Today Agenda:
- Unsupervised Machine Learning
- KMeans

In [1]:

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

In [2]:

```
mall_data = pd.read_csv("https://raw.githubusercontent.com/AP-State-Skill-Development-(
mall_data.head()
```

Out[2]:

	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

In [3]:

```
mall_data.columns
```

Out[3]:

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

In [4]:

```
mall_data_dum = mall_data.drop('CustomerID',axis=1)
mall_data_dum.head()
```

Out[4]:

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

In [5]:

```
mall_data_dum.isna().sum()
```

Out[5]:

```
Genre                     0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [6]:

```
mall_data_dum.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
Genre                     200 non-null object
Age                       200 non-null int64
Annual Income (k$)        200 non-null int64
Spending Score (1-100)    200 non-null int64
dtypes: int64(3), object(1)
memory usage: 6.3+ KB
```

In [7]:

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

In [8]:

```
mall_data_dum['Genre'] = lbe.fit_transform(
    mall_data_dum['Genre'])
```

In [9]:

```
mall_data_dum['Genre'].unique()
```

Out[9]:

```
array([1, 0], dtype=int64)
```

In [10]:

```
mall_data_dum['Genre'].value_counts()
```

Out[10]:

```
0    112
1     88
Name: Genre, dtype: int64
```

In [11]:

```
1 mall_data_dum
```

Out[11]:

	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	19	15	39
1	1	21	15	81
2	0	20	16	6
3	0	23	16	77
4	0	31	17	40
5	0	22	17	76
6	0	35	18	6
7	0	23	18	94
8	1	64	19	3
9	0	30	19	72
10	1	67	19	14
11	0	35	19	99
12	0	58	20	15
13	0	24	20	77
14	1	37	20	13
15	1	22	20	79
16	0	35	21	35
17	1	20	21	66
18	1	52	23	29
19	0	35	23	98
20	1	35	24	35
21	1	25	24	73
22	0	46	25	5
23	1	31	25	73
24	0	54	28	14
25	1	29	28	82
26	0	45	28	32
27	1	35	28	61
28	0	40	29	31
29	0	23	29	87
...	...	...	...	...
170	1	40	87	13
171	1	28	87	75
172	1	36	87	10

	Genre	Age	Annual Income (k$)	Spending Score (1-100)
173	1	36	87	92
174	0	52	88	13
175	0	30	88	86
176	1	58	88	15
177	1	27	88	69
178	1	59	93	14
179	1	35	93	90
180	0	37	97	32
181	0	32	97	86
182	1	46	98	15
183	0	29	98	88
184	0	41	99	39
185	1	30	99	97
186	0	54	101	24
187	1	28	101	68
188	0	41	103	17
189	0	36	103	85
190	0	34	103	23
191	0	32	103	69
192	1	33	113	8
193	0	38	113	91
194	0	47	120	16
195	0	35	120	79
196	0	45	126	28
197	1	32	126	74
198	1	32	137	18
199	1	30	137	83

200 rows × 4 columns

Check types of scallers and know about each scaller

In [12]:

```
from sklearn.model_selection import train_test_split
data_train,data_test = train_test_split(mall_data_dum,
                                        random_state=2,
                                        test_size=0.25)
```

In [13]:

```
from sklearn.cluster import KMeans
km = KMeans(n_clusters=4)
```

In [14]:

```
km.fit(data_train)
```

Out[14]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [15]:

```
help(km.fit(data_train))
```

```
 |  cluster_centers_ : array, [n_clusters, n_features]
 |      Coordinates of cluster centers. If the algorithm stops before full
y
 |      converging (see ``tol`` and ``max_iter``), these will not be
 |      consistent with ``labels_``.
 |
 |  labels_ :
 |      Labels of each point
 |
 |  inertia_ : float
 |      Sum of squared distances of samples to their closest cluster cente
r.
 |
 |  n_iter_ : int
 |      Number of iterations run.
 |
 |  Examples
 |  --------
 |
```

In [16]:

```
pred = km.predict(data_train)
pred
```

Out[16]:

```
array([1, 0, 0, 1, 3, 3, 3, 3, 2, 1, 3, 1, 3, 3, 1, 3, 2, 3, 3, 1, 2, 2,
       2, 3, 0, 3, 2, 1, 2, 2, 1, 3, 1, 1, 3, 0, 0, 1, 3, 3, 3, 2, 2, 2,
       3, 3, 2, 0, 3, 1, 0, 1, 1, 1, 3, 1, 0, 3, 3, 2, 0, 0, 0, 1, 3, 2,
       1, 2, 2, 0, 3, 0, 3, 0, 3, 1, 2, 1, 0, 0, 0, 1, 3, 0, 3, 3, 3, 3,
       3, 2, 3, 1, 3, 3, 1, 3, 3, 3, 0, 3, 3, 2, 1, 1, 2, 2, 0, 0, 2, 2,
       3, 0, 3, 1, 3, 1, 3, 0, 0, 3, 1, 0, 0, 3, 0, 0, 3, 2, 0, 3, 1, 0,
       3, 2, 2, 0, 3, 0, 2, 0, 1, 2, 0, 3, 0, 0, 3, 3, 0, 2])
```

In [17]:

```
pred_test = km.predict(data_test)
pred_test
```

Out[17]:

```
array([3, 0, 2, 1, 1, 3, 3, 3, 0, 0, 3, 3, 0, 2, 2, 3, 3, 0, 0, 3, 0, 0,
       3, 2, 1, 1, 0, 0, 0, 3, 2, 3, 0, 3, 0, 3, 2, 2, 1, 3, 1, 1, 0, 1,
       3, 1, 0, 2, 2, 0])
```

Task:

- Take cust_segmentation dataset and apply kmeans algorith
- Work on different types of Scallers

In [ ]:

```

```