Report on Major Project

Introduction :-

Name - Sushmitha Santhosh

Name of College - Shree L.R.Tiwari . College Of Engineering

Year of Study- Second Year

Branch - Electronics and Computer Science

MAJOR PROJECT 1

Problem Statement : Choose any dataset of your choice and apply a suitable CLASSIFIER/REGRESSOR and if possible deploy it on heroku.

1. **About the dataset :** This data set dates from 1988 and consists of four databases: Cleveland, Hungary, Switzerland, and Long Beach V. It contains 76 attributes, including the predicted attribute, but all published experiments refer to using a subset of 14 of them. The "target" field refers to the presence of heart disease in the patient. It is integer valued 0 = no disease and 1 = disease.

Dataset link - https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

2. **Methodology :**

1. **Take the data and create dataframe**
   We imported the csv into google colab cell The dataset consists of 14 columns and 1025 elements .
2. **Visualization**
   According to accuracy among the plots in age vs cholesterol levels and age vs Fasting Blood Sugar . The age vs cholesterol level plot was more reliable .
3. **Divide the data into i/p and o/p**
   We divided the data into input as age, sex, cp,trestbps, chol, fbs,restecg, thalach,exang,old peak,slope, ca, thal . And the output as target .
4. **Test and train the variables**
   Using sklearn.model_selection library we need to train and test both variables of input and output .
5. **Normalisation**
   For the variables x_train and x_test , we need to use the library from sklearn.preprocessing and the import MinMaxScaler.
6. **Apply regressor**
   We need to apply logistic regression for the model we trained and tested .
7. **Fitting the model**
   The models which are tested and trained x_train and y_train in logistic regression
8. **Predict the Output**
   As we have completed the process of training and testing, we need to create predicted values to check accuracy .
9. **Accuracy**

The accuracy score of our model is 85.99221789883269.

10. Individual predictions

The individual predictions can be made from using values according to the array .

3. Screenshots of Code

```
[ ] #major project 1 on Choose any dataset of your choice and apply a suitable CLASSIFIER/REGRESSOR
```

```
[ ] #dataset -Heart Disease Dataset
    #link for dataset- https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset
    #Context
    #"target" field refers to the presence of heart disease in the patient. It is integer valued 0 = no disease and 1 = disease.
```

```
[ ] #1.Take the data and create dataframe
    import pandas as pd
    df = pd.read_csv('/content/heart.csv')
    df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0

```
[ ]
```

1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

1025 rows × 14 columns

```
[ ] #target - 0 and
    #0 = no disease
    #1 = disease
```

```
[ ] %%capture
    """Attribute Information:
    age: Age of the individual (integer)
    sex: Gender of the individual (integer) 1= male; 0 = female
    cp: Chest pain type (integer)
    trestbps: Resting blood pressure (integer)
    chol: Serum cholesterol level (integer)
    fbs: Fasting blood sugar level (integer) (1 = true; 0 = false)
    restecg: Resting electrocardiographic results (integer)
    thalach: Maximum heart rate achieved (integer)
    exang: Exercise-induced angina (integer)
    oldpeak: ST depression induced by exercise relative to rest (float)
    slope: Slope of the peak exercise ST segment (integer)
    ca: Number of major vessels colored by fluoroscopy (integer)
    thal: Thalassemia (integer)
```

```
[ ] thal: Thalassemia (integer)
    target: Presence of heart disease (integer)

    """
```

```
[ ] df.shape
```

```
(1025, 14)
```

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
```

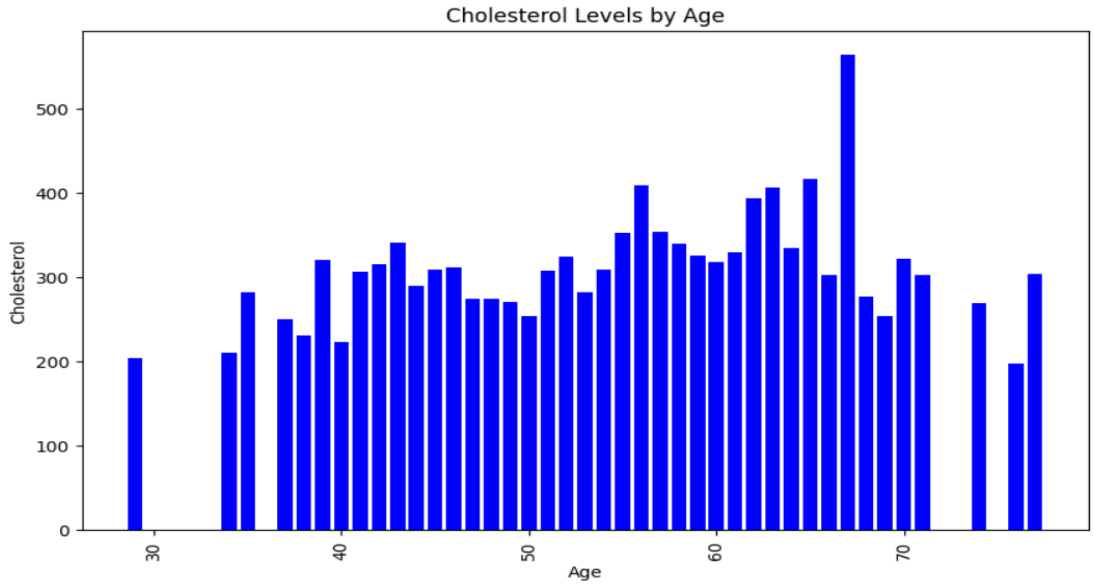
```
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

```
#exact count of target i.e total number of presence of disease and no disease
df['target'].value_counts()
```

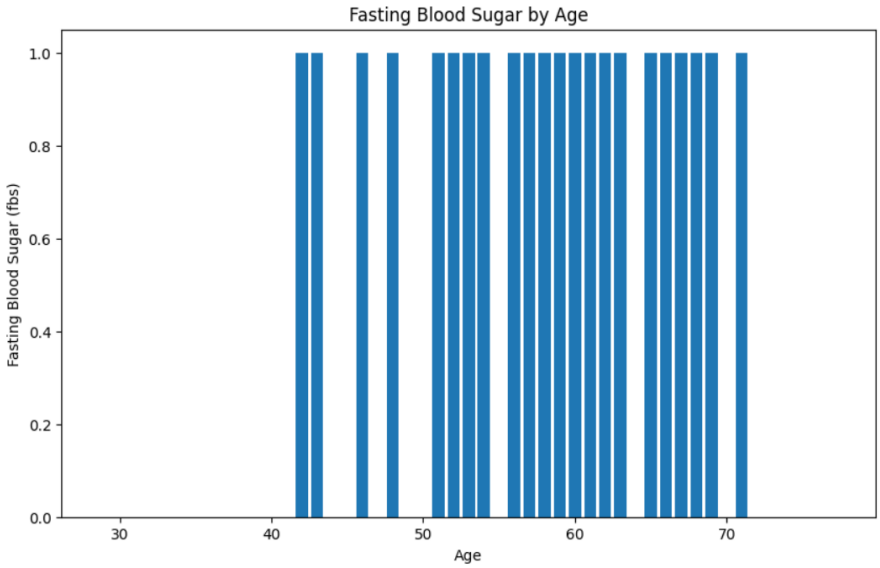
```
1    526
0    499
Name: target, dtype: int64
```

```
#Visualization
#age vs cholesterol
import pandas as pd
import matplotlib.pyplot as plt
age = df['age']
chol = df['chol']

# Plotting the bar plot
plt.figure(figsize=(10, 6))
plt.bar(age, chol, color='blue')

# Customizing the plot
plt.title('Cholesterol Levels by Age')
plt.xlabel('Age')
plt.ylabel('Cholesterol')
plt.xticks(rotation=90)
```

```
plt.xticks(rotation=90)

# Displaying the plot
plt.show()
```



```
[ ]  #age vs Fasting Blood Sugar
     plt.figure(figsize=(10, 6))
     plt.bar(df['age'], df['fbs'])
     plt.xlabel('Age')
     plt.ylabel('Fasting Blood Sugar (fbs)')
     plt.title('Fasting Blood Sugar by Age')
     plt.show()
```



```
[ ]  #considering age and cholestrol as inputs would be more accurate
     #also taking Target as output
```

```
[ ]  #divide the data into i/p and o/p
     x = df.iloc[:,0:13].values
     x
```

```
array([[52.,  1.,  0., ...,  2.,  2.,  3.],
       [53.,  1.,  0., ...,  0.,  0.,  3.],
       [70.,  1.,  0., ...,  0.,  0.,  3.],
       ...,
       [47.,  1.,  0., ...,  1.,  1.,  2.],
       [50.,  0.,  0., ...,  2.,  0.,  2.],
       [54.,  1.,  0., ...,  1.,  1.,  3.]])
```

```
[ ]  y = df.iloc[:,13].values
     y
```

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

```
[ ]  #5.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) #- 75%
     print(x_test.shape)
```

```
[ ] print(y.shape)
    print(y_train.shape) #- 75%
    print(y_test.shape) #- 25%
```

```
(1025, 13)
(768, 13)
(257, 13)
(1025,)
(768,)
(257,)
```

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

```
[ ] #7.Apply Classifier,Regressor or Clusterer
    from sklearn.linear_model import LogisticRegression
    model = LogisticRegression()
```

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

```
▾ LogisticRegression
LogisticRegression()
```

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

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

```
[ ]  y_test
```

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

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

```
85.99221789883269
```

```
[ ] #Individual Prediction1
    model.predict([[66,0,0,178,228,1,1,165,1,1,1,2,3]])
```

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

```
[ ] #Individual Prediction2
    model.predict([[58,0,0,100,248,0,0,122,0,1,1,0,2]])
```

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

```
[ ] #Individual Prediction3
    model.predict([[59,1,2,150,212,1,1,157,0,1.6,2,0,2]])
```

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

Conclusion : As we got the accurate array values from individual predictions , we can conclude that accurate logical regression on the dataset .

GITHUB ACCOUNT LINK - https://github.com/sushi41

Major project 2

INTRODUCTION :

Name - Sushmitha Santhosh

Name of College - Shree L.R.Tiwari . College Of Engineering

Year of Study- Second Year

Branch - Electronics and Computer Science

Problem statement : Choose any dataset of your choice and apply K Means Clustering .

1.About the dataset : Social network ads One of the most basic data sets to learn and implement some of the most easy and basic algorithms of machine learning and visualization
Social Network Ads A categorical dataset to determine whether a user purchased a particular product

Dataset link : https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/Social_Network_Ads.csv

2. Methodology:

1. **Data Collection**: The dataset used for this analysis was collected from a reliable source. It consists of information about 850 customers, including their age, education level, employment details, income, debt, default status, and other relevant attributes.
2. **Data Preprocessing**: The dataset was preprocessed to ensure data quality and consistency. This involved handling missing values, dropping irrelevant columns such as "Customer Id" and "Address," and applying appropriate data transformations or scaling if necessary.
3. **K-means Clustering**: K-means clustering, an unsupervised learning algorithm, was chosen to perform customer segmentation. The goal was to group similar customers together based on their attributes and identify distinct clusters within the dataset.
4. **Determining Optimal Number of Clusters:** The Elbow method was used to determine the optimal number of clusters for this analysis. The SILHOUETTE SCORE METHOD was then used to determine the number of clusters . It was identified that there are 6 types of clusters .
5. **Feature Selection**: Based on domain knowledge and data analysis, two columns were selected as input features for the K-means clustering algorithm. These columns were chosen considering their relevance to customer segmentation and potential impact on differentiating customer groups.
6. **Standardization**: The selected input features were standardized to ensure that they were on the same scale. Standardization helps prevent features with

larger magnitudes from dominating the clustering process and ensures equal importance to each feature.

7. **Applying K-means Clustering**: The K-means clustering algorithm was applied with the selected number of clusters and the preprocessed and standardized data. The algorithm iteratively assigned data points to clusters based on their proximity to cluster centroids and updated the centroids until convergence.
8. **Cluster Analysis**: After clustering, the resulting clusters were analyzed to understand their characteristics and identify any patterns or insights. Each cluster was examined based on the attributes of its members, including age, education level, employment details, income, debt, default status, and other relevant information.
9. **Final visualization** : The final visualization of clusters were plotted on a scatter plot . It was identified that a total of 6 centroids were achieved in the dataset .

3. Screenshot of code :

```
[ ]  #major project 2
     #Choose any dataset of your choice and apply K Means Clustering .
```

```
[ ]  #DATASET NAME -Social_Network_Ads
     #DATASET - https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/Social_Network_Ads.csv
```

```
[ ]  #1.take data and create dataframe
     import pandas as pd
     df = pd.read_csv('https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/Social_Network_Ads.csv')
     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, 5)
```

[ ]
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

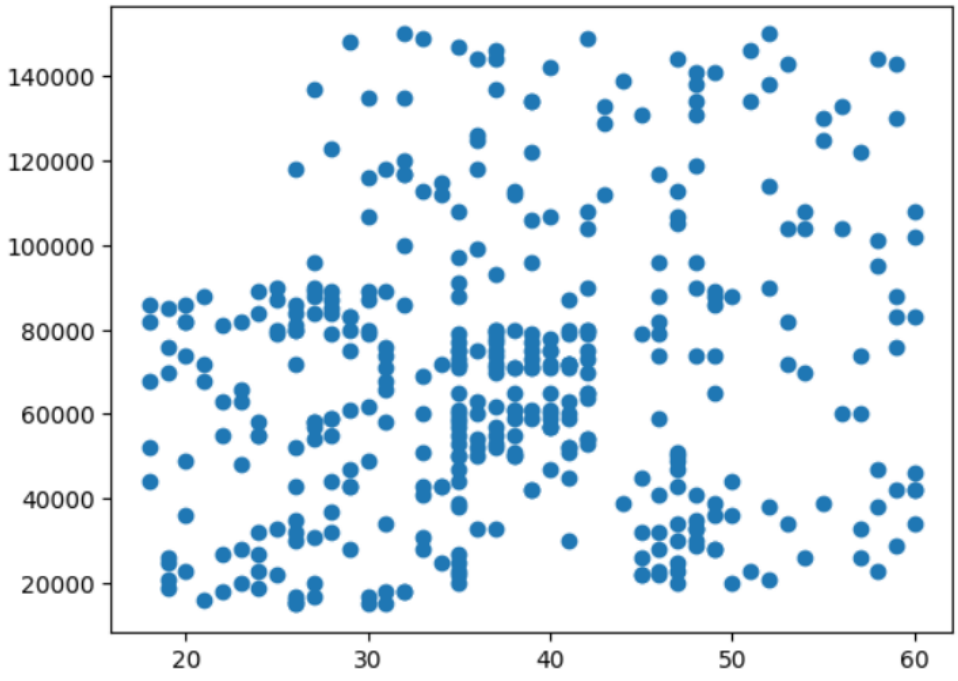
[ ]
```
#inputs- Age and Estimated salary
x = df.iloc[:,2:4].values
x
```

```
[    38,  65000],
[    47,  51000],
[    47, 105000],
[    41,  63000],
[    53,  72000],
[    54, 108000],
[    39,  77000],
[    38,  61000],
[    38, 113000],
[    37,  75000],
[    42,  90000],
[    37,  57000],
```

```
[    36,  33000],
[    49,  36000]])
```

```
[ ] #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 0x7fb61e197ca0>
```



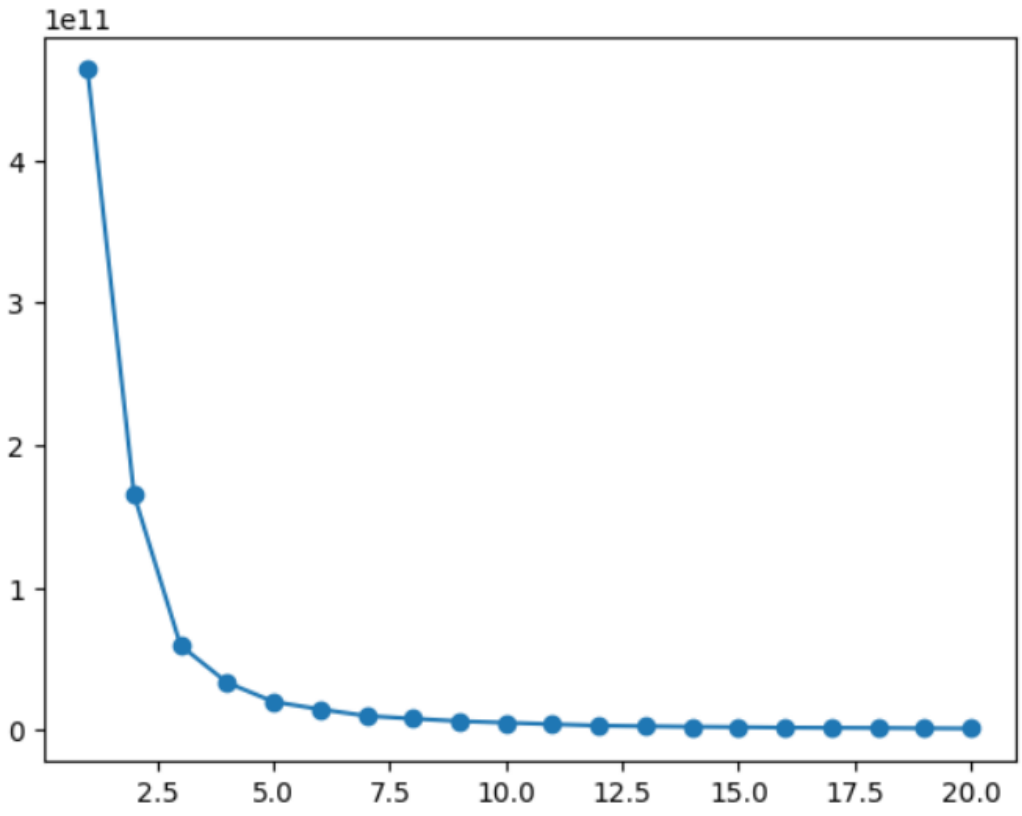
```
[ ] #Here our main task is to find out the number of clusters(k)
    import numpy as np
    X = df[['Age', 'EstimatedSalary']]
    k_range = range(1, int(np.sqrt(len(X)))+1)
    k_range
```

```
range(1, 21)
```

```
[ ] #We need to find out the number of clusters(k)
    #1.ELBOW METHOD
    #2.SILHOUETTE SCORE METHOD
```

```
[ ] #1.ELBOW METHOD
    from sklearn.cluster import KMeans
    k = range(1,21)# my range is in between 1 and 21

    sse = [] #blank list

    #for i in range(1,21):
    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)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value o
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value o
  warnings.warn(
```



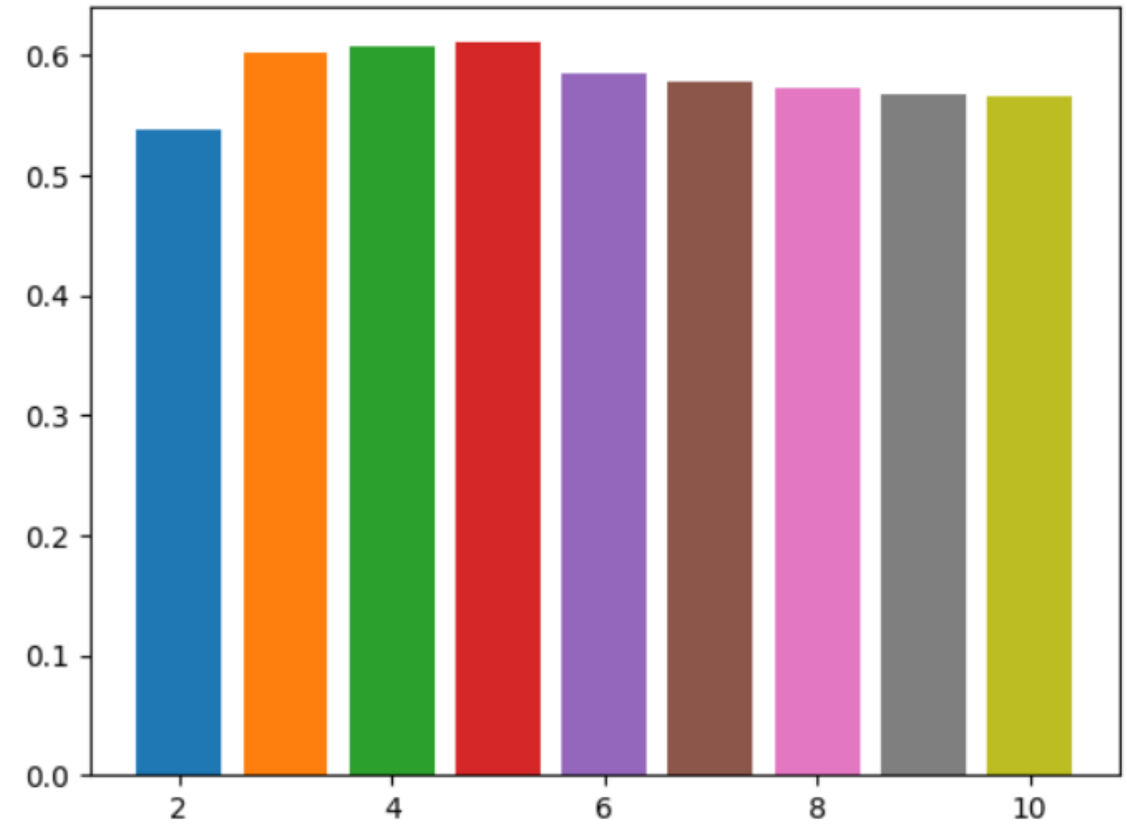
```
[ ] #2.SILHOUETTE SCORE METHOD
    k_range = range(2,11)
    silhouette_scores = []
    for k in k_range:
        kmeans = KMeans(n_clusters=k, random_state=42)
        labels = kmeans.fit_predict(X)
        score = silhouette_score(X, labels)
        silhouette_scores.append(score)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. :
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. :
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. :
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. :
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. :
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. :
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. :
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. :
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. :
  warnings.warn(
```

```
[ ] #2.SILHOUETTE SCORE METHOD
    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))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init
  warnings.warn(
2 Clusters ,Score = 0.5383447769895185
3 Clusters ,Score = 0.6014958224112057
4 Clusters ,Score = 0.6065989841357814
5 Clusters ,Score = 0.6102051324759187
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init
  warnings.warn(
6 Clusters ,Score = 0.5845746920707843
7 Clusters ,Score = 0.5771254474001397
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init
  warnings.warn(
8 Clusters ,Score = 0.5733466101369712
```

```
9 Clusters ,Score = 0.5678580889891727
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_ini
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_ini
  warnings.warn(
10 Clusters ,Score = 0.5657683924101718
```




```
[ ]  #CONFIRMATION : THE No OF CLUSTERS TO BE CONSIDERED IS 6
```

```
[ ]  #7.APPLY CLUSTERER
     k = 6
     from sklearn.cluster import KMeans

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

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: Futur
  warnings.warn(
```

```
KMeans
KMeans(n_clusters=6, random_state=0)
```

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

```
array([0, 0, 5, 2, 1, 2, 1, 4, 5, 2, 1, 2, 1, 0, 1, 1, 0, 0, 0, 0, 0, 2,
       5, 0, 0, 0, 0, 0, 5, 0, 1, 4, 0, 5, 1, 0, 0, 2, 1, 0, 0, 2, 3, 0,
       1, 0, 1, 2, 4, 1, 5, 5, 1, 0, 2, 2, 5, 1, 0, 3, 0, 1, 2, 3, 1, 2,
       0, 1, 2, 2, 1, 0, 0, 3, 0, 3, 2, 0, 1, 0, 1, 5, 2, 1, 2, 3, 2, 1,
       1, 2, 1, 3, 0, 0, 1, 5, 0, 3, 1, 5, 1, 2, 1, 4, 0, 1, 5, 1, 1, 1,
       1, 1, 2, 2, 1, 2, 1, 2, 2, 2, 1, 1, 1, 2, 2, 2, 2, 5, 0, 1, 2, 0,
       1, 2, 2, 2, 1, 3, 2, 0, 1, 2, 2, 1, 0, 1, 3, 0, 2, 1, 0, 5, 1, 2,
       5, 0, 2, 1, 0, 4, 3, 1, 5, 5, 1, 1, 2, 1, 4, 5, 1, 3, 3, 5, 1, 0,
       5, 0, 0, 5, 0, 1, 3, 5, 2, 2, 1, 5, 1, 5, 1, 0, 5, 1, 1, 5, 1, 5,
       1, 0, 5, 1, 4, 1, 3, 5, 4, 3, 4, 0, 3, 4, 5, 2, 5, 3, 2, 1, 3, 4,
       1, 1, 4, 3, 2, 2, 4, 4, 1, 1, 4, 5, 3, 1, 3, 1, 2, 1, 1, 4, 4, 2,
       1, 3, 1, 4, 2, 3, 2, 3, 5, 2, 4, 4, 5, 1, 1, 2, 3, 4, 1, 4, 4, 1,
       1, 3, 1, 1, 4, 2, 4, 1, 5, 3, 0, 1, 1, 1, 5, 5, 1, 2, 1, 0, 4, 1,
       2, 4, 1, 1, 4, 1, 5, 1, 2, 2, 1, 3, 1, 3, 5, 1, 4, 1, 2, 2, 4, 3,
       4, 2, 1, 3, 2, 4, 1, 1, 3, 2, 5, 2, 4, 1, 2, 0, 4, 2, 1, 1, 3, 3,
       2, 3, 2, 2, 2, 2, 4, 1, 2, 3, 3, 1, 2, 2, 3, 2, 1, 3, 1, 2, 3, 1,
       1, 2, 3, 5, 1, 1, 1, 2, 4, 5, 2, 1, 3, 0, 5, 1, 1, 0, 5, 1, 1, 4,
       1, 5, 1, 2, 1, 0, 2, 5, 4, 0, 5, 2, 5, 1, 5, 5, 5, 0, 5, 5, 2, 5,
       0, 0, 5, 5], dtype=int32)
```

```
[ ] y.size
```

```
400
```

```
[ ] x[y == 1,1]
    #Here the first '1' is cluster no 1 and the second '1' is column index
```

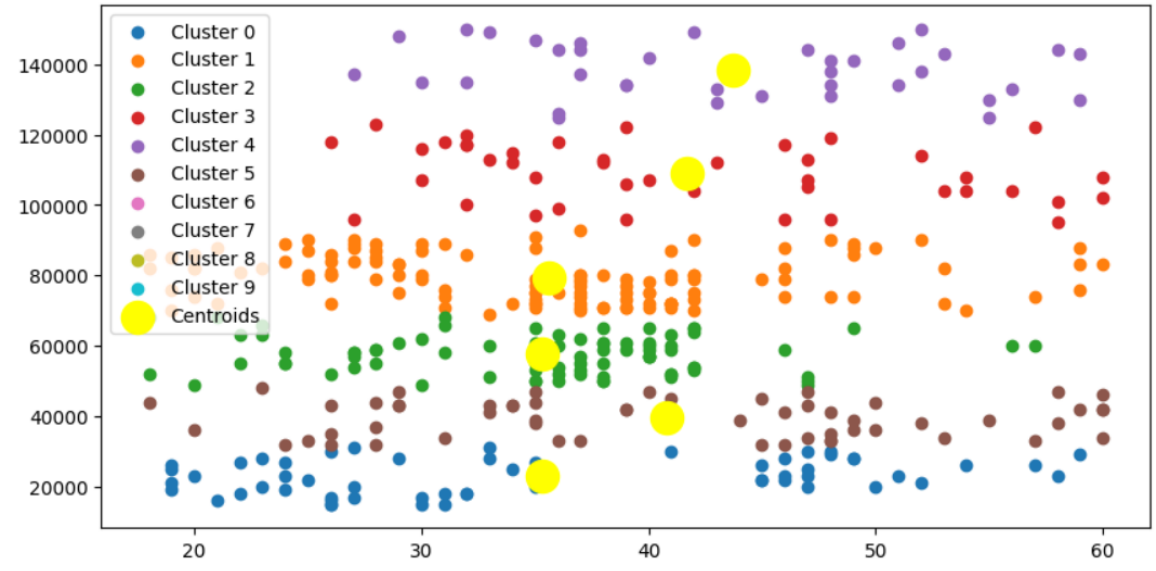
```
array([76000, 84000, 80000, 86000, 82000, 80000, 74000, 90000, 72000,
       84000, 79000, 89000, 83000, 79000, 87000, 83000, 82000, 80000,
       87000, 80000, 88000, 85000, 81000, 81000, 83000, 73000, 88000,
       86000, 72000, 89000, 86000, 80000, 71000, 71000, 80000, 75000,
       75000, 72000, 75000, 84000, 87000, 82000, 85000, 89000, 89000,
       74000, 76000, 75000, 90000, 69000, 86000, 71000, 88000, 72000,
       71000, 82000, 72000, 84000, 70000, 89000, 79000, 80000, 74000,
       71000, 78000, 80000, 91000, 72000, 80000, 86000, 79000, 80000,
       82000, 88000, 72000, 90000, 72000, 77000, 72000, 90000, 75000,
       74000, 76000, 74000, 71000, 88000, 88000, 70000, 93000, 79000,
       78000, 89000, 77000, 73000, 79000, 74000, 79000, 70000, 79000,
       75000, 82000, 72000, 75000, 79000, 75000, 72000, 77000, 75000,
       90000, 70000, 72000, 71000, 79000, 88000, 71000, 83000, 73000,
       80000, 74000, 87000, 71000])
```

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

```
(array([0, 1, 2, 3, 4, 5], dtype=int32), array([ 59, 121,  83,  43,  38,  56]))
```

```
#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 0x7fb60e3b0ee0>
```



Conclusion : As we can see from above outputs and final visualization , it can be concluded that we have performed k-means clustering from the dataset of social network ads.

GITHUB ACCOUNT LINK - https://github.com/sushi41