Name - Jinal Shah
UID - 2019230070

Exp 3 - To implement a fuzzy controller logic

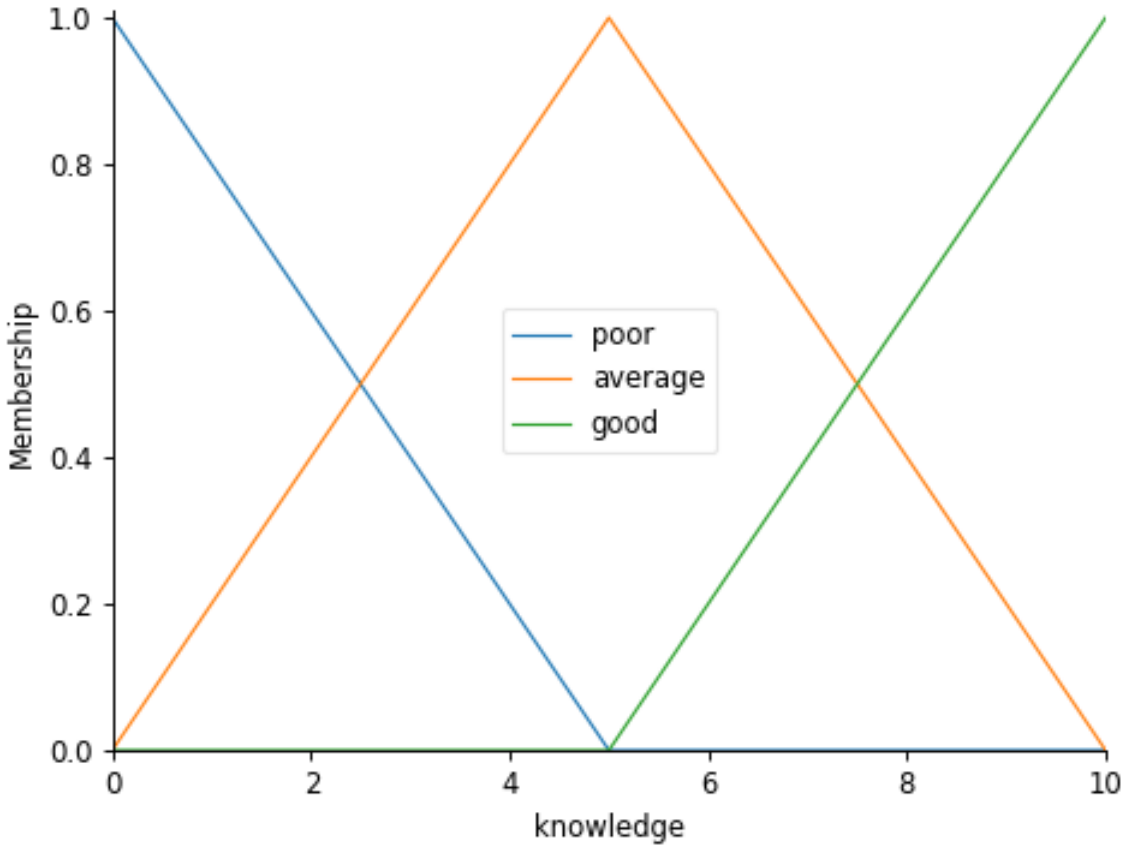
Aim - To determine marks based on efforts and knowledge

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

```
Collecting scikit-fuzzy
  Downloading scikit-fuzzy-0.4.2.tar.gz (993 kB)
Requirement already satisfied: numpy>=1.6.0 in c:\users\ritik\appdata\local\programs\pyt
Requirement already satisfied: scipy>=0.9.0 in c:\users\ritik\appdata\local\programs\pyt
Collecting networkx>=1.9.0
  Downloading networkx-2.5-py3-none-any.whl (1.6 MB)
Requirement already satisfied: decorator>=4.3.0 in c:\users\ritik\appdata\local\programs
Building wheels for collected packages: scikit-fuzzy
  Building wheel for scikit-fuzzy (setup.py): started
  Building wheel for scikit-fuzzy (setup.py): finished with status 'done'
  Created wheel for scikit-fuzzy: filename=scikit_fuzzy-0.4.2-py3-none-any.whl size=894
  Stored in directory: c:\users\ritik\appdata\local\pip\cache\wheels\d5\74\fc\38588a3d2
Successfully built scikit-fuzzy
Installing collected packages: networkx, scikit-fuzzy
Successfully installed networkx-2.5 scikit-fuzzy-0.4.2
WARNING: You are using pip version 20.1.1; however, version 21.0.1 is available.
You should consider upgrading via the 'c:\users\ritik\appdata\local\programs\python\pyth
```

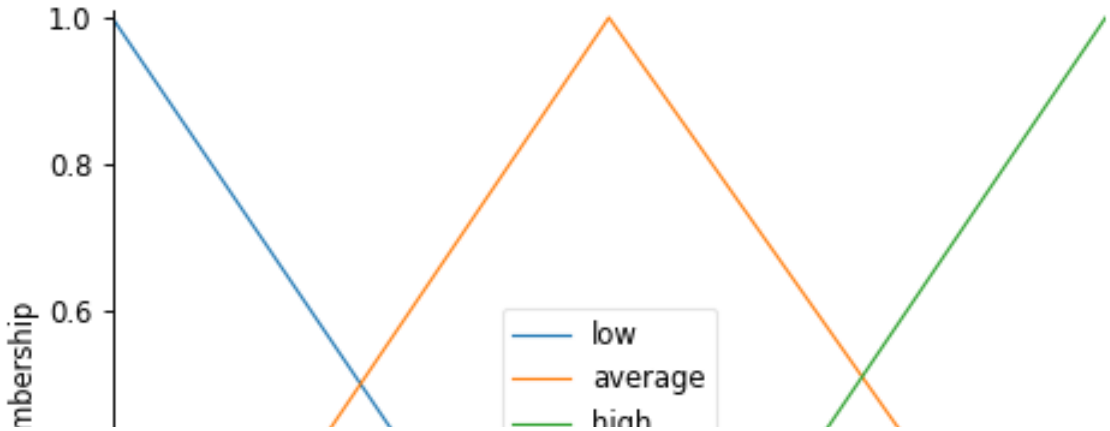
```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

# knowledge scale
knowledge = ctrl.Antecedent(np.arange(0, 11, 1), 'knowledge')

# effort scale
effort = ctrl.Antecedent(np.arange(0, 11, 1), 'effort')

marks = ctrl.Consequent(np.arange(0, 51, 1), 'marks',defuzzify_method='mom')

#auto membership func for knowledge and effort
knowledge.automf(3)
effort.automf(3)

#triangular membership func for marks
marks['low'] = fuzz.trimf(marks.universe, [0, 0, 25])
marks['average'] = fuzz.trimf(marks.universe, [0, 25, 51])
marks['high'] = fuzz.trimf(marks.universe, [25, 50, 51])
```

```
%matplotlib notebook

knowledge.view()
```




```
effort.view()
```

```
marks.view()
```

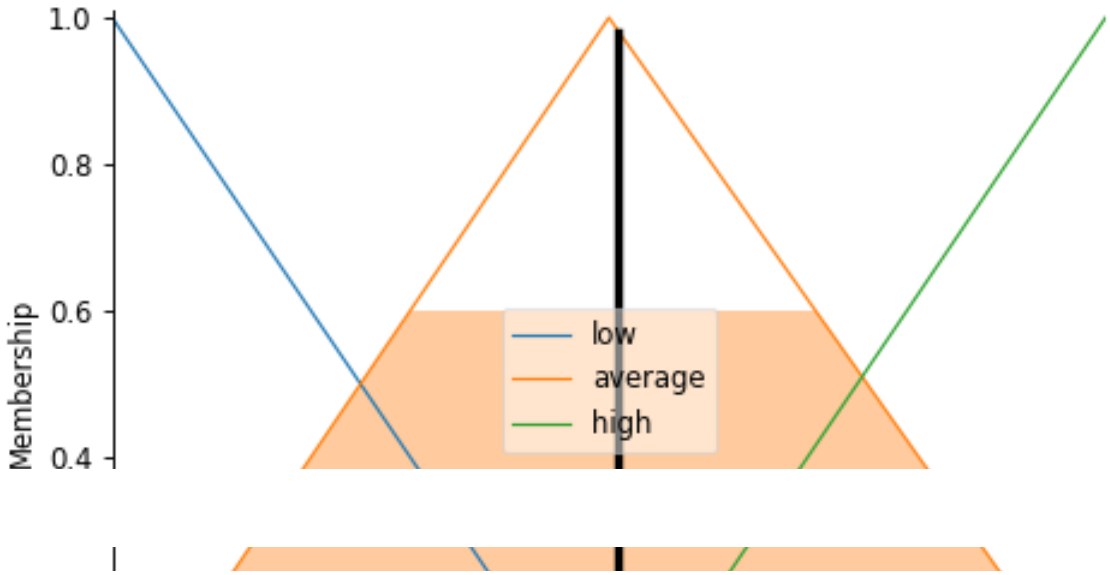



```
#rules

rule1 = ctrl.Rule(knowledge['poor'] & effort['poor'], marks['low'])
rule2 = ctrl.Rule(effort['average'] & knowledge["average"], marks['average'])
rule3 = ctrl.Rule(effort['good'] & knowledge['good'] , marks['high'])
rule4 = ctrl.Rule(effort['poor'] & knowledge['average'], marks["low"])
rule5 = ctrl.Rule(effort['average'] & knowledge['good'], marks["average"])
rule6 = ctrl.Rule(effort['good'] & knowledge['poor'], marks["average"])
rule7 = ctrl.Rule(effort['average'] & knowledge['poor'],marks['low'])
rule8 = ctrl.Rule(effort['good'] & knowledge['average'], marks["average"])
rule9 = ctrl.Rule(effort['poor'] & knowledge['good'], marks["average"])

rule1.view()
```

```
eval_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8, rule9
evaluator = ctrl.ControlSystemSimulation(eval_ctrl)
evaluator.input['knowledge'] = 6
evaluator.input['effort'] = 8

evaluator.compute()
```

```
print('Marks:',evaluator.output['marks'])
marks.view(sim=evaluator)
```

```
Marks: 25.472727272727273
```



Conclusion:

1. In the real world many times we encounter a situation when we can't determine whether the state is true or false, their fuzzy logic provides a very valuable flexibility for reasoning. In this way, we can consider the inaccuracies and uncertainties of any situation.
2. In this experiment, i have tried to determine marks on the basis of knowledge and efforts put by the student. Inputs 'knowledge' and 'efforts' have the descriptors poor, average and good. 9 rules have been defined based on the combination of these 3 parameters. Depending on these input values of knowledge and efforts, Marks are predicted.