Heart Disease Prediction using ANN

Difficulty Level : Medium • Last Updated : 11 May, 2020

Deep Learning is a technology of which mimics a human brain in the sense that it consists of multiple neurons with multiple layers like a human brain. The network so formed consists of an input layer, an output layer, and one or more hidden layers. The network tries to learn from the data that is fed into it and then performs predictions accordingly. The most basic type of neural network is the ANN (Artificial Neural Network). The ANN does not have any special structure, it just comprises of multiple neural layers to be used for prediction.
Let's build a model that predicts whether a person has heart disease or not by using ANN.

About the data:
In the dataset, we have *13* columns in which we are given different attributes such as sex, age, cholesterol level, etc. and we are given a target column which tells us whether that person has heart disease or not. We will keep all the columns as independent variables other than the target column because it will be our dependent variable. We will build an ANN which will predict whether a person has heart disease or not given other attributes of the person.

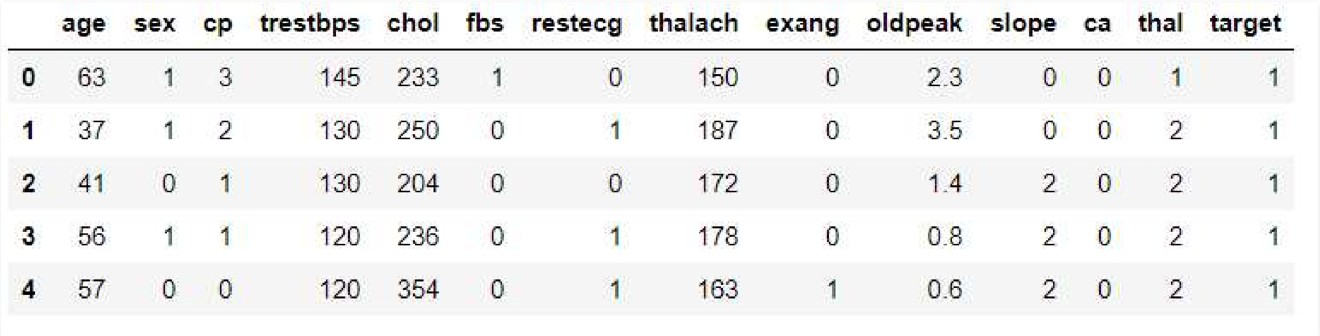
You can find the dataset here heart disease dataset

Code: Importing Libraries

```
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import confusion_matrix
```

Code: Importing Dataset

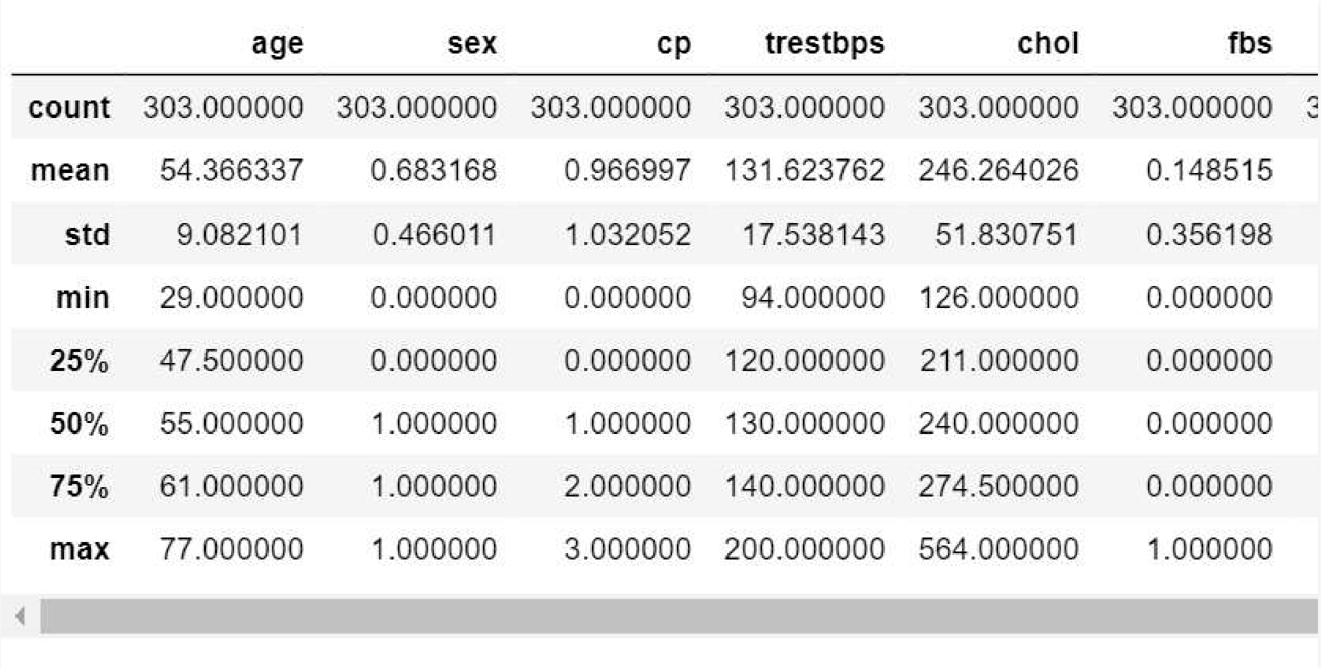
```
data = pd.read_csv('heart.csv')
data.head()
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Data Description:

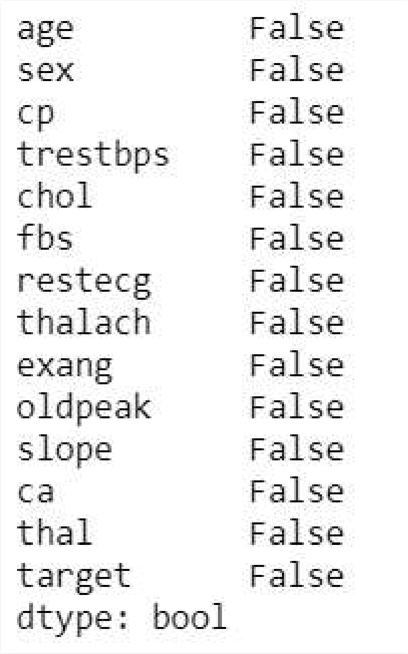
```
data.describe()
```



	age	sex	cp	trestbps	chol	fbs
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000

Code: Check for null values

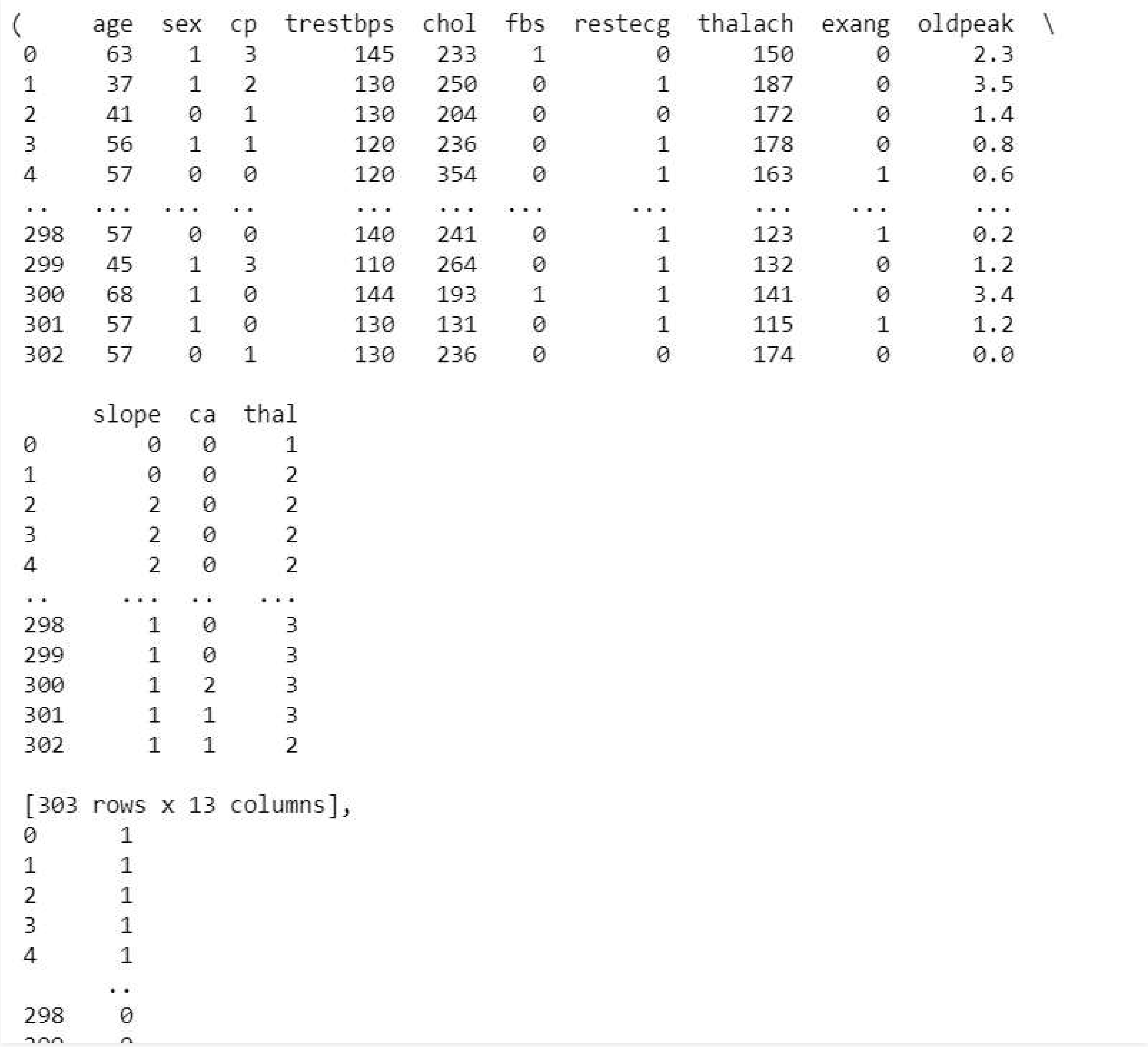
```
data.isnull().any()
```



```
age         False
sex         False
cp          False
trestbps    False
chol        False
fbs         False
restecg     False
thalach     False
exang       False
oldpeak     False
slope       False
ca          False
thal        False
target      False
dtype: bool
```

Assign Dependent and Independent variable

```
X = data.iloc[:,:13].values
y = data["target"].values
```



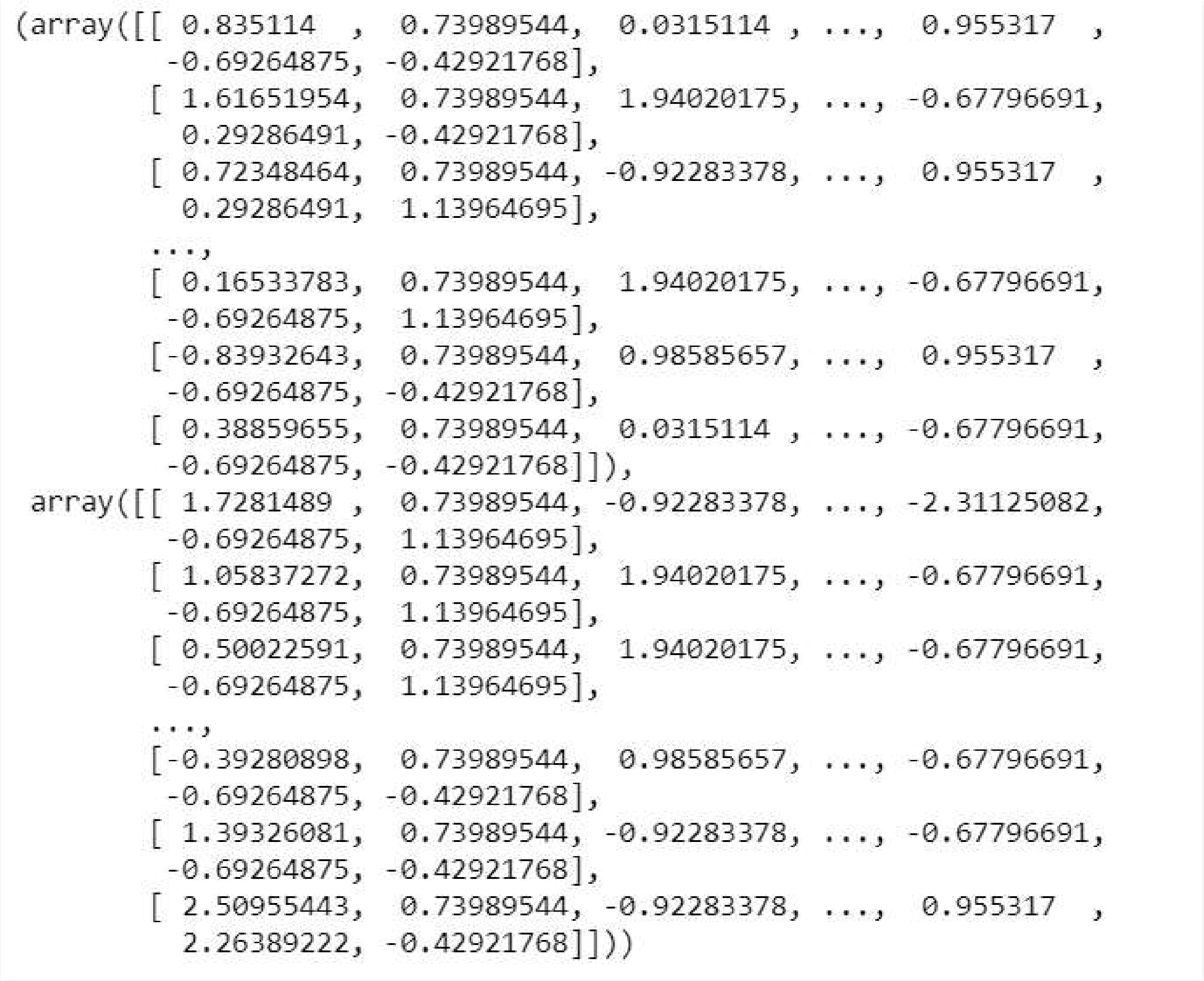
```
(     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
 0     63    1   3       145   233    1        0      150      0      2.3
 1     37    1   2       130   250    0        1      187      0      3.5
 2     41    0   1       130   204    0        0      172      0      1.4
 3     56    1   1       120   236    0        1      178      0      0.8
 4     57    0   0       120   354    0        1      163      1      0.6
 ..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
 298   57    0   0       140   241    0        1      123      1      0.2
 299   45    1   3       110   264    0        1      132      0      1.2
 300   68    1   0       144   193    1        1      141      0      3.4
 301   57    1   0       130   131    0        1      115      1      1.2
 302   57    0   1       130   236    0        0      174      0      0.0

      slope  ca  thal
 0        0   0     1
 1        0   0     2
 2        2   0     2
 3        2   0     2
 4        2   0     2
 ..     ...  ..   ...
 298      1   0     3
 299      1   0     3
 300      1   2     3
 301      1   1     3
 302      1   1     2

 [303 rows x 13 columns],
 0      1
 1      1
 2      1
 3      1
 4      1
       ..
 298    0
 299    0
```

Code : Split data into Train and Test dataset

```
X_train,X_test,y_train, y_test = train_test_split(X,y,test_size = 0.3 , random_s
```

Code: Scale the data.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```



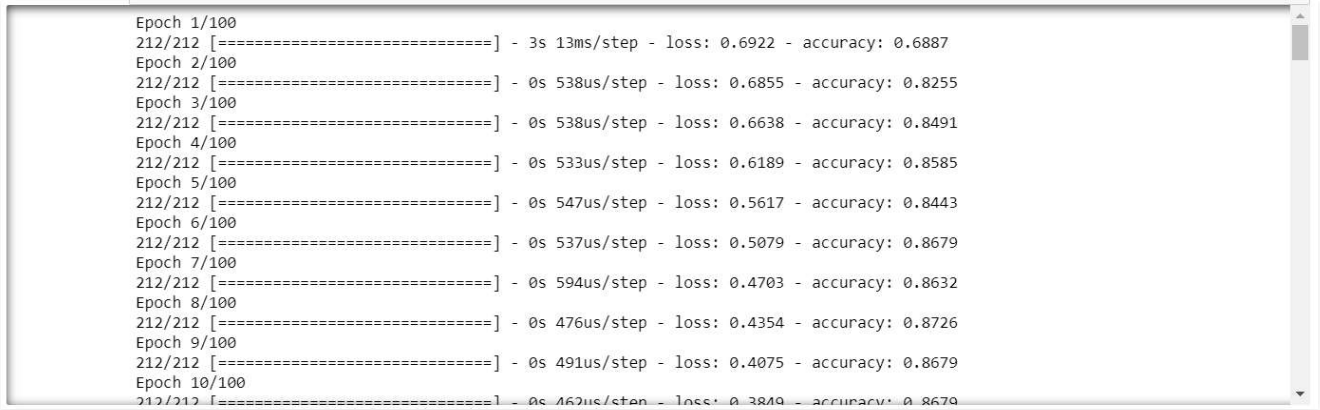
```
(array([[ 0.835114  ,  0.73989544,  0.0315114 , ...,  0.955317  ,
         -0.69264875, -0.42921768],
        [ 1.61651954,  0.73989544,  1.94020175, ..., -0.67796691,
          0.29286491, -0.42921768],
        [ 0.72348464,  0.73989544, -0.92283378, ...,  0.955317  ,
          0.29286491,  1.13964695],
        ...,
        [ 0.16533783,  0.73989544,  1.94020175, ..., -0.67796691,
         -0.69264875,  1.13964695],
        [-0.83932643,  0.73989544,  0.98585657, ...,  0.955317  ,
         -0.69264875, -0.42921768],
        [ 0.38859655,  0.73989544,  0.0315114 , ..., -0.67796691,
         -0.69264875, -0.42921768]]),
 array([[ 1.7281489 ,  0.73989544, -0.92283378, ..., -2.31125082,
         -0.69264875,  1.13964695],
        [ 1.05837272,  0.73989544,  1.94020175, ..., -0.67796691,
         -0.69264875,  1.13964695],
        [ 0.50022591,  0.73989544,  1.94020175, ..., -0.67796691,
         -0.69264875,  1.13964695],
        ...,
        [-0.39280898,  0.73989544,  0.98585657, ..., -0.67796691,
         -0.69264875, -0.42921768],
        [ 1.39326081,  0.73989544, -0.92283378, ..., -0.67796691,
         -0.69264875, -0.42921768],
        [ 2.50955443,  0.73989544, -0.92283378, ...,  0.955317  ,
          2.26389222, -0.42921768]]))
```

Code: Building the Model

```
classifier = Sequential()
classifier.add(Dense(activation = "relu", input_dim = 13,
                     units = 8, kernel_initializer = "uniform"))
classifier.add(Dense(activation = "relu", units = 14,
                     kernel_initializer = "uniform"))
classifier.add(Dense(activation = "sigmoid", units = 1,
                     kernel_initializer = "uniform"))
classifier.compile(optimizer = 'adam' , loss = 'binary_crossentropy',
                   metrics = ['accuracy'] )
```

Code : Fitting the Model

```
classifier.fit(X_train , y_train , batch_size = 8 ,epochs = 100   )
```



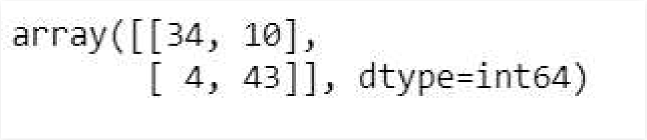
```
Epoch 1/100
212/212 [==============================] - 3s 13ms/step - loss: 0.6922 - accuracy: 0.6887
Epoch 2/100
212/212 [==============================] - 0s 538us/step - loss: 0.6855 - accuracy: 0.8255
Epoch 3/100
212/212 [==============================] - 0s 538us/step - loss: 0.6638 - accuracy: 0.8491
Epoch 4/100
212/212 [==============================] - 0s 533us/step - loss: 0.6189 - accuracy: 0.8585
Epoch 5/100
212/212 [==============================] - 0s 547us/step - loss: 0.5617 - accuracy: 0.8443
Epoch 6/100
212/212 [==============================] - 0s 537us/step - loss: 0.5079 - accuracy: 0.8679
Epoch 7/100
212/212 [==============================] - 0s 594us/step - loss: 0.4703 - accuracy: 0.8632
Epoch 8/100
212/212 [==============================] - 0s 476us/step - loss: 0.4354 - accuracy: 0.8726
Epoch 9/100
212/212 [==============================] - 0s 491us/step - loss: 0.4075 - accuracy: 0.8679
Epoch 10/100
212/212 [==============================] - 0s 462us/step - loss: 0.3849 - accuracy: 0.8679
```

Code : Performing prediction and rescaling

```
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
```

Code: Confusion Matrix

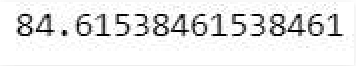
```
cm = confusion_matrix(y_test,y_pred)
cm
```



```
array([[34, 10],
       [ 4, 43]], dtype=int64)
```

Code: Accuracy

```
accuracy = (cm[0][0]+cm[1][1])/(cm[0][1] + cm[1][0] +cm[0][0] +cm[1][1])
print(accuracy*100)
```



```
84.61538461538461
```

We will get accuracy approximately around 85%.

♡ Like 0

Next >I

ML | Heart Disease Prediction Using Logistic Regression .

RECOMMENDED ARTICLES

Page : 1 2 3

01 ML | Heart Disease Prediction Using Logistic Regression .
23, Mar 20

02 Draw Heart Using Turtle Graphics in Python
30, Jun 20

03 Python program to print the Inverted heart pattern
22, Sep 20

04 ML | Rainfall prediction using Linear regression
12, Jun 19

05 Prediction of Wine type using Deep Learning
09, Apr 19

06 Word Prediction using concepts of N - grams and CDF
17, Jun 19

07 Link Prediction - Predict edges in a network using Networkx
25, Apr 20

08 Scrapping Weather prediction Data using Python and BS4
02, Jun 20

Article Contributed By :

KaranGupta5
@KaranGupta5

Vote for difficulty

Current difficulty : Medium

Easy Normal Medium Hard Expert

Article Tags : Deep-Learning , Neural Network , Machine Learning , Python

Practice Tags : Machine Learning

Improve Article Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company

About Us

Careers

Privacy Policy

Contact Us

Copyright Policy

Learn

Algorithms

Data Structures

Languages

CS Subjects

Video Tutorials

Practice

Courses

Contribute

Write an Article

Company-wise
Write Interview Experience
Topic-wise
Internships
How to begin?
Videos