```
#1.taken data and creat datarame
import pandas as pd
df=pd.read_csv('/content/Live_20210128.csv')
df
```

	status_id	status_type	status_published	num_reactions	num_comments	num_shar
0	1	video	4/22/2018 6:00	529	512	2
1	2	photo	4/21/2018 22:45	150	0	
2	3	video	4/21/2018 6:17	227	236	
3	4	photo	4/21/2018 2:29	111	0	
4	5	photo	4/18/2018 3:22	213	0	
...	...	...	...	...	...	
7045	7046	photo	9/24/2016 2:58	89	0	
7046	7047	photo	9/23/2016 11:19	16	0	
7047	7048	photo	9/21/2016 23:03	2	0	
7048	7049	photo	9/20/2016 0:43	351	12	
7049	7050	photo	9/10/2016 10:30	17	0	

7050 rows × 16 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.shape
```

```
(7050, 16)
```

```
df.size
```

```
112800
```

```
df.info()#view summary of dataset
```

```
<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   int64
 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(10), object(2)
memory usage: 881.4+ KB
```

```
df.head()
```

	status_id	status_type	status_published	num_reactions	num_comments	num_shares
0	1	video	4/22/2018 6:00	529	512	262
1	2	photo	4/21/2018 22:45	150	0	0
2	3	video	4/21/2018 6:17	227	236	57
3	4	photo	4/21/2018 2:29	111	0	0
4	5	photo	4/18/2018 3:22	213	0	0

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.isnull().sum()#checking for missing values
```

	0
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

```
#removeinf the last Column1 Column2 Column3 Column4
df.drop(['Column1','Column2','Column3','Column4'],axis=1,inplace=True)


df.info()# now we can see the last 4 columns has removed ..
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7050 entries, 0 to 7049
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   status_id         7050 non-null   int64 
 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 
dtypes: int64(10), object(2)
memory usage: 661.1+ KB
```

```
#view the statistical summary of numerical variables
df.describe()
```

	status_id	num_reactions	num_comments	num_shares	num_likes	num_loves
count	7050.000000	7050.000000	7050.000000	7050.000000	7050.000000	7050.000000
mean	3525.500000	230.117163	224.356028	40.022553	215.043121	12.728652
std	2035.304031	462.625309	889.636820	131.599965	449.472357	39.972930
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1763.250000	17.000000	0.000000	0.000000	17.000000	0.000000
50%	3525.500000	59.500000	4.000000	0.000000	58.000000	0.000000
75%	5287.750000	219.000000	23.000000	4.000000	184.750000	3.000000
max	7050.000000	4710.000000	20990.000000	3424.000000	4710.000000	657.000000

```
#eploreing the (status_id variable)
df['status_id'].unique()
```

```
array([   1,    2,    3, ..., 7048, 7049, 7050])
```

```
len(df['status_id'].unique())
#we can see that there are 7050 uniue labels .so it is approximately a unique idntifier f
#hences.i will drop it
```

```
7050
```

```
#Expore ( status_published)variable
df['status_published'].unique()
```

```
array(['4/22/2018 6:00', '4/21/2018 22:45', '4/21/2018 6:17', ...,
       '9/21/2016 23:03', '9/20/2016 0:43', '9/10/2016 10:30'],
      dtype=object)
```

```
len(df['status_published'].unique())
#it is approximately a unique identifier .hence i will drop it also
```

```
6913
```

```
#explore (status_type ) variable
df['status_type'].unique()
```

```
array(['video', 'photo', 'link', 'status'], dtype=object)
```

```
len(df['status_type'].unique())
#we see that there are 4 ctegories of labels in the status_type  variable
```

```
4
```

```
#dropping the status_id and status_publlishe variable form the dataset
df.drop(['status_id','status_published'],axis=1,inplace=True)
```



```
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
```

```
df.head()#After droping the data viewing the dataset
```

	status_type	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wo
0	video	529	512	262	432	92	
1	photo	150	0	0	150	0	
2	video	227	236	57	204	21	
3	photo	111	0	0	111	0	
4	photo	213	0	0	204	9	


```
#declare feature vector and target variable
x=df
y=df['status_type']
```

```
#convert categorical variable into integers
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
x['status_type'] = le.fit_transform(x['status_type'])
y = le.transform(y)
```

```
x.head()
```

	status_type	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wo
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	

Next steps: Generate code with x View recommended plots New interactive sheet

```
cols = x.columns#feature scaling
```

```
from sklearn.preprocessing import MinMaxScaler
ms = MinMaxScaler()
x = ms.fit_transform(x)
```

```
x = pd.DataFrame(x, columns=[cols])
```

```
x.head()
```

	status_type	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wo
0	1.000000	0.112314	0.024393	0.076519	0.091720	0.140030	0.0107
1	0.333333	0.031847	0.000000	0.000000	0.031847	0.000000	0.0000
2	1.000000	0.048195	0.011243	0.016647	0.043312	0.031963	0.0035
3	0.333333	0.023567	0.000000	0.000000	0.023567	0.000000	0.0000
4	0.333333	0.045223	0.000000	0.000000	0.043312	0.013699	0.0000

Next steps: Generate code with x View recommended plots New interactive sheet

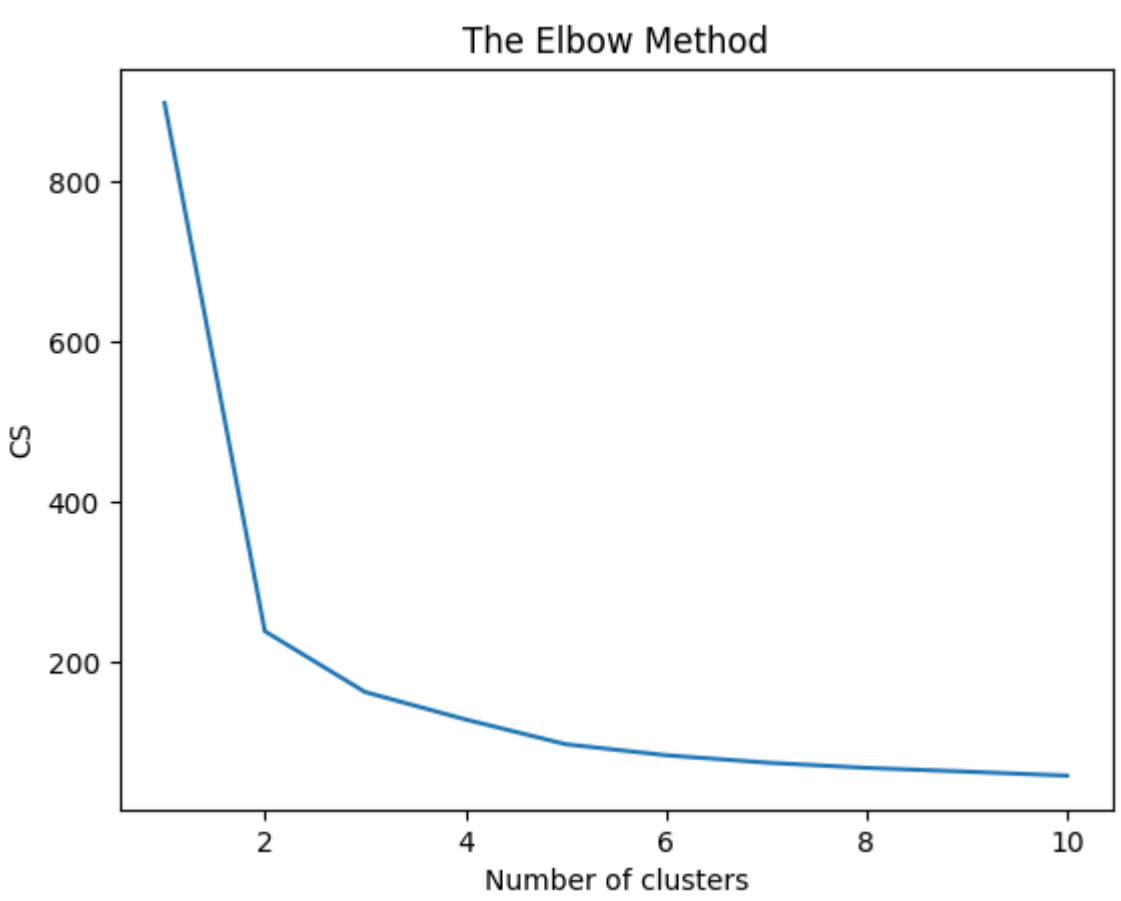
```
#k-mean model with two clusters
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2, random_state=0)
kmeans.fit(x)
```

```
KMeans
KMeans(n_clusters=2, random_state=0)
```

```
#k-means model parameters study
kmeans.cluster_centers_
```

```
array([[9.54921576e-01, 6.46330441e-02, 2.67028654e-02, 2.93171709e-02,
        5.71231462e-02, 4.71007076e-02, 8.18581889e-03, 9.65207685e-03,
        8.04219428e-03, 7.19501847e-03],
       [3.28506857e-01, 3.90710874e-02, 7.54854864e-04, 7.53667113e-04,
        3.85438884e-02, 2.17448568e-03, 2.43721364e-03, 1.20039760e-03,
        2.75348016e-03, 1.45313276e-03]])
```

```
#useing elbow method to find optimal number of clusters
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 = 10
    kmeans.fit(x)
    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()
```




```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2,random_state=0)
kmeans.fit(x)
labels = kmeans.labels_
# check how many of the samples were correctly labeled
correct_labels = sum(y == labels)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Result: 4288 out of 7050 samples were correctly labeled.
Accuracy score: 0.61
```

```
#K-Means model with 3 clusters
kmeans.fit(x)
# check how many of the samples were correctly labeled
labels = kmeans.labels_
correct_labels = sum(y == labels)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Result: 4288 out of 7050 samples were correctly labeled.
Accuracy score: 0.61
```

```
kmeans = KMeans(n_clusters=4, random_state=0)
kmeans.fit(x)
# check how many of the samples were correctly labeled
labels = kmeans.labels_
correct_labels = sum(y == labels)
```

```
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size)
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Result: 4112 out of 7050 samples were correctly labeled.
Accuracy score: 0.58
```