all librarys

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

Data set upload

```
df=pd.read_csv('diabetes2.csv')
```

```
df.head(15)
```

```
    Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
\
0             6      148             72             35        0  33.6

1             1       85             66             29        0  26.6

2             8      183             64              0        0  23.3

3             1       89             66             23       94  28.1

4             0      137             40             35      168  43.1

5             5      116             74              0        0  25.6

6             3       78             50             32       88  31.0

7            10      115              0              0        0  35.3

8             2      197             70             45      543  30.5

9             8      125             96              0        0   0.0

10            4      110             92              0        0  37.6

11           10      168             74              0        0  38.0

12           10      139             80              0        0  27.1

13            1      189             60             23      846  30.1

14            5      166             72             19      175  25.8
```

```
    DiabetesPedigreeFunction  Age  Outcome
0                      0.627   50        1
1                      0.351   31        0
2                      0.672   32        1
3                      0.167   21        0
4                      2.288   33        1
5                      0.201   30        0
6                      0.248   26        1
7                      0.134   29        0
8                      0.158   53        1
9                      0.232   54        1
10                     0.191   30        0
11                     0.537   34        1
12                     1.441   57        0
13                     0.398   59        1
14                     0.587   51        1
```

Data information

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
df.describe()
```

```
       Pregnancies     Glucose  BloodPressure  SkinThickness
Insulin  \
count   768.000000  768.000000     768.000000     768.000000
768.000000
mean      3.845052  120.894531      69.105469      20.536458
79.799479
std       3.369578   31.972618      19.355807      15.952218
115.244002
min       0.000000    0.000000       0.000000       0.000000
0.000000
```

```
25%       1.000000   99.000000      62.000000       0.000000
0.000000
50%       3.000000  117.000000      72.000000      23.000000
30.500000
75%       6.000000  140.250000      80.000000      32.000000
127.250000
max      17.000000  199.000000     122.000000      99.000000
846.000000
```

```
              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

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

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
df.duplicated().sum()
```

```
0
```

if duplicate vals are thare, droping duplicate data

```
df.drop_duplicates(inplace = True)
```

```
print("total no of rows :: {} ".format(len(df)))
print("total no of rows missing Pregnancies :: {}
".format(len(df.loc[df['Pregnancies'] == 0])))
print("total no of rows missing glucose :: {}
".format(len(df.loc[df['Glucose'] == 0])))
print("total no of rows missing bp :: {}
".format(len(df.loc[df['BloodPressure'] == 0])))
print("total no of rows missing insulin :: {}
".format(len(df.loc[df['Insulin'] == 0])))
print("total no of rows missing SkinThickness :: {}
".format(len(df.loc[df['SkinThickness'] == 0])))
```

```
print("total no of rows missing DiabetesPedigreeFunction :: {}
".format(len(df.loc[df['DiabetesPedigreeFunction'] == 0])))
print("total no of rows missing bmi :: {}
".format(len(df.loc[df['BMI'] == 0])))
print("total no of rows missing age :: {}
".format(len(df.loc[df['Age'] == 0])))
```

```
total no of rows :: 768
total no of rows missing Pregnancies :: 111
total no of rows missing glucose :: 5
total no of rows missing bp :: 35
total no of rows missing insulin :: 374
total no of rows missing SkinThickness :: 227
total no of rows missing DiabetesPedigreeFunction :: 0
total no of rows missing bmi :: 11
total no of rows missing age :: 0
```

```
# ignore last "Outcom" column
df.iloc[:,:-1].columns
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age'],
      dtype='object')
```

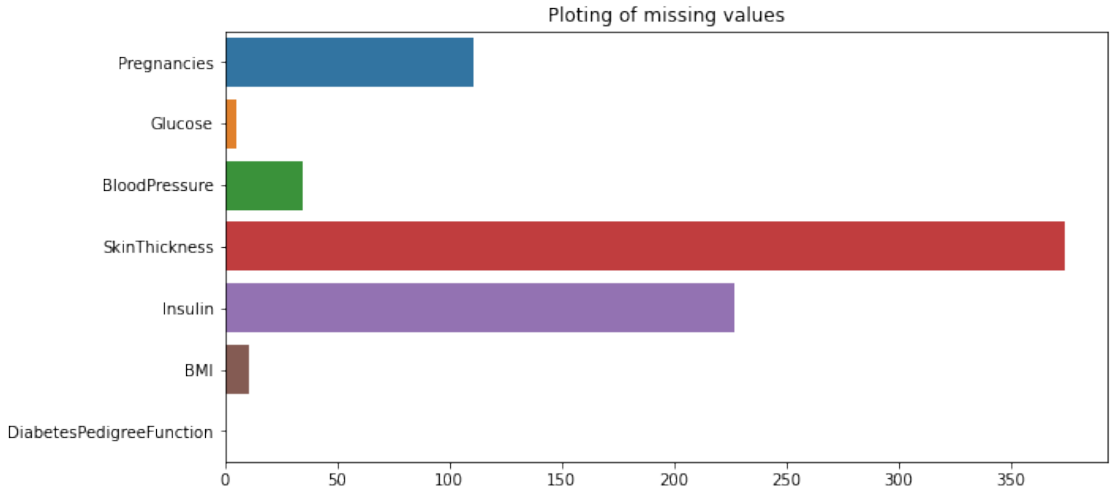
```
 len(df.loc[df['Pregnancies'] == 0])
```

```
111
```

plotting missing values

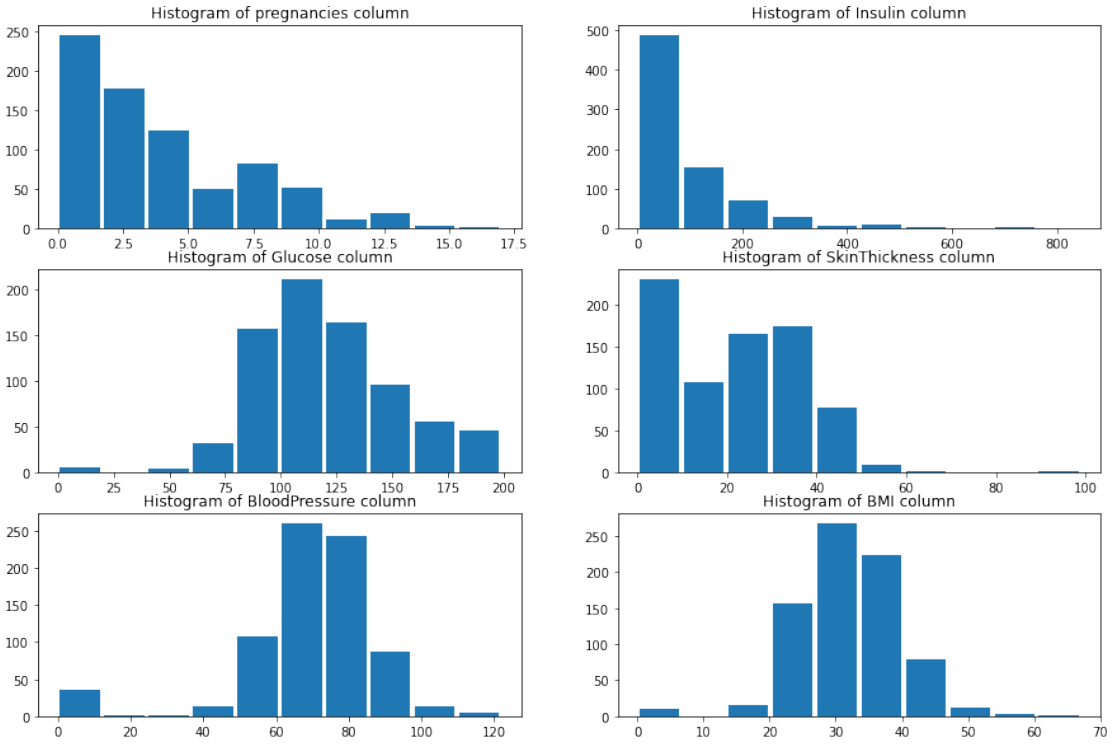
```
fig = plt.figure(figsize = (10,5))
plt.title("Ploting of missing values")
sns.barplot([ len(df.loc[df['Pregnancies'] == 0]),
              len(df.loc[df['Glucose'] == 0]),
              len(df.loc[df['BloodPressure'] == 0]),
              len(df.loc[df['Insulin'] == 0]),
              len(df.loc[df['SkinThickness'] == 0]),
              len(df.loc[df['BMI'] == 0]),
              len(df.loc[df['Age'] == 0])
            ], df.iloc[:,:-2].columns)
plt.show()
```

```
C:\Users\CREATOR\AppData\Local\Programs\Python\Python310\lib\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the following
variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



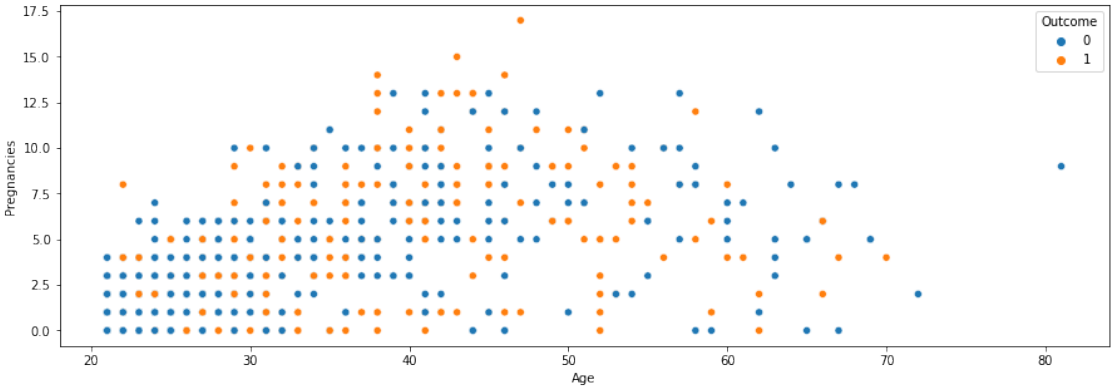
Histograms of colunms

```
fig , s= plt.subplots(3,2, figsize = (15,10))
s[0][0].set_title("Histogram of pregnancies column")
s[1][0].set_title("Histogram of Glucose column")
s[2][0].set_title("Histogram of BloodPressure column")
s[0][1].set_title("Histogram of Insulin column")
s[1][1].set_title("Histogram of SkinThickness column")
s[2][1].set_title("Histogram of BMI column")

s[0][0].hist(df['Pregnancies'], rwidth = 0.9)
s[1][0].hist(df['Glucose'], rwidth = 0.9)
s[2][0].hist(df['BloodPressure'], rwidth = 0.9)
s[0][1].hist(df['Insulin'] ,rwidth = 0.9)
s[1][1].hist(df['SkinThickness'],rwidth = 0.9)
s[2][1].hist(df['BMI'], rwidth = 0.9)
plt.show()
```



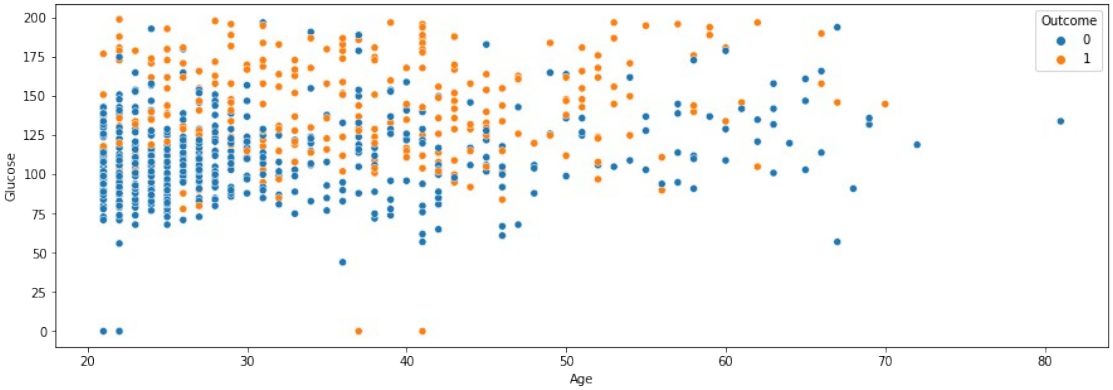
plot of age | pregnancies

```
plt.figure(figsize=(15,5))
sns.scatterplot(x= 'Age',y= 'Pregnancies', hue = 'Outcome', data = df)
plt.show()
```



plot of age | Glucose

```
plt.figure(figsize=(15,5))
sns.scatterplot(x= 'Age',y= 'Glucose', hue = 'Outcome', data = df)
plt.show()
```



correlated metrics

```
cdf = df.corr()                          # correlated metrics
top_corr = cdf.index

top_corr

Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')

cdf

                          Pregnancies   Glucose  BloodPressure
SkinThickness  \
Pregnancies                  1.000000  0.129459       0.141282      -
0.081672
Glucose                      0.129459  1.000000       0.152590
0.057328
BloodPressure                0.141282  0.152590       1.000000
0.207371
SkinThickness               -0.081672  0.057328       0.207371
1.000000
Insulin                     -0.073535  0.331357       0.088933
0.436783
BMI                          0.017683  0.221071       0.281805
0.392573
DiabetesPedigreeFunction    -0.033523  0.137337       0.041265
0.183928
Age                          0.544341  0.263514       0.239528      -
0.113970
Outcome                      0.221898  0.466581       0.065068
0.074752

                            Insulin       BMI  DiabetesPedigreeFunction
\
Pregnancies               -0.073535  0.017683                 -0.033523
```

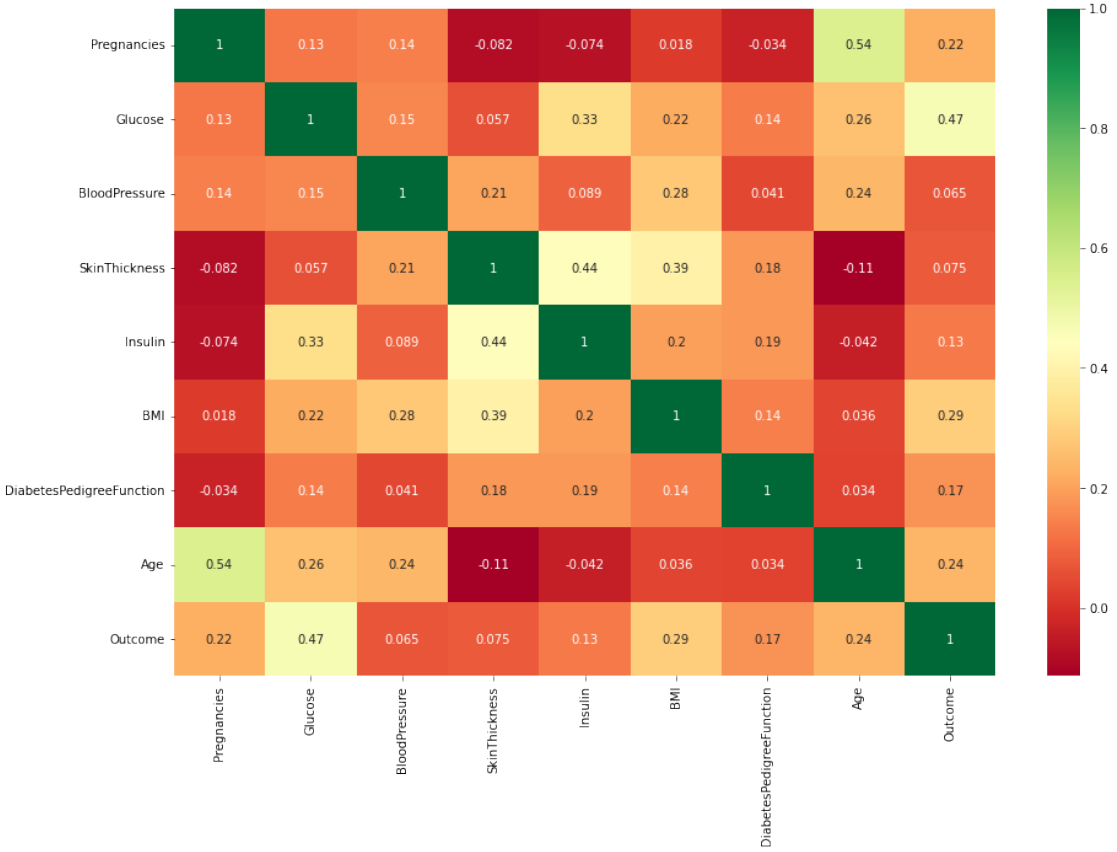
```
Glucose                    0.331357  0.221071                  0.137337

BloodPressure              0.088933  0.281805                  0.041265

SkinThickness              0.436783  0.392573                  0.183928

Insulin                    1.000000  0.197859                  0.185071

BMI                        0.197859  1.000000                  0.140647

DiabetesPedigreeFunction   0.185071  0.140647                  1.000000

Age                       -0.042163  0.036242                  0.033561

Outcome                    0.130548  0.292695                  0.173844
```

```
                              Age   Outcome
Pregnancies                0.544341  0.221898
Glucose                    0.263514  0.466581
BloodPressure              0.239528  0.065068
SkinThickness             -0.113970  0.074752
Insulin                   -0.042163  0.130548
BMI                        0.036242  0.292695
DiabetesPedigreeFunction   0.033561  0.173844
Age                        1.000000  0.238356
Outcome                    0.238356  1.000000
```

correlated heatmap

```
plt.figure(figsize = (15,10))
# annot is used to show each values
# cmap is used for color map on the graph
sns.heatmap(cdf, annot = True, cmap = 'RdYlGn')
```

```
<AxesSubplot:>
```




```
# features set (independent data(x), dependent data(y))
x = df.drop('Outcome',axis=1)
y = df['Outcome']
```

Scaling data with Robust Scaler

```
from sklearn.preprocessing import RobustScaler

rs= RobustScaler()

sx=rs.fit_transform(x)

col=[df.iloc[:,:-1].columns]
col
```

```
[Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age'],
      dtype='object')]
```

Convert into data frame

```
frame_sx=pd.DataFrame(sx,columns=col)
frame_sx
```

```
    Pregnancies   Glucose BloodPressure SkinThickness   Insulin
BMI  \
0           0.6  0.751515      0.000000       0.37500 -0.239686
0.172043
1          -0.4 -0.775758     -0.333333       0.18750 -0.239686 -
0.580645
2           1.0  1.600000     -0.444444      -0.71875 -0.239686 -
0.935484
3          -0.4 -0.678788     -0.333333       0.00000  0.499018 -
0.419355
4          -0.6  0.484848     -1.777778       0.37500  1.080550
1.193548
..          ...       ...           ...           ...       ...
...
763         1.4 -0.387879      0.222222       0.78125  1.174853
0.096774
764        -0.2  0.121212     -0.111111       0.12500 -0.239686
0.516129
765         0.4  0.096970      0.000000       0.00000  0.640472 -
0.623656
766        -0.4  0.218182     -0.666667      -0.71875 -0.239686 -
0.204301
767        -0.4 -0.581818     -0.111111       0.25000 -0.239686 -
0.172043

    DiabetesPedigreeFunction       Age
0                   0.665359  1.235294
1                  -0.056209  0.117647
2                   0.783007  0.176471
3                  -0.537255 -0.470588
4                   5.007843  0.235294
..                       ...       ...
763                -0.526797  2.000000
764                -0.084967 -0.117647
765                -0.333333  0.058824
766                -0.061438  1.058824
767                -0.150327 -0.352941

[768 rows x 8 columns]
```

assaining dependant variable and split data

```
x=frame_sx
y=df["Outcome"]

# slit data

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =
0.3, random_state = 11)
```

```
x_train.shape, y_train.shape  , x_test.shape, y_test.shape
```

```
((537, 8), (537,), (231, 8), (231,))
```

calling model and set peramiters

```
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n_estimators = 11, random_state = 11)
# n_estimators ==> determine the no of decision trees
# random_state ==> It ensures that the splits that you generate are
reproducible.

# train the model
rf.fit( x_train , y_train)
```

```
C:\Users\CREATOR\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\utils\validation.py:1688: FutureWarning: Feature
names only support names that are all strings. Got feature names with
dtypes: ['tuple']. An error will be raised in 1.2.
  warnings.warn(
```

```
RandomForestRegressor(n_estimators=11, random_state=11)
```

```
# Predicting values from the model
y_pred = rf.predict(x_test)
y_pred
```

```
C:\Users\CREATOR\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\utils\validation.py:1688: FutureWarning: Feature
names only support names that are all strings. Got feature names with
dtypes: ['tuple']. An error will be raised in 1.2.
  warnings.warn(
```

```
array([0.        , 0.09090909, 0.09090909, 0.09090909, 0.        ,
       0.        , 0.09090909, 0.09090909, 0.27272727, 0.36363636,
       0.36363636, 0.        , 0.        , 0.72727273, 0.36363636,
       0.        , 0.09090909, 0.72727273, 0.81818182, 1.        ,
       0.36363636, 0.18181818, 0.81818182, 0.        , 0.27272727,
       0.27272727, 0.        , 0.09090909, 0.90909091, 0.36363636,
       0.        , 1.        , 0.09090909, 0.        , 0.36363636,
       0.45454545, 0.63636364, 0.63636364, 0.18181818, 0.36363636,
       0.18181818, 0.        , 0.27272727, 0.        , 0.45454545,
       0.36363636, 0.63636364, 0.36363636, 0.36363636, 0.        ,
       0.72727273, 0.        , 0.54545455, 0.        , 0.        ,
       0.63636364, 0.36363636, 0.18181818, 0.72727273, 0.09090909,
       0.63636364, 0.27272727, 0.36363636, 0.        , 0.18181818,
       0.18181818, 0.09090909, 0.63636364, 0.18181818, 0.09090909,
       0.63636364, 0.63636364, 0.81818182, 0.54545455, 0.27272727,
       0.09090909, 0.        , 0.81818182, 0.27272727, 0.09090909,
       0.36363636, 0.81818182, 0.18181818, 0.18181818, 0.09090909,
```

```
       0.        , 0.        , 0.45454545, 0.        , 0.45454545,
       0.45454545, 0.90909091, 0.27272727, 1.        , 0.18181818,
       0.36363636, 0.54545455, 0.54545455, 0.09090909, 0.54545455,
       0.09090909, 0.72727273, 0.72727273, 0.27272727, 0.54545455,
       0.63636364, 0.09090909, 0.27272727, 0.        , 0.90909091,
       0.63636364, 0.81818182, 0.09090909, 0.90909091, 0.        ,
       0.36363636, 0.27272727, 0.36363636, 0.        , 0.45454545,
       0.09090909, 0.36363636, 0.        , 0.09090909, 0.        ,
       0.        , 0.09090909, 0.81818182, 0.        , 0.72727273,
       0.        , 0.09090909, 0.81818182, 0.09090909, 0.        ,
       0.        , 0.09090909, 0.63636364, 0.36363636, 0.72727273,
       0.45454545, 0.27272727, 0.18181818, 0.09090909, 0.09090909,
       0.09090909, 0.09090909, 0.81818182, 0.        , 0.18181818,
       0.09090909, 0.90909091, 0.        , 0.09090909, 0.63636364,
       0.        , 0.72727273, 1.        , 0.        , 0.63636364,
       0.36363636, 0.        , 0.27272727, 0.18181818, 0.36363636,
       0.        , 0.09090909, 0.45454545, 0.63636364, 0.90909091,
       0.63636364, 0.36363636, 0.        , 0.09090909, 0.09090909,
       0.        , 0.27272727, 0.18181818, 1.        , 0.63636364,
       0.09090909, 0.18181818, 0.09090909, 0.        , 0.36363636,
       0.        , 0.18181818, 0.18181818, 0.81818182, 0.18181818,
       0.27272727, 0.09090909, 0.54545455, 0.27272727, 0.90909091,
       0.18181818, 0.63636364, 0.27272727, 0.54545455, 0.36363636,
       0.36363636, 0.45454545, 0.54545455, 0.        , 0.36363636,
       0.        , 0.27272727, 0.        , 0.27272727, 0.72727273,
       0.72727273, 0.        , 0.81818182, 0.63636364, 0.        ,
       0.18181818, 0.        , 0.        , 0.09090909, 0.27272727,
       0.54545455, 0.81818182, 0.45454545, 0.        , 0.27272727,
       0.54545455, 0.09090909, 0.72727273, 0.45454545, 0.        ,
       0.45454545])
```

converting predicted vals between 0 and 1

```
y_pred = np.array([0 if i < 0.5 else 1 for i in y_pred])
y_pred
```

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

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

```
from sklearn.metrics import confusion_matrix, accuracy_score
acc = accuracy_score(y_test, y_pred)
acc
```

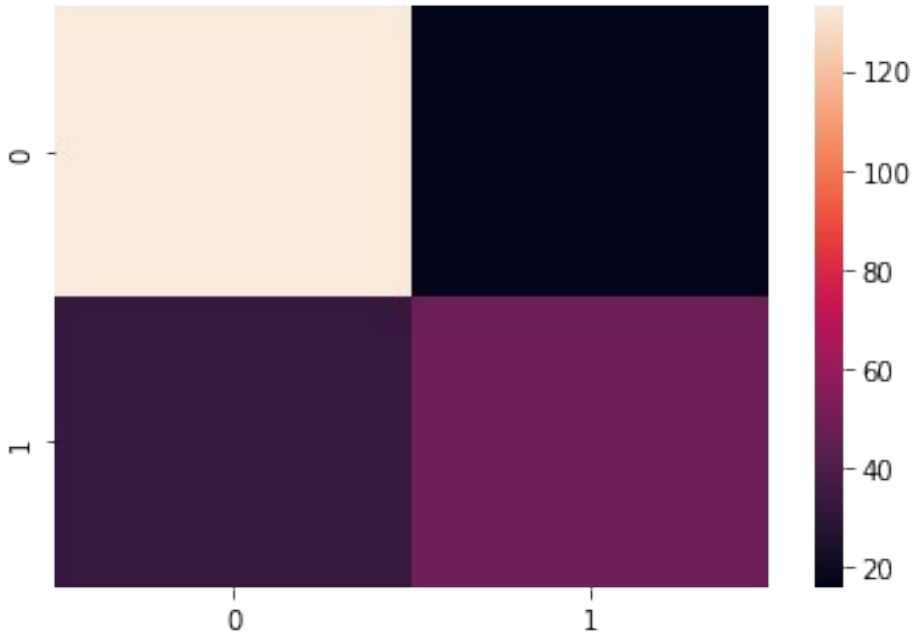
```
0.7878787878787878
```

```
m = confusion_matrix(y_test, y_pred)
m
```

```
array([[133,  16],
       [ 33,  49]], dtype=int64)
```

```
sns.heatmap(m)
```

```
<AxesSubplot:>
```




```
from sklearn.metrics import classification_report
class_report = classification_report(y_test, y_pred)
class_report
```

```
'              precision    recall  f1-score   support\n\n           0
0.80      0.89      0.84       149\n           1       0.75      0.60
0.67        82\n\n    accuracy                           0.79
```

```
231\n   macro avg       0.78      0.75      0.76       231\nweighted
avg       0.78      0.79      0.78       231\n'

print("confusion matrix :: {} \n\n Accuracy = {} \n\n classification
report :: \n{}".format(m,acc,class_report))

confusion matrix :: [[133  16]
 [ 33  49]]

 Accuracy = 0.7878787878787878

 classification report ::
              precision    recall  f1-score   support

           0       0.80      0.89      0.84       149
           1       0.75      0.60      0.67        82

    accuracy                           0.79       231
   macro avg       0.78      0.75      0.76       231
weighted avg       0.78      0.79      0.78       231
```