# Import Libraries

import numpy as np import pandas as pd import seaborn as sns

import matplotlib.pyplot as plt

# Load the Data

from google.colab import files uploaded = files.upload()

heart.csv

Choose Files

**heart.csv**(application/vnd.ms-excel) - 11328 bytes, last modified: 12/11/2021 - 100% done

Saving heart.csv to heart.csv

# Store the data into a variable

df = pd.read\_csv("heart.csv")

# Print First 5 rows of data

df.head()

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

# Get the Shape of Data

df.shape

(303, 14)

# Count the null or empty value in each column

df.isna().sum()

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

# Another way to check null or missing values

df.isnull().values.any() False

# View some basic statistics

df.describe()

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **age** | **sex** | **cp** | **trestbps** | **chol** | **fbs** | **res** |
| **count** | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.00 |
| **mean** | 54.366337 | 0.683168 | 0.966997 | 131.623762 | 246.264026 | 0.148515 | 0.52 |
| **std** | 9.082101 | 0.466011 | 1.032052 | 17.538143 | 51.830751 | 0.356198 | 0.52 |

**Get the count of the number of patients with and without heart disease**

df['target'].value\_counts()

**min** 29.000000 0.000000 0.000000 94.000000 126.000000 0.000000 0.00

**25%** 47.500000 0.000000 0.000000 120.000000 211.000000 0.000000 0.00

**50%** 55.000000 1.000000 1.000000 130.000000 240.000000 0.000000 1.00

**75%** 61.000000 1.000000 2.000000 140.000000 274.500000 0.000000 1.00

**max** 77.000000 1.000000 3.000000 200.000000 564.000000 1.000000 2.00

1 165

0 138

Name: target, dtype: int64

# Visualize the count

sns.countplot(df['target'])

/usr/local/lib/python3.7/dist-packages/seaborn/\_decorators.py:43: FutureWarning: P FutureWarning

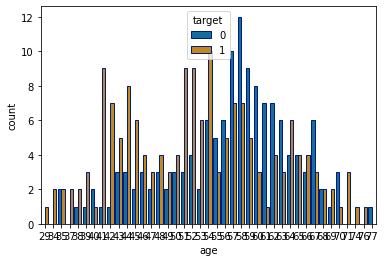
<matplotlib.axes.\_subplots.AxesSubplot at 0x7fc0f460a490>



**Look at the number of people with disease that exceed the number of people without disease**

#Visualize the count

sns.countplot(x='age' , hue = 'target',data=df,palette='colorblind',edgecolor=sns.color\_pa

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fc0f3dd9410>

#Get the correlation of the columns df.corr()

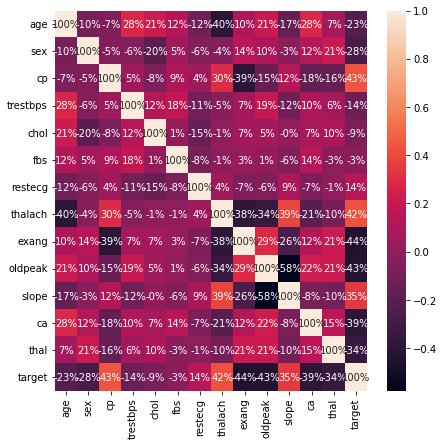
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **age** | **sex** | **cp** | **trestbps** | **chol** | **fbs** | **restecg** | **th** |
| **age** | 1.000000 | -0.098447 | -0.068653 | 0.279351 | 0.213678 | 0.121308 | -0.116211 | -0.3 |
| **sex** | -0.098447 | 1.000000 | -0.049353 | -0.056769 | -0.197912 | 0.045032 | -0.058196 | -0.0 |
| **cp** | -0.068653 | -0.049353 | 1.000000 | 0.047608 | -0.076904 | 0.094444 | 0.044421 | 0.2 |
| **trestbps** | 0.279351 | -0.056769 | 0.047608 | 1.000000 | 0.123174 | 0.177531 | -0.114103 | -0.0 |
| **chol** | 0.213678 | -0.197912 | -0.076904 | 0.123174 | 1.000000 | 0.013294 | -0.151040 | -0.0 |
| **fbs** | 0.121308 | 0.045032 | 0.094444 | 0.177531 | 0.013294 | 1.000000 | -0.084189 | -0.0 |
| **restecg** | -0.116211 | -0.058196 | 0.044421 | -0.114103 | -0.151040 | -0.084189 | 1.000000 | 0.0 |
| **thalach** | -0.398522 | -0.044020 | 0.295762 | -0.046698 | -0.009940 | -0.008567 | 0.044123 | 1.0 |
| **exang** | 0.096801 | 0.141664 | -0.394280 | 0.067616 | 0.067023 | 0.025665 | -0.070733 | -0.3 |
| **oldpeak** | 0.210013 | 0.096093 | -0.149230 | 0.193216 | 0.053952 | 0.005747 | -0.058770 | -0.3 |
| **slope** | -0.168814 | -0.030711 | 0.119717 | -0.121475 | -0.004038 | -0.059894 | 0.093045 | 0.3 |
| **ca** | 0.276326 | 0.118261 | -0.181053 | 0.101389 | 0.070511 | 0.137979 | -0.072042 | -0.2 |

#Visualize the data plt.figure(figsize=(7,7)) sns.heatmap(df.corr(),annot=True,fmt='.0%')

**thal** 0.068001 0.210041 -0.161736 0.062210 0.098803 -0.032019 -0.011981 -0.0

**target** -0.225439 -0.280937 0.433798 -0.144931 -0.085239 -0.028046 0.137230 0.4

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fc0f450c750>



# Split the data into feature data and target data

X = df.iloc[:,:-1].values

Y= df.iloc[:,-1].values

# Split the data into 75% training data set and 25% testing data set

from sklearn.model\_selection import train\_test\_split

X\_train , X\_test , Y\_train , Y\_test = train\_test\_split(X,Y ,test\_size=0.25 ,stratify=Y, ra

# Features Scaling

## Scale the values in the data to be values between 0 and 1 inclusive

from sklearn.preprocessing import StandardScaler sc=StandardScaler()

X\_train = sc.fit\_transform(X\_train) X\_test = sc.transform(X\_test)

# Training the LogisticRegression model with training data

from sklearn.linear\_model import LogisticRegression model = LogisticRegression() model.fit(X\_train,Y\_train)

LogisticRegression()

# Model Evaluation

## Accuracy Score

from sklearn.metrics import accuracy\_score X\_train\_prediction = model.predict(X\_train)

training\_data\_accuracy = accuracy\_score(X\_train\_prediction,Y\_train)

print("Accuracy on Training Data: ",training\_data\_accuracy) Accuracy on Training Data: 0.8325991189427313

X\_test\_prediction = model.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_prediction,Y\_test)

print("Accuracy on Testing Data: ",test\_data\_accuracy) Accuracy on Testing Data: 0.8157894736842105

# Building Prediction System

input\_data=(59,1,0,135,234,0,1,161,0,0.5,1,0,3)

#Changing the input data into numpy array input\_data\_narray= np.asarray(input\_data)

#reshaping the numpy array and predicting for only on instance input\_data\_reshape = input\_data\_narray.reshape(1,-1)

prediction = model.predict(input\_data\_reshape) print(prediction)

if(prediction[0] == 0):

print("The Person does not have any Heart Disease") else:

print("Person has Heart Disease !")

[1]

Person has Heart Disease !

# Credits

## Created By:- Mayur Mahesh More

 0s completed at 6:27 PM