Heart_disease

May 7, 2023

1 Introduction:

*This Heart disease set given whole information about patient.

Aim: The aim of this project is to make

1)data data cleansing

2)data preparing

3)data understanding

and aim to find patient have disease or not

2 Data Preprocessing :

2.0.1 1) In dataset consists of two steps, i.e., Data Collection and Data Pre-processing. Data can be referred to as the raw material.

2.0.2 EDA OF Heart Disease Data

3 Import Libraries:

Here , Various predefined python libraries were used for data pre- processing. Some of the libraries used are Numpy, Pandas, Seaborn, matplotlib

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
data=pd.read_csv("heartdisease.csv")  # here we  read the dataset using pandas
```

[4]:
```
data
```

[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	\
0	63	Male	3	145.0	233	1	0	150	0	2.3	
1	37	Male	2	130.0	250	0	1	187	0	3.5	
2	41	Female	1	130.0	204	0	0	172	0	1.4	
3	56	Male	1	120.0	236	0	1	178	0	0.8	
4	57	Female	0	120.0	354	0	1	163	1	0.6	

```
..   ..     … ..       …   …  …      …      …    …      …
298  57  Female  0     140.0   241    0        1      123      1      0.2
299  45    Male  3     110.0   264    0        1      132      0      1.2
300  68    Male  0     144.0   193    1        1      141      0      3.4
301  57    Male  0     130.0   131    0        1      115      1      1.2
302  57  Female  1     130.0   236    0        0      174      0      0.0

     slope   ca  thal  target Unnamed: 14
0        0  0.0     1     1.0        Male
1        0  0.0     2     1.0        Male
2        2  0.0     2     1.0      Female
3        2  0.0     2     1.0        Male
4        2  0.0     2     1.0      Female
..     …  …   …     …     …          …
298      1  0.0     3     0.0      Female
299      1  0.0     3     0.0        Male
300      1  2.0     3     0.0        Male
301      1  1.0     3     0.0        Male
302      1  1.0     2     0.0      Female

[303 rows x 15 columns]
```

4 The following attributes describe each of the datasets features used by the model:

1)AGE-age in years

2)sex =(0:female,1:male)

3)cp =chest pain type

0: Typical angina: chest pain related decrease blood supply to the heart. 1: Atypical angina: chest pain not related to heart 2: Non-anginal pain: typically esophageal spasms (non heart related) 3: Asymptomatic: chest pain not showing signs of disease

4)trestbps= resting of blood pressure(anything above 130-140 is a typical cause of concern

5)chol =serum cholestoral in mg/dl serum = LDL + HDL + .2 * triglycerides above 200 is cause for concern

6)fbs = fasting blood sugar (1=true and 0=false), anything greater than 126 signals diabatics

7)restecg - resting electrocardiographic results

0: Nothing to note 1: ST-T Wave abnormality can range from mild symptoms to severe problems signals non-normal heart beat 2: Possible or definite left ventricular hypertrophy Enlarged heart's main pumping chamber

8)thalach - maximum heart rate achieved

9)exang - exercise induced angina (1 = yes; 0 = no)

10)oldpeak - ST depression induced by exercise relative to rest looks at stress of heart during excercise unhealthy heart will stress more

11)slope - the slope of the peak exercise ST segment

0: Upsloping: better heart rate with excercise (uncommon) 1: Flatsloping: minimal change (typical healthy heart) 2: Downslopins: signs of unhealthy heart

12)ca - number of major vessels (0-3) colored by flourosopy colored vessel means the doctor can see the blood passing through

the more blood movement the better (no clots)

13)thal - thalium stress result

1,3: normal 6: fixed defect: used to be defect but ok now

7: reversable defect: no proper blood movement when excercising

14)target - have disease or not (1=yes, 0=no)

[5]:
```
data.shape   # here we know that how many rows and how many colums in dataset␣
↪using "shape"
```

[5]:
```
(303, 15)
```

[6]:
```
data.head()   # here We shows 1st five rows using head() function
```

[6]:
```
   age     sex cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0   63    Male  3     145.0   233    1        0      150      0      2.3
1   37    Male  2     130.0   250    0        1      187      0      3.5
2   41  Female  1     130.0   204    0        0      172      0      1.4
3   56    Male  1     120.0   236    0        1      178      0      0.8
4   57  Female  0     120.0   354    0        1      163      1      0.6

   slope   ca  thal  target Unnamed: 14
0      0  0.0     1     1.0        Male
1      0  0.0     2     1.0        Male
2      2  0.0     2     1.0      Female
3      2  0.0     2     1.0        Male
4      2  0.0     2     1.0      Female
```

[7]:
```
data.tail()  # here We shows last five rows using tail() function
```

[7]:
```
     age     sex cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
298   57  Female  0     140.0   241    0        1      123      1      0.2
299   45    Male  3     110.0   264    0        1      132      0      1.2
300   68    Male  0     144.0   193    1        1      141      0      3.4
301   57    Male  0     130.0   131    0        1      115      1      1.2
302   57  Female  1     130.0   236    0        0      174      0      0.0

     slope   ca  thal  target Unnamed: 14
```

```
298      1  0.0     3     0.0      Female
299      1  0.0     3     0.0        Male
300      1  2.0     3     0.0        Male
301      1  1.0     3     0.0        Male
302      1  1.0     2     0.0      Female
```

[8]:
```
data.sample(5)  #EDA
```

[8]:
```
     age     sex cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
49    53  Female  0     138.0   234    0        0      160      0      0.0
208   49    Male  2     120.0   188    0        1      139      0      2.0
105   68  Female  2     120.0   211    0        0      115      0      1.5
158   58    Male  1     125.0   220    0        1      144      0      0.4
83    52    Male  3     152.0   298    1        1      178      0      1.2

     slope   ca  thal  target Unnamed: 14
49       2  0.0     2     1.0      Female
208      1  NaN     3     0.0        Male
105      1  0.0     2     1.0      Female
158      1  4.0     3     1.0        Male
83       1  NaN     3     1.0        Male
```

[9]:
```
data.dtypes  # here Check d the data type of each columns
```

[9]:
```
age             object
sex             object
cp              object
trestbps       float64
chol             int64
fbs              int64
restecg          int64
thalach          int64
exang            int64
oldpeak        float64
slope            int64
ca             float64
thal             int64
target         float64
Unnamed: 14     object
dtype: object
```

[10]:
```
data.info()   # we get columns name,not null count and data types using info
```

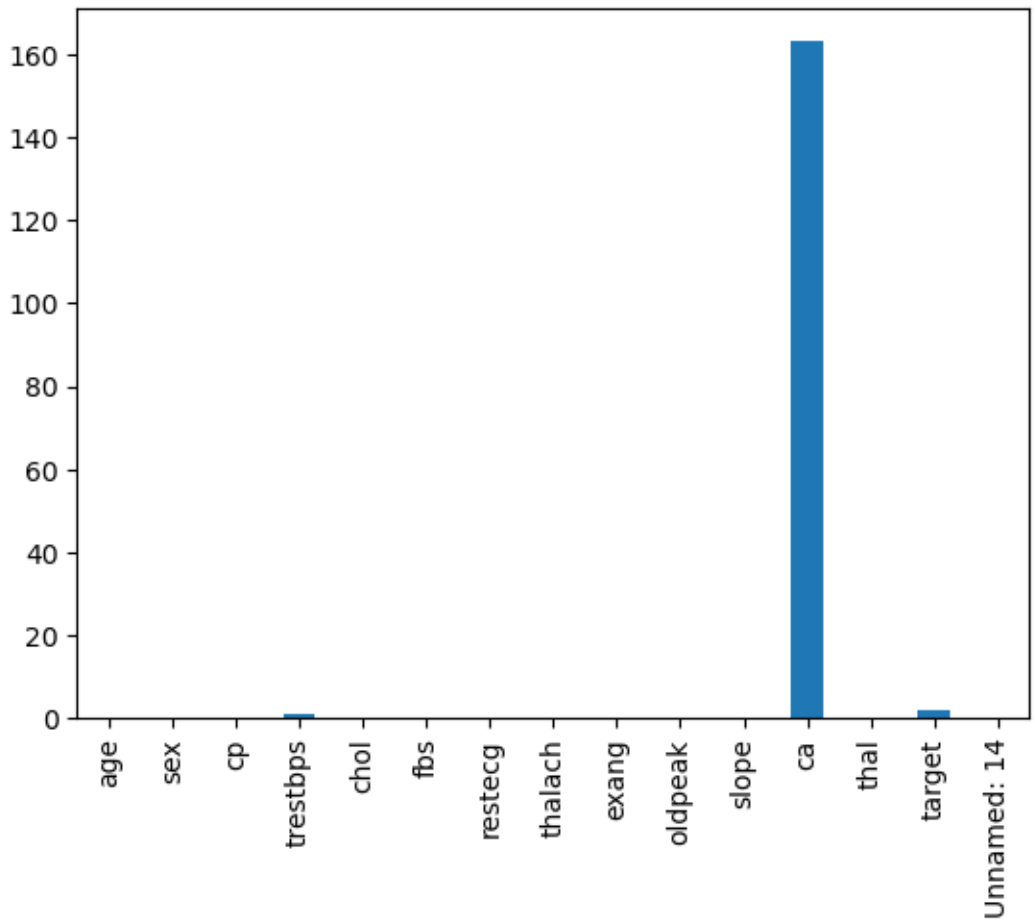
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
```

```
 0   age          303 non-null    object
 1   sex          303 non-null    object
 2   cp           303 non-null    object
 3   trestbps     302 non-null    float64
 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           140 non-null    float64
 12  thal         303 non-null    int64
 13  target       301 non-null    float64
 14  Unnamed: 14  303 non-null    object
dtypes: float64(4), int64(7), object(4)
memory usage: 35.6+ KB
```

[10]:
```
# we see that  datatypes for and cp  not look fare
```

[11]:
```
data.isnull().sum().plot(kind="bar")
```

[11]: <AxesSubplot:>



[12]:
```
# for the above plot  we can understandthat "ca"  features has missing values␣
↪is most
```

[13]:
```
# so we can drop that variable
```

[11]:
```
data.describe()    # here we use describe ()use to find count,mean,std␣
↪deviation,min,max valus,
                   # 25th percentile,50th percentile,75th percentile
```

[11]:

	trestbps	chol	fbs	restecg	thalach	exang	\
count	302.000000	303.000000	303.000000	303.000000	303.000000	303.000000	
mean	131.556291	246.264026	0.148515	0.528053	149.646865	0.326733	
std	17.527818	51.830751	0.356198	0.525860	22.905161	0.469794	
min	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	
25%	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	
50%	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	
75%	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	
max	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	

```
          oldpeak       slope          ca        thal      target
count  303.000000  303.000000  140.000000  303.000000  301.000000
mean     1.039604    1.399340    0.914286    2.313531    0.548173
std      1.161075    0.616226    1.121957    0.612277    0.498503
min      0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000    1.000000    0.000000    2.000000    0.000000
50%      0.800000    1.000000    0.000000    2.000000    1.000000
75%      1.600000    2.000000    2.000000    3.000000    1.000000
max      6.200000    2.000000    4.000000    3.000000    1.000000
```

[101]:
```
data.columns  # here using columns we know columns name
```

[101]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target', 'Unnamed: 14'],
      dtype='object')
```

[14]:
```
print(data['age'].unique()) # here  we found 40+ object value so
```

```
['63' '37' '41' '56' '57' '44' '52' '54' '48' '49' '64' '58' '50' '66'
 '43' '69' '59' '42' '61' '40' '71' '51' '0' '53' '65' '46' '45' '39' '47'
 '62' '34' '35' '29' '55' '60' '67' '68' '74' '76' '70' '38' '77' '40+']
```

[ ]:
```
data = data[data['age'] != '40+']
```

#age values containing '40+'. To delete rows with the value '40+', we use the drop() method with the condition of the value being not equal to '40+' as an argument. We store the result in the "data" variable to update the DataFrame.

#we use the != operator to check for inequality with '40+'. The resulting DataFrame df will have all rows except the ones with the value '40+'.

[ ]:
```
# use the lambda function to  chnage the age "40+" to 40
```

[16]:
```
data['age']=data['age'].apply(lambda x:"40"if x=="40+" else x)
```

[17]:
```
data[data['age']=="40+"]
```

[17]:
```
Empty DataFrame
Columns: [age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak,
slope, ca, thal, target, Unnamed: 14]
Index: []
```

[ ]:
```
# noe we can change datatype for "age"
```

[18]:
```
data['age']=data['age'].astype(int)
```

[22]:
```
data['age'].dtype
```

[22]:
```
dtype('int32')
```

[20]:
```
print(data['cp'].unique())  # here we found  a string
```

```
['3' '2' '1' '0' 'a']
```

[21]:
```
data=data[data['cp']!='a']
```

"cp" values containing string 'a'. To delete rows with the value 'a', we use the drop() method with the condition of the value being not equal to 'a'. We store the result in the "data" variable to update the DataFrame.

[23]:
```
data['cp'] = data['cp'].astype('int')  # convert data type into int
```

```
C:\Users\payal\AppData\Local\Temp\ipykernel_17984\2542486490.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['cp'] = data['cp'].astype('int')  # convert data type into int
```

[24]:
```
data['cp'].dtype
```

[24]:
```
dtype('int32')
```

[31]:
```
# encode the sex colums using replace function

data['sex'] = data['sex'].replace(['Female','Male'],[0,1])
```

[36]:
```
data['Unnamed: 14'] = data['Unnamed: 14'].replace(['Female','Male'],[0,1])
```

[33]:
```
data.head(5)
```

[33]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	\
0	63	1	3	145.0	233	1	0	150	0	2.3	0	
1	37	1	2	130.0	250	0	1	187	0	3.5	0	
2	41	0	1	130.0	204	0	0	172	0	1.4	2	
3	56	1	1	120.0	236	0	1	178	0	0.8	2	
4	57	0	0	120.0	354	0	1	163	1	0.6	2	

	thal	target	Unnamed: 14
0	1	1.0	Male
1	2	1.0	Male
2	2	1.0	Female
3	2	1.0	Male
4	2	1.0	Female

[ ]:
```
# ca columns lots of null values so drop the colums
```

[28]:
```
data=data.drop('ca',axis=1)
```

[29]:
```
data.head(3)
```

[29]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	thal	target	Unnamed: 14
0	63	Male	3	145.0	233	1	0	150	0	2.3	0	1	1.0	Male
1	37	Male	2	130.0	250	0	1	187	0	3.5	0	2	1.0	Male
2	41	Female	1	130.0	204	0	0	172	0	1.4	2	2	1.0	Female

[41]:
```
#Next ,I observed that sex and unnamed:14 columns are same
```

[34]:
```
print(data['sex'].unique())   #I have to check colimns contail null values, so␣
 ↪i get "male" and " female" two values are unique
```

```
[1 0]
```

[37]:
```
print(data['Unnamed: 14'].unique())

#I have to check colimns contail null values, so i get "male" and " female" two␣
 ↪values are unique
# both the colums same unique values
```

```
[1 0]
```

here,calculate the number of rows where the values in the two columns are different

in below code the "!=" operator is used to compare the values in the 'Sex' and 'Unnamed: 14' columns element-wise.

The sum() function is then used to count the number of rows where the values are different.

If the count is zero, it means that the two columns contain the same information.

[38]:
```
diff_count = sum(data['sex'] != data['Unnamed: 14'])

print(diff_count)
```

```
0
```

I got differnce 0

so I deleted one colum that is "Unnamed: 14"

[39]:
```
data.drop(['Unnamed: 14'], axis=1, inplace=True)  #using drop() delete colum
```

[42]:
```
data.head(5) # deleted unnamed 14
```

[42]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	thal	target
0	63	1	3	145.0	233	1	0	150	0	2.3	0	1	1.0
1	37	1	2	130.0	250	0	1	187	0	3.5	0	2	1.0
2	41	0	1	130.0	204	0	0	172	0	1.4	2	2	1.0
3	56	1	1	120.0	236	0	1	178	0	0.8	2	2	1.0
4	57	0	0	120.0	354	0	1	163	1	0.6	2	2	1.0

[30]:
```
data.isnull().sum()   # here i havve to check any null value pesent or not
↪using isnull() function. so we get null values here
```

[30]:
```
age           0
sex           0
cp            0
trestbps      1
chol          0
fbs           0
restecg       0
thalach       0
exang         0
oldpeak       0
slope         0
thal          0
target        2
Unnamed: 14   0
dtype: int64
```

how do you handle each null values

There are 2 columns have null values 1)trestbps 2) target

4.1 1)trestbsp:

1st i have to handle this null value ,i got only one null values so depending on the number of null values, you can choose to either drop the rows containing null values or impute the null values with appropriate values.

so , i got only one null value so i want to drop this null value using dropna()method

but if i don't want to drop then i use fillna()method ,null value replace either "mean" or "median"

as this way :=>data['trestbps'].fillna(data['trestbps'].median(), inplace=True)

[48]:
```
data.dropna(subset=['trestbps'],inplace=True)
```

[49]:
```
data.isnull().sum()
```

[49]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
thal        0
target      0
dtype: int64
```

4.2 2)target

target means have disease or not (1=yes, 0=no)

Impute null values with the mean or median: If the column with null values is important for the problem at hand, you can impute the null values with the mean or median of the column

I think this columns with null values is important that's why I was putting mean or median

[46]:
```
data['target'].fillna(data['target'].mean(), inplace=True)
```

[ ]:
```
# now check any null value present or not in data set
```

[51]:
```
data['trestbps'] = data['trestbps'].astype('int')  # convert data type into int
data['oldpeak'] = data['oldpeak'].astype('int')
data['target'] = data['target'].astype('int')
```

[52]:
```
data
```

[52]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	\
0	63	1	3	145	233	1	0	150	0	2	
1	37	1	2	130	250	0	1	187	0	3	
2	41	0	1	130	204	0	0	172	0	1	
3	56	1	1	120	236	0	1	178	0	0	
4	57	0	0	120	354	0	1	163	1	0	
..	…	…	..	…	…	…	…	…	…	…	
298	57	0	0	140	241	0	1	123	1	0	
299	45	1	3	110	264	0	1	132	0	1	
300	68	1	0	144	193	1	1	141	0	3	
301	57	1	0	130	131	0	1	115	1	1	

```
302   57    0   1       130   236    0        0      174      0        0

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

[301 rows x 13 columns]
```

[53]:
```
# Now check datatype
data.dtypes
```

[53]:
```
age         int32
sex         int64
cp          int32
trestbps    int32
chol        int64
fbs         int64
restecg     int64
thalach     int64
exang       int64
oldpeak     int32
slope       int64
thal        int64
target      int32
dtype: object
```

Now ,I got clean data set .it is raedy for analysis.

[49]:
```
data
```

[49]:
```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0      63    1   3       145   233    1        0      150      0        2
1      37    1   2       130   250    0        1      187      0        3
2      41    0   1       130   204    0        0      172      0        1
3      56    1   1       120   236    0        1      178      0        0
4      57    0   0       120   354    0        1      163      1        0
..    …  …  ..       …    …  …      …       …     …       …
298    57    0   0       140   241    0        1      123      1        0
299    45    1   3       110   264    0        1      132      0        1
```

```
300   68    1   0       144   193    1        1      141      0        3
301   57    1   0       130   131    0        1      115      1        1
302   57    0   1       130   236    0        0      174      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

[300 rows x 14 columns]
```