MACHINE LEARNING 2021

HEART ATTACK ANALYSIS AND PREDICTION.

STUDENT NAME : SURAJ MALLICK
SIC NO : 20BCTB60
BRANCH NAME : COMPUTER SCIENCE AND TECH (CST)

(i) Preparing a pre-processing dataset

+ Importing libraries

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

+ Studying the Dataset

. age

. sex : 1 = Male, 0 = Female (Binary)

. (cp) chest pain [type (4 values, Ordinal)]

. (trestbps) resting blood pressure

. (chol) serum cholestoral in mg/dl

. (fbs) fasting blood sugar > 120 mg/dl (Binary) [1 = true; 0 = false]

. (restecg) resting electrocardiographic results [values 0,1,2]

. (thalach) maximum heart rate achieved

. (exang) exercise induced angina (Binary) [1 = yes; 0 = no]

. (oldpeak) = ST depression induced by exercise relative to rest

. (slope) of the peak exercise ST segment (Ordinal) [1: upsloping, 2: flat , 3: downsloping)

. (ca) number of major vessels (0-3, Ordinal) colored by fluoroscopy

. (thal) maximum heart rate achieved (Ordinal) [3 = normal; 6 = fixed defect; 7 = reversable defect]

+ Importing dataset

In [ ]:
```
dataset1=pd.read_csv('heart.csv')
dataset2=pd.read_csv('o2saturation.csv')
```

In [ ]:
```
dataset1
```

Out[ ]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

+ Exploring Type of data Dataset1 contains

In [ ]:
```
dataset1.dtypes
```

Out[ ]:
```
age           int64
sex           int64
cp            int64
trtbps        int64
chol          int64
fbs           int64
restecg       int64
thalachh      int64
exng          int64
oldpeak     float64
slp           int64
caa           int64
thall         int64
output        int64
dtype: object
```

In [ ]:
```
dataset2
```

Out[ ]:

	98.6

	98.6
0	98.6
1	98.6
2	98.6
3	98.1
4	97.5
...	...
3580	98.6
3581	98.6
3582	98.6
3583	98.6
3584	98.6

3585 rows × 1 columns

+ Dividing the dataset into feature matrix and dependent variable vector

In [ ]:

```
X = dataset1.iloc[:,:-1].values
Y = dataset1.iloc[:,-1].values
```

In [ ]:

```
X
```

Out[ ]:

```
array([[63.,  1.,  3., ...,  0.,  0.,  1.],
       [37.,  1.,  2., ...,  0.,  0.,  2.],
       [41.,  0.,  1., ...,  2.,  0.,  2.],
       ...,
       [68.,  1.,  0., ...,  1.,  2.,  3.],
       [57.,  1.,  0., ...,  1.,  1.,  3.],
       [57.,  0.,  1., ...,  1.,  1.,  2.]])
```

In [ ]:

```
X.reshape(1,-1)
```

Out[ ]:

```
array([[63.,  1.,  3., ...,  1.,  1.,  2.]])
```

In [ ]:

```
Y
```

Out[ ]:

```
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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 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], dtype=int64)
```

+ Replacing missing data

In [ ]:
```
from sklearn.impute import SimpleImputer
imputer=SimpleImputer(missing_values=np.nan,strategy='mean')
imputer.fit(X[:,:])
X[:,:]=imputer.transform(X[:,:])
```

+ Spliting the data into Training and Testing Data

In [ ]:
```
from sklearn.model_selection import train_test_split
Xtrain,Xtest,Ytrain,Ytest=train_test_split(X,Y,test_size=0.2,random_state=1)
```

+ Feature scaling

In [ ]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
Xtrain=sc.fit_transform(Xtrain)
Xtest=sc.fit_transform(Xtest)
```

In [ ]:
```
Xtrain
```

Out[ ]:
```
array([[-0.27090572,  0.6636838 ,  1.9766492 , ..., -0.66896473,
        -0.72428597, -2.11701865],
       [ 1.3708101 , -1.50674161,  0.99843017, ...,  0.96628239,
         0.27160724, -0.47497213],
       [ 0.27633288,  0.6636838 ,  0.99843017, ...,  0.96628239,
         0.27160724,  1.16707438],
       ...,
       [-2.78820331,  0.6636838 ,  0.02021114, ...,  0.96628239,
        -0.72428597, -0.47497213],
       [-0.38035344,  0.6636838 , -0.95800789, ...,  0.96628239,
        -0.72428597,  1.16707438],
       [-0.05201028,  0.6636838 ,  0.99843017, ...,  0.96628239,
        -0.72428597,  1.16707438]])
```

In [ ]:
```
Xtrain.reshape(1,-1)
```

Out[ ]:
```
array([[-0.27090572,  0.6636838 ,  1.9766492 , ...,  0.96628239,
        -0.72428597,  1.16707438]])
```

ii) Building a Heart attack classification Model (obtained in (i))

+ Taining the model

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
KC=KNeighborsClassifier(n_neighbors=5, weights='uniform', p=2)
KC.fit(Xtrain,Ytrain)
```

Out[ ]:
```
KNeighborsClassifier()
```

+ Testing the model

In [ ]:
```
Y_estimated=KC.predict(Xtest)
Y_estimated
```

Out[ ]:
```
array([0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
       0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1], dtype=int64)
```

+ Performance Matrix

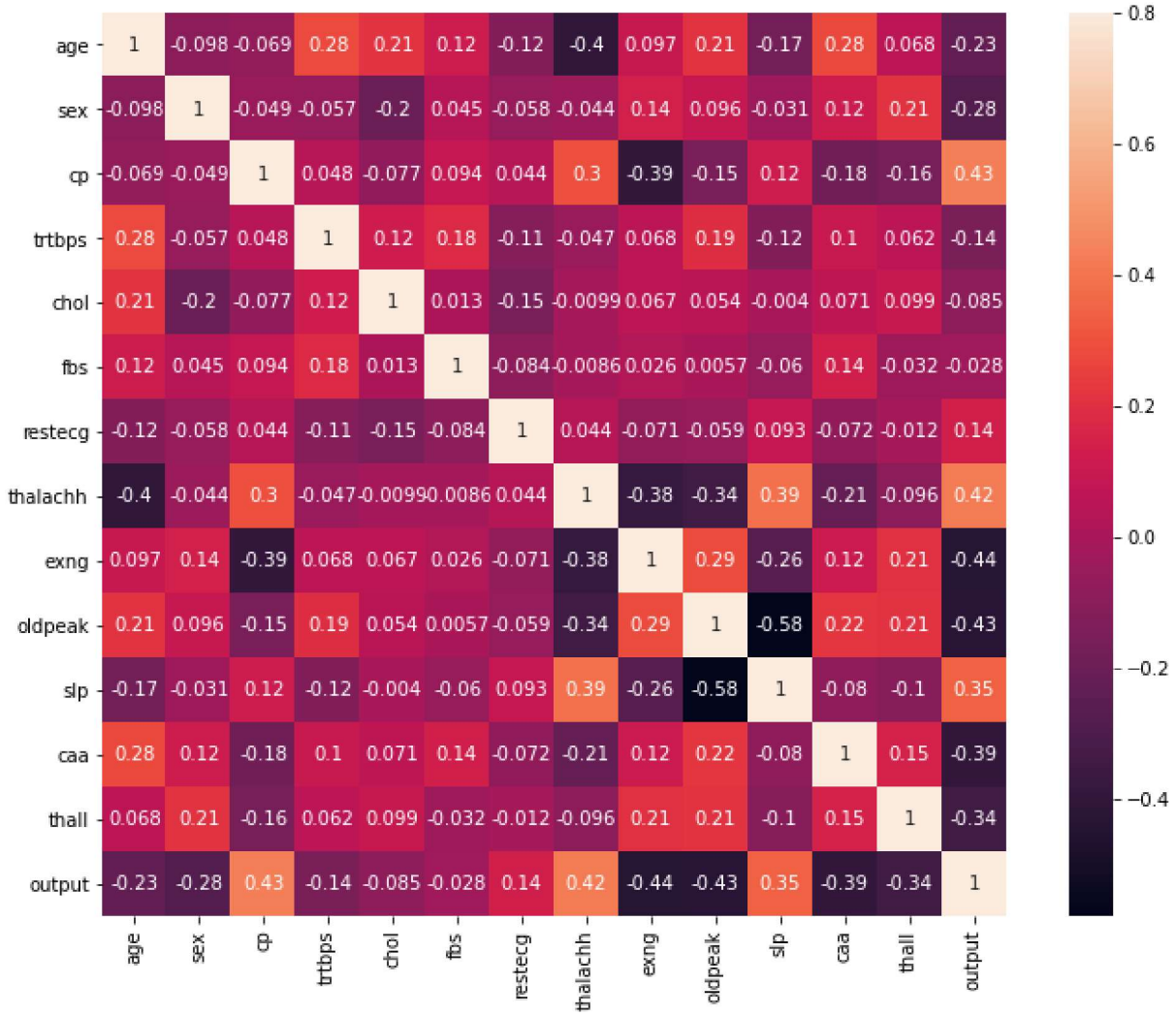
In [ ]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,precision_score,recall_s
cm=confusion_matrix(Ytest,Y_estimated)
print(cm)
print('\n')
print('KNN algorithm:-')
print('\n')
print(f"Accuracy score : {accuracy_score(Ytest,Y_estimated)}")
print(f"Precision score : {precision_score(Ytest,Y_estimated)}")
print(f"Recall score : {recall_score(Ytest,Y_estimated)}")
```

```
[[21  9]
 [ 6 25]]


KNN algorithm:-


Accuracy score : 0.7540983606557377
Precision score : 0.7352941176470589
Recall score : 0.8064516129032258
```

In [ ]:
```
error_rate=[]
for i in range(1,30):
    KNN=KNeighborsClassifier(n_neighbors=i)
    KNN.fit(Xtrain,Ytrain)
    ypred_i=KNN.predict(Xtest)
    error_rate.append(np.mean(ypred_i!=Ytest))
```

+ visualization of Dataset

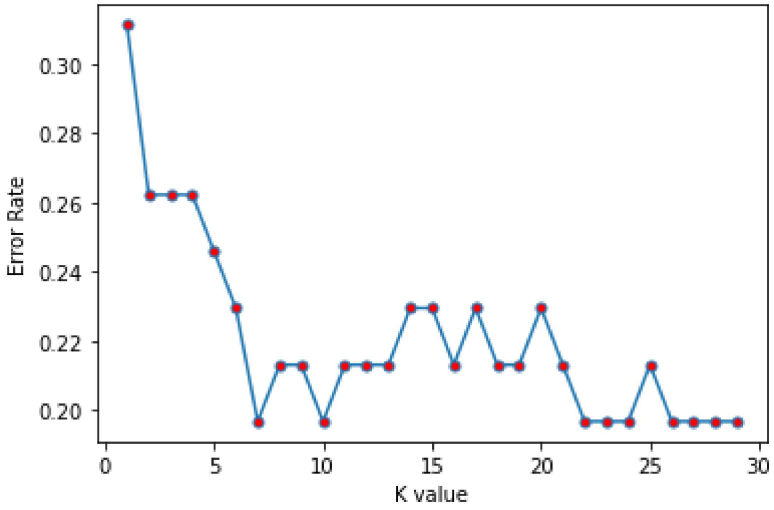
In [ ]:
```
 import seaborn as sns

corrmat = dataset1.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True, annot=True);
```



+ Ploting Error Rate vs K value Graph to Max Accuracy

In [ ]:
```
plt.plot(range(1,30),error_rate,marker='o',markerfacecolor='red',markersize=5)
plt.xlabel('K value')
plt.ylabel('Error Rate')
```

Out[ ]:
```
Text(0, 0.5, 'Error Rate')
```



~ Since Accuracy is higher at 7 and 10

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
KC=KNeighborsClassifier(n_neighbors=7, weights='uniform', p=2)
KC.fit(Xtrain,Ytrain)
```

Out[ ]:
```
KNeighborsClassifier(n_neighbors=7)
```

In [ ]:
```
Y_estimated=KC.predict(Xtest)
from sklearn.metrics import confusion_matrix,accuracy_score,precision_score,recall_s
cm=confusion_matrix(Ytest,Y_estimated)
print(cm)
print('\n')
print('KNN algorithm:-')
print('\n')
print(f"Accuracy score : {accuracy_score(Ytest,Y_estimated)}")
print(f"Precision score : {precision_score(Ytest,Y_estimated)}")
print(f"Recall score : {recall_score(Ytest,Y_estimated)}")
```

```
[[22  8]
 [ 4 27]]


KNN algorithm:-


Accuracy score : 0.8032786885245902
Precision score : 0.7714285714285715
Recall score : 0.8709677419354839
```

Prediction that a person will have heart attack from random data to the final model :-

```
age         24
sex         1
cp          2
trtbps      140
chol        200
fbs         1
restecg     0
thalachh    130
exng        0
oldpeak     2
slp         0
caa         0
thall       1
```

In [ ]:
```
KC.predict([[24,1,2,140,200,1,0,130,0,2,0,0,1]])
```

Out[ ]:
```
array([1], dtype=int64)
```

+ Result

```
As the predicted value is " 1 "  which means More chance of heart
attack for the person details which we have provided
```

In [ ]: