Problem Statement:2

In [1]:

```
import pandas as pd
```

In [3]:

```
Hospital=pd.read_csv("E:\Data science training\R AND PYTHON KPMG\stat and ml\ASSIGNMENT\Grey Sloan Hospital Data.
Hospital.head()
```

Out[3]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRat
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.

In [5]:

```
Hospital.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4235 entries, 0 to 4234
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   male             4235 non-null   int64  
 1   age              4235 non-null   int64  
 2   education        4131 non-null   float64
 3   currentSmoker    4235 non-null   int64  
 4   cigsPerDay       4206 non-null   float64
 5   BPMeds           4183 non-null   float64
 6   prevalentStroke  4235 non-null   int64  
 7   prevalentHyp     4235 non-null   int64  
 8   diabetes         4235 non-null   int64  
 9   totChol          4185 non-null   float64
 10  sysBP            4235 non-null   float64
 11  diaBP            4235 non-null   float64
 12  BMI              4216 non-null   float64
 13  heartRate        4234 non-null   float64
 14  glucose          3853 non-null   float64
 15  CHD chance       4235 non-null   int64  
dtypes: float64(9), int64(7)
memory usage: 529.5 KB
```

In [4]:

```
### CHeck how many missing values are there
print(Hospital.isnull().sum())
print(Hospital.shape)
```

```
male                 0
age                  0
education          104
currentSmoker        0
cigsPerDay          29
BPMeds              52
prevalentStroke      0
prevalentHyp         0
diabetes             0
totChol             50
sysBP                0
diaBP                0
BMI                 19
heartRate            1
glucose            382
CHD chance           0
dtype: int64
(4235, 16)
```

In [6]:

```
### Data is Normal --> we will replace the missing values with the Mean of it
### Data is Not-Normal ---> we will replace missing values with the Median values.

#### We will check the Normality of our data by measuring the skewness value.
#### If skewness is between -1 and +1. This indicates data is Normal -> replace with mean
#### If skewness is < -1 or > 1 . This indicates data is Not-Normal. -> replace with median
```

```
##### Check the skewness

print(Hospital['education'].skew())
print(Hospital['cigsPerDay'].skew())
print(Hospital['BPMeds'].skew())
print(Hospital['totChol'].skew())
print(Hospital['BMI'].skew())
print(Hospital['heartRate'].skew())
print(Hospital['glucose'].skew())
```

```
0.6886411572562287
1.2472028409989622
5.548558220881741
0.8717332513085908
0.9827318034290645
0.645224131915317
6.215121872910823
```

In [7]:

```
# Lets replace the missing values

Hospital['education'] = Hospital['education'].fillna(Hospital['education'].mean())
Hospital['cigsPerDay'] = Hospital['cigsPerDay'].fillna(Hospital['cigsPerDay'].median())
Hospital['BPMeds'] = Hospital['BPMeds'].fillna(0)
Hospital['totChol'] = Hospital['totChol'].fillna(Hospital['totChol'].mean())
Hospital['BMI'] = Hospital['BMI'].fillna(Hospital['BMI'].mean())
Hospital['heartRate'] = Hospital['heartRate'].fillna(Hospital['heartRate'].mean())
Hospital['glucose'] = Hospital['glucose'].fillna(Hospital['glucose'].mean())
```

In [8]:

```
## checking Null value
print(Hospital.isnull().sum())
```

```
male               0
age                0
education          0
currentSmoker      0
cigsPerDay         0
BPMeds             0
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol            0
sysBP              0
diaBP              0
BMI                0
heartRate          0
glucose            0
CHD chance         0
dtype: int64
```

In [10]:

```
# a) Lets build the logistic regression model and check accuracy

# STEP 1: Selecting the X and Y

X = Hospital.drop(columns=['CHD chance'])
Y = Hospital[['CHD chance']]

# STEP 2: Split the data into training and test
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, random_state = 1234)

len(X_train), len(X_test), len(Y_train), len(Y_test)
```

Out[10]:

```
(3388, 847, 3388, 847)
```

In [13]:

```
# create a model object

from sklearn.linear_model import LogisticRegression

LR = LogisticRegression()

# fit the model object on training data for building the model

model_lr = LR.fit(X_train, Y_train)

model_lr
```

Out[13]:

```
LogisticRegression()
```

In [14]:

```
Y_test['Pred_CHD'] = model_lr.predict(X_test)
```

In [15]:

```
Y_test
```

Out[15]:

	CHD chance	Pred_CHD
3165	1	0
3893	0	0
3106	0	0
350	0	0
1386	0	0
...	...	...
2027	0	0
85	0	0
381	0	0
1466	0	0
2075	0	0

847 rows × 2 columns

In [16]:

```
# Lets create a confusion matrix to check our model accuracy. by using our CHD chance column and Pred_CHD column

pd.crosstab(index=Y_test['CHD chance'], columns = Y_test['Pred_CHD'], margins=True)
```

Out[16]:

Pred_CHD	0	1	All
CHD chance			
0	731	2	733
1	111	3	114
All	842	5	847

In [17]:

```
# Create a confusion matrix and Evaluate the accuracy of model..... by using inbuilt functions

from sklearn.metrics import confusion_matrix, accuracy_score

confusion_matrix(Y_test['CHD chance'],Y_test['Pred_CHD'])
```

Out[17]:

```
array([[731,   2],
       [111,   3]], dtype=int64)
```

In [18]:

```
# Accuracy

accuracy = accuracy_score(Y_test['CHD chance'], Y_test['Pred_CHD'])

accuracy
```

Out[18]:

```
0.8665879574970484
```

In [ ]:

```

```

b) Using Decesion Tree For Identifying Accuracy

In [19]:

```
# Lets build the Decesion Tree model

# STEP 1: Selecting the X and Y

X1 = Hospital.drop(columns=['CHD chance'])
Y1= Hospital[['CHD chance']]
```

```
# STEP 2: Split the data into training and test
from sklearn.model_selection import train_test_split

X1_train, X1_test, Y1_train, Y1_test = train_test_split(X1,Y1, test_size = 0.2, random_state = 1234)

len(X1_train), len(X1_test), len(Y1_train), len(Y1_test)
```

Out[19]:

```
(3388, 847, 3388, 847)
```

In [20]:

```
## Build our model
# i will create a model object
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()

# I will fit the model object on my training data

model = dt.fit(X1_train, Y1_train)
```

In [21]:

```
# Evaluate our model Accuracy..... I will perform prediction on my test data

Y1_test['predicted_CHD'] = model.predict(X1_test)
```

In [22]:

```
Y1_test
```

Out[22]:

	CHD chance	predicted_CHD
3165	1	0
3893	0	0
3106	0	0
350	0	0
1386	0	0
...	...	...
2027	0	0
85	0	0
381	0	0
1466	0	0
2075	0	0

847 rows × 2 columns

In [23]:

```
# Create a confusion matrix and Evaluate the accuracy of model..... by using inbuilt functions

from sklearn.metrics import confusion_matrix, accuracy_score

confusion_matrix(Y1_test['CHD chance'],Y1_test['predicted_CHD'])
```

Out[23]:

```
array([[621, 112],
       [ 76,  38]], dtype=int64)
```

In [24]:

```
accuracy = accuracy_score(Y1_test['CHD chance'], Y1_test['predicted_CHD'])

accuracy
```

Out[24]:

```
0.7780401416765053
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js