# ASSIGNMENT 5

Harsh Kumar

21BDS0391

harsh.kumar2021@vitstudent.ac.in

```
Assignment 5:
    Take all the columns in mall_customers.csv
    gender age annual income spending score
    perform label encoding on gender
    train your data
```

# 1. import necessary libraries.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

# 2. import dataset.

```
df = pd.read_csv("Mall_Customers.csv")
df.head()
```

| | CustomerID | Genre | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|---|
| **0** | 1 | Male | 19 | 15 | 39 |
| **1** | 2 | Male | 21 | 15 | 81 |
| **2** | 3 | Female | 20 | 16 | 6 |
| **3** | 4 | Female | 23 | 16 | 77 |
| **4** | 5 | Female | 31 | 17 | 40 |

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              200 non-null    int64 
 1   Genre                   200 non-null    object
 2   Age                     200 non-null    int64 
 3   Annual Income (k$)      200 non-null    int64 
 4   Spending Score (1-100)  200 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

# 4. Label Encoding

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
df["Genre"] = le.fit_transform(df["Genre"])
```

# 3. Select the feature to cluster.

```
# Selecting annual income and spending as features for clustering.
```

```
X = df.iloc[:, 1:].values
X
```

```
       [  0,  28,  76,  40],
       [  0,  32,  76,  87],
       [  1,  25,  77,  12],
       [  1,  28,  77,  97],
       [  1,  48,  77,  36],
       [  0,  32,  77,  74],
       [  0,  34,  78,  22],
       [  1,  34,  78,  90],
       [  1,  43,  78,  17],
       [  1,  39,  78,  88],
       [  0,  44,  78,  20],
       [  0,  38,  78,  76],
       [  0,  47,  78,  16],
       [  0,  27,  78,  89],
       [  1,  37,  78,   1],
       [  0,  30,  78,  78],
       [  1,  34,  78,   1],
       [  0,  30,  78,  73],
       [  0,  56,  79,  35],
       [  0,  29,  79,  83],
       [  1,  19,  81,   5],
       [  0,  31,  81,  93],
       [  1,  50,  85,  26],
       [  0,  36,  85,  75],
       [  1,  42,  86,  20],
       [  0,  33,  86,  95],
       [  0,  36,  87,  27],
       [  1,  32,  87,  63],
       [  1,  40,  87,  13],
       [  1,  28,  87,  75],
       [  1,  36,  87,  10],
       [  1,  36,  87,  92],
       [  0,  52,  88,  13],
       [  0,  30,  88,  86],
       [  1,  58,  88,  15],
       [  1,  27,  88,  69],
       [  1,  59,  93,  14],
       [  1,  35,  93,  90],
       [  0,  37,  97,  32],
       [  0,  32,  97,  86],
       [  1,  46,  98,  15],
       [  0,  29,  98,  88],
       [  0,  41,  99,  39],
       [  1,  30,  99,  97],
       [  0,  54, 101,  24],
       [  1,  28, 101,  68],
       [  0,  41, 103,  17],
       [  0,  36, 103,  85],
       [  0,  34, 103,  23],
       [  0,  32, 103,  69],
       [  1,  33, 113,   8],
       [  0,  38, 113,  91],
       [  0,  47, 120,  16],
       [  0,  35, 120,  79],
       [  0,  45, 126,  28],
       [  1,  32, 126,  74],
       [  1,  32, 137,  18],
       [  1,  30, 137,  83]])
```

```
type(X)
```

```
numpy.ndarray
```

## 5. Find the optimal number of clusters -- elbow method.

```
from sklearn.cluster import KMeans

# Trying different values of k and calculating WCSS for each value of k
wcss = []
for k in range(1,11):
  kmeans = KMeans(n_clusters = k, init="k-means++", random_state=0)
  kmeans.fit(X)
  wcss.append(kmeans.inertia_) #savind wcss value in a list
```

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

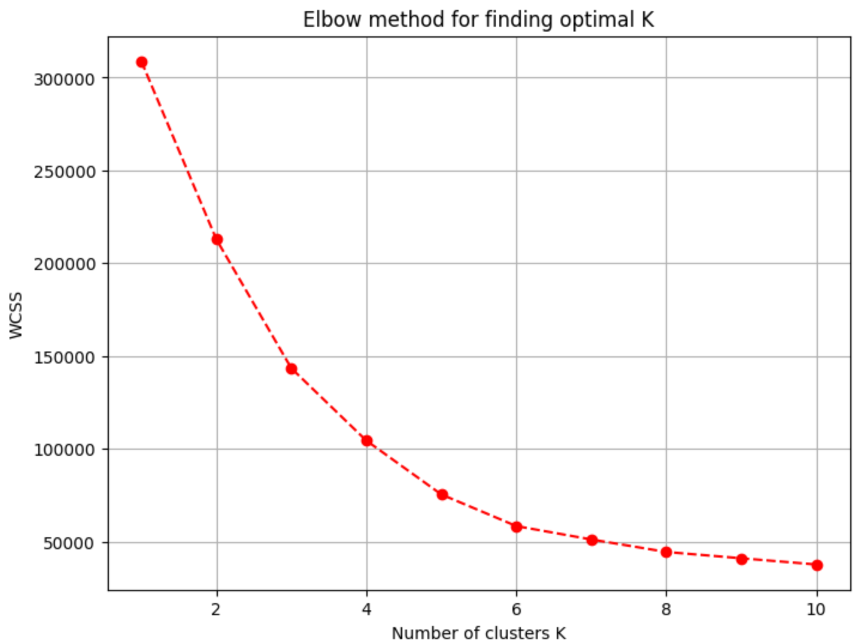
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(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

```
wcss
```

```
[308862.06000000006,
 212889.44245524303,
 143391.59236035676,
 104414.67534220168,
 75399.61541401484,
 58348.641363315044,
 51132.703212576904,
 44392.11566567935,
 41000.8742213207,
 37649.69225429742]
```

```
# PLOTTING ELBOW METHOD GRAPH

plt.figure(figsize = (8,6))
plt.plot(range(1,11), wcss, "o--", color="red")
plt.title("Elbow method for finding optimal K")
plt.xlabel("Number of clusters K")
plt.ylabel("WCSS")
plt.grid()
plt.show()
```




```
# Taking K=4 instead of K=3 coz WCSS value is low
```

## 6. Train the model on dataset using optimal cluster value k.

```
kmeans = KMeans(n_clusters = 4, init="k-means++", random_state=0) #"k-means++" means initializing the random clusters
#return a label for data based on their cluster.
Y_kmeans = kmeans.fit_predict(X)
```

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

```
Y_kmeans
```

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

```
# Age that belongs to cluster 0
X[Y_kmeans == 0,1]
```

```
array([23, 43, 59, 47, 25, 20, 44, 19, 57, 28, 25, 48, 34, 43, 44, 47, 37,
       34, 56, 19, 50, 42, 36, 40, 36, 52, 58, 59, 37, 46, 41, 54, 41, 34,
       33, 47, 45, 32])
```

```
# Age that belongs to cluster 1
X[Y_kmeans == 1,1]
```

```
array([39, 31, 40, 38, 39, 31, 29, 32, 35, 32, 32, 28, 32, 34, 39, 38, 27,
       30, 30, 29, 31, 36, 33, 32, 28, 36, 30, 27, 35, 32, 29, 30, 28, 36,
       32, 38, 35, 32, 30])
```

```
# Age that belongs to cluster 2
X[Y_kmeans == 2,1]
```

```
array([20, 31, 35, 64, 67, 58, 37, 35, 52, 35, 46, 54, 45, 40, 60, 53, 49,
       42, 36, 65, 48, 49, 50, 27, 29, 31, 49, 31, 59, 50, 47, 51, 69, 27,
       53, 70, 67, 54, 63, 43, 68, 32, 70, 47, 60, 60, 59, 26, 45, 40, 23,
       49, 57, 38, 67, 46, 21, 48, 55, 22, 34, 50, 68, 18, 48, 40, 32, 24,
       47, 27, 48, 20, 23, 49, 67, 26, 49, 21, 66, 54, 68, 66, 65, 19, 38,
       19, 18, 19, 63, 49, 51, 50, 27, 38, 40])
```

```
# Data points that belong to cluster 0
X[Y_kmeans == 0]
```

```
array([[  0,  23,  70,  29],
       [  1,  43,  71,  35],
       [  1,  59,  71,  11],
       [  1,  47,  71,   9],
       [  0,  25,  72,  34],
       [  1,  20,  73,   5],
       [  0,  44,  73,   7],
       [  1,  19,  74,  10],
       [  0,  57,  75,   5],
       [  0,  28,  76,  40],
       [  1,  25,  77,  12],
       [  1,  48,  77,  36],
       [  0,  34,  78,  22],
       [  1,  43,  78,  17],
       [  0,  44,  78,  20],
       [  0,  47,  78,  16],
       [  1,  37,  78,   1],
       [  1,  34,  78,   1],
       [  0,  56,  79,  35],
       [  1,  19,  81,   5],
       [  1,  50,  85,  26],
       [  1,  42,  86,  20],
       [  0,  36,  87,  27],
       [  1,  40,  87,  13],
       [  1,  36,  87,  10],
       [  0,  52,  88,  13],
       [  1,  58,  88,  15],
       [  1,  59,  93,  14],
       [  0,  37,  97,  32],
       [  1,  46,  98,  15],
       [  0,  41,  99,  39],
       [  0,  54, 101,  24],
       [  0,  41, 103,  17],
       [  0,  34, 103,  23],
       [  1,  33, 113,   8],
       [  0,  47, 120,  16],
       [  0,  45, 126,  28],
       [  1,  32, 137,  18]])
```