In [ ]:
```
import pandas  as pd
import numpy as np
import warnings # to avoid warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

In [ ]:
```
data =  pd.read_csv("heartDiease.csv")
data.head(10)
```

Out[ ]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type
0	9046	Male	67.0	0	1	Yes	Private	Urban
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural
2	31112	Male	80.0	0	1	Yes	Private	Rural
3	60182	Female	49.0	0	0	Yes	Private	Urban
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural
5	56669	Male	81.0	0	0	Yes	Private	Urban
6	53882	Male	74.0	1	1	Yes	Private	Rural
7	10434	Female	69.0	0	0	No	Private	Urban
8	27419	Female	59.0	0	0	Yes	Private	Rural
9	60491	Female	78.0	0	0	Yes	Private	Urban

In [ ]:
```
data = data.drop(columns = ['id']) # Droping column id
```

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   gender             5110 non-null   object 
 1   age                5110 non-null   float64
 2   hypertension       5110 non-null   int64  
 3   heart_disease      5110 non-null   int64  
 4   ever_married       5110 non-null   object 
 5   work_type          5110 non-null   object 
 6   Residence_type     5110 non-null   object 
 7   avg_glucose_level  5110 non-null   float64
```

```
 8   bmi                4909 non-null   float64
 9   smoking_status     5110 non-null   object
 10  stroke             5110 non-null   int64
dtypes: float64(3), int64(3), object(5)
memory usage: 439.3+ KB
```

In [ ]:
```
data = data.dropna() #dropping null values
```

In [ ]:
```
data.shape
```

Out[ ]:
```
(4909, 11)
```

In [ ]:
```
## pivot implmentation on sample values
df = pd.DataFrame({'A': ['John', 'Boby', 'Mina'],
      'B': ['Masters', 'Graduate', 'Graduate'],
      'C': [27, 23, 21]})
```

In [ ]:
```
df.pivot('A','B','C')
```

Out[ ]:

A	B	Graduate	Masters
Boby		23.0	NaN
John		NaN	27.0
Mina		21.0	NaN

In [ ]:
```
data['age'] = data['age'].astype('int') # converting data type of age from floa
```

In [ ]:
```
data['gender'].value_counts()
```

Out[ ]:
```
Female    2897
Male      2011
Other        1
Name: gender, dtype: int64
```

In [ ]:
```
data = data.drop(data[data['gender']=='Other'].index) #removing row with gender
```

Replacing textual data to numeric values

In [ ]:
```
data['gender'] = data['gender'].replace('Male',1)
data['gender'] = data['gender'].replace('Female',0)
data['ever_married'] = data['gender'].replace('Yes',1)
data['ever_married'] = data['gender'].replace('No',0)
data['Residence_type'] = data['gender'].replace('Urban',1)
data['Residence_type'] = data['gender'].replace('Rural',0)
data=data[data['smoking_status']!='Unknown'] #removing row with gender as 'Unkn
data
```

Out[ ]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg
0	1	67	0	1	1	Private	1	
2	1	80	0	1	1	Private	1	
3	0	49	0	0	0	Private	0	
4	0	79	1	0	0	Self-employed	0	
5	1	81	0	0	1	Private	1	
...	...	...	...	...	...	...	...	
5100	1	82	1	0	1	Self-employed	1	
5102	0	57	0	0	0	Private	0	
5106	0	81	0	0	0	Self-employed	0	
5107	0	35	0	0	0	Self-employed	0	
5108	1	51	0	0	1	Private	1	

3425 rows × 11 columns

In [ ]:
```
data.corr() # performing data correlation
```

Out[ ]:

	gender	age	hypertension	heart_disease	ever_married	Residence_typ
gender	1.000000	0.044580	0.037853	0.101834	1.000000	1.00000
age	0.044580	1.000000	0.267030	0.260077	0.044580	0.04458
hypertension	0.037853	0.267030	1.000000	0.111683	0.037853	0.03785
heart_disease	0.101834	0.260077	0.111683	1.000000	0.101834	0.10183
ever_married	1.000000	0.044580	0.037853	0.101834	1.000000	1.00000
Residence_type	1.000000	0.044580	0.037853	0.101834	1.000000	1.00000
avg_glucose_level	0.070169	0.233974	0.168769	0.143152	0.070169	0.07016
bmi	0.014284	0.079317	0.132544	0.001208	0.014284	0.01428
stroke	0.012413	0.242465	0.143626	0.138538	0.012413	0.01241

In [ ]:
```
# One Hot encoding smoking_status, work_type
data_dummies = data[['smoking_status','work_type']]
data_dummies=pd.get_dummies(data_dummies)
# get_dummies to convert the categorical variables into binary columns
```

```
data.drop(columns=['smoking_status','work_type'],inplace=True)  #dropping colum
```

In [ ]:
```
data_stroke=data['stroke']
data.drop(columns=['stroke'],inplace=True) #droping stroke column
data=data.merge(data_dummies,left_index=True, right_index=True,how='left')#merg
```

Splitting the data into training and testing sets.

In [ ]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(data,data_stroke,test_size=0.2
```

In [ ]:
```
# Standardizing training and testing data.
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

Training the model

In [ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
#from sklearn.metrics import plot_confusion_matrix
```

In [ ]:
```
def Model(model):
    model.fit(x_train,y_train)
    score = model.score(x_test, y_test)
    model_train_score = model.score(x_train, y_train)
```