Health Care

Cardiovascular diseases are the leading cause of death globally. It is therefore necessary to identify the causes and develop a system to predict heart attacks in an effective manner. The data below has the information about the factors that might have an impact on cardiovascular health.

In [30]:
```
#Iporting all required libraries.
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import matplotlib as plt
```

In [31]:
```
# Importing given Dataset
```

In [32]:
```
cvd=pd.read_csv("CVD.csv")
```

In [33]:
```
#viewing top 5 rows data.
cvd.head()
```

Out[33]:

	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

In [34]:
```
# to find out the dimentions of the given complete Dataset.
cvd.shape
```

Out[34]:
```
(303, 14)
```

In [35]:
```
cvd.tail(2)
```

Out[35]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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

In [36]:
```
#to know more about the datatypes, Column Headings & Null Values.
cvd.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
```

In [37]:
```
#Descriptive Stats
cvd.describe()
```

Out[37]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	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	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

In [38]:
```
#to search Null values
cvd.isnull().sum()
```

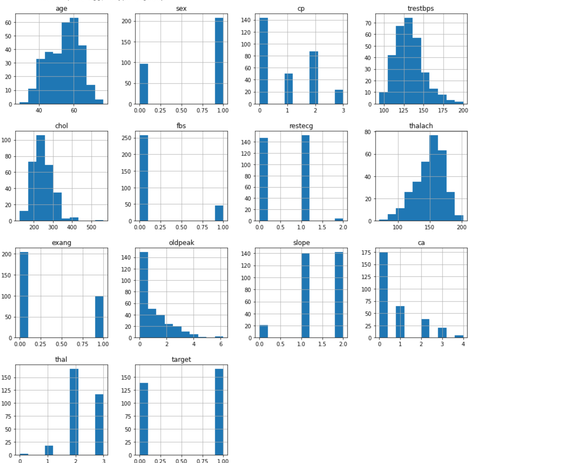
Out[38]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

In [39]:
```
# Count 0s and 1s in target column and to which we are considering an output label.
N,P=cvd["target"].value_counts()
print("number of positive people", N)
print("number of negative people", P)
```

```
number of positive people 165
number of negative people 138
```

In [56]:
```
cvd.hist(bins=10, figsize=(15,15))
```

Out[56]:
```
array([[<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'sex'}>,
        <AxesSubplot:title={'center':'cp'}>,
        <AxesSubplot:title={'center':'trestbps'}>],
       [<AxesSubplot:title={'center':'chol'}>,
        <AxesSubplot:title={'center':'fbs'}>,
        <AxesSubplot:title={'center':'restecg'}>,
        <AxesSubplot:title={'center':'thalach'}>],
       [<AxesSubplot:title={'center':'exang'}>,
        <AxesSubplot:title={'center':'oldpeak'}>,
        <AxesSubplot:title={'center':'slope'}>,
        <AxesSubplot:title={'center':'ca'}>],
       [<AxesSubplot:title={'center':'thal'}>,
        <AxesSubplot:title={'center':'target'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```



In [57]:
```
#droping the target column from X and saving it into Y or Seperating input and output labels from given dataset.
X=cvd.drop(columns="target", axis=1)
Y=cvd["target"]
```

In [43]:
```
Y
```

Out[43]:
```
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
```

Spliting Dataset into 2 for training and testing

In [44]:
```
X_train, X_test, Y_train,Y_test=train_test_split(X,Y, test_size=0.2, stratify=Y,random_state=2)
```

In [45]:
```
X_train.shape
```

Out[45]:
```
(242, 13)
```

Model Building

In [47]:
```
#saving Logistic Regression function into model variable for its easy use.
model=LogisticRegression()
```

In [48]:
```
model.fit(X_train, Y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: 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(
```

Out[48]:
```
LogisticRegression()
```

In [49]:
```
# using predict method of Logistic Regression function.
X_model_cvd=model.predict(X_train)
training_accuracy_cvd=accuracy_score(X_model_cvd, Y_train)
```

In [50]:
```
print("accuracy score is :", training_accuracy_cvd)
```

```
accuracy score is : 0.8512396694214877
```

In [51]:
```
X_test_model_cvd=model.predict(X_test)
testing_accuracy_cvd=accuracy_score(X_test_model_cvd, Y_test)
print("accuracy score is :", testing_accuracy_cvd)
```

```
accuracy score is : 0.819672131147541
```

it is an Overfitting model

Predictive System

In [52]:
```
input_data=(63,1,3,145,233,1,0,150,0,2.3,0,0,1)
#Converting input data to an array
np_input=np.asarray(input_data)
```

In [53]:
```
#reshaping input data to predict better.
reshape_np_input=np_input.reshape(1,-1)
```

In [54]:
```
prediction=model.predict(reshape_np_input)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```

In [55]:
```
print(prediction)
if prediction==0:
    print("The person has lesser chanches of having CVD")
else:
    print("The person has greater chanches of having CVD")
```

```
[1]
The person has greater chanches of having CVD
```

As accuracy score is >75 hence our predictive model is acceptable.

In [ ]:
```

```

In [ ]:
```

```