NAME- RAJANANDINI MOHARANA

COLLEGE- FAKIR MOHAN MEDICAL COLLEGE & HOSPITAL

STREAM- MBBS

YEAR- 1ST

```
#MAJOR PROJECT-1

#Choose any dataset of your choice and apply a suitable CLASSIFIER/REGRESSOR


#DATASET-BRAIN-STROKE
#URL-/content/brain_stroke.csv


#CREATION OF DATAFRAME
import pandas as pd
df=pd.read_csv('/content/brain_stroke.csv')
df
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
2	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
3	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
4	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
...	...	...	...	...	...	...	...	...	...	...	...
4976	Male	41.0	0	0	No	Private	Rural	70.15	29.8	formerly smoked	0
4977	Male	40.0	0	0	Yes	Private	Urban	191.15	31.1	smokes	0
4978	Female	45.0	1	0	Yes	Govt_job	Rural	95.02	31.8	smokes	0
4979	Male	40.0	0	0	Yes	Private	Rural	83.94	30.0	smokes	0
4980	Female	80.0	1	0	Yes	Private	Urban	83.75	29.1	never smoked	0

4981 rows × 11 columns

```
df.shape
#Rows-4981,Columns-11
```

```
(4981, 11)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4981 entries, 0 to 4980
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   gender             4981 non-null   object 
 1   age                4981 non-null   float64
 2   hypertension       4981 non-null   int64  
 3   heart_disease      4981 non-null   int64  
 4   ever_married       4981 non-null   object 
 5   work_type          4981 non-null   object 
 6   Residence_type     4981 non-null   object 
 7   avg_glucose_level  4981 non-null   float64
 8   bmi                4981 non-null   float64
 9   smoking_status     4981 non-null   object 
 10  stroke             4981 non-null   int64  
dtypes: float64(3), int64(3), object(5)
memory usage: 428.2+ KB
```

```
#Here I create another dataframe from the existing one which contains only float and int datatypes values
df_num= df.select_dtypes(include = ['float64','int64'])
df_num
```

	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
0	67.0	0	1	228.69	36.6	1
1	80.0	0	1	105.92	32.5	1
2	49.0	0	0	171.23	34.4	1
3	79.0	1	0	174.12	24.0	1
4	81.0	0	0	186.21	29.0	1
...	...	...	...	...	...	...
4976	41.0	0	0	70.15	29.8	0
4977	40.0	0	0	191.15	31.1	0
4978	45.0	1	0	95.02	31.8	0
4979	40.0	0	0	83.94	30.0	0
4980	80.0	1	0	83.75	29.1	0

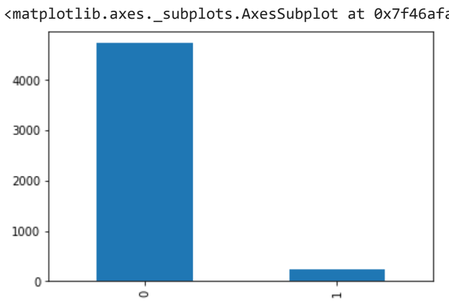
4981 rows × 6 columns

```
#I want to know how many no of stokes are there
df['stroke'].value_counts()
```

```
0    4733
1     248
Name: stroke, dtype: int64
```

```
#Visualisation
df['stroke'].value_counts().plot(kind = 'bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f46afa0c6d0>
```




```
#Divide the data into input and output
#input data
x = df_num.iloc[:,0:5].values
x
```

```
array([[ 67.  ,   0.  ,   1.  , 228.69,  36.6 ],
       [ 80.  ,   0.  ,   1.  , 105.92,  32.5 ],
       [ 49.  ,   0.  ,   0.  , 171.23,  34.4 ],
       ...,
       [ 45.  ,   1.  ,   0.  ,  95.02,  31.8 ],
       [ 40.  ,   0.  ,   0.  ,  83.94,  30.  ],
       [ 80.  ,   1.  ,   0.  ,  83.75,  29.1 ]])
```

```
#output data
y = df_num.iloc[:,5].values
y
```

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

```
#Train_test_split/train and test variables
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 0)


print(x.shape)
print(x_train.shape)
print(x_test.shape)
```

```
(4981, 5)
(3735, 5)
(1246, 5)
```

```
print(y.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(4981,)
(3735,)
(1246,)
```

```
#NORMALISATION or SCALING
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
```

```
x_test = scaler.fit_transform(x_test)
```

```
#Apply Classifier/Regressor
#Here I apply Regressor
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

```
#Fitting the model
model.fit(x_train,y_train)
```

```
     LogisticRegression()
```

```
#Predict the output
y_pred = model.predict(x_test)
y_pred #PREDCITED VALUES
```

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

```
y_test
```

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

```
#Accuracy
from sklearn.metrics import accuracy_score
accuracy_score(y_pred,y_test)* 100
```

```
     95.26484751203851
```

```
#Individual Prediction
m= scaler.transform([[81.0,0,0,186.21,29.0]])
```

```
model.predict(m)
```

```
     array([0])
```

```
m
```

```
array([[0.98779297, 0.        , 0.        , 0.61970098, 0.42816092]])
```

```
#MAJOR PROJECT 2

#Choose any dataset of your choice and apply K Means Clustering


#DATASET-CORRUPTION
#URL-/content/corruption (1).csv


#Create DataFrame
import pandas as pd
df=pd.read_csv('/content/corruption (1).csv')
df
```

	country	annual_income	corruption_index
0	Denmark	68110	12
1	Finland	53660	12
2	New Zealand	45340	12
3	Norway	84090	15
4	Singapore	64010	15
...	...	...	...
105	Yemen	670	84
106	Venezuela	13080	86
107	Somalia	450	87
108	Syria	1170	87
109	South Sudan	460	89

110 rows × 3 columns

```
df.shape
#Rows=110
#Columns=3
```

```
(110, 3)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110 entries, 0 to 109
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   country           110 non-null    object
 1   annual_income     110 non-null    int64 
 2   corruption_index  110 non-null    int64 
dtypes: int64(2), object(1)
memory usage: 2.7+ KB
```

```
#I want to create a dataframe having numeric values only
df_num = df.select_dtypes(include = ['int64'])
df_num
```

	annual_income	corruption_index
0	68110	12
1	53660	12
2	45340	12
3	84090	15
4	64010	15
...	...	...
105	670	84
106	13080	86
107	450	87
108	1170	87
109	460	89

110 rows × 2 columns

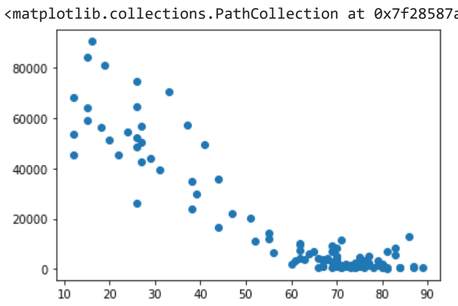
```
df_num.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110 entries, 0 to 109
Data columns (total 2 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   annual_income     110 non-null    int64
 1   corruption_index  110 non-null    int64
dtypes: int64(2)
memory usage: 1.8 KB
```

```
#Input-'annual_income' and 'corruption_index' column
#Divide the data ino input
#Slicing of input columns
x=df_num.iloc[:,0:2].values
x
```

```
#Visualisation before clustering
import matplotlib.pyplot as plt
plt.scatter(df['corruption_index'],df['annual_income'])
```

```
<matplotlib.collections.PathCollection at 0x7f28587a1f90>
```



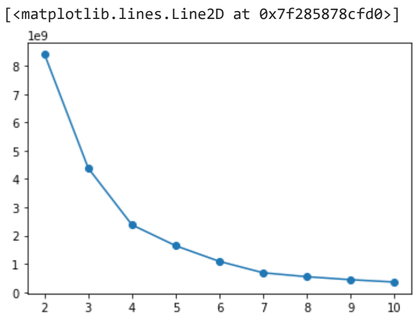

```
import numpy as np
np.sqrt(110)#110 is the total number of points
#No of cluster=k
```

```
10.488088481701515
```

```
#Find out the value of k
#1.Elbow method
from sklearn.cluster import KMeans
k=range(2,11)#k range is in b/w 2 and 11
sse=[]

for i in k:
  model_demo=KMeans(n_clusters=i,random_state=0)
  model_demo.fit(x)
  sse.append(model_demo.inertia_)
plt.scatter(k,sse)
plt.plot(k,sse)
```

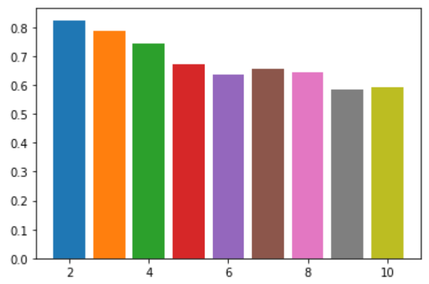
```
[<matplotlib.lines.Line2D at 0x7f285878cfd0>]
```




```
#2.Silhouette score method to find out k value
from sklearn.metrics import silhouette_score
k=range(2,11)
for i in k:
  model_demo=KMeans(n_clusters=i,random_state=0)
```

```
  model_demo.fit(x)
  y_pred=model_demo.predict(x)
  print(f"{i} Clusters ,Score={silhouette_score(x,y_pred)}")
  plt.bar(i,silhouette_score(x,y_pred))
```

```
    2 Clusters ,Score=0.8250423417665396
    3 Clusters ,Score=0.7873768660102557
    4 Clusters ,Score=0.7445340205192341
    5 Clusters ,Score=0.6727183336946292
    6 Clusters ,Score=0.6372161082857853
    7 Clusters ,Score=0.6546866089993955
    8 Clusters ,Score=0.6456656559747936
    9 Clusters ,Score=0.5859720648427018
    10 Clusters ,Score=0.5947956746599683
```




```
#Apply clusterer
k=2
from sklearn.cluster import KMeans
model=KMeans(n_clusters=k,random_state=0)
model.fit(x)
```

```
    KMeans(n_clusters=2, random_state=0)
```

```
#Output prediction
y=model.predict(x)
y
```

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

```
y.size
```

```
    110
```

```
x[y==1,0]
#The value of input when cluster 1 is selcted and column index 0 is selected
```

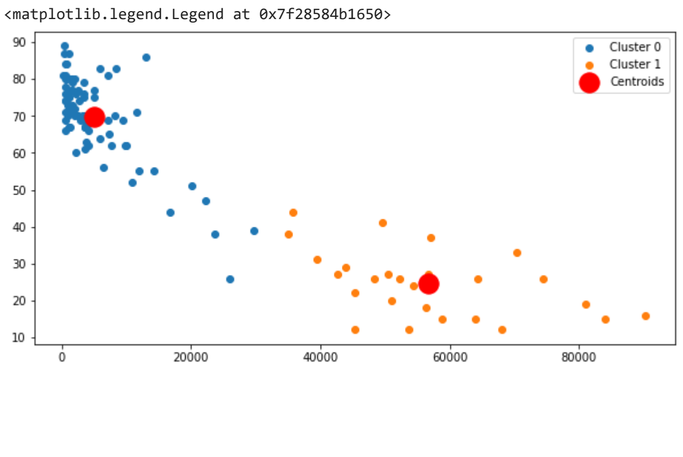
```
    array([68110, 53660, 45340, 84090, 64010, 58890, 90360, 56370, 81110,
           51040, 45380, 54450, 52210, 48310, 64410, 74520, 56760, 50510,
           42620, 43880, 39410, 70430, 57120, 34980, 49560, 35710])
```

```
np.unique(y,return_counts=True)
```

```
    (array([0, 1], dtype=int32), array([84, 26]))
```

```
#Final Visualisation after clustering
plt.figure(figsize=(10,5))
for i in range(k):
  plt.scatter(x[y==i,0],x[y==i,1],label=f'Cluster {i}')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],s=300,c='red',label='Centroids')
plt.legend()
```

```
    <matplotlib.legend.Legend at 0x7f28584b1650>
```



Github link-https://github.com/Rj0246/Major-Projects