RAMAIAH UNIVERSITY OF APPLIED SCIENCES

Assignment

Course Code	CSC401A
Course Name	Computational Intelligence
Programme	B.Tech
Department	CSE
Faculty	FET

Name of the Student	Satyajit Ghana
Reg. No.	17ETCS002159
Semester/Year	07/2021
Course Leader(s)	Mr. Sagar U

Declaration Sheet

Student Name	Satyajit Ghana		
Reg. No	17ETCS002159		
Programme	B.Tech	Semester/Year	07/2021
Course Code	CSC401A		
Course Title	Computational Intelligence		
Course Date		to	
Course Leader	Mr. Sagar U		

Declaration

The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly.

Signature of the Student		Date	
Submission date stamp (by Examination & Assessment Section)			

Signature of the Course Leader and date	Signature of the Reviewer and date

Contents

List of Figures

1 Question 1

Solution to Question No. 1 Part A

1.1 A critical review of cognitive abilities of ANNs

> *"… neural networks, genetic algorithms, fuzzy systems, evolutionary programming and artificial life are the building blocks of computational intelligence."* – *Marks*

While the processing element in an artificial neural network (ANN) is generally considered to be the very roughly analogous to a biological neuron, the cell body in an ANN is modeled by a linear activation function. The activation function in general, attempts to enhance the signal contribution received through different dendrons.

But how do we know that our human neuron system uses a firing function to start the neural process? There was this Retinotopic Mapping performed on a Cat, basically a cat was strapped to a place, and the brain was injected with a radioactive fluid, now an image was shown to the cat, and it was killed as soon as the cat saw the image, the cat's brain was laid out flat and X-Ray was taken of it, to a surprise the image had a complete print over the brain, as shown below,

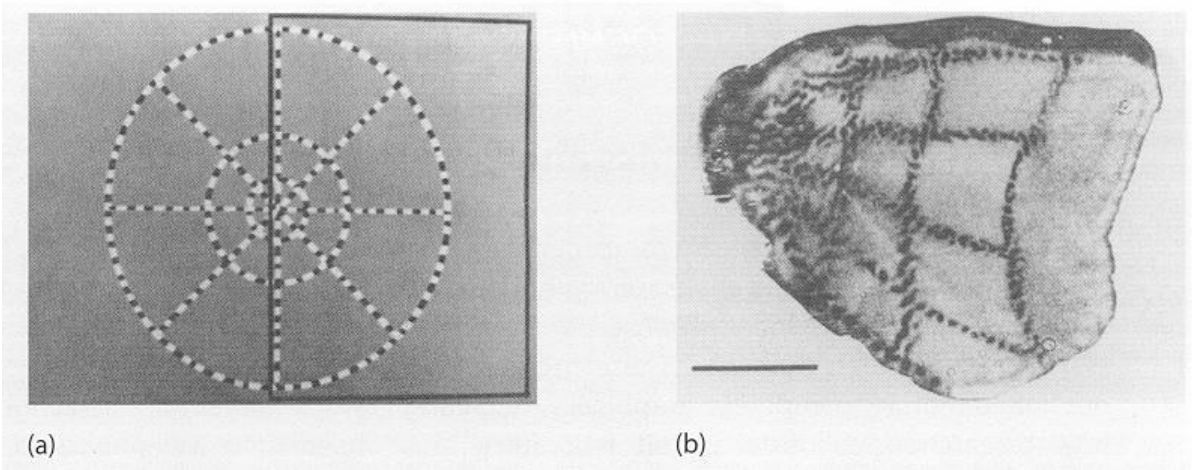


Figure 1 Retinotopic Mapping of a Cat's Cortex

This experiment paved the way that Neural Networks would be the future of Artificial Intelligence, even in our modern neural network architectures for computer vision specifically, the entire input image is fed to the network, and several convolutions are performed to make sense of the input and get some meaningful output.

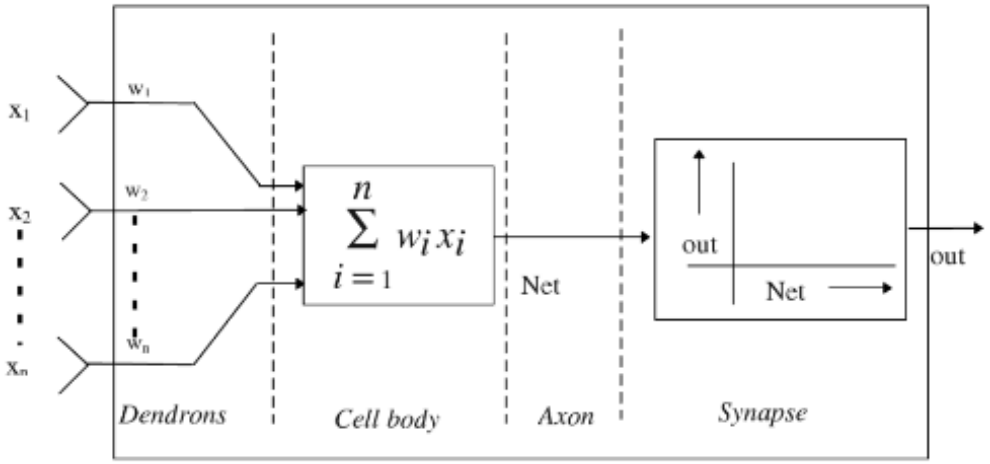


Figure 2 An electrical equivalent of the biological neuron

Abstraction comes naturally to the human brain. Neural networks have to work for it. As with the brain, neural networks are made of building blocks called "neurons" that are connected in various ways. (The neurons in a neural network are inspired by neurons in the brain but do not imitate them directly.) Each neuron might represent an attribute, or a combination of attributes, that the network considers at each level of abstraction.

When joining these neurons together, engineers have many choices to make. They have to decide how many layers of neurons the network should have (or how "deep" it should be). Consider, for example, a neural network with the task of recognizing objects in images. The image enters the system at the first layer. At the next layer, the network might have neurons that simply detect edges in the image. The next layer combines lines to identify curves in the image. Then the next layer combines curves into shapes and textures, and the final layer processes shapes and textures to reach a conclusion about what it's looking at: woolly mammoth!

So, if you have a specific task in mind, how do you know which neural network architecture will accomplish it best? There are some broad rules of thumb. For image-related tasks, engineers typically use "convolutional" neural networks, which feature the same pattern of connections between layers repeated over and over. For natural language processing — like speech recognition, or language generation — engineers have found that "recurrent" neural networks seem to work best. In these, neurons can be connected to non-adjacent layers.

Researchers today describe wide, flat networks as "expressive," meaning that they're capable in theory of capturing a richer set of connections between possible inputs (such as an image) and outputs (such as descriptions of the image). Yet these networks are extremely difficult to train, meaning it's almost impossible to teach them how to actually produce those outputs. They're also more computationally intensive than any computer can handle. (Harnett, 2019)

1.2 The application areas in which ANNs have been successful in delivering human-like performance

Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains. Specifically, neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analog. ANNs have given rise to Deep Learning, which is a broader view of ANNs, the performance benefits of DNNs are just phenomenal, which only caveat that DNNs require huge amounts of data.

1. Self-Driving Cars

The last decade witnessed increasingly rapid progress in self-driving vehicle technology, mainly backed up by advances in the area of deep learning and artificial intelligence. Self-driving cars are autonomous decision-making systems that process streams of observations coming from different on-board sources, such as cameras, radars, LiDARs, ultrasonic sensors, GPS units and/or inertial sensors. These observations are used by the car's computer to make driving decisions.

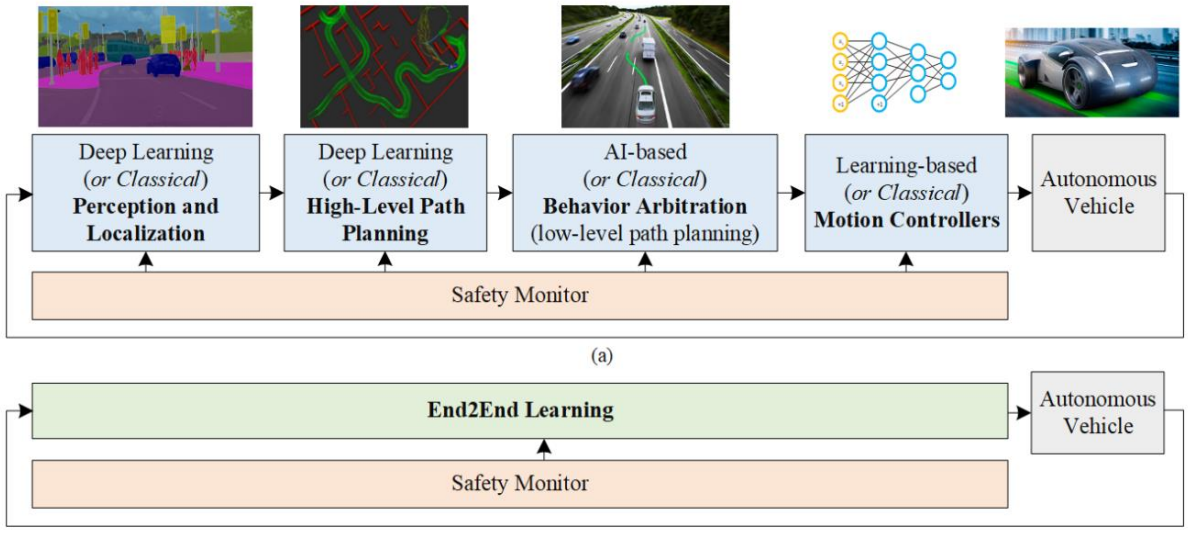


Figure 3 Deep Learning based Self Driving Car

2. Pixel Restoration

The concept of zooming into videos beyond its actual resolution was unrealistic until Deep Learning came into play. In 2017, Google Brain researchers trained a Deep Learning network to take very low resolution images of faces and predict the person's face through it. This method was known as the Pixel Recursive Super Resolution. It enhances the resolution of photos significantly, pinpointing prominent features in order that is just enough for personality identification.

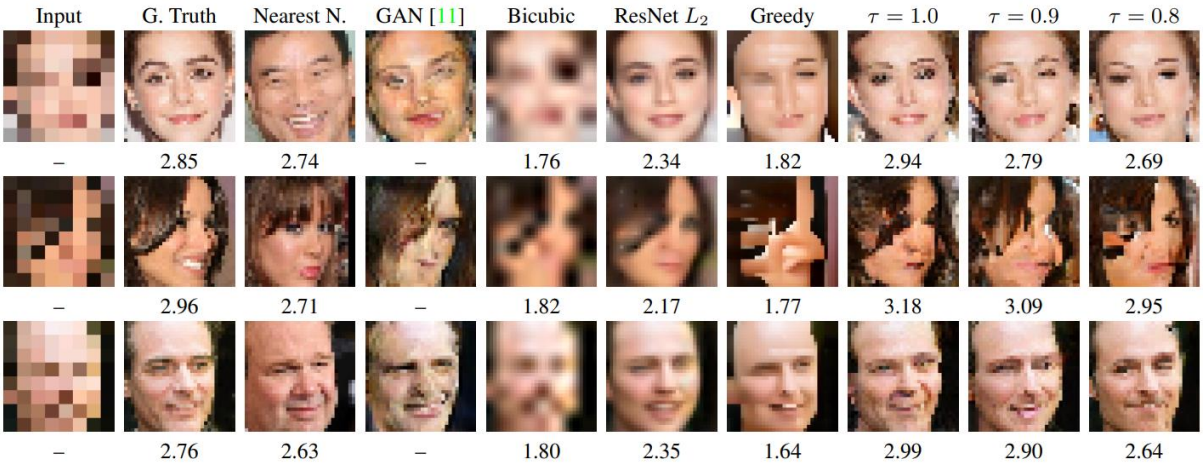
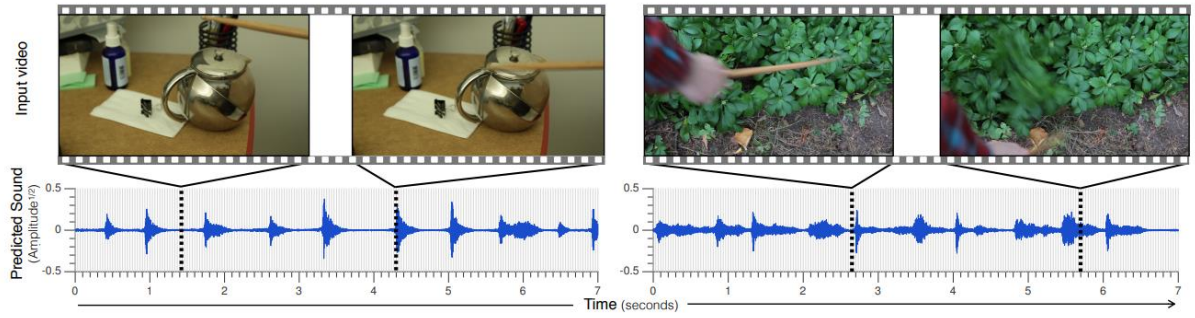


Figure 4 Pixel Restoration

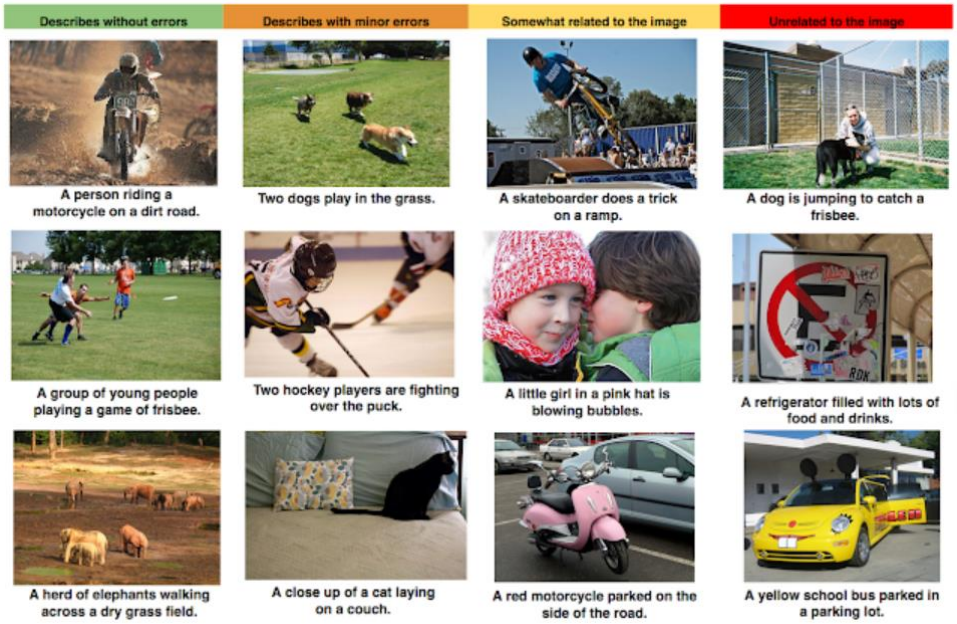
3. Adding Sound to Images

An application of both convolutional neural networks and LSTM recurrent neural networks involves synthesizing sounds to match silent videos. A deep learning model tends to associate the video frames with a database of pre-recorded sounds to select appropriate sounds for the scene. This task is done using training 1000 videos – that have drum sticks sound striking on different surfaces and creating different sounds. These videos are then used by Deep learning models to predict the best suited sound in the video. And later to predict if the sound is fake or real, a Turing-test like setup is built to achieve the best results.



4. Image Caption Generation

Automatic image captioning is the task where given an image the system must generate a caption that describes the contents of the image. Generally, the systems involve the use of very large convolutional neural networks for the object detection in the photographs and then a recurrent neural network like an LSTM to turn the labels into a coherent sentence.



5. Machine Translation

The technology behind Google Translate is called Machine Translation. It has changed the world by allowing people to communicate when it wouldn't otherwise be possible. It turns out that over the past two years, deep learning has totally rewritten our approach to machine translation. Deep learning researchers who know almost nothing about language translation are throwing together relatively simple machine learning solutions that are beating the best expert-built language translation systems in the world.

Context →	In no case may they be used for commercial purposes. =
Target Completion →	Keinesfalls dürfen diese für den kommerziellen Gebrauch verwendet werden.

Figure G.37: Formatted dataset example for En→De

Context →	Analysis of instar distributions of larval I. verticalis collected from a series of ponds also indicated that males were in more advanced instars than females. =
Target Completion →	L'analyse de la distribution de fréquence des stades larvaires d'I. verticalis dans une série d'étangs a également démontré que les larves mâles étaient à des stades plus avancés que les larves femelles.

Figure G.38: Formatted dataset example for En→Fr

Context →	L'analyse de la distribution de fréquence des stades larvaires d'I. verticalis dans une série d'étangs a également démontré que les larves mâles étaient à des stades plus avancés que les larves femelles. =
Target Completion →	Analysis of instar distributions of larval I. verticalis collected from a series of ponds also indicated that males were in more advanced instars than females.

Figure G.39: Formatted dataset example for Fr→En

Figure 5 Machine Translation example from GPT3

1.3 Conclusion

With the way AI and machine learning is being adopted by companies today, we could see more advancements in the applications of neural networks. There will be personalized choices for users all over the world. All mobile and web applications try to give you an enhanced customized experience based on your search history. Hyper-intelligent virtual assistants will make life easier. If you have ever used Google assistant, Siri, or any of those assistants, you

can comprehend how they're slowly evolving. They may even predict your email response in the future.

We can expect a few intriguing discoveries on algorithms to support learning methods. We are just in the infant stage of applying artificial intelligence and neural networks to the real world. Neural networks will be a lot faster in the future, and neural network tools can get embedded in every design surface. We already have a little mini neural network that plugs into an inexpensive processing board, or even into your laptop. Focusing on the hardware, instead of the software, would make devices even faster.

Neural networks will find its usage in the field of medicine, agriculture, physics, discoveries, and everything else you can imagine. Neural networks are also used in shared data systems.

2 Question 2

Solution to Question 1 Part B

2.1 A model of generalized fuzzy decision system for project evaluation

Pandey et al. (2015) proposed a fuzzy logic based grading system for student projects using quality attributes, most of this assignment is inspired from their proposed solution.

But first, we build the whole pipeline of how the whole system would work,

Fuzzy logic is branch of logic specially designed for representing knowledge and human reasoning in such a way that it amenable to processing by a computer. Fuzziness pertains to uncertainty associated with a system i.e. the fact that nothing can be predicted with exact precision. Fuzziness is property of language. Its main source is the imprecision in defining and using symbol. A fuzzy set is a collection of distinct elements with a varying degree of relevance or inclusion. There are two commonly used ways of denoting fuzzy sets. **(Pandey et al, 2015)**

A fuzzy variable has a crisp value which takes on some number over a pre-defined domain (in fuzzy logic terms, called a universe). The crisp value is how we think of the variable using normal mathematics. For example, if my fuzzy variable was how much to tip someone, it's universe would be 0 to 25% and it might take on a crisp value of 15%.

If X is the Universe of discourse and x is a particular element of X, then a fuzzy set A defined on X may be written as a collection of ordered pairs:

$$A = \{\ x, \mu_A(A)\ \}, x \in X$$

Where each pair $x, \mu_A(A)$ is called a singleton, where x is followed by its membership function $\mu_A(A)$,

Singleton can also be written as $\mu_A(x)/x$ and fuzzy set A can also be represented as.

$$A = \sum_{x_i} \frac{x_i}{\mu_A(x_i)}$$

Membership function also known as characteristic function can take value between 0 and 1 and indicates degree of membership. Since there are infinite numbers between 0 and 1, infinite degrees of membership are possible.

For this assignment, we need to create a generalized system for project evaluation, where students show their project with their project report, and power point presentation. Experts judge on different criteria, and assign an opinion as a linguistic term (excellent, very good, good, average or bad). Their opinion has to be combined for grading the project.

To solve this, first a set of project attributes are selected, and their possible linguistic terms are selected. A fuzzy decision set is formed which indicate expert opinion for each project attributes. Fuzzy set is then defined for each attribute. A fuzzy subset is formed for all linguistic terms. For example, presentation can be Excellent, Very Good, Good, Average, Fair or Bad. Similarly, Modularity can be High, Medium, Low, Very Low or Nil.

Following are the attributes and their linguistic terms used to denote the evaluation of a student project,

1. Documentation
    a. Excellent (E)
    b. Very Good (VG)
    c. Satisfying (S)
    d. Moderate (D)
    e. Limited (L)
    f. Bad (B)
2. Presentation
    a. Excellent (E)
    b. Very Good (VG)

c. Good (G)
d. Average (AV)
e. Fair (F)
f. Bad (B)

3. Security/Authentication

a. Excellent (E)
b. Very Good (VG)
c. Good (G)
d. Average (AV)
e. Fair (F)
f. Bad (B)

4. Functionality

a. Excellent (E)
b. Very Good (VG)
c. Good (G)
d. Average (AV)
e. Fair (F)
f. Bad (B)

5. Modularity

a. Very High (VH)
b. High (H)
c. Medium (M)
d. Low (L)
e. Very Low (VL)
f. Nil (N)

6. Design of User Interface

a. Excellent (E)
b. Very Good (VG)
c. Good (G)
d. Average (AV)

e. Fair (F)

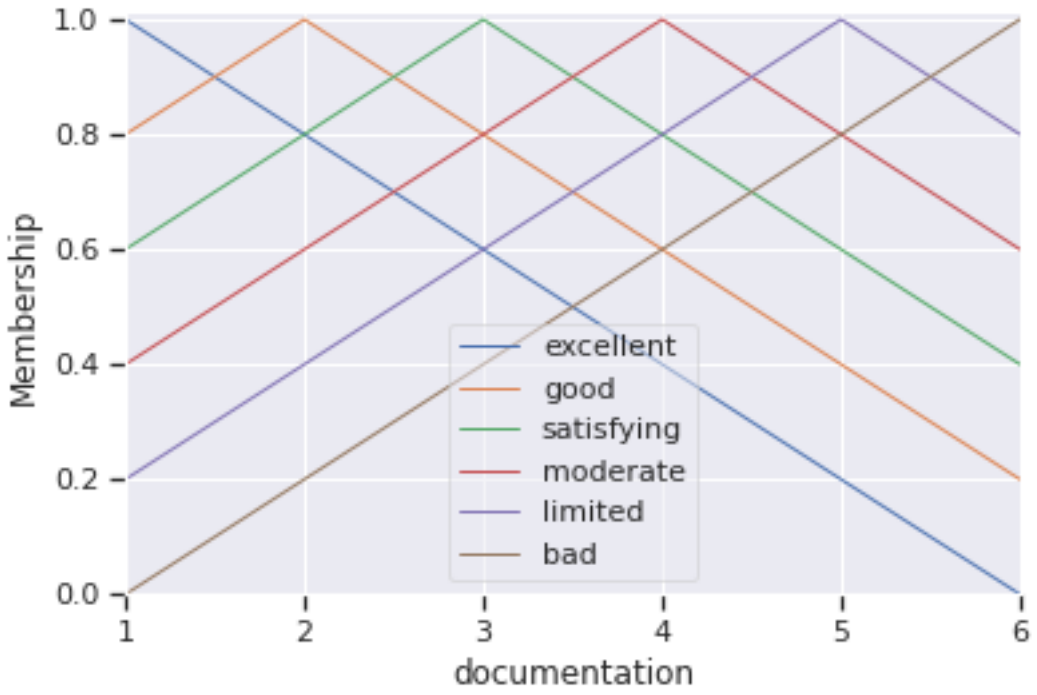
f. Bad (B)

Now that these linguistic terms are fixed, we can assign a membership function to each of them, i.e. the fuzzy subset representation to them. Further we will proceed by creating a logic of how to aggregate these opinions into numeric values and then defuzzify the value to assign a final grade to the student like Ex, A+, A, B, B+, or C.

2.2 Fuzzy sets with attributes

Following are the six attributes and considerable corresponding linguistic terms that are selected for project evaluation along with their fuzzy subsets:

1. Documentation
    a. Excellent (E): `1/1.0+2/0.8+3/0.6+4/0.4+5/0.2+6/0.0`
    b. Very Good (VG): `1/0.8+2/1.0+3/0.8+4/0.6+5/0.4+6/0.2`
    c. Satisfying (S): `1/0.6+2/0.8+3/1.0+4/0.8+5/0.6+6/0.4`
    d. Moderate (D): `1/0.4+2/0.6+3/0.8+4/1.0+5/0.8+6/0.6`
    e. Limited (L): `1/0.2+2/0.4+3/0.6+4/0.8+5/1.0+6/0.8`
    f. Bad (B): `1/0.0+2/0.2+3/0.4+4/0.6+5/0.8+6/1.0`



2. Presentation

    a. Excellent (E): `1/1.0+2/0.8+3/0.6+4/0.4+5/0.2+6/0.0`

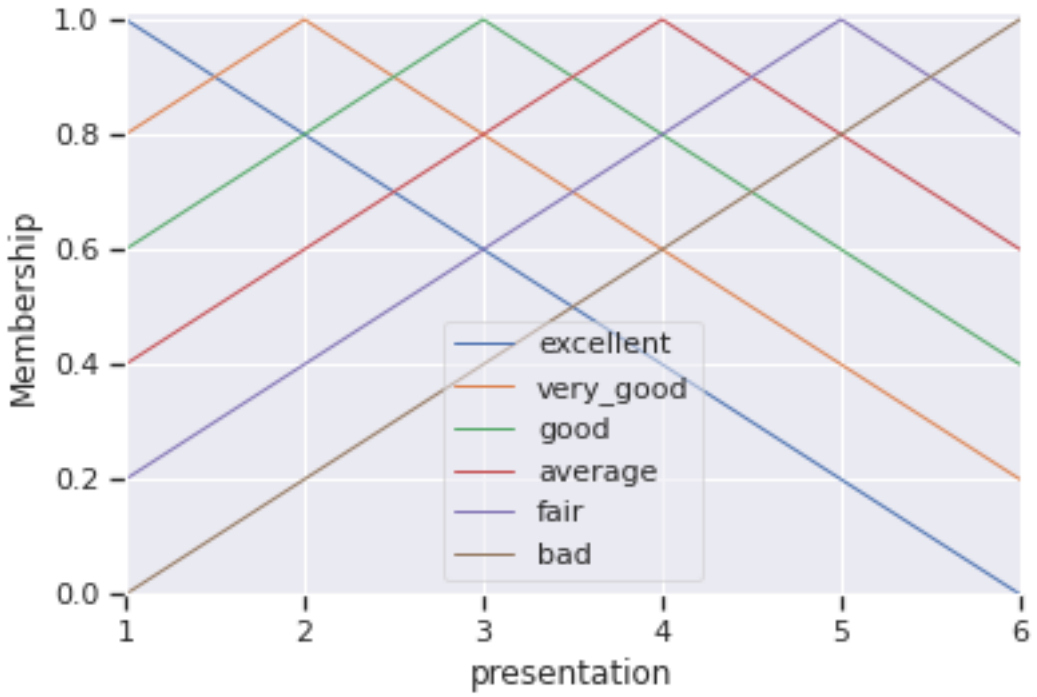
    b. Very Good (VG): `1/0.8+2/1.0+3/0.8+4/0.6+5/0.4+6/0.2`

    c. Good (G): `1/0.6+2/0.8+3/1.0+4/0.8+5/0.6+6/0.4`

    d. Average (AV): `1/0.4+2/0.6+3/0.8+4/1.0+5/0.8+6/0.6`

    e. Fair (F): `1/0.2+2/0.4+3/0.6+4/0.8+5/1.0+6/0.8`

    f. Bad (B): `1/0.0+2/0.2+3/0.4+4/0.6+5/0.8+6/1.0`



3. Security/Authentication

    a. Excellent (E): `1/1.0+2/0.8+3/0.6+4/0.4+5/0.2+6/0.0`

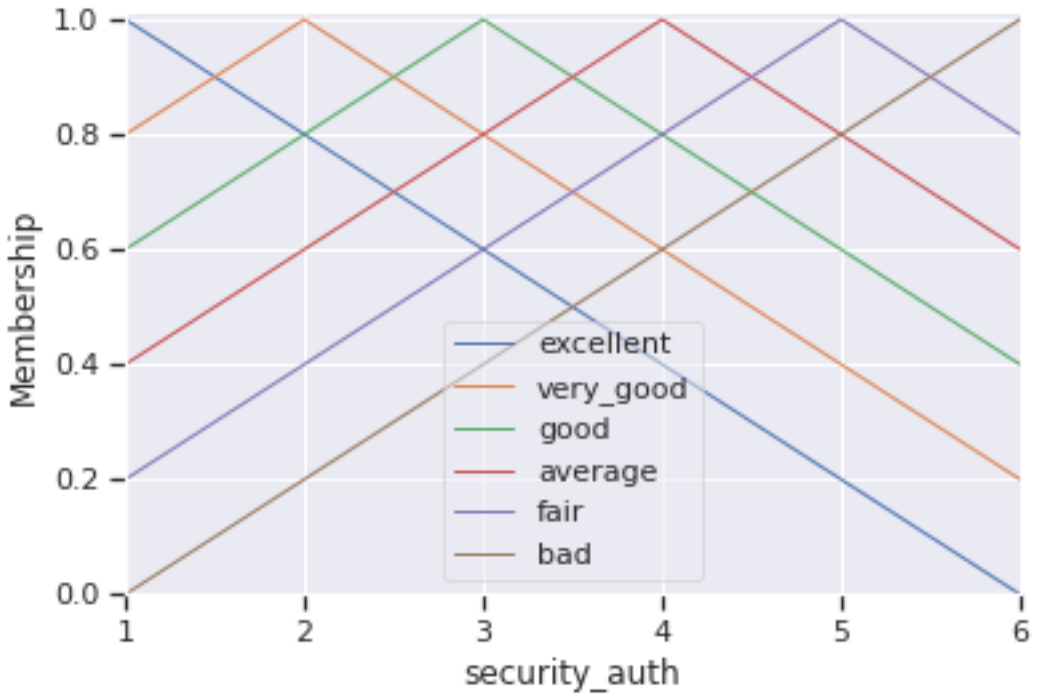
    b. Very Good (VG): `1/0.8+2/1.0+3/0.8+4/0.6+5/0.4+6/0.2`

    c. Good (G): `1/0.6+2/0.8+3/1.0+4/0.8+5/0.6+6/0.4`

    d. Average (AV): `1/0.4+2/0.6+3/0.8+4/1.0+5/0.8+6/0.6`

    e. Fair (F): `1/0.2+2/0.4+3/0.6+4/0.8+5/1.0+6/0.8`

    f. Bad (B): `1/0.0+2/0.2+3/0.4+4/0.6+5/0.8+6/1.0`



4. Functionality

    a. Excellent (E): `1/1.0+2/0.8+3/0.6+4/0.4+5/0.2+6/0.0`

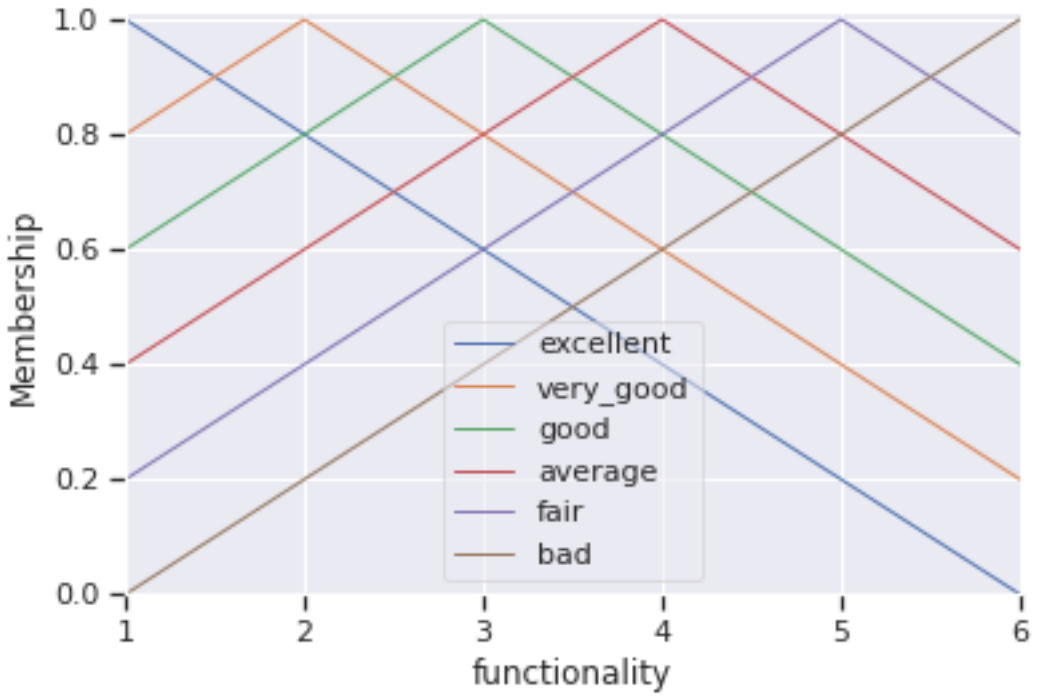
    b. Very Good (VG): `1/0.8+2/1.0+3/0.8+4/0.6+5/0.4+6/0.2`

    c. Good (G): `1/0.6+2/0.8+3/1.0+4/0.8+5/0.6+6/0.4`

    d. Average (AV): `1/0.4+2/0.6+3/0.8+4/1.0+5/0.8+6/0.6`

    e. Fair (F): `1/0.2+2/0.4+3/0.6+4/0.8+5/1.0+6/0.8`

    f. Bad (B): `1/0.0+2/0.2+3/0.4+4/0.6+5/0.8+6/1.0`



5. Modularity

    a. Very High (VH): `1/1.0+2/0.8+3/0.6+4/0.4+5/0.2+6/0.0`

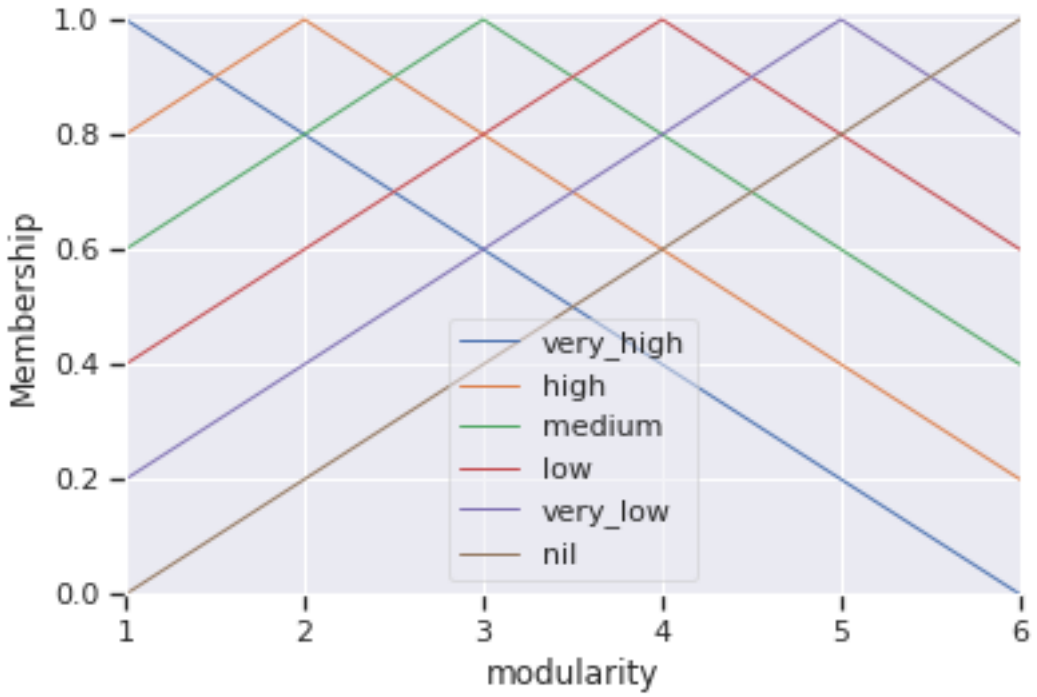
    b. High (H): `1/0.8+2/1.0+3/0.8+4/0.6+5/0.4+6/0.2`

    c. Medium (M): `1/0.6+2/0.8+3/1.0+4/0.8+5/0.6+6/0.4`

    d. Low (L): `1/0.4+2/0.6+3/0.8+4/1.0+5/0.8+6/0.6`

    e. Very Low (VL): `1/0.2+2/0.4+3/0.6+4/0.8+5/1.0+6/0.8`

    f. Nil (N): `1/0.0+2/0.2+3/0.4+4/0.6+5/0.8+6/1.0`



6. Design of User Interface

    a. Excellent (E): `1/1.0+2/0.8+3/0.6+4/0.4+5/0.2+6/0.0`

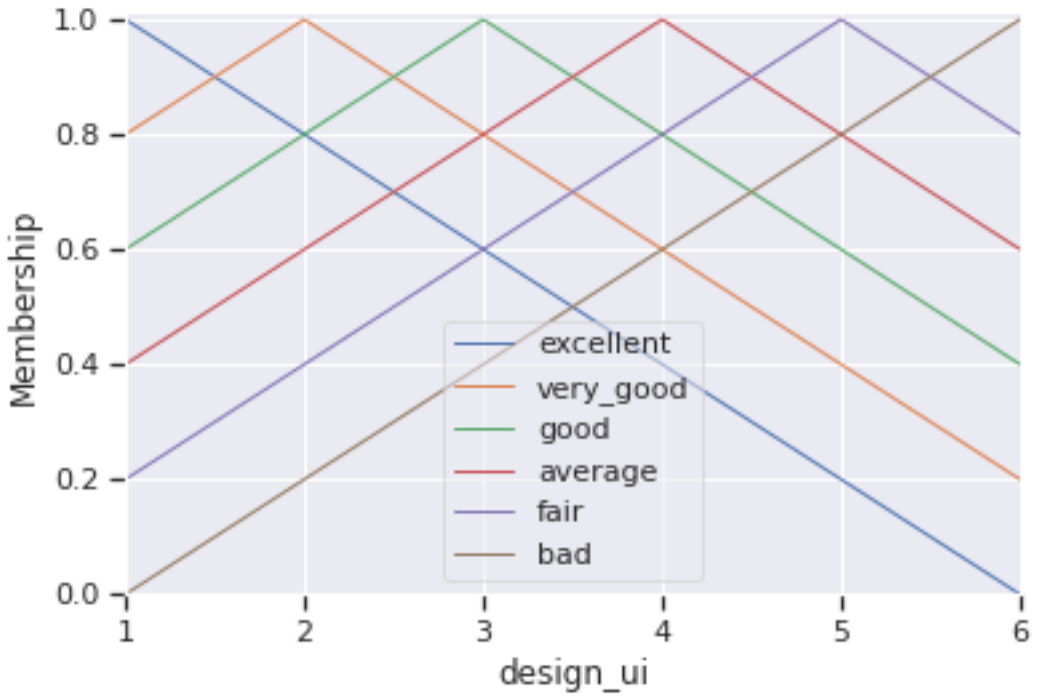
    b. Very Good (VG): `1/0.8+2/1.0+3/0.8+4/0.6+5/0.4+6/0.2`

    c. Good (G): `1/0.6+2/0.8+3/1.0+4/0.8+5/0.6+6/0.4`

    d. Average (AV): `1/0.4+2/0.6+3/0.8+4/1.0+5/0.8+6/0.6`

    e. Fair (F): `1/0.2+2/0.4+3/0.6+4/0.8+5/1.0+6/0.8`

    f. Bad (B): `1/0.0+2/0.2+3/0.4+4/0.6+5/0.8+6/1.0`



2.3 Dataset (test cases) generation

Note: Some of the not-so-important code is omitted, to view the entire code, refer Appendix

We first define our project attributes, so that it can easily be referenced at later parts of our program

```
project_features = dict(
    documentation=['excellent', 'good', 'satisfying', 'moderate', 'limited', 'bad'],
    presentation=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'],
    security_auth=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'],
    functionality=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'],
    design_ui=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'],
    modularity=['very_high', 'high', 'medium', 'low', 'very_low', 'nil']
)
```

```
MAX_STUDENTS = 50
MAX_EVALUATORS = 3

dataset = {}

for evaluator in range(MAX_EVALUATORS):
    df = pd.DataFrame(data={
                        feat: [choice(attrs) for _ in range(MAX_STUDENTS)]
                        for feat, attrs in project_features.items()
                    }, index=[f'student_{num+1:02}' for num in range(MAX_STUDENTS)], dtype ="category")

    dataset[f'evaluator_{evaluator:02}'] = df.copy()
```

Now we can print the dataset and visualize the table

```
for evaluator, data in dataset.items():
    print(f"Data for {evaluator}")
    display(data.head())
    print()
```

Data for evaluator_00

	documentation	presentation	security_auth	functionality	design_ui	modularity
student_01	bad	very_good	bad	bad	fair	high
student_02	satisfying	average	bad	fair	good	very_low
student_03	excellent	bad	excellent	good	good	medium
student_04	satisfying	good	average	very_good	bad	medium
student_05	moderate	average	bad	bad	bad	medium

Data for evaluator_01

	documentation	presentation	security_auth	functionality	design_ui	modularity
student_01	limited	fair	excellent	average	very_good	low
student_02	limited	excellent	bad	good	very_good	very_low
student_03	good	average	fair	average	good	very_low
student_04	good	fair	very_good	fair	good	medium
student_05	moderate	average	bad	average	good	low

Data for evaluator_02

	documentation	presentation	security_auth	functionality	design_ui	modularity
student_01	good	excellent	very_good	excellent	very_good	high
student_02	bad	fair	bad	bad	good	high
student_03	limited	excellent	fair	good	bad	high
student_04	bad	excellent	average	fair	excellent	nil
student_05	satisfying	average	very_good	excellent	average	low

The above is the first 5 samples of each of the evaluators, in total we have 3 evaluators and 50 students, that is in total of 50 test case (each student = one test case).

2.4 Fuzzy Logic System Creation

We'll be using `skfuzzy`, an easy to use fuzzy logic library for python. Each of our attribute (document, design ...) needs a fuzzy set with it, and for that we require a membership function, this membership function can be created using the fuzzy set values we defined earlier.

Following is a helper function that converts the fuzzy set values into 3 points that define a triangular membership function.

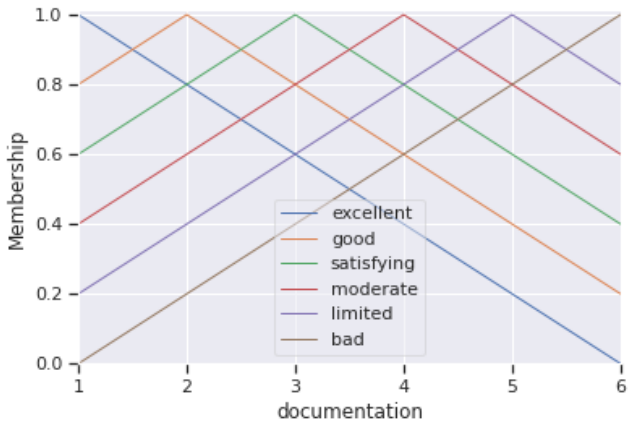
```python
def get_tri_vals(vals, x_min=1, x_max=6):
    start_val = vals[0]
    end_val = vals[-1]
    max_point, max_x = max(vals), vals.index(max(vals))
    max_x += 1

    # x = (y - y1)*(x2 - x1)/(y2 - y1) + x1

    # (x1   , y1       ), (x2    , y2      )
    # (x_min, start_val), (max_x, max_point)
    line_one_fun = lambda y: (y - start_val) * (max_x - x_min) / (max_point - start_val) + x_min

    # (x1   , y1       ), (x2   , y2      )
    # (max_x, max_point), (x_max, end_val)
    line_two_fun = lambda y: (y - max_point) * (x_max - max_x) / (end_val - max_point) + max_x

    if max_point == start_val:
        x_left = x_min
    else:
        x_left = line_one_fun(0)

    if end_val == max_point:
        x_right = x_max
    else:
        x_right = line_two_fun(0)

    return [x_left, max_x, x_right]
```

```python
FuzzySet = namedtuple('FuzzySet', field_names=['attrs', 'set'])

fuzzy_sets = defaultdict(dict)
for proj_feat, proj_attrs in project_features.items():
    num_terms = len(proj_attrs)

    fuzzy_grid = get_fuzzy_set_grid(0, 1, grid_size=num_terms)

    # create the fuzzy set
    proj_set = ctrl.Antecedent(np.arange(1, num_terms + 1, 1), proj_feat)

    fuzzy_sets[proj_feat] = FuzzySet(proj_attrs, proj_set)

    # assign the membership function
    for idx, attr in enumerate(fuzzy_sets[proj_feat].attrs):
        fuzzy_sets[proj_feat].set[attr] = fuzz.trimf(fuzzy_sets[proj_feat].set.universe,
                                                     get_tri_vals(list(fuzzy_grid[idx, :])))
```

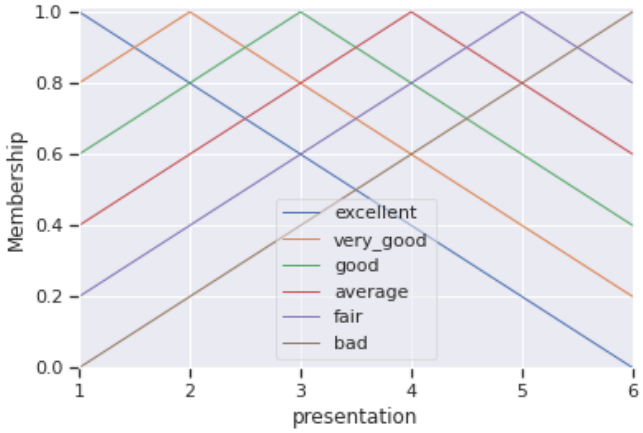
```
defaultdict(dict,
            {'design_ui': FuzzySet(attrs=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'], set=Antecedent: design_ui),
             'documentation': FuzzySet(attrs=['excellent', 'good', 'satisfying', 'moderate', 'limited', 'bad'], set=Antecedent: documentation),
             'functionality': FuzzySet(attrs=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'], set=Antecedent: functionality),
             'modularity': FuzzySet(attrs=['very_high', 'high', 'medium', 'low', 'very_low', 'nil'], set=Antecedent: modularity),
             'presentation': FuzzySet(attrs=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'], set=Antecedent: presentation),
             'security_auth': FuzzySet(attrs=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'], set=Antecedent: security_auth)})
```

Also, we can view the membership functions,

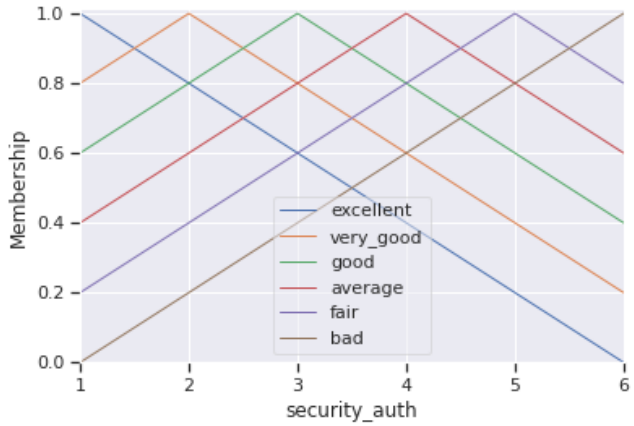
```
1 fuzzy_sets['documentation'].set.view()
```



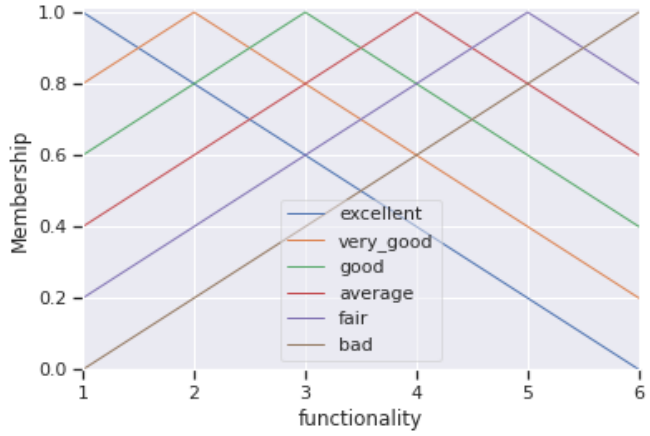

```
1 fuzzy_sets['presentation'].set.view()
```



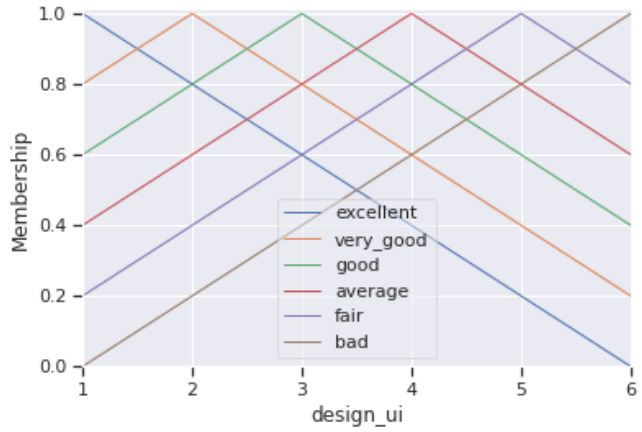

```
1 fuzzy_sets['security_auth'].set.view()
```



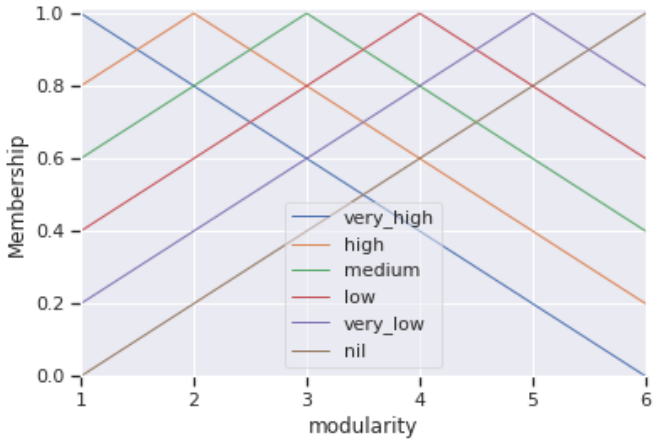

```
1 fuzzy_sets['functionality'].set.view()
```




```
1 fuzzy_sets['design_ui'].set.view()
```




```
1 fuzzy_sets['modularity'].set.view()
```



Now we define the ideal performance of each of them,

```
ideal_performance_sets = {
    'documentation': fuzzy_sets['documentation'].set['excellent'].mf,
    'presentation': fuzzy_sets['presentation'].set['excellent'].mf,
    'security_auth': fuzzy_sets['security_auth'].set['excellent'].mf,
    'functionality': fuzzy_sets['functionality'].set['excellent'].mf,
    'modularity': fuzzy_sets['modularity'].set['very_high'].mf,
    'design_ui': fuzzy_sets['design_ui'].set['excellent'].mf
}
```

```
{'design_ui': array([1. , 0.8, 0.6, 0.4, 0.2, 0. ]),
 'documentation': array([1. , 0.8, 0.6, 0.4, 0.2, 0. ]),
 'functionality': array([1. , 0.8, 0.6, 0.4, 0.2, 0. ]),
 'modularity': array([1. , 0.8, 0.6, 0.4, 0.2, 0. ]),
 'presentation': array([1. , 0.8, 0.6, 0.4, 0.2, 0. ]),
 'security_auth': array([1. , 0.8, 0.6, 0.4, 0.2, 0. ])}
```

2.5 Evaluating Student Performance

A panel of 3 evaluators are formed that evaluate each of the student performance based on our 6 criteria, these evaluators state their opinion linguistically, and an opinion matrix is formed based on that.

We can demonstrate how this happens, consider the following student, with evaluation for 6 criteria by 3 evaluators,

	evaluator_00	evaluator_01	evaluator_02
documentation	bad	limited	good
presentation	very_good	fair	excellent
security_auth	bad	excellent	very_good
functionality	bad	average	excellent
design_ui	fair	very_good	very_good
modularity	high	low	high

For each of the linguistic value, we get the membership value, and compare the distance from the “ideal” performance we defined before,

```
Evaluation

documentation
overall opinion:  [0.  0.2 0.4 0.6 0.4 0.2]
distance from ideal:  2.4000000000000004

presentation
overall opinion:  [0.2 0.4 0.6 0.4 0.2 0. ]
distance from ideal:  1.2000000000000002

security_auth
overall opinion:  [0.  0.2 0.4 0.4 0.2 0. ]
distance from ideal:  1.8

functionality
overall opinion:  [0.  0.2 0.4 0.4 0.2 0. ]
distance from ideal:  1.8

design_ui
overall opinion:  [0.2 0.4 0.6 0.6 0.4 0.2]
distance from ideal:  1.8

modularity
overall opinion:  [0.4 0.6 0.8 0.6 0.4 0.2]
distance from ideal:  1.5999999999999999
```

$$\text{documentation} = \text{bad} \nearrow \text{limited} \nearrow \text{good}$$

$$\text{presentation} = \text{very good} \nearrow \text{fair} \nearrow \text{excellent}$$

$$\text{security auth} = \text{bad} \nearrow \text{excellent} \nearrow \text{very good}$$

$$\text{functionality} = \text{bad} \nearrow \text{average} \nearrow \text{excellent}$$

$$\text{design ui} = \text{fair} \nearrow \text{very good} \nearrow \text{very good}$$

$$\text{modularity} = \text{high} \nearrow \text{low} \nearrow \text{high}$$

And the distance from ideal is calculated as,

$$d\ O, E\ = \sum_{i=1}^{6} |\mu o\ x_i\ - \mu E(x_i)|$$

	documentation	presentation	security_auth	functionality	design_ui	modularity	total_dist
student_00	2.4	1.2	1.8	1.8	1.8	1.6	10.6

Using this logic, we apply it on all of the students in the dataset,

```
def create_student_df(student_idx, dataset):
    student_remarks = pd.DataFrame(
      [dataset[evaluator].iloc[student_idx] for evaluator in dataset.keys()]
    ).T
    student_remarks.columns=dataset.keys()

    return student_remarks

def evaluate_student(student_remarks, debug=False):
    performance_metric = {'total_dist': 0}
    for idx, evaluation in student_remarks.iterrows():

        if debug:
            print(evaluation.name)

        overall_opinion = np.array(
          list(map(lambda x: fuzzy_sets[evaluation.name].set[x].mf, evaluation.values))
        ).min(axis=0)

        distance_from_ideal = np.abs(ideal_performance_sets[evaluation.name] - overall_opinion).sum()
        performance_metric[evaluation.name] = distance_from_ideal
        performance_metric['total_dist'] += distance_from_ideal

        if debug:
            print('overall opinion: ', overall_opinion)
            print('distance from ideal: ', distance_from_ideal)
            print()

    return performance_metric

// Photo by Siim Lukka on Unsplash
```

```
student_evaluation = pd.DataFrame(data={eval_param: [] for eval_param in project_features.keys()})
student_evaluation['total_dist'] = []
for student_idx in range(0, MAX_STUDENTS):
    student_remarks = create_student_df(student_idx, dataset)
    evaluation = evaluate_student(student_remarks)
    student_evaluation.loc[f'student_{student_idx:02}'] = evaluation
```

```
1 student_evaluation.sort_values('total_dist', ascending=True)
--INSERT--
```

	documentation	presentation	security_auth	functionality	design_ui	modularity	total_dist
student_41	1.2	0.2	0.2	0.2	1.2	0.2	3.2
student_49	0.8	2.2	1.2	2.4	0.2	0.2	7.0
student_27	1.8	0.2	1.8	0.8	1.8	1.2	7.6
student_33	1.2	1.2	1.2	2.4	0.8	0.8	7.6
student_08	1.2	1.2	1.2	1.8	1.2	1.2	7.8
student_44	1.2	2.4	1.8	0.4	0.8	1.8	8.4
student_16	2.4	1.2	1.8	1.8	0.4	1.2	8.8
student_14	2.4	1.8	2.8	0.8	0.8	0.4	9.0
student_26	0.4	0.4	2.4	1.8	2.4	1.8	9.2
student_17	0.8	1.2	1.6	2.8	1.2	1.8	9.4
student_23	0.0	1.2	1.8	2.4	1.8	2.4	9.6
student_05	1.8	0.8	1.8	1.2	1.8	2.4	9.8
student_36	1.2	2.8	1.8	1.8	1.2	1.2	10.0
student_46	2.4	0.4	2.8	1.8	1.2	1.8	10.4
student_39	1.2	2.4	2.4	1.8	0.8	1.8	10.4
student_37	1.8	0.4	1.2	2.4	2.8	1.8	10.4
student_24	1.8	1.2	3.4	0.4	1.8	1.8	10.4
student_06	2.4	2.4	1.2	0.8	2.4	1.2	10.4
student_48	2.4	0.8	2.8	2.2	1.2	1.2	10.6
student_00	2.4	1.2	1.8	1.8	1.8	1.6	10.6
student_38	2.4	1.8	0.4	1.8	2.4	1.8	10.6

For grading we choose 6 grades,

- **Ex**: Equivalent to Excellent of Documentation, Presentation, Security, Functionality and User Interface and Very High of Modularity
- **A+**: Equivalent to Good of Documentation, Very Good of Presentation, Security, Functionality and User Interface, and High of Modularity
- **A**: Equivalent to Satisfying of Documentation, Good of Presentation, Security, Functionality and User Interface, and Medium of Modularity
- **B**: Equivalent of Moderate of Documentation, Average of Presentation, Security, Functionality and User Interface, and Low of Modularity
- **B+**: Equivalent of Limited of Documentation, Moderate of Presentation, Security, Functionality and User Interface, and Very Low of Modularity
- **C**: Equivalent of Bad of Documentation, Presentation, Security, Functionality and User Interface and Nil of Modularity

Now we compute the distance of Ex and other grades,

```
grading_dists = {
    'Ex': 6 * np.abs(fuzzy_grid[0, :] - fuzzy_grid[0, :]).sum(),
    'A+': 6 * np.abs(fuzzy_grid[0, :] - fuzzy_grid[1, :]).sum(),
    'A':  6 * np.abs(fuzzy_grid[0, :] - fuzzy_grid[2, :]).sum(),
    'B+': 6 * np.abs(fuzzy_grid[0, :] - fuzzy_grid[3, :]).sum(),
    'B':  6 * np.abs(fuzzy_grid[0, :] - fuzzy_grid[4, :]).sum(),
    'C':  6 * np.abs(fuzzy_grid[0, :] - fuzzy_grid[5, :]).sum()
}
```

Here, the FuzzyGrid is as follows,

```
1 get_fuzzy_set_grid(0, 1, grid_size=6)

--VISUAL--

array([[1. , 0.8, 0.6, 0.4, 0.2, 0. ],
       [0.8, 1. , 0.8, 0.6, 0.4, 0.2],
       [0.6, 0.8, 1. , 0.8, 0.6, 0.4],
       [0.4, 0.6, 0.8, 1. , 0.8, 0.6],
       [0.2, 0.4, 0.6, 0.8, 1. , 0.8],
       [0. , 0.2, 0.4, 0.6, 0.8, 1. ]])
```

And the grading distance matrix is,

```
1 sorted_grading_dists = sorted(grading_dists.items(), key=lambda x: x[1])
2 sorted_grading_dists

--INSERT--

[('Ex', 0.0),
 ('A+', 7.199999999999999),
 ('A', 12.0),
 ('B+', 16.8),
 ('B', 19.200000000000003),
 ('C', 21.6)]
```

So,

```
IF total < 7.19 THEN grade = Ex

IF total >= 7.19 AND total < 12.00 THEN grade = A+

IF total >= 12.0 AND total < 16.80 THEN grade = A

IF total >= 16.80 AND total < 19.20 THEN grade = B+

IF total >= 19.20 AND total < 21.60 THEN grade = B

IF total >= 21.60 THEN grade = C
```

We can apply the above logic in our dataset as,

```
student_evaluation['grade'] = pd.cut(
  student_evaluation['total_dist'],
  bins=[gdist[1] for gdist in sorted_grading_dists] + [np.inf],
  labels=[gdist[0] for gdist in sorted_grading_dists]
)
```

```
[232]  1 student_evaluation.sort_values('total_dist', ascending=True)[['total_dist', 'grade']]
```

	total_dist	grade
student_41	3.2	Ex
student_49	7.0	Ex
student_27	7.6	A+
student_33	7.6	A+
student_08	7.8	A+
student_44	8.4	A+
student_16	8.8	A+
student_14	9.0	A+
student_26	9.2	A+
student_17	9.4	A+
student_23	9.6	A+
student_05	9.8	A+
student_36	10.0	A+
student_46	10.4	A+
student_39	10.4	A+
student_37	10.4	A+
student_24	10.4	A+
student_06	10.4	A+
student_48	10.6	A+
student_00	10.6	A+
student_38	10.6	A+
student_02	11.0	A+
student_31	11.0	A+

The above figure shows the total distance of the student from idea, and also the grade assigned to them based on the distance, lower the distance, higher the grade.

Refer to Results and Conclusion in this Assignment to see the grade of all the 50 students.

2.6 Results

	documentation	presentation	security-auth	functionality	design-ui	modularity	total dist	grade
student_41	1.2	0.2	0.2	0.2	1.2	0.2	3.2	Ex
student_49	0.8	2.2	1.2	2.4	0.2	0.2	7	Ex
student_27	1.8	0.2	1.8	0.8	1.8	1.2	7.6	A+
student_33	1.2	1.2	1.2	2.4	0.8	0.8	7.6	A+
student_08	1.2	1.2	1.2	1.8	1.2	1.2	7.8	A+
student_44	1.2	2.4	1.8	0.4	0.8	1.8	8.4	A+
student_16	2.4	1.2	1.8	1.8	0.4	1.2	8.8	A+
student_14	2.4	1.8	2.8	0.8	0.8	0.4	9	A+
student_26	0.4	0.4	2.4	1.8	2.4	1.8	9.2	A+
student_17	0.8	1.2	1.6	2.8	1.2	1.8	9.4	A+
student_23	0	1.2	1.8	2.4	1.8	2.4	9.6	A+
student_05	1.8	0.8	1.8	1.2	1.8	2.4	9.8	A+
student_36	1.2	2.8	1.8	1.8	1.2	1.2	10	A+
student_46	2.4	0.4	2.8	1.8	1.2	1.8	10.4	A+
student_39	1.2	2.4	2.4	1.8	0.8	1.8	10.4	A+
student_37	1.8	0.4	1.2	2.4	2.8	1.8	10.4	A+
student_24	1.8	1.2	3.4	0.4	1.8	1.8	10.4	A+
student_06	2.4	2.4	1.2	0.8	2.4	1.2	10.4	A+
student_48	2.4	0.8	2.8	2.2	1.2	1.2	10.6	A+
student_00	2.4	1.2	1.8	1.8	1.8	1.6	10.6	A+
student_38	2.4	1.8	0.4	1.8	2.4	1.8	10.6	A+
student_02	1.2	1.8	1.2	2.2	2.8	1.8	11	A+
student_31	3.2	0.4	2.4	1.8	0.8	2.4	11	A+
student_42	1.8	1.2	2.4	2.8	2.4	0.4	11	A+
student_11	2.2	2.8	0.4	2.4	0.8	2.4	11	A+
student_35	1.8	2.8	1.2	1.2	0.8	3.4	11.2	A+
student_19	0.4	1.8	1.6	3.4	1.8	2.4	11.4	A+
student_34	3.2	1.2	0.2	1.2	2.4	3.2	11.4	A+
student_13	1.6	0.8	2.4	1.8	1.6	3.2	11.4	A+
student_03	2.4	1.2	1.6	1.8	1.8	2.8	11.6	A+
student_10	1.2	1.8	2.4	1.8	2.4	2.4	12	A
student_12	2.4	2.4	1.8	1.8	1.8	1.8	12	A
student_28	0.2	2.8	2.4	2.4	1.8	2.4	12	A
student_45	2.4	3.4	2.4	1.2	1.8	0.8	12	A
student_21	1.8	2.8	1.8	1.8	0.8	3.2	12.2	A
student_29	1.2	2.8	1.8	0.8	2.8	2.8	12.2	A
student_07	1.2	2.8	1.2	2.8	1.8	2.4	12.2	A
student_30	1.8	2.4	3.2	3.4	1.2	0.4	12.4	A
student_20	1.2	1.2	3.2	1.2	2.4	3.2	12.4	A
student_22	0.4	1.8	2.4	1.8	3.2	2.8	12.4	A

student_40	3.2	1.2	1.2	1.8	2.4	2.8	12.6	A
student_09	3.4	1.8	2.4	3.2	1.2	0.8	12.8	A
student_15	2.4	2.2	2.8	0.8	1.8	2.8	12.8	A
student_32	1.8	3.4	1.6	2.4	3.4	0.4	13	A
student_01	2.8	1.2	3.6	2.8	1.2	1.8	13.4	A
student_43	0.2	2.4	2.8	3.4	2.2	2.4	13.4	A
student_18	2.8	2.8	1.8	1.8	2.4	1.8	13.4	A
student_04	2.2	2.8	2.4	1.8	2.8	2.2	14.2	A
student_47	3.4	1.8	2.4	2.4	1.8	3.4	15.2	A
student_25	3.6	2.4	2.4	1.8	2	3.4	15.6	A

Bibliography

1. Pandey, M., Srivastava, P.K. and Bhattacharjee, V., 2015. Fuzzy logic based grading system for student projects using quality attributes. International Journal of Engineering and Technology, 7, pp.1304-1308.
2. https://github.com/scikit-fuzzy/scikit-fuzzy
3. https://www.quantamagazine.org/foundations-built-for-a-general-theory-of-neural-networks-20190131/
4. https://www.mygreatlearning.com/blog/deep-learning-applications/
5. https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-neural-network

Appendix

```
In [ ]: !pip install scikit-fuzzy
```

Fuzzy Logic: Grading Student Performance

Author: Satyajit Ghana

Fuzzy Logic Grading System

```
In [97]:
import numpy as np
import pandas as pd
import seaborn as sns
import skfuzzy as fuzz

from skfuzzy import control as ctrl
from random import choice
from collections import defaultdict, namedtuple
from pprint import pprint

from IPython.display import display

sns.set()
```

Project Evaluation Attributes

```
In [33]:
project_features = dict(
    documentation=['excellent', 'good', 'satisfying', 'moderate', 'limited', 'bad'],
    presentation=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'],
    security_auth=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'],
    functionality=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'],
    design_ui=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'],
    modularity=['very_high', 'high', 'medium', 'low', 'very_low', 'nil']
)
```

Dataset

```
In [49]:
MAX_STUDENTS = 50
MAX_EVALUATORS = 3
```

```
In [57]:
dataset = {}

for evaluator in range(MAX_EVALUATORS):
    df = pd.DataFrame(data={
                        feat: [choice(attrs) for _ in range(MAX_STUDENTS)]
                        for feat, attrs in project_features.items()
                    }, index=[f'student_{num+1:02}' for num in range(MAX_STUDENTS)], dtype ="category")

    dataset[f'evaluator_{evaluator:02}'] = df.copy()
```

```
In [58]: dataset.keys()
Out[58]: dict_keys(['evaluator_00', 'evaluator_01', 'evaluator_02'])
```

```
In [68]:
for evaluator, data in dataset.items():
    print(f"Data for {evaluator}")
    display(data.info())
    print()
```

```
Data for evaluator_00
<class 'pandas.core.frame.DataFrame'>
Index: 50 entries, student_01 to student_50
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   documentation  50 non-null     category
 1   presentation   50 non-null     category
 2   security_auth  50 non-null     category
 3   functionality  50 non-null     category
 4   design_ui      50 non-null     category
 5   modularity     50 non-null     category
dtypes: category(6)
memory usage: 1.9+ KB

None

Data for evaluator_01
<class 'pandas.core.frame.DataFrame'>
Index: 50 entries, student_01 to student_50
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   documentation  50 non-null     category
 1   presentation   50 non-null     category
 2   security_auth  50 non-null     category
 3   functionality  50 non-null     category
 4   design_ui      50 non-null     category
 5   modularity     50 non-null     category
dtypes: category(6)
memory usage: 1.9+ KB

None

Data for evaluator_02
<class 'pandas.core.frame.DataFrame'>
Index: 50 entries, student_01 to student_50
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   documentation  50 non-null     category
 1   presentation   50 non-null     category
 2   security_auth  50 non-null     category
 3   functionality  50 non-null     category
 4   design_ui      50 non-null     category
 5   modularity     50 non-null     category
dtypes: category(6)
memory usage: 1.9+ KB

None
```

```
In [67]:
for evaluator, data in dataset.items():
    print(f"Data for {evaluator}")
    display(data.head())
    print()
```

Data for evaluator_00

	documentation	presentation	security_auth	functionality	design_ui	modularity
student_01	bad	very_good	bad	bad	fair	high
student_02	satisfying	average	bad	fair	good	very_low
student_03	excellent	bad	excellent	good	good	medium
student_04	satisfying	good	average	very_good	bad	medium
student_05	moderate	average	bad	bad	bad	medium

Data for evaluator_01

	documentation	presentation	security_auth	functionality	design_ui	modularity
student_01	limited	fair	excellent	average	very_good	low
student_02	limited	excellent	bad	good	very_good	very_low
student_03	good	average	fair	average	good	very_low
student_04	good	fair	very_good	fair	good	medium
student_05	moderate	average	bad	average	good	low

Data for evaluator_02

	documentation	presentation	security_auth	functionality	design_ui	modularity
student_01	good	excellent	very_good	excellent	very_good	high
student_02	bad	fair	bad	bad	good	high
student_03	limited	excellent	fair	good	bad	high
student_04	bad	excellent	average	fair	excellent	nil
student_05	satisfying	average	very_good	excellent	average	low

Fuzzy Logic

```
In [3]:
def get_tri_vals(vals, x_min=1, x_max=6):
    start_val = vals[0]
    end_val = vals[-1]
    max_point, max_x = max(vals), vals.index(max(vals))
    max_x += 1

    # x = (y - y1)*(x2 - x1)/(y2 - y1) + x1

    # (x1    , y1        ), (x2    , y2        )
    # (x_min, start_val), (max_x, max_point)
    line_one_fun = lambda y: (y - start_val) * (max_x - x_min) / (max_point - start_val) + x_min

    # (x1    , y1        ), (x2    , y2        )
    # (max_x, max_point), (x_max, end_val)
    line_two_fun = lambda y: (y - max_point) * (x_max - max_x) / (end_val - max_point) + max_x

    if max_point == start_val:
        x_left = x_min
    else:
        x_left = line_one_fun(0)

    if end_val == max_point:
        x_right = x_max
    else:
        x_right = line_two_fun(0)

    return [x_left, max_x, x_right]
```

```
In [4]: get_tri_vals([1, 0.8, 0.6, 0.4, 0.2, 0])
Out[4]: [1, 1, 6.0]
```

```
In [5]:
def get_fuzzy_set_grid(val_min, val_max, grid_size=6):

    init_vals = np.linspace(val_max, val_min, grid_size)
    grid_vals = np.array([init_vals])

    for val in np.linspace(val_max, val_min, grid_size)[1:]:
        init_vals = init_vals[:-1]
        init_vals = np.concatenate((np.array([val]), init_vals))

        grid_vals = np.append(grid_vals, np.array([init_vals.copy()]), axis=0)

    return grid_vals
```

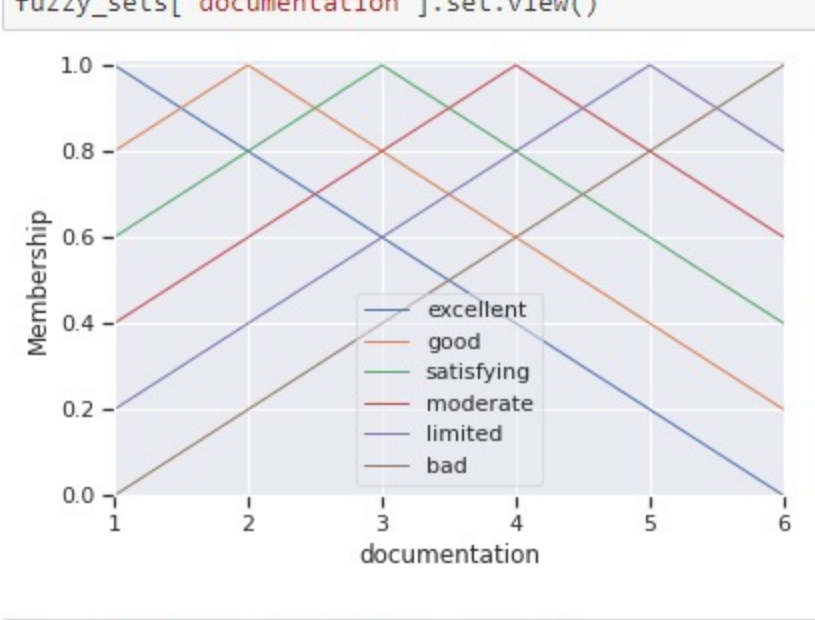
```
In [6]: get_fuzzy_set_grid(0, 1, grid_size=6)
Out[6]: array([[1. , 0.8, 0.6, 0.4, 0.2, 0. ],
               [0.8, 1. , 0.8, 0.6, 0.4, 0.2],
               [0.6, 0.8, 1. , 0.8, 0.6, 0.4],
               [0.4, 0.6, 0.8, 1. , 0.8, 0.6],
               [0.2, 0.4, 0.6, 0.8, 1. , 0.8],
               [0. , 0.2, 0.4, 0.6, 0.8, 1. ]])
```

```
In [74]: FuzzySet = namedtuple('FuzzySet', field_names=['attrs', 'set'])
```

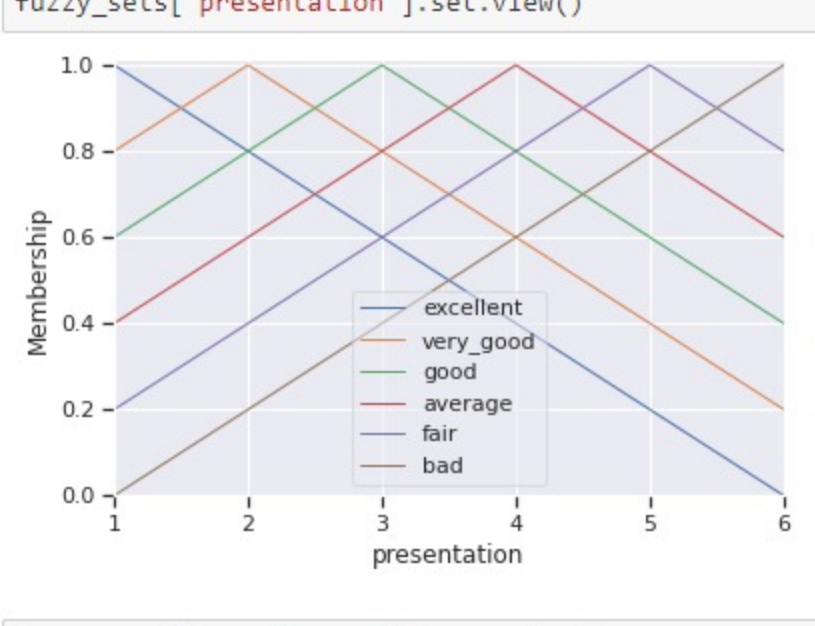
```
In [76]:
fuzzy_sets = defaultdict(dict)
for proj_feat, proj_attrs in project_features.items():
    num_terms = len(proj_attrs)

    fuzzy_grid = get_fuzzy_set_grid(0, 1, grid_size=num_terms)

    # create the fuzzy set
    proj_set = ctrl.Antecedent(np.arange(1, num_terms + 1, 1), proj_feat)

    fuzzy_sets[proj_feat] = FuzzySet(proj_attrs, proj_set)

    # assign the membership function
    for idx, attr in enumerate(fuzzy_sets[proj_feat].attrs):
        fuzzy_sets[proj_feat].set[attr] = fuzz.trimf(fuzzy_sets[proj_feat].set.universe, get_tri_vals(list(fuzzy_grid[idx, :])))
```

```
In [108]: fuzzy_sets
Out[108]: defaultdict(dict,
            {'design_ui': FuzzySet(attrs=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'], set=Antecedent: design_u
i),
             'documentation': FuzzySet(attrs=['excellent', 'good', 'satisfying', 'moderate', 'limited', 'bad'], set=Antecedent:
documentation),
             'functionality': FuzzySet(attrs=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'], set=Antecedent: func
tionality),
             'modularity': FuzzySet(attrs=['very_high', 'high', 'medium', 'low', 'very_low', 'nil'], set=Antecedent: modularit
y),
             'presentation': FuzzySet(attrs=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'], set=Antecedent: prese
ntation),
             'security_auth': FuzzySet(attrs=['excellent', 'very_good', 'good', 'average', 'fair', 'bad'], set=Antecedent: secu
rity_auth)})
```

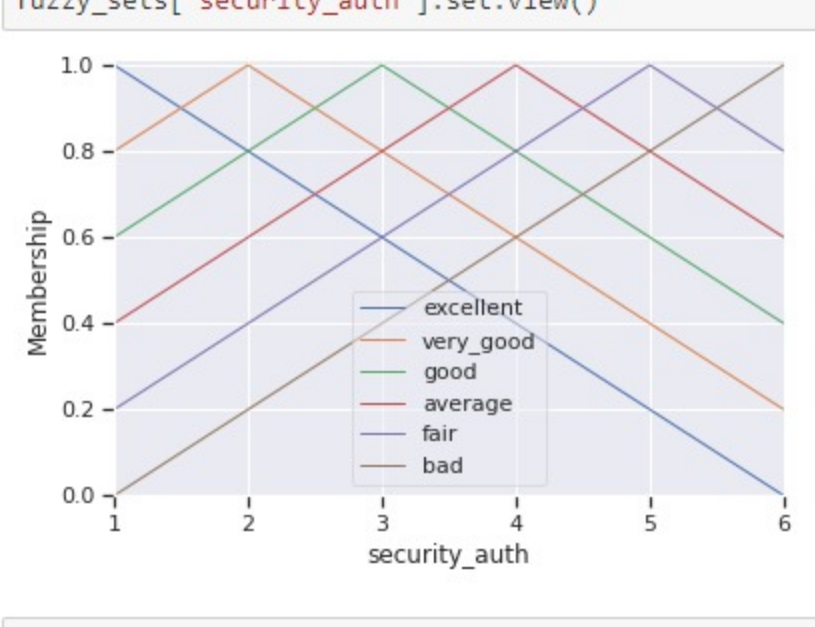
```
In [84]: fuzzy_sets['documentation'].set.view()
```



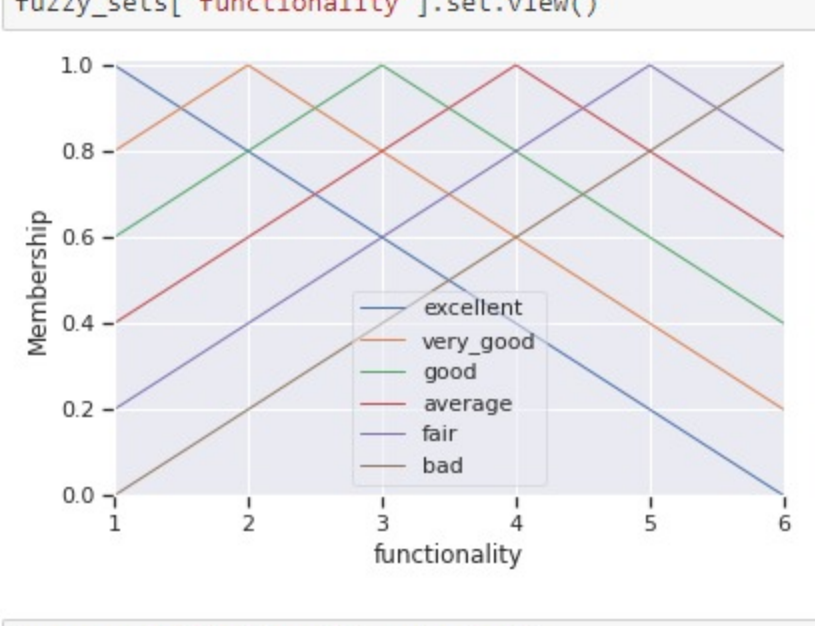

```
In [85]: fuzzy_sets['presentation'].set.view()
```



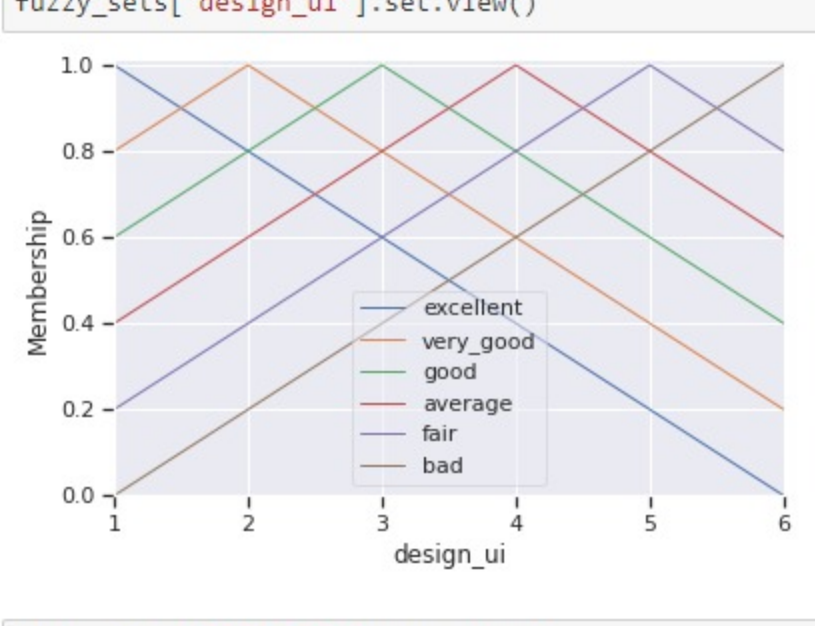

```
In [86]: fuzzy_sets['security_auth'].set.view()
```



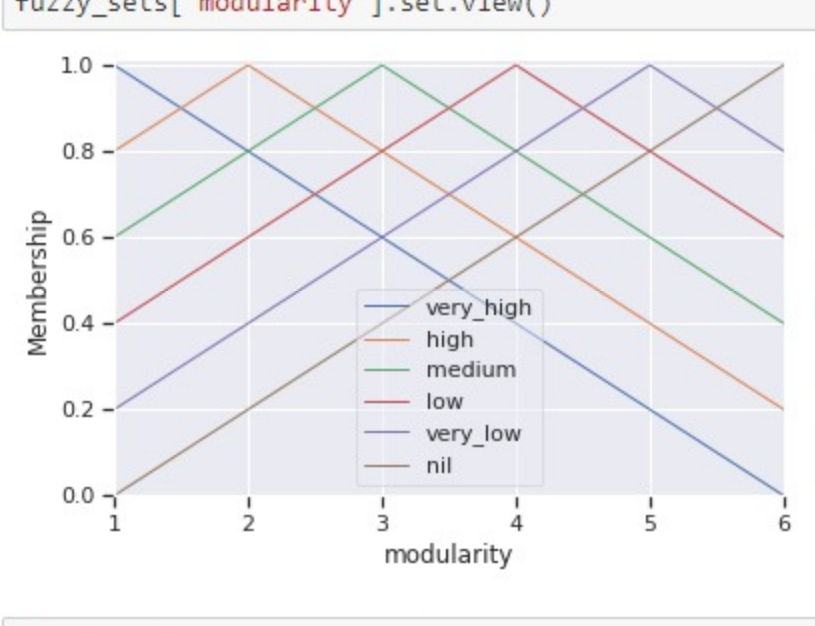

```
In [87]: fuzzy_sets['functionality'].set.view()
```




```
In [88]: fuzzy_sets['design_ui'].set.view()
```




```
In [89]: fuzzy_sets['modularity'].set.view()
```




```
In [94]:
ideal_performance_sets = {
    'documentation': fuzzy_sets['documentation'].set['excellent'].mf,
    'presentation': fuzzy_sets['presentation'].set['excellent'].mf,
    'security_auth': fuzzy_sets['security_auth'].set['excellent'].mf,
    'functionality': fuzzy_sets['functionality'].set['excellent'].mf,
    'modularity': fuzzy_sets['modularity'].set['very_high'].mf,
    'design_ui': fuzzy_sets['design_ui'].set['excellent'].mf
}
```

```
In [106]: pprint(ideal_performance_sets)

{'design_ui': array([1. , 0.8, 0.6, 0.4, 0.2, 0. ]),
 'documentation': array([1. , 0.8, 0.6, 0.4, 0.2, 0. ]),
 'functionality': array([1. , 0.8, 0.6, 0.4, 0.2, 0. ]),
 'modularity': array([1. , 0.8, 0.6, 0.4, 0.2, 0. ]),
 'presentation': array([1. , 0.8, 0.6, 0.4, 0.2, 0. ]),
 'security_auth': array([1. , 0.8, 0.6, 0.4, 0.2, 0. ])}
```

Evaluating Student Performance

```
In [110]: dataset.keys()
Out[110]: dict_keys(['evaluator_00', 'evaluator_01', 'evaluator_02'])
```

Let's see how it works out for one student

```
In [170]:
def create_student_df(student_idx, dataset):
    student_remarks = pd.DataFrame([dataset[evaluator].iloc[student_idx] for evaluator in dataset.keys()]).T
    student_remarks.columns=dataset.keys()

    return student_remarks
```

```
In [199]:
def evaluate_student(student_remarks, debug=False):
    performance_metric = {'total_dist': 0}
    for idx, evaluation in student_remarks.iterrows():
        if debug:
            print(evaluation.name)
        overall_opinion = np.array(list(map(lambda x: fuzzy_sets[evaluation.name].set[x].mf, evaluation.values))).min(axis=0)
        distance_from_ideal = np.abs(ideal_performance_sets[evaluation.name] - overall_opinion).sum()
        performance_metric[evaluation.name] = distance_from_ideal
        performance_metric['total_dist'] += distance_from_ideal

        if debug:
            print('overall opinion: ', overall_opinion)
            print('distance from ideal: ', distance_from_ideal)
            print()

    return performance_metric
```

```
In [200]:
student_evaluation = pd.DataFrame(data={eval_param: [] for eval_param in project_features.keys()})
student_evaluation['total_dist'] = []
for student_idx in range(0, MAX_STUDENTS):
    student_remarks = create_student_df(student_idx, dataset)
    display(student_remarks)

    print("\nEvaluation\n")

    evaluation = evaluate_student(student_remarks, debug=True)

    print(evaluation)
    print()

    student_evaluation.loc[f'student_{student_idx:02}'] = evaluation

    break # do only for one student

display(student_evaluation)
```

	evaluator_00	evaluator_01	evaluator_02
documentation	bad	limited	good
presentation	very_good	fair	excellent
security_auth	bad	excellent	very_good
functionality	bad	average	excellent
design_ui	fair	very_good	very_good
modularity	high	low	high

```
Evaluation

documentation
overall opinion:  [0.  0.2 0.4 0.6 0.4 0.2]
distance from ideal:  2.4000000000000004

presentation
overall opinion:  [0.2 0.4 0.6 0.4 0.2 0. ]
distance from ideal:  1.2000000000000002

security_auth
overall opinion:  [0.  0.2 0.4 0.4 0.2 0. ]
distance from ideal:  1.8

functionality
overall opinion:  [0.  0.2 0.4 0.4 0.2 0. ]
distance from ideal:  1.8

design_ui
overall opinion:  [0.2 0.4 0.6 0.6 0.4 0.2]
distance from ideal:  1.8

modularity
overall opinion:  [0.4 0.6 0.8 0.6 0.4 0.2]
distance from ideal:  1.5999999999999999

{'total_dist': 10.6, 'documentation': 2.4000000000000004, 'presentation': 1.2000000000000002, 'security_auth': 1.8, 'functional
ity': 1.8, 'design_ui': 1.8, 'modularity': 1.5999999999999999}
```

	documentation	presentation	security_auth	functionality	design_ui	modularity	total_dist
student_00	2.4	1.2	1.8	1.8	1.8	1.6	10.6

Now apply the same logic over all the student

```
In [201]:
student_evaluation = pd.DataFrame(data={eval_param: [] for eval_param in project_features.keys()})
student_evaluation['total_dist'] = []
for student_idx in range(0, MAX_STUDENTS):
    student_remarks = create_student_df(student_idx, dataset)
    evaluation = evaluate_student(student_remarks)
    student_evaluation.loc[f'student_{student_idx:02}'] = evaluation
```

```
In [231]: student_evaluation.describe().T
Out[231]:
```

	count	mean	std	min	25%	50%	75%	max
documentation	50.0	1.812	0.911970	0.0	1.20	1.8	2.40	3.6
presentation	50.0	1.752	0.874640	0.2	1.20	1.8	2.40	3.4
security_auth	50.0	1.936	0.794306	0.2	1.30	1.8	2.40	3.6
functionality	50.0	1.868	0.781009	0.2	1.35	1.8	2.40	3.4
design_ui	50.0	1.728	0.749160	0.2	1.20	1.8	2.40	3.4
modularity	50.0	1.912	0.919769	0.2	1.20	1.8	2.40	3.4
total_dist	50.0	11.008	2.179294	3.2	10.10	11.1	12.35	15.6

Display the ranking of students

```
In [203]: student_evaluation.sort_values('total_dist', ascending=True)
Out[203]:
```

	documentation	presentation	security_auth	functionality	design_ui	modularity	total_dist
student_41	1.2	0.2	0.2	0.2	1.2	0.2	3.2
student_49	0.8	2.2	1.2	2.4	0.2	0.2	7.0
student_27	1.8	0.2	1.8	0.8	1.8	1.2	7.6
student_33	1.2	1.2	1.2	2.4	0.8	0.8	7.6
student_08	1.2	1.2	1.2	1.8	1.2	1.2	7.8
student_44	1.2	2.4	1.8	0.4	0.8	1.8	8.4
student_16	2.4	1.2	1.8	1.8	0.4	1.2	8.8
student_14	2.4	1.8	2.8	0.8	0.8	0.4	9.0
student_26	0.4	0.4	2.4	1.8	2.4	1.8	9.2
student_17	0.8	1.2	1.6	2.8	1.2	1.8	9.4
student_23	0.0	1.2	1.8	2.4	1.8	2.4	9.6
student_05	1.8	0.8	1.8	1.2	1.8	2.4	9.8
student_36	1.2	2.8	1.8	1.8	1.2	1.2	10.0
student_46	2.4	0.4	2.8	1.8	1.2	1.8	10.4
student_39	1.2	2.4	2.4	1.8	0.8	1.8	10.4
student_37	1.8	0.4	1.2	2.4	2.8	1.8	10.4
student_24	1.8	1.2	3.4	0.4	1.8	1.8	10.4
student_06	2.4	2.4	1.2	0.8	2.4	1.2	10.4
student_48	2.4	0.8	2.8	2.2	1.2	1.2	10.6
student_00	2.4	1.2	1.8	1.8	1.8	1.6	10.6
student_38	2.4	1.8	0.4	1.8	2.4	1.8	10.6
student_02	1.2	1.8	1.2	2.2	2.8	1.8	11.0
student_31	3.2	0.4	2.4	1.8	0.8	2.4	11.0
student_42	1.8	1.2	2.4	2.8	2.4	0.4	11.0
student_11	2.2	2.8	0.4	2.4	0.8	2.4	11.0
student_35	1.8	2.8	1.2	1.2	0.8	3.4	11.2
student_19	0.4	1.8	1.6	3.4	1.8	2.4	11.4
student_34	3.2	1.2	0.2	1.2	2.4	3.2	11.4
student_13	1.6	0.8	2.4	1.8	1.6	3.2	11.4
student_03	2.4	1.2	1.6	1.8	1.8	2.8	11.6
student_10	1.2	1.8	2.4	1.8	2.4	2.4	12.0
student_12	2.4	2.4	1.8	1.8	1.8	1.8	12.0
student_28	0.2	2.8	2.4	2.4	1.8	2.4	12.0
student_45	2.4	3.4	2.4	1.2	1.8	0.8	12.0
student_21	1.8	2.8	1.8	1.8	0.8	3.2	12.2
student_29	1.2	2.8	1.8	0.8	2.8	2.8	12.2
student_07	1.2	2.8	1.2	2.8	1.8	2.4	12.2
student_30	1.8	2.4	3.2	3.4	1.2	0.4	12.4
student_20	1.2	1.2	3.2	1.2	2.4	3.2	12.4
student_22	0.4	1.8	2.4	1.8	3.2	2.8	12.4
student_40	3.2	1.2	1.2	1.8	2.4	2.8	12.6
student_09	3.4	1.8	2.4	3.2	1.2	0.8	12.8
student_15	2.4	2.2	2.8	0.8	1.8	2.8	12.8
student_32	1.8	3.4	1.6	2.4	3.4	0.4	13.0
student_01	2.8	1.2	3.6	2.8	1.2	1.8	13.4
student_43	0.2	2.4	2.8	3.4	2.2	2.4	13.4
student_18	2.8	2.8	1.8	1.8	2.4	1.8	13.4
student_04	2.2	2.8	2.4	1.8	2.8	2.2	14.2
student_47	3.4	1.8	2.4	2.4	1.8	3.4	15.2
student_25	3.6	2.4	2.4	1.8	2.0	3.4	15.6

```
In [208]:
[fuzzy_sets['documentation'].set['good'].mf,
 fuzzy_sets['presentation'].set['very_good'].mf,
 fuzzy_sets['security_auth'].set['very_good'].mf,
 fuzzy_sets['functionality'].set['very_good'].mf,
 fuzzy_sets['design_ui'].set['very_good'].mf,
 fuzzy_sets['modularity'].set['high'].mf]
Out[208]:
[array([0.8, 1. , 0.8, 0.6, 0.4, 0.2]),
 array([0.8, 1. , 0.8, 0.6, 0.4, 0.2]),
 array([0.8, 1. , 0.8, 0.6, 0.4, 0.2]),
 array([0.8, 1. , 0.8, 0.6, 0.4, 0.2]),
 array([0.8, 1. , 0.8, 0.6, 0.4, 0.2]),
 array([0.8, 1. , 0.8, 0.6, 0.4, 0.2])]
```

```
In [213]:
grading_dists = {
    'Ex': 6 * np.abs(fuzzy_grid[0, :] - fuzzy_grid[0, :]).sum(),
    'A+': 6 * np.abs(fuzzy_grid[0, :] - fuzzy_grid[1, :]).sum(),
    'A':  6 * np.abs(fuzzy_grid[0, :] - fuzzy_grid[2, :]).sum(),
    'B+': 6 * np.abs(fuzzy_grid[0, :] - fuzzy_grid[3, :]).sum(),
    'B':  6 * np.abs(fuzzy_grid[0, :] - fuzzy_grid[4, :]).sum(),
    'C':  6 * np.abs(fuzzy_grid[0, :] - fuzzy_grid[5, :]).sum()
}
```

```
In [216]:
sorted_grading_dists = sorted(grading_dists.items(), key=lambda x: x[1])
sorted_grading_dists
Out[216]:
[('Ex', 0.0),
 ('A+', 7.199999999999999),
 ('A', 12.0),
 ('B+', 16.8),
 ('B', 19.200000000000003),
 ('C', 21.6)]
```

```
In [222]: student_evaluation['grade'] = pd.cut(student_evaluation['total_dist'], bins=[gdist[1] for gdist in sorted_grading_dists] + [np.i
nf], labels=[gdist[0] for gdist in sorted_grading_dists])
```

display grades of students

```
In [228]: student_evaluation[['total_dist', 'grade']]
Out[228]:
```

	total_dist	grade
student_00	10.6	A+
student_01	13.4	A
student_02	11.0	A+
student_03	11.6	A+
student_04	14.2	A
student_05	9.8	A+
student_06	10.4	A+
student_07	12.2	A
student_08	7.8	A+
student_09	12.8	A
student_10	12.0	A
student_11	11.0	A+
student_12	12.0	A
student_13	11.4	A+
student_14	9.0	A+
student_15	12.8	A
student_16	8.8	A+
student_17	9.4	A+
student_18	13.4	A
student_19	11.4	A+
student_20	12.4	A
student_21	12.2	A
student_22	12.4	A
student_23	9.6	A+
student_24	10.4	A+
student_25	15.6	A
student_26	9.2	A+
student_27	7.6	A+
student_28	12.0	A
student_29	12.2	A
student_30	12.4	A
student_31	11.0	A+
student_32	13.0	A
student_33	7.6	A+
student_34	11.4	A+
student_35	11.2	A+
student_36	10.0	A+
student_37	10.4	A+
student_38	10.6	A+
student_39	10.4	A+
student_40	12.6	A
student_41	3.2	Ex
student_42	11.0	A+
student_43	13.4	A
student_44	8.4	A+
student_45	12.0	A
student_46	10.4	A+
student_47	15.2	A
student_48	10.6	A+
student_49	7.0	Ex

Display sorted Grades

```
In [232]: student_evaluation.sort_values('total_dist', ascending=True)[['total_dist', 'grade']]
Out[232]:
```

	total_dist	grade
student_41	3.2	Ex
student_49	7.0	Ex
student_27	7.6	A+
student_33	7.6	A+
student_08	7.8	A+
student_44	8.4	A+
student_16	8.8	A+
student_14	9.0	A+
student_26	9.2	A+
student_17	9.4	A+
student_23	9.6	A+
student_05	9.8	A+
student_36	10.0	A+
student_46	10.4	A+
student_39	10.4	A+
student_37	10.4	A+
student_24	10.4	A+
student_06	10.4	A+
student_48	10.6	A+
student_00	10.6	A+
student_38	10.6	A+
student_02	11.0	A+
student_31	11.0	A+
student_42	11.0	A+
student_11	11.0	A+
student_35	11.2	A+
student_19	11.4	A+
student_34	11.4	A+
student_13	11.4	A+
student_03	11.6	A+
student_10	12.0	A
student_12	12.0	A
student_28	12.0	A
student_45	12.0	A
student_21	12.2	A
student_29	12.2	A
student_07	12.2	A
student_30	12.4	A
student_20	12.4	A
student_22	12.4	A
student_40	12.6	A
student_09	12.8	A
student_15	12.8	A
student_32	13.0	A
student_01	13.4	A
student_43	13.4	A
student_18	13.4	A
student_04	14.2	A
student_47	15.2	A
student_25	15.6	A

Convert this notebook to pdf

```
In [233]:
%%capture
!apt update
!apt install texlive-xetex texlive-fonts-recommended texlive-generic-recommended
```

```
In [234]:
import subprocess
import shlex
```

Convert to pdf

```
In [235]:
s = subprocess.Popen(shlex.split(
    f'jupyter nbconvert /content/fuzzy-logic-grading-system.ipynb --to pdf'
    ), shell = False, stdout = subprocess.PIPE, stderr = subprocess.PIPE)
s.wait()
s.stdout.read()
Out[235]: b''
```

```
In [ ]:
```