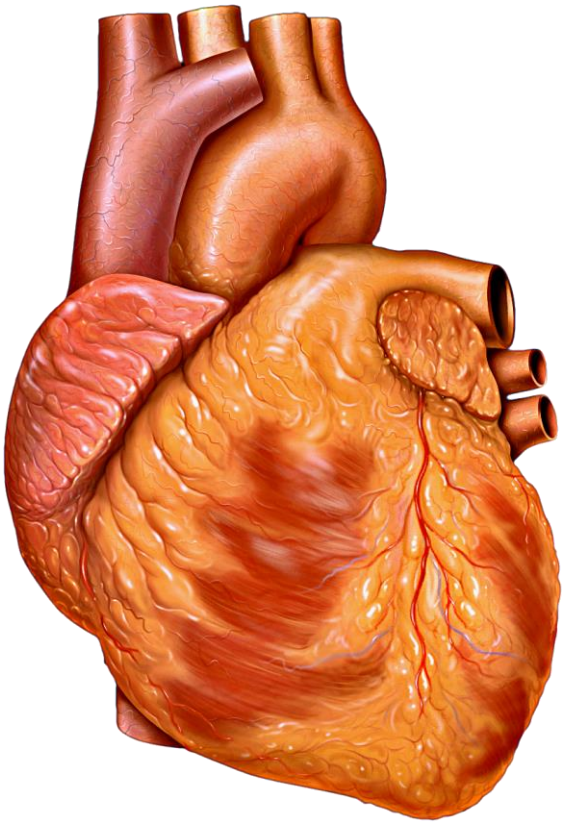


Heart-Disease-Prediction

Given clinical parameters about a patient, can we predict whether or not they have heart disease?

MAJOR PROJECT 1

NAME: ANIVERTHY AMRUTESH

COLLEGE: DR. AMBEDKAR INSTITUTE OF TECHNOLOGY

COURSE: BE

BRANCH: COMPUTER SCIENCE

SEMISTER:2

YEAR:2021-2022

EMAIL:aniverthyamruteshgs@gmail.com

PHNO:6361370806

Major links

1:Problem Statement

2: GitHub Details

3:Web Deployment Details

4:ML Code

Predicting heart disease using machine learning

Problem Definition

In a statement,

Given clinical parameters about a patient, can we predict whether or not they have heart disease?

Features

This is where you'll get different information about each of the features in your data. You can do this via doing your own research (such as looking at the links above) or by talking to a subject matter expert (someone who knows about the dataset).

Dataset Details:

1. age - age in years
2. sex - (1 = male; 0 = female)
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 blood pressure (in mm Hg on admission to the hospital) anything above 130-140 is typically cause for concern
5. chol - serum cholestoral in mg/dl
    - serum = LDL + HDL + .2 * triglycerides
    - above 200 is cause for concern
6. fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
    - '>126' mg/dL signals diabetes
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) (= the predicted attribute)

Aniverthy/Heart-Disease-Prediction

https://heartdiseasepredictbyaniverthy.herokuapp.com/

ML CODE

```
#MAJOR PROJECT - 1
#Choose any dataset of your choice and apply a suitable CLASSIFIER/REGRESSOR and if
 possible deploy it on heroku.
#NOTE: Heroku deployment is not compulsory
#Dataset = "https://github.com/Aniverthy/Heart-Disease-
Prediction/blob/main/heart.csv"

#step1:dataframe creation
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv("https://raw.githubusercontent.com/Aniverthy/Heart-Disease-
Prediction/main/heart.csv")
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

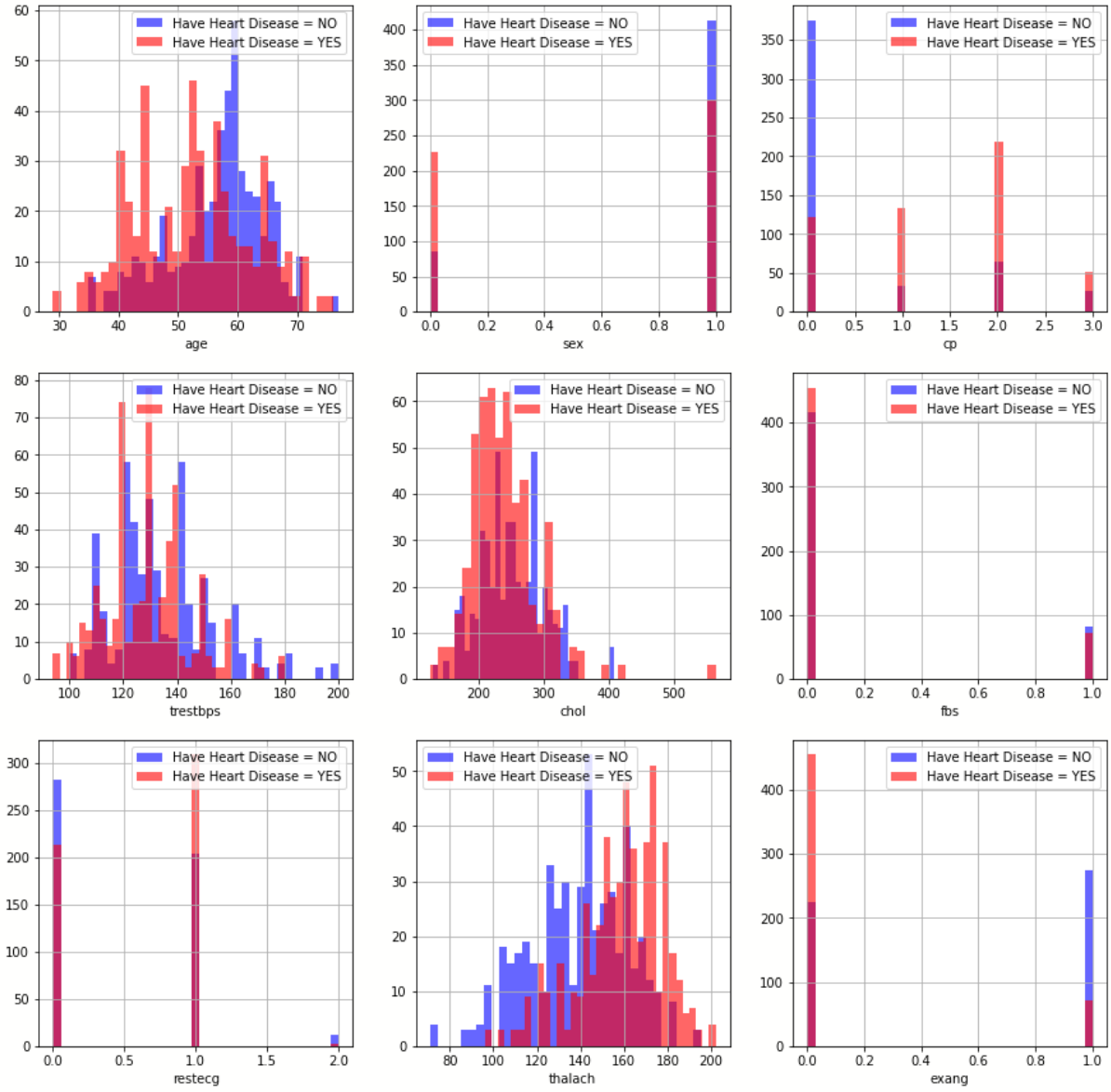
```
#Step2: Exploratory Data Analysis
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

```
df.shape
```

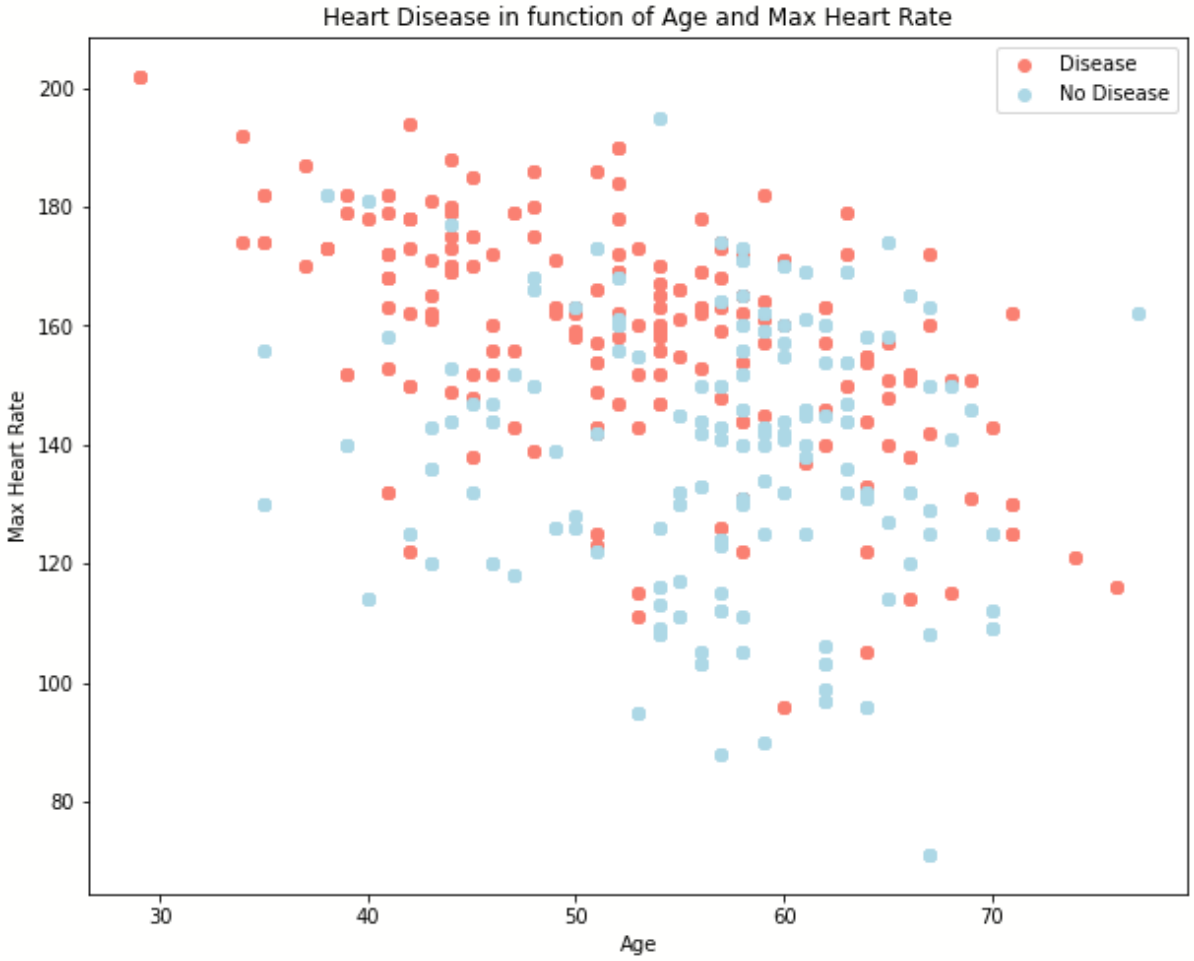
```
(1025, 14)

df.isnull().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

df=df.drop(['slope'], axis=1)
df.describe()
age   sex   cp    trestbps    chol  fbs   restecg     thalach     exang oldpeak     ca
      thal  target
count 1025.000000 1025.000000 1025.000000 1025.000000 1025.00000  1025.000000
      1025.000000 1025.000000 1025.000000 1025.000000 1025.000000 1025.000000
      1025.000000
mean  54.434146   0.695610    0.942439    131.611707  246.00000   0.149268    0.529756
      149.114146  0.336585    1.071512    0.754146    2.323902    0.513171
std   9.072290    0.460373    1.029641    17.516718   51.59251    0.356527    0.527878
      23.005724   0.472772    1.175053    1.030798    0.620660    0.500070
min   29.000000   0.000000    0.000000    94.000000   126.00000   0.000000    0.000000
      71.000000   0.000000    0.000000    0.000000    0.000000    0.000000
25%   48.000000   0.000000    0.000000    120.000000  211.00000   0.000000    0.000000
      132.000000  0.000000    0.000000    0.000000    2.000000    0.000000
50%   56.000000   1.000000    1.000000    130.000000  240.00000   0.000000    1.000000
      152.000000  0.000000    0.800000    0.000000    2.000000    1.000000
75%   61.000000   1.000000    2.000000    140.000000  275.00000   0.000000    1.000000
      166.000000  1.000000    1.800000    1.000000    3.000000    1.000000
max   77.000000   1.000000    3.000000    200.000000  564.00000   1.000000    2.000000
      202.000000  1.000000    6.200000    4.000000    3.000000    1.000000

df.target.value_counts()
#gives no of person have heart disease
#1 represents having heart disease and 0 represents donot having heart disease
1    526
0    499
Name: target, dtype: int64

#Step 3: Data visualization
plt.figure(figsize=(15, 15))
for i, column in enumerate(df):
  try:
    plt.subplot(3, 3, i+1)
```

```
    df[df["target"] == 0][column].hist(bins=35, color='blue', label='Have Heart Dis
ease = NO', alpha=0.6)
    df[df["target"] == 1][column].hist(bins=35, color='red', label='Have Heart Dise
ase = YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)
  except ValueError:
        break
```




```
# Create another figure
plt.figure(figsize=(10, 8))

# Scatter with postivie examples
plt.scatter(df.age[df.target==1],
            df.thalach[df.target==1],
            c="salmon")

# Scatter with negative examples
plt.scatter(df.age[df.target==0],
            df.thalach[df.target==0],
            c="lightblue")
```

```
# Add some helpful info
plt.title("Heart Disease in function of Age and Max Heart Rate")
plt.xlabel("Age")
plt.ylabel("Max Heart Rate")
plt.legend(["Disease", "No Disease"]);
```




```
#step 4 divide into input and output:
#step 5 TEST and TRAIN data
from sklearn.model_selection import train_test_split

x= df.drop('target', axis=1).values
y= df.target.values

x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=42)



#step 6:SCALING or NORMALISATION -DONE ONLY FOR INPUTS
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)
```

```
#step 7:model classification
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()



#step 8: fitting the training values
model.fit(x_train, y_train)

LogisticRegression()




#step 9:predict the output
y_pred=model.predict(x_test)
y_pred

array([1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0,
       1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1,
       1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0,
       1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0])


y_test

array([1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0])
```

```
#accuracy
test_score = accuracy_score(y_test, model.predict(x_test)) * 100
train_score = accuracy_score(y_train, model.predict(x_train)) * 100

results_df = pd.DataFrame(data=[["Logistic Regression", train_score, test_score]],
               columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
results_df
```

	Model Training	Accuracy %	Testing Accuracy %
0	Logistic Regression	86.328125	80.544747

```
#individual prediction
model.predict([x_train[10]])
#1 implys presence of heart disease
```

This is deployed in a Web APP using Heroku Cloud Data

Aniverthy/Heart-Disease-Prediction

https://heartdiseasepredictbyaniverthy.herokuapp.com/