Hello this is @Ranjeet_Kumbhar, Enjoy the Notebook

GitHub:https://github.com/RanjeetKumbhar01/TE_IT_ML_ASSIGNMENTS_SPPU

Question

1. 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 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

```
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
```

```
pwd
```

```
'/kaggle/working'
```

```
df = pd.read_csv('../input/heartcsv/Heart.csv')
```

```
df.head()
```

```
   Unnamed: 0  Age  Sex     ChestPain  RestBP  Chol  Fbs  RestECG
MaxHR  \
0           1   63    1       typical     145   233    1        2
150
```

```
1            2   67    1  asymptomatic     160   286    0        2
108
2            3   67    1  asymptomatic     120   229    0        2
129
3            4   37    1    nonanginal     130   250    0        0
187
4            5   41    0    nontypical     130   204    0        2
172

   ExAng  Oldpeak  Slope   Ca        Thal  AHD
0      0      2.3      3  0.0       fixed   No
1      1      1.5      2  3.0      normal  Yes
2      1      2.6      2  2.0  reversable  Yes
3      0      3.5      3  0.0      normal   No
4      0      1.4      1  0.0      normal   No
```

a) Find Shape of Data

```
df.shape #303, 15
```

```
(303, 15)
```

b) Find Missing Values

```
df.isnull().sum()
```

```
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
```

```
df.count()
```

```
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

```
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
```

```
df.dtypes
```

```
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

```
df==0
```

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

     MaxHR  ExAng  Oldpeak  Slope     Ca   Thal    AHD
0    False   True    False  False   True  False  False
1    False  False    False  False  False  False  False
2    False  False    False  False  False  False  False
3    False   True    False  False   True  False  False
4    False   True    False  False   True  False  False
```

```
..     ...    ...      ...    ...    ...    ...    ...
298  False   True    False  False   True  False  False
299  False   True    False  False  False  False  False
300  False  False    False  False  False  False  False
301  False   True     True  False  False  False  False
302  False   True     True  False  False  False  False

[303 rows x 15 columns]
```

```
df[df==0]
```

```
     Unnamed: 0  Age  Sex ChestPain  RestBP  Chol  Fbs  RestECG  MaxHR
ExAng  \
0           NaN  NaN  NaN       NaN     NaN   NaN  NaN      NaN    NaN
0.0
1           NaN  NaN  NaN       NaN     NaN   NaN  0.0      NaN    NaN
NaN
2           NaN  NaN  NaN       NaN     NaN   NaN  0.0      NaN    NaN
NaN
3           NaN  NaN  NaN       NaN     NaN   NaN  0.0      0.0    NaN
0.0
4           NaN  NaN  0.0       NaN     NaN   NaN  0.0      NaN    NaN
0.0
..          ...  ...  ...       ...     ...   ...  ...      ...    ...
...
298         NaN  NaN  NaN       NaN     NaN   NaN  0.0      0.0    NaN
0.0
299         NaN  NaN  NaN       NaN     NaN   NaN  NaN      0.0    NaN
0.0
300         NaN  NaN  NaN       NaN     NaN   NaN  0.0      0.0    NaN
NaN
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

     Oldpeak  Slope   Ca Thal  AHD
0        NaN    NaN  0.0  NaN  NaN
1        NaN    NaN  NaN  NaN  NaN
2        NaN    NaN  NaN  NaN  NaN
3        NaN    NaN  0.0  NaN  NaN
4        NaN    NaN  0.0  NaN  NaN
..       ...    ...  ...  ...  ...
298      NaN    NaN  0.0  NaN  NaN
299      NaN    NaN  NaN  NaN  NaN
300      NaN    NaN  NaN  NaN  NaN
301      0.0    NaN  NaN  NaN  NaN
302      0.0    NaN  NaN  NaN  NaN

[303 rows x 15 columns]
```

```
(df == 0).sum()
```

```
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

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

```
54.43894389438944
```

```
df.Age.mean()
```

```
54.43894389438944
```

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

```
df.columns
```

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

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

```
#Cross validation
```

```
train,test = train_test_split(data,test_size=0.25,random_state=1)
```

```
train.shape
```

```
(227, 5)
```

```
test.shape
```

```
(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

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

```
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.])
```

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

predicted = np.concatenate((np.ones(100),np.zeros(400)))
predicted
```

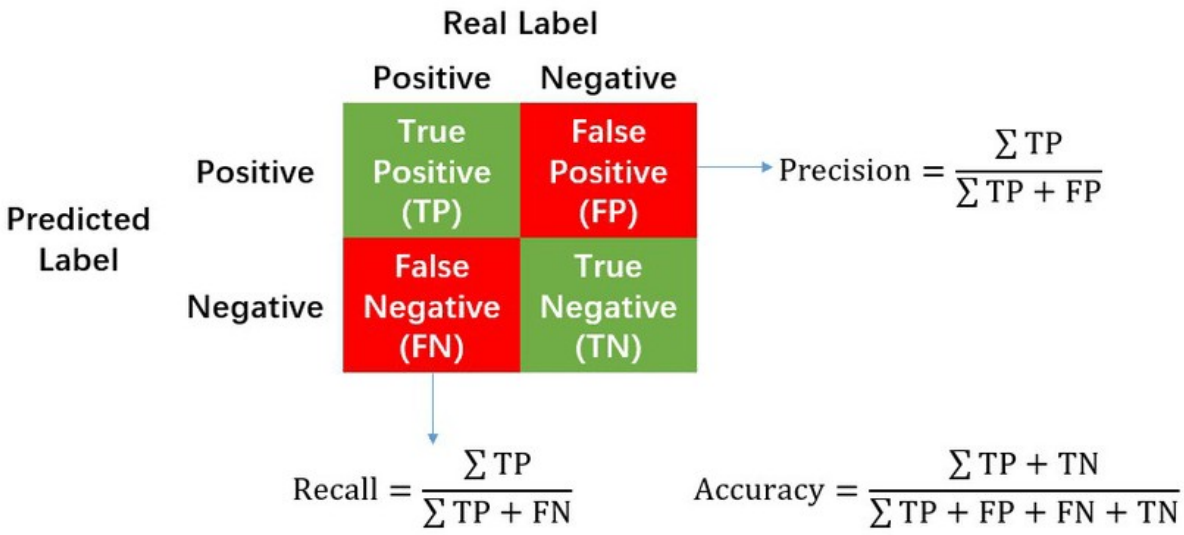
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1.,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

```python
type(predicted)
```

```
numpy.ndarray
```

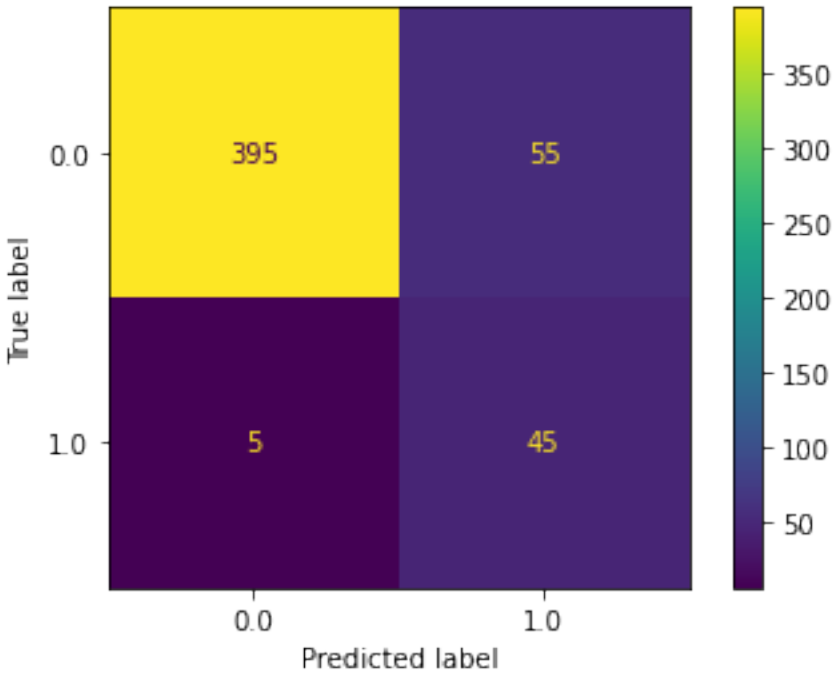
Confusion Matrix




```
from sklearn.metrics import ConfusionMatrixDisplay

ConfusionMatrixDisplay.from_predictions(actual,predicted)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f133ed13d10>
```



```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

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
```

```
accuracy_score(actual,predicted)
```

```
0.88
```