```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn.impute import SimpleImputer
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
import sklearn.cluster as cluster
import sklearn.metrics as metrics
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report,confusion_matrix,r2_score
plt.style.use ("ggplot")
```

```
#process data
```

```
df = pd.read_csv(r'heart.csv') #getting from keggal database
df
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0
...	...	...	...	...	...	...	...	...	...	...	...	...
913	45	M	TA	110	264	0	Normal	132	N	1.2	Flat	1
914	68	M	ASY	144	193	1	Normal	141	N	3.4	Flat	1
915	57	M	ASY	130	131	0	Normal	115	Y	1.2	Flat	1
916	57	F	ATA	130	236	0	LVH	174	N	0.0	Flat	1
917	38	M	NAP	138	175	0	Normal	173	N	0.0	Up	0

918 rows × 12 columns

```
df.columns
```

```
Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS',
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
```

```
       'HeartDisease'],
      dtype='object')
```

```
df.isna().sum() # to check if we have any null values
```

```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

```
df = df.drop_duplicates()  #to drop all duplicates
```

```
df
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0
...	...	...	...	...	...	...	...	...	...	...	...	...
913	45	M	TA	110	264	0	Normal	132	N	1.2	Flat	1
914	68	M	ASY	144	193	1	Normal	141	N	3.4	Flat	1
915	57	M	ASY	130	131	0	Normal	115	Y	1.2	Flat	1
916	57	F	ATA	130	236	0	LVH	174	N	0.0	Flat	1
917	38	M	NAP	138	175	0	Normal	173	N	0.0	Up	0

918 rows × 12 columns

```
df.nunique() # to let us know how many differet kind of information we have in each column
```

```
Age               50
Sex                2
ChestPainType      4
RestingBP         67
Cholesterol      222
```

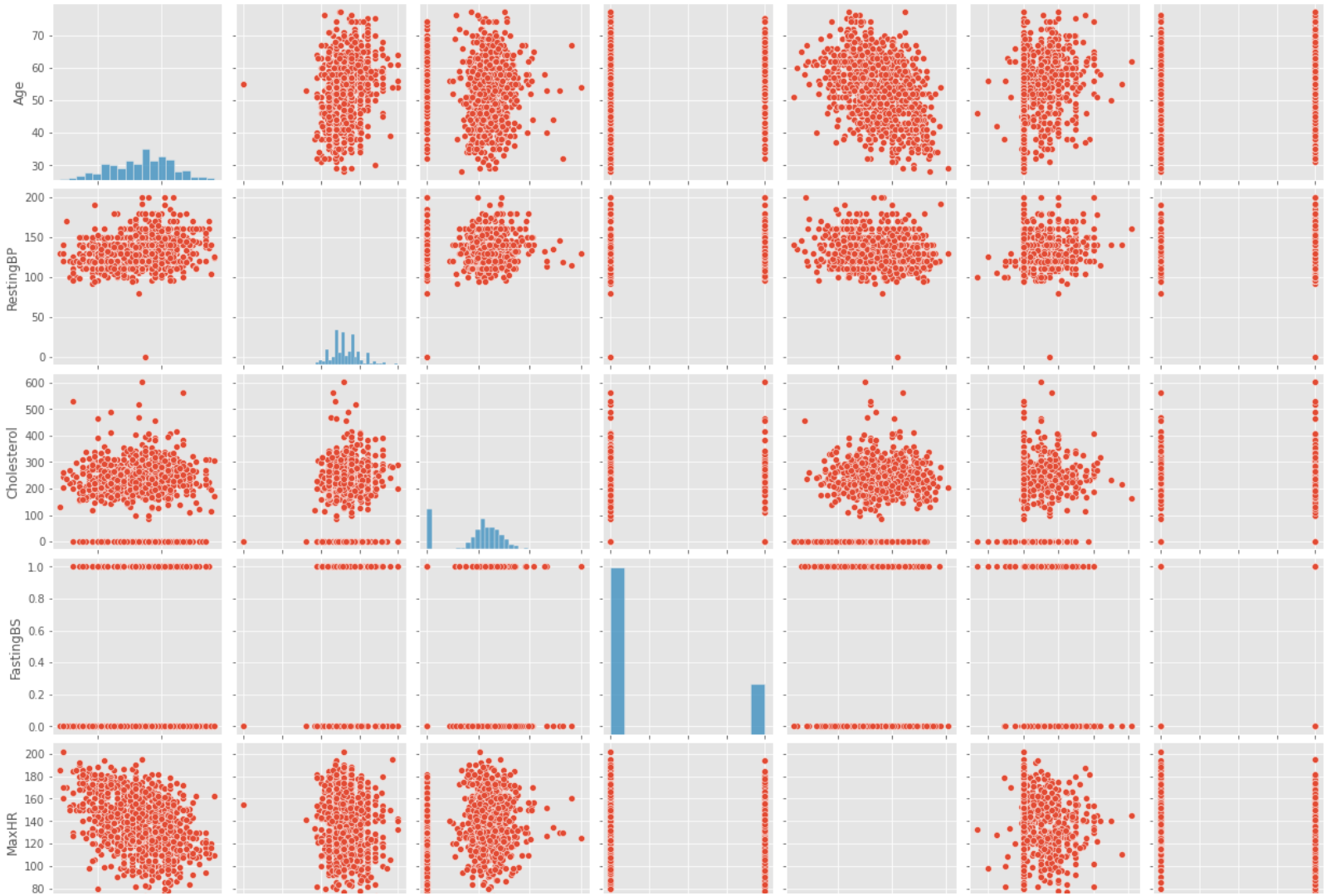
```
FastingBS           2
RestingECG          3
MaxHR             119
ExerciseAngina      2
Oldpeak            53
ST_Slope            3
HeartDisease        2
dtype: int64
```

```
df.corr() # the correlation between the attributes
```

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
Age	1.000000	0.254399	-0.095282	0.198039	-0.382045	0.258612	0.282039
RestingBP	0.254399	1.000000	0.100893	0.070193	-0.112135	0.164803	0.107589
Cholesterol	-0.095282	0.100893	1.000000	-0.260974	0.235792	0.050148	-0.232741
FastingBS	0.198039	0.070193	-0.260974	1.000000	-0.131438	0.052698	0.267291
MaxHR	-0.382045	-0.112135	0.235792	-0.131438	1.000000	-0.160691	-0.400421
Oldpeak	0.258612	0.164803	0.050148	0.052698	-0.160691	1.000000	0.403951
HeartDisease	0.282039	0.107589	-0.232741	0.267291	-0.400421	0.403951	1.000000

```
sn.pairplot(df) # correlation illustration
```

```
<seaborn.axisgrid.PairGrid at 0x7fd2adff5940>
```



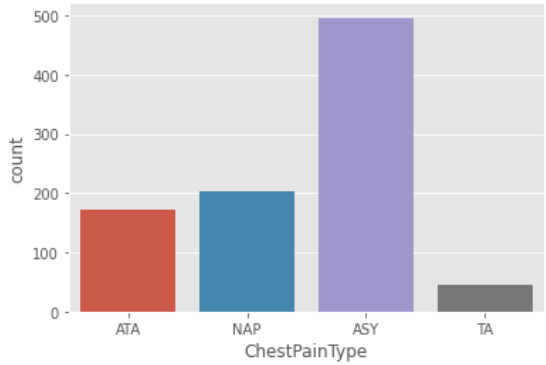

```
df.info() #an overview to our data
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
```

```
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(6), object(5)
memory usage: 125.5+ KB
```

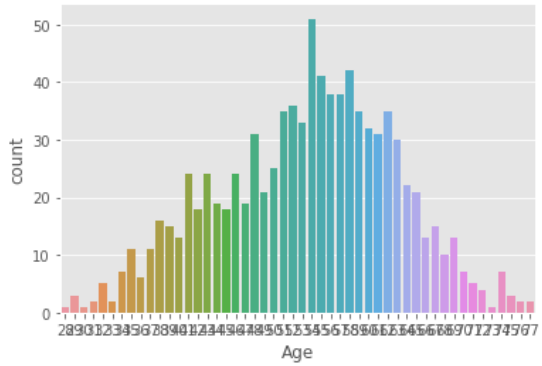
```
sn.countplot(df['ChestPainType'])
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From v
  warnings.warn(
<AxesSubplot:xlabel='ChestPainType', ylabel='count'>
```



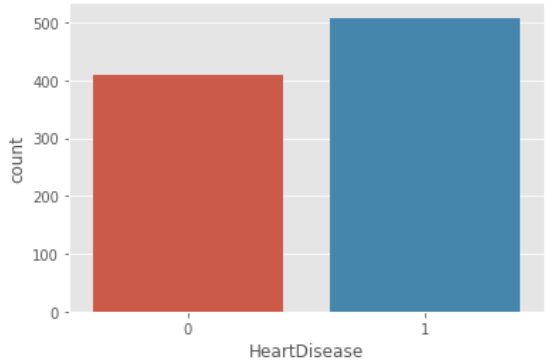

```
sn.countplot(df['Age'])
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From v
  warnings.warn(
<AxesSubplot:xlabel='Age', ylabel='count'>
```



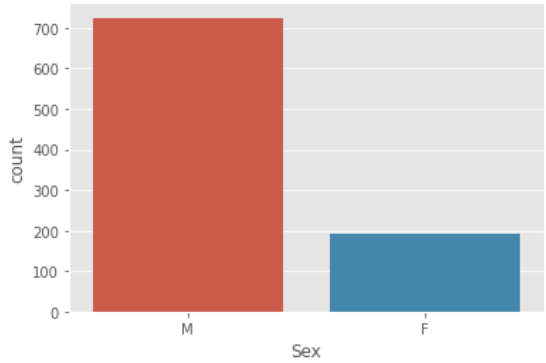
```
sn.countplot(df['HeartDisease'])
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From v
  warnings.warn(
<AxesSubplot:xlabel='HeartDisease', ylabel='count'>
```




```
sn.countplot(df['Sex'])
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From v
  warnings.warn(
<AxesSubplot:xlabel='Sex', ylabel='count'>
```

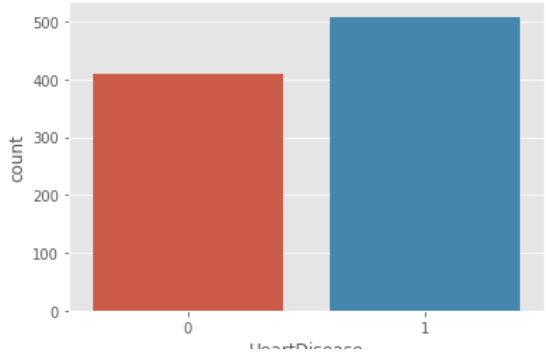



```
df['HeartDisease'].value_counts()
```

```
1    508
0    410
Name: HeartDisease, dtype: int64
```

```
sn.countplot(df['HeartDisease'])
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From v
  warnings.warn(
<AxesSubplot:xlabel='HeartDisease', ylabel='count'>
```




```
df['Sex'].value_counts()
```

```
M    725
F    193
Name: Sex, dtype: int64
```

```
df['Sex']
```

```
0      M
1      F
2      M
3      F
4      M
      ..
913    M
914    M
915    M
916    F
917    M
Name: Sex, Length: 918, dtype: object
```

```
#convert object types into numercal values
```

```
df.Sex = df.Sex.map( {'M':0 , 'F':1} )
```

```
df.Sex
```

```
0      0
1      1
2      0
3      1
4      0
      ..
913    0
914    0
915    0
```

```
916    1
917    0
Name: Sex, Length: 918, dtype: int64
```

```
df.ChestPainType = df.ChestPainType.map( {'TA':0 , 'ATA':1, 'NAP':2, 'ASY':3} )
```

```
df.RestingECG = df.RestingECG.map( {'Normal':0 , 'ST':1, 'LVH':2} )
```

```
df.ExerciseAngina = df.ExerciseAngina.map( {'N':0 , 'Y':1} )
```

```
df.ST_Slope = df.ST_Slope.map( {'Up':1 , 'Flat':0, 'Down':-1} )
```

```
df.describe()
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	0.210240	2.251634	132.396514	198.799564	0.233115	0.603486	136.809368	0.404139	0.887364	0.361656
std	9.432617	0.407701	0.931031	18.514154	109.384145	0.423046	0.805968	25.460334	0.490992	1.066570	0.607056
min	28.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	60.000000	0.000000	-2.600000	-1.000000
25%	47.000000	0.000000	2.000000	120.000000	173.250000	0.000000	0.000000	120.000000	0.000000	0.000000	0.000000
50%	54.000000	0.000000	3.000000	130.000000	223.000000	0.000000	0.000000	138.000000	0.000000	0.600000	0.000000
75%	60.000000	0.000000	3.000000	140.000000	267.000000	0.000000	1.000000	156.000000	1.000000	1.500000	1.000000
max	77.000000	1.000000	3.000000	200.000000	603.000000	1.000000	2.000000	202.000000	1.000000	6.200000	1.000000

```
plt.figure(figsize=(15, 10))
sn.heatmap(df)
```

```
<AxesSubplot:>
```


```
# Support vector machine
```



```
x = df.iloc[:,:-1].values #getting our data from the dataframe
x
```

```
array([[40. ,  0. ,  1. , ...,  0. ,  0. ,  1. ],
       [49. ,  1. ,  2. , ...,  0. ,  1. ,  0. ],
       [37. ,  0. ,  1. , ...,  0. ,  0. ,  1. ],
       ...,
       [57. ,  0. ,  3. , ...,  1. ,  1.2,  0. ],
       [57. ,  1. ,  1. , ...,  0. ,  0. ,  0. ],
       [38. ,  0. ,  2. , ...,  0. ,  0. ,  1. ]])
```

```
y = df.iloc[:,-1].values
y
```

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

```
SC = StandardScaler() #standarise the values to work in unsupervised method
x[:,0:] = SC.fit_transform(x[:,0:])
x[:,0:]
```

```
array([[-1.4331398 , -0.51595242, -1.34508565, ..., -0.8235563 ,
        -0.83243239,  1.05211381],
       [-0.47848359,  1.93816322, -0.27042192, ..., -0.8235563 ,
         0.10566353, -0.59607813],
       [-1.75135854, -0.51595242, -1.34508565, ..., -0.8235563 ,
        -0.83243239,  1.05211381],
       ...,
       [ 0.37009972, -0.51595242,  0.80424181, ...,  1.21424608,
         0.29328271, -0.59607813],
       [ 0.37009972,  1.93816322, -1.34508565, ..., -0.8235563 ,
        -0.83243239, -0.59607813],
       [-1.64528563, -0.51595242, -0.27042192, ..., -0.8235563 ,
        -0.83243239,  1.05211381]])
```

```
#defining our new data frame after standarising and show our new heatmap
df = pd.DataFrame(x,columns = ['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope'])
#plt.figure(figsize=(20, 10))
#sn.heatmap(df)
```

```
X = df.values
X
```

```
array([[-1.4331398 , -0.51595242, -1.34508565, ..., -0.8235563 ,
        -0.83243239,  1.05211381],
       [-0.47848359,  1.93816322, -0.27042192, ..., -0.8235563 ,
         0.10566353, -0.59607813],
```

```
              [-1.75135854, -0.51595242, -1.34508565, ..., -0.8235563 ,
               -0.83243239,  1.05211381],
              ...,
              [ 0.37009972, -0.51595242,  0.80424181, ...,  1.21424608,
                0.29328271, -0.59607813],
              [ 0.37009972,  1.93816322, -1.34508565, ..., -0.8235563 ,
               -0.83243239, -0.59607813],
              [-1.64528563, -0.51595242, -0.27042192, ..., -0.8235563 ,
               -0.83243239,  1.05211381]])
```

```
x_train,x_test,y_train,y_test = train_test_split (X ,y ,test_size = 0.3 ,random_state = 0) #splitting the data
```

```
svc = SVC()
svc.fit(x_train, y_train) # fitting data into the support vec machine
y_pred = svc.predict(x_test)
print(y_pred)
```

```
     [1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1 0 0 0 1 0 1
      0 1 0 0 1 1 0 1 0 0 1 1 0 1 0 0 1 1 1 1 1 0 0 0 1 1 1 0 1 1 1 1 1 0 1 1 0 1
      1 1 1 0 1 1 1 1 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 0 0 0 1 1 1 1 1 1 1 0 0 1 0
      1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 0 1 0 1 0 1 1 1 1 0 1 1 0 0 1 1 0 1 0 0 1 1
      1 1 0 1 0 0 1 0 1 1 1 0 0 1 1 0 1 1 1 0 1 1 1 1 0 0 1 1 0 1 0 1 1 0 0 1 1
      1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 1 0 1 1 1 1 1 1
      1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 0 1 0 0 0 1 0 1 0 1 0 0 0 1 0 0
      1 0 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1]
```

```
#evaluation
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
     0.8695652173913043
```

```
print(classification_report(y_test,y_pred))
```

```
                   precision    recall  f1-score   support

                0       0.87      0.80      0.83       113
                1       0.87      0.92      0.89       163

         accuracy                           0.87       276
        macro avg       0.87      0.86      0.86       276
     weighted avg       0.87      0.87      0.87       276
```

```
#confusion matrix of FF, FT, TF, TT as a matrix and a heatmap
CM = confusion_matrix (y_test,y_pred)
sn.heatmap(CM)
print (CM)
```

```
[[ 90  23]
 [ 13 150]]
```

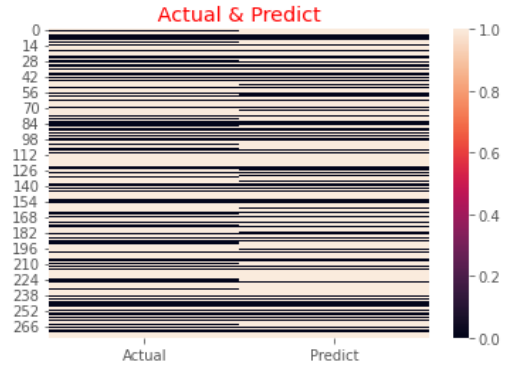

```
#illustartion to the actual and predicted values
df_comp = pd.DataFrame({'Actual':y_test , 'Predict':y_pred})
df_comp
```

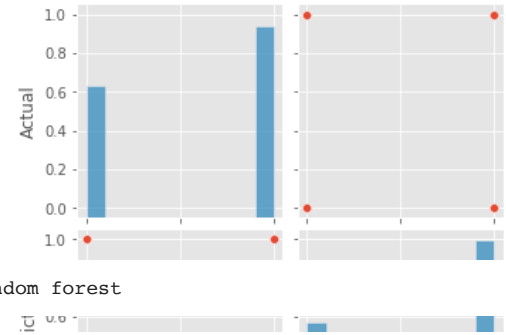
	Actual	Predict
0	1	1
1	0	1
2	1	1
3	1	1
4	0	0
...	...	...
271	1	1
272	1	1
273	1	1
274	1	1
275	1	1

276 rows × 2 columns

```
plt.title (' Actual & Predict ',color = 'r')
sn.heatmap(df_comp)
sn.pairplot(df_comp)
```

```
<seaborn.axisgrid.PairGrid at 0x7fd2a78e2970>
```





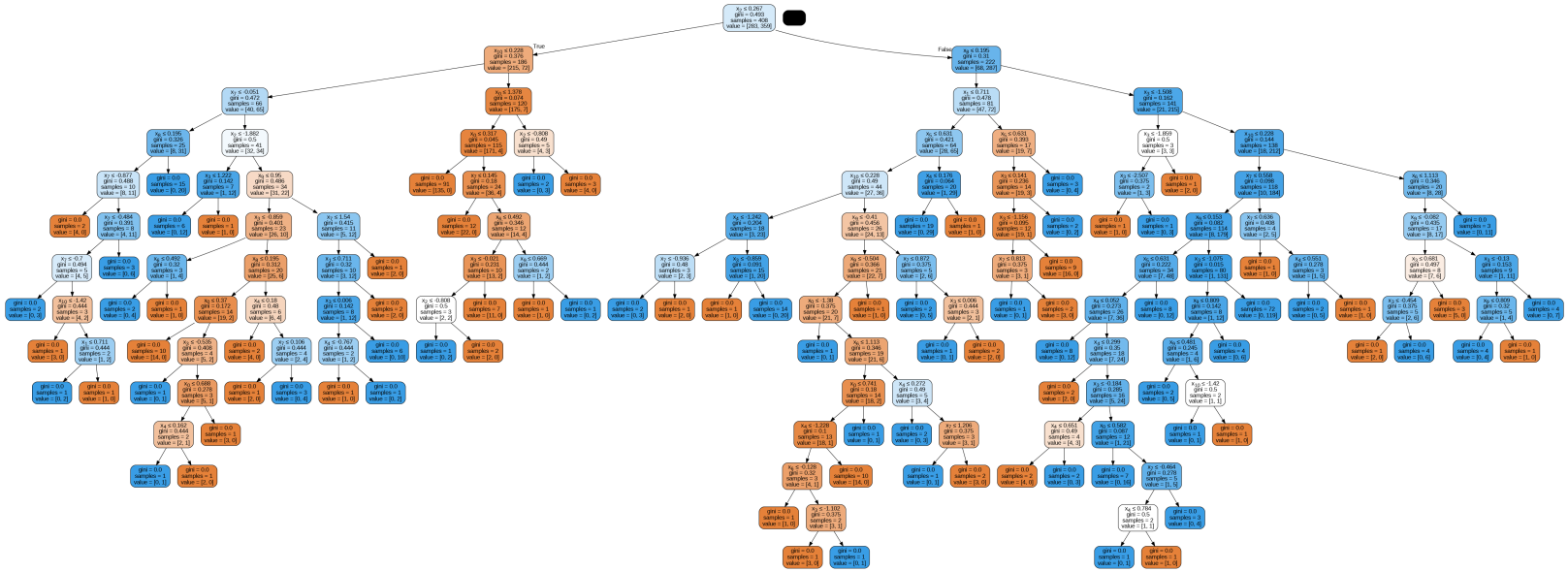

```
#random forest
```

```
RFC=RandomForestClassifier(n_estimators=10,random_state=0)
RFC.fit(x_train,y_train)
y_pred_2 = RFC.predict(x_test)
y_pred_2
```

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

```
from sklearn.tree import export_graphviz
from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
```

```
# Create a RandomForestClassifier object
RFC = RandomForestClassifier(n_estimators=10, random_state=0)

# Fit the model to the training data
RFC.fit(x_train, y_train)

# Predict the labels for the test data
y_pred_2 = RFC.predict(x_test)

# Extract a single decision tree from the forest
tree = RFC.estimators_[8]

# Export the tree to a DOT format
dot_data = StringIO()
export_graphviz(tree, out_file=dot_data, filled=True, rounded=True, special_characters=True)

# Convert the DOT data to an image
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```



```
accuracy_score(y_test, y_pred_2)
```

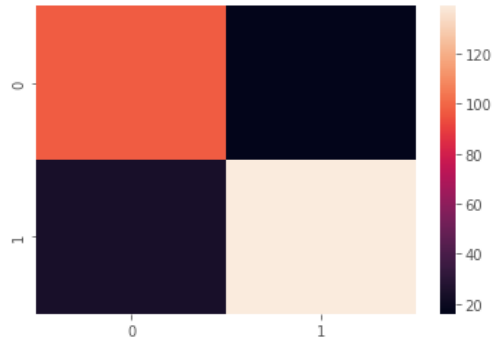
```
0.855072463768116
```

```
print(classification_report(y_test,y_pred_2))
```

```
                  precision    recall  f1-score   support

               0       0.80      0.86      0.83       113
               1       0.90      0.85      0.87       163

        accuracy                           0.86       276
       macro avg       0.85      0.86      0.85       276
    weighted avg       0.86      0.86      0.86       276
```

```
CM = confusion_matrix (y_test,y_pred_2)
sn.heatmap(CM)
print (CM)
```

```
[[ 97  16]
 [ 24 139]]
```

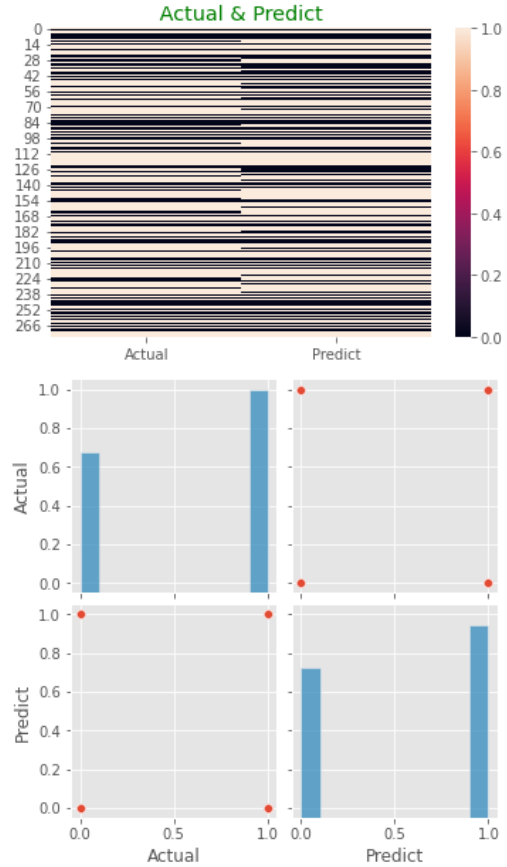



```
df_comp_2 = pd.DataFrame({'Actual':y_test , 'Predict':y_pred_2})
df_comp_2
```

	Actual	Predict
0	1	1

```
plt.title (' Actual & Predict ',color = 'g')
sn.heatmap(df_comp_2)
sn.pairplot(df_comp_2)
```

```
<seaborn.axisgrid.PairGrid at 0x7fd2a68f3220>
```




```
#knn
```

```
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X, y)
y_pred_3  = neigh.predict(x_test)
y_pred_3
```

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

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

```
accuracy_score(y_test, y_pred_3)
```

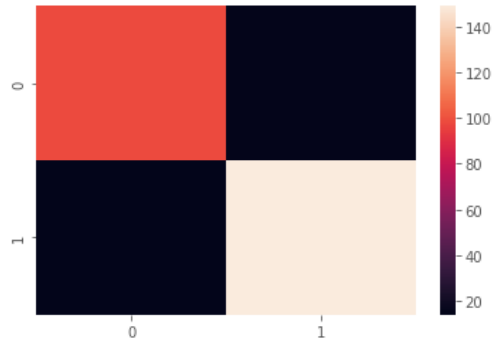
```
0.894927536231884
```

```
print(classification_report(y_test,y_pred_3))
```

```
                  precision    recall  f1-score   support

               0       0.88      0.87      0.87       113
               1       0.91      0.91      0.91       163

        accuracy                           0.89       276
       macro avg       0.89      0.89      0.89       276
    weighted avg       0.89      0.89      0.89       276
```

```
CM = confusion_matrix (y_test,y_pred_3)
sn.heatmap(CM)
print (CM)
```

```
[[ 98  15]
 [ 14 149]]
```




```
df_comp_3 = pd.DataFrame({'Actual':y_test , 'Predict':y_pred_3})
df_comp_3
```

	Actual	Predict
0	1	1
1	0	1
2	1	1
3	1	1
4	0	0
...	...	...
271	1	1
272	1	1
273	1	1
274	1	1
275	1	1

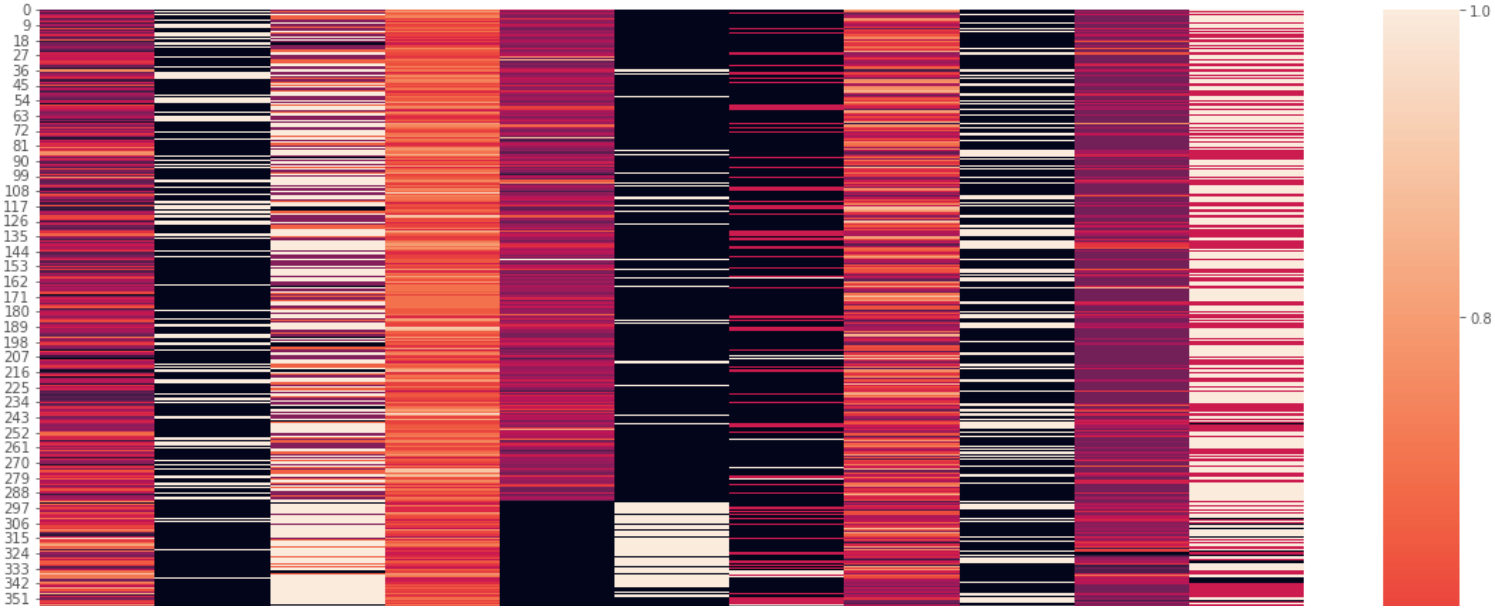
```
plt.title (' Actual & Predict ',color = 'g')
sn.heatmap(df_comp_3)
sn.pairplot(df_comp_3)
```

```
<seaborn.axisgrid.PairGrid at 0x7fd2a6f8e370>
```


```
#k-mean
```

```
scaler = MinMaxScaler()
scale = scaler.fit_transform(X, y)
df_scale = pd.DataFrame(scale, columns = ['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slop
plt.figure(figsize=(20, 20))
sn.heatmap(df_scale)
df_scale.head(5)
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope
0	0.244898	0.0	0.333333	0.70	0.479270	0.0	0.0	0.788732	0.0	0.295455	1.0
1	0.428571	1.0	0.666667	0.80	0.298507	0.0	0.0	0.676056	0.0	0.409091	0.5
2	0.183673	0.0	0.333333	0.65	0.469320	0.0	0.5	0.267606	0.0	0.295455	1.0
3	0.408163	1.0	1.000000	0.69	0.354892	0.0	0.0	0.338028	1.0	0.465909	0.5
4	0.530612	0.0	0.666667	0.75	0.323383	0.0	0.0	0.436620	0.0	0.295455	1.0




```
km=KMeans(n_clusters=2)
y_pred_4 = km.fit_predict(x_test)
y_pred_4
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value c
  warnings.warn(
array([1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0], dtype=int32)
```

```
accuracy_score(y_test, y_pred_4)
```

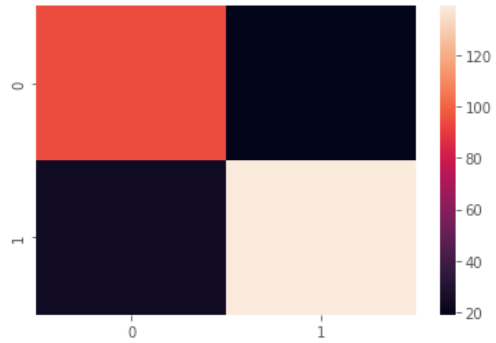
```
0.8442028985507246
```

```
print(classification_report(y_test,y_pred_4))
```

```
              precision    recall  f1-score   support

           0       0.80      0.83      0.81       113
           1       0.88      0.85      0.87       163

    accuracy                           0.84       276
   macro avg       0.84      0.84      0.84       276
weighted avg       0.85      0.84      0.84       276
```

```
CM = confusion_matrix (y_test,y_pred_4)
sn.heatmap(CM)
print (CM)
```

```
[[ 94  19]
 [ 24 139]]
```

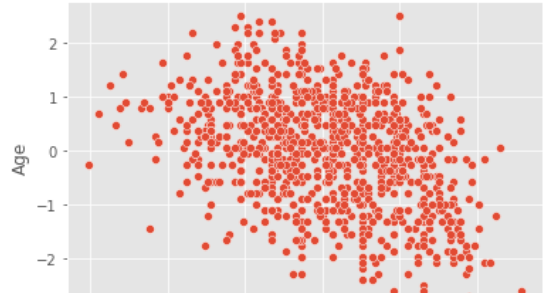



```
km.cluster_centers_
```

```
array([[-0.5297222 ,  0.5031295 , -0.57096347, -0.17531128,  0.34515883,
        -0.19056341,  0.02933825,  0.58297688, -0.77174776, -0.5319237 ,
         0.63308196],
       [ 0.3613722 , -0.37616102,  0.55258005,  0.03192961, -0.3253772 ,
         0.33182883,  0.02867239, -0.50711586,  0.54357694,  0.3888735 ,
        -0.48133059]])
```

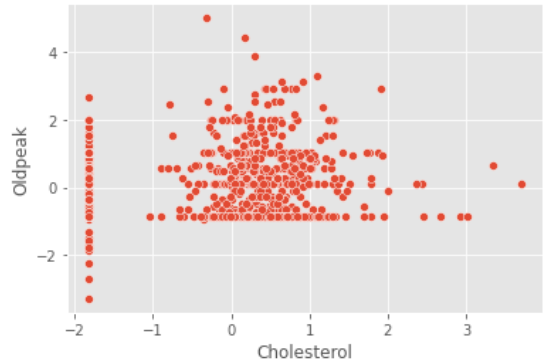
```
a= km.labels_
sn.scatterplot(x="MaxHR", y= "Age",data=df)
a
```

```
array([1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1,
       1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0], dtype=int32)
```




```
sn.scatterplot(x="Cholesterol", y= "Oldpeak",data=df)
```

```
<AxesSubplot:xlabel='Cholesterol', ylabel='Oldpeak'>
```




```
scaler = MinMaxScaler()
```

```
df.head(4)
```

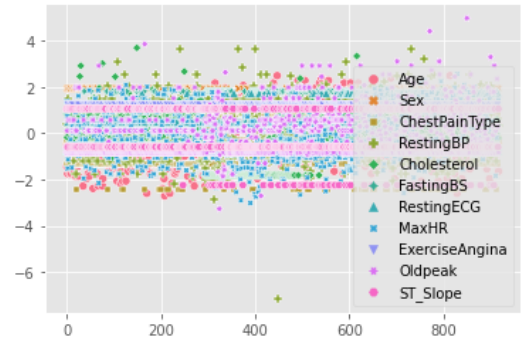
	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope
0	-1.433140	-0.515952	-1.345086	0.410909	0.825070	-0.551341	-0.749180	1.382928	-0.823556	-0.832432	1.052114

```
scale = scaler.fit_transform(df)

df_scale = pd.DataFrame(scale, columns = ['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slop
df_scale.head(5)
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope
0	0.244898	0.0	0.333333	0.70	0.479270	0.0	0.0	0.788732	0.0	0.295455	1.0
1	0.428571	1.0	0.666667	0.80	0.298507	0.0	0.0	0.676056	0.0	0.409091	0.5
2	0.183673	0.0	0.333333	0.65	0.469320	0.0	0.5	0.267606	0.0	0.295455	1.0
3	0.408163	1.0	1.000000	0.69	0.354892	0.0	0.0	0.338028	1.0	0.465909	0.5
4	0.530612	0.0	0.666667	0.75	0.323383	0.0	0.0	0.436620	0.0	0.295455	1.0

```
#PCA


sn.scatterplot(data=df)
```

```
<AxesSubplot:>
```




```
from sklearn.decomposition import PCA

pca = PCA(n_components=2) # choosing 2 for better visualisation
principalComponents = pca.fit_transform(df_scale)
pca_df = pd.DataFrame(data = principalComponents, columns = ['component 1', 'component 2'])
pca_df.head()
```

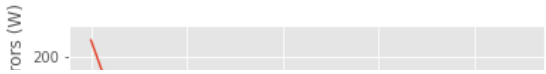
	component 1	component 2
0	-0.644167	-0.079920
1	-0.624046	-0.285940
2	-0.533714	0.009604

```
N = range(2,12)
w = []
for k in range(2,12):
    kmeans=cluster.KMeans(n_clusters=k)
    kmeans=kmeans.fit(pca_df)
    w_iter = kmeans.inertia_
    w.append(w_iter)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value c
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value c
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value c
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value c
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value c
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value c
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value c
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value c
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value c
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value c
  warnings.warn(
```

```
plt.xlabel('N')
plt.ylabel('Within-Cluster-Sum of Squared Errors (W)')
plt.plot(N,w)
```

```
[<matplotlib.lines.Line2D at 0x7fd2a75736a0>]
```




```
for i in range(2,12):
    labels=cluster.KMeans(n_clusters=i,random_state=200).fit(pca_df).labels_
    print ("Silhouette score for k(clusters) = "+str(i)+" is "
           +str(metrics.silhouette_score(pca_df,labels,metric="euclidean",sample_size=1000,random_state=200)))
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value o
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value o
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value o
  warnings.warn(
Silhouette score for k(clusters) = 2 is 0.6592059486546701
Silhouette score for k(clusters) = 3 is 0.6689134780231564
Silhouette score for k(clusters) = 4 is 0.7119766490426919
Silhouette score for k(clusters) = 5 is 0.6071253575377029
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value o
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value o
  warnings.warn(
Silhouette score for k(clusters) = 6 is 0.5617125852521456
Silhouette score for k(clusters) = 7 is 0.5674921827754179
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value o
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value o
  warnings.warn(
Silhouette score for k(clusters) = 8 is 0.5550550658550716
Silhouette score for k(clusters) = 9 is 0.5264482662369026
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value o
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value o
  warnings.warn(
Silhouette score for k(clusters) = 10 is 0.46832576907434614
Silhouette score for k(clusters) = 11 is 0.4505841671377786
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value o
  warnings.warn(
```

```
kmeans = KMeans(n_clusters=2)
kmeans.fit(principalComponents)

# To predict the cluster for a new data point
```

```
reduced_new_data_point = pca.transform(x_test)
y_pred_5 = kmeans.predict(reduced_new_data_point)
y_pred_5
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value o
  warnings.warn(
```

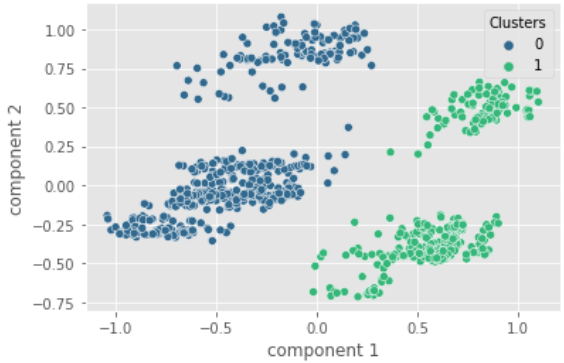
```
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but PCA was fitted with feature names
  warnings.warn(
array([0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1,
       1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1,
       1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0], dtype=int32)
```

```
accuracy_score(y_test, y_pred_5)
```

```
0.7210144927536232
```

```
pca_df['Clusters'] = kmeans.labels_ #getting our output

sn.scatterplot(x="component 1", y="component 2",hue = 'Clusters',  data=pca_df,palette='viridis')
```

```
<AxesSubplot:xlabel='component 1', ylabel='component 2'>
```

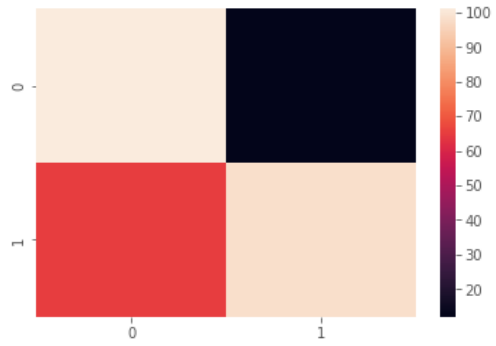



```
print(classification_report(y_test,y_pred_5))
```

```
                  precision    recall  f1-score   support

               0       0.61      0.89      0.72       113
               1       0.89      0.60      0.72       163

        accuracy                           0.72       276
       macro avg       0.75      0.75      0.72       276
    weighted avg       0.78      0.72      0.72       276
```

```
CM = confusion_matrix (y_test,y_pred_5)
sn.heatmap(CM)
print (CM)
```

```
[[101  12]
 [ 65  98]]
```



✓ 0s completed at 12:16