Assignment 1

```
Name : Kaustubh Gajanan Indulkar
Roll No. : 35026(25027)
Class : TE-IT-A
```

Question

1. Data preparation:
   Download heart dataset from following link.
   https://www.kaggle.com/zhaoyingzhu/heartcsv
   (https://www.kaggle.com/zhaoyingzhu/heartcsv)
   Perform following operation on given dataset.
   a) Find Shape of Data
   b) Find Missing Values
   c) Find data type of each column
   d) Finding out Zero's
   e) Find Mean age of patients
   f) Now extract only Age, Sex, ChestPain, RestBP, Chol. Randomly divide dataset in training
   (75%) and testing (25%).
   Through the diagnosis test I predicted 100 report as COVID positive, but only 45 of those were
   actually positive. Total 50 people in my sample were actually COVID positive. I have total 500
   samples.
   Create confusion matrix based on above data and find
   I. Accuracy
   II. Precision
   III. Recall
   IV. F-1 score

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
```

In [ ]:
```
pwd
```

Out[2]:
```
'/kaggle/working'
```

In [ ]:
```
df = pd.read_csv('../input/heartcsv/Heart.csv')
```

In [ ]:
```
df.head()
```

Out[4]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpea
0	1	63	1	typical	145	233	1	2	150	0	2.
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.
3	4	37	1	nonanginal	130	250	0	0	187	0	3.
4	5	41	0	nontypical	130	204	0	2	172	0	1.

a) Find Shape of Data

In [ ]:
```
df.shape #303, 15
```

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

b) Find Missing Values

In [ ]:
```
df.isnull().sum()
```

Out[6]:
```
Unnamed: 0    0
Age           0
Sex           0
ChestPain     0
RestBP        0
Chol          0
Fbs           0
RestECG       0
MaxHR         0
ExAng         0
Oldpeak       0
Slope         0
Ca            4
Thal          2
AHD           0
dtype: int64
```

In [ ]:
```
df.count()
```

Out[7]:
```
Unnamed: 0    303
Age           303
Sex           303
ChestPain     303
RestBP        303
Chol          303
Fbs           303
RestECG       303
MaxHR         303
ExAng         303
Oldpeak       303
Slope         303
Ca            299
Thal          301
AHD           303
dtype: int64
```

c) Find data type of each column

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  303 non-null    int64  
 1   Age         303 non-null    int64  
 2   Sex         303 non-null    int64  
 3   ChestPain   303 non-null    object 
 4   RestBP      303 non-null    int64  
 5   Chol        303 non-null    int64  
 6   Fbs         303 non-null    int64  
 7   RestECG     303 non-null    int64  
 8   MaxHR       303 non-null    int64  
 9   ExAng       303 non-null    int64  
 10  Oldpeak     303 non-null    float64
 11  Slope       303 non-null    int64  
 12  Ca          299 non-null    float64
 13  Thal        301 non-null    object 
 14  AHD         303 non-null    object 
dtypes: float64(2), int64(10), object(3)
memory usage: 35.6+ KB
```

In [ ]:
```
df.dtypes
```

Out[9]:
```
Unnamed: 0      int64
Age             int64
Sex             int64
ChestPain      object
RestBP          int64
Chol            int64
Fbs             int64
RestECG         int64
MaxHR           int64
ExAng           int64
Oldpeak       float64
Slope           int64
Ca            float64
Thal           object
AHD            object
dtype: object
```

d) Finding out Zero's

In [ ]:
```
df==0
```

Out[10]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Old
0	False	False	False	False	False	False	False	False	False	True	
1	False	False	False	False	False	False	True	False	False	False	
2	False	False	False	False	False	False	True	False	False	False	
3	False	False	False	False	False	False	True	True	False	True	
4	False	False	True	False	False	False	True	False	False	True	
...	...	...	...	...	...	...	...	...	...	...	
298	False	False	False	False	False	False	True	True	False	True	
299	False	False	False	False	False	False	False	True	False	True	
300	False	False	False	False	False	False	True	True	False	False	
301	False	False	True	False	False	False	True	False	False	True	
302	False	False	False	False	False	False	True	True	False	True	

303 rows × 15 columns

In [ ]:
```
df[df==0]
```

Out[11]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpe
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	Na
1	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN	Na
2	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN	Na
3	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	NaN	0.0	Na
4	NaN	NaN	0.0	NaN	NaN	NaN	0.0	NaN	NaN	0.0	Na
...	...	...	...	...	...	...	...	...	...	...	
298	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	NaN	0.0	Na
299	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	0.0	Na
300	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	NaN	NaN	Na
301	NaN	NaN	0.0	NaN	NaN	NaN	0.0	NaN	NaN	0.0	(
302	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	NaN	0.0	(

303 rows × 15 columns

In [ ]:
```
(df == 0).sum()
```

Out[12]:
```
Unnamed: 0      0
Age             0
Sex            97
ChestPain       0
RestBP          0
Chol            0
Fbs           258
RestECG       151
MaxHR           0
ExAng         204
Oldpeak        99
Slope           0
Ca            176
Thal            0
AHD             0
dtype: int64
```

e) Find Mean age of patients

In [ ]:
```
np.mean(df['Age'])
```

Out[13]:
```
54.43894389438944
```

In [ ]:
```
df.Age.mean()
```

Out[14]:
```
54.43894389438944
```

f) Now extract only Age, Sex, ChestPain, RestBP, Chol. Randomly divide dataset in training (75%) and testing (25%).

In [ ]:
```
df.columns
```

Out[15]:
```
Index(['Unnamed: 0', 'Age', 'Sex', 'ChestPain', 'RestBP', 'Chol', 'Fbs',
       'RestECG', 'MaxHR', 'ExAng', 'Oldpeak', 'Slope', 'Ca', 'Thal', 'AH
D'],
      dtype='object')
```

In [ ]:
```
data = df[['Age', 'Sex', 'ChestPain', 'RestBP', 'Chol']]
```

In [ ]:
```
#Cross validation
```

In [ ]:
```
train,test = train_test_split(data,test_size=0.25,random_state=1)
```

In [ ]:
```
train.shape
```

Out[19]:
```
(227, 5)
```

In [ ]:
```
test.shape
```

Out[20]:
```
(76, 5)
```

Through the diagnosis test I predicted 100 report as COVID positive, but only 45 of those were
actually positive. Total 50 people in my sample were actually COVID positive. I have total 500
samples.
Create confusion matrix based on above data and find
I. Accuracy
II. Precision
III. Recall
IV. F-1 score

In [ ]:

```
actual = np.concatenate((np.ones(45),np.zeros(450),np.ones(5)))
actual
```

Out[21]:

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

In [ ]:

```
# run = np.array([1,0,1,1,1])
```

In [ ]:
```python
predicted = np.concatenate((np.ones(100),np.zeros(400)))
predicted
```

Out[23]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0.])
```

In [ ]:
```python
type(predicted)
```

Out[24]:
```
numpy.ndarray
```

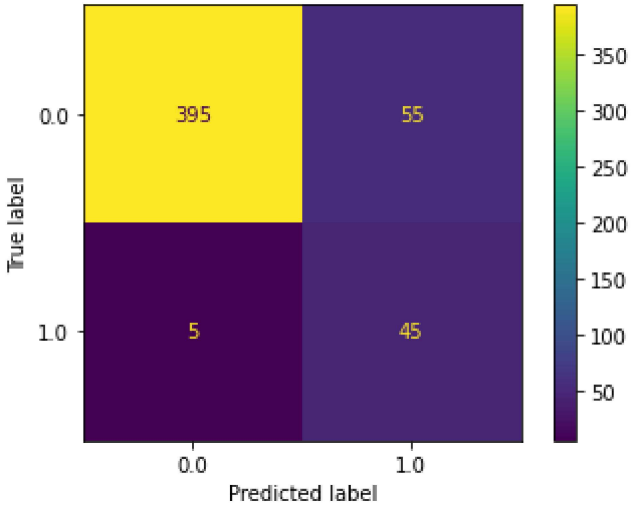
Confusion Matrix

image.png

In [ ]:
```python
from sklearn.metrics import ConfusionMatrixDisplay
```

In [ ]:
```
ConfusionMatrixDisplay.from_predictions(actual,predicted)
```

Out[26]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f133e
d13d10>
```



In [ ]:
```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

In [ ]:
```
print(classification_report(actual,predicted))
```

```
              precision    recall  f1-score   support

         0.0       0.99      0.88      0.93       450
         1.0       0.45      0.90      0.60        50

    accuracy                           0.88       500
   macro avg       0.72      0.89      0.76       500
weighted avg       0.93      0.88      0.90       500
```

In [ ]:
```
accuracy_score(actual,predicted)
```

Out[29]:
```
0.88
```