Tipping based on Food and Service Quality

Consider three simple rules:

1. If the food is bad OR the service is poor, then the tip will be low
2. If the service is acceptable, then the tip will be medium
3. If the food is great OR the service is amazing, then the tip will be high.

Most people would agree on these rules, but the rules are fuzzy. Mapping the imprecise rules into a defined, actionable tip is a challenge. This is the kind of task at which fuzzy logic excels.

Required Libraries

https://pythonhosted.org/scikit-fuzzy/auto_examples/plot_tipping_problem.html

https://pythonhosted.org/scikit-fuzzy/auto_examples/plot_tipping_problem_newapi.html

Input variables

Two inputs

- food quality
- service

Output variable

the output variable is the tip amount.

Import Packages

```
!py -m pip install scikit-fuzzy
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt
```

```
Requirement already satisfied: scikit-fuzzy in c:\users\sehre\appdata\local\programs\python\python311\lib\site-packages (0.4
Requirement already satisfied: numpy>=1.6.0 in c:\users\sehre\appdata\local\programs\python\python311\lib\site-packages (fro
Requirement already satisfied: scipy>=0.9.0 in c:\users\sehre\appdata\local\programs\python\python311\lib\site-packages (fro
Requirement already satisfied: networkx>=1.9.0 in c:\users\sehre\appdata\local\programs\python\python311\lib\site-packages (

[notice] A new release of pip is available: 23.3 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Generate universe variables

Remember, all possible values that a variable can take are referred to as Universe of Discourse, in the context of linguistic variables.

```
#   * Quality and service on subjective ranges [0, 10] -> using .5 as step
#   * Tip has a range of [0, 25] in units of percentage points
x_qual = np.arange(0, 10.5, .5)
x_serv = np.arange(0, 10.5, .5)
x_tip  = np.arange(0, 26, 1)
```

```
print(x_qual)
print(x_serv)
print(x_tip)
```

```
[ 0.   0.5  1.   1.5  2.   2.5  3.   3.5  4.   4.5  5.   5.5  6.   6.5
  7.   7.5  8.   8.5  9.   9.5 10. ]
[ 0.   0.5  1.   1.5  2.   2.5  3.   3.5  4.   4.5  5.   5.5  6.   6.5
```

```
  7.   7.5  8.   8.5  9.   9.5 10. ]
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25]
```

Generate fuzzy membership functions

Triangular membership function (trinmf) is used for fuzzification of the variables

- food quality
  - low
  - medium
  - high
- service
  - low
  - medium
  - high
- tip amount
  - low
  - medium
  - high
- Triangular membership function Formula

$$f(x;a,b,c) = \begin{cases} 0, & x \le a \\ \frac{x-a}{b-a}, & a \le x \le b \\ \frac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases}$$

- [abc]
- $a \leq b \leq c$
- zero for $x \leq a$ and $x \geq c$
- increases a to b
- decreases b to c

```
# Generate fuzzy membership functions
# food quality
qual_lo = fuzz.trimf(x_qual, abc=[0, 0, 5])
qual_md = fuzz.trimf(x_qual, abc=[0, 5, 10])
qual_hi = fuzz.trimf(x_qual, abc=[5, 10, 10])
# service quality
serv_lo = fuzz.trimf(x_serv, abc=[0, 0, 5])
serv_md = fuzz.trimf(x_serv, abc=[0, 5, 10])
serv_hi = fuzz.trimf(x_serv, abc=[5, 10, 10])
# tip
tip_lo = fuzz.trimf(x_tip, abc=[0, 0, 13])
tip_md = fuzz.trimf(x_tip, abc=[0, 13, 25])
tip_hi = fuzz.trimf(x_tip, abc=[13, 25, 25])
```

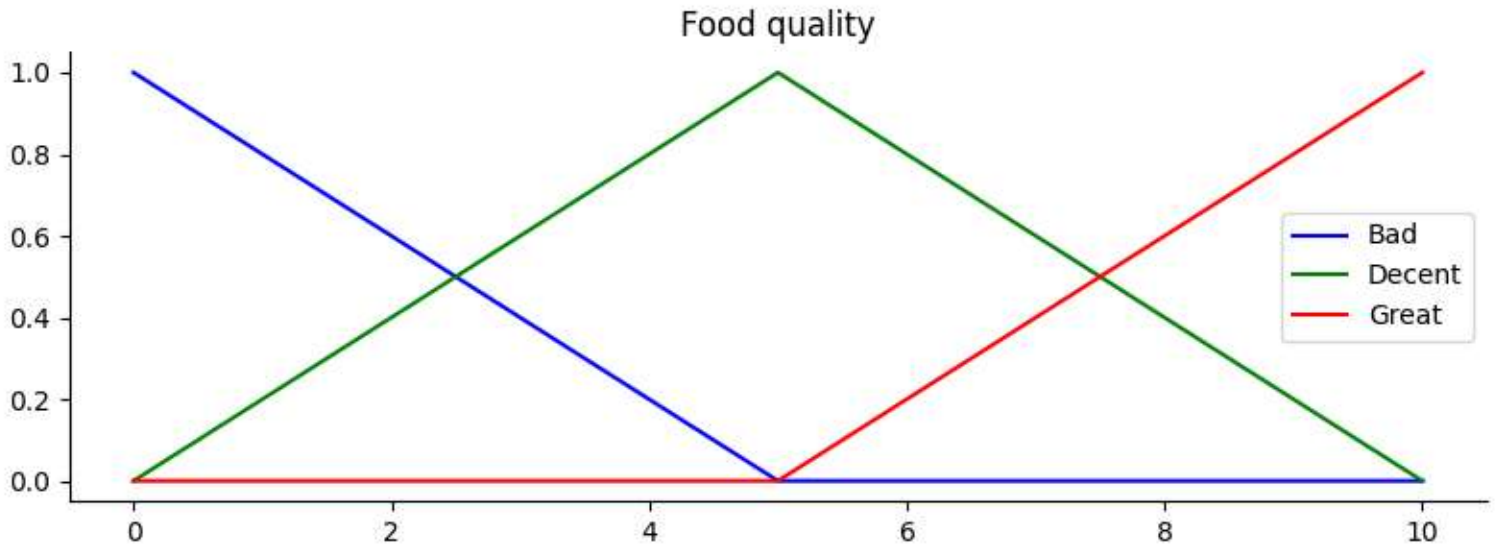
Have a look on the food quality membership functions for low, medium and high

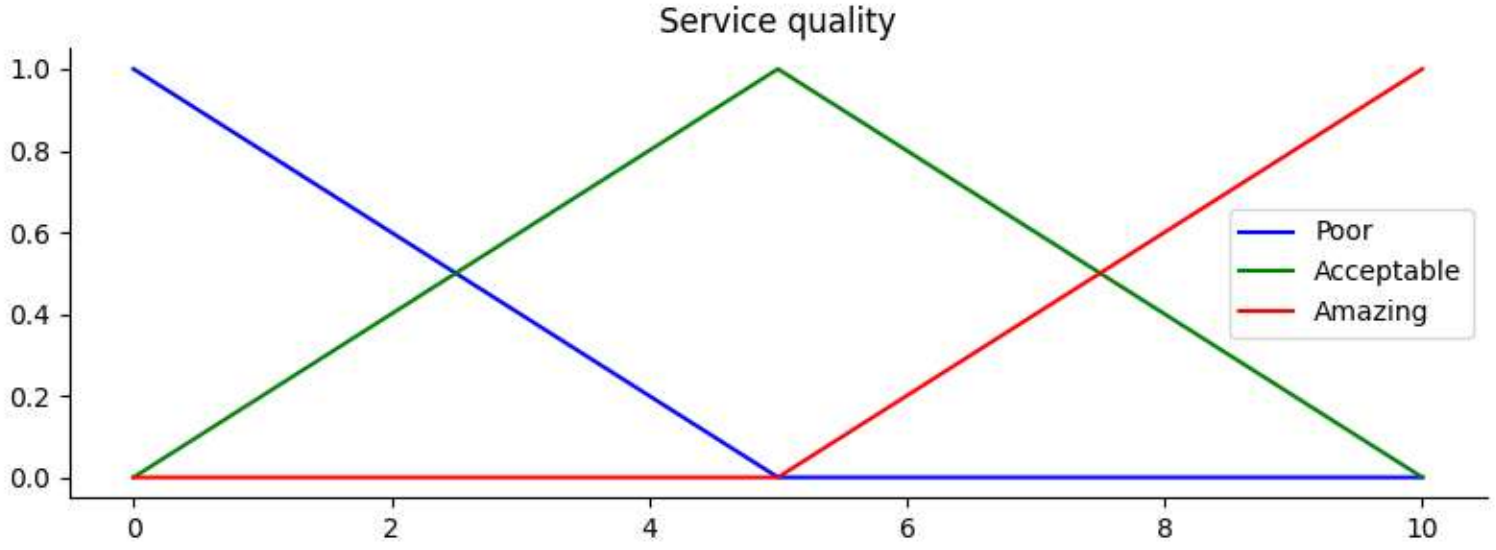
```
print('x_qual',x_qual)
print('qual_lo', qual_lo)
print('qual_md',qual_md)
print('qual_hi',qual_hi)
```

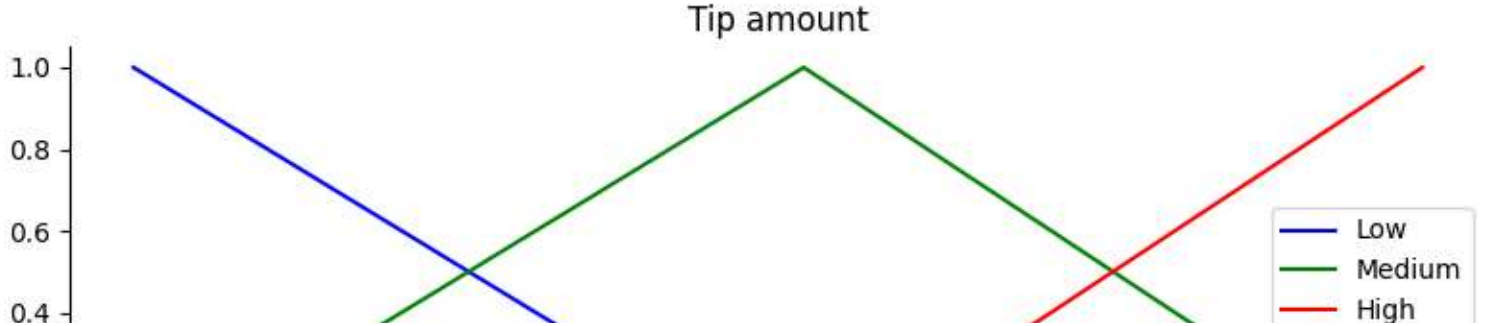
```
x_qual [ 0.   0.5  1.   1.5  2.   2.5  3.   3.5  4.   4.5  5.   5.5  6.   6.5
  7.   7.5  8.   8.5  9.   9.5 10. ]
qual_lo [1.  0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0. ]
qual_md [0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  0.9 0.8 0.7 0.6 0.5 0.4 0.3
 0.2 0.1 0. ]
qual_hi [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7
 0.8 0.9 1. ]
```

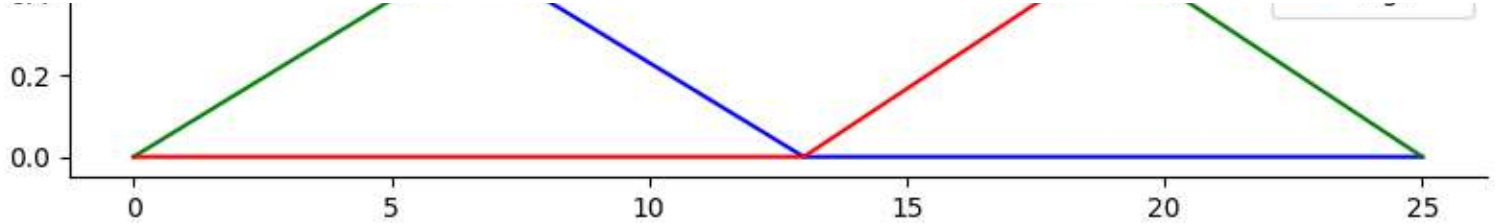
Visualize these universes and membership functions

```
# Visualize these universes and membership functions
fig, (ax0, ax1, ax2) = plt.subplots(nrows=3, figsize=(8, 9))

ax0.plot(x_qual, qual_lo, 'b', linewidth=1.5, label='Bad')
ax0.plot(x_qual, qual_md, 'g', linewidth=1.5, label='Decent')
ax0.plot(x_qual, qual_hi, 'r', linewidth=1.5, label='Great')
ax0.set_title('Food quality')
ax0.legend()

ax1.plot(x_serv, serv_lo, 'b', linewidth=1.5, label='Poor')
ax1.plot(x_serv, serv_md, 'g', linewidth=1.5, label='Acceptable')
ax1.plot(x_serv, serv_hi, 'r', linewidth=1.5, label='Amazing')
ax1.set_title('Service quality')
ax1.legend()

ax2.plot(x_tip, tip_lo, 'b', linewidth=1.5, label='Low')
ax2.plot(x_tip, tip_md, 'g', linewidth=1.5, label='Medium')
ax2.plot(x_tip, tip_hi, 'r', linewidth=1.5, label='High')
ax2.set_title('Tip amount')
ax2.legend()

# Turn off top/right axes
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()
```


Food quality
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
Bad
Decent
Great


Service quality
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
Poor
Acceptable
Amazing


Tip amount
1.0
0.8
0.6
0.4
Low
Medium
High



Fuzzy Rule based Tipping

Now, to make these triangles useful, we define the fuzzy relationship between input and output variables. For the purposes of our example, consider three simple rules:

1. If the food is bad OR the service is poor, then the tip will be low
2. If the service is acceptable, then the tip will be medium
3. If the food is great OR the service is amazing, then the tip will be high.

Most people would agree on these rules, but the rules are fuzzy. Mapping the imprecise rules into a defined, actionable tip is a challenge. This is the kind of task at which fuzzy logic excels.

Question 1

What would the tip be in the following circumstance:

- Food quality was 6.5
- Service was 9.8

Fuzzification: interp_membership function

interpolate

Find the degree of membership u(xx) for a given value of x = xx.

We need the activation of our fuzzy membership functions at these values. The exact values 6.5 and 9.8 do not exist on our universes...
This is what fuzz.interp_membership exists for!

```
qual_level_lo = fuzz.interp_membership(x_qual, qual_lo, 6.5)
qual_level_md = fuzz.interp_membership(x_qual, qual_md, 6.5)
qual_level_hi = fuzz.interp_membership(x_qual, qual_hi, 6.5)
print(qual_level_lo, qual_level_md, qual_level_hi)
```

```
0.0 0.7 0.3
```

```
serv_level_lo = fuzz.interp_membership(x_serv, serv_lo, 9.8)
serv_level_md = fuzz.interp_membership(x_serv, serv_md, 9.8)
serv_level_hi = fuzz.interp_membership(x_serv, serv_hi, 9.8)
print(serv_level_lo, serv_level_md, serv_level_hi)
```

```
0.0 0.03999999999999986 0.9600000000000002
```

Rule Evaluation (Inference)

1. If the food is bad OR the service is poor, then the tip will be low

```
import math

active_rule1 = np.fmax(qual_level_lo, serv_level_lo) # fuzzy-max = Boolean OR
print('active_rule1', active_rule1)
# Now we apply this by clipping the top off the corresponding output
# membership function with `np.fmin`
print('tip_lo')
print('consequent', tip_lo.round(2))
tip_activation_lo = np.fmin(active_rule1, tip_lo)  # fuzzy-min = Boolean AND
print('Rule 1: tip_activation_lo')
print(tip_activation_lo)
```

```
active_rule1 0.0
tip_lo
consequent [1.   0.92 0.85 0.77 0.69 0.62 0.54 0.46 0.38 0.31 0.23 0.15 0.08 0.
```

```
 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]
Rule 1: tip_activation_lo
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0.]
```

2. If the service is acceptable, then the tip will be medium

```
# For rule 2 we connect acceptable service to medium tipping
tip_activation_md = np.fmin(serv_level_md, tip_md) # fuzzy-min = Boolean AND

print('antecedent (serv_level_md)')
print(serv_level_md.round(4))
print(tip_md.round(1))
print(tip_activation_md)
```

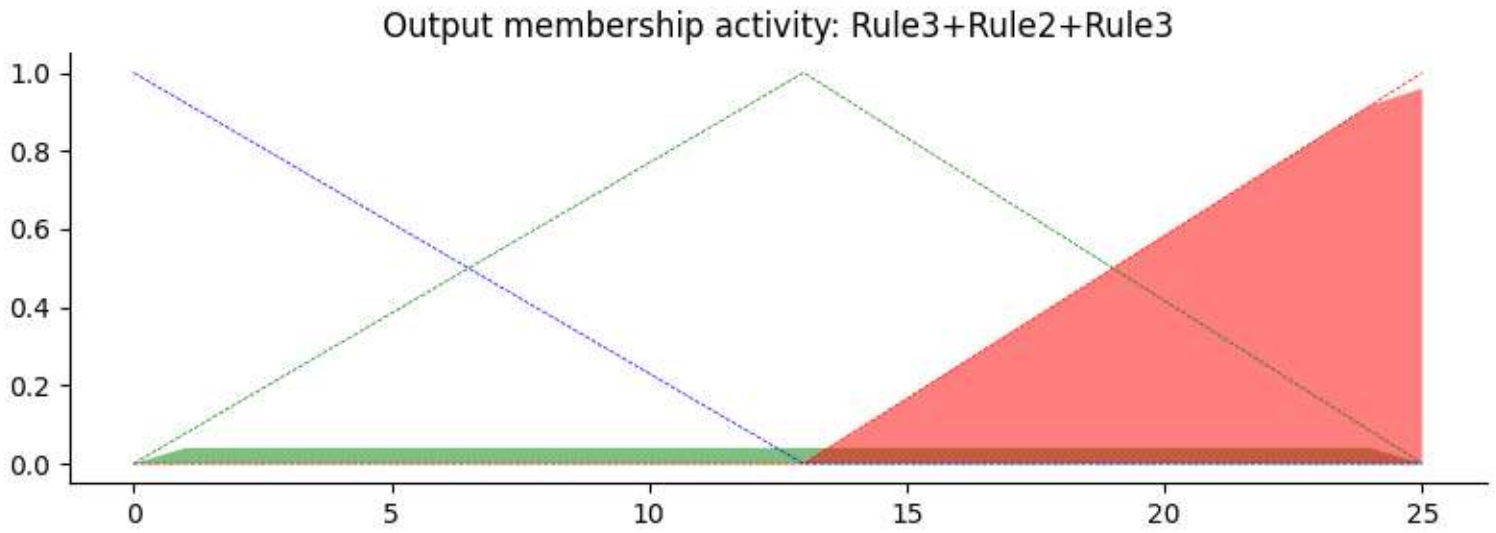
```
antecedent (serv_level_md)
0.04
[0.  0.1 0.2 0.2 0.3 0.4 0.5 0.5 0.6 0.7 0.8 0.8 0.9 1.  0.9 0.8 0.8 0.7
 0.6 0.5 0.4 0.3 0.2 0.2 0.1 0. ]
[0.   0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04
 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.  ]
```

3. If the food is great OR the service is amazing, then the tip will be high.

```
# For rule 3 we connect high service OR high food with high tipping
active_rule3 = np.fmax(qual_level_hi, serv_level_hi) # fuzzy-max = Boolean OR
print('active_rule3', active_rule3)
print(tip_hi)
tip_activation_hi = np.fmin(active_rule3, tip_hi)
print(tip_activation_hi)
```

```
active_rule3 0.9600000000000002
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.08333333 0.16666667 0.25       0.33333333
 0.41666667 0.5        0.58333333 0.66666667 0.75       0.83333333
 0.91666667 1.        ]
```

```
     [0.         0.         0.         0.         0.         0.
      0.         0.         0.         0.         0.         0.
      0.         0.         0.08333333 0.16666667 0.25       0.33333333
      0.41666667 0.5        0.58333333 0.66666667 0.75       0.83333333
      0.91666667 0.96      ]


tip0 = np.zeros_like(x_tip)
# Visualize this
fig, ax0 = plt.subplots(figsize=(8, 3))
ax0.fill_between(x=x_tip, y1=tip0, y2=tip_activation_lo, facecolor='b', alpha=0.5)
ax0.plot(x_tip, tip_lo, 'b', linewidth=0.5, linestyle='--', )
ax0.fill_between(x_tip, tip0, tip_activation_md, facecolor='g', alpha=0.5)
ax0.plot(x_tip, tip_md, 'g', linewidth=0.5, linestyle='--')
ax0.fill_between(x_tip, tip0, tip_activation_hi, facecolor='r', alpha=0.5)
ax0.plot(x_tip, tip_hi, 'r', linewidth=0.5, linestyle='--')
ax0.set_title('Output membership activity: Rule3+Rule2+Rule3')

# Turn off top/right axes
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()
```



Aggregation (Composition)

- Aggregate all three output membership functions together

```
aggregated = np.fmax(tip_activation_lo, np.fmax(tip_activation_md, tip_activation_hi))
print(aggregated)
```

```
[0.         0.04       0.04       0.04       0.04       0.04
 0.04       0.04       0.04       0.04       0.04       0.04
 0.04       0.04       0.08333333 0.16666667 0.25       0.33333333
 0.41666667 0.5        0.58333333 0.66666667 0.75       0.83333333
 0.91666667 0.96      ]
```

Defuzzification

Calculate defuzzified result using the centroid method

```
tip = fuzz.defuzz(x_tip, aggregated, 'centroid')
print(tip)
# tip = fuzz.defuzz(x_tip, aggregated, 'som')
# print(tip)
# tip = fuzz.defuzz(x_tip, aggregated, 'mom')
# print(tip)
# tip = fuzz.defuzz(x_tip, aggregated, 'lom')
# print(tip)
```

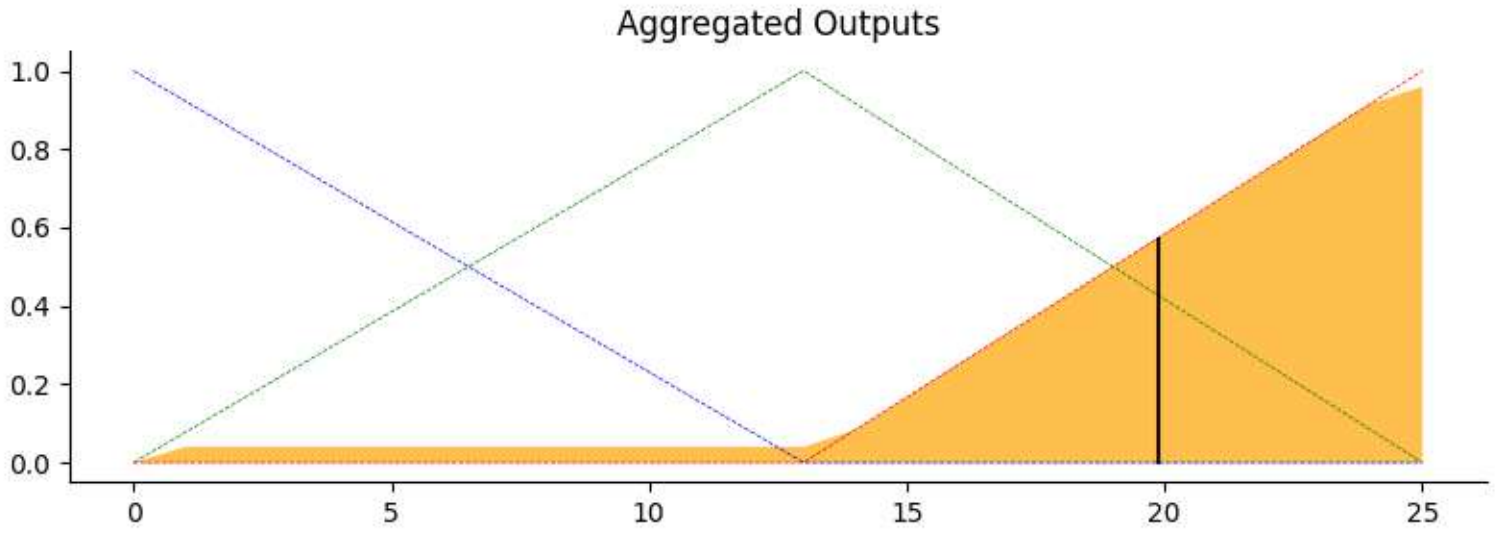
```
19.86871794871795
```

Plot the output

```
tip_activation = fuzz.interp_membership(x_tip, aggregated, tip)  # for plot
tip_activation
```

```
0.5723931623931626
```

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

ax0.plot(x_tip, tip_lo, 'b', linewidth=0.5, linestyle='--', )
ax0.plot(x_tip, tip_md, 'g', linewidth=0.5, linestyle='--')
ax0.plot(x_tip, tip_hi, 'r', linewidth=0.5, linestyle='--')
ax0.fill_between(x_tip, 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 Outputs')

# Turn off top/right axes
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()
```



Control API

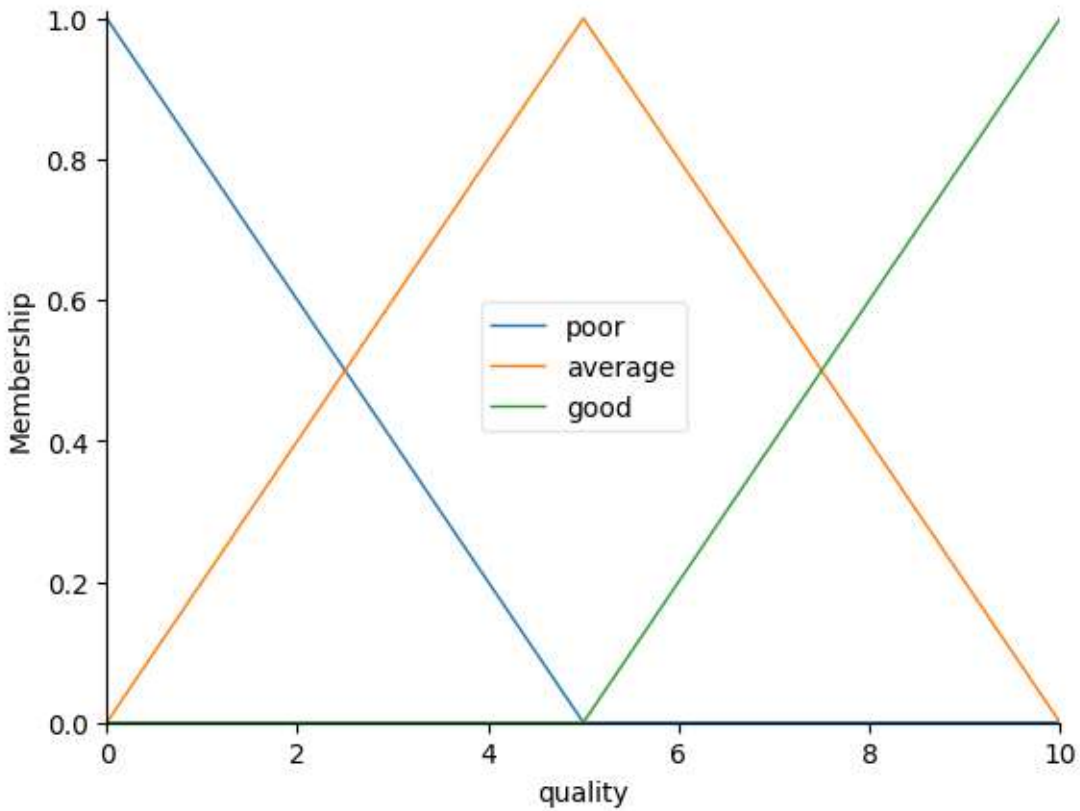
UofD

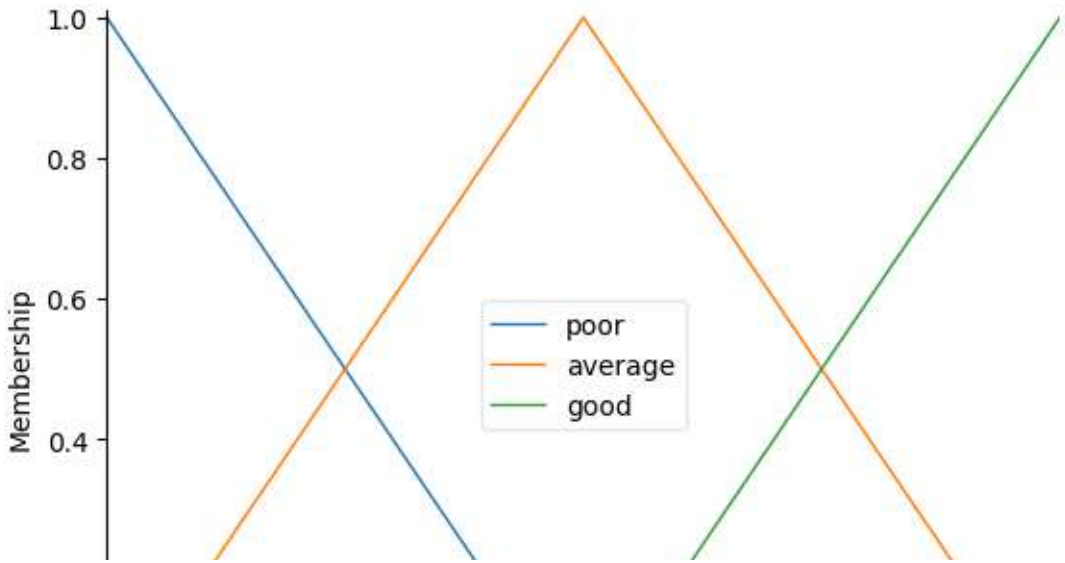
```
from skfuzzy import control as ctrl
# New Antecedent/Consequent objects hold universe variables and membership
# functions
quality = ctrl.Antecedent(np.arange(0, 11, 1), 'quality')
service = ctrl.Antecedent(np.arange(0, 11, 1), 'service')
tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')
```

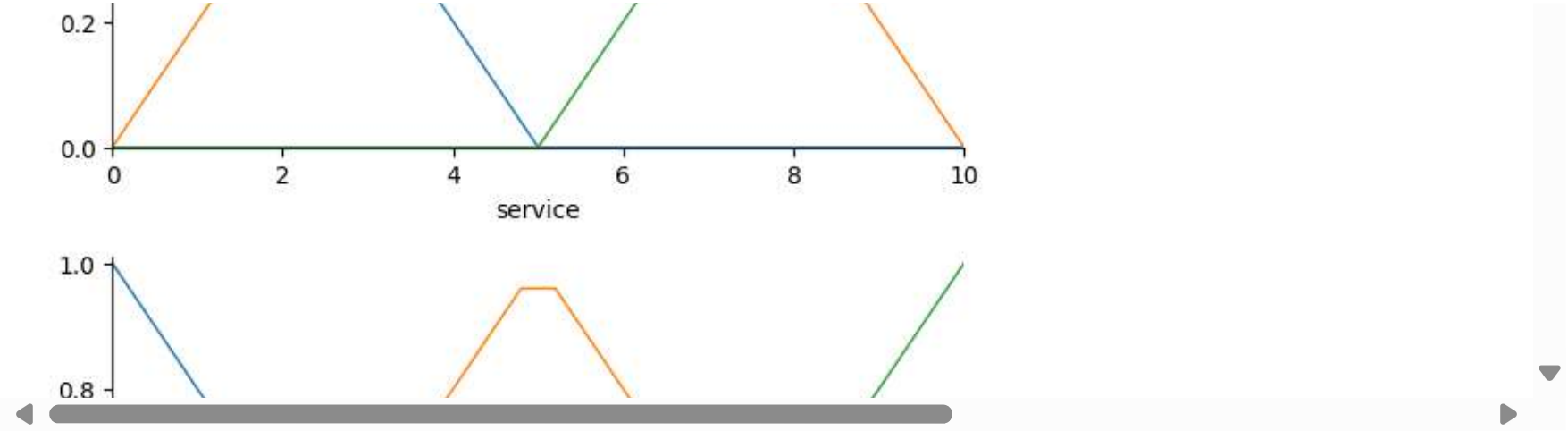
Membership Functions

```
# Auto-membership function population is possible with .automf(3, 5, or 7)
quality.automf(3)
quality.view()
service.automf(3)
service.view()
tip.automf(3)
tip.view()
```

```
c:\Users\sehre\AppData\Local\Programs\Python\Python311\Lib\site-packages\skfuzzy\control\fuzzyvariable.py:122: UserW
  fig.show()
```







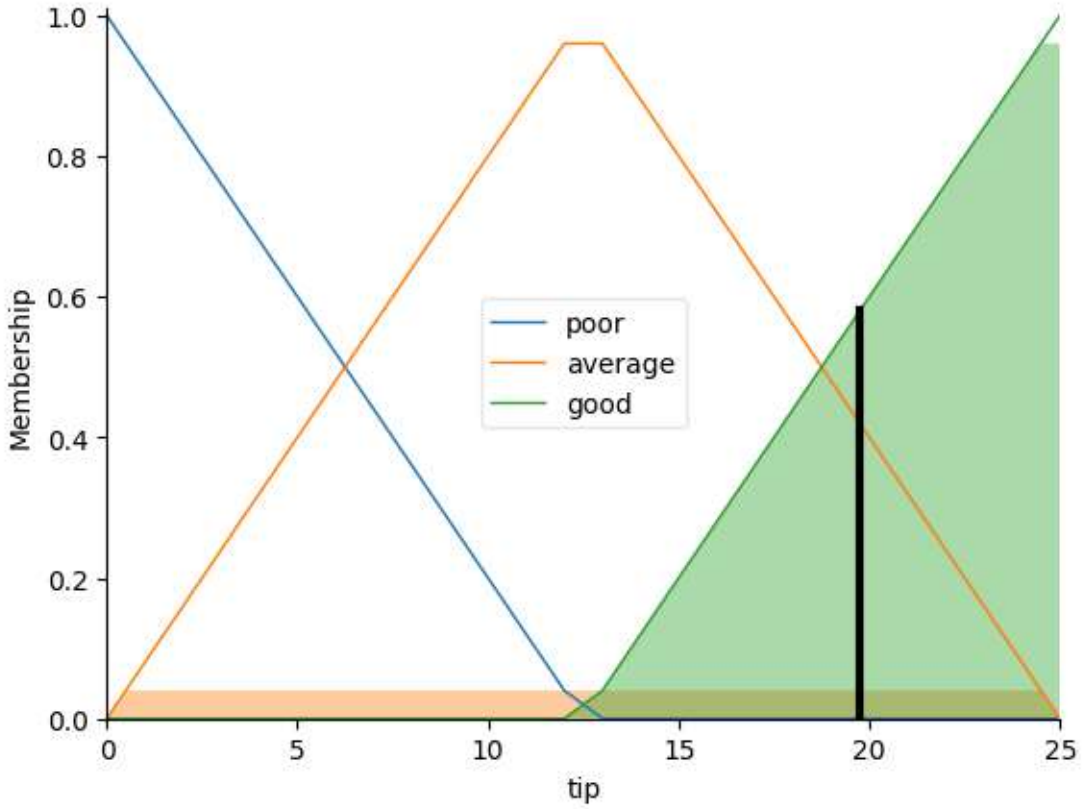
Rules

```
rule1 = ctrl.Rule(quality['poor'] | service['poor'], tip['poor'])
rule2 = ctrl.Rule(service['average'], tip['average'])
rule3 = ctrl.Rule(service['good'] | quality['good'], tip['good'])
```

Control System

```
tipping_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
tipping = ctrl.ControlSystemSimulation(tipping_ctrl)
tipping.input['quality'] = 6.5
tipping.input['service'] = 9.8

# Crunch the numbers
tipping.compute()
print (tipping.output['tip'])
tip.view(sim=tipping)
```

```
19.76409495548962
```



Write 2 nested for loops :

1. from 0 to 10 with step .5 (20)
2. from 0 to 10 with step .5 (20)

invokes the fuzzy logic and creates the tip amount

df:

foodQ | Service Quality | Tip Payment

0 0 ? 0.5 0.5 ? . . ? . . ? . . ? . . ? . . ?

visualize :

a. scatter plot foodq - servq x, y

b. parallel coordinates - fq, sq, tip

Lab 3 - submit as pdf

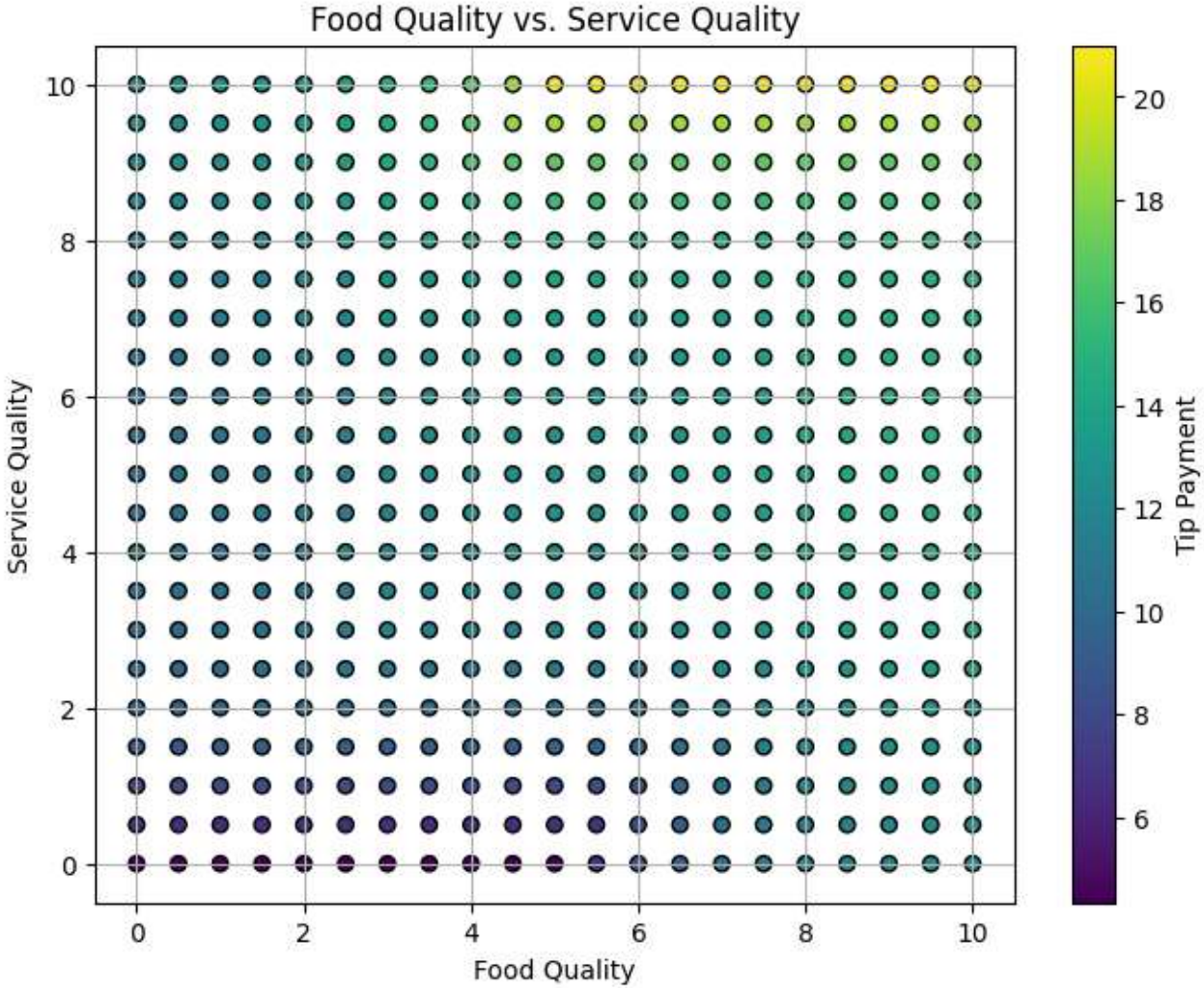
```
print(x_qual)
print(x_serv)
print(x_tip)
```

```
import pandas as pd

results = []
for food_quality in x_qual:
    for service_quality in x_serv:
        # Fuzzification
        qual_level_lo = fuzz.interp_membership(x_qual, qual_lo, food_quality)
        qual_level_md = fuzz.interp_membership(x_qual, qual_md, food_quality)
        qual_level_hi = fuzz.interp_membership(x_qual, qual_hi, food_quality)

        serv_level_lo = fuzz.interp_membership(x_serv, serv_lo, service_quality)
        serv_level_md = fuzz.interp_membership(x_serv, serv_md, service_quality)
        serv_level_hi = fuzz.interp_membership(x_serv, serv_hi, service_quality)

        # Rule Evaluation
        active_rule1 = np.fmax(qual_level_lo, serv_level_lo)
        tip_activation_lo = np.fmin(active_rule1, tip_lo)
        tip_activation_md = np.fmin(serv_level_md, tip_md)

        active_rule3 = np.fmax(qual_level_hi, serv_level_hi)
        tip_activation_hi = np.fmin(active_rule3, tip_hi)
        aggregated = np.fmax(tip_activation_lo, np.fmax(tip_activation_md, tip_activation_hi))
        tip = fuzz.defuzz(x_tip, aggregated, 'centroid')
        results.append([food_quality, service_quality, tip])

df = pd.DataFrame(results, columns=['Food Quality', 'Service Quality', 'Tip Payment'])

print(df)
```

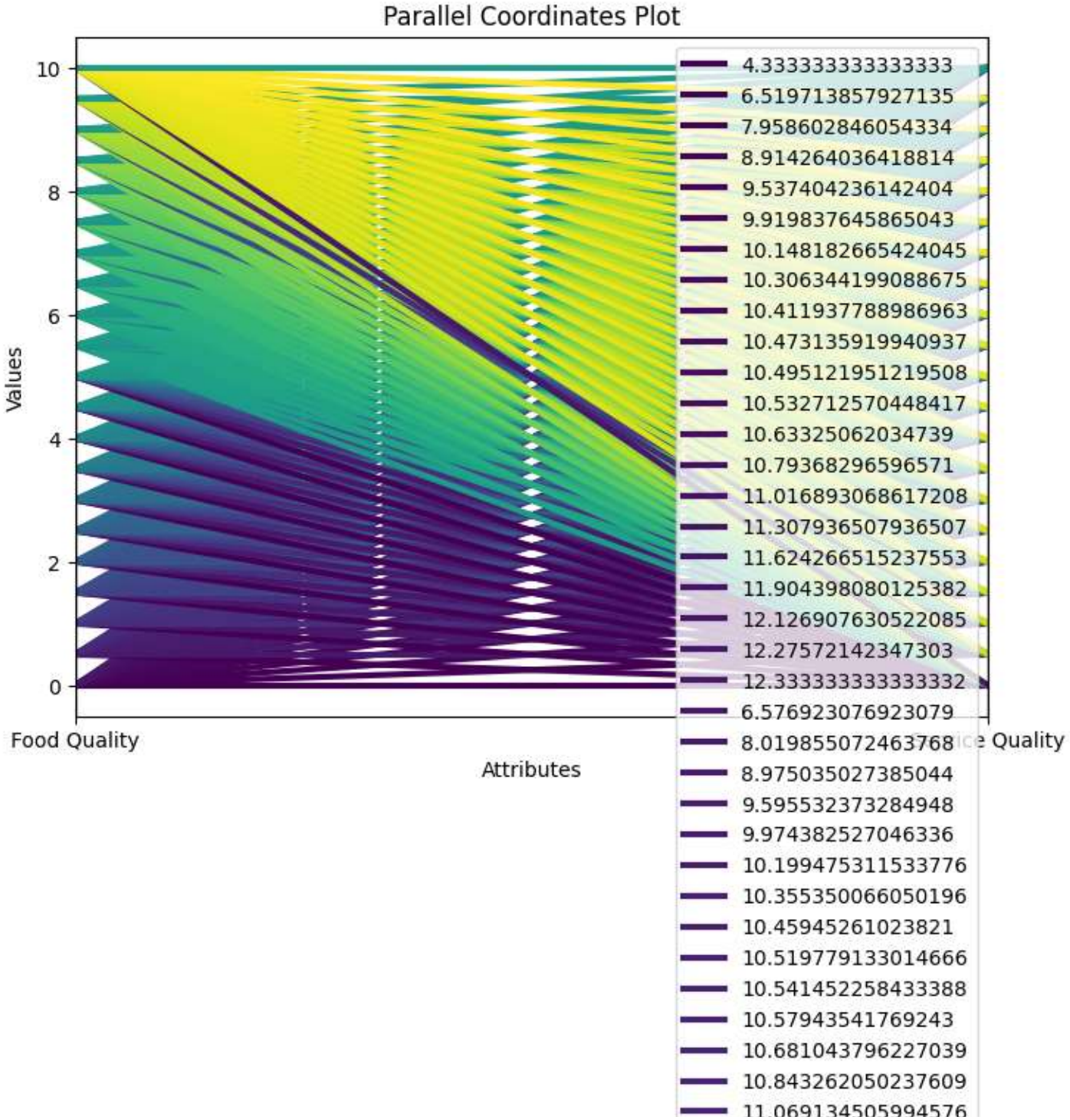
```
     Food Quality  Service Quality  Tip Payment
0             0.0              0.0     4.333333
1             0.0              0.5     6.519714
2             0.0              1.0     7.958603
3             0.0              1.5     8.914264
4             0.0              2.0     9.537404
..            ...              ...          ...
436          10.0              8.0    15.475674
437          10.0              8.5    16.063443
438          10.0              9.0    17.028537
439          10.0              9.5    18.551942
```

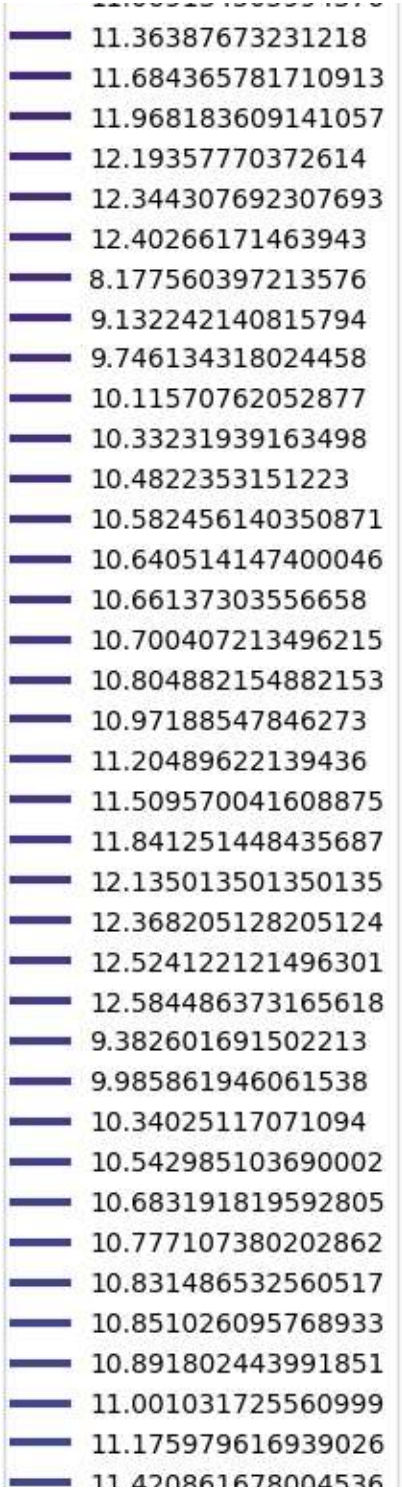
```
     440          10.0             10.0    21.000000

     [441 rows x 3 columns]
```

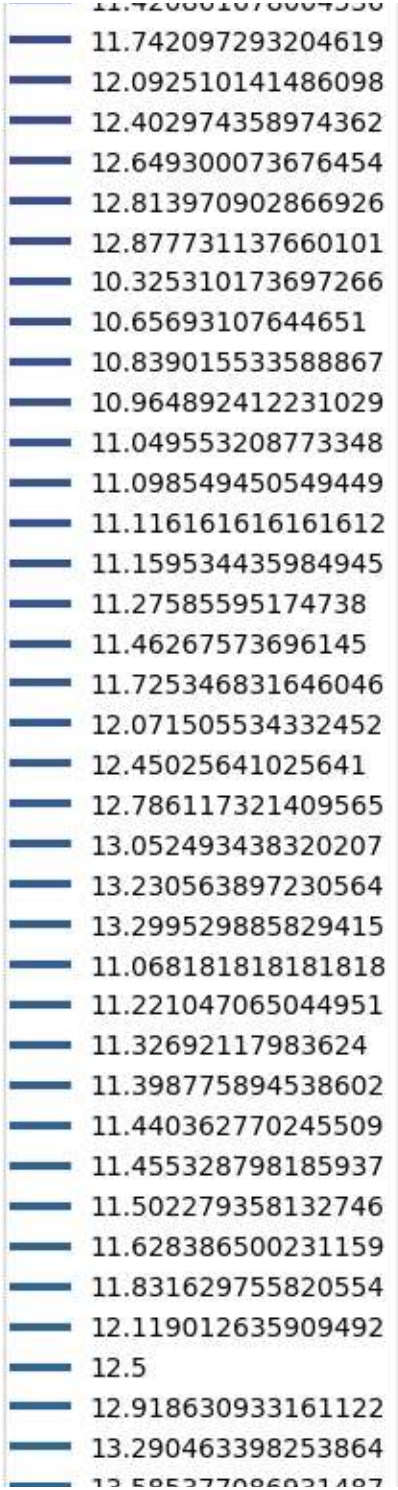
```
import matplotlib.pyplot as plt
from pandas.plotting import parallel_coordinates

# Scatter plot of Food Quality vs. Service Quality
plt.figure(figsize=(8, 6))
plt.scatter(df['Food Quality'], df['Service Quality'], c=df['Tip Payment'], cmap='viridis', edgecolor='k')
plt.colorbar(label='Tip Payment')
plt.xlabel('Food Quality')
plt.ylabel('Service Quality')
plt.title('Food Quality vs. Service Quality')
plt.grid(True)
plt.show()
```




```
# Parallel coordinates plot
plt.figure(figsize=(8, 6))
parallel_coordinates(df, 'Tip Payment', colormap='viridis', linewidth=3)
plt.xlabel('Attributes')
plt.ylabel('Values')
plt.title('Parallel Coordinates Plot')
plt.grid(False)
plt.show()
```


Parallel Coordinates Plot
Values
Attributes
Food Quality
4.333333333333333
6.519713857927135
7.958602846054334
8.914264036418814
9.537404236142404
9.919837645865043
10.148182665424045
10.306344199088675
10.411937788986963
10.473135919940937
10.495121951219508
10.532712570448417
10.63325062034739
10.79368296596571
11.016893068617208
11.307936507936507
11.624266515237553
11.904398080125382
12.126907630522085
12.27572142347303
12.333333333333332
6.576923076923079
8.019855072463768
8.975035027385044
9.595532373284948
9.974382527046336
10.199475311533776
10.355350066050196
10.45945261023821
10.519779133014666
10.541452258433388
10.57943541769243
10.681043796227039
10.843262050237609
11.069134505994576


11.36387673231218
11.684365781710913
11.968183609141057
12.19357770372614
12.344307692307693
12.40266171463943
8.177560397213576
9.132242140815794
9.746134318024458
10.11570762052877
10.33231939163498
10.4822353151223
10.582456140350871
10.640514147400046
10.66137303556658
10.700407213496215
10.804882154882153
10.97188547846273
11.20489622139436
11.509570041608875
11.841251448435687
12.135013501350135
12.368205128205124
12.524122121496301
12.584486373165618
9.382601691502213
9.985861946061538
10.34025117071094
10.542985103690002
10.683191819592805
10.777107380202862
10.831486532560517
10.851026095768933
10.891802443991851
11.001031725560999
11.175979616939026


11.742097293204619
12.092510141486098
12.402974358974362
12.649300073676454
12.813970902866926
12.877731137660101
10.325310173697266
10.65693107644651
10.839015533588867
10.964892412231029
11.049553208773348
11.098549450549449
11.116161616161612
11.159534435984945
11.27585595174738
11.46267573696145
11.725346831646046
12.071505534332452
12.45025641025641
12.786117321409565
13.052493438320207
13.230563897230564
13.299529885829415
11.068181818181818
11.221047065044951
11.32692117983624
11.398775894538602
11.440362770245509
11.455328798185937
11.502279358132746
11.628386500231159
11.831629755820554
12.119012635909492
12.5
12.918630933161122
13.290463398253864