1) Defines the problem with at least 3 independent variable and at least 3 rules to solve a problem

Three Independent Vavriables: Price(P) Demand(D) Surge(S)

Either Demand or surge is more than price is more if demand or surge is less than price is less if demand or surge than price is medium

2) Install necessary packages and import them in your Jupyter notebook or python ID

In [26]:

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

3) Generate universe variables

In [27]:

```
x_D = np.arange(0, 11, 1)
x_S = np.arange(0, 11, 1)
x_P  = np.arange(0, 26, 1)
```

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

Price low medium high Demand low medium high Surge

low medium high

4) Generate fuzzy membership functions

In [28]:

```
# Generate fuzzy membership functions
D_lo = fuzz.trimf(x_TP, [0, 0, 10])
D_md = fuzz.trimf(x_TP, [0, 10, 15])
D_hi = fuzz.trimf(x_TP, [10, 15, 15])
S_lo = fuzz.trimf(x_FQ, [0, 0, 5])
S_md = fuzz.trimf(x_FQ, [0, 5, 10])
S_hi = fuzz.trimf(x_FQ, [5, 10, 10])
P_lo = fuzz.trimf(x_M, [0, 0, 13])
P_md = fuzz.trimf(x_M, [0, 13, 25])
P_hi = fuzz.trimf(x_M, [13, 25, 25])
```

In [29]:

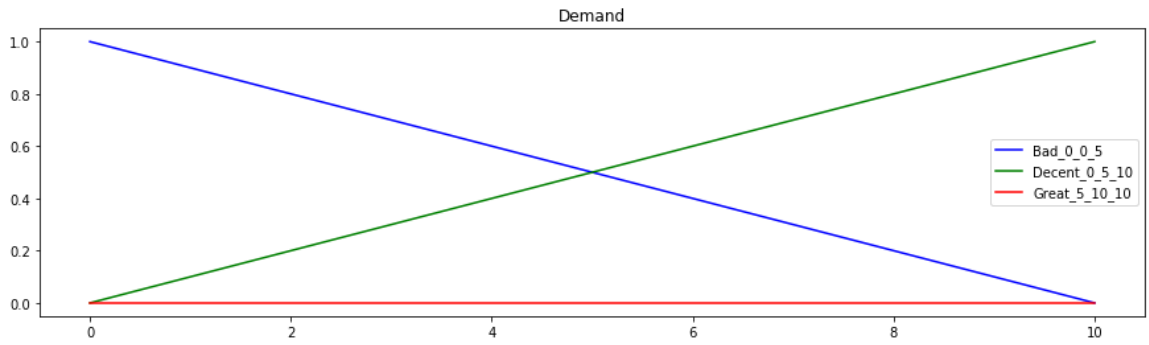
```
print(x_D)
print(D_lo)
print(x_D)
print(D_md)
print(x_D)
print(D_hi)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10]
[1.  0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0. ]
[ 0  1  2  3  4  5  6  7  8  9 10]
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1. ]
[ 0  1  2  3  4  5  6  7  8  9 10]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

5) Visualize these universes and membership functions

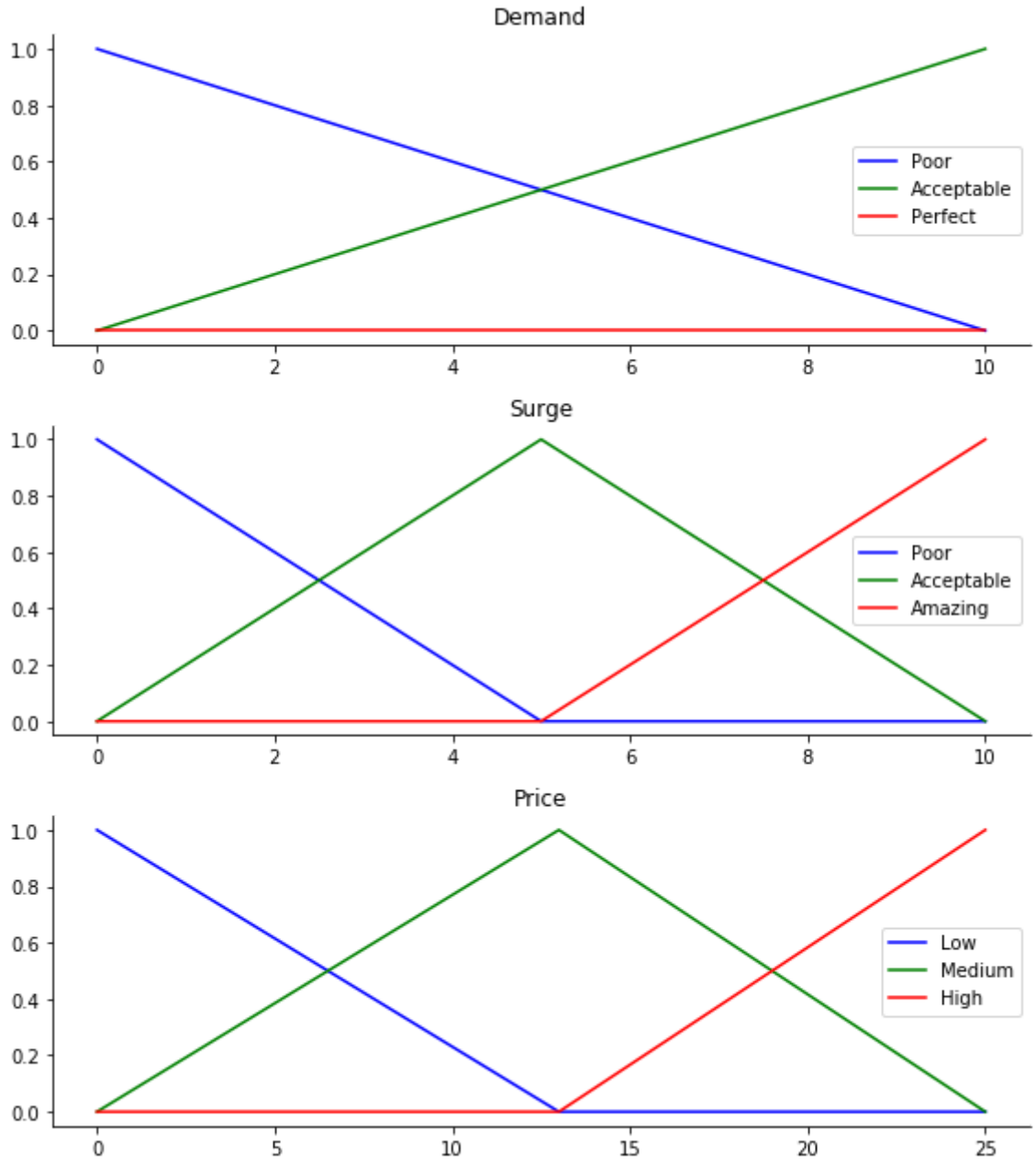
In [30]:

```
fig, (ax0) = plt.subplots(nrows=1, figsize=(15, 4))

ax0.plot(x_D, D_lo, 'b', linewidth=1.5, label='Bad_0_0_5')
ax0.plot(x_D, D_md, 'g', linewidth=1.5, label='Decent_0_5_10')
ax0.plot(x_D, D_hi, 'r', linewidth=1.5, label='Great_5_10_10')

ax0.set_title('Demand')
ax0.legend()

plt.show()
```



In [32]:

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

ax0.plot(x_D, D_lo, 'b', linewidth=1.5, label='Poor')
ax0.plot(x_D, D_md, 'g', linewidth=1.5, label='Acceptable')
ax0.plot(x_D, D_hi, 'r', linewidth=1.5, label='Perfect')
ax0.set_title('Demand')
ax0.legend()

ax1.plot(x_S, S_lo, 'b', linewidth=1.5, label='Poor')
ax1.plot(x_S, S_md, 'g', linewidth=1.5, label='Acceptable')
ax1.plot(x_S, S_hi, 'r', linewidth=1.5, label='Amazing')
ax1.set_title('Surge')
ax1.legend()

ax2.plot(x_P, P_lo, 'b', linewidth=1.5, label='Low')
ax2.plot(x_P, P_md, 'g', linewidth=1.5, label='Medium')
ax2.plot(x_P, P_hi, 'r', linewidth=1.5, label='High')
ax2.set_title('Price')
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()
```



6) Define rules

Either Demand or surge is more than price is more if demand or surge is less than price is less if demand or surge than price is medium

In [7]:

```
S_LOW = fuzz.interp_membership(x_D, D_lo, 3.4)
S_MED = fuzz.interp_membership(x_D, D_md, 3.4)
S_HI = fuzz.interp_membership(x_D, D_hi, 3.4)
print(S_LOW,S_MED,S_HI)
```

```
0.6599999999999999 0.33999999999999997 0.0
```

In [8]:

```
D_LOW = fuzz.interp_membership(x_S, S_lo, 8.8)
D_MED = fuzz.interp_membership(x_S, S_md, 8.8)
D_HI = fuzz.interp_membership(x_S, S_hi, 8.8)
print(D_LOW,D_MED,D_HI)
```

```
0.0 0.23999999999999988 0.7600000000000002
```

In [9]:

```
# RULE 1
active_rule1 = np.fmax(S_LOW,S_LOW)
PRICE_LOW = np.fmin(active_rule1, P_lo)
print(PRICE_LOW)
```

```
[0.66       0.66       0.66       0.66       0.66       0.61538462
 0.53846154 0.46153846 0.38461538 0.30769231 0.23076923 0.15384615
 0.07692308 0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.        ]
```

In [10]:

```
# RULE 2
active_rule2 = np.fmax(S_MED,D_MED)
PRICE_MED = np.fmin(active_rule2, P_md)
print(PRICE_MED)
```

```
[0.         0.07692308 0.15384615 0.23076923 0.30769231 0.34
 0.34       0.34       0.34       0.34       0.34       0.34
 0.34       0.34       0.34       0.34       0.34       0.34
 0.34       0.34       0.34       0.33333333 0.25       0.16666667
 0.08333333 0.        ]
```

In [11]:

```
# RULE 3
active_rule3 = np.fmax(S_HI,D_HI)
PRICE_HI = np.fmin(active_rule3, P_hi)
print(PRICE_HI)
```

```
[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       0.76
 0.76       0.76       0.76      ]
```

In [13]:

```
aggregated = np.fmax(PRICE_LOW,np.fmax(PRICE_MED, PRICE_HI))
print(aggregated)
```

```
[0.66       0.66       0.66       0.66       0.66       0.61538462
 0.53846154 0.46153846 0.38461538 0.34       0.34       0.34
 0.34       0.34       0.34       0.34       0.34       0.34
 0.41666667 0.5        0.58333333 0.66666667 0.75       0.76
 0.76       0.76       0.76      ]
```

7) Defuzzification

In [14]:

```
PRICE = fuzz.defuzz(x_P, aggregated, 'centroid')
print(PRICE)
```

```
12.716558221228695
```

In [15]:

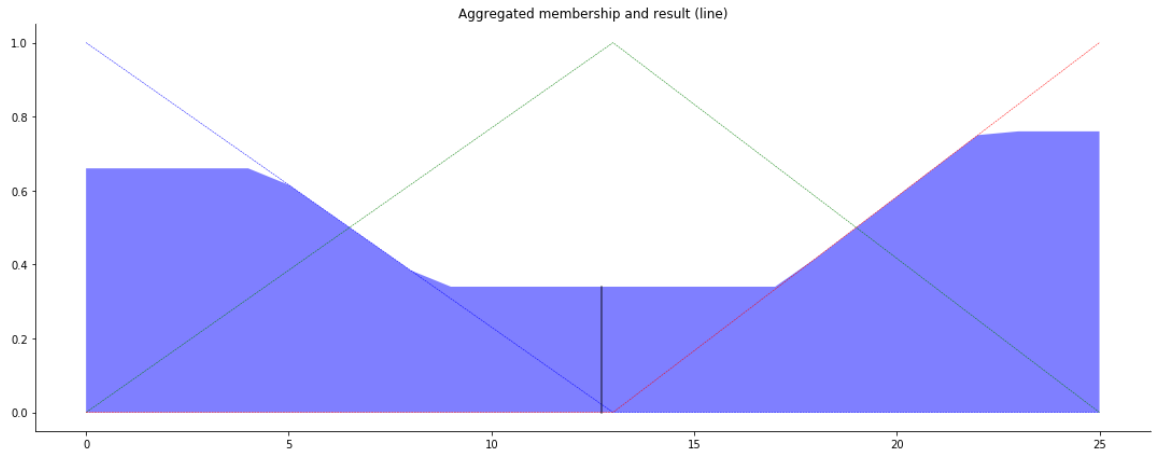
```
PRICE_DE = fuzz.interp_membership(x_P, aggregated,PRICE)  # for plot
PRICE_DE
```

Out[15]:

```
0.33999999999999997
```

In [16]:

```
# Visualize this
fig, ax0 = plt.subplots(figsize=(15, 6))
ax0.plot(x_P, P_lo, 'b', linewidth=0.5, linestyle='--', )
ax0.plot(x_P, P_md, 'g', linewidth=0.5, linestyle='--')
ax0.plot(x_P, P_hi, 'r', linewidth=0.5, linestyle='--')
ax0.fill_between(x_M, PRICE0, aggregated, facecolor='blue', alpha=0.5)
ax0.plot([PRICE, PRICE], [0, PRICE_DE], 'k', linewidth=1.5, alpha=0.6)
ax0.set_title('Aggregated membership and result (line)')
# 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()
```

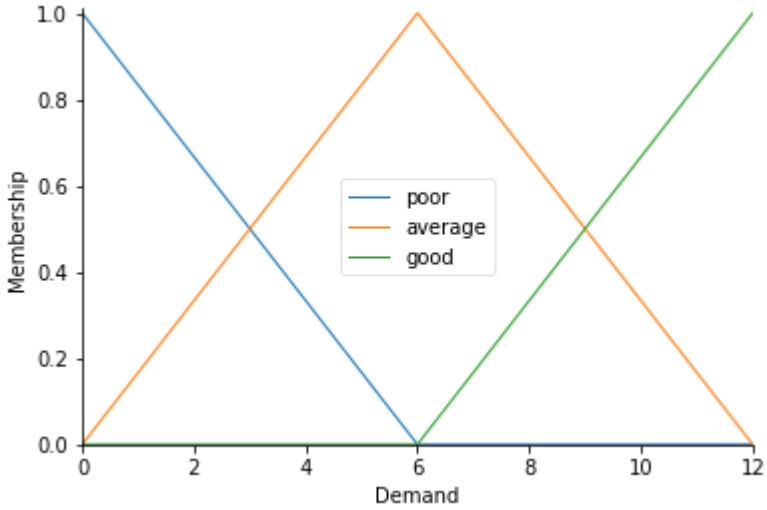


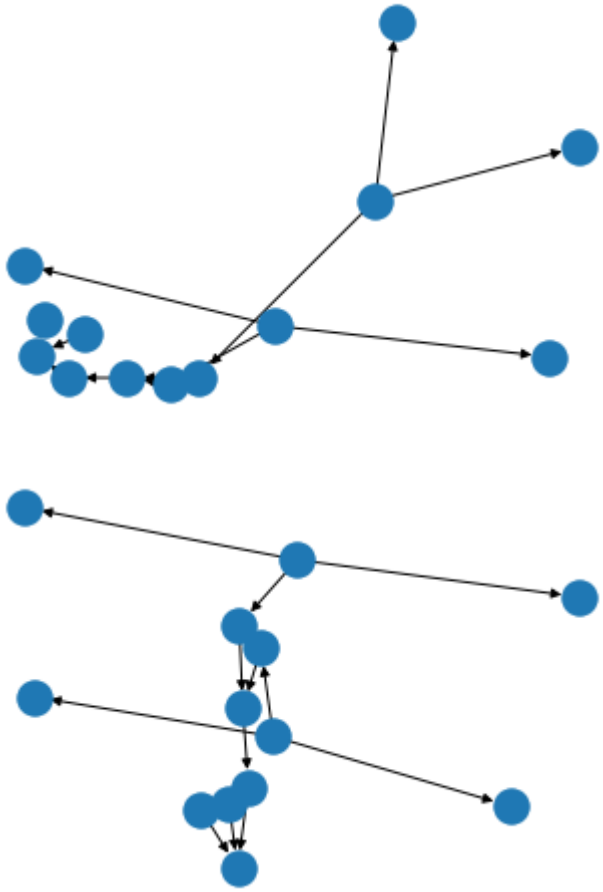
8) report your final outcome

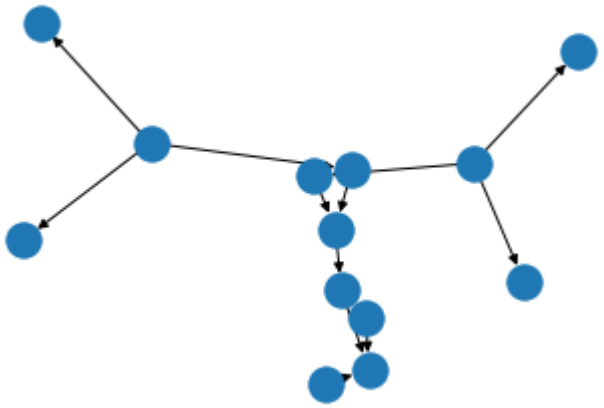
In [19]:

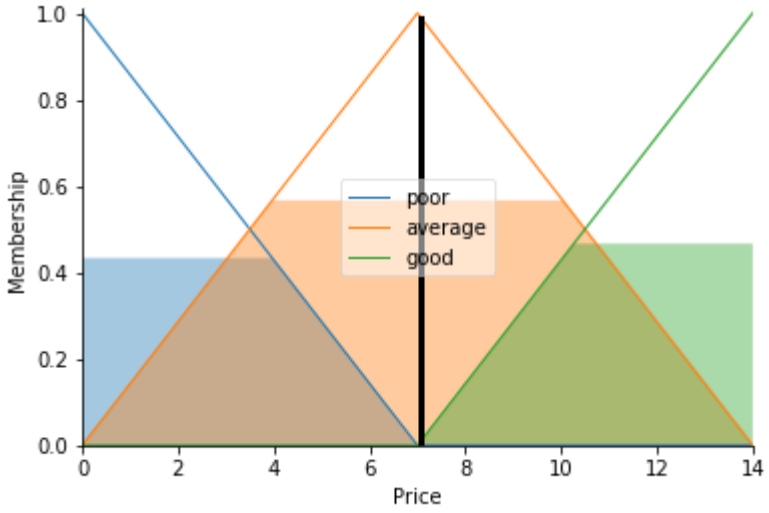
```
from skfuzzy import control as ctrl

D = ctrl.Antecedent(np.arange(0, 13, 1), 'Demand')
S = ctrl.Antecedent(np.arange(0, 13, 1), 'Surge')
P = ctrl.Consequent(np.arange(0, 15, 1), 'Price')

D.automf(3)
S.automf(3)
M.automf(3)
D.view()


rule1 = ctrl.Rule(D['poor'] | S['poor'], P['poor'])
rule2 = ctrl.Rule(D['average']| S['average'], P['average'])
rule3 = ctrl.Rule(D['good'] | S['good'], P['good'])

rule1.view()
rule2.view()
rule3.view()

Price_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
Price = ctrl.ControlSystemSimulation(Price_ctrl)
Price.input['Demand'] = 3.4
Price.input['Surge'] = 8.8

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

```
7.083471690911773
```







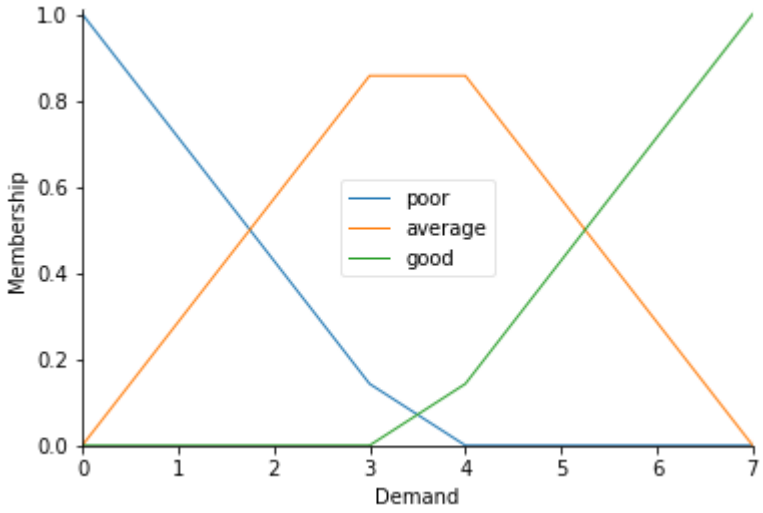

1.0
0.8
0.6
0.4
0.2
0.0
Membership
poor
average
good
0
2
4
6
8
10
12
14
Price

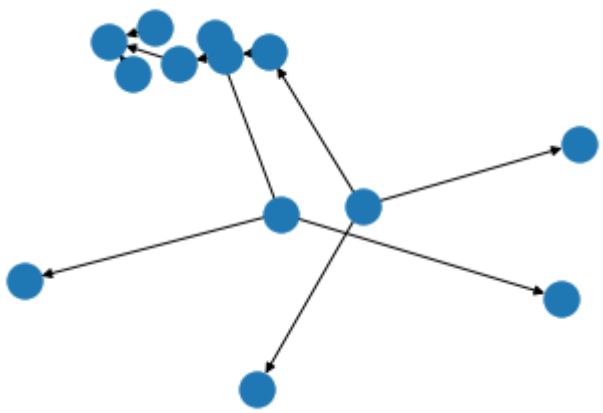
In [25]:

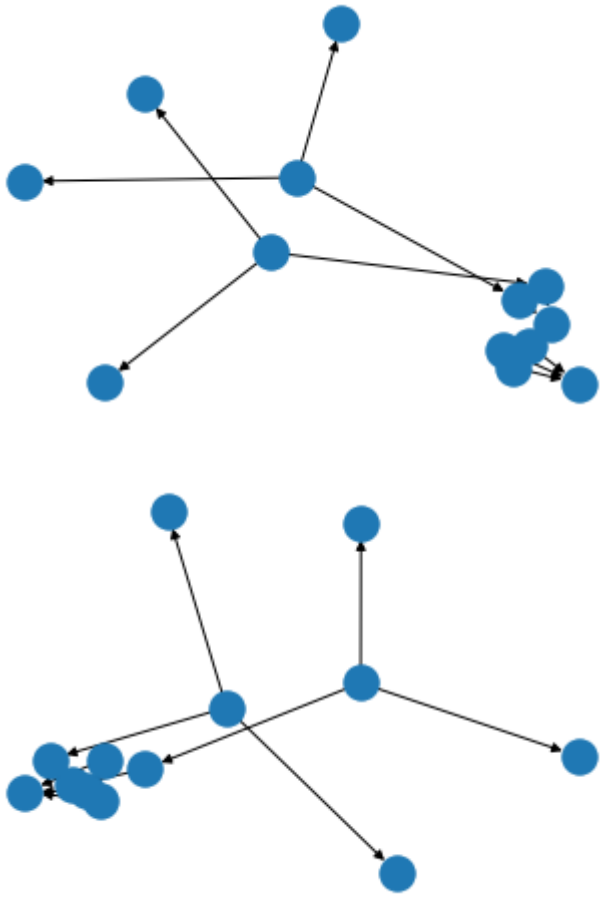
```
#Condition 1 TP =4.4 and FQ= 9.1 - MILEAGE 10

from skfuzzy import control as ctrl

D = ctrl.Antecedent(np.arange(0, 8, 1), 'Demand')
S = ctrl.Antecedent(np.arange(0, 8, 1), 'Surge')
P = ctrl.Consequent(np.arange(0, 18, 1), 'Price')

D.automf(3)
S.automf(3)
P.automf(3)
D.view()


rule1 = ctrl.Rule(D['poor'] | S['poor'], P['poor'])
rule2 = ctrl.Rule(D['average']| S['average'], P['average'])
rule3 = ctrl.Rule(D['good'] | S['good'], P['good'])

rule1.view()
rule2.view()
rule3.view()

Mileage_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
Mileage = ctrl.ControlSystemSimulation(Mileage_ctrl)
Mileage.input['Demand'] = 4.4
Mileage.input['Surge'] = 9.1

# Crunch the numbers
Price.compute()
print (Mileage.output['Price'])
P.view(sim=Price)
```

```
C:\Users\Admin\Anaconda3\lib\site-packages\skfuzzy\control\fuzzyvariable.p
y:122: UserWarning: Matplotlib is currently using module://ipykernel.pyla
b.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
C:\Users\Admin\Anaconda3\lib\site-packages\networkx\drawing\nx_pylab.py:57
9: MatplotlibDeprecationWarning:
The iterable function was deprecated in Matplotlib 3.1 and will be removed
in 3.3. Use np.iterable instead.
  if not cb.iterable(width):
C:\Users\Admin\Anaconda3\lib\site-packages\networkx\drawing\nx_pylab.py:67
6: MatplotlibDeprecationWarning:
The iterable function was deprecated in Matplotlib 3.1 and will be removed
in 3.3. Use np.iterable instead.
  if cb.iterable(node_size):  # many node sizes
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call las
t)
<ipython-input-25-9695de7641d8> in <module>
     28 # Crunch the numbers
     29 Price.compute()
---> 30 print (Mileage.output['Price'])
     31 P.view(sim=Price)

KeyError: 'Price'
```



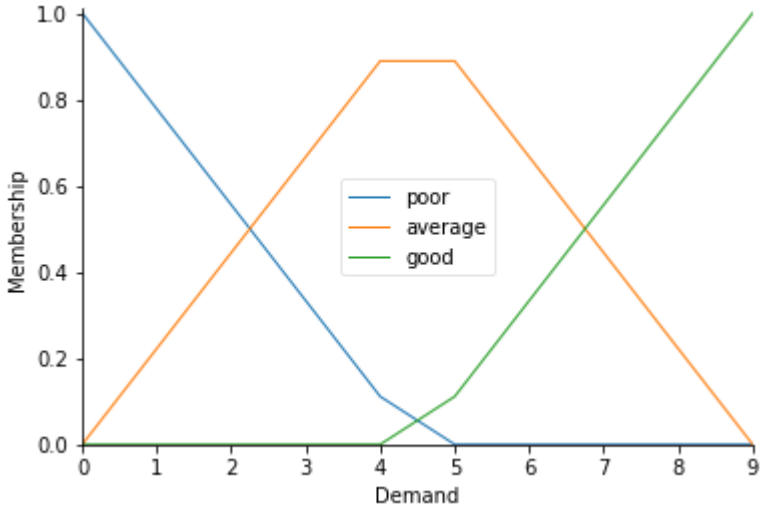


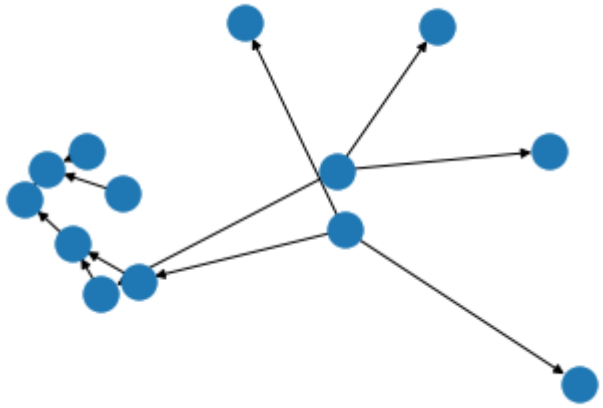


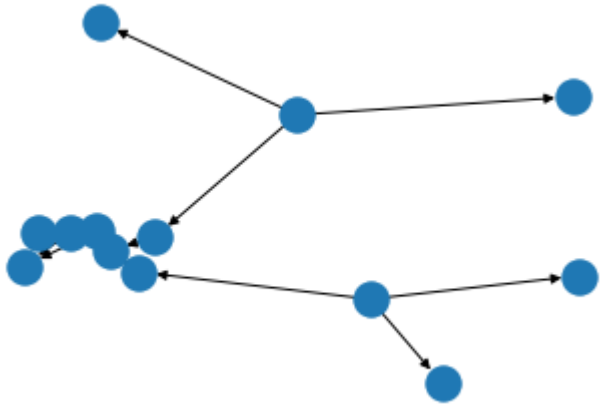
In [20]:

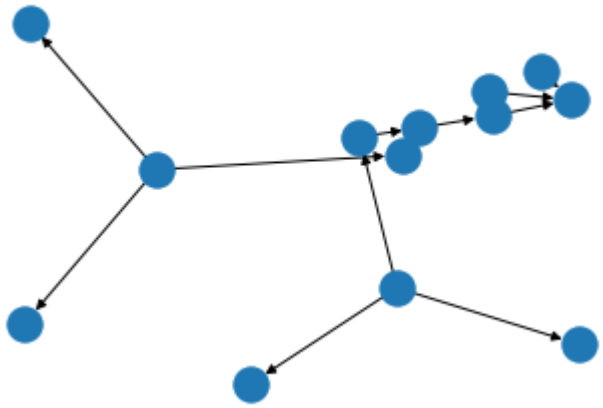
```
#Condition 2 TP = 9.1 and FQ =9.1 and MILEAGE = 19.99

from skfuzzy import control as ctrl

D = ctrl.Antecedent(np.arange(0, 10, 1), 'Demand')
S = ctrl.Antecedent(np.arange(0, 10, 1), 'Surge')
P = ctrl.Consequent(np.arange(0, 25, 1), 'Price')

D.automf(3)
S.automf(3)
P.automf(3)
D.view()


rule1 = ctrl.Rule(D['poor'] | S['poor'], P['poor'])
rule2 = ctrl.Rule(D['average']| S['average'], P['average'])
rule3 = ctrl.Rule(D['good'] | S['good'], P['good'])

rule1.view()
rule2.view()
rule3.view()

Price_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
Price = ctrl.ControlSystemSimulation(Price_ctrl)
Price.input['Demand'] = 9.1
Price.input['Surge'] = 9.1

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

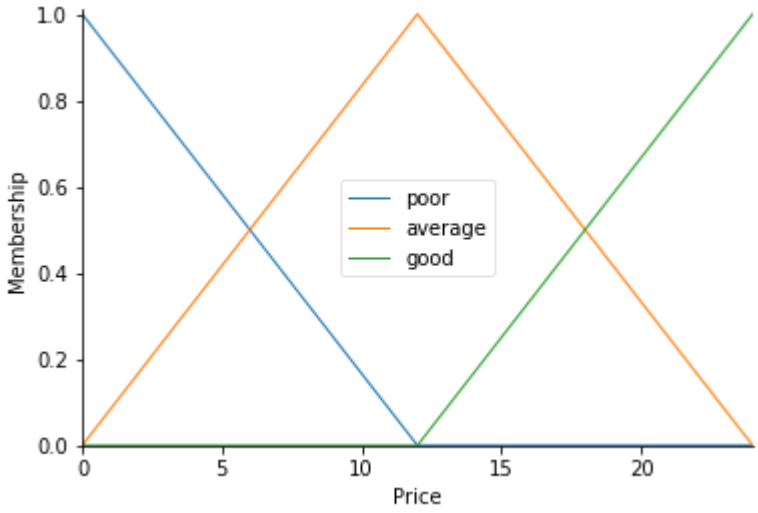
```
7.895092776850136
```







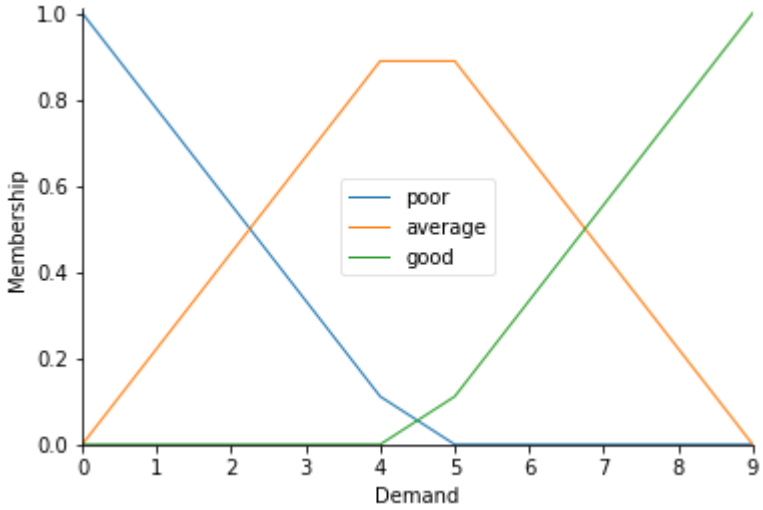


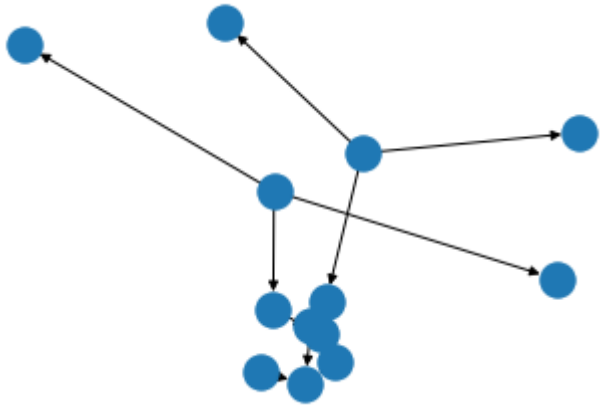

1.0
0.8
0.6
0.4
0.2
0.0
Membership
poor
average
good
0
5
10
15
20
Price

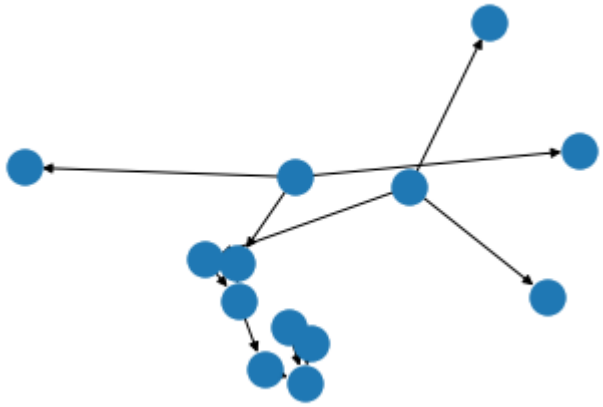
In [21]:

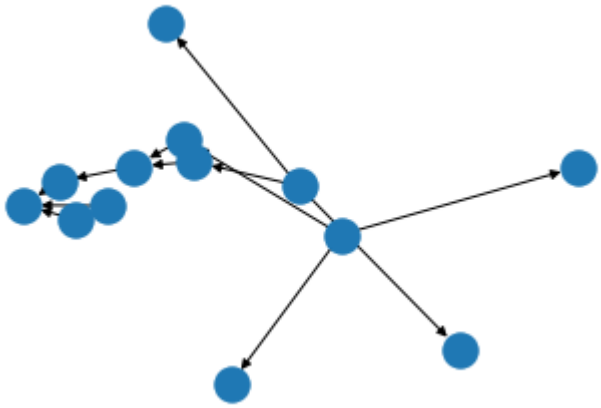
```
#Condition 3 TP = 4.5 and FQ =5.1 and MILEAGE = 12.08

from skfuzzy import control as ctrl

D = ctrl.Antecedent(np.arange(0, 10, 1), 'Demand')
S = ctrl.Antecedent(np.arange(0, 10, 1), 'Surge')
P = ctrl.Consequent(np.arange(0, 25, 1), 'Price')

D.automf(3)
S.automf(3)
P.automf(3)
D.view()


rule1 = ctrl.Rule(D['poor'] | S['poor'], P['poor'])
rule2 = ctrl.Rule(D['average']| S['average'], P['average'])
rule3 = ctrl.Rule(D['good'] | S['good'], P['good'])

rule1.view()
rule2.view()
rule3.view()

Price_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
Price = ctrl.ControlSystemSimulation(Price_ctrl)
Price.input['Demand'] = 4.5
Price.input['Surge'] = 5.1

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

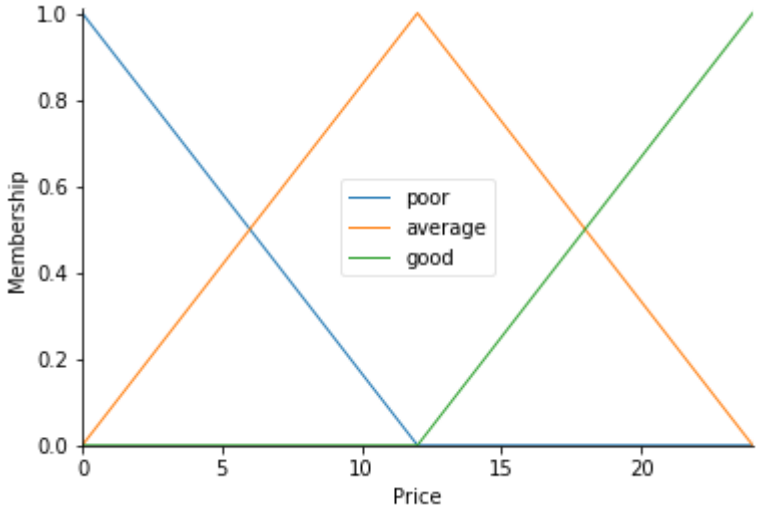
```
6.791340636411059
```











In [22]:

```
pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\admin\anaconda3\lib\site
-packages (0.24.2)
Requirement already satisfied: python-dateutil>=2.5.0 in c:\users\admin\an
aconda3\lib\site-packages (from pandas) (2.8.0)
Requirement already satisfied: numpy>=1.12.0 in c:\users\admin\anaconda3\l
ib\site-packages (from pandas) (1.16.4)
Requirement already satisfied: pytz>=2011k in c:\users\admin\anaconda3\lib
\site-packages (from pandas) (2019.1)
Requirement already satisfied: six>=1.5 in c:\users\admin\anaconda3\lib\si
te-packages (from python-dateutil>=2.5.0->pandas) (1.12.0)
Note: you may need to restart the kernel to use updated packages.
```

In [23]:

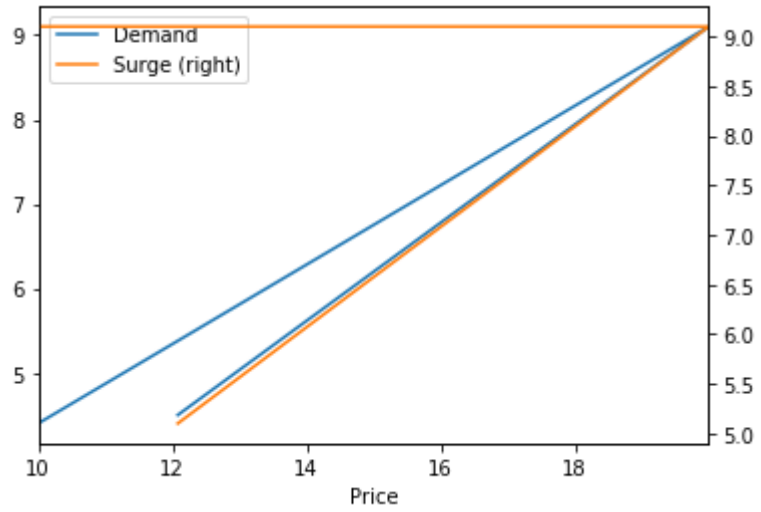
```
#Condition 1 TP =4.4 and FQ= 9.1 - Price 10
#Condition 2 TP = 9.1 and FQ =9.1 and Price = 19.99
#Condition 3 TP = 4.5 and FQ =5.1 and Price = 12.08
import pandas as pd

df = pd.DataFrame(data={'Demand': [4.4, 9.1, 4.5],
                        'Surge': [9.1, 9.1, 5.1],
                        'Price': [10, 19.99, 12.08]})
```

In [24]:

```
fig, ax = plt.subplots()

df.plot(x = 'Price', y = 'Demand', ax = ax)

df.plot(x = 'Price', y = 'Surge', ax = ax, secondary_y = True)
```

Out[24]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d18ed87dd8>
```



In [ ]: