Heart Disease Detection Using Fuzzy Logic

By:

Anuradha Ramchandran

What is Heart Disease?

- The term “heart disease” refers to several types of heart conditions.
- There are many different heart conditions and problems which are collectively called heart disease.
- **The four main types are:** coronary heart disease, stroke, peripheral arterial disease and aortic disease.
- Coronary artery disease or CAD is the most common type of heart disease in the United States and affects the blood flow to the heart.
- Decreased blood flow can cause a heart attack.

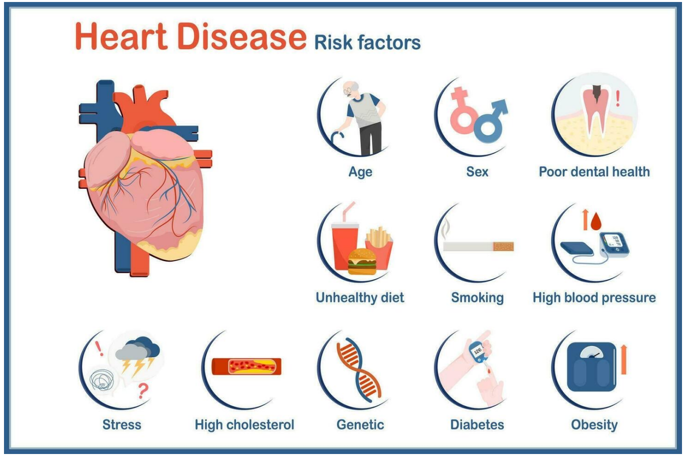
Symptoms of Heart Disease

Sometimes heart disease may be “silent” and not diagnosed until a person experiences signs or symptoms of a heart attack, heart failure, or an arrhythmia. When these events happen, symptoms may include:

- **Heart attack:** Chest pain or discomfort, upper back or neck pain, indigestion, heartburn, nausea or vomiting, extreme fatigue, upper body discomfort, dizziness, and shortness of breath.
- **Arrhythmia:** Fluttering feelings in the chest (palpitations).
- **Heart failure:** Shortness of breath, fatigue, or swelling of the feet, ankles, legs, abdomen, or neck veins.

Risk Factors Affecting Heart Disease Risk

- High blood pressure, high blood cholesterol, and smoking are key risk factors for heart disease.
- Several other medical conditions and lifestyle choices can also put people at a higher risk for heart disease, including:
  - ❖ Diabetes
  - ❖ Obesity
  - ❖ Unhealthy diet
  - ❖ Physical inactivity
  - ❖ Excessive alcohol use



Importance of Early Detection

- Impact of Late Diagnosis:
  - Increased Treatment Complexity
  - Reduced Treatment Success
  - Higher Healthcare Costs
- Benefits of Early Detection:
  - Improved Treatment Efficacy
  - Enhanced Quality of Life
  - Lower Healthcare Costs
- Role in Preventive Healthcare:
  - Empowering Patients
  - Population Health Impact
  - Importance of Regular Check-ups

Role of Expert Systems and Machine Learning

- Expert systems are computer-based systems mimicking problem-solving ability of a human expert.
- They can be used to analyze medical data and emulate expert decision-making.
- Machine learning models can also be used to analyze healthcare data for predictions, diagnosis, and treatment planning.
- Combining expert systems and machine learning gives better and more accurate results.

What is Fuzzy Logic?

- Introduced by Lotfi Zadeh in the 1960s.
- Deals with reasoning and decision-making in situations where uncertainty, imprecision, and vagueness are present.
- Unlike classical (or "crisp") logic, which operates with binary values of true or false, it allows for the representation of degrees of truth.

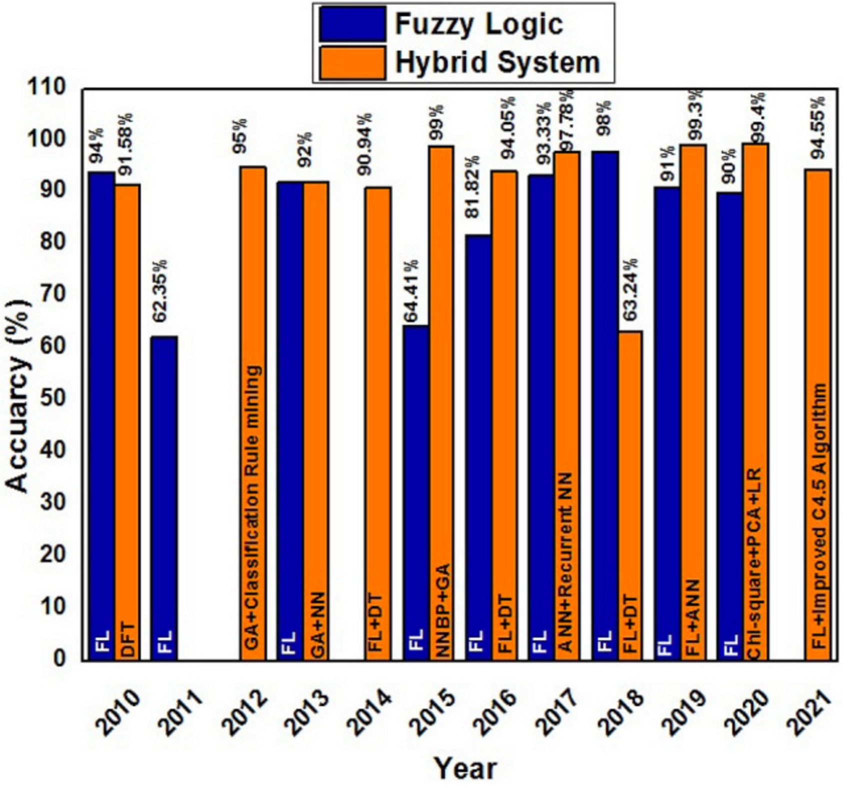
How can Fuzzy Logic be used?

- Can handle uncertain medical parameters with membership functions.
- Models how “fuzzy” risk factors can be.
- Capable of considering multiple factors at once -> complex case support.
- Personalized treatment plans.
- Adaptive drug prescriptions and dosages.

What are Decision Trees and Random Forests?

- Decision Trees:
  - Supervised machine-learning algorithm for both classification and regression problems.
  - Flowchart-like tree structure where each internal node denotes the feature, branches denote the rules, and the leaf nodes denote the result of the algorithm.
- C4.5 addresses its limitations by handling both numeric and categorical attributes, introducing pruning, and using a more robust attribute selection measure, Gain Ratio.
- Random forests are ensemble algorithms of many decision trees.
- They are known for high accuracy and handling large datasets.

Fuzzy System vs Hybrid Approach



How does the Dataset look like?

The Cleveland Heart Disease dataset is taken from the UCI ML repository.

- Consists of 303 records with 76 attributes.
- Attributes include information such as age, sex, cholesterol levels, and electrocardiogram measurements.
- Research papers tend to use only a subset of 14.

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	45	1	1	110	264	0	0	132	0	1.2	2	0.0	7.0
299	68	1	4	144	193	1	0	141	0	3.4	2	2.0	7.0
300	57	1	4	130	131	0	0	115	1	1.2	2	1.0	7.0
301	57	0	2	130	236	0	2	174	0	0.0	2	1.0	3.0
302	38	1	3	138	175	0	0	173	0	0.0	1	0.0	3.0

303 rows × 13 columns

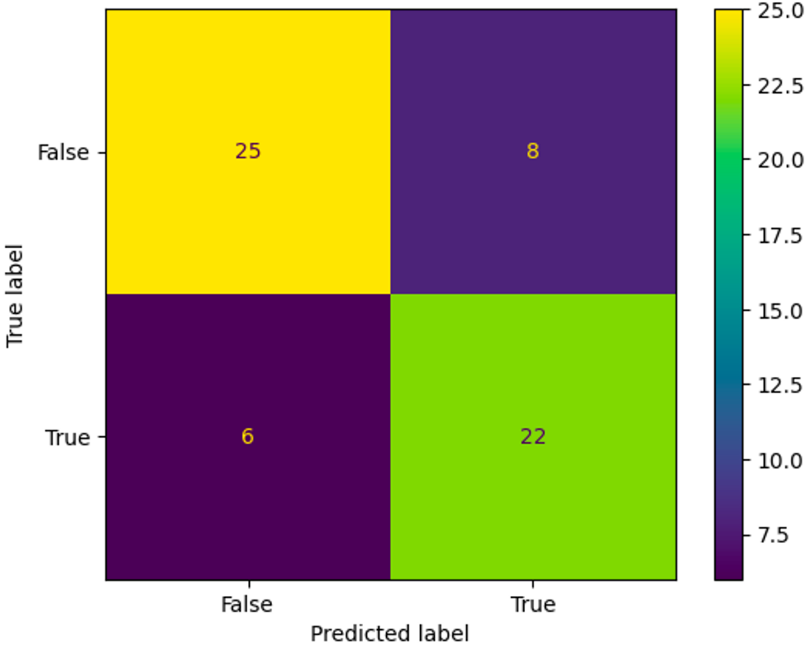
How is a Fuzzy Logic Based Expert System Designed?

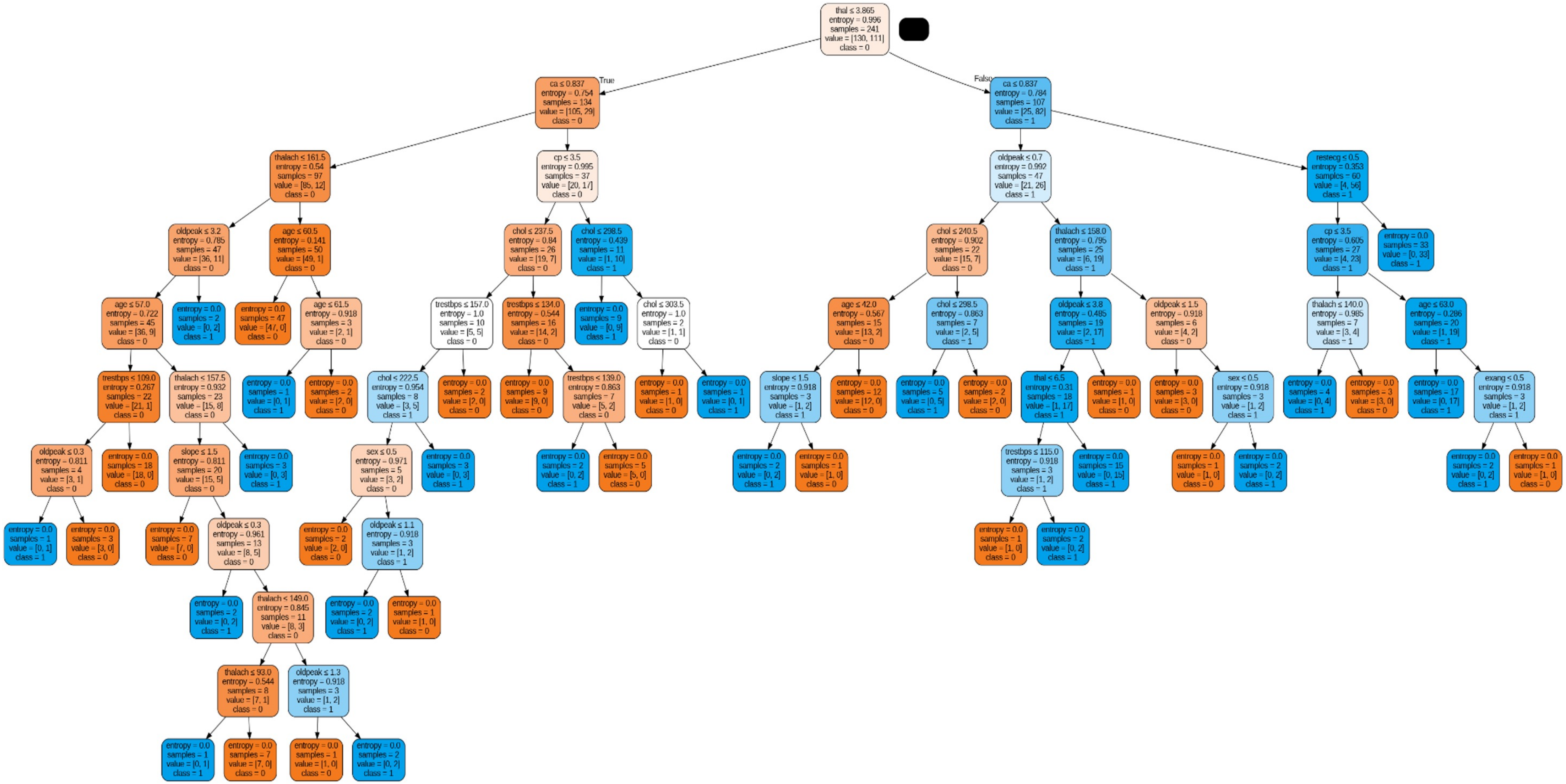
- Knowledge Acquisition
- Rule-base Construction
- Implementation and Integration
- Testing and Maintenance

Decision Tree Modelling and Analysis

- Scikit-learn’s decision tree algorithm with the criterion set to “entropy” to model C4.5 Algorithm.

  Accuracy Obtained: 77%

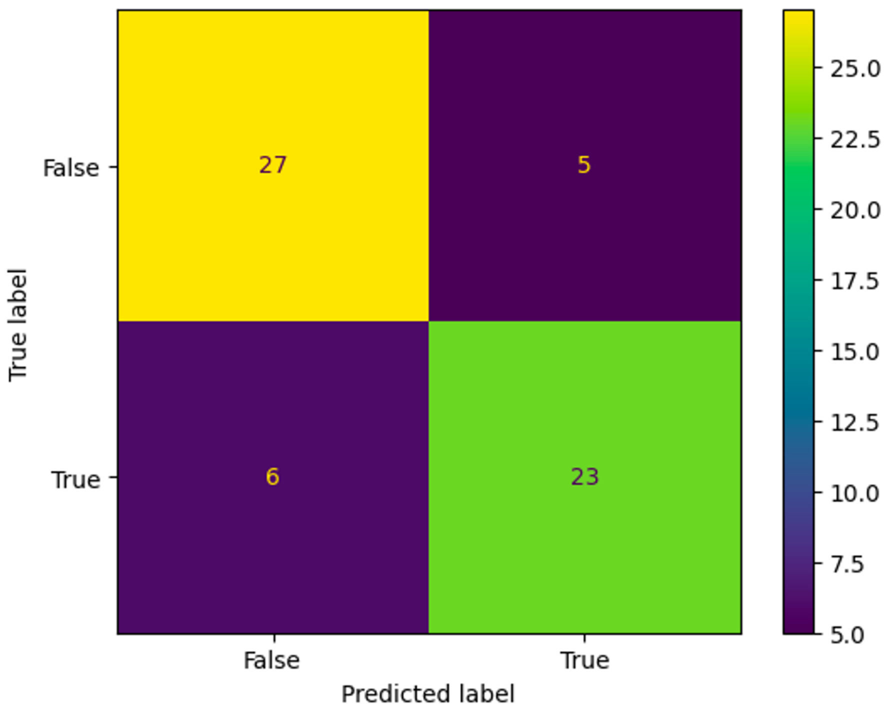


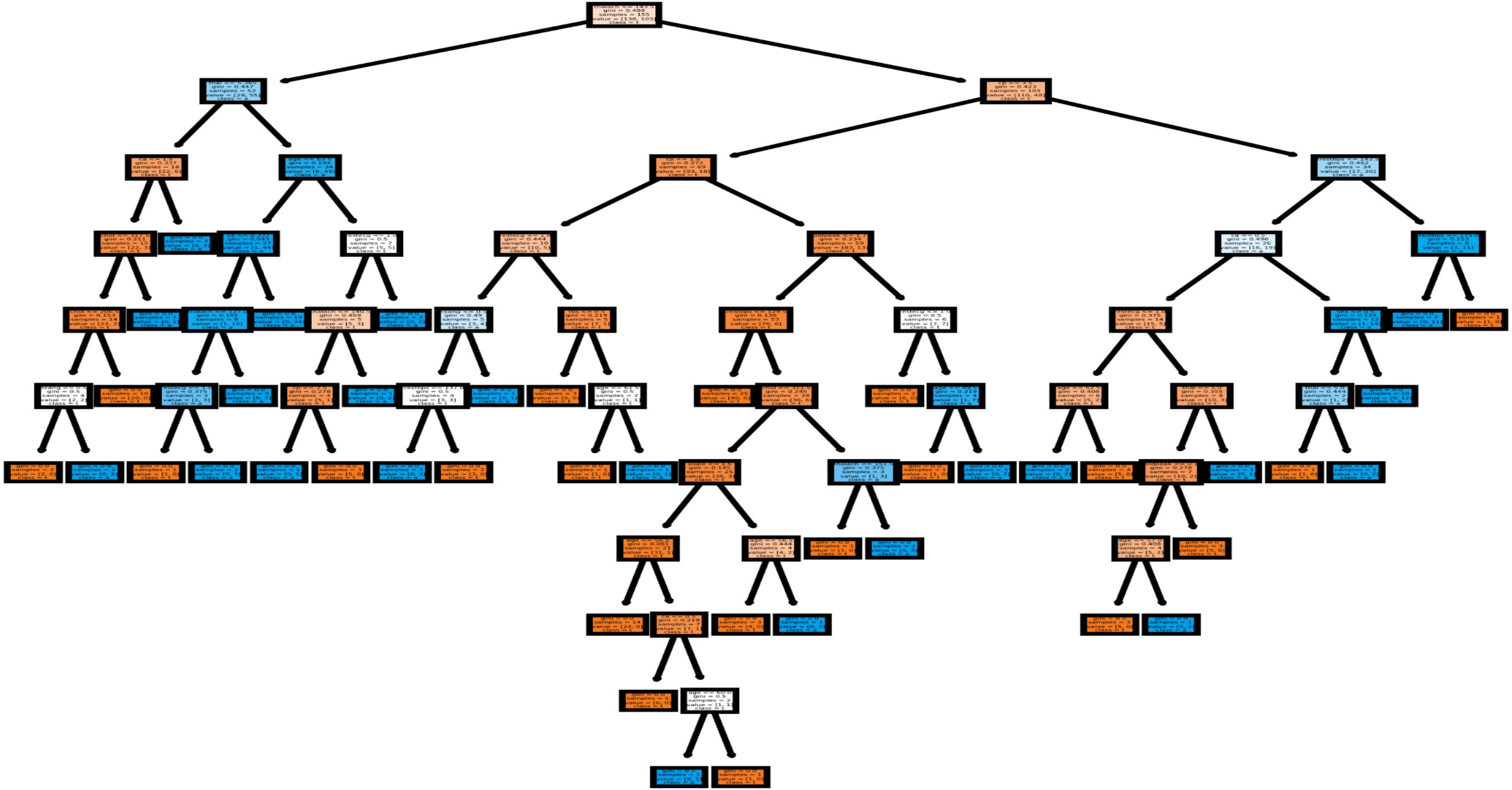


thal ≤ 3.865 entropy = 0.996 samples = 241 value = [130, 111] class = 0
True
False
ca ≤ 0.837 entropy = 0.754 samples = 134 value = [105, 29] class = 0
ca ≤ 0.837 entropy = 0.784 samples = 107 value = [25, 82] class = 1
thalach ≤ 161.5 entropy = 0.54 samples = 97 value = [85, 12] class = 0
cp ≤ 3.5 entropy = 0.995 samples = 37 value = [20, 17] class = 0
oldpeak ≤ 0.7 entropy = 0.992 samples = 47 value = [21, 26] class = 1
restecg ≤ 0.5 entropy = 0.353 samples = 60 value = [4, 56] class = 1
oldpeak ≤ 3.2 entropy = 0.785 samples = 47 value = [36, 11] class = 0
age ≤ 60.5 entropy = 0.141 samples = 50 value = [49, 1] class = 0
chol ≤ 237.5 entropy = 0.84 samples = 26 value = [19, 7] class = 0
chol ≤ 298.5 entropy = 0.439 samples = 11 value = [1, 10] class = 1
chol ≤ 240.5 entropy = 0.902 samples = 22 value = [15, 7] class = 0
thalach ≤ 158.0 entropy = 0.795 samples = 25 value = [6, 19] class = 1
cp ≤ 3.5 entropy = 0.605 samples = 27 value = [4, 23] class = 1
entropy = 0.0 samples = 33 value = [0, 33] class = 1
age ≤ 57.0 entropy = 0.722 samples = 45 value = [36, 9] class = 0
entropy = 0.0 samples = 2 value = [0, 2] class = 1
entropy = 0.0 samples = 47 value = [47, 0] class = 0
age ≤ 61.5 entropy = 0.918 samples = 3 value = [2, 1] class = 0
trestbps ≤ 157.0 entropy = 1.0 samples = 10 value = [5, 5] class = 0
trestbps ≤ 134.0 entropy = 0.544 samples = 16 value = [14, 2] class = 0
entropy = 0.0 samples = 9 value = [0, 9] class = 1
chol ≤ 303.5 entropy = 1.0 samples = 2 value = [1, 1] class = 0
age ≤ 42.0 entropy = 0.567 samples = 15 value = [13, 2] class = 0
chol ≤ 298.5 entropy = 0.863 samples = 7 value = [2, 5] class = 1
oldpeak ≤ 3.8 entropy = 0.485 samples = 19 value = [2, 17] class = 1
oldpeak ≤ 1.5 entropy = 0.918 samples = 6 value = [4, 2] class = 0
thalach ≤ 140.0 entropy = 0.985 samples = 7 value = [3, 4] class = 1
age ≤ 63.0 entropy = 0.286 samples = 20 value = [1, 19] class = 1
trestbps ≤ 109.0 entropy = 0.267 samples = 22 value = [21, 1] class = 0
thalach ≤ 157.5 entropy = 0.932 samples = 23 value = [15, 8] class = 0
entropy = 0.0 samples = 1 value = [0, 1] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = 0
chol ≤ 222.5 entropy = 0.954 samples = 8 value = [3, 5] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = 0
entropy = 0.0 samples = 9 value = [9, 0] class = 0
trestbps ≤ 139.0 entropy = 0.863 samples = 7 value = [5, 2] class = 0
entropy = 0.0 samples = 1 value = [1, 0] class = 0
entropy = 0.0 samples = 1 value = [0, 1] class = 1
slope ≤ 1.5 entropy = 0.918 samples = 3 value = [1, 2] class = 1
entropy = 0.0 samples = 12 value = [12, 0] class = 0
entropy = 0.0 samples = 5 value = [0, 5] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = 0
thal ≤ 6.5 entropy = 0.31 samples = 18 value = [1, 17] class = 1
entropy = 0.0 samples = 1 value = [1, 0] class = 0
entropy = 0.0 samples = 3 value = [3, 0] class = 0
sex ≤ 0.5 entropy = 0.918 samples = 3 value = [1, 2] class = 1
entropy = 0.0 samples = 4 value = [0, 4] class = 1
entropy = 0.0 samples = 3 value = [3, 0] class = 0
entropy = 0.0 samples = 17 value = [0, 17] class = 1
exang ≤ 0.5 entropy = 0.918 samples = 3 value = [1, 2] class = 1
oldpeak ≤ 0.3 entropy = 0.811 samples = 4 value = [3, 1] class = 0
entropy = 0.0 samples = 18 value = [18, 0] class = 0
slope ≤ 1.5 entropy = 0.811 samples = 20 value = [15, 5] class = 0
entropy = 0.0 samples = 3 value = [0, 3] class = 1
sex ≤ 0.5 entropy = 0.971 samples = 5 value = [3, 2] class = 0
entropy = 0.0 samples = 3 value = [0, 3] class = 1
entropy = 0.0 samples = 2 value = [0, 2] class = 1
entropy = 0.0 samples = 5 value = [5, 0] class = 0
entropy = 0.0 samples = 2 value = [0, 2] class = 1
entropy = 0.0 samples = 1 value = [1, 0] class = 0
trestbps ≤ 115.0 entropy = 0.918 samples = 3 value = [1, 2] class = 1
entropy = 0.0 samples = 15 value = [0, 15] class = 1
entropy = 0.0 samples = 1 value = [1, 0] class = 0
entropy = 0.0 samples = 2 value = [0, 2] class = 1
entropy = 0.0 samples = 2 value = [0, 2] class = 1
entropy = 0.0 samples = 1 value = [1, 0] class = 0
entropy = 0.0 samples = 1 value = [0, 1] class = 1
entropy = 0.0 samples = 3 value = [3, 0] class = 0
entropy = 0.0 samples = 7 value = [7, 0] class = 0
oldpeak ≤ 0.3 entropy = 0.961 samples = 13 value = [8, 5] class = 0
entropy = 0.0 samples = 2 value = [2, 0] class = 0
oldpeak ≤ 1.1 entropy = 0.918 samples = 3 value = [1, 2] class = 1
entropy = 0.0 samples = 1 value = [1, 0] class = 0
entropy = 0.0 samples = 2 value = [0, 2] class = 1
entropy = 0.0 samples = 2 value = [0, 2] class = 1
thalach ≤ 149.0 entropy = 0.845 samples = 11 value = [8, 3] class = 0
entropy = 0.0 samples = 2 value = [0, 2] class = 1
entropy = 0.0 samples = 1 value = [1, 0] class = 0
thalach ≤ 93.0 entropy = 0.544 samples = 8 value = [7, 1] class = 0
oldpeak ≤ 1.3 entropy = 0.918 samples = 3 value = [1, 2] class = 1
entropy = 0.0 samples = 1 value = [0, 1] class = 1
entropy = 0.0 samples = 7 value = [7, 0] class = 0
entropy = 0.0 samples = 1 value = [1, 0] class = 0
entropy = 0.0 samples = 2 value = [0, 2] class = 1

Random Forest Modelling and Analysis

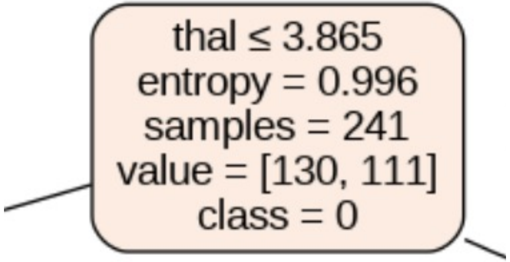
Accuracy: 79%

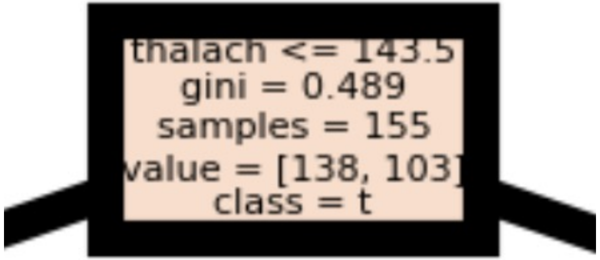




Comparison of Decision Tree and Random Forest

- **Accuracy:** 77% Vs 79%
- **Root Node Selection:**
    - Decision tree selects ‘thallium scan’ as the root node, whereas Random Forest selects ‘max heart rate’.





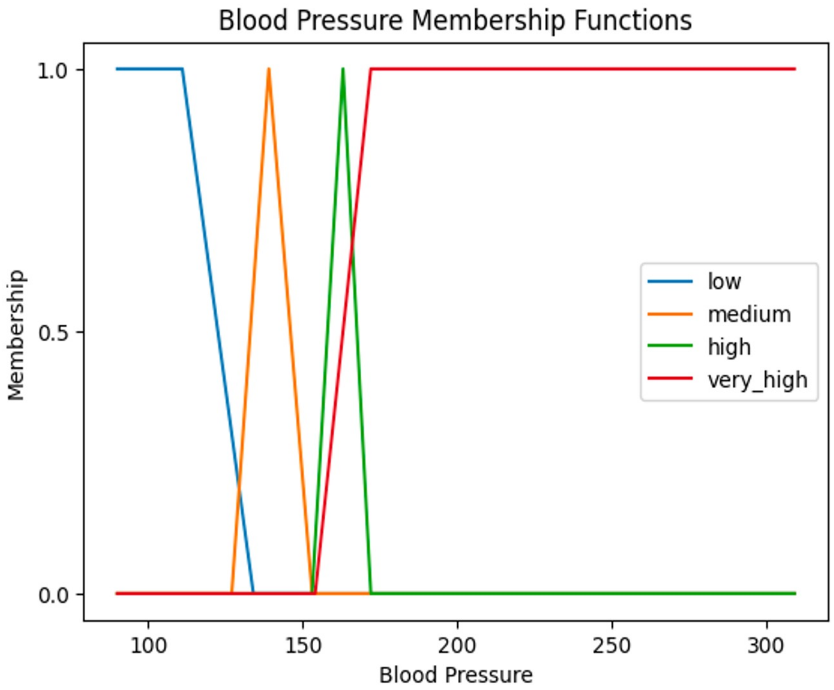
Fuzzy System Design

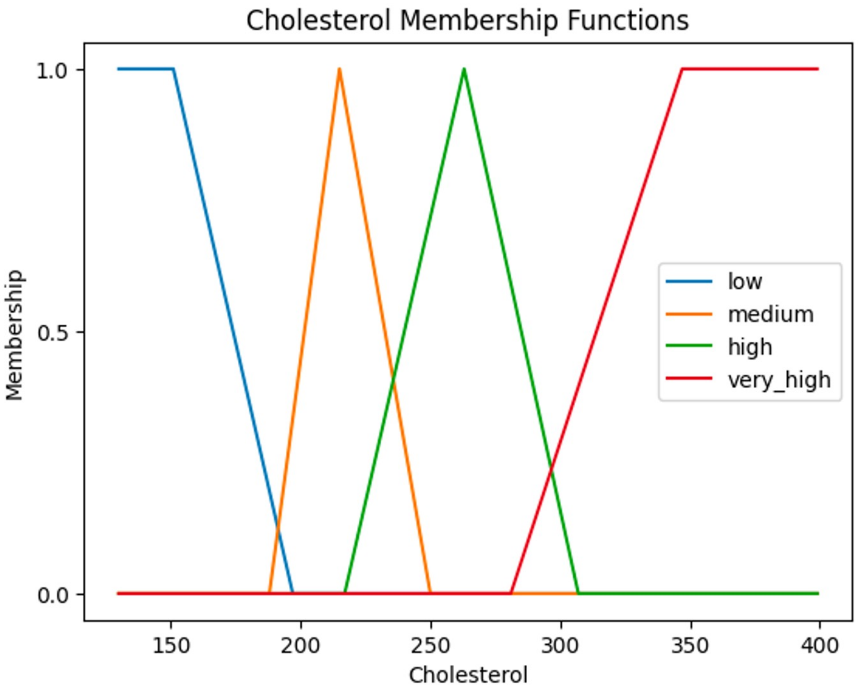
"Control" is a subpackage in the skfuzzy toolkit in Python that is used to provide a high-level API for fuzzy system design.

- **Fuzzy Control System Definition:** involves defining fuzzy input and output variables, specifying membership functions, and creating fuzzy rules within a Control System.
- **Simulation and Decision Making:** facilitates the simulation of the fuzzy control system by providing input values, evaluating fuzzy rules using an inference engine, and obtaining crisp output decisions through aggregation and defuzzification.
- **Implementation Steps:** enables the step-by-step implementation of fuzzy control systems, covering fuzzification, rule evaluation, aggregation, and defuzzification, allowing users to design and simulate decision-making processes based on fuzzy logic.

Membership Functions of Input and Output Variables

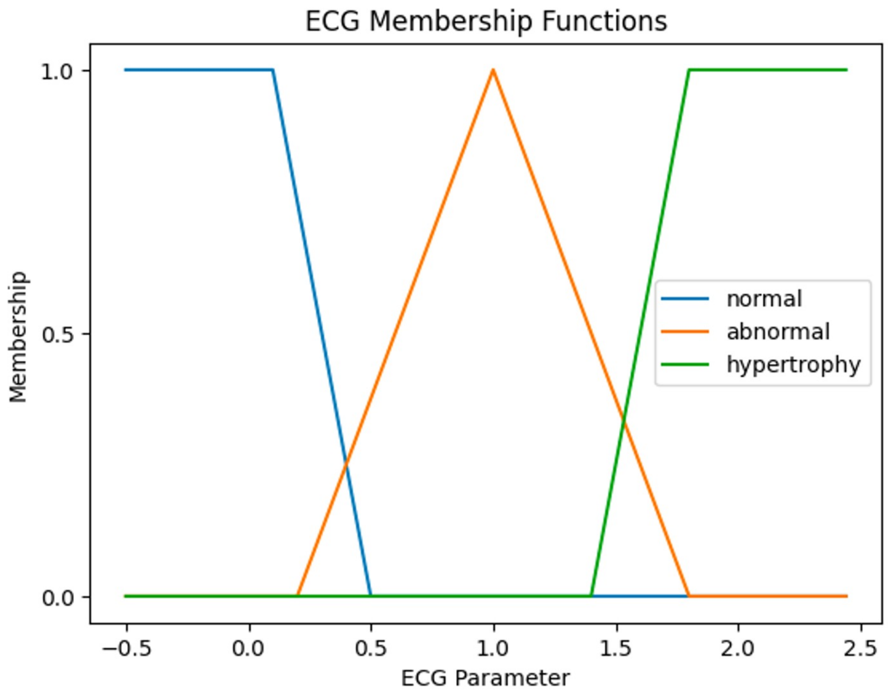
- chest_pain_values = { 'typical_anginal': 1, 'atypical_anginal': 2, 'non_aginal_pain': 3, 'asymptomatic': 4 }
- exercise_values = { 'true': 1, 'false': 0 }
- thallium_scan_values = { 'normal': 3, 'fixed_defect': 6, 'reversible_defect': 7 }
- sex_values = { 'female': 1, 'male': 0 }

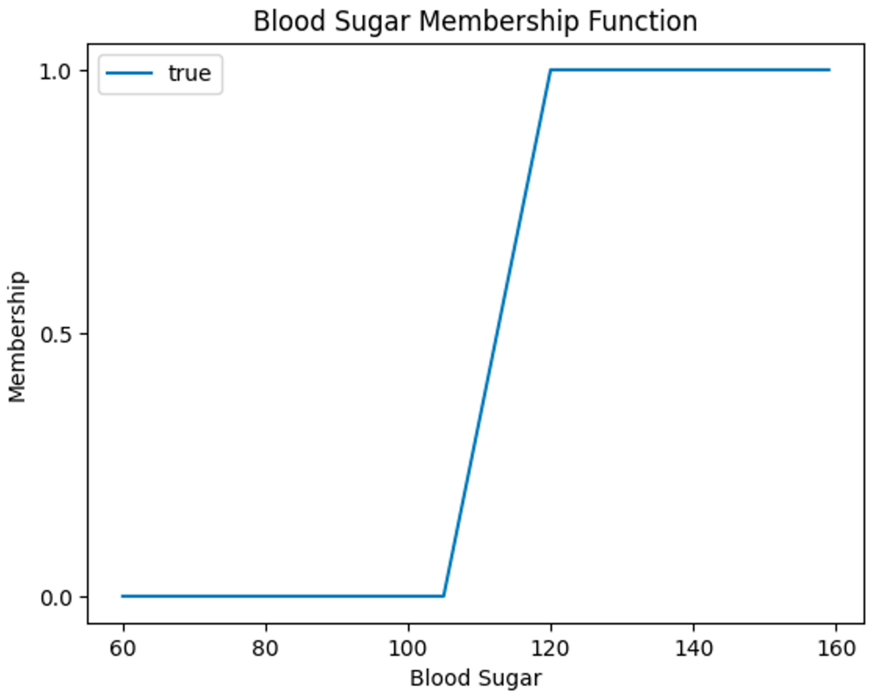




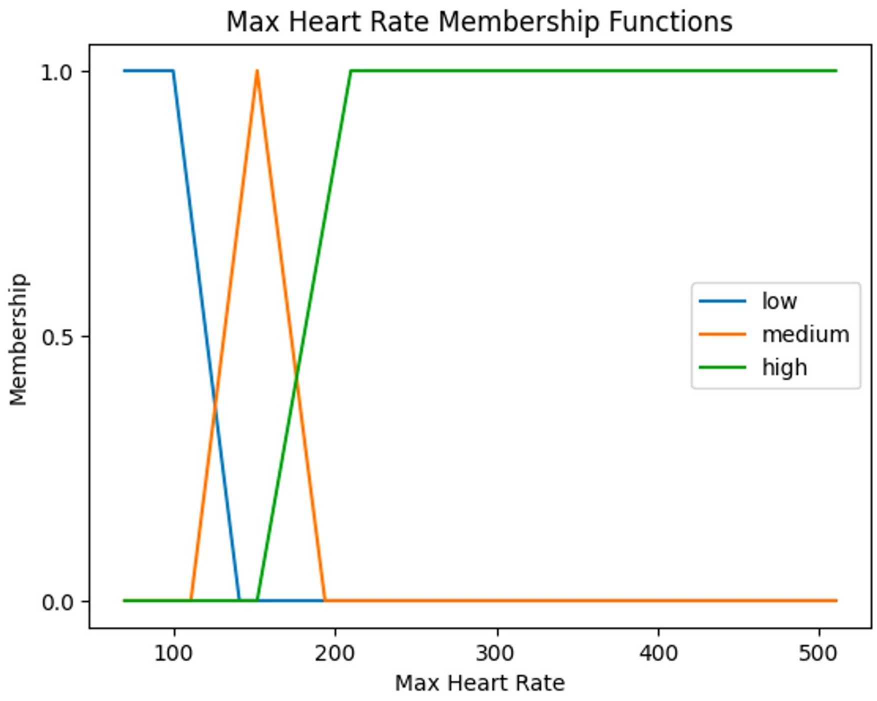
INPUT FIELD	RANGE	FUZZY SETS
Systolic Blood Pressure	<134	Low
	127-153	Medium
	142-172	High
	154>	Very high

INPUT FIELD	RANGE	FUZZY SETS
Cholesterol	<197	Low
	188-250	Medium
	217-307	High
	281>	Very high

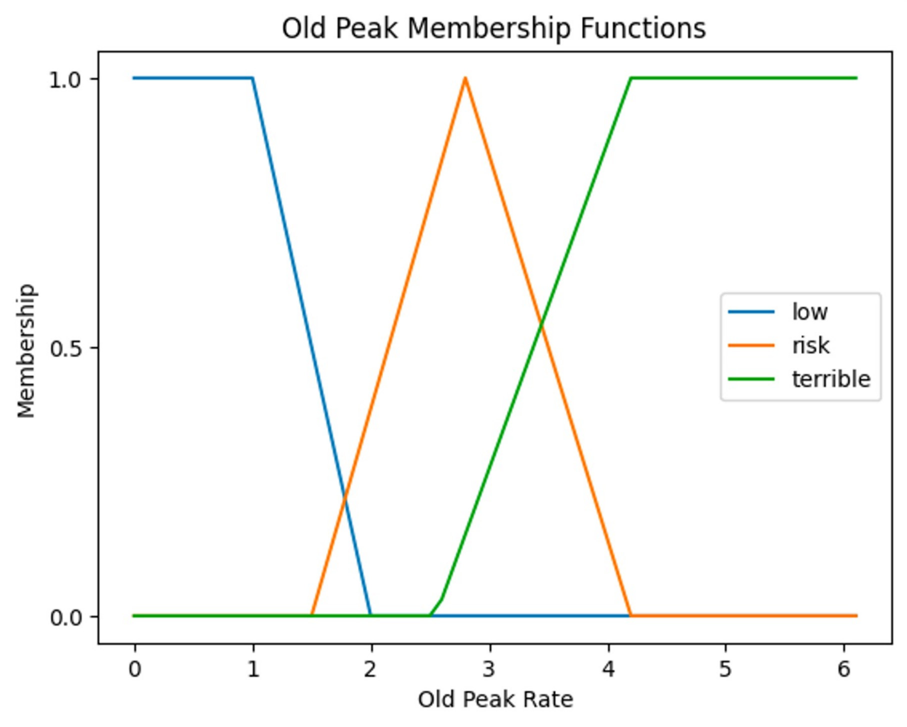




INPUT FIELD	RANGE	FUZZY SETS
Resting Electrocardiography (ECG)	(0) [-0.5, 0.4] (1) [2.45, 1.8] (2) [1.4, 2.5]	Normal ST-T abnormal Hypertrophy

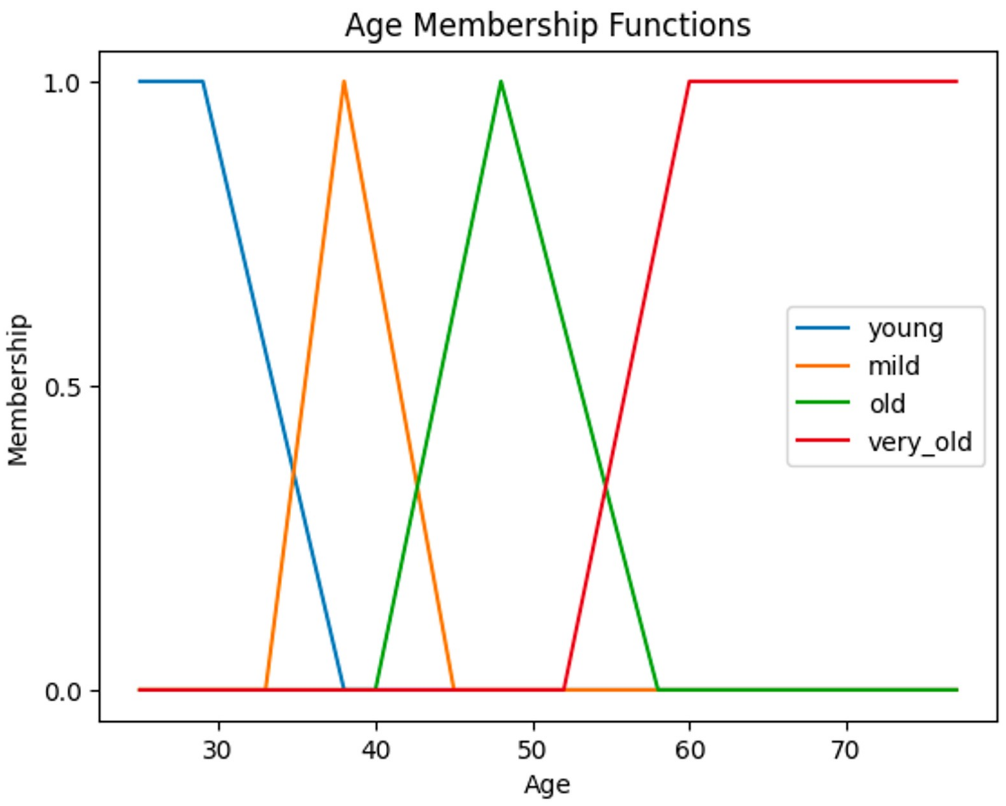


INPUT FIELD	RANGE	FUZZY SETS
Maximum Heart Rate	<141 111-194 152>	Low Medium High

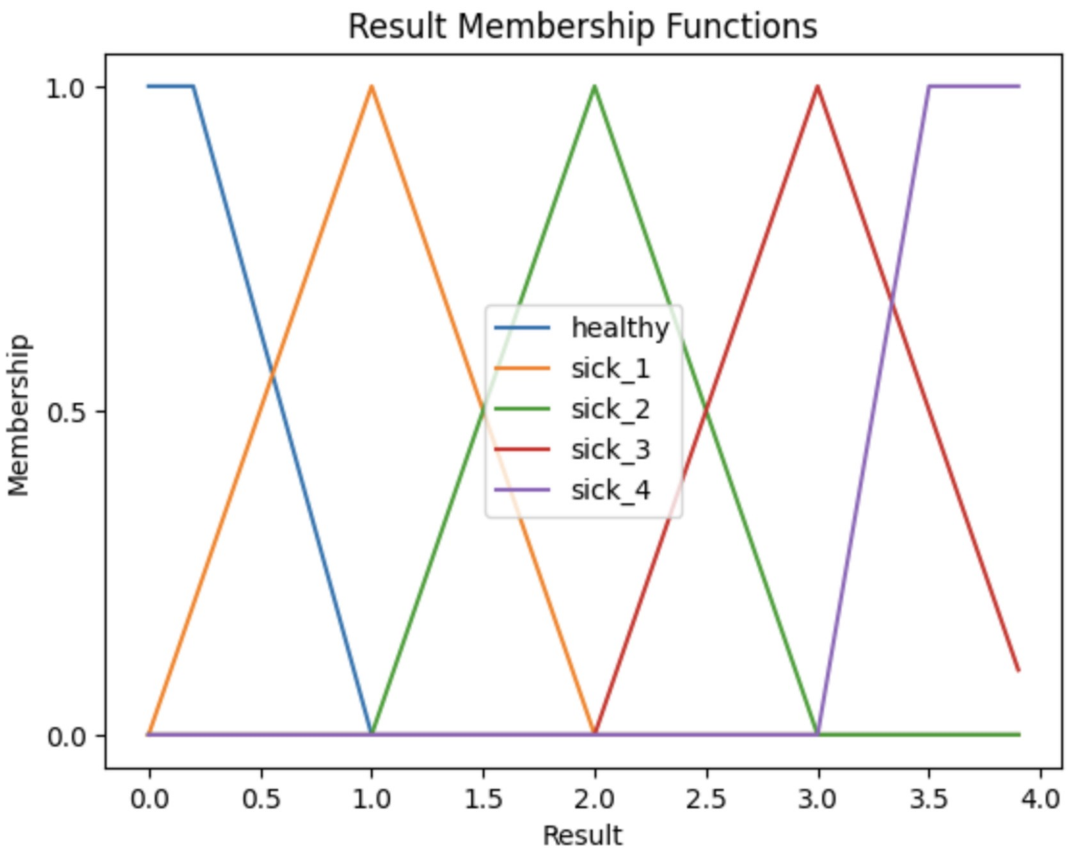


INPUT FIELD	RANGE	FUZZY SETS
Old Peak	<2 1.5-4.2 2.55>	Low Risk Terrible

INPUT FIELD	RANGE	FUZZY SETS
Age	<38	Young
	33-45	Mild
	40-58	Old
	52>	Very old



OUTPUT FIELD	RANGE	FUZZY SETS
Result	<1.78	Healthy
	1-2.51	Sick (s1)
	1.78-3.25	Sick (s2)
	1.5-4.5	Sick (s3)
	3.25>	Sick (s4)



Writing and Adding the Rules to the Control System

- The original paper had 44 rules, I added in a few more rules implementing AND and OR based complex rules based on rules from other papers and analysis from the machine learning models.
- Each rule has an antecedent from the input variables, and the consequent output result.

```python
rules.append(ctrl.Rule(antecedent=((age['very_old'] & chest_pain['atypical_anginal'])), consequent=health['sick_4']))
rules.append(ctrl.Rule(antecedent=((maximum_heart_rate['high'] & age['old'])), consequent=health['sick_4']))
rules.append(ctrl.Rule(antecedent=((sex['male'] & maximum_heart_rate['medium'])), consequent=health['sick_3']))
rules.append(ctrl.Rule(antecedent=((sex['female'] & maximum_heart_rate['medium'])), consequent=health['sick_2']))
rules.append(ctrl.Rule(antecedent=((chest_pain['non_anginal_pain'] & blood_pressure['high'])), consequent=health['sick_3']))
rules.append(ctrl.Rule(antecedent=((chest_pain['typical_anginal'] & maximum_heart_rate['medium'])), consequent=health['sick_2'])
rules.append(ctrl.Rule(antecedent=((blood_sugar['true'] & age['mild'])), consequent=health['sick_3']))
rules.append(ctrl.Rule(antecedent=((blood_sugar['false'] & blood_pressure['very_high'])), consequent=health['sick_2']))
rules.append(ctrl.Rule(antecedent=((chest_pain['asymptomatic'] | age['very_old'])), consequent=health['sick_1']))
rules.append(ctrl.Rule(antecedent=((blood_pressure['high'] | maximum_heart_rate['low'])), consequent=health['sick_1']))

rules.append(ctrl.Rule(antecedent=(chest_pain['typical_anginal']), consequent=health['healthy']))
rules.append(ctrl.Rule(antecedent=(chest_pain['atypical_anginal']), consequent=health['sick_1']))
rules.append(ctrl.Rule(antecedent=(chest_pain['non_anginal_pain']), consequent=health['sick_2']))
rules.append(ctrl.Rule(antecedent=(chest_pain['asymptomatic']), consequent=health['sick_3']))

rules.append(ctrl.Rule(antecedent=(chest_pain['asymptomatic']), consequent=health['sick_4']))

rules.append(ctrl.Rule(antecedent=(sex['female']), consequent=health['sick_1']))
rules.append(ctrl.Rule(antecedent=(sex['male']), consequent=health['sick_2']))

rules.append(ctrl.Rule(antecedent=(blood_pressure['low']), consequent=health['healthy']))
rules.append(ctrl.Rule(antecedent=(blood_pressure['medium']), consequent=health['sick_1']))
rules.append(ctrl.Rule(antecedent=(blood_pressure['high']), consequent=health['sick_2']))
rules.append(ctrl.Rule(antecedent=(blood_pressure['high']), consequent=health['sick_3']))
rules.append(ctrl.Rule(antecedent=(blood_pressure['very_high']), consequent=health['sick_4']))

rules.append(ctrl.Rule(antecedent=(cholesterol['low']), consequent=health['healthy']))
rules.append(ctrl.Rule(antecedent=(cholesterol['medium']), consequent=health['sick_1']))
rules.append(ctrl.Rule(antecedent=(cholesterol['high']), consequent=health['sick_2']))
rules.append(ctrl.Rule(antecedent=(cholesterol['high']), consequent=health['sick_3']))
rules.append(ctrl.Rule(antecedent=(cholesterol['very_high']), consequent=health['sick_4']))

rules.append(ctrl.Rule(antecedent=(blood_sugar['true']), consequent=health['sick_2']))
```

```python
rules.append(ctrl.Rule(antecedent=(ECG['normal']), consequent=health['healthy']))
rules.append(ctrl.Rule(antecedent=(ECG['normal']), consequent=health['sick_1']))
rules.append(ctrl.Rule(antecedent=(ECG['abnormal']), consequent=health['sick_2']))
rules.append(ctrl.Rule(antecedent=(ECG['hypertrophy']), consequent=health['sick_3']))
rules.append(ctrl.Rule(antecedent=(ECG['hypertrophy']), consequent=health['sick_4']))

rules.append(ctrl.Rule(antecedent=(exercise['true']), consequent=health['sick_2']))

rules.append(ctrl.Rule(antecedent=(maximum_heart_rate['low']), consequent=health['healthy']))
rules.append(ctrl.Rule(antecedent=(maximum_heart_rate['medium']), consequent=health['sick_1']))
rules.append(ctrl.Rule(antecedent=(maximum_heart_rate['medium']), consequent=health['sick_2']))
rules.append(ctrl.Rule(antecedent=(maximum_heart_rate['high']), consequent=health['sick_3']))
rules.append(ctrl.Rule(antecedent=(maximum_heart_rate['high']), consequent=health['sick_4']))

rules.append(ctrl.Rule(antecedent=(old_peak['low']), consequent=health['healthy']))
rules.append(ctrl.Rule(antecedent=(old_peak['low']), consequent=health['sick_1']))
rules.append(ctrl.Rule(antecedent=(old_peak['terrible']), consequent=health['sick_2']))
rules.append(ctrl.Rule(antecedent=(old_peak['terrible']), consequent=health['sick_3']))
rules.append(ctrl.Rule(antecedent=(old_peak['risk']), consequent=health['sick_4']))

rules.append(ctrl.Rule(antecedent=(thallium['normal']), consequent=health['healthy']))
rules.append(ctrl.Rule(antecedent=(thallium['normal']), consequent=health['sick_1']))
rules.append(ctrl.Rule(antecedent=(thallium['fixed_defect']), consequent=health['sick_2']))
rules.append(ctrl.Rule(antecedent=(thallium['reversible_direct']), consequent=health['sick_3']))
rules.append(ctrl.Rule(antecedent=(thallium['reversible_direct']), consequent=health['sick_4']))

rules.append(ctrl.Rule(antecedent=(age['young']), consequent=health['healthy']))
rules.append(ctrl.Rule(antecedent=(age['mild']), consequent=health['sick_1']))
rules.append(ctrl.Rule(antecedent=(age['old']), consequent=health['sick_2']))
rules.append(ctrl.Rule(antecedent=(age['old']), consequent=health['sick_3']))
rules.append(ctrl.Rule(antecedent=(age['very_old']), consequent=health['sick_4']))
```

Sample Test Case

```
health_prediction.input['age'] = 75

health_prediction.input['chest_pain'] = 8

health_prediction.input['maximum_heart_rate'] =
110

health_prediction.input['sex'] = 1

health_prediction.input['blood_pressure'] = 120

health_prediction.input['blood_sugar'] = 1

health_prediction.input['cholesterol'] = 180

health_prediction.input['ECG'] = 2

health_prediction.input['old_peak'] = 3

health_prediction.input['thallium'] = 3

health_prediction.input['exercise'] = 1
```

Predict results

```
health_prediction.compute()
predicted_health = health_prediction.output['health']
```

```
predicted_health
```

```
2.0
```

Find health status percentage

```
for level, (lower, upper) in health_levels.items():
  if lower <= predicted_health <= upper:
    percentage = ((predicted_health - lower) / (upper - lower)) * 100
    print(f"Predicted Health: {percentage:.2f}% ({level})")
```

```
Predicted Health: 8.00% (healthy)
```

Challenges

When building this system, there were a few challenges that I faced:

- Defining the right ranges for the membership functions.
- Correctly fuzzifying and defuzzifying the input.
- Building the rule base - adding extra rules to the ones from the paper.

Future Potential of Fuzzy Logic and Expert Systems

- **Remote Patient Monitoring:** real-time assessments, trigger interventions and ensure proactive healthcare.
- **Drug Discovery:** analyze potential side effects, predict drug interactions and analyze complex biological data.
- **Chronic Disease Management:** continuous monitoring, personalized treatment plans, and early detection of worsening health.

Q & A

Thank You!