MAJOR PROJECT

NAME : DEEKSHA R SHETTY

COLLEGE : DAYANANDA SAGAR UNIVERSITY

YEAR : FIRST YEAR

```
'''
                                                    MAJOR PROJECT - 1
NAME              :  DEEKSHA R SHETTY
COLLEGE           :  DAYANANDA SAGAR UNIVERSITY
BRANCH            :  AEROSPACE ENGINEERING
YEAR              :  FIRST YEAR
DATASET SOURCE    :  KAGGLE
MAIL ID           :  deekshashetty082@gmail.com
'''
```

```
# creating dataframe
import pandas as pd
df = pd.read_csv('/content/heart.csv')
df
```

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

1025 rows × 14 columns

```
# information about the dataframe
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
```

```
# 1025 rows, 14 columns
df.shape
```

```
(1025, 14)
```

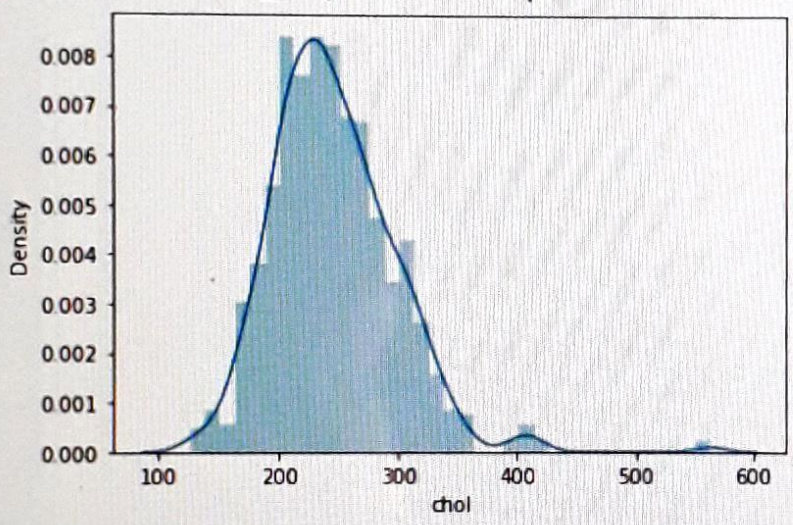
```
# total number of elements in the dataframe
df.size
```

```
14350
```

```
# visualization
```

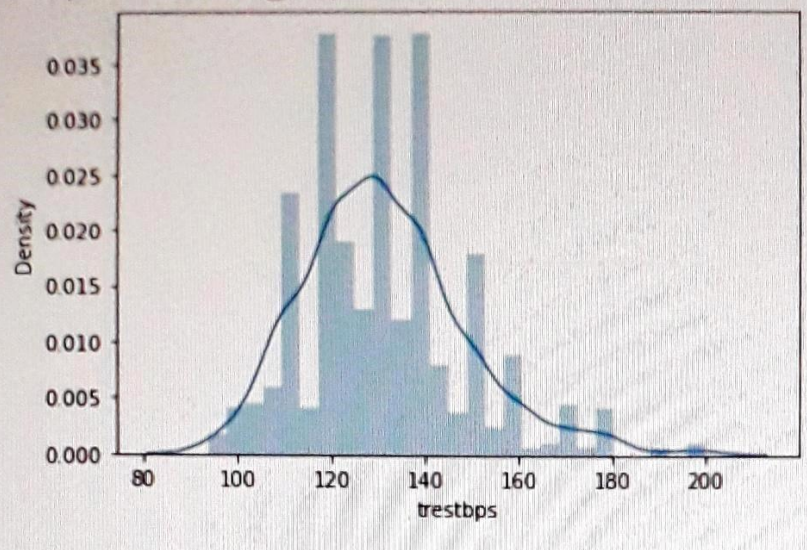
```
# visulization on serum cholestero in mg/dl
import seaborn as sns
sns.distplot(df['chol'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e6b034b90>
```



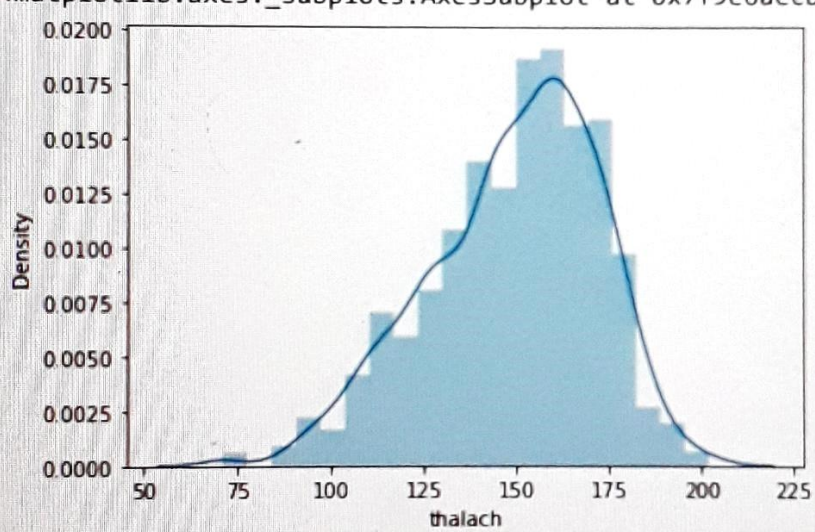

```
# visualization on resting blood pressure in mmHg
import seaborn as sns
sns.distplot(df['trestbps'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e6af84550>
```




```
# visualization on maximum heart rate achieved
import seaborn as sns
sns.distplot(df['thalach'])
```

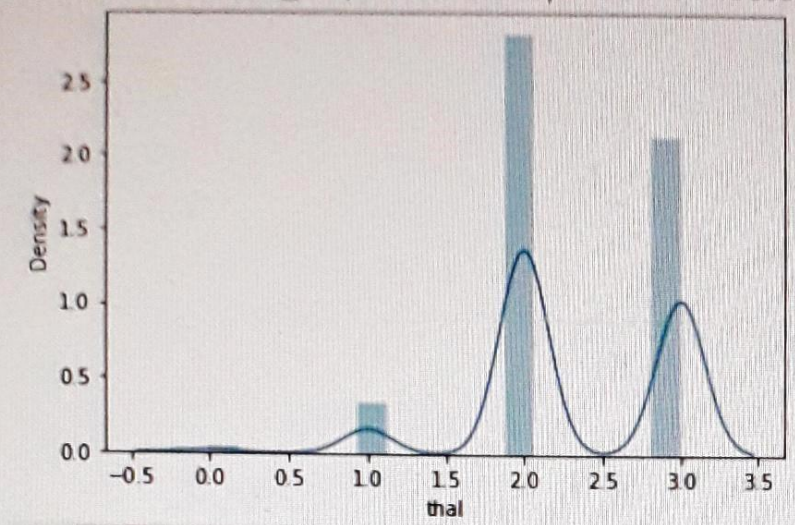
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e6aecb7d0>
```



+ Code + Text

```
# visualization on Thalassemia (3 = normal, 6 = fixed defect, 7 = reversible defect )
import seaborn as sns
sns.distplot(df['thal'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e6ae05290>
```




```
# removing 'oldpeak','age','sex','cp','exang','slope','ca' columns
df=df.drop(['oldpeak','age','sex','cp','exang','slope','ca'],axis=1)
df
```

	trestbps	chol	fbs	restecg	thalach	thal	target
0	125	212	0	1	168	3	0
1	140	203	1	0	155	3	0
2	145	174	0	1	125	3	0
3	148	203	0	1	161	3	0
4	138	294	1	1	106	2	0
...	...	...	...	...	...	...	...
1020	140	221	0	1	164	2	1
1021	125	258	0	0	141	3	0
1022	110	275	0	0	118	2	0
1023	110	254	0	0	159	2	1
1024	120	188	0	1	113	3	0

1025 rows × 7 columns

```
# input
x=df.iloc[:,0:6].values
x
```

```
array([[125, 212,   0,   1, 168,   3],
       [140, 203,   1,   0, 155,   3],
       [145, 174,   0,   1, 125,   3],
       ...,
       [110, 275,   0,   0, 118,   2],
```

```
       [110, 254,   0,   0, 159,   2],
       [120, 188,   0,   1, 113,   3]])
```

```
# output
y=df.iloc[:,6].values
y
```

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

```
#Train and test variables
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0)
```

```
#7.Running a regressor (applying a suitable algorithm)
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

```
# Model fitting
model.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
# Predicting the output
y_pred=model.predict(x_test)
y_pred
```

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

```
y_test
```

```
array([1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1,
       1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1,
       0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1,
```

```
       0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1])
```

```
# Accuracy of the model
from sklearn.metrics import accuracy_score
accuracy_score(y_pred,y_test)*100
```

```
75.4863813229572
```

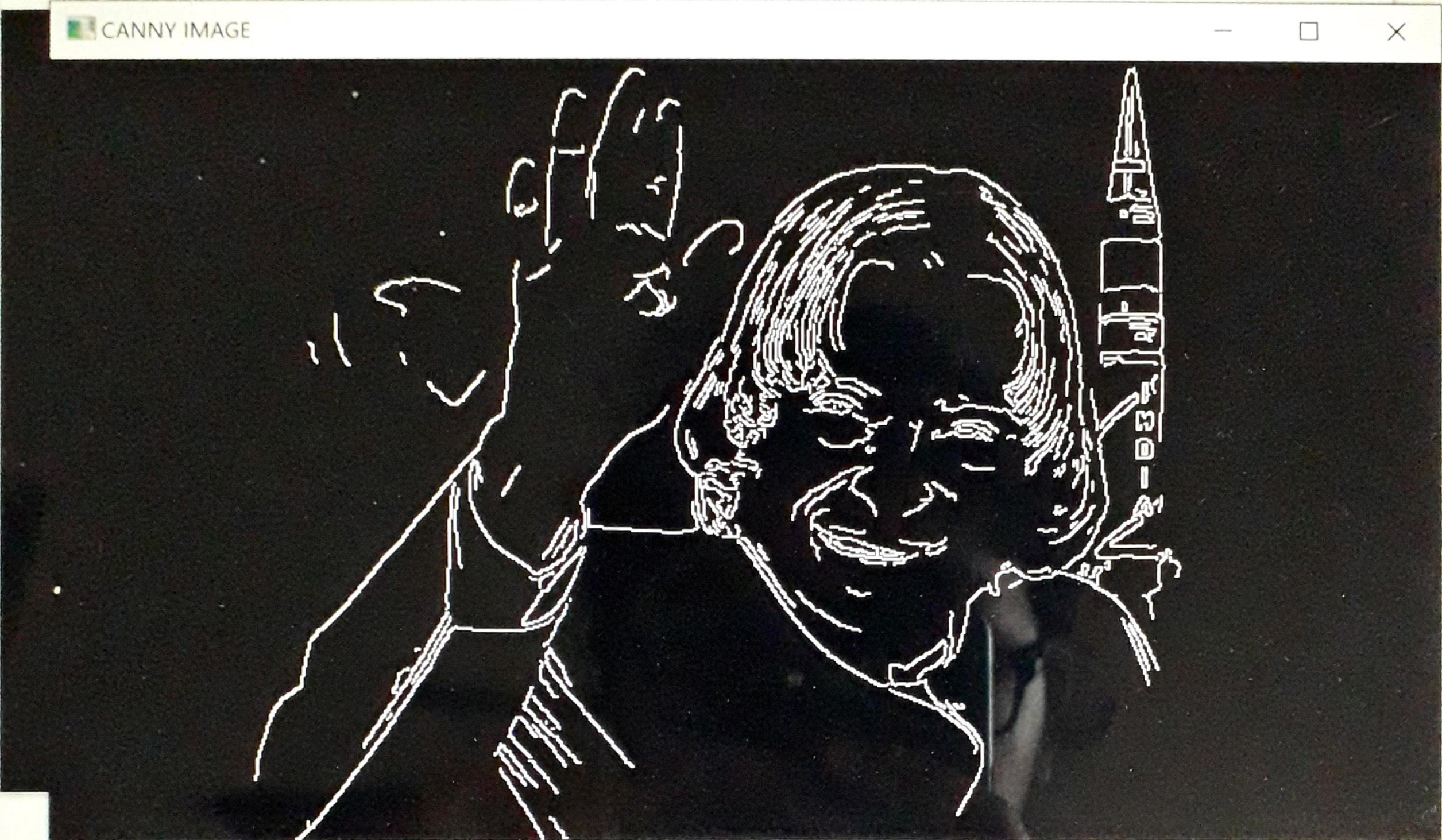
```
# individual predicition - o indicates that the person is not affected by the heart attack
model.predict([x_train[13]])
```

```
array([0])
```

MAJOR PROJECT – 2

IMAGE PROCESSING

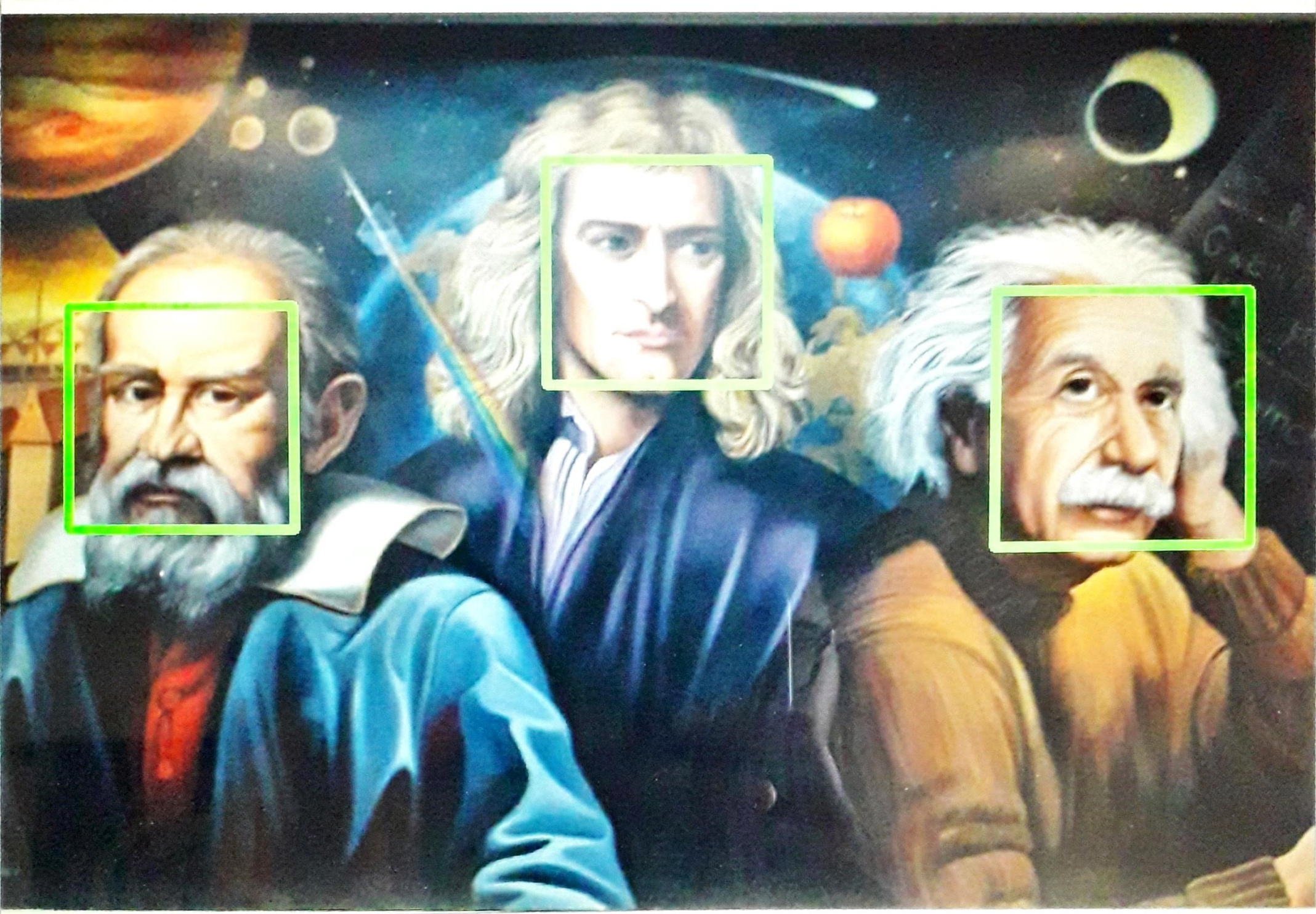
```
# LAPLACIAN AND CANNY
import cv2
img=cv2.imread('00.jpg',0)
laplacian=cv2.Laplacian(img,cv2.CV_8U)
cv2.imshow('LAPLACIAN IMAGE',laplacian)

canny = cv2.Canny(img,125,200)
cv2.imshow('CANNY IMAGE',canny)

cv2.waitKey(0)
cv2.destroyAllWindows()
```


CANNY IMAGE


INDIA

```python
# FACE DETECTION IN AN IMAGE
import cv2

face_cascade=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
img=cv2.imread('scientists.jpg')
gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

faces=face_cascade.detectMultiScale(gray,1.1,20)

for x,y,w,h in faces:
    img=cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),5)



cv2.imshow('FACE DETECTION',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

FACE DETECTION



GITHUB ACCOUNT LINK : https://github.com/12-03-2003/RINEX