Heart Disease Prediction

April 20, 2025

1 Heart Disease Prediction System

[1]:
```
# Importing Dependencies
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
import warnings
warnings.filterwarnings('ignore')
```

Data Collection and Pre-Processing

[2]:
```
# Loading the csv data to a pandas dataframe
heart_data = pd.read_csv('dataset/heart_disease_data.csv')
```

[3]:
```
# printing the first 5 rows
heart_data.head()
```

[3]:

	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

[4]:
```
# checking for any null values
heart_data.isna().sum()
```

[4]:
```
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
```

[5]:
```
# printing the last 5 rows of the dataset
heart_data.tail()
```

[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

[6]:
```
# Checking the toal no of rows and cols in dataset
heart_data.shape
```

[6]:
```
(303, 14)
```

[7]:
```
# Getting some info about the data
heart_data.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
```

Statistical measures of the data

[8]:
```
heart_data.describe()
```

[8]:

	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

[9]:
```
# Total target values
heart_data['target'].value_counts()
```

[9]:
```
target
1    165
0    138
Name: count, dtype: int64
```

Here : 1 represents that the person has a heart diesease 0 represents that the person has no heart diseases

[10]:
```
# Splitting the features and targets
X = heart_data.drop(columns='target',axis=1)
y = heart_data['target']
```

[11]:
```
# Checking the values inside the variables
print(X)
print(y)
```

```
     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
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

Splitting the data into train and test data

[12]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
↪2,stratify=y,random_state=2)
```

[13]:
```
# Checking the splitted data
print(X.shape,X_train.shape,X_test.shape)
```

```
(303, 13) (242, 13) (61, 13)
```

Training the model

[14]:
```
model = LogisticRegression()
```

[15]:
```
model.fit(X_train,y_train)
```

[15]:
```
LogisticRegression()
```

As its a small dataset its giving this output

Evaluating the model

[16]:
```
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction,y_train)
print("The accuracy on train data is :",training_data_accuracy)
```

```
The accuracy on train data is : 0.8512396694214877
```

[17]:
```
X_test_prediction = model.predict(X_test)
testing_data_accuracy = accuracy_score(X_test_prediction,y_test)
print("The accuracy on test data is :",testing_data_accuracy)
```

```
The accuracy on test data is : 0.819672131147541
```

Building a predicting system

[18]:
```
# Getting the input values
input_data = (41,0,1,130,204,0,0,172,0,1.4,2,0,2)

# change the input data to a np arr
input_data_as_numpy_array = np.asarray(input_data)
```

```
# reshaping the array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# predicting the answer
prediction = model.predict(input_data_reshaped)

# printing the output
print(prediction)

if (prediction[0]==1):
    print("The patient is having a heart disease")
else:
    print("The patient is not having a heart disease")
```

```
[1]
The patient is having a heart disease
```