Project2-Heart disease prediction

April 8, 2025

1 Project: Heart Disease Prediction Model using Logistic Regression

Objective of Project: To predict the presence of heart disease in patients using clinical data.

Dataset: The classic Heart Disease dataset (Cleveland dataset from UCI).

[3]:
```
# import libraries
import pandas as pd
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

[5]:
```
# load the dataset
df = pd.read_csv('heart.csv')
df.head()
```

[5]:
```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   63    1   3       145   233    1        0      150      0      2.3      0
1   37    1   2       130   250    0        1      187      0      3.5      0
2   41    0   1       130   204    0        0      172      0      1.4      2
3   56    1   1       120   236    0        1      178      0      0.8      2
4   57    0   0       120   354    0        1      163      1      0.6      2

   ca  thal  target
0   0     1       1
1   0     2       1
2   0     2       1
3   0     2       1
4   0     2       1
```

[9]:
```
df.shape
```

[9]: (303, 14)

1.1 Data Dictionary (from domain experties)

age: age in years

sex: sex

1 = male

0 = female

cp: chest pain type

Value 0: typical angina

Value 1: atypical angina

Value 2: non-anginal pain

Value 3: asymptomatic

trestbps: resting blood pressure (in mm Hg on admission to the hospital)

chol: serum cholestoral in mg/dl

fbs: (fasting blood sugar > 120 mg/dl)

1 = true;

0 = false

restecg: resting electrocardiographic results

Value 0: normal

Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)

Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria

thalach: maximum heart rate achieved

exang: exercise induced angina

1 = yes

0 = no

oldpeak = ST depression induced by exercise relative to rest

slope: the slope of the peak exercise ST segment

Value 0: upsloping

Value 1: flat

Value 2: downsloping

ca: number of major vessels (0-3) colored by flourosopy

thal:

0 = error (in the original dataset 0 maps to NaN's)

1 = fixed defect

2 = normal

3 = reversable defect

target (the l

Note on the target label:

Diagnosis of heart disease (angiographic disease status) Value 0: < 50Value 1: > 50able):

0 = no disease,

1 = disease

[13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

[15]:
```
df.describe()
```

[15]:
```
              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

          restecg     thalach       exang     oldpeak       slope          ca  \
```

```
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean     0.528053  149.646865    0.326733    1.039604    1.399340    0.729373
std      0.525860   22.905161    0.469794    1.161075    0.616226    1.022606
min      0.000000   71.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000  133.500000    0.000000    0.000000    1.000000    0.000000
50%      1.000000  153.000000    0.000000    0.800000    1.000000    0.000000
75%      1.000000  166.000000    1.000000    1.600000    2.000000    1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000    4.000000

             thal      target
count  303.000000  303.000000
mean     2.313531    0.544554
std      0.612277    0.498835
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

1.2 Data pre-processing

[24]:
```python
# check the missing values

df.isnull().sum()
```

[24]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

[28]:
```python
# split the data in independent and dependent variables
x= df.drop('target', axis = 1)  # independent feature
y = df['target']                # Dependent feature (target variable)
```

[44]:
```python
# train test split
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2,␣
↪random_state=42)
```

[46]:
```
x_train
```

[46]:
```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
132   42    1   1       120   295    0        1      162      0      0.0
202   58    1   0       150   270    0        0      111      1      0.8
196   46    1   2       150   231    0        1      147      0      3.6
75    55    0   1       135   250    0        0      161      0      1.4
176   60    1   0       117   230    1        1      160      1      1.4
..   …  …  ..       …   …  …      …      …      …      …
188   50    1   2       140   233    0        1      163      0      0.6
71    51    1   2        94   227    0        1      154      1      0.0
106   69    1   3       160   234    1        0      131      0      0.1
270   46    1   0       120   249    0        0      144      0      0.8
102   63    0   1       140   195    0        1      179      0      0.0

     slope  ca  thal
132      2   0     2
202      2   0     3
196      1   0     2
75       1   0     2
176      2   2     3
..     …  ..   …
188      1   1     3
71       2   1     3
106      1   1     2
270      2   0     3
102      2   2     2

[242 rows x 13 columns]
```

[48]:
```
y_train
```

[48]:
```
132    1
202    0
196    0
75     1
176    0
      ..
188    0
71     1
106    1
270    0
102    1
Name: target, Length: 242, dtype: int64
```

[50]:
```
from sklearn.linear_model import LogisticRegression
```

[52]:
```
logreg = LogisticRegression()

logreg.fit(x_train, y_train)
```

[52]:
```
LogisticRegression()
```

[54]:
```
# Make the predictions
y_pred = logreg.predict(x_test)
```

[56]:
```
y_pred
```

[56]:
```
array([0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0], dtype=int64)
```

[58]:
```
y_pred_prob = logreg.predict_proba(x_test)[:, 1]
```

[60]:
```
y_pred_prob
```

[60]:
```
array([1.17702855e-01, 7.89568007e-01, 8.11578819e-01, 2.59305607e-02,
       9.30504069e-01, 9.05120923e-01, 6.06542584e-01, 1.14143705e-03,
       4.83333573e-03, 5.47484409e-01, 7.89531849e-01, 7.11814629e-02,
       9.25950579e-01, 2.19235026e-02, 9.82205572e-01, 9.52954802e-01,
       9.77178756e-01, 5.04351348e-02, 6.34987408e-03, 1.02764744e-02,
       7.38387361e-01, 1.05222577e-02, 1.37311576e-01, 8.00450586e-01,
       9.19348268e-01, 6.67266260e-01, 9.08248067e-01, 6.86177887e-01,
       6.56726108e-03, 9.10155769e-01, 4.21340094e-02, 2.93431981e-02,
       5.62116979e-03, 7.24593241e-02, 6.90302723e-01, 6.84994624e-02,
       6.38464062e-01, 8.73556106e-01, 8.09302006e-01, 8.53214445e-01,
       5.17951451e-01, 8.35213292e-01, 8.15581972e-01, 6.86176207e-01,
       8.41197991e-01, 5.46301117e-03, 8.01275776e-01, 9.52859388e-01,
       7.11805039e-02, 3.04629283e-02, 6.34226202e-02, 1.25017980e-02,
       8.70138566e-01, 9.77604269e-01, 2.22735234e-01, 5.80512878e-04,
       4.32857695e-02, 9.63002856e-01, 1.00577902e-02, 3.81726863e-03,
       2.93434520e-02])
```

[62]:
```
# Evalauation of the logistic regression

from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report
```

[64]:
```
print('Accuracy score is:', accuracy_score(y_test, y_pred))
```

```
Accuracy score is: 0.8852459016393442
```

[66]:
```
print('Confusion Matrix')
print(confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix
[[25  4]
 [ 3 29]]
```

[68]:
```
print('classification Report')
print(classification_report(y_test, y_pred))
```

```
classification Report
              precision    recall  f1-score   support

           0       0.89      0.86      0.88        29
           1       0.88      0.91      0.89        32

    accuracy                           0.89        61
   macro avg       0.89      0.88      0.88        61
weighted avg       0.89      0.89      0.89        61
```

[72]:
```
import joblib

# Save your trained model ( Logistic Regression)
joblib.dump(logreg, "heart_disease_model.pkl")
```

[72]: ['heart_disease_model.pkl']

This project successfully demonstrates the application of machine learning techniques to predict the likelihood of heart disease based on patient attributes. Among the models tested, [logreg] performed the best with an accuracy of 88%. This indicates that the model can assist healthcare professionals in early screening of high-risk individuals, potentially improving patient outcomes through timely interventions.

2 Thank You!!

[ ]:
```
```