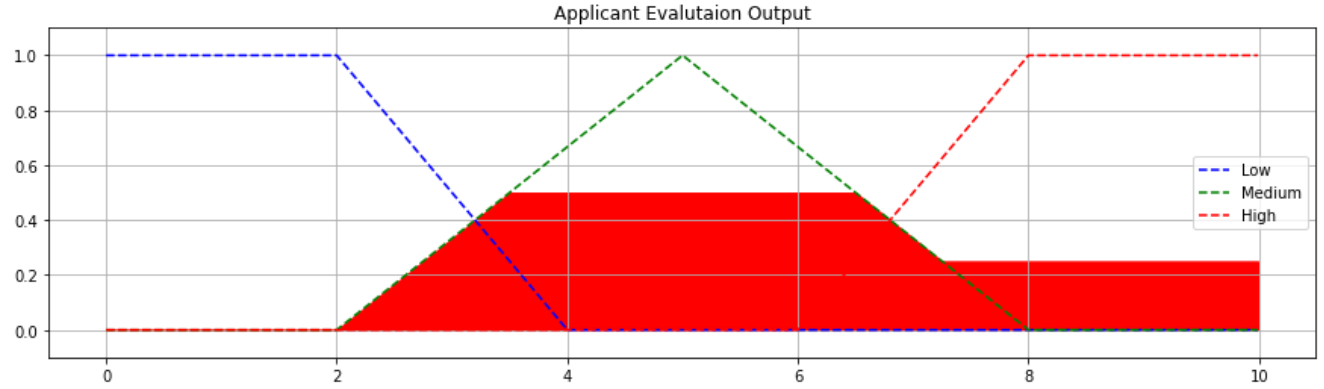



```
def apply_applicant_rules(assets, income, verbose=0):
    # person asset
    p_asset_level_low = fuzz.interp_membership(x_person_asset, p_asset_low, assets)
    p_asset_level_medium = fuzz.interp_membership(x_person_asset, p_asset_medium, assets)
    p_asset_level_high = fuzz.interp_membership(x_person_asset, p_asset_high, assets)

    # person income
```

```
    p_income_level_low = fuzz.interp_membership(x_person_income, p_income_low, income)
    p_income_level_medium = fuzz.interp_membership(x_person_income, p_income_medium, income)
    p_income_level_high = fuzz.interp_membership(x_person_income, p_income_high, income)
    p_income_level_very_high = fuzz.interp_membership(x_person_income, p_income_very_high, income)


    # 1. If (Asset is Low) and (Income is Low) then (Applicant is Low)
    applicant_act_low1 = and_rule(p_asset_level_low, p_income_level_low, applicant_low)
    # 2. If (Asset is Low) and (Income is Medium) then (Applicant is Low)
    applicant_act_low2 = and_rule(p_asset_level_low, p_income_level_medium, applicant_low)
    # 3. If (Asset is Low) and (Income is High) then (Applicant is Medium)
    applicant_act_medium1 = and_rule(p_asset_level_low, p_income_level_high, applicant_medium)
    # 4. If (Asset is Low) and (Income is Very_high) then (Applicant is High)
    applicant_act_high1 = and_rule(p_asset_level_low, p_income_level_very_high, applicant_high)
    # 5. If (Asset is Medium) and (Income is Low) then (Applicant is Low)
    applicant_act_low3 = and_rule(p_asset_level_medium, p_income_level_low, applicant_low)
    # 6. If (Asset is Medium) and (Income is Medium) then (Applicant is Medium)
    applicant_act_medium2 = and_rule(p_asset_level_medium, p_income_level_medium, applicant_medium)
    # 7. If (Asset is Medium) and (Income is High) then (Applicant is High)
    applicant_act_high2 = and_rule(p_asset_level_medium, p_income_level_high, applicant_high)
    # 8. If (Asset is Medium) and (Income is Very_high) then (Applicant is High)
    applicant_act_high3 = and_rule(p_asset_level_medium, p_income_level_very_high, applicant_high)
    # 9. If (Asset is High) and (Income is Low) then (Applicant is Medium)
    applicant_act_medium3 = and_rule(p_asset_level_high, p_income_level_low, applicant_medium)
    # 10. If (Asset is High) and (Income is Medium) then (Applicant is Medium)
    applicant_act_medium4 = and_rule(p_asset_level_high, p_income_level_medium, applicant_medium)
    # 11. If (Asset is High) and (Income is High) then (Applicant is High)
    applicant_act_high4 = and_rule(p_asset_level_high, p_income_level_high, applicant_high)
    # 12. If (Asset is High) and (Income is Very_high) then (Applicant is High)
    applicant_act_high5 = and_rule(p_asset_level_high, p_income_level_very_high, applicant_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_medium3)
    applicant_act_medium = np.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_high)

    # if we want to see the graph of the output
    if verbose == 1:
        plt.rcParams["figure.figsize"] = 15, 4
        plt.plot(x_applicant, applicant_low, 'b', linestyle='--', linewidth=1.5, label='Low')
        plt.plot(x_applicant, applicant_medium, 'g', linestyle='--', linewidth=1.5, label='Medium')
        plt.plot(x_applicant, applicant_high, 'r', linestyle='--', linewidth=1.5, label='High'),plt.title("Ap
        plt.legend()

        plt.fill_between(x_applicant, applicant, color='r')
        plt.ylim(-0.1, 1.1)
        plt.grid(True)
        plt.show()

    return applicant
```

Example Output For Rule Base 2

```
## triying the function with example
a_eval = apply_applicant_rules(550, 45, verbose=1)
```

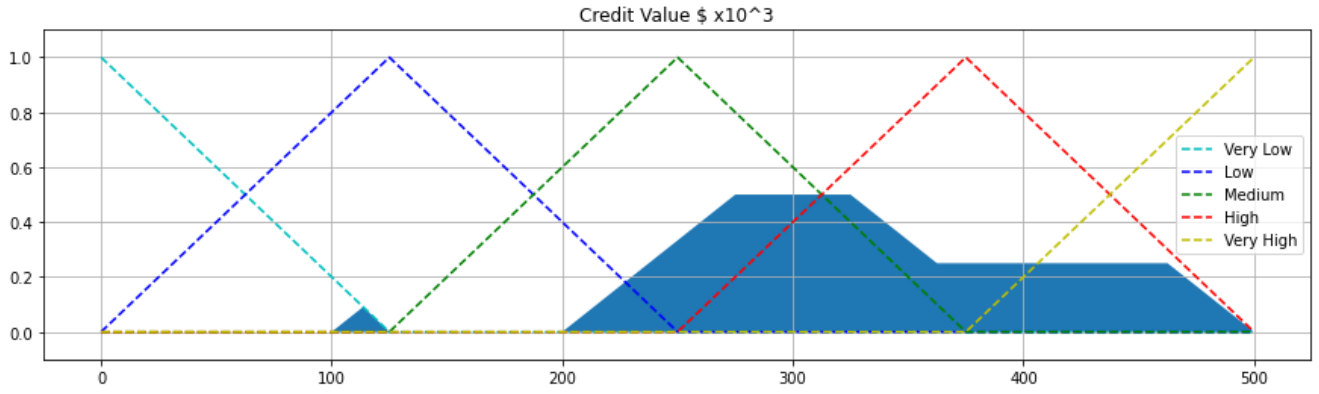



```
def apply_credit_rules(house, income, interest, applicant):
    # house
    house_level_very_low = np.fmin(house, house_low)
    house_level_low = np.fmin(house, house_low)
    house_level_medium = np.fmin(house, house_medium)
    house_level_high = np.fmin(house, house_high)
    house_level_very_high = np.fmin(house, house_very_high)

    # person income
    p_income_level_low = fuzz.interp_membership(x_person_income, p_income_low, income)
    p_income_level_medium = fuzz.interp_membership(x_person_income, p_income_medium, income)
    p_income_level_high = fuzz.interp_membership(x_person_income, p_income_high, income)
    p_income_level_very_high = fuzz.interp_membership(x_person_income, p_income_very_high, income)

    # interest
    b_interest_level_low = fuzz.interp_membership(x_interest, b_interest_low, interest)
    b_interest_level_medium = fuzz.interp_membership(x_interest, b_interest_medium, interest)
    b_interest_level_high = fuzz.interp_membership(x_interest, b_interest_high, interest)

    # applicant
    applicant_level_low = np.fmin(applicant, applicant_low)
    applicant_level_medium = np.fmin(applicant, applicant_medium)
    applicant_level_high = np.fmin(applicant, applicant_high)

    # 1. If (Income is Low) and (Interest is Medium) then (Credit is Very_low)
    credit_act_very_low1 = and_rule(p_income_level_low, b_interest_level_medium, credit_very_low)
    # 2. If (Income is Low) and (Interest is High) then (Credit is Very_low)
    credit_act_very_low2 = and_rule(p_income_level_low, b_interest_level_high, credit_very_low)
    # 3. If (Income is Medium) and (Interest is High) then (Credit is Low)
    credit_act_low1 = and_rule(p_income_level_medium, b_interest_level_high, credit_low)
    # 4. If (Applicant is Low) then (Credit is Very_low)
    credit_act_very_low3 = np.fmin(applicant_level_low, credit_very_low)
    # 5. If (House is Very_low) then (Credit is Very_low)
    credit_act_very_low4 = np.fmin(house_level_very_low, credit_very_low)
    # 6. If (Applicant is Medium) and (House is Very_low) then (Credit is Low)
    credit_act_low2 = and_rule(applicant_level_medium, house_level_very_low, credit_low)
    # 7. If (Applicant is Medium) and (House is Low) then (Credit is Low)
    credit_act_low3 = and_rule(applicant_level_medium, house_level_low, credit_low)
    # 8. If (Applicant is Medium) and (House is Medium) then (Credit is Medium)
    credit_act_medium1 = and_rule(applicant_level_medium, house_level_medium, credit_medium)
    # 9. If (Applicant is Medium) and (House is High) then (Credit is High)
    credit_act_high1 = and_rule(applicant_level_medium, house_level_high, credit_high)
    # 10. If (Applicant is Medium) and (House is Very_high) then (Credit is High)
    credit_act_high2 = and_rule(applicant_level_medium, house_level_very_high, credit_high)
    # 11. If (Applicant is High) and (House is Very_low) then (Credit is Low)
    credit_act_low4 = and_rule(applicant_level_high, house_level_very_low, credit_low)
    # 12. If (Applicant is High) and (House is Low) then (Credit is Medium)
    credit_act_medium2 = and_rule(applicant_level_high, house_level_low, credit_medium)
    # 13. If (Applicant is High) and (House is Medium) then (Credit is High)
    credit_act_high3 = and_rule(applicant_level_high, house_level_medium, credit_high)
    # 14. If (Applicant is High) and (House is High) then (Credit is High)
    credit_act_high4 = and_rule(applicant_level_high, house_level_high, credit_high)
```

```
    # 15. If (Applicant is High) and (House is Very_high) then (Credit is Very_high)
    credit_act_very_high = and_rule(applicant_level_high, house_level_very_high, credit_very_high)


    step = or_rule(credit_act_very_low1, credit_act_very_low2, credit_act_very_low3)
    credit_act_very_low = np.fmax(step, credit_act_very_low4)

    step = or_rule(credit_act_low1, credit_act_low2, credit_act_low3)
    credit_act_low = np.fmax(step, credit_act_low4)

    credit_act_medium = np.fmax(credit_act_medium1, credit_act_medium2)

    step = or_rule(credit_act_high1, credit_act_high2, credit_act_high3)
    credit_act_high = np.fmax(step, credit_act_high4)
    # credit_act_very_high there is just one

    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
```

Example Output For Base 3

```
# triying the function with example
credit_eval = apply_credit_rules(h_eval, 45, 4, a_eval)

plt.rcParams["figure.figsize"] = 15, 4
plt.plot(x_credit, credit_very_low, 'c', linestyle='--', linewidth=1.5, label='Very Low')
plt.plot(x_credit, credit_low, 'b', linestyle='--', linewidth=1.5, label='Low')
plt.plot(x_credit, credit_medium, 'g', linestyle='--', linewidth=1.5, label='Medium')
plt.plot(x_credit, credit_high, 'r', linestyle='--', linewidth=1.5, label='High')
plt.plot(x_credit, credit_very_high, 'y', linestyle='--', linewidth=1.5, label='Very High'),plt.title("Credit
plt.legend()

plt.fill_between(x_credit, credit_eval)
plt.ylim(-0.1, 1.1)
plt.grid(True)
plt.show()
```

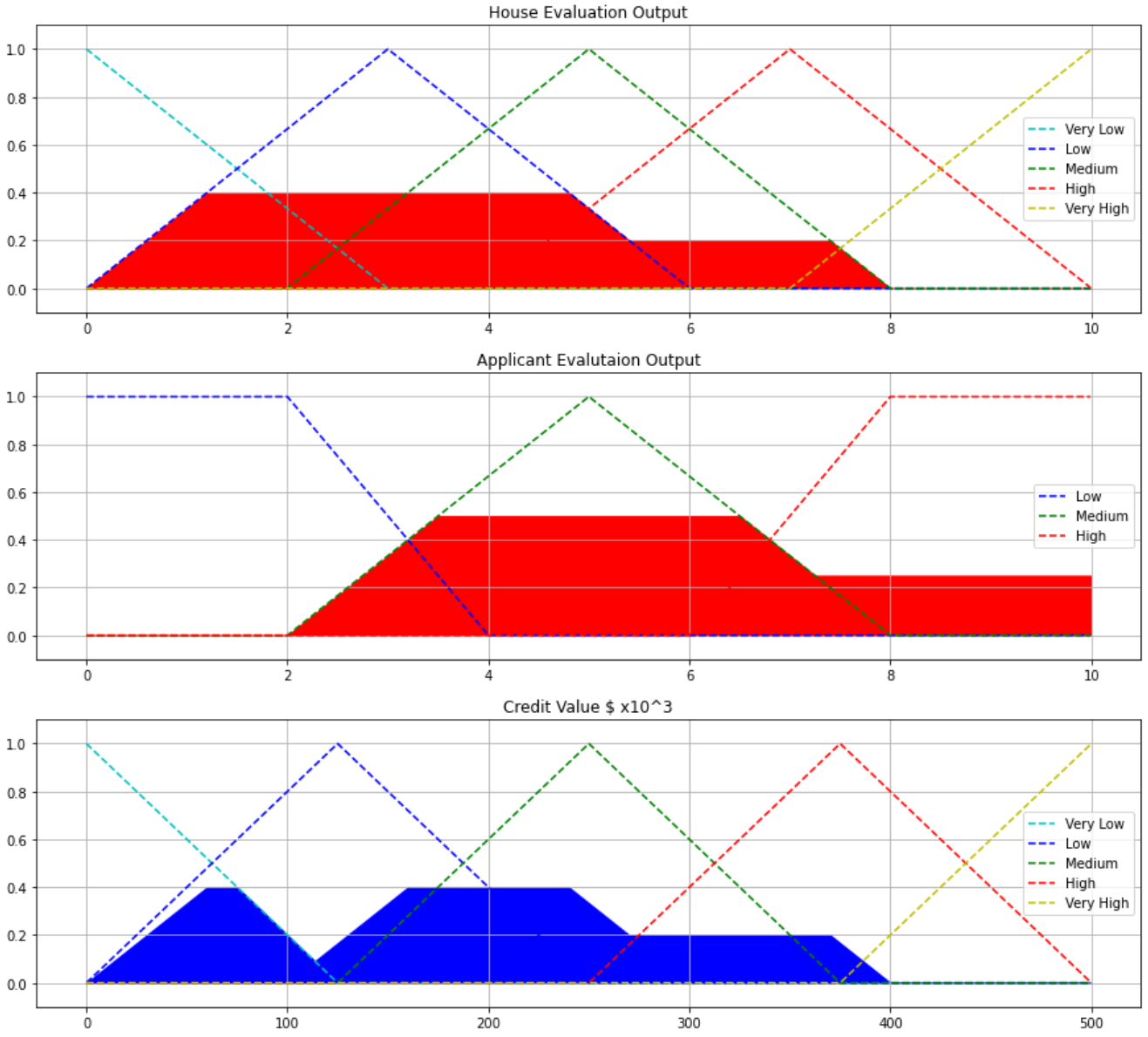


Calling the ruleing functions sequentialy

```
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
```

Example Output After All Bases

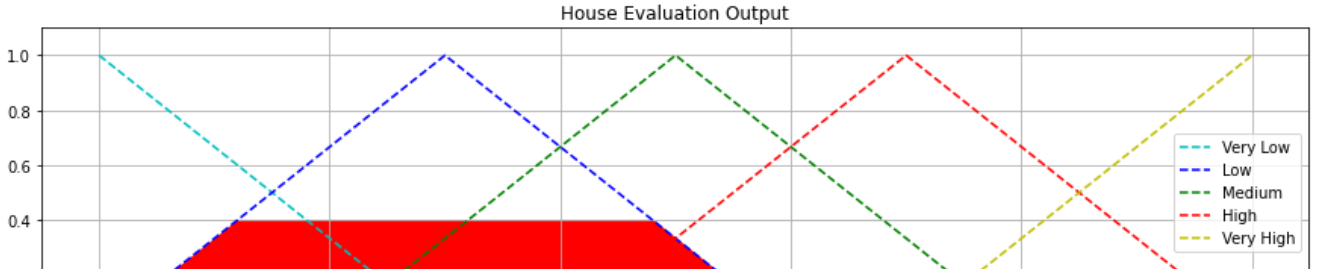
```
credit = apply_all_rules(150, 3, 550, 45, 4, verbose=1)

plt.rcParams["figure.figsize"] = 15, 4
plt.plot(x_credit, credit_very_low, 'c', linestyle='--', linewidth=1.5, label='Very Low')
plt.plot(x_credit, credit_low, 'b', linestyle='--', linewidth=1.5, label='Low')
plt.plot(x_credit, credit_medium, 'g', linestyle='--', linewidth=1.5, label='Medium')
plt.plot(x_credit, credit_high, 'r', linestyle='--', linewidth=1.5, label='High')
plt.plot(x_credit, credit_very_high, 'y', linestyle='--', linewidth=1.5, label='Very High'),plt.title("Credit
plt.legend()

plt.fill_between(x_credit, credit, color='b')
plt.ylim(-0.1, 1.1)
plt.grid(True)
plt.show()
```



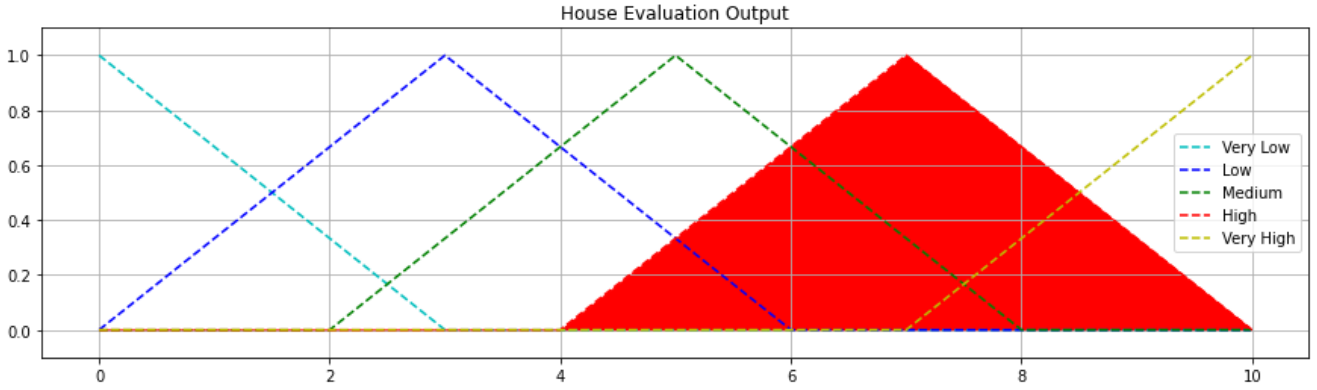
Making a Decision

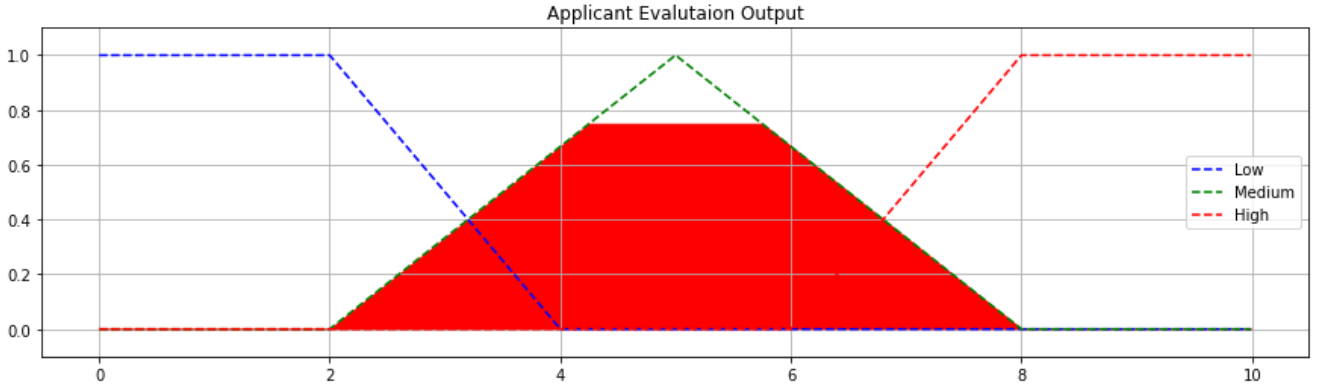
- After all the rules applied, we defuzify the output of the rules with mean of maximum and we generate a single value as a decision of the system

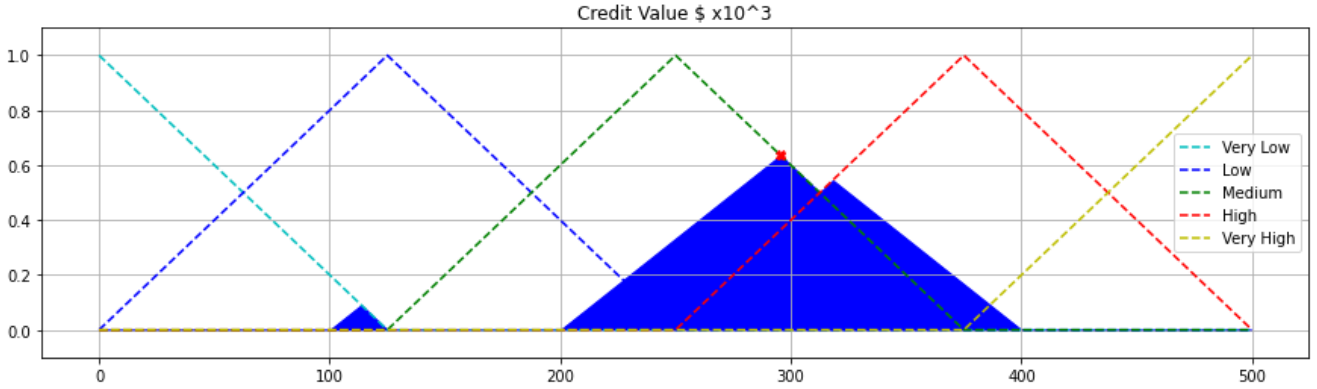
```
def make_decision(market_value, location, assets, income, interest, verbose=0):
    credit = apply_all_rules(market_value, location, assets, income, interest, verbose)
    # defuzzification with mean of maximum
    defuzz_credit = fuzz.defuzz(x_credit, credit, 'mom')
    max_n = np.max(credit)

    if (verbose == 1):
        plt.rcParams["figure.figsize"] = 15, 4
        plt.plot(x_credit, credit_very_low, 'c', linestyle='--', linewidth=1.5, label='Very Low')
        plt.plot(x_credit, credit_low, 'b', linestyle='--', linewidth=1.5, label='Low')
        plt.plot(x_credit, credit_medium, 'g', linestyle='--', linewidth=1.5, label='Medium')
        plt.plot(x_credit, credit_high, 'r', linestyle='--', linewidth=1.5, label='High')
        plt.plot(x_credit, credit_very_high, 'y', linestyle='--', linewidth=1.5, label='Very High'),plt.title
        plt.legend()

        plt.fill_between(x_credit, credit, color='b')
        plt.ylim(-0.1, 1.1)
        plt.grid(True)

        plt.plot(defuzz_credit, max_n, 'X', color='r')
        plt.show()

    print("Output: ", defuzz_credit, "x10^3 $")
    return defuzz_credit


credit_decision = make_decision(150, 3, 550, 45, 4, verbose=1)
```




```
credit_decision = make_decision(600, 6, 500, 40, 5, verbose=1)
```








```
Output:  295.5 x10^3 $
```