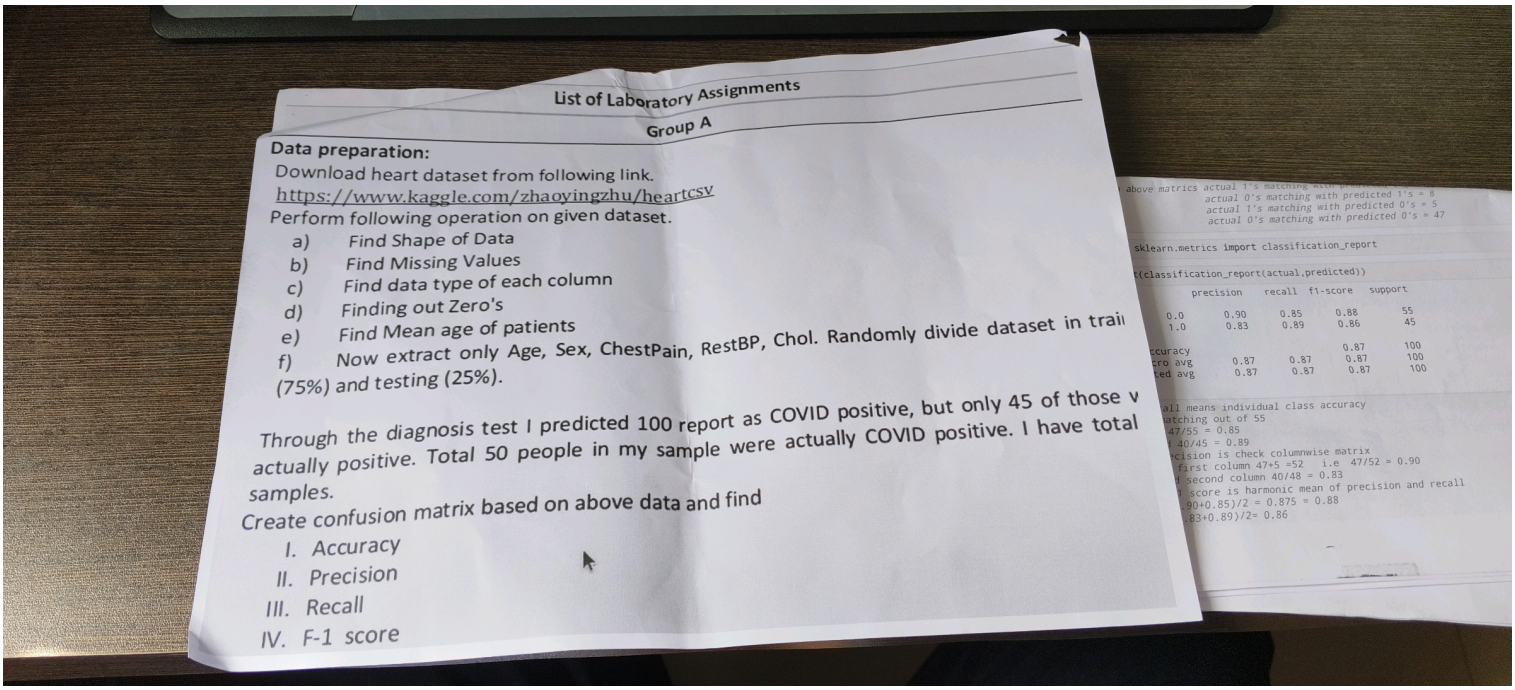


List of Laboratory Assignments

Group A

Data preparation:

Download heart dataset from following link.

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 trai (75%) and testing (25%).

Through the diagnosis test I predicted 100 report as COVID positive, but only 45 of those v actually positive. Total 50 people in my sample were actually COVID positive. I have total samples.

Create confusion matrix based on above data and find

I. Accuracy
II. Precision
III. Recall
IV. F-1 score

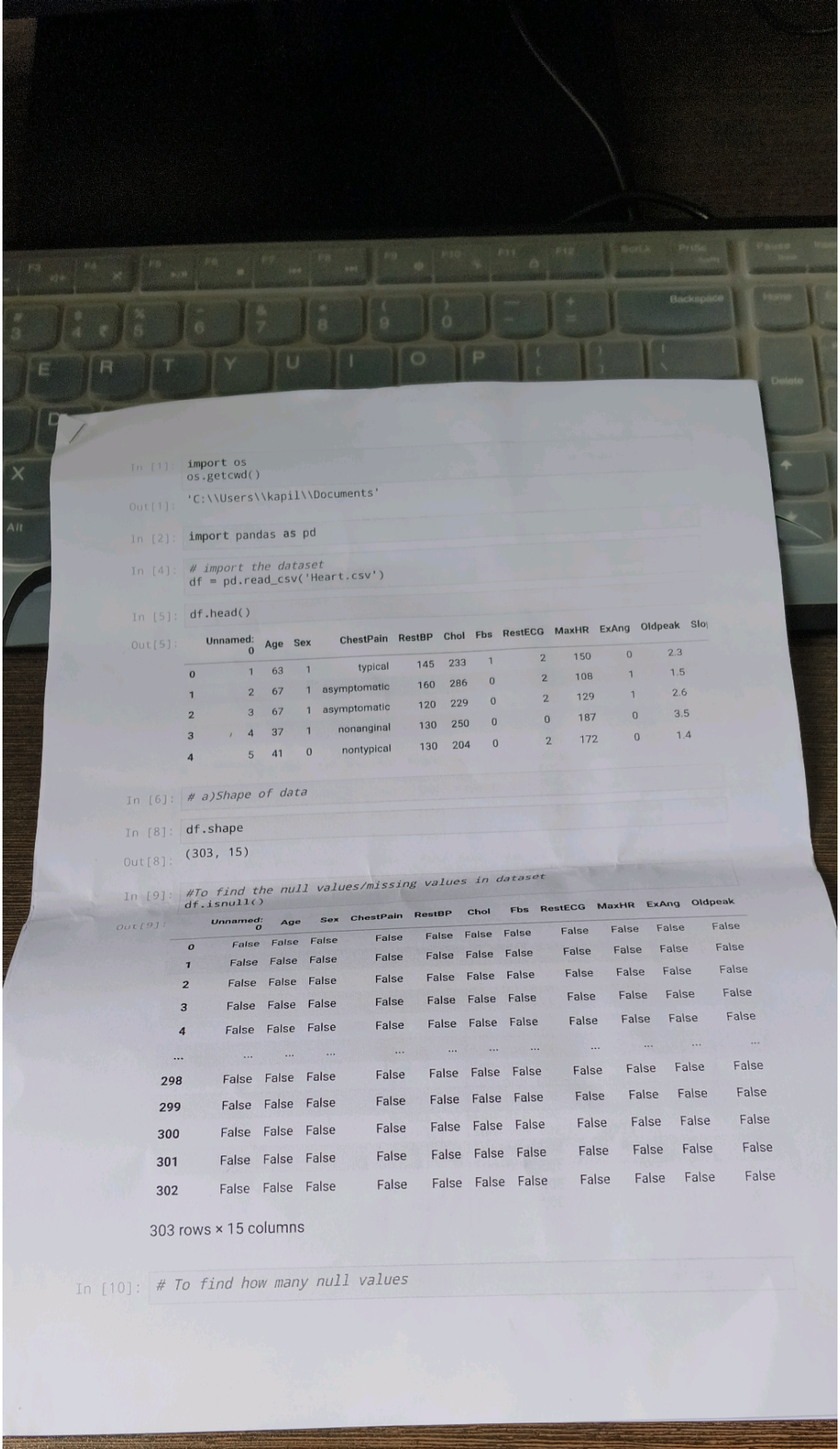
above matrics actual 1's matching with predicted 1's = 8
actual 0's matching with predicted 0's = 5
actual 1's matching with predicted 0's = 47

sklearn.metrics import classification_report

(classification_report(actual,predicted))

	precision	recall	f1-score	support
0.0	0.90	0.85	0.88	55
1.0	0.83	0.89	0.86	45
curacy			0.87	100
cro avg	0.87	0.87	0.87	100
ted avg	0.87	0.87	0.87	100

all means individual class accuracy
atching out of 55
47/55 = 0.85
40/45 = 0.89
cision is check columnwise matrix
first column 47+5 =52 i.e 47/52 = 0.90
second column 40/48 = 0.83
score is harmonic mean of precision and recall
.90+0.85)/2 = 0.875 = 0.88
.83+0.89)/2= 0.86



In [1]:
```
import os
os.getcwd()
```

Out[1]:
```
'C:\\Users\\kapil\\Documents'
```

In [2]:
```
import pandas as pd
```

In [4]:
```
# import the dataset
df = pd.read_csv('Heart.csv')
```

In [5]:
```
df.head()
```

Out[5]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slo
0	1	63	1	typical	145	233	1	2	150	0	2.3	
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6	
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5	
4	5	41	0	nontypical	130	204	0	2	172	0	1.4	

In [6]:
```
# a)Shape of data
```

In [8]:
```
df.shape
```

Out[8]:
```
(303, 15)
```

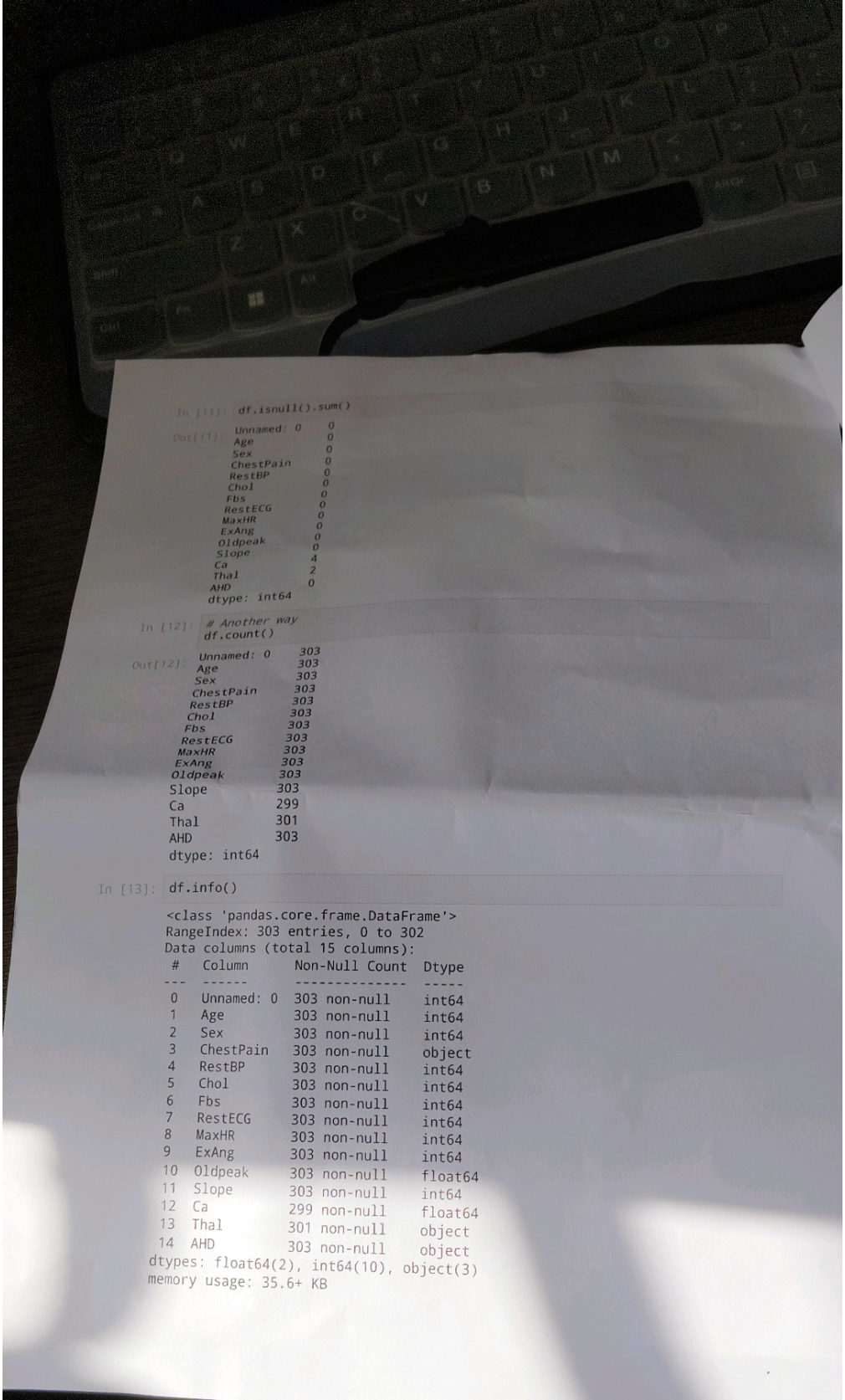
In [9]:
```
#To find the null values/missing values in dataset
df.isnull()
```

Out[9]:

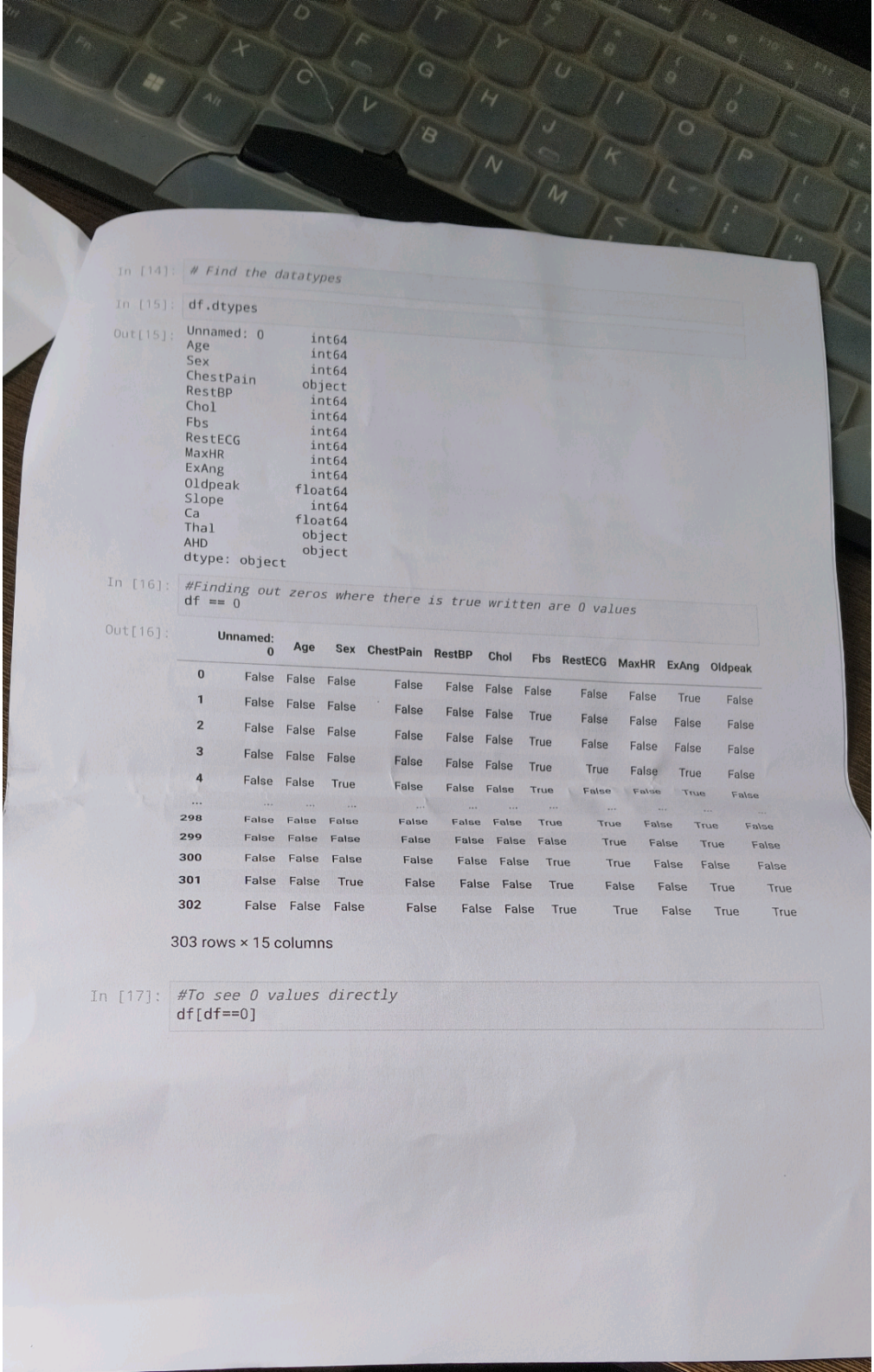
	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	
0	False	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	False	
3	False	False	False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

303 rows × 15 columns

In [10]:
```
# To find how many null values
```



```
In [11]: df.isnull().sum()

Out[11]: Unnamed: 0    0
         Age           0
         Sex           0
         ChestPain     0
         RestBP        0
         Chol          0
         Fbs           0
         RestECG       0
         MaxHR         0
         ExAng         0
         Oldpeak       0
         Slope         4
         Ca            2
         Thal          0
         AHD           0
         dtype: int64

In [12]: # Another way
         df.count()

Out[12]: 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

In [13]: 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 [14]:
```
# Find the datatypes
```

In [15]:
```
df.dtypes
```

Out[15]:
```
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
```

In [16]:
```
#Finding out zeros where there is true written are 0 values
df == 0
```

Out[16]:

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

303 rows × 15 columns

In [17]:
```
#To see 0 values directly
df[df==0]
```

Out[17]:

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

303 rows × 15 columns

In [18]:
```
# Finding mean age of patient
df.columns
```

Out[18]:
```
Index(['Unnamed: 0', 'Age', 'Sex', 'ChestPain', 'RestBP', 'Chol', 'Fbs',
       'RestECG', 'MaxHR', 'ExAng', 'Oldpeak', 'Slope', 'Ca', 'Thal', 'AH
D'],
      dtype='object')
```

In [19]:
```
df['Age']
```

Out[19]:
```
0      63
1      67
2      67
3      37
4      41
      ..
298    45
299    68
300    57
301    57
302    38
Name: Age, Length: 303, dtype: int64
```

In [20]:
```
# Find the mean age
df['Age'].mean()
```

Out[20]:
```
54.43894389438944
```

In [22]:
```
# Extract the only Age, Sex, ChestPain, RestBP,Chol,Randomly divide dataset
newdf = df[['Age','Sex','ChestPain','RestBP','Chol']]
```

In [23]:
```
newdf
```

Out[23]:

	Age	Sex	ChestPain	RestBP	Chol
0	63	1	typical	145	233
1	67	1	asymptomatic	160	286
2	67	1	asymptomatic	120	229
3	37	1	nonanginal	130	250
4	41	0	nontypical	130	204
...	...	...	...	...	...
298	45	1	typical	110	264
299	68	1	asymptomatic	144	193
300	57	1	asymptomatic	130	131
301	57	0	nontypical	130	236
302	38	1	nonanginal	138	175

303 rows × 5 columns

In [24]:
```
# Cross Validation
from sklearn.model_selection import train_test_split
```

In [25]:
```
train,test = train_test_split(df,random_state=0,test_size=0.25)
```

In [26]:
```
train.shape
```

Out[26]:
```
(227, 15)
```

In [27]:
```
test.shape
```

Out[27]:
```
(76, 15)
```

In [28]:
```
# Through the diagnosis test I predicted 100 report as COVID posiive, but or
# Total 50 people in my sample were actully COVID positive. I have total 50(
# Create confusion matrix based on above data and find
# 1. Accuracy 2. Precision  3. Recall  4. F-1 Score
```

In [29]:
```
import numpy as np
```

In [30]:
```
actual = list(np.ones(45)) + list(np.zeros(55))
```

In [31]:
```
np.array(actual)
```

Out[31]:
```
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., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 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 [32]:
```
predicted = list(np.ones(40)) + list(np.zeros(52)) + list(np.ones(8))
```

In [33]:
```
np.array(predicted)
```

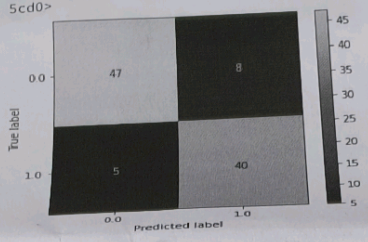
Out[33]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       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., 1., 1., 1., 1., 1.])
```

In [34]:
```
# Now if we match the above actual values with predicted values sequentially
# we will find that 1 mapped with 1, 1 mapped with 0, 0 mapped with 0 and 0
# To draw the matrix of it is called confusion matrix
```

In [35]:
```
from sklearn.metrics import ConfusionMatrixDisplay
```

In [36]:
```
ConfusionMatrixDisplay.from_predictions(actual,predicted)
```

Out[36]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2b318dc
5cd0>
```



In [37]:
```
# in above matrics actual 1's matching with predicted 1's = 40
#                  actual 0's matching with predicted 1's = 8
#                  actual 1's matching with predicted 0's = 5
#                  actual 0's matching with predicted 0's = 47
#
```

In [38]:
```
from sklearn.metrics import classification_report
```

In [39]:
```
print(classification_report(actual,predicted))
```

```
              precision    recall  f1-score   support

         0.0       0.90      0.85      0.88        55
         1.0       0.83      0.89      0.86        45

    accuracy                           0.87       100
   macro avg       0.87      0.87      0.87       100
weighted avg       0.87      0.87      0.87       100
```

In [40]:
```
# Recall means individual class accuracy
#47 matching out of 55
# so 47/55 = 0.85
# and 40/45 = 0.89
# precision is check columnwise matrix
# so first column 47+5 =52   i.e  47/52 = 0.90
# and second column 40/48 = 0.83
# f-1 score is harmonic mean of precision and recall
# (0.90+0.85)/2 = 0.875 = 0.88
# (0.83+0.89)/2= 0.86
```