Introduction

Name: Srinjoy Sur
College: Kalinga Institute Of Industrial Technology
Branch: CSE
Year: 2nd

AI/ML Model:
Major Project-1(Choose any dataset of your choice and apply a suitable CLASSIFIER/REGRESSOR):

```
#1. Taking data and creating data frame
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/heart_disease.csv', encoding = 'latin-1')
df
```

	Unnamed: 0	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 15 columns

```
#2. Dividing the data into input and output
x = df.iloc[:,0:4].values
x
```

```
array([[  0,  63,   1,   3],
       [  1,  37,   1,   2],
       [  2,  41,   0,   1],
       ...,
       [300,  68,   1,   0],
       [301,  57,   1,   0],
       [302,  57,   0,   1]])
```

```
[ ] y = df.iloc[:,4].values
    y
```

```
array([145, 130, 130, 120, 120, 140, 140, 120, 172, 150, 140, 130, 130,
       110, 150, 120, 120, 150, 150, 140, 135, 130, 140, 150, 140, 160,
       150, 110, 140, 130, 105, 120, 130, 125, 125, 142, 135, 150, 155,
       160, 140, 130, 104, 130, 140, 120, 140, 138, 128, 138, 130, 120,
       130, 108, 135, 134, 122, 115, 118, 128, 110, 108, 118, 135, 140,
       138, 100, 130, 120, 124, 120,  94, 130, 140, 122, 135, 125, 140,
       128, 105, 112, 128, 102, 152, 102, 115, 118, 101, 110, 100, 124,
       132, 138, 132, 112, 142, 140, 108, 130, 130, 148, 178, 140, 120,
       129, 120, 160, 138, 120, 110, 180, 150, 140, 110, 130, 120, 130,
       120, 105, 138, 130, 138, 112, 108,  94, 118, 112, 152, 136, 120,
       160, 134, 120, 110, 126, 130, 120, 128, 110, 128, 120, 115, 120,
       106, 140, 156, 118, 150, 120, 130, 160, 112, 170, 146, 138, 130,
       130, 122, 125, 130, 120, 132, 120, 138, 138, 160, 120, 140, 130,
       140, 130, 110, 120, 132, 130, 110, 117, 140, 120, 150, 132, 150,
       130, 112, 150, 112, 130, 124, 140, 110, 130, 128, 120, 145, 140,
       170, 150, 125, 120, 110, 110, 125, 150, 180, 160, 128, 110, 150,
       120, 140, 128, 120, 118, 145, 125, 132, 130, 130, 135, 130, 150,
       140, 138, 200, 110, 145, 120, 120, 170, 125, 108, 165, 160, 120,
       130, 140, 125, 140, 125, 126, 160, 174, 145, 152, 132, 124, 134,
       160, 192, 140, 140, 132, 138, 100, 160, 142, 128, 144, 150, 120,
       178, 112, 123, 108, 110, 112, 180, 118, 122, 130, 120, 134, 120,
       100, 110, 125, 146, 124, 136, 138, 136, 128, 126, 152, 140, 140,
       134, 154, 110, 128, 148, 114, 170, 152, 120, 140, 124, 164, 140,
       110, 144, 130, 130])
```

```
[ ] #3.training and testing 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)
```

```
[ ] #4.APPLYING CLASSIFIER/REGRESSOR/CLUSTERER
    from sklearn.linear_model import LogisticRegression
    model = LogisticRegression()
```

```
[ ] #5.fitting the model
    model.fit(x_train,y_train)
```

```
[ ] #6.Predicting the output
    y_pred = model.predict(x_test)
    y_pred  #predicted output
```

```
array([140, 120, 120, 140, 130, 120, 120, 130, 140, 140, 130, 140, 140,
       130, 130, 130, 130, 130, 140, 120, 130, 120, 120, 120, 120, 120,
       120, 126, 120, 130, 120, 126, 120, 130, 120, 140, 120, 130, 140,
       140, 130, 130, 130, 126, 130, 130, 120, 120, 140, 130, 130, 130,
       120, 130, 120, 126, 120, 120, 130, 130, 130, 130, 130, 130, 140,
       120, 120, 140, 130, 130, 130, 130, 120, 140, 140, 130])
```

```
[ ] y_test
```

```
array([145, 170, 170, 125, 130, 124, 110, 130, 200, 130, 150, 130, 135,
       130, 120, 100, 108, 124, 120, 120, 140, 118, 120, 110, 110, 130,
       125, 120, 138, 134, 140, 142, 126, 172, 122, 174, 160, 128, 125,
       140, 140, 128, 138, 110, 125, 156, 130, 140, 134, 120, 120, 120,
       112, 135,  94, 120, 118, 130, 150, 120, 100, 122, 146, 140, 124,
       154, 134, 125, 112, 140, 135, 140, 120, 160, 150, 110])
```

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

```
9.210526315789473
```

```
[ ] #Individual Prediction
    model.predict([[5.1,3.5,1.4,0.2]])
```

```
array([130])
```

```
[ ] model.predict([[5.9,3.2,4.8,1.8]])
```

```
array([120])
```

```
[ ] model.predict([[6.7,3.1,4.7,1.5]])
```

```
array([120])
```

```
[ ] model.predict([[6.8,3.2,5.9,2.3]])
```

```
array([120])
```

Major Project-2(Choose any dataset of your choice and apply K Means Clustering):

```
#UNSUPERVISED LEARNING -CLUSTERING -  K MEANS CLUSTERING
#1. Taking data and creating data frame
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/Social_Network_Ads.csv', encoding = 'latin-1')
df
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

```
[ ] df.shape #400 rows and 5 columnns
```

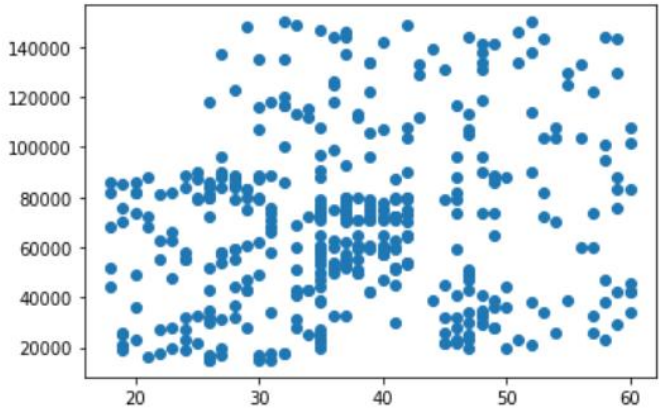
```
(400, 5)
```

```
[ ]  #2.divide the into input and output
     x = df.iloc[:,3:5].values
     x
```

```
array([[ 19000,      0],
       [ 20000,      0],
       [ 43000,      0],
       [ 57000,      0],
       [ 76000,      0],
       [ 58000,      0],
       [ 84000,      0],
       [150000,      1],
       [ 33000,      0],
       [ 65000,      0],
       [ 80000,      0],
       [ 52000,      0],
       [ 86000,      0],
       [ 18000,      0],
       [ 82000,      0],
       [ 80000,      0],
       [ 25000,      1],
       [ 26000,      1],
       [ 28000,      1],
       [ 29000,      1],
       [ 22000,      1],
       [ 49000,      1],
       [ 41000,      1],
       [ 22000,      1],
```

```
[ ]  #3.VISUALISATION
     import matplotlib.pyplot as plt
     plt.scatter(df['Age'],df['EstimatedSalary'])
     #Here we have got only one cluster before applying any clustering technique
```

```
<matplotlib.collections.PathCollection at 0x7f904e6a4790>
```




```
[ ]  #4.Finding out the number of clusters(k)
     import numpy as np
     np.sqrt(80) # 80 is the total no of points
     #No of cluster - k
     #k value should not exceed the square root of the total no of points
     #Hence k value should be in the range of 2 to 14
```

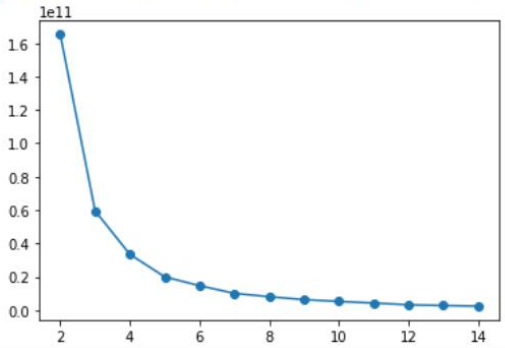
```
8.94427190999916
```

```
[ ] #5.ELBOW METHOD
    from sklearn.cluster import KMeans
    k = range(2,15)# my range is in between 2 and 14

    sse = [] #blank list

    #for i in range(2,15):
    for i in k :
      model_demo = KMeans(n_clusters = i,random_state = 0)
      model_demo.fit(x)
      sse.append(model_demo.inertia_)#.inertia_ - calculates the sum of squared error
    plt.scatter(k,sse)
    plt.plot(k,sse)
```

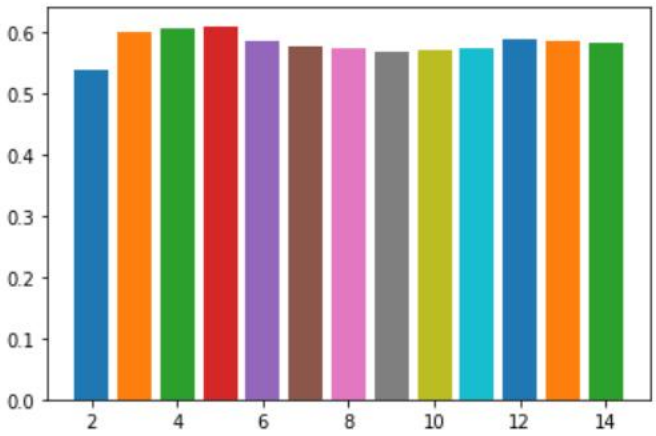
```
[<matplotlib.lines.Line2D at 0x7f904e99f290>]
```




```
[ ] #6.SILHOUETTE SCORE METHOD
    from sklearn.metrics import silhouette_score
    k = range(2,15)
    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 = 0.537477824964991
3 Clusters ,Score = 0.6015031263266077
4 Clusters ,Score = 0.6066146767843497
5 Clusters ,Score = 0.610223988128696
6 Clusters ,Score = 0.5846075186044158
7 Clusters ,Score = 0.5771757527583528
8 Clusters ,Score = 0.5734148086437275
9 Clusters ,Score = 0.567938363430772
10 Clusters ,Score = 0.5707280151556086
11 Clusters ,Score = 0.5745161226163872
12 Clusters ,Score = 0.5899406839951461
13 Clusters ,Score = 0.5856208761624656
14 Clusters ,Score = 0.5815430121040257
```

```
8 Clusters ,Score = 0.5734148086437275
9 Clusters ,Score = 0.567938363430772
10 Clusters ,Score = 0.5707280151556086
11 Clusters ,Score = 0.5745161226163872
12 Clusters ,Score = 0.5899406839951461
13 Clusters ,Score = 0.5856208761624656
14 Clusters ,Score = 0.5815430121040257
```



[10]
```
#7.APPLYING CLUSTERER
k = 5
from sklearn.cluster import KMeans

model = KMeans(n_clusters = k,random_state = 0)
model.fit(x)
```

```
KMeans(n_clusters=5, random_state=0)
```

[11]
```
#8.Predicting output
y = model.predict(x) # predicted output
y
```

```
array([2, 2, 1, 1, 3, 1, 3, 4, 2, 1, 3, 1, 3, 2, 3, 3, 2, 2, 2, 2, 2, 1,
       1, 2, 2, 2, 2, 2, 1, 2, 3, 4, 2, 1, 3, 2, 2, 1, 3, 2, 2, 1, 0, 2,
       3, 2, 3, 1, 4, 3, 2, 1, 3, 2, 1, 1, 1, 3, 2, 0, 2, 3, 1, 0, 3, 1,
       2, 3, 1, 3, 3, 2, 2, 0, 2, 0, 1, 2, 3, 2, 3, 1, 1, 3, 1, 0, 1, 3,
       3, 1, 3, 0, 2, 2, 3, 1, 2, 0, 3, 2, 3, 1, 3, 4, 2, 3, 2, 3, 3, 3,
       3, 3, 1, 1, 3, 1, 3, 1, 1, 1, 3, 3, 3, 1, 1, 1, 1, 2, 2, 3, 1, 2,
       3, 3, 1, 1, 3, 0, 1, 2, 3, 3, 1, 3, 2, 3, 0, 2, 1, 3, 2, 1, 3, 1,
       1, 2, 1, 3, 2, 4, 0, 3, 2, 2, 3, 3, 1, 3, 4, 1, 3, 0, 0, 1, 3, 2,
       1, 2, 2, 2, 2, 3, 0, 1, 1, 1, 3, 1, 3, 2, 3, 2, 1, 3, 3, 1, 3, 2,
       3, 2, 2, 3, 4, 3, 0, 1, 4, 0, 4, 2, 0, 4, 1, 1, 1, 0, 1, 3, 0, 4,
       3, 3, 4, 0, 1, 1, 4, 4, 3, 3, 4, 1, 0, 3, 0, 3, 1, 3, 3, 4, 4, 1,
       3, 0, 3, 4, 1, 0, 1, 0, 2, 1, 4, 4, 1, 3, 3, 1, 0, 4, 3, 4, 4, 3,
       3, 0, 3, 3, 4, 1, 4, 3, 1, 0, 2, 3, 3, 3, 2, 2, 3, 1, 3, 2, 4, 3,
       1, 4, 3, 3, 4, 3, 2, 3, 1, 1, 3, 0, 3, 0, 2, 3, 4, 3, 1, 1, 4, 0,
       4, 1, 3, 0, 1, 4, 3, 3, 0, 1, 2, 1, 4, 3, 1, 2, 4, 1, 3, 3, 0, 0,
       1, 0, 1, 1, 1, 1, 4, 3, 1, 0, 0, 3, 1, 1, 0, 1, 3, 0, 3, 1, 0, 3,
       3, 1, 0, 2, 3, 3, 3, 1, 4, 2, 1, 3, 0, 2, 1, 3, 3, 2, 1, 3, 3, 4,
       3, 2, 3, 1, 3, 2, 1, 2, 4, 2, 2, 1, 2, 3, 2, 2, 2, 2, 1, 1, 1, 1,
       2, 2, 2, 2], dtype=int32)
```

```
[12] #9.Finding size of y
     y.size
```

```
400
```

```
[13] #10.Selecting no. of rows and columns
     x[y == 1,1]
     #so the first '1' is cluster no 1 and the second '1' is column index 1
     #the value of input,when cluster 1 is selected and column index 1 selected
```

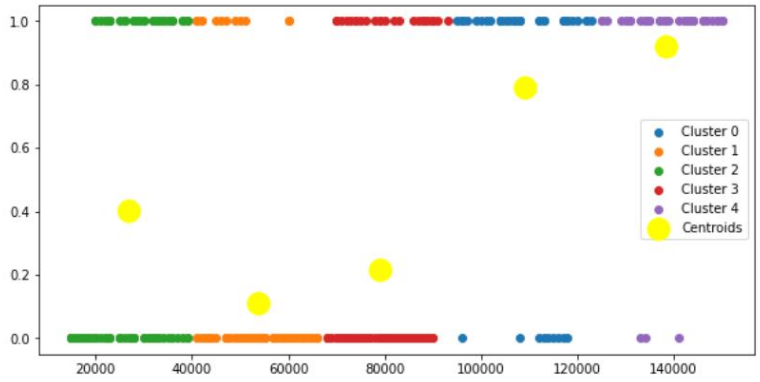
```
array([0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0,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

```
[14] #12.Selecting the unique elements in the rows and columns
     np.unique(y,return_counts = True)
```

```
(array([0, 1, 2, 3, 4], dtype=int32), array([ 43, 108,  87, 124,  38]))
```

```
#13.FINAL VISUALISATION
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 = 'yellow',
            label = 'Centroids')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fe430d39a10>
```



Github link: https://github.com/SrinjoySur/Rinex-Major-Projects