Ontario Tech University

Machine Learning and Data Mining

Mini Project 1

Deadline: Wed., Feb. 19, 11:59 PM

Instructor: Dr. Shahryar Rahnamayan, PEng

Design and Implementation of

Fuzzy Expert Match-Making System

Team Members	Student Numbers
Jasindan Rasalingam	100584595
Ashwin Kamalakannan	100584423

Problem

- To determine marriage compatibility between two people based on money, attractiveness, and age.

Linguistic variables

- Money, Age, Attractiveness of Person A and B

Fuzzy sets/Membership Functions

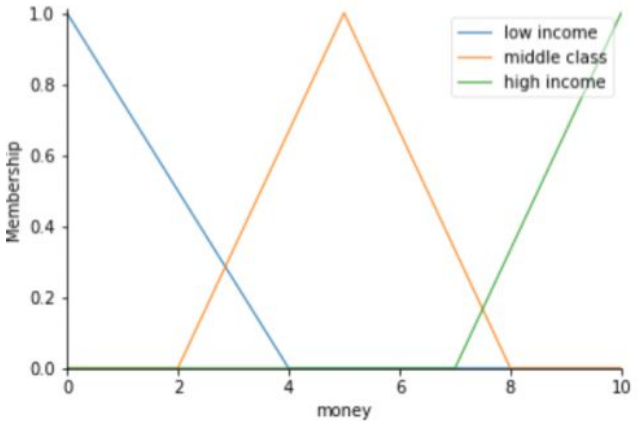
We built our fuzzy logic solution in python using the **scikit-fuzzy** module and Jupyter Notebook. With this, we were able to quickly build membership functions and show the relationships for each antecedent. This module also made it easy to tweak our rules in case of unrealistic outcomes.

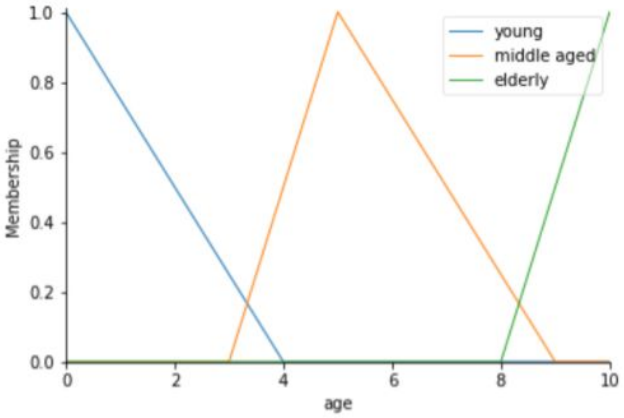
For our membership functions we tried to depict a realistic distribution of data based on each trait. We had to make some assumptions, such as not many people wanting to call themselves unattractive, and less elderly people being interested in marriage. These assumptions helped our model seem more realistic.

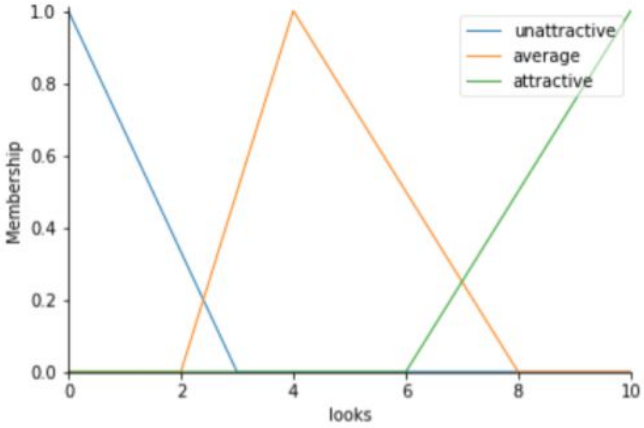
```
age['young'] = fuzz.trimf(age.universe, [0, 0, 4])
age['middle aged'] = fuzz.trimf(age.universe, [3, 5, 9])
age['elderly'] = fuzz.trimf(age.universe, [8, 10, 10])
money['low income'] = fuzz.trimf(money.universe, [0, 0, 4])
money['middle class'] = fuzz.trimf(money.universe, [2, 5, 8])
money['high income'] = fuzz.trimf(money.universe, [7, 10, 10])
looks['unattractive'] = fuzz.trimf(looks.universe, [0, 0, 3])
looks['average'] = fuzz.trimf(looks.universe, [2, 4, 8])
looks['attractive'] = fuzz.trimf(looks.universe, [6, 10, 10])

age2['young'] = fuzz.trimf(age.universe, [0, 0, 3])
age2['middle aged'] = fuzz.trimf(age.universe, [2, 5, 9])
age2['elderly'] = fuzz.trimf(age.universe, [8, 10, 10])
money2['low income'] = fuzz.trimf(money.universe, [0, 0, 5])
money2['middle class'] = fuzz.trimf(money.universe, [2, 5, 8])
money2['high income'] = fuzz.trimf(money.universe, [5, 10, 10])
looks2['unattractive'] = fuzz.trimf(looks.universe, [0, 0, 4])
looks2['average'] = fuzz.trimf(looks.universe, [3, 5, 8])
looks2['attractive'] = fuzz.trimf(looks.universe, [6, 10, 10])

compatibility['low'] = fuzz.trimf(compatibility.universe, [0, 0, 4])
compatibility['medium'] = fuzz.trimf(compatibility.universe, [1, 5, 9])
compatibility['high'] = fuzz.trimf(compatibility.universe, [6, 10, 10])
```

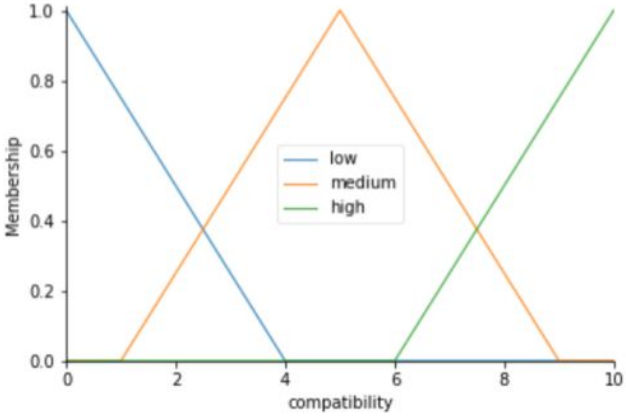
Membership Functions

Below, you can see where each input will fall for an antecedent based on our defined fuzzy logic. Some of the graphs are skewed one way or another to take into consideration real world scenarios and biases.









Fuzzy Rules

```
# Fuzzy rules based on age
rule1_age = ctrl.Rule(age['young'] & age2['young'], compat['high'])
rule2_age = ctrl.Rule(age['middle aged'] & age2['middle aged'], compat['high'])
rule3_age = ctrl.Rule(age['elderly'] & age2['elderly'], compat['high'])
rule4_age = ctrl.Rule(age['middle aged'] & (age2['young'] | age2['elderly']), compat['medium'])
rule5_age = ctrl.Rule(age['young'] & age2['elderly'], compat['low'])
age_ctrl = ctrl.ControlSystem([rule1_age, rule2_age, rule3_age, rule4_age, rule5_age])
age_compat = ctrl.ControlSystemSimulation(age_ctrl)

# Fuzzy rules based on money
rule1_mon = ctrl.Rule(money['low income'] & money2['low income'], compat['medium'])
rule2_mon = ctrl.Rule(money['middle class'] & money2['middle class'], compat['high'])
rule3_mon = ctrl.Rule(money['high income'] & money2['high income'], compat['high'])
rule4_mon = ctrl.Rule(money['middle class'] & (money2['low income'] | money2['high income']), compat['medium'])
rule5_mon = ctrl.Rule(money['low income'] & money2['high income'], compat['medium'])
money_ctrl = ctrl.ControlSystem([rule1_mon, rule2_mon, rule3_mon, rule4_mon, rule5_mon])
money_compat = ctrl.ControlSystemSimulation(money_ctrl)

# Fuzzy rules based on looks
rule1_lks = ctrl.Rule(looks['unattractive'] & looks2['unattractive'], compat['medium'])
rule2_lks = ctrl.Rule(looks['average'] & looks2['average'], compat['high'])
rule3_lks = ctrl.Rule(looks['attractive'] & looks2['attractive'], compat['high'])
rule4_lks = ctrl.Rule(looks['average'] & (looks2['unattractive'] | looks2['attractive']), compat['medium'])
rule5_lks = ctrl.Rule(looks['unattractive'] & looks['attractive'], compat['low'])
looks_ctrl = ctrl.ControlSystem([rule1_lks, rule2_lks, rule3_lks, rule4_lks, rule5_lks])
looks_compatibility = ctrl.ControlSystemSimulation(looks_ctrl)
```

In general, our rules are as follows:

1. Same membership will result in high compatibility
2. Opposite membership will result in low compatibility
3. Similar memberships will result in medium compatibility

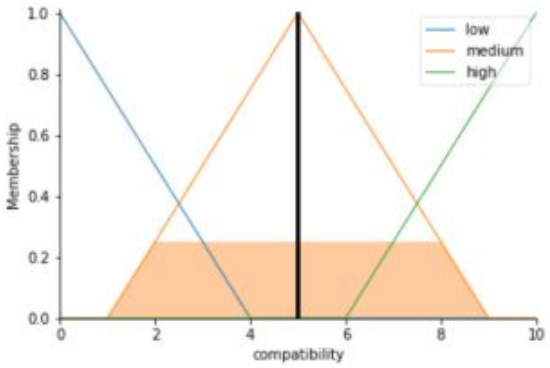
Based on our testing, we modified and added some rules so our consequences are more in line with real world outcomes. For example, two low income people may not necessarily have high compatibility because one of them might want the other to support them. For scenarios such as this, we had to tweak our rules to produce more accurate output.

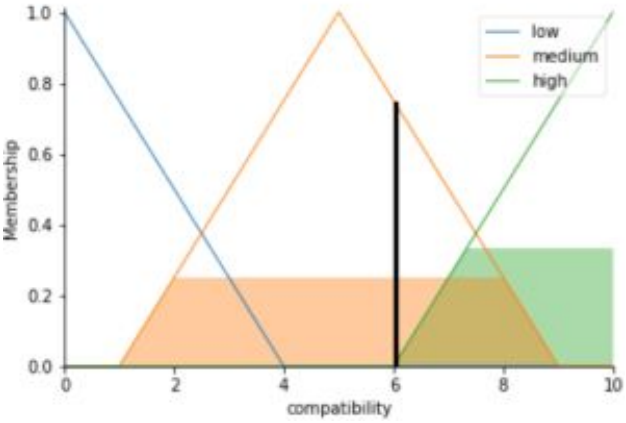
Test Case #1:

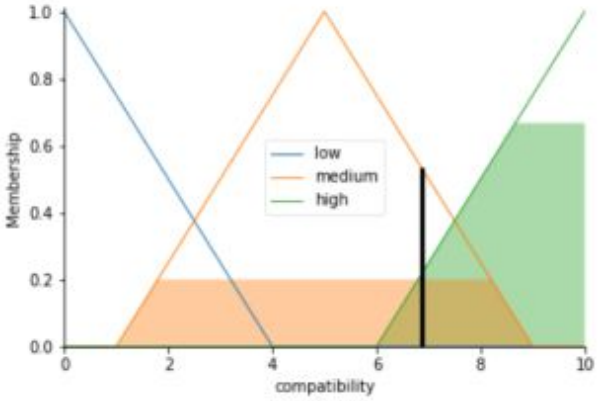
Person A(3,5,6) and Person B(4,4,7)
This is an example of Medium compatibility between Person A and Person B.

```
age_compat.input['age'] = 3
age_compat.input['age2'] = 4
age_compat.compute()
money_compat.input['money'] = 5
money_compat.input['money2'] = 4
money_compat.compute()
looks_compat.input['looks'] = 6
looks_compat.input['looks2'] = 7
looks_compat.compute()
```

```
('Age Compatibility', 5.0)
('Money Compatibility', 6.8775571441179055)
('Looks Compatibility', 6.028758169934639)
```







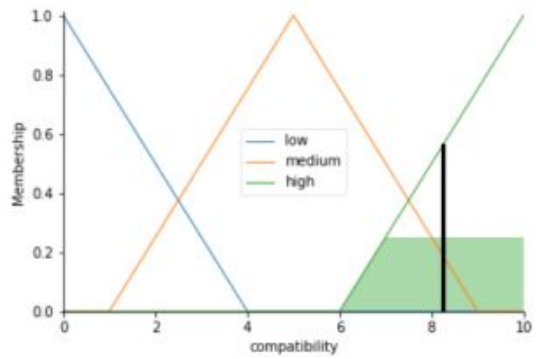
For this example, we specifically wanted to show two individuals with medium compatibility. For that, we used values that are close but not quite similar to each other which falls under most of the medium outcome rules in our fuzzy logic. The looks compatibility ended up being higher because in our rules we defined medium and highly attractive individuals to be highly compatible.

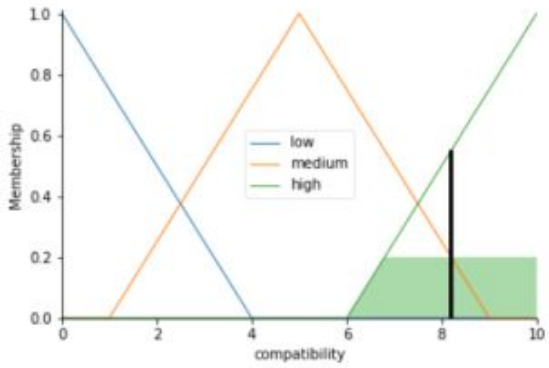
Test Case #2:

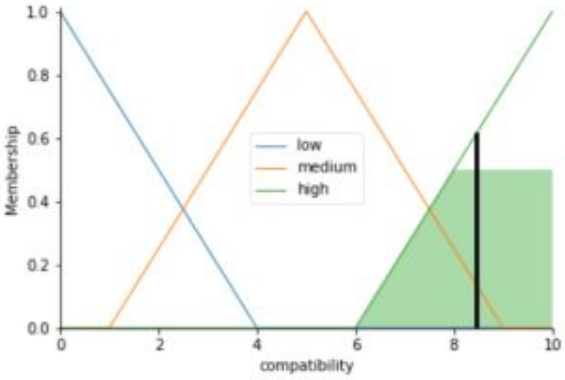
Person A(3,8,3) and Person B (2,6,4)
This is an example of very high Compatibility between Person A and Person B.

```
age_compat.input['age'] = 3
age_compat.input['age2'] = 2
age_compat.compute()
money_compat.input['money'] = 8
money_compat.input['money2'] = 6
money_compat.compute()
looks_compat.input['looks'] = 3
looks_compat.input['looks2'] = 4
looks_compat.compute()
```

```
('Age Compatibility', 8.238095238095239)
('Money Compatibility', 8.192592592592593)
('Looks Compatibility', 8.444444444444445)
```







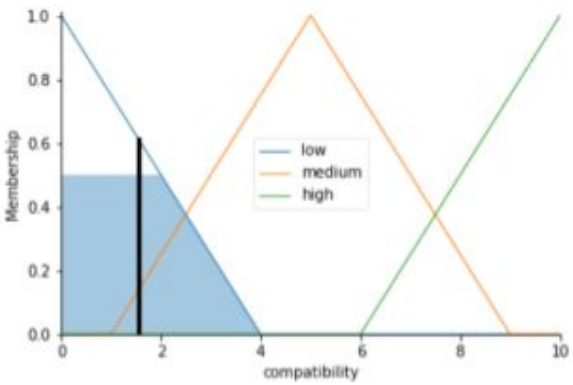
For this example, we wanted to show high compatibility between two individuals. To do this we chose values that were very similar to each other and fell under higher outcomes based on our fuzzy rules.

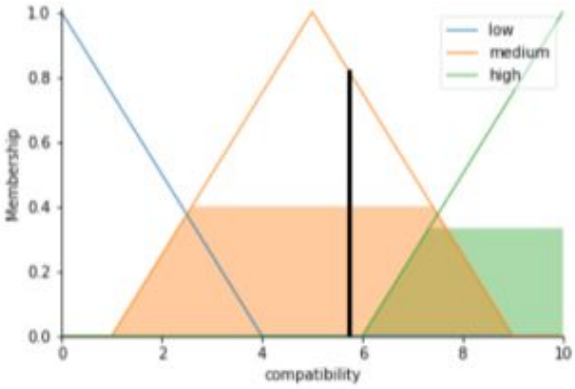
Test Case #3:

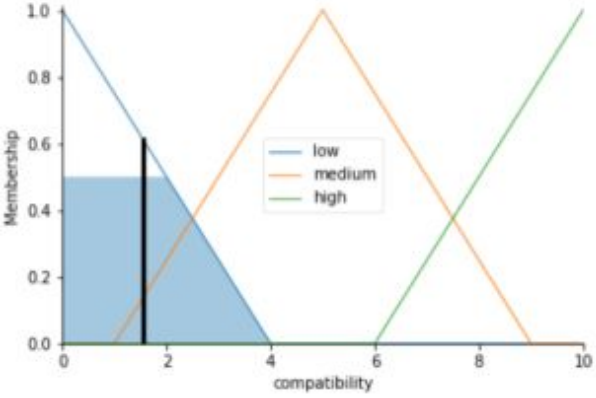
Person A(2,4,8) and Person B(9,7,2)
This is an example of Low compatibility between Person A and Person B

```
age_compat.input['age'] = 2
age_compat.input['age2'] = 9
age_compat.compute()
money_compat.input['money'] = 4
money_compat.input['money2'] = 7
money_compat.compute()
looks_compat.input['looks'] = 8
looks_compat.input['looks2'] = 2
looks_compat.compute()
```

```
('Age Compatibility', 1.5555555555555556)
('Money Compatibility', 5.742004264392324)
('Looks Compatibility', 1.5555555555555556)
```







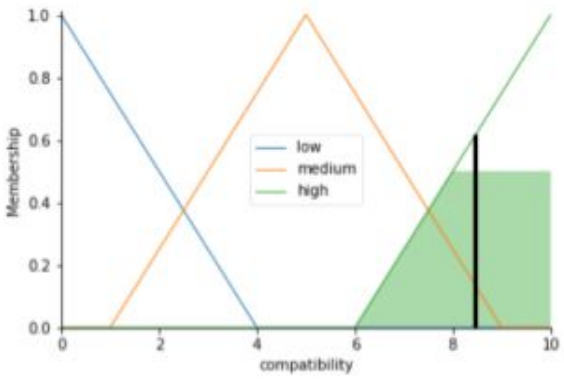
For this example, we wanted to show low compatibility between two individuals. To do this, we made sure to pass opposite input values for each trait to produce very different memberships. Based on our rules, this caused most of the outputs to show low compatibility. The money compatibility was medium mainly because we assumed that a low income person would prefer a higher income person to help support them, this reflected in our fuzzy rules and cased them to have a higher compatibility.

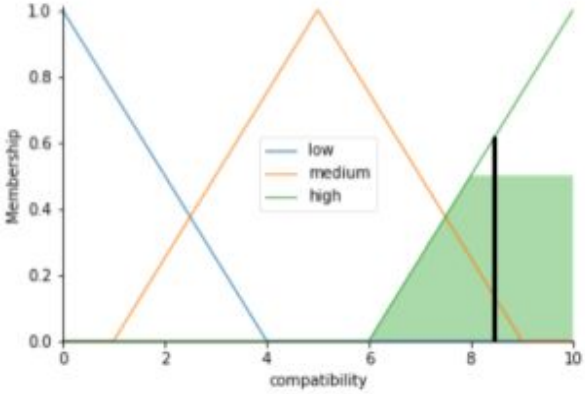
Test Case #4:

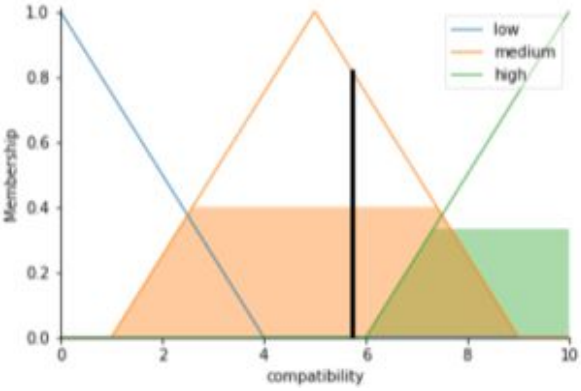
Person A(4,5,6) and Person B(6,3,5)
This is another example high compatibility between Person A and Person B

```
age_compat.input['age'] = 4
age_compat.input['age2'] = 6
age_compat.compute()
money_compat.input['money'] = 5
money_compat.input['money2'] = 3
money_compat.compute()
looks_compat.input['looks'] = 6
looks_compat.input['looks2'] = 5
looks_compat.compute()
```

```
('Age Compatibility', 8.444444444444445)
('Money Compatibility', 5.742004264392324)
('Looks Compatibility', 8.444444444444445)
```







For this example, we wanted to show another high compatibility between two individuals. In this case, we chose values that were very similar to each other and fell under higher outcomes based on our fuzzy rules. The money compatibility is medium in this case for the same reason as in test case 3.