# ML LAB 07 - K Means Clustering


### Name : Tulasi Raman R

### Register Number : 21MIS1170


# Importing Libraries

In [69]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline


from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import seaborn as sns
from matplotlib import pyplot as pl
import numpy as np
from sklearn import metrics
```

# Import Dataset

In [70]:
```
df = pd.read_csv("Live.csv")
df.head()
```

Out[70]:

| | status_id | status_type | status_published | num_reactions | nu |
|---|---|---|---|---|---|
| **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 [71]:
```
df.shape
```

Out[71]:
```
(7050, 16)
```

In [72]:
```
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 [73]:
```
df.isnull().sum()
```

Out[73]:
```
status_id              0
status_type            0
status_published       0
num_reactions          0
num_comments           0
num_shares             0
num_likes              0
num_loves              0
num_wows               0
num_hahas              0
num_sads               0
num_angrys             0
Column1             7050
Column2             7050
Column3             7050
Column4             7050
dtype: int64
```

## Fortunately we don't have any null values in between the dataset. But in Column1, Column2, Column3 and Column4, it is found that the entire coulmn has missing values. So we'll these columns.

In [74]:
```
df.drop(['Column1', 'Column2', 'Column3', 'Column4'], axis=1, inplace=True)
```

In [75]:
```
df.drop(['status_id', 'status_published'], axis=1, inplace=True)
```

- 'status_id' is removed because it only has 4 uinque vairables.
- 'status_published' is a string and cannot be converted to numerical datatype as it an approxiamately unique identifier

In [76]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7050 entries, 0 to 7049
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   status_type     7050 non-null   object
 1   num_reactions   7050 non-null   int64 
 2   num_comments    7050 non-null   int64 
 3   num_shares      7050 non-null   int64 
 4   num_likes       7050 non-null   int64 
 5   num_loves       7050 non-null   int64 
 6   num_wows        7050 non-null   int64 
 7   num_hahas       7050 non-null   int64 
 8   num_sads        7050 non-null   int64 
 9   num_angrys      7050 non-null   int64 
dtypes: int64(9), object(1)
memory usage: 550.9+ KB
```

# Converting Categorical to Numerical Data for Clustering

In [77]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

df['status_type'] = le.fit_transform(df['status_type'])
```

In [78]:
```
df.head()
```

Out[78]:

| | status_type | num_reactions | num_comments | num_shares | num_likes | num_loves | nur |
|---|---|---|---|---|---|---|---|
| **0** | 3 | 529 | 512 | 262 | 432 | 92 | |
| **1** | 1 | 150 | 0 | 0 | 150 | 0 | |
| **2** | 3 | 227 | 236 | 57 | 204 | 21 | |
| **3** | 1 | 111 | 0 | 0 | 111 | 0 | |
| **4** | 1 | 213 | 0 | 0 | 204 | 9 | |

# K means with two clusters

In [79]:
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, random_state=0)
kmeans.fit(df)

y_predicted = kmeans.fit_predict(df)
y_predicted
```

Out[79]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [80]:
```
df['cluster']=y_predicted
```

In [81]:
```
kmeans.cluster_centers_
```

Out[81]:
```
array([[1.67882472e+00, 2.24142278e+02, 1.22109857e+02, 3.08061948e+01,
        2.11227529e+02, 1.09083804e+01, 1.15298886e+00, 5.56809958e-01,
        1.97423650e-01, 9.63960052e-02],
       [2.98581560e+00, 5.22886525e+02, 5.23441844e+03, 4.91624113e+02,
        4.02007092e+02, 1.01921986e+02, 7.97163121e+00, 7.53900709e+00,
        2.51063830e+00, 9.36170213e-01]])
```

In [82]:
```
kmeans.inertia_
```

Out[82]:
```
4986397875.914604
```

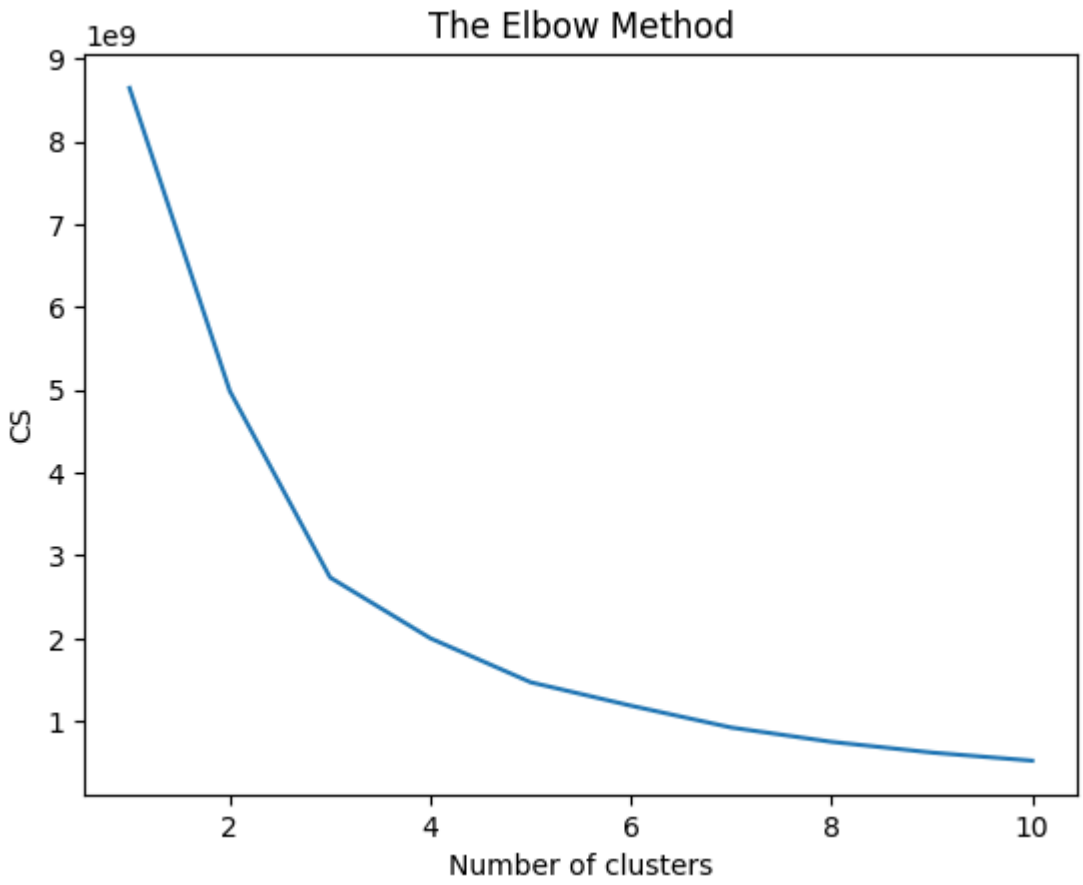
- K-means algorithm aims to choose centroids that minimize the inertia, or within-cluster sum of squared criterion.
- The lesser the model inertia, the better the model fit. Here we have high value, so not a good method.

In [83]:
```
reduced_data = PCA(n_components=2).fit_transform(df)
results = pd.DataFrame(reduced_data,columns=['pca1','pca2'])
pal = ["#29487D", "#9CB2CE"]

sns.scatterplot(x="pca1", y="pca2", hue=df['cluster'], data=results, palette=pal
plt.title('KMeans Clustering with 2 clusters')
plt.show()
```

# Elbow method

In [84]:
```
from sklearn.cluster import KMeans
cs = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init =
    kmeans.fit(df)
    cs.append(kmeans.inertia_)
plt.plot(range(1, 11), cs)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('CS')
plt.show()
```




- By the above plot, we can see that there is a kink at k=3.
- Hence k=3 can be considered a good number of the cluster to cluster this data.

# K Means with 3 Clusters

In [85]:
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3, random_state=0)
kmeans.fit(df)
```

```
y_predicted = kmeans.fit_predict(df)
y_predicted
```

Out[85]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [86]:
```
df['cluster']=y_predicted
```

In [87]:
```
kmeans.cluster_centers_
```

Out[87]:
```
array([[1.66884430e+00, 1.13000000e+02, 1.08951003e+02, 2.47901695e+01,
        1.03412350e+02, 8.33551097e+00, 5.78783637e-01, 4.09706020e-01,
        1.82454503e-01, 7.82392285e-02, 1.55545186e-04],
       [2.98540146e+00, 4.86737226e+02, 5.30959124e+03, 4.80408759e+02,
        3.76072993e+02, 9.50729927e+01, 4.80291971e+00, 7.39416058e+00,
        2.51824818e+00, 8.75912409e-01, 1.00000000e+00],
       [1.82231405e+00, 1.71315289e+03, 3.17871901e+02, 1.17700413e+02,
        1.65226033e+03, 4.77747934e+01, 9.73347107e+00, 2.60950413e+00,
        4.13223140e-01, 3.61570248e-01, 6.19834711e-03]])
```

In [88]:
```
kmeans.inertia_
```

Out[88]:
```
2735263889.752016
```

In [89]:
```
reduced_data = PCA(n_components=2).fit_transform(df)
results = pd.DataFrame(reduced_data,columns=['pca1','pca2'])
pal = ["#29487D", "#9CB2CE", "#83f5c5"]

sns.scatterplot(x="pca1", y="pca2", hue=df['cluster'], data=results, palette=pal
plt.title('KMeans Clustering with 3 clusters')
plt.show()
```

# Model Interpretation

- The dataset is unlabelled and we need to find the similarities between the dataset. So we go for unsupervised clustering methods such as K means.
- Intially we tried with k=2, but later we concluded that the optimal number of clusters are k=3 using Elbow method by noticing a major kink at k = 3.
- We after moving to k=3 clusters, we found that the inietria = 2735263889.752016 is still extremely high. Lower the initeria, better the model is, so we can conclude that the model is not a good fit.