heartdisease

September 2, 2023

[1]:
```
import pandas as pd
import numpy as np
import seaborn  as sns
from matplotlib import pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score as ac
import pickle
```

[ ]:
```

```

[2]:
```
df = pd.read_csv("heart_disease_data.csv")
```

About this file Age Sex : male : 1 female : 0

chest pain type – Value 1: typical angina – Value 2: atypical angina – Value 3: non-anginal pain – Value 4: asymptomatic

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

serum cholestoral in mg/dl

(fasting blood sugar $>$ 120 mg/dl) (1 = true; 0 = false)

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

exercise induced angina (1 = yes; 0 = no)

Angina is chest pain or discomfort caused when your heart muscle doesn't get enough oxygen-rich blood. It may feel like pressure or squeezing in your chest.

oldpeak = ST depression induced by exercise relative to rest

slope: the slope of the peak exercise ST segment

–Value 1: upsloping – Value 2: flat – Value 3: downsloping

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

A blood disorder called thalassemia (3 = normal; 6 = fixed defect; 7 = reversable defect)

Target : 0 No Heart disease 1 Heart disease

[3]:
```
df
```

[3]:
```
     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  target
0        0   0     1       1
1        0   0     2       1
2        2   0     2       1
3        2   0     2       1
4        2   0     2       1
..     ...  ..   ...     ...
298      1   0     3       0
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0

[303 rows x 14 columns]
```

[4]:
```
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
```

no missing values

[5]:
```
df.describe()
```

[5]:

	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

[6]:
```
df.groupby('sex').mean().target
```

[6]:
```
sex
0    0.750000
```

```
1    0.449275
Name: target, dtype: float64
```

women are most likely to come down with heart diseases

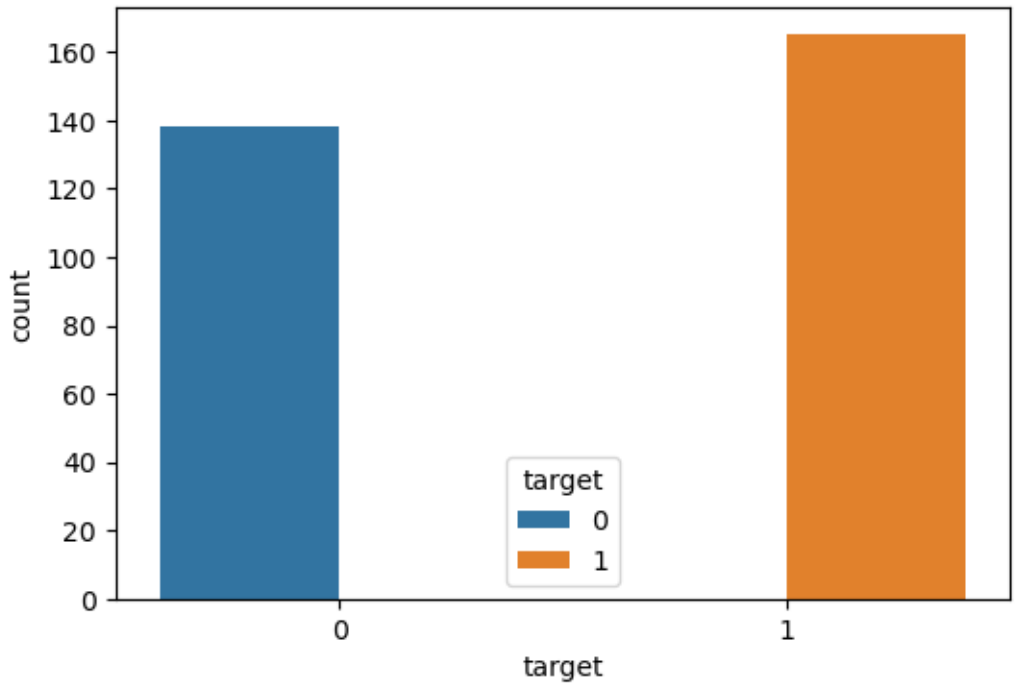
[7]:
```
df.shape
```

[7]:
```
(303, 14)
```

[8]:
```
df.target.value_counts()
```

[8]:
```
1    165
0    138
Name: target, dtype: int64
```

[9]:
```
plt.figure(figsize=(6,4))
sns.countplot(x= 'target',hue = "target" , data=df)
plt.show()
```



[10]:
```
str_columns=df.columns
X = str_columns[:-1]
y = str_columns [-1]
X_train = df[X]
y_train = df[y]
```

[11]:
```
X_train
```

[11]:
```
     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]
```

[12]:
```
X_train,X_val = train_test_split(X_train,test_size=0.2,random_state = 3)
X_train,X_test = train_test_split(X_train,test_size=0.2,random_state = 3)
y_train,y_val = train_test_split(y_train,test_size=0.2,random_state = 3)
y_train,y_test = train_test_split(y_train,test_size=0.2,random_state = 3)
```

[13]:
```
X_train.shape
```

[13]:
```
(193, 13)
```

[14]:
```
print(X_val.shape,y_val)
```

```
(61, 13) 245    0
162    1
10     1
161    1
73     1
      ..
```

```
102    1
83     1
259    0
121    1
61     1
Name: target, Length: 61, dtype: int64
```

Model Training

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

```
C:\Users\PC\conda\lib\site-packages\sklearn\linear_model\_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

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

[16]:
```
X_pred = model.predict(X_train)
X_pred_test1 = model.predict(X_test)
```

[17]:
```
ac_measuring=ac(X_pred,y_train)
ac_measuring_test1=ac(X_pred_test1,y_test)
```

[18]:
```
print("The accuracy of the training :",ac_measuring)

print("The accuracy of the testing :",ac_measuring_test1)
```

```
The accuracy of the training : 0.8549222797927462
The accuracy of the testing : 0.8163265306122449
```

[19]:
```
model_2 = LogisticRegression()
model_2.fit(X_val,y_val )
```

```
C:\Users\PC\conda\lib\site-packages\sklearn\linear_model\_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
```

```
regression
  n_iter_i = _check_optimize_result(
```

[19]:
```
LogisticRegression()
```

[20]:
```
X_pred_val = model_2.predict(X_val)
ac_measuring_val=ac(X_pred_val,y_val)
ac_measuring_val
```

[20]:
```
0.9016393442622951
```

[21]:
```
X_pred_test = model_2.predict(X_test)
ac_measuring_test2 = ac(X_pred_test,y_test)
ac_measuring_test2
```

[21]:
```
0.8775510204081632
```

the second model is better than the first one so we'll use it

[22]:
```
def pred_system(input_array):
   input_array = np.asarray(input_data)
   reshaped_data = input_array.reshape(1,-1)
   input_pred = model_2.predict(reshaped_data)
   if input_pred == [1]:
      print ("this person has a heart disease ")
   else:
      print ("this person does not have a heart disease ")
```

[25]:
```
input_data =(44,1,0,110,197,0,0,177,0,0,2,1,2)
#Turn the data into an numpy_array
pred_system(input_data)
```

```
this person does not have a heart disease

C:\Users\PC\conda\lib\site-packages\sklearn\base.py:420: UserWarning: X does not
have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```

[ ]:

[ ]:

[ ]: