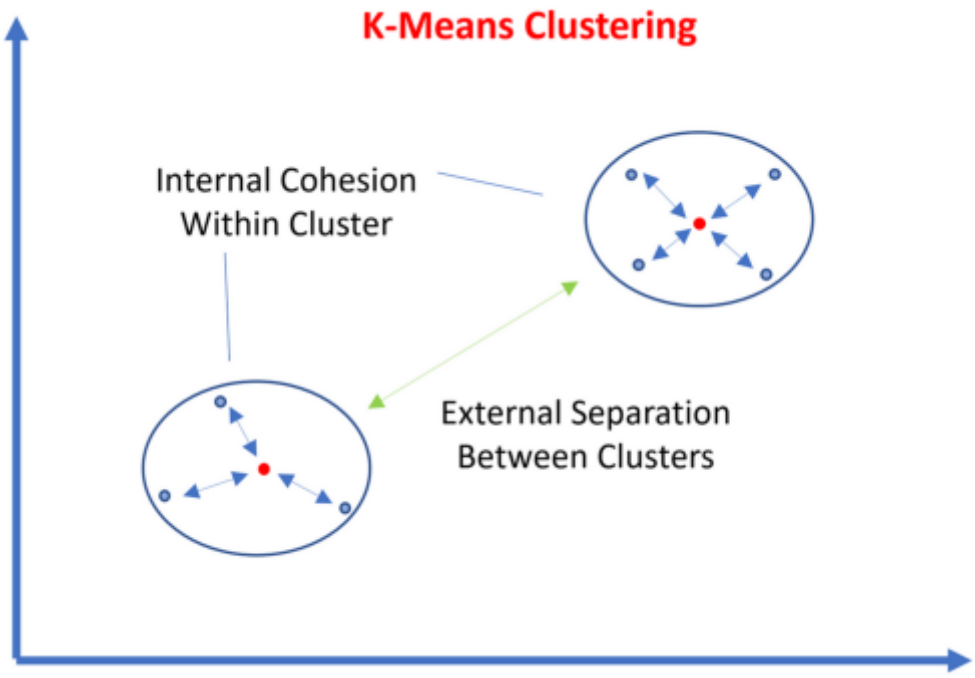
What is K-Means? The objective of K-Means is to put data points with similar characteristics in the same cluster (i.e., internal cohesion) and separate data points with different characteristics into different clusters (i.e., external separation).

Intra-cluster variance (a.k.a., the squared error function or sum of squares within (SSW) or sum of squares error (SSE)) is used to quantify internal cohesion. It is defined as the sum of the squared distance between the average point (called Centroid) and each point of the cluster. The smaller the value, the better the clustering is.

Inter-cluster variance (a.k.a, Sum of squares Between (SSB))is used to quantify external separation. It is defined as the sum of the squared distance between the global average point and each Centroid. The bigger the value, the better the clustering is.

In [15]:



Business Problem:

Live Shopping on Facebook is an interactive way to sell items, connect straight with viewers, and gain likely customers, all in real-time. When you sell products through Live Shopping on Facebook, you are live streaming as you feature and demonstrate your products. Small vendors can now reach a more expansive audience and connect with many clients. You need to implement K-Means clustering to find intrinsic batches within the dataset that depict the same status_type behavior. The status_type behavior variable consists of posts of a distinct nature (video, photos, statuses, and links).

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import *
```

In [2]:
```
df= pd.read_csv('Live.csv')
```

In [3]:
```
df.head(5)
```

Out[3]:

	status_id	status_type	status_published	num_reactions	num_cc
0	246675545449582_1649696485147474	video	4/22/2018 6:00	529	
1	246675545449582_1649426988507757	photo	4/21/2018 22:45	150	
2	246675545449582_1648730588577397	video	4/21/2018 6:17	227	
3	246675545449582_1648576705259452	photo	4/21/2018 2:29	111	
4	246675545449582_1645700502213739	photo	4/18/2018 3:22	213	

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7050 entries, 0 to 7049
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   status_id         7050 non-null   object 
 1   status_type       7050 non-null   object 
 2   status_published  7050 non-null   object 
 3   num_reactions     7050 non-null   int64  
 4   num_comments      7050 non-null   int64  
 5   num_shares        7050 non-null   int64  
 6   num_likes         7050 non-null   int64  
 7   num_loves         7050 non-null   int64  
 8   num_wows          7050 non-null   int64  
 9   num_hahas         7050 non-null   int64  
 10  num_sads          7050 non-null   int64  
 11  num_angrys        7050 non-null   int64  
 12  Column1           0 non-null      float64
 13  Column2           0 non-null      float64
 14  Column3           0 non-null      float64
 15  Column4           0 non-null      float64
dtypes: float64(4), int64(9), object(3)
memory usage: 881.4+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows
count	7050.000000	7050.000000	7050.000000	7050.000000	7050.000000	7050.000000
mean	230.117163	224.356028	40.022553	215.043121	12.728652	1.289362
std	462.625309	889.636820	131.599965	449.472357	39.972930	8.719650
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	17.000000	0.000000	0.000000	17.000000	0.000000	0.000000
50%	59.500000	4.000000	0.000000	58.000000	0.000000	0.000000
75%	219.000000	23.000000	4.000000	184.750000	3.000000	0.000000
max	4710.000000	20990.000000	3424.000000	4710.000000	657.000000	278.000000

In [6]:
```
df.drop(columns = ['Column1', 'Column2', 'Column3','Column4' ], inplace = Tr
```

In [7]:
```
# Convert the date and time to a pandas datetime object
df['status_published'] = pd.to_datetime(df['status_published'])

#How many likes have been given to the photo posted on 4/19/2018 at 22:26?

# Filter the DataFrame for the specified date and time and for photo posts
filtered_df = df[(df['status_published'].dt.date == pd.to_datetime('2018-04
                  (df['status_published'].dt.hour == 22) &
                  (df['status_published'].dt.minute == 26) &
                  (df['status_type'] == 'photo')]

# Calculate the total number of likes for the filtered photo posts
total_likes_photo_4226 = filtered_df['num_likes'].sum()

print("Total number of likes for the photo posted on 4/19/2018 at 22:26:",
```

```
Total number of likes for the photo posted on 4/19/2018 at 22:26: 379
```

Perform the following operations on the dataset:
- Drop Status id and status published column.
- Use a label encoder to encode the status type column.
- Standardize the data using min-max scalar
- Create a K means model for 2 clusters.

From the above model, what is the inter-cluster variance of the model?

In [8]:
```
df.drop(columns = ['status_id', 'status_published' ], inplace = True)
```

In [9]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['status_type'] = le.fit_transform(df['status_type'])
```

In [10]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df)
```

In [11]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2)
kmeans.fit(scaled_data)
```

```
C:\Users\SIMRAN\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:87
0: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

Out[11]:

```
KMeans
KMeans(n_clusters=2)
```

In [12]:
```
kmeans.inertia_
```

Out[12]:
```
237.75726404419547
```

How to train the K-means model?

- fit()