K-Means Clustering

Importing the libraries

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

Importing the dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
# Reading the data file into a DATAFRAME and checking the shape
dataset=pd.read_csv("/content/drive/My Drive/Colab Notebooks/Mall_Customers.csv")
print(dataset.shape)
dataset
```

```
(200, 5)
```

	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
...	...	...	...	...	...

```
# calling head() method
# storing in new variable
data_top = dataset.head()
```

```
# iterating the columns
for row in data_top.index:
    print(row, end = " ")
```

```
0 1 2 3 4
```

Getting column names in Pandas dataframe

Now let’s try to get the columns name from the dataset.

Method #1: Simply iterating over columns

```
for col in dataset.columns:
    print(col)
```

```
CustomerID
Genre
Age
Annual Income (k$)
Spending Score (1-100)
```

Method #2: Using columns with dataframe object

```
# list(data) or
list(dataset.columns)
```

```
['CustomerID', 'Genre', 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']
```

Method #3: column.values method returns an array of indexes.

```
list(dataset.columns.values)
```

```
['CustomerID', 'Genre', 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']
```

Method #4: Using tolist() method with values with given the list of columns.

```
list(dataset.columns.values.tolist())
```

```
['CustomerID', 'Genre', 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']
```

Method #5: Using sorted() method

Sorted() method will return the list of columns sorted in alphabetical order.

```
# using sorted() method
sorted(dataset)
```

```
['Age', 'Annual Income (k$)', 'CustomerID', 'Genre', 'Spending Score (1-100)']
```

Select rows and columns using labels

```
dataset.loc[:,"Age"]
```

```
0      19
1      21
2      20
3      23
4      31
       ..
195    35
196    45
197    32
198    32
199    30
Name: Age, Length: 200, dtype: int64
```

```
dataset["Age"]
```

```
0      19
1      21
2      20
3      23
4      31
       ..
195    35
196    45
197    32
```

```
198    32
199    30
Name: Age, Length: 200, dtype: int64
```

```
dataset.Age
```

```
0      19
1      21
2      20
3      23
4      31
       ..
195    35
196    45
197    32
198    32
199    30
Name: Age, Length: 200, dtype: int64
```

Double-click (or enter) to edit

To select multiple columns.

Double-click (or enter) to edit

```
dataset.loc[:, ["Age", "Genre"]]
```

	Age	Genre
0	19	Male
1	21	Male
2	20	Female
3	23	Female
4	31	Female
...	...	...
195	35	Female
196	45	Female
197	32	Male

```
dataset[["Age", "Genre"]]
```

	Age	Genre
0	19	Male

Select a row by its label.

3 23 Female

```
dataset.loc[0]
```

```
CustomerID                   1
Genre                     Male
Age                         19
Annual Income (k$)          15
Spending Score (1-100)      39
Name: 0, dtype: object
```

Select multiple rows by label.

200 rows × 2 columns

```
dataset.loc[[0,1]]
```

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81

Accessing values by row and column label.

```
dataset.loc[0,"Age"]
```

```
     19
```

Accessing values from multiple columns of same row.

```
dataset.loc[1,["Age", "Genre"]]
```

```
     Age        21
     Genre    Male
     Name: 1, dtype: object
```

Select by Index Position

we can select data from a Pandas DataFrame by its location.

Note, Pandas indexing starts from zero.

Select a row by index location.

```
dataset.iloc[0]
```

```
     CustomerID                   1
     Genre                     Male
     Age                         19
     Annual Income (k$)          15
     Spending Score (1-100)      39
     Name: 0, dtype: object
```

Select a column by index location.

```
dataset.iloc[:, 3]
```

```
0       15
1       15
2       16
3       16
4       17
      ...
195    120
196    126
197    126
198    137
199    137
Name: Annual Income (k$), Length: 200, dtype: int64
```

#Select data at the specified row and column location.

```
dataset.iloc[0,3]
```

```
15
```

Select list of rows and columns.

```
dataset.iloc[[1,2],[0, 1]]
```

	CustomerID	Genre
1	2	Male
2	3	Female

Slicing Rows and Columns by position

To slice a Pandas dataframe by position use the iloc attribute.

Remember index starts from 0 to (number of rows/columns - 1).

To slice rows by index position.

```
dataset.iloc[0:2,:]
```

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81

To slice columns by index position.

```
dataset.iloc[:,1:3]
```

	Genre	Age
0	Male	19
1	Male	21
2	Female	20

To slice row and columns by index position.

... ...

```
dataset.iloc[1:2,1:3]
```

	Genre	Age
1	Male	21

```
dataset.iloc[:2,:2]
```

	CustomerID	Genre
0	1	Male
1	2	Male

Subsetting by boolean conditions

we can use boolean conditions to obtain a subset of the data from the DataFrame.

Select rows based on column value

To select all rows whose column contain the specified value(s).

```
dataset[dataset.CustomerID == 9]
```

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
8	9	Male	64	19	3

```
dataset.loc[dataset.Age == 64]
```

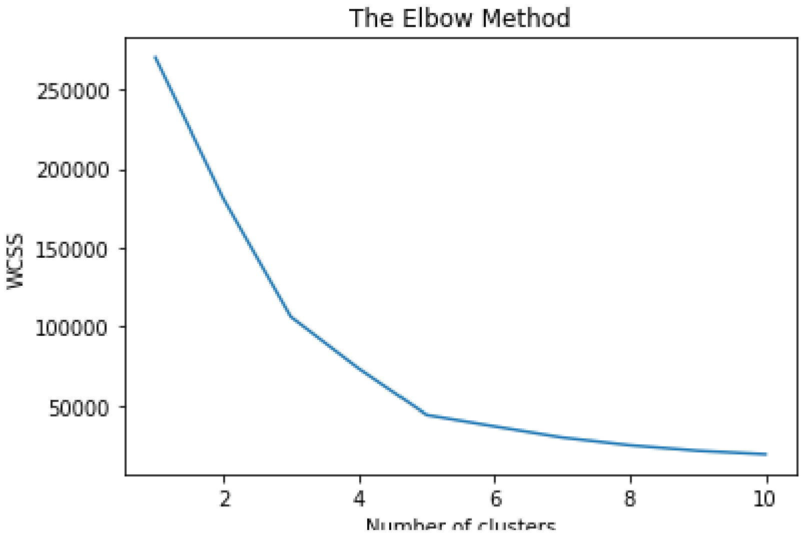
	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
8	9	Male	64	19	3

Double-click (or enter) to edit

```
X = dataset.iloc[:, [3, 4]].values
```

Using the elbow method to find the optimal number of clusters

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

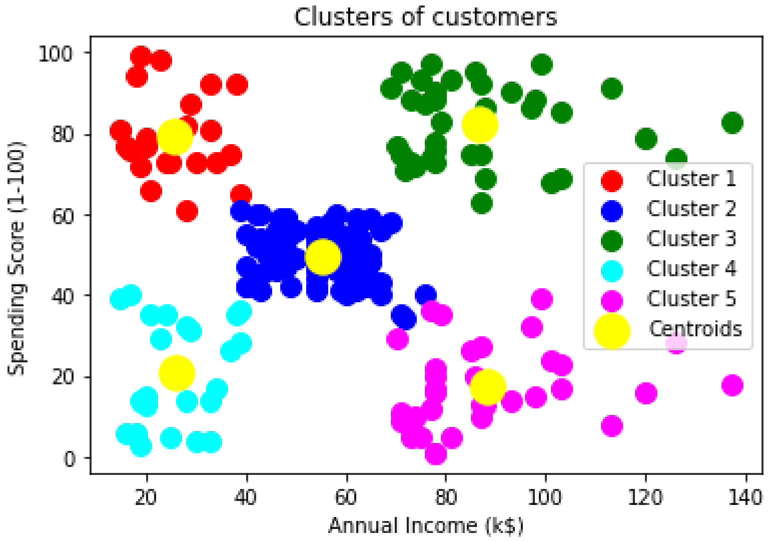


Training the K-Means model on the dataset

```
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X)
```

Visualising the clusters

```
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroids')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```


Clusters of customers
Spending Score (1-100)
100
80
60
40
20
0
20
40
60
80
100
120
140
Annual Income (k$)
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Centroids

✓ 0s completed at 9:14 PM ● ×