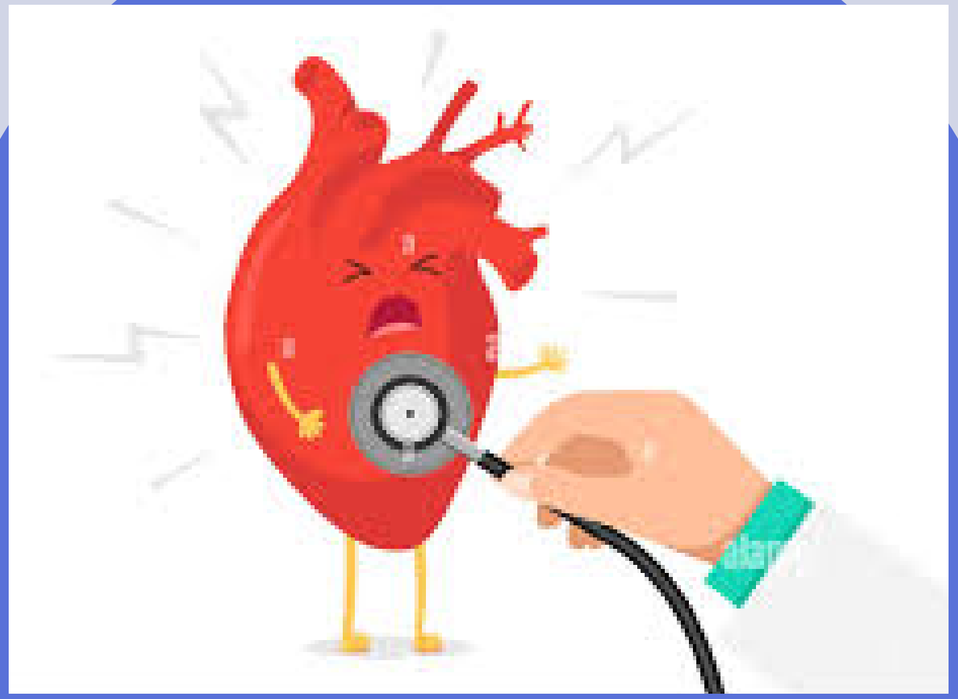
CARDIAC DISEASE DETECTION USING ANFIS-GA



NUR BIDAYATUL HIDAYAH BINTI ABDUL KADER
2013180
SITI SALWA BINTI MUHAMMAD RAIS
2015454
NUR AZSHIMADATHUL ASYQIN BINTI DARWIS
1910828

Table Of Contents

About Project

Intelligent Model

Why Neural Network
why ANFIS-GA

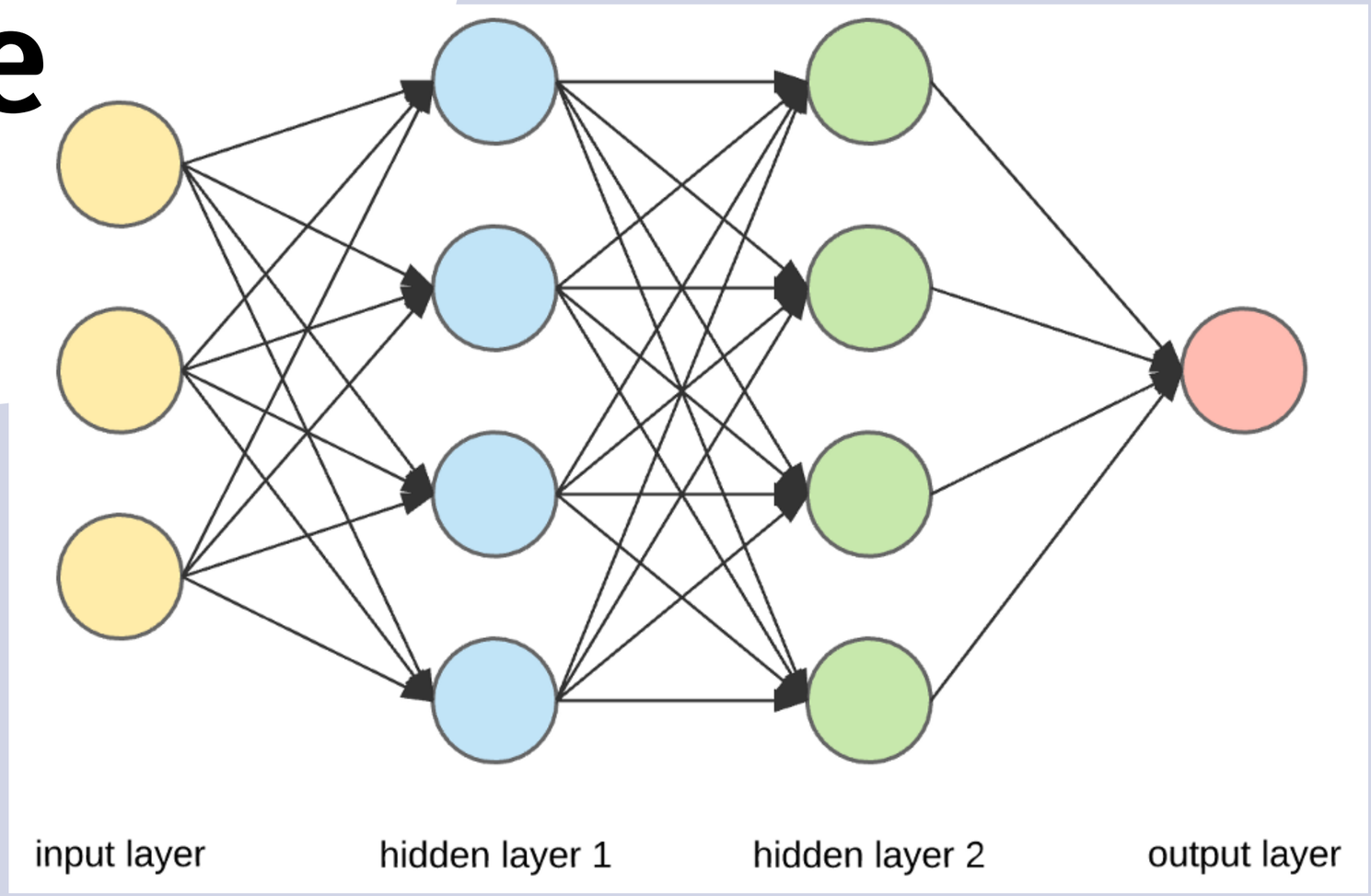
Heart disease prediction

- consider related data with disease
- gives values to parameter
- normalization
- training

01

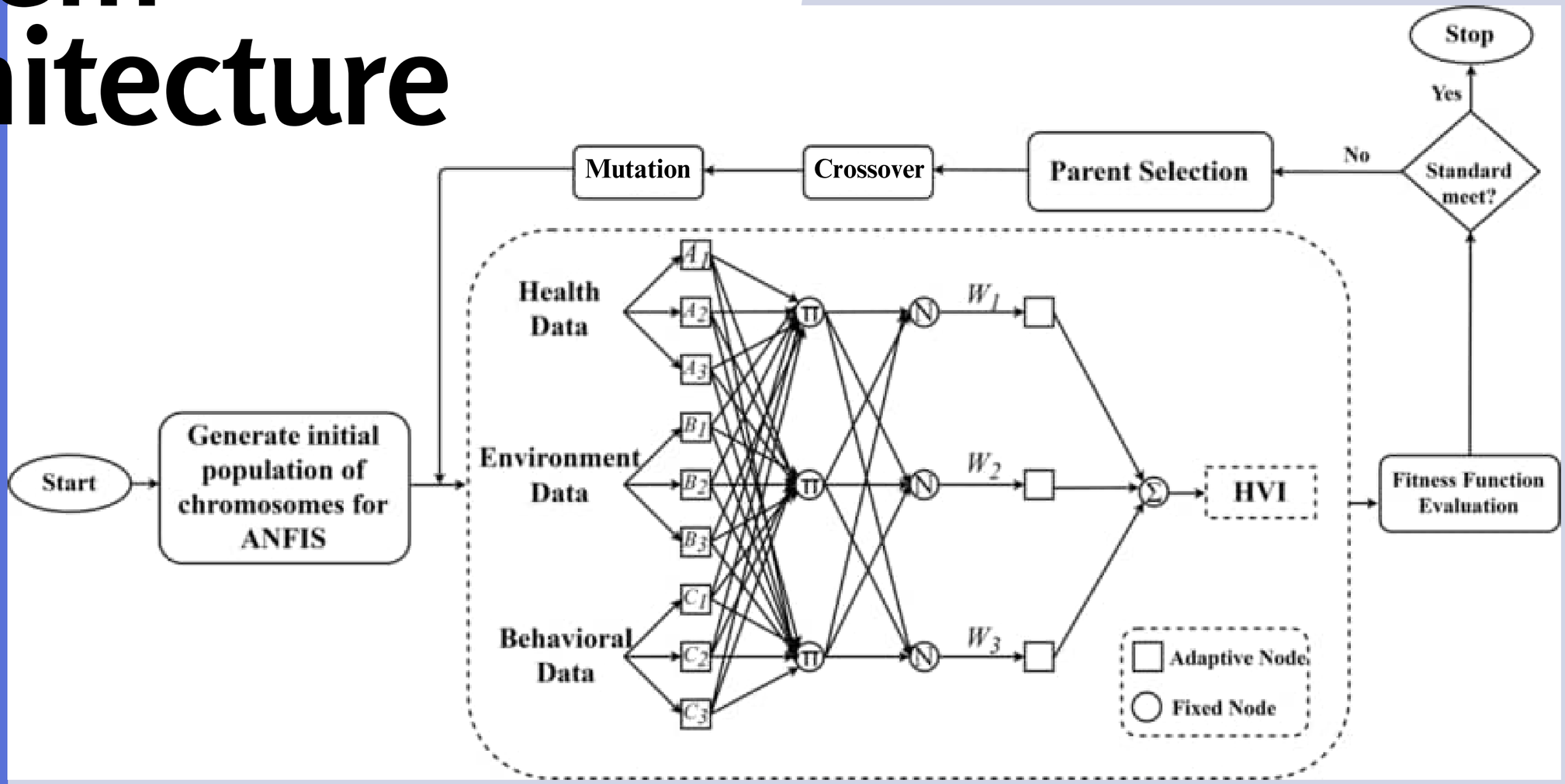
02

System Architecture



02

System Architecture



03

Methodology

1. **Data Processing**

- Normalizing data

2. Data Splitting

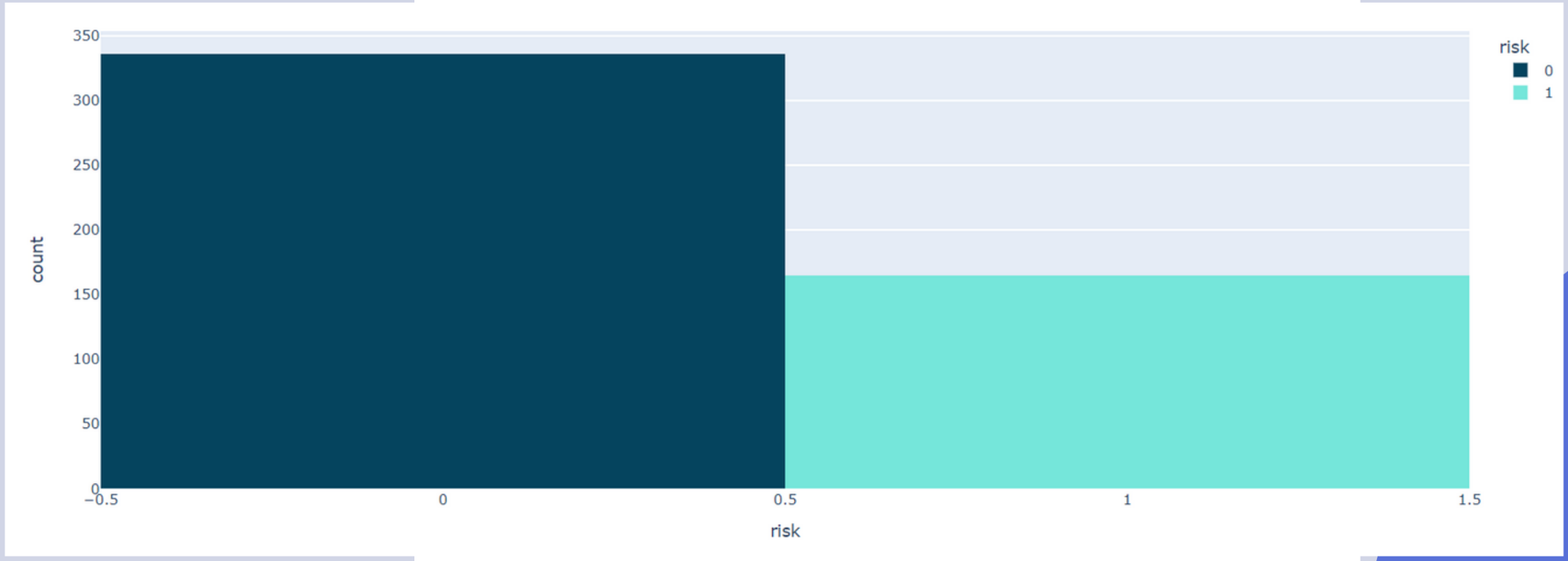
- Split the dataset into training and testing sets.

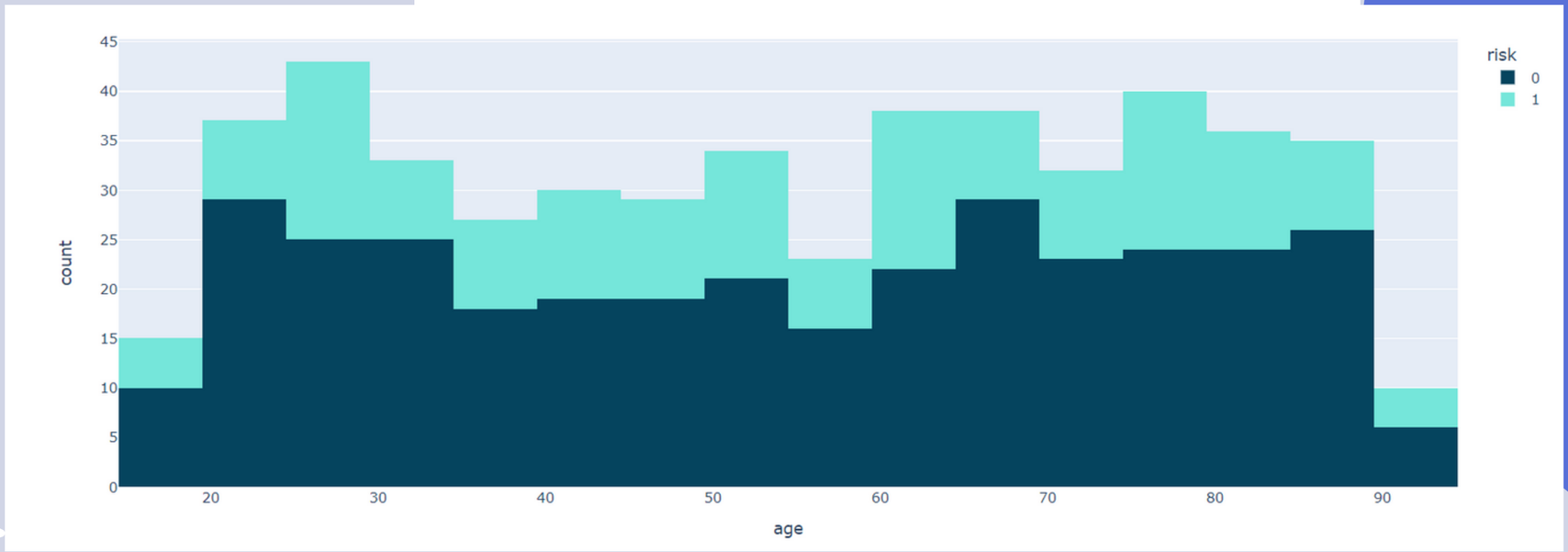
3. Initialization and Fine Tuning Parameters

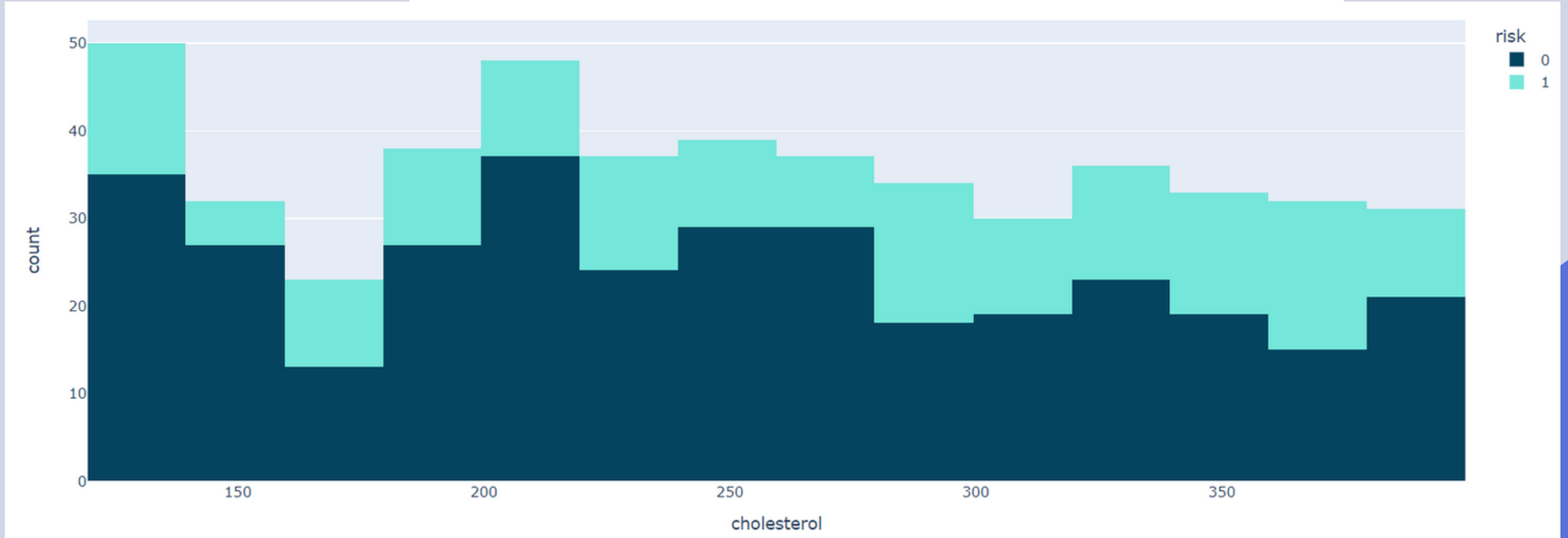
- functions = 3
- generations = 300, 50, 500
- offsprings = 10
- mutationRate = 0.1, 0.002
- CrossoverRate = 0.9
- learningRate = 0.01,
- chance = 0.5
- ruleComb = "simple"

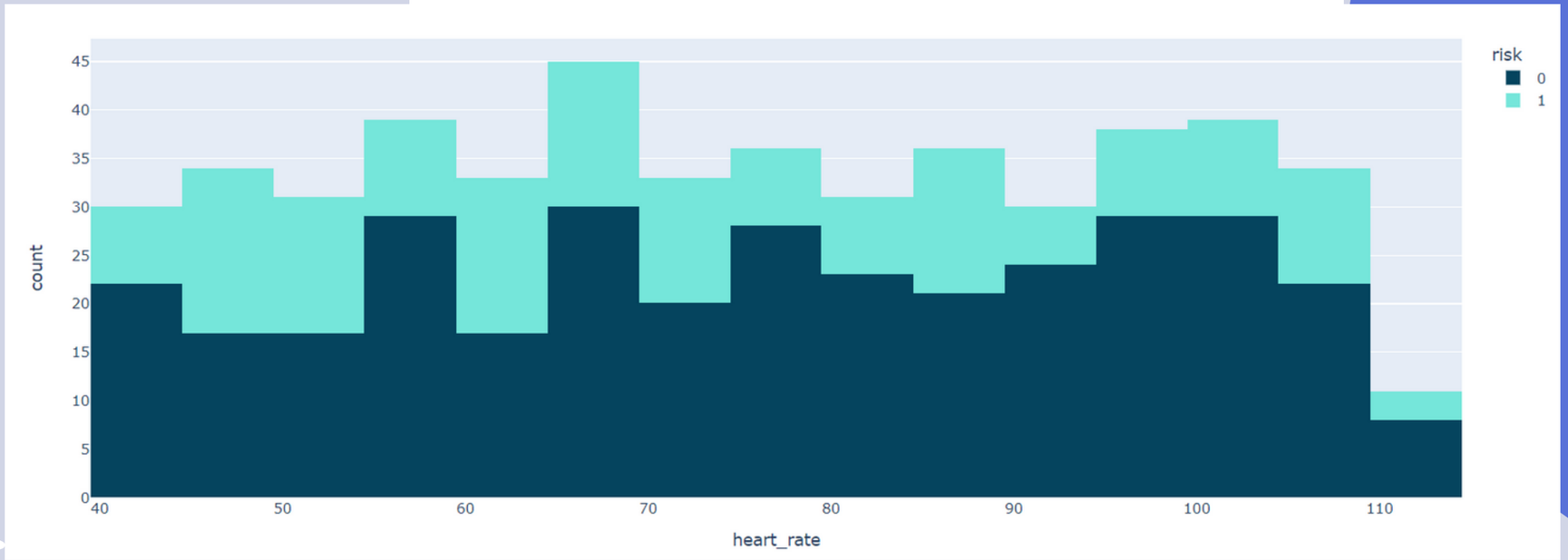
04

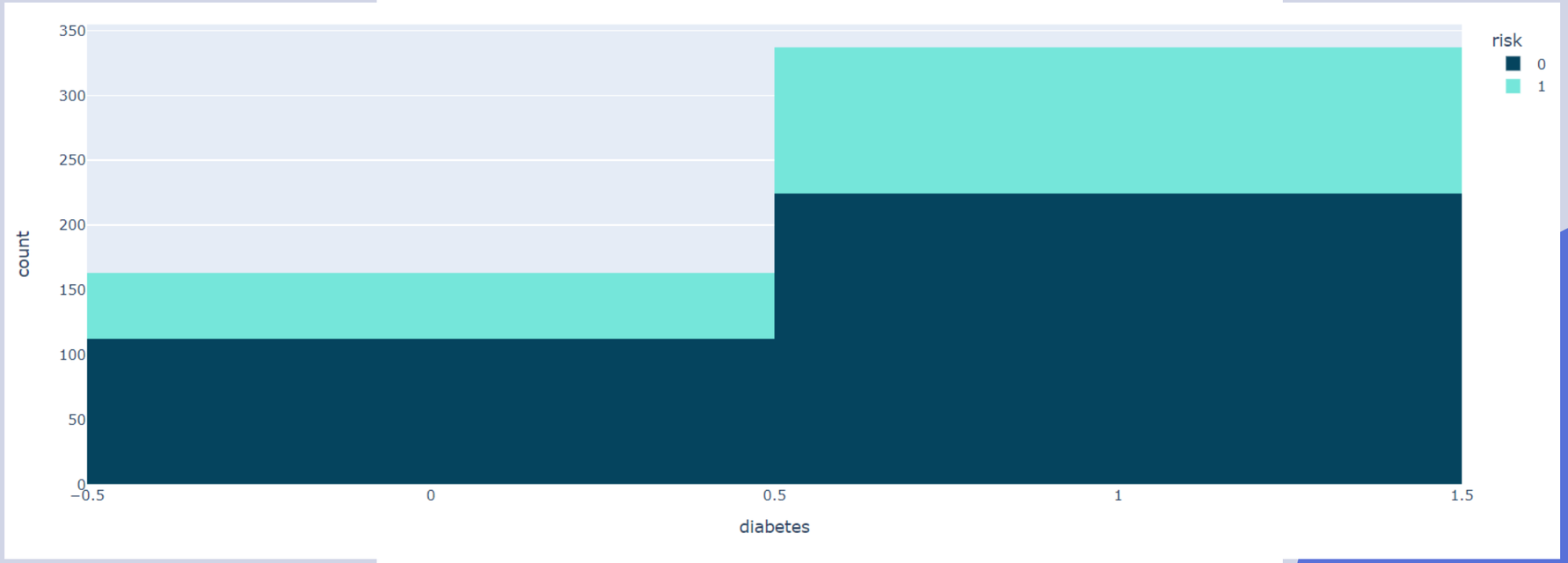
Data Analysis

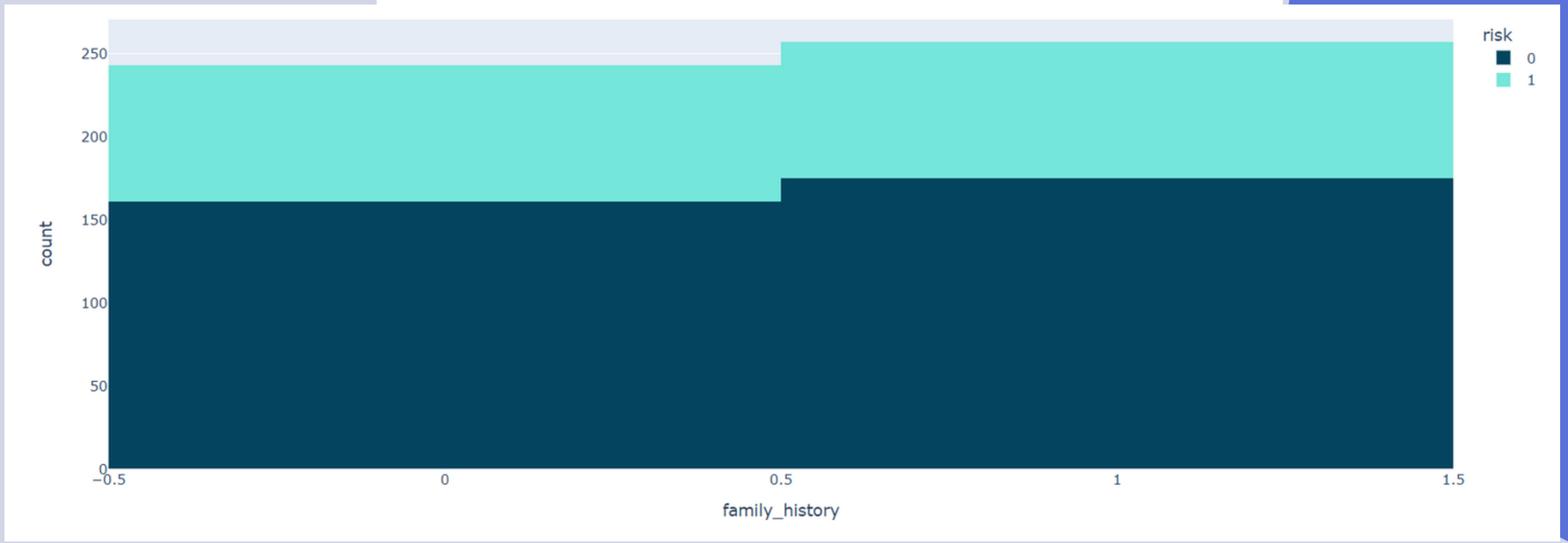

risk
0
1
count
350
300
250
200
150
100
50
0
−0.5
0
0.5
1
1.5
risk


risk
0
1
count
45
40
35
30
25
20
15
10
5
0
20
30
40
50
60
70
80
90
age


risk
0
1
count
50
40
30
20
10
0
150
200
250
300
350
cholesterol


risk
0
1
count
45
40
35
30
25
20
15
10
5
0
40
50
60
70
80
90
100
110
heart_rate


risk
0
1
count
350
300
250
200
150
100
50
0
−0.5
0
0.5
1
1.5
diabetes


risk
0
1
count
250
200
150
100
50
0
−0.5
0
0.5
1
1.5
family_history

	count	mean	std	min	25%	50%	75%	max
risk	1000.0	0.334000	0.471876	0.000000	0.000000	0.000000	1.000000	1.000000
age	1000.0	53.857000	21.724099	18.000000	35.000000	54.000000	73.000000	90.000000
cholesterol	1000.0	257.169000	80.583212	120.000000	194.000000	252.000000	326.000000	400.000000
heart_rate	1000.0	75.617000	20.087355	40.000000	58.000000	76.000000	93.000000	110.000000
diabetes	1000.0	0.672000	0.469720	0.000000	0.000000	1.000000	1.000000	1.000000
family_history	1000.0	0.505000	0.500225	0.000000	0.000000	1.000000	1.000000	1.000000
smoking	1000.0	0.881000	0.323951	0.000000	1.000000	1.000000	1.000000	1.000000
obesity	1000.0	0.477000	0.499721	0.000000	0.000000	0.000000	1.000000	1.000000
alcohol	1000.0	0.597000	0.490746	0.000000	0.000000	1.000000	1.000000	1.000000
exercise	1000.0	9.795311	5.876251	0.004000	4.479250	9.510000	15.037000	19.999000
previous_problems	1000.0	0.485000	0.500025	0.000000	0.000000	0.000000	1.000000	1.000000
medication	1000.0	0.497000	0.500241	0.000000	0.000000	0.000000	1.000000	1.000000
stress_level	1000.0	5.398000	2.806327	1.000000	3.000000	5.000000	8.000000	10.000000
sedentary	1000.0	6.134730	3.462298	0.002000	3.082320	6.189809	9.262017	11.992000
bmi	1000.0	28.822092	6.368839	18.004211	23.200693	28.728500	34.309725	39.997211
trigl	1000.0	419.299000	221.226827	30.000000	231.750000	419.500000	601.250000	800.000000
physical_activity	1000.0	3.468000	2.283460	0.000000	1.000000	3.000000	5.000000	7.000000
sleep_hour	1000.0	6.973000	2.013038	4.000000	5.000000	7.000000	9.000000	10.000000

04

Findings

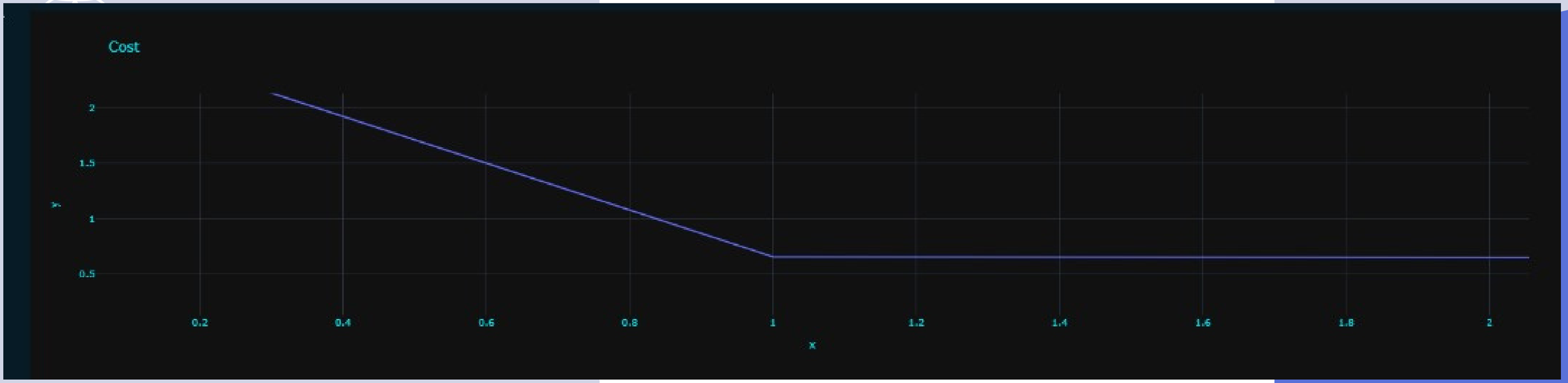
Neural Network

	Learning_Rate	Layers	Epochs	Accuracy
Model_1	0.100000	[16, 100, 100, 100, 1]	2000	0.315000
Model_2	0.001000	[16, 16, 1]	3000	0.315000
Model_3	0.000100	[16, 16, 1]	3000	0.685000
Model_4	0.000100	[16, 16, 1]	30000	0.685000
Model_5	0.000100	[16, 16, 16, 1]	30000	0.685000
Model_6	0.000100	[16, 16, 16, 16, 1]	30000	0.315000
Model_7	0.000100	[16, 32, 32, 1]	30000	0.685000
Model_8	0.000100	[16, 128, 128, 1]	30000	0.685000

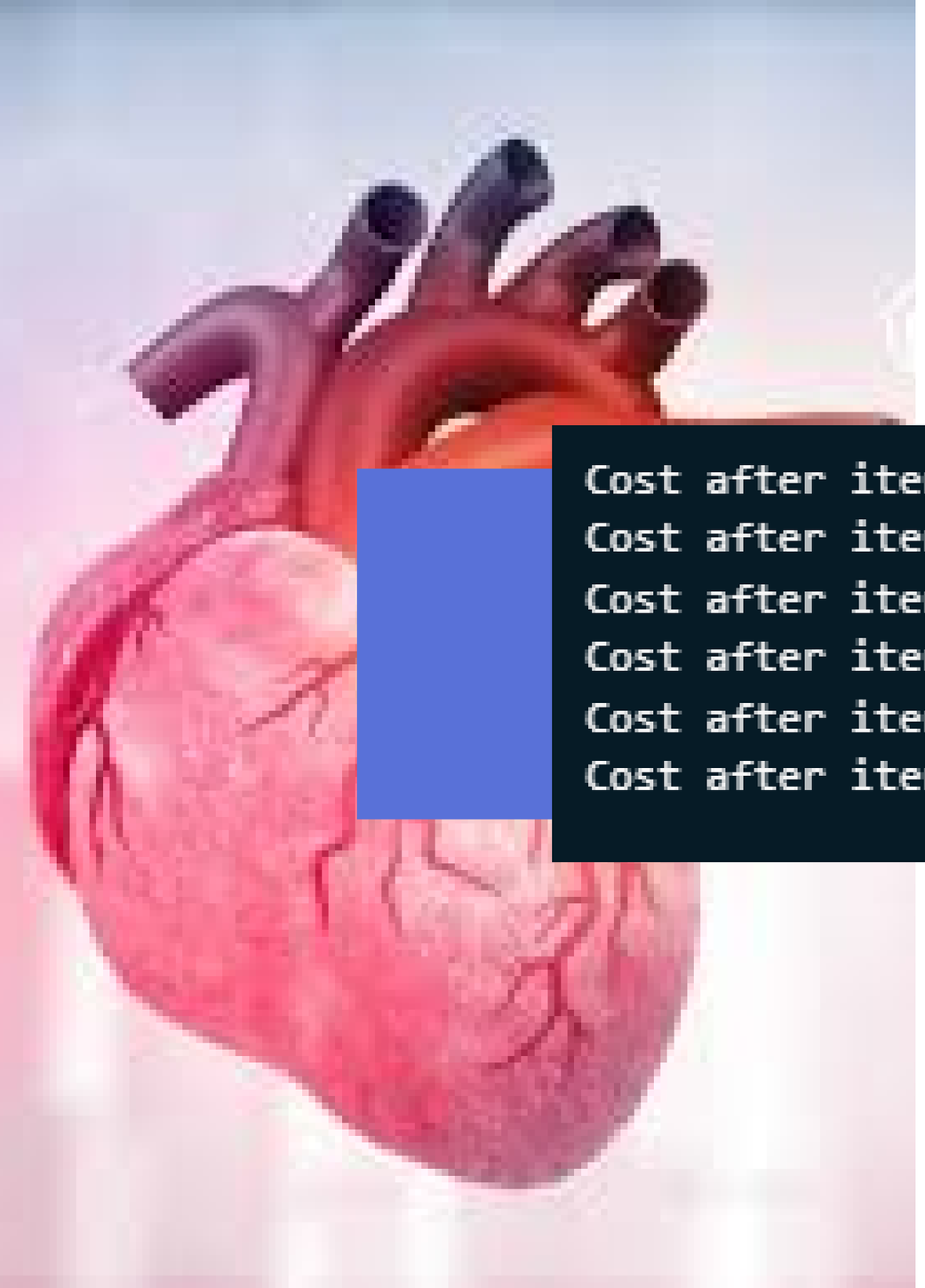
-result fine tune hidden layer

-no of neuron hidden layer

Neural Network



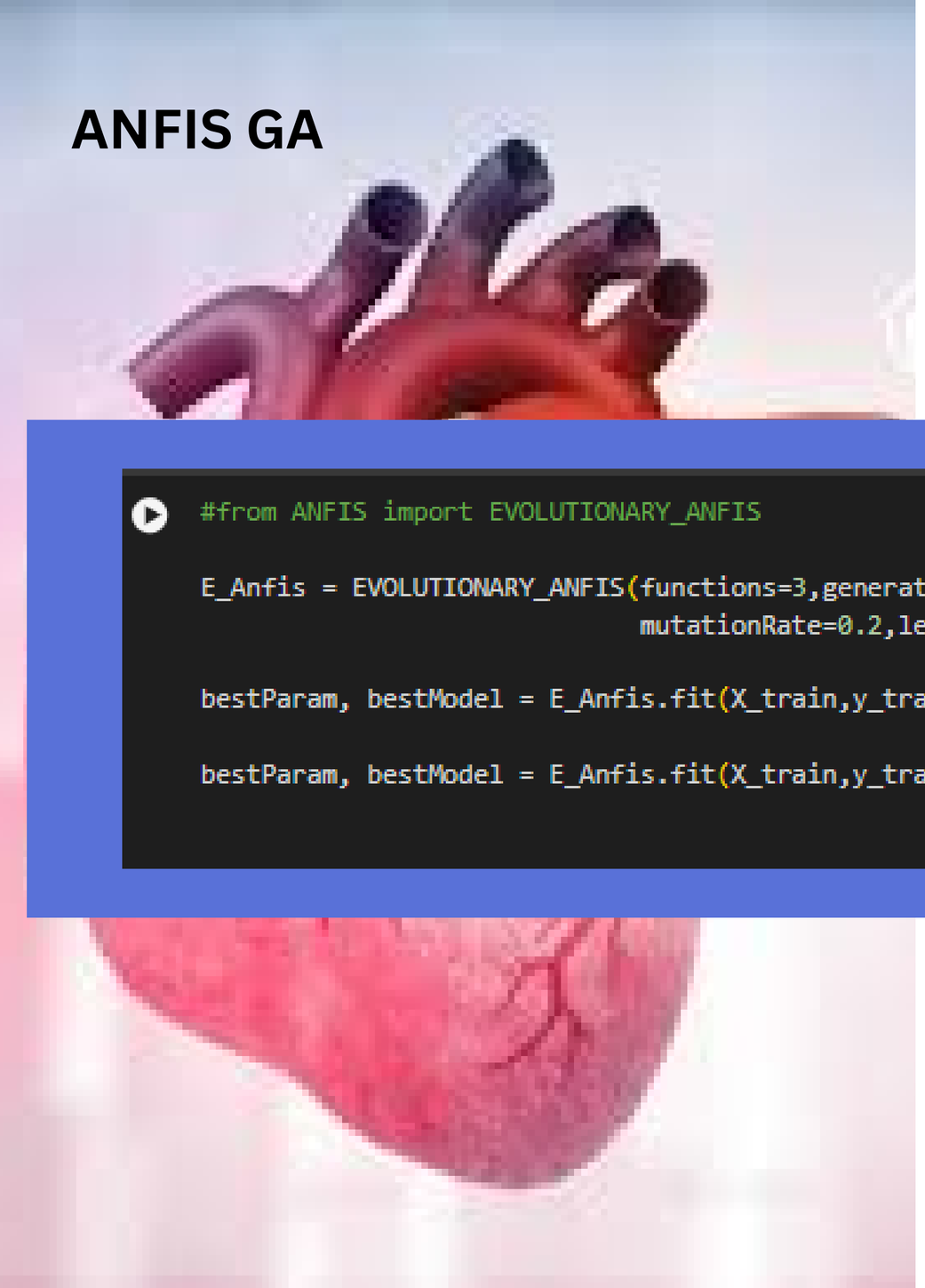
The results, converge



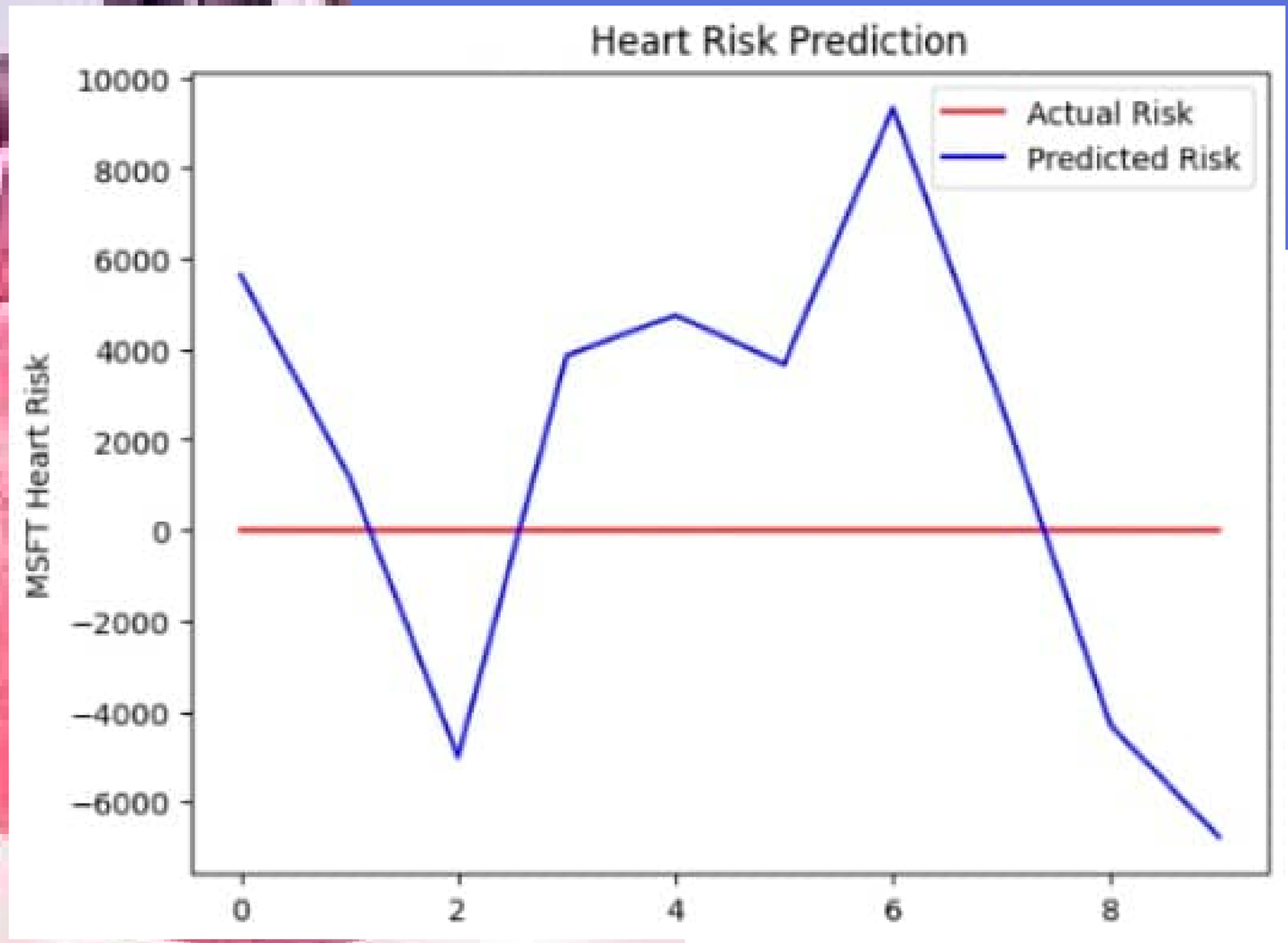
```
Cost after iteration 0: 2.761261976546644
Cost after iteration 5000: 0.6556041911703235
Cost after iteration 10000: 0.6507210899167413
Cost after iteration 15000: 0.6474279800161966
Cost after iteration 20000: 0.6451902173713114
Cost after iteration 25000: 0.6437173549032525
```

```
Test_risk = [0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 0 1 0
 0 0 1 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0
 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 1 0 1 0 1 0 1 1 1 0 1 0 0 1 0 1 1 1 0 1 0
 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0
 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 1 0 1
 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0]
Accuracy =  0.6849999999999999
```

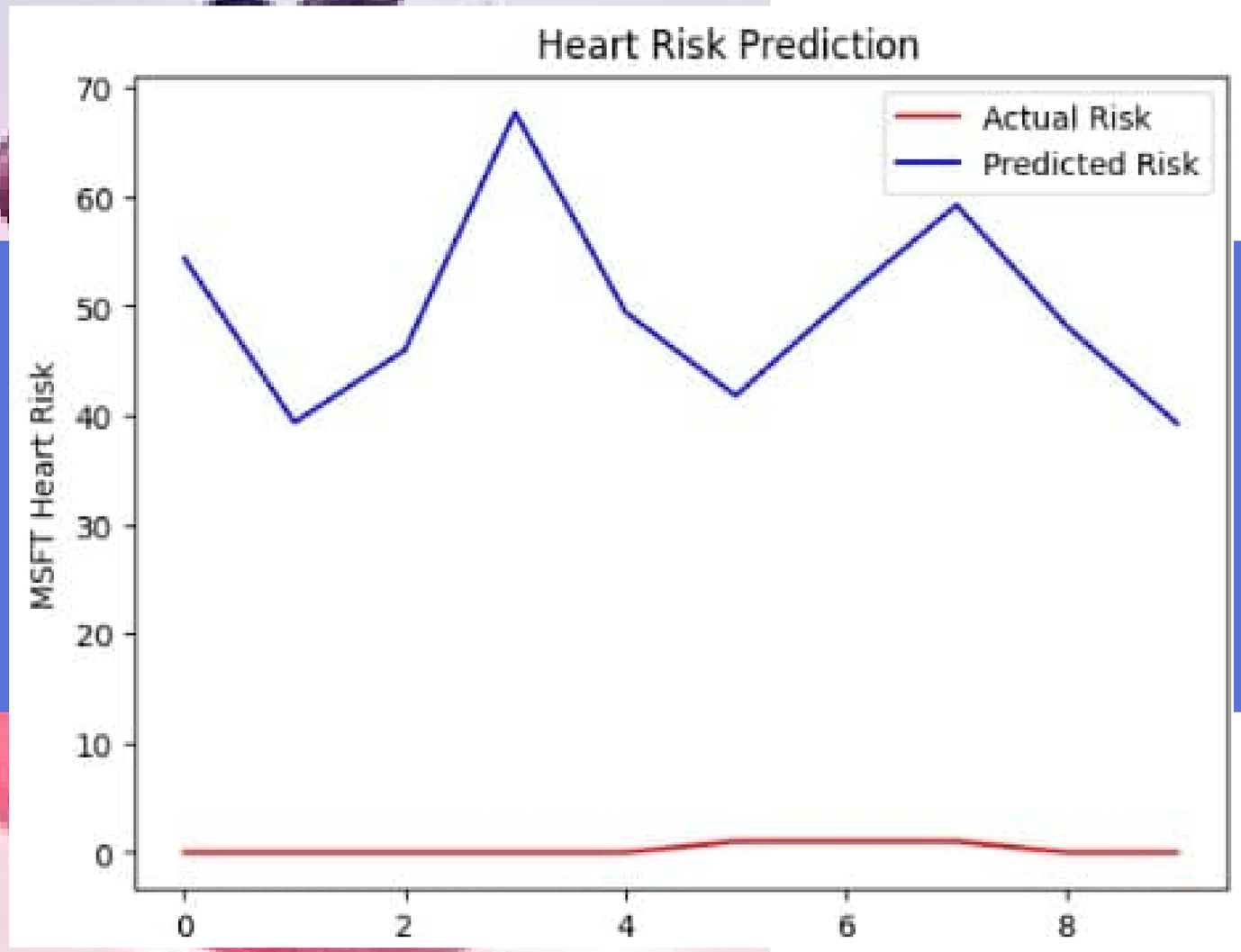
ANFIS GA



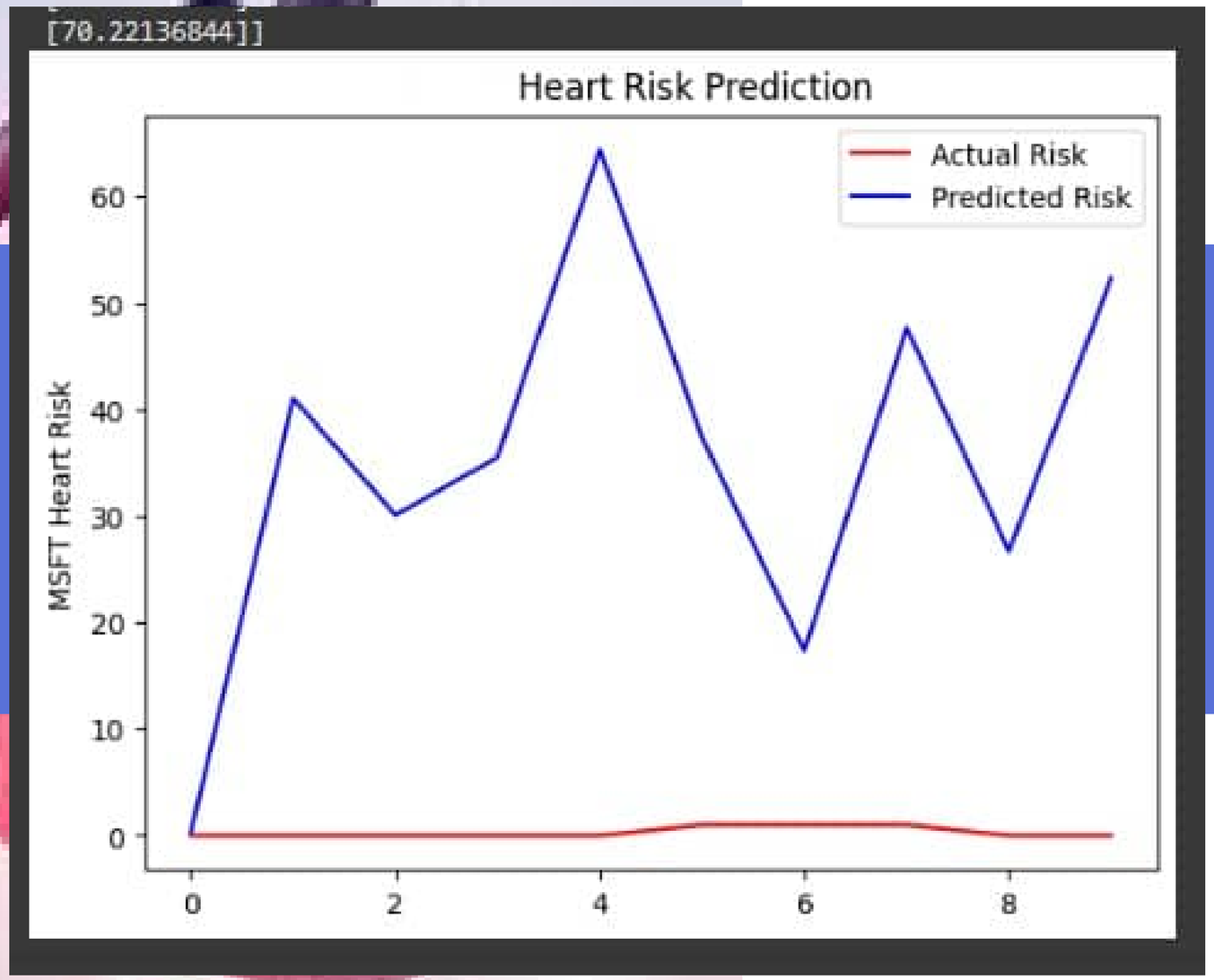
```
#from ANFIS import EVOLUTIONARY_ANFIS

E_Anfis = EVOLUTIONARY_ANFIS(functions=3,generations=300,offsprings=10,
                             mutationRate=0.2,learningRate=0.2,chance=0.7,ruleComb="simple")

bestParam, bestModel = E_Anfis.fit(X_train,y_train,optimize_test_data=False)

bestParam, bestModel = E_Anfis.fit(X_train,y_train,X_test,y_test,optimize_test_data=True)
```



Mutation rate = 0.1
Learning rate = 0.01
Generation = 300



Mutation rate = 0.002
Learning rate = 0.001
Generation = 50



Mutation rate = 0.0002
Learning rate = 0.001
Generation = 200

Conclusion

Thank You

CLASSES 2024