Customer Segmentation Using K-Means Clustering

1. UNDERSTANDING THE DATA :

- Importing required libraries.
- Getting the data and understanding it
- Checking for null values.
- Preparng data for clustering.

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```
df= pd.read_csv("Mall_Customers.csv")
```

In [11]:
```
df.head(10)
```

Out[11]:

	CustomerID	Gender	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
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72

In [12]:
```
df.shape
```

Out[12]:
```
(200, 5)
```

=> We have 200 rows and 5 columns

In [13]:
```
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   Gender                  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
```

=> The "df.info()" function is typically used in pandas.it provides a summary of the DataFrame, including the following information: 1.The number of rows and columns in the DataFrame. 2.The column names and their corresponding data types. 3.The number of non-null values in each column. 4.The memory usage of the DataFrame.

In [ ]:
```
=> Interpretation:
- We don't have any null valurs in this data frame.
- All of the columns have integer values except for the gender column is a categorical feature having object va
```

In [22]:
```
#retrieving all rows and columns 3(Annual income) and 4(Spending score) from the DataFrame df using integer-bas
X = df.iloc[:,[3,4]].values
```

=> The "df.iloc" attribute is used in pandas, to access and retrieve specific rows and columns from a DataFrame using integer-based indexing.The iloc attribute stands for "integer location" and allows you to access DataFrame elements using their integer positions, similar to indexing in Python lists.

2.FINDING THE OPTIMAL NUMBER OF CLUSTERS: ELBOW METHOD

- Using K-means to iterate cluters and ploting the elbow plot.
- Deciding the optimal number of clusters

In [8]:
```
from sklearn.cluster import KMeans
```

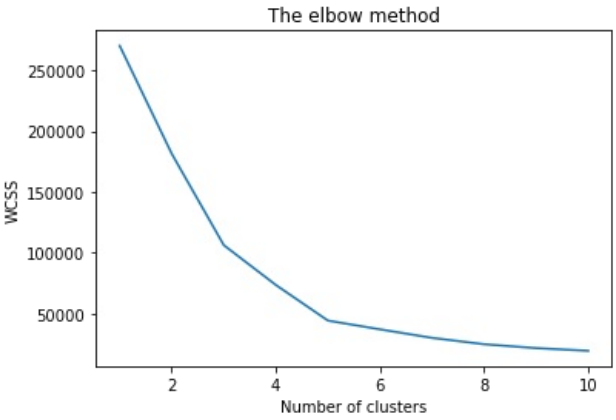
=> WCSS stands for "Within-Cluster Sum of Squares".In k-means clustering, the goal is to minimize the WCSS value. It represents the sum of the squared Euclidean distances between each data point in a cluster and its centroid

In [ ]:
```
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, max_iter = 50, n_init = 10, random_state = 0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

In [10]:
```
wcss
```

Out[10]:
```
[269981.28,
 181363.59595959593,
 106348.37306211122,
 73679.78903948836,
 44448.4554479337,
 37265.86520484346,
 30259.65720728547,
 25050.832307547527,
 21862.092672182895,
 19657.78360870395]
```

In [14]:
```
plt.plot(range(1, 11), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



=> The optimal number of clusters is equal to 5

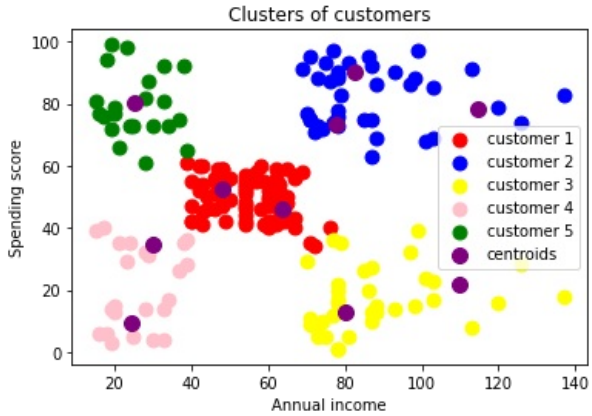
3.TRAINING THE MODEL (USING UN SUPERVISED LEARNING ALGORITHM K-MEANS):

In [17]:
```
kmeansmodel= KMeans(n_clusters=5, max_iter = 50, n_init = 10, random_state = 0)
p=kmeansmodel.fit_predict(x)
```

```
C:\Users\Client\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to hav
e a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by sett
ing the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

4. PLOTTING THE CLUSTERS:

In [21]:
```
plt.scatter(x[p==0,0], x[p==0,1], s=80, c="red", label="customer 1")
plt.scatter(x[p==1,0], x[p==1,1], s=80, c="blue", label="customer 2")
plt.scatter(x[p==2,0], x[p==2,1], s=80, c="yellow", label="customer 3")
plt.scatter(x[p==3,0], x[p==3,1], s=80, c="pink", label="customer 4")
plt.scatter(x[p==4,0], x[p==4,1], s=80, c="green", label="customer 5")
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=100, c='purple', label='centroids')
plt.title("Clusters of customers")
plt.xlabel("Annual income")
plt.ylabel("Spending score")
plt.legend()
plt.show()
```



- The green cluster on the top left side represents customers with low annual income and high spending score. - The pink cluster on the bottom left side represents customers with low annual income and spending score. - The yellow cluster on the bottom right side represents customers with high annual income and spending score. - The blue cluster on the bottom left side represents customers with high annual income and spending score. - The red cluster in the center represents customers with an average annual income and spending score.