Task -3

Objective: - To design a Fuzzy Inference System (Mamdani Based)

Idea: -To take highest and average marks as input and generate a suitable cutoff.

Code:-

```
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt

# Generate universe variables
#   * Highest and average on subjective ranges [50,100] and [20,90]
#   * Cutoff has a range of [60 95]
# Input Variables , highest,average and output is cutoff
x_highest = np.arange(50, 101, 1)
x_average = np.arange(20, 91, 1)
x_cutoff  = np.arange(60, 96, 1)

# Generate fuzzy membership functions
highest_lo = fuzz.trimf(x_highest, [60, 70, 80])
highest_md = fuzz.trimf(x_highest, [70, 80, 90])
highest_hi = fuzz.trimf(x_highest, [80, 90, 100])
average_lo = fuzz.trimf(x_average, [40, 50, 60])
average_md = fuzz.trimf(x_average, [50, 60, 70])
average_hi = fuzz.trimf(x_average, [60, 70, 80])
cutoff_lo = fuzz.trimf(x_cutoff, [50, 60, 70])
cutoff_md = fuzz.trimf(x_cutoff, [60, 70, 80])
```

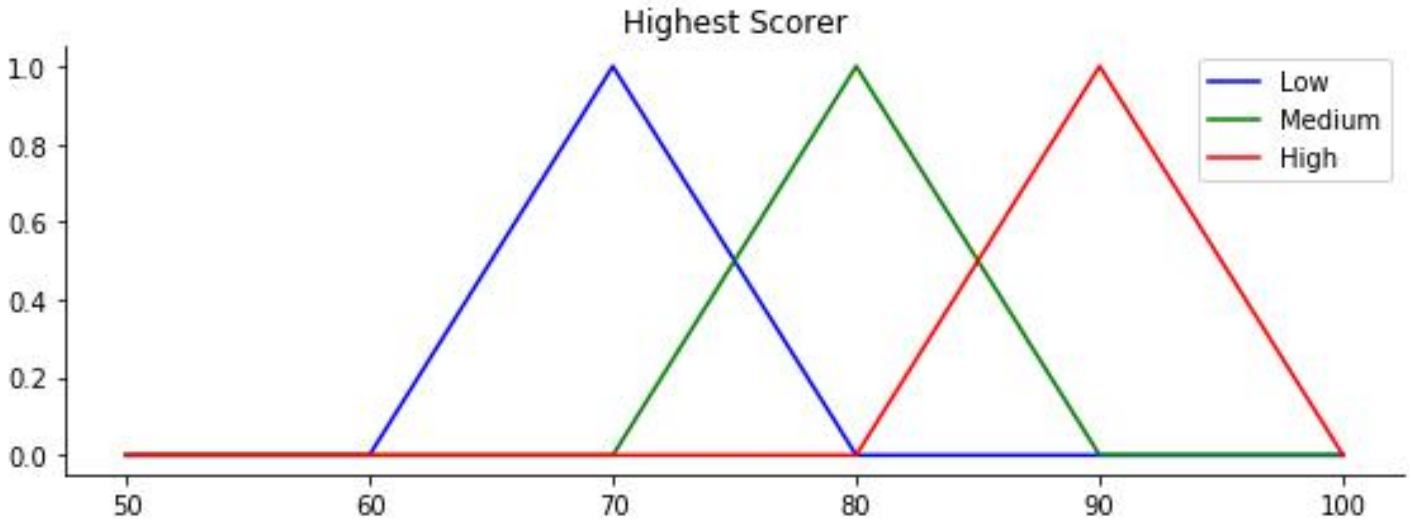
```
cutoff_hi = fuzz.trimf(x_cutoff, [70, 80, 95])

# Visualizing membership functions
fig, (ax0, ax1, ax2) = plt.subplots(nrows=3, figsize=(8, 9))

ax0.plot(x_highest, highest_lo, 'b', linewidth=1.5, label='Low')
ax0.plot(x_highest, highest_md, 'g', linewidth=1.5, label='Medium')
ax0.plot(x_highest, highest_hi, 'r', linewidth=1.5, label='High')
ax0.set_title('Highest Scorer')
ax0.legend()

ax1.plot(x_average, average_lo, 'b', linewidth=1.5, label='Low')
ax1.plot(x_average, average_md, 'g', linewidth=1.5, label='Medium')
ax1.plot(x_average, average_hi, 'r', linewidth=1.5, label='High')
ax1.set_title('Average Marks')
ax1.legend()

ax2.plot(x_cutoff, cutoff_lo, 'b', linewidth=1.5, label='Low')
ax2.plot(x_cutoff, cutoff_md, 'g', linewidth=1.5, label='Medium')
ax2.plot(x_cutoff, cutoff_hi, 'r', linewidth=1.5, label='High')
ax2.set_title('Cutoff ')
ax2.legend()



for ax in (ax0, ax1, ax2):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
```

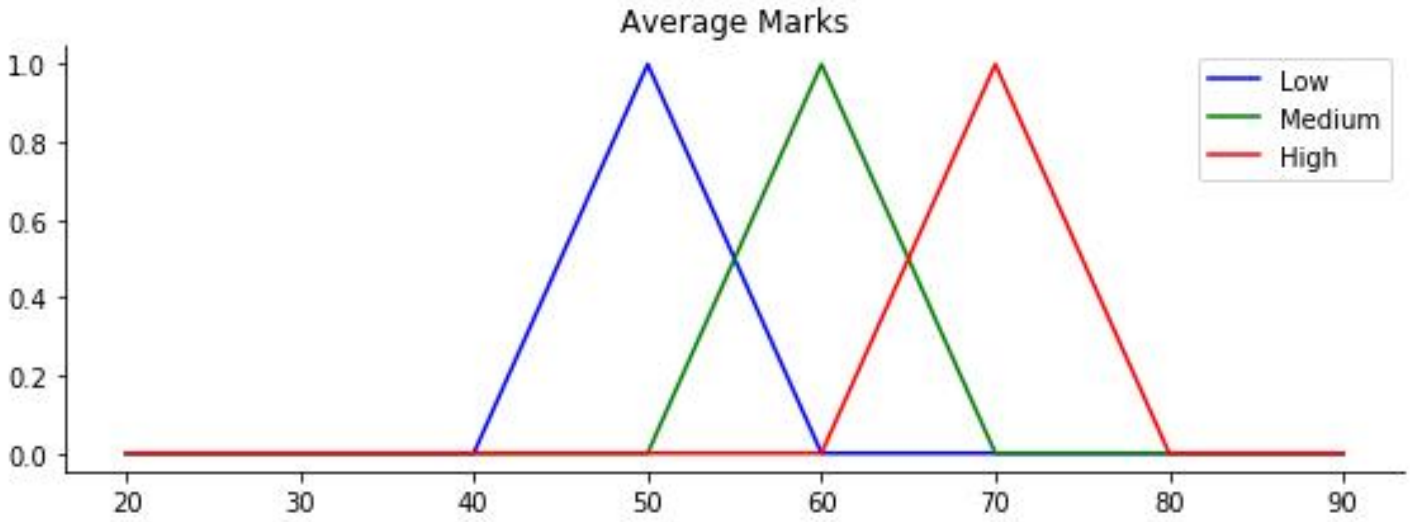
```
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()

plt.tight_layout()

# Lets suppose highest=97 and average=68
qual_level_lo = fuzz.interp_membership(x_highest, highest_lo, 97)
qual_level_md = fuzz.interp_membership(x_highest, highest_md, 97)
qual_level_hi = fuzz.interp_membership(x_highest, highest_hi, 97)

serv_level_lo = fuzz.interp_membership(x_average, average_lo, 68)
serv_level_md = fuzz.interp_membership(x_average, average_md, 68)
serv_level_hi = fuzz.interp_membership(x_average, average_hi, 68)

# Defining rules , low highest marks and low average.
# The OR operator means we take the maximum of these two.
active_rule1 = np.fmax(qual_level_lo, serv_level_lo)


# membership function with `np.fmin`
tip_activation_lo = np.fmin(active_rule1, cutoff_lo)

# For rule 2 we connect acceptable service to medium cutoff
tip_activation_md = np.fmin(serv_level_md, cutoff_md)

# For rule 3 we connect high highest marks OR high average with high cutoff
active_rule3 = np.fmax(qual_level_hi, serv_level_hi)
```

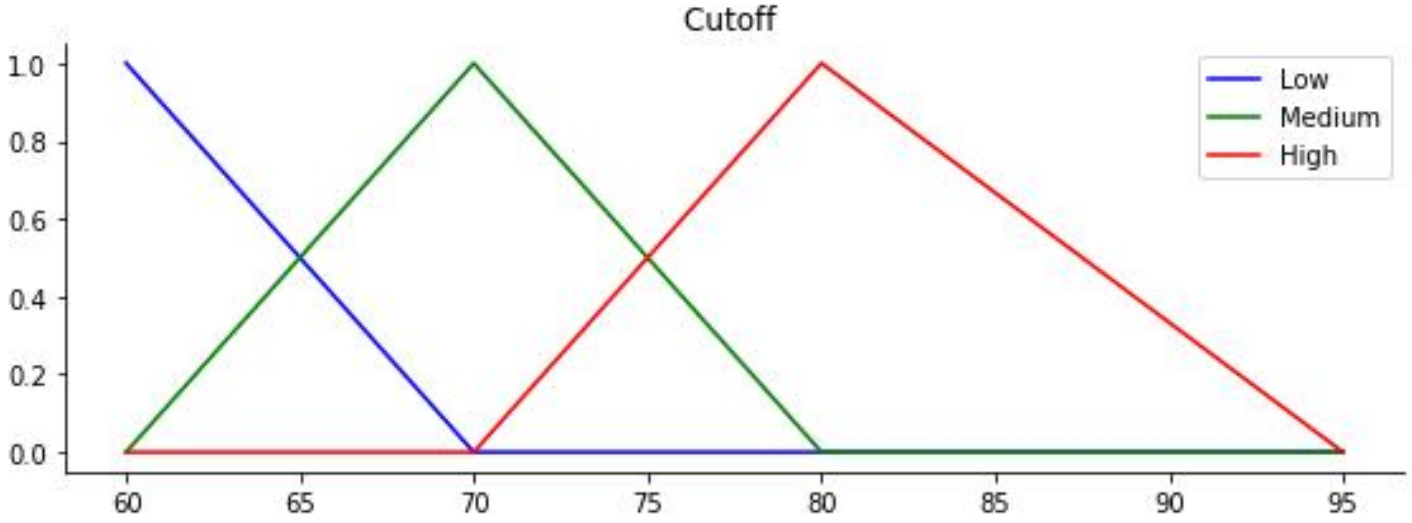
```
tip_activation_hi = np.fmin(active_rule3, cutoff_hi)
tip0 = np.zeros_like(x_cutoff)

# Visualize this
fig, ax0 = plt.subplots(figsize=(8, 3))

ax0.fill_between(x_cutoff, tip0, tip_activation_lo, facecolor='b', alpha=0.7)
ax0.plot(x_cutoff, cutoff_lo, 'b', linewidth=0.5, linestyle='--', )
ax0.fill_between(x_cutoff, tip0, tip_activation_md, facecolor='g', alpha=0.7)
ax0.plot(x_cutoff, cutoff_md, 'g', linewidth=0.5, linestyle='--')
ax0.fill_between(x_cutoff, tip0, tip_activation_hi, facecolor='r', alpha=0.7)
ax0.plot(x_cutoff, cutoff_hi, 'r', linewidth=0.5, linestyle='--')
ax0.set_title('Output membership activity')



for ax in (ax0,):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()

plt.tight_layout()

# Aggregate all three output membership functions together
aggregated = np.fmax(tip_activation_lo,
                     np.fmax(tip_activation_md, tip_activation_hi))
```

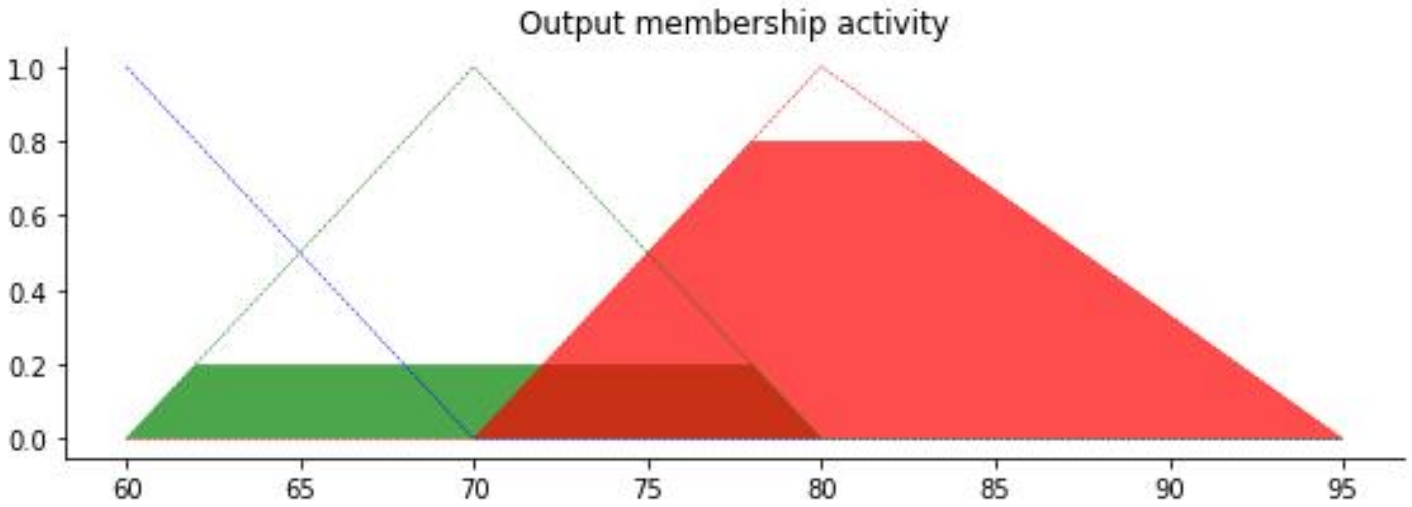
```
# Deffuzification using centroid method
tip = fuzz.defuzz(x_cutoff, aggregated, 'centroid')
tip_activation = fuzz.interp_membership(x_cutoff, aggregated, tip)

# Visualize this
fig, ax0 = plt.subplots(figsize=(8, 3))

ax0.plot(x_cutoff, cutoff_lo, 'b', linewidth=0.5, linestyle='--', )
ax0.plot(x_cutoff, cutoff_md, 'g', linewidth=0.5, linestyle='--')
ax0.plot(x_cutoff, cutoff_hi, 'r', linewidth=0.5, linestyle='--')
ax0.fill_between(x_cutoff, tip0, aggregated, facecolor='Orange', alpha=0.7)
ax0.plot([tip, tip], [0, tip_activation], 'k', linewidth=1.5, alpha=0.9)
ax0.set_title('Aggregated membership and result (line)')


for ax in (ax0,):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()

plt.tight_layout()
```

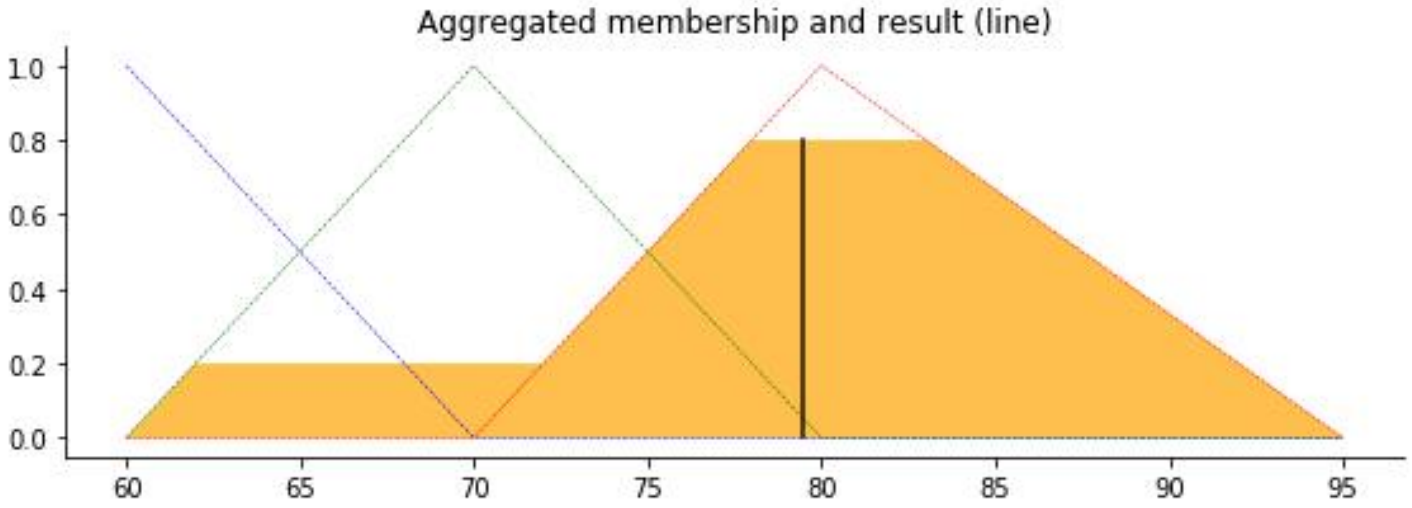
OUTPUT:-


Highest Scorer
Low
Medium
High
1.0
0.8
0.6
0.4
0.2
0.0
50
60
70
80
90
100


Average Marks
Low
Medium
High
1.0
0.8
0.6
0.4
0.2
0.0
20
30
40
50
60
70
80
90


Cutoff
Low
Medium
High
1.0
0.8
0.6
0.4
0.2
0.0
60
65
70
75
80
85
90
95


Output membership activity
1.0
0.8
0.6
0.4
0.2
0.0
60
65
70
75
80
85
90
95


Aggregated membership and result (line)
1.0
0.8
0.6
0.4
0.2
0.0
60
65
70
75
80
85
90
95