

Faculty of Computers and Artificial Intelligence,

Cairo University

**Reasoning and Knowledge**

**Representation**

**Detection of Cardiac Arrhythmia**

**Coronary Heart Diseases (CHD)**

**Prepared By:**

Christina Montasser 20190382 S1

Mirette Amin 20190570 S1

Nader Atef 20190575 S1

Spring 2022

**Contents**

[1. CHD 3](#_Toc103796352)

[1.1. App Description 3](#_Toc103796353)

[1.2. Fuzzy Logic 3](#_Toc103796354)

[2. Knowledge Base 4](#_Toc103796355)

[2.1. Input 4](#_Toc103796356)

[2.2. Output 4](#_Toc103796357)

[2.3. Rules 5](#_Toc103796358)

[3. Code 5](#_Toc103796359)

[4. Code Outputs 8](#_Toc103796360)

[5. Fuzzify & Defuzzify Implementation 10](#_Toc103796361)

[5.1. Fuzzification Implementation from scratch 10](#_Toc103796367)

[5.2. Defuzzification Implementation from scratch 11](#_Toc103796368)

[5.3. Codes Output 12](#_Toc103796369)

[References 13](#_Toc103796370)

# **CHD**

## **App Description**

In this application, we’re targeting to classify people who are more likely to get Coronary Heart Disease (CHD) in 10 years.

We used dataset downloaded from Kaggle and is collected from an ongoing cardiovascular study on residents of the town of Framingham, Massachusetts (Framingham\_CHD\_preprocessed\_data, n.d.).

The chosen features and rules were decided by asking some experts (doctors) to help us encode their experiences so we created the model using Fuzzy logic as it’s a useful technique to create a human-like thinking and predict the results accordingly as if it’s a doctor who predicts.

The dataset we chose contains the person’s:

• Sex   
• Age   
• Education: 0: Less than High School and High School degrees, 1: College Degree and Higher  
• Current Smoker: If the patient is a current smoker  
• Cigs Per Day: Cigarettes’ number that the patient smoke on average daily.   
• BP Meds: If the patient previously was on blood pressure medication   
• Prevalent Stroke: If the patient previously had a stroke   
• Prevalent Hyp: If the patient is hypertensive   
• Diabetes: If the patient previously has diabetes   
• Tot Chol: Cholesterol level

• Sys BP: Systolic blood pressure   
• Dia BP: Diastolic blood pressure   
• BMI: Body Mass Index   
• Heart Rate

• Glucose: Glucose level

• Ten Year CHD: binary output

## **Fuzzy Logic**

Fuzzy logic is a technique to encode an expert’s experience in a specific field.

One variable can process many possible truths and each truth value lies between 0 and 1 which means that the computing is based on the degree of the truth which gives the enough flexibility to embody human-like thinking in control systems and make decisions accordingly.

# **Knowledge Base**

## **Input**

**The input** we chose was: {Age, Cigs Per Day, Tot Chol, BMI, Sys BP}

According to experts, these features significantly affect the judgment whether the patient can get CHD or not, and the other features some of them help but we decided to take the most significant ones and dropped the others.

* + 1. **Membership Functions**

1. **age:** ranges from 0 to 100 years

* FuzzySet “Child” trapezoidal x-coordinates = (-1, 0, 16, 26)
* FuzzySet “Adult” trapezoidal x-coordinates = (18, 28, 42, 50)
* FuzzySet “Old” trapezoidal x-coordinates = (45, 50, 79, 80)

1. **cigsPerDay:** ranges from 0 to 60

* FuzzySet “Normal” trapezoidal x-coordinates = (-1, 0, 5, 12)
* FuzzySet “Heavy” trapezoidal x-coordinates = (10, 30, 59, 60)

1. **totChol:** ranges from 0 to 400

* FuzzySet “Normal” trapezoidal x-coordinates = (-1, 0, 150, 200)
* FuzzySet “Medium” triangular x-coordinates = (170, 200, 240)
* FuzzySet “Medium” trapezoidal x-coordinates = (235, 265, 499, 500)

1. **BMI:** ranges from 0 to 60

* FuzzySet “Normal” triangular x-coordinates = (0, 18, 23)
* FuzzySet “OverWeight” triangular x-coordinates = (23, 26, 30)
* FuzzySet “Obese” trapezoidal x-coordinates = (30, 40, 59, 60)

1. **sysBP:** ranges from 0 to 400

* FuzzySet “Normal” trapezoidal x-coordinates = (-1, 0, 120, 130)
* FuzzySet “Medium” triangular x-coordinates = (125, 135, 160)
* FuzzySet “High” trapezoidal x-coordinates = (150, 170, 399, 400)

## **Output**

10 years risk of coronary heart disease CHD

* + 1. **Membership Function**

**TenYearCHD:** ranges from 0 to 1

## **Rules**

We created 25 rules by asking doctors, searching on the internet and depending on Detection of Cardiac arrhythmia using fuzzy logic paper (Detection of Cardiac arrhythmia using fuzzy logic).

# **Code**

1. import skfuzzy as fuzz
2. import numpy as np
3. import pandas as pd
4. from skfuzzy import control as ctrl
5. dataFrame = pd.read\_csv('CHD.csv')
6. dataFrame.head()
7. # Membership Functions
8. cigsPerDay   = ctrl.Antecedent(np.arange(0, 60), 'cigsPerDay')
9. cigsPerDay['Normal'] = fuzz.trapmf(cigsPerDay.universe, [-1, 0, 5, 12])
10. cigsPerDay['Heavy'] = fuzz.trapmf(cigsPerDay.universe, [10,30 ,59, 60])
11. cigsPerDay.view()
12. age = ctrl.Antecedent(np.arange(0, 100), 'age')
13. age['Child'] = fuzz.trapmf(age.universe, [-1, 0, 16, 26])
14. age['Adult'] = fuzz.trapmf(age.universe, [18, 28, 42, 50])
15. age['Old'] = fuzz.trapmf(age.universe, [45, 50, 79, 80])
16. age.view()
17. sysBP  = ctrl.Antecedent(np.arange(0, 400), 'sysBP')
18. sysBP['Normal'] = fuzz.trapmf(sysBP.universe, [-1, 0, 120, 130])
19. sysBP['Medium'] = fuzz.trimf(sysBP.universe, [125, 135, 160])
20. sysBP['High'] = fuzz.trapmf(sysBP.universe, [150, 170, 399, 400])
21. sysBP.view()
22. totChol = ctrl.Antecedent(np.arange(0, 400), 'totChol')
23. totChol['Normal'] = fuzz.trapmf(totChol.universe, [-1, 0, 150, 200])
24. totChol['Medium'] = fuzz.trimf(totChol.universe, [170, 200, 240])
25. totChol['High'] = fuzz.trapmf(totChol.universe, [235, 265, 499, 500])
26. totChol.view()
27. BMi = ctrl.Antecedent(np.arange(0, 60,2), 'BMi')
28. BMi['Normal'] = fuzz.trimf(BMi.universe, [0, 18 , 23])
29. BMi['OverWeight'] = fuzz.trimf(BMi.universe, [23, 26, 30])
30. BMi['Obese'] = fuzz.trapmf(BMi.universe, [30, 40, 59, 60])
31. BMi.view()
32. TenYearCHD = ctrl.Consequent(np.arange(0, 1.1, 0.1), 'TenYearCHD')
33. TenYearCHD['True'] = fuzz.trapmf(TenYearCHD.universe, [0.3, 0.6, 1, 1.5])
34. TenYearCHD['False'] = fuzz.trapmf(TenYearCHD.universe, [-1, 0, 0.4, 0.6])
35. TenYearCHD['True'].view()

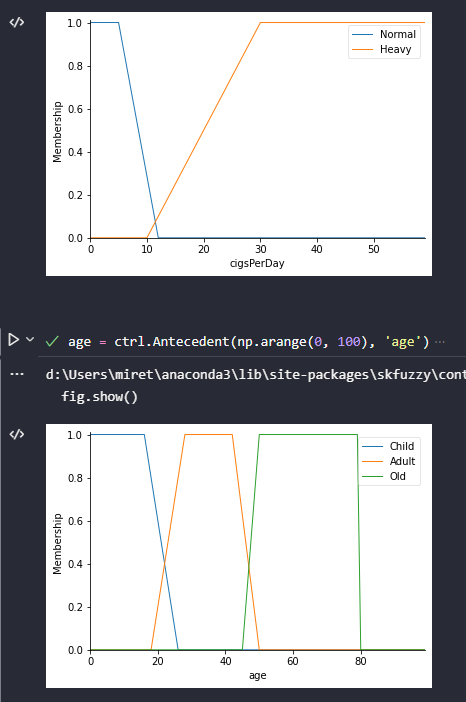
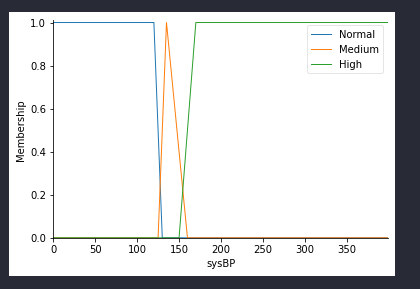
**# Rules for CHD --> False**

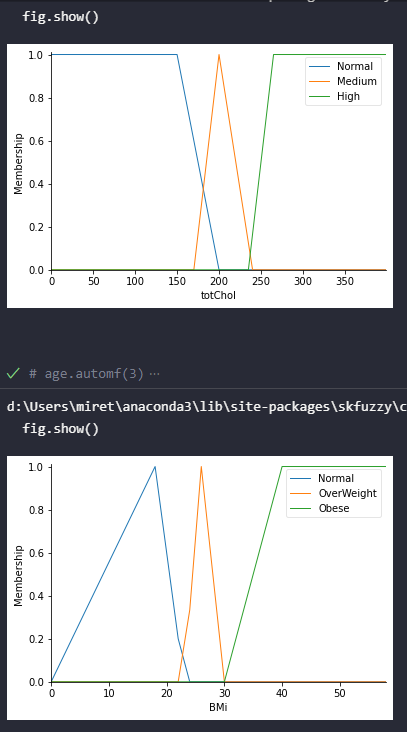
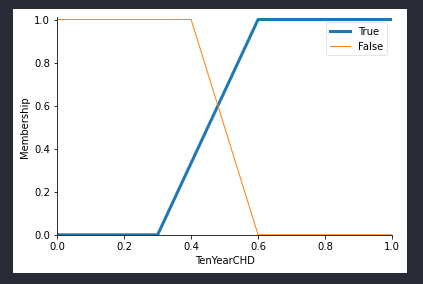
1. rule1 = ctrl.Rule( BMi['Normal'] & totChol['Normal'] & sysBP['Normal'] & cigsPerDay['Normal'], TenYearCHD['False'])
2. rule2 = ctrl.Rule( BMi['OverWeight'] & age['Adult'] & totChol['Medium'], TenYearCHD['False'])
3. rule3 = ctrl.Rule( age['Old'] & totChol['Normal'] & cigsPerDay['Normal'], TenYearCHD['False'])
4. rule4 = ctrl.Rule( BMi['OverWeight'] & age['Old'] & totChol['Medium'], TenYearCHD['False'])
5. rule5 = ctrl.Rule( BMi['Normal'] & age['Old'] & totChol['High'] & cigsPerDay['Normal'], TenYearCHD['False'])
6. rule6 = ctrl.Rule( BMi['Obese'] & age['Old'] & totChol['Normal'], TenYearCHD['False'])
7. rule7 = ctrl.Rule( BMi['OverWeight'] & age['Old'] & totChol['Normal'], TenYearCHD['False'])
8. rule8 = ctrl.Rule( BMi['Obese'] & age['Old'] & totChol['Normal'], TenYearCHD['False'])
9. rule9 = ctrl.Rule( BMi['OverWeight'] & age['Old'] & totChol['Normal'], TenYearCHD['False'])
10. rule10 = ctrl.Rule(age['Old'] & totChol['Normal'] & sysBP['High'] & cigsPerDay['Normal'], TenYearCHD['False'])
11. rule11 = ctrl.Rule(age['Old'] & totChol['Normal'] & sysBP['Normal'] , TenYearCHD['False'])
12. rule12 = ctrl.Rule(age['Old'] & sysBP['Normal'] & cigsPerDay['Normal'], TenYearCHD['False'])
13. rule13 = ctrl.Rule(age['Old'] & totChol['High'] & sysBP['High'] & cigsPerDay['Normal'], TenYearCHD['False'])
14. rule14 = ctrl.Rule(age['Child'] & BMi['Normal'], TenYearCHD['False'])

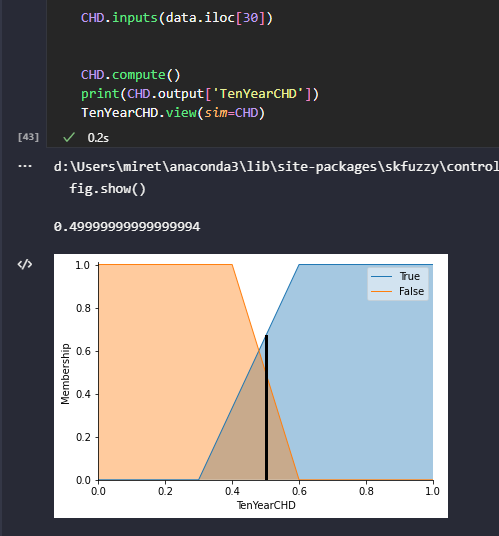
**# Rules for CHD --> True**

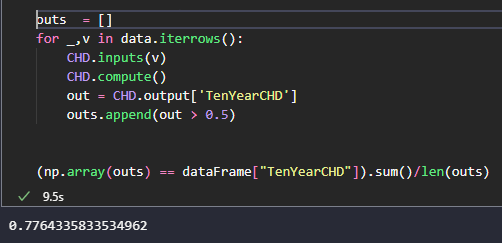
1. rule15 = ctrl.Rule( BMi['Obese'] | age['Adult'] | totChol['High'] | sysBP['High'] | cigsPerDay['Heavy'], TenYearCHD['True'])
2. rule16 = ctrl.Rule( BMi['Obese'] | age['Adult']  | totChol['High'] | sysBP['High'] | cigsPerDay['Heavy'], TenYearCHD['True'])
3. rule17 = ctrl.Rule( BMi['Obese'] & age['Old'] & totChol['Medium'] & sysBP['High'] & cigsPerDay['Heavy'], TenYearCHD['True'])
4. rule18 = ctrl.Rule( BMi['Obese'] & age['Old'] & totChol['High'] & sysBP['High'] & cigsPerDay['Heavy'], TenYearCHD['True'])
5. rule19 = ctrl.Rule( BMi['Obese'] & age['Old'] & totChol['Medium'] & sysBP['High'] & cigsPerDay['Heavy'], TenYearCHD['True'])
6. rule20 = ctrl.Rule( BMi['Obese'] & age['Old'] & totChol['High'] & sysBP['High'] & cigsPerDay['Heavy'], TenYearCHD['True'])
7. rule21 = ctrl.Rule( BMi['OverWeight'] & age['Old'] & totChol['High'] & sysBP['High'] & cigsPerDay['Heavy'], TenYearCHD['True'])
8. rule22 = ctrl.Rule( BMi['OverWeight'] & age['Old'] & totChol['High'] & sysBP['High'] & cigsPerDay['Heavy'], TenYearCHD['True'])
9. rule23 = ctrl.Rule( cigsPerDay['Normal'], TenYearCHD['False'])
10. rule24 = ctrl.Rule( totChol['Medium'], TenYearCHD['False'])
11. rule25 = ctrl.Rule( BMi['Normal'] | age['Adult']  | totChol['Normal'] | sysBP['Normal'] | cigsPerDay['Normal'], TenYearCHD['False'])
12. CHD\_Rules = ctrl.ControlSystem([rule1, rule3, rule4, rule5, rule6, rule7, rule8,rule9,rule10,rule11,rule12,rule13,rule14,rule15,rule16,rule17,rule18,rule19,rule20,rule21, rule22, rule23, rule24, rule25])
13. CHD = ctrl.ControlSystemSimulation(CHD\_Rules)
14. dataFrame=dataFrame.drop([  'education','currentSmoker','BPMeds','diabetes','male','prevalentStroke','prevalentHyp','diaBP','heartRate','glucose'], axis=1)
15. dataFrame= dataFrame.rename({'BMI':'BMi'}, axis=1)
16. data = dataFrame.drop(['TenYearCHD'], axis=1)
17. outs  = []
18. for \_,v in data.iterrows():
19. CHD.inputs(v)
20. CHD.compute()
21. out = CHD.output['TenYearCHD']
22. outs.append(out > 0.5)
23. (np.array(outs) == dataFrame["TenYearCHD"]).sum()/len(outs)
24. dataFrame["TenYearCHD"].value\_counts() /len(dataFrame)

# **Code Outputs**









# **Fuzzify & Defuzzify Implementation**



## **Fuzzification Implementation from scratch**

* + 1. **Description**
* ***When the crisp value lies in the lines’ shape ranges:***

Mainly using slope to find the fuzzy value.

Slope of each line for both shape (trapezoid, triangle) is calculated by that given ranges points.

**Triangle:**

Equation for left-line:

Slope = = (calculated slope by the given points)

fuzzyValue= slope\*(CrispValue – range0)

Equation for right-line:

Slope = = (calculated slope by the given points)

fuzzyValue= slope\*(CrispValue – range1) + 1

**Trapezoid** is similar, but there are two peaks (range1 & range2) which make a little change in the equations but with the same concept…

* ***When lies in the edge points or outside the shape ranges, fuzzy value = 0***
* ***When lies in/ between the middle or the peak point/s, fuzzy value =1***
  + 1. **Code**

1. def trapmf(fuzzySets, crispValue):
2. if(fuzzySets[1]- fuzzySets[0]!=0):
3. slope1 = (1-0)/( fuzzySets[1]- fuzzySets[0])
4. if(fuzzySets[2]- fuzzySets[3]!=0):
5. slope2 = (1-0)/( fuzzySets[2]- fuzzySets[3])
7. if (fuzzySets[1]<=crispValue <= fuzzySets[2]):
8. fuzzyValue = 1
9. elif(fuzzySets[0]<=crispValue < fuzzySets[1]):
10. fuzzyValue = (crispValue -fuzzySets[0])\* slope1
11. elif(fuzzySets[2]< crispValue <= fuzzySets[3]):
12. fuzzyValue = ((crispValue -fuzzySets[2]) \* slope2 ) + 1
13. else:
14. fuzzyValue =0
15. return fuzzyValue
16. def trimf(fuzzySets, crispValue):
17. if( fuzzySets[1]- fuzzySets[0] !=0 ):
18. slope1 = (1-0)/( fuzzySets[1]- fuzzySets[0])
19. if(fuzzySets[1]- fuzzySets[2]!=0):
20. slope2 = (1-0)/( fuzzySets[1]- fuzzySets[2])
21. if (crispValue==fuzzySets[1]):
22. fuzzyValue = 1
23. elif(fuzzySets[0]<=crispValue < fuzzySets[1]):
24. fuzzyValue = (crispValue -fuzzySets[0])\* slope1
25. elif(fuzzySets[1]< crispValue <= fuzzySets[2]):
26. fuzzyValue = ((crispValue -fuzzySets[1]) \* slope2) + 1
27. else:
28. fuzzyValue =0
29. return fuzzyValue
30. def fuzzify(fuzzySets, crispValue):
31. memberships = []
32. for i in range (len(fuzzySets)):
33. if(len(fuzzySets[i])==4):
34. memberships.append(trapmf(fuzzySets[i], crispValue))
35. elif(len(fuzzySets[i])==3):
36. memberships.append(trimf(fuzzySets[i], crispValue))
37. else:
38. memberships.append("wrong membership")
39. return memberships

## **Defuzzification Implementation from scratch**

* + 1. **Description**

Defuzzify function will take 2 inputs: membership functions values and fuzzysets values and return the crisp value.

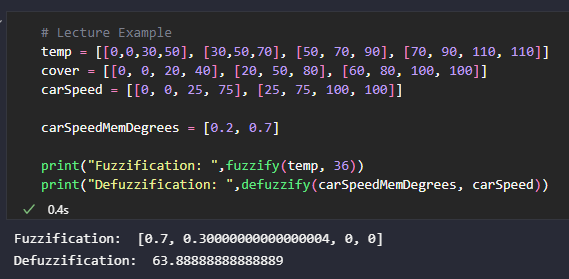
By calculating the centroids of each fuzzy set by summing its values then dividing by its length (if it’s trapezoid it will add 4 values and divide it by 4 and triangular will be 3)

Then the resulted centroids will be multiplied by membership values then divide it by membership values sum which gives crisp out value.

* + 1. **Code**

1. import numpy as np
2. def defuzzify(memberships, fuzzySets):
3. centroids = []
4. for i in range(len(fuzzySets)):
5. temp = np.array(fuzzySets[i])
6. centroids.append(np.sum(temp)/len(fuzzySets[i]))
7. centroids = np.array(centroids)
8. crispOutput = np.sum((centroids \* memberships))/ np.sum(memberships)
9. return crispOutput

## **Codes Output**



# **References**

*Detection of Cardiac arrhythmia using fuzzy logic.* (n.d.). Retrieved from ScienceDirect: https://www.sciencedirect.com/science/article/pii/S2352914819301996?via%3Dihub

*Framingham\_CHD\_preprocessed\_data*. (n.d.). Retrieved from Kaggle: https://www.kaggle.com/datasets/captainozlem/framingham-chd-preprocessed-data