Table of Contents

1. Title : Analysis of Engagement patterns for Facebook Live sellers in Thailand using K-means Clustering

Student Name: Kirti Bendigeri Student ID: 01967822

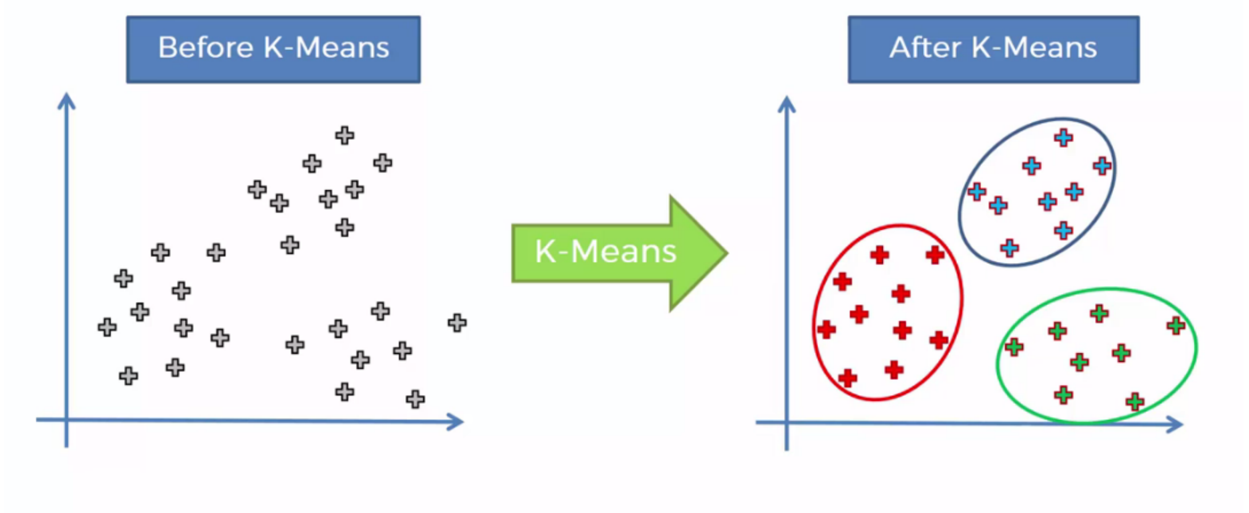
2. Abstract

This project report describes a Comma Separated Values (CSV) dataset consisting of 7050 Facebook posts of various types (text, deferred and live videos, images). These posts were extracted from the Facebook pages of 10 Thai fashion and cosmetics retail sellers from March 2012, to June 2018. For each Facebook post, the dataset records the resulting engagement metrics comprising shares, comments, and emoji reactions within which we distinguish traditional “likes” from recently introduced emoji reactions, that are “love”, “wow”, “haha”, “sad” and “angry”.This dataset could serve as a basis for research on customer engagement with the novel sales channel that is Facebook Live, through comparative studies with other forms of content, as well as the statistical analysis of the seasonality of engagement and outlier posts.We will use k-means clustering algorithm to find the number of clusters in the data and run the algorithm for different values of K and try to find the best higher classification accuracy of the model.

3. Introduction And Background

Before the advent of live streaming, statistical studies of customer engagement associated with Facebook posts had a common observation pattern is that photos were the most commonly used medium and typically generated the most likes and comments, followed by videos. In addition to traditional types of posts, the dataset we have includes live videos. For each individual post (rows), the columns of the dataset record the type of posts, their date, and engagement metrics comprising shares, comments, and emoji reactions within which we distinguish traditional "likes" from recently introduced emoji reactions, that are "love", "wow", "haha", "sad" and "angry", reflecting more varied sentiments than the more neutral "like". Descriptive statistics of the engagement metrics per post, for the Facebook pages of the 10 sellers considered in the dataset. For each seller, this table presents the mean, standard deviation and maximum value of the considered engagement metrics.

K-Means clustering is one of the methods that can be used when we have unlabelled data which is data without defined categories or groups. The algorithm follows an easy or simple way to classify a given data set through a certain number of clusters. It works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. The K-Means algorithm depends upon finding the number of clusters and data labels for a pre-defined value of K. To find the number of clusters in the data, we need to run the K-Means clustering algorithm for different values of K and compare the results. So, the performance of K-Means algorithm depends upon the value of K. We should choose the optimal value of K that gives us best performance. There are different techniques available to find the optimal value of K. The most common technique is the elbow method. The K-Means clustering algorithm uses an iterative procedure to deliver a final result. The algorithm requires number of clusters K and the data set as input. The data set is a collection of features for each data point.



4.Methods

I am using python programming language to perform all the operations in this project. I am using python because it has lot of modules that can be used to perform statistical operations and data manipulation. We will be using k-means clustering, a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

The following are the steps used to do this project: i. Import the libraries ii. Load the data sets iii. Data cleaning iv. Data Analysis v. Produce plots and Graphs vi. Model fitting vii. K means clustering viii. Predict the outcomes

In [11]:
```
#import libraries
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # for data visualization
import seaborn as sns # for statistical data visualization
%matplotlib inline
import os
for dirname, _, filenames in os.walk('/Users/mad_over_minions/Desktop/
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [12]:
```
#ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

In [13]:
```
#Import dataset
data = '/Users/sriramj98/Desktop/kmeans/Live_20210128.csv'

df = pd.read_csv(data)
```

In [27]:
```
#check the shape of dataset
df.shape
```

Out[27]:
```
(7050, 12)
```

We can see that there are 7050 instances and 16 attributes in the dataset. In the dataset description, it is given that there are 7051 instances and 12 attributes in the dataset.So, we can infer that the first instance is the row header and there are 4 extra attributes in the dataset.

In [28]:
```
df.head() #preview the data set
```

Out[28]:

	status_id	status_type	status_published	num_reactions	num_comments	num_shares	num_
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	

In [29]:
```
df.info() #summary of dataset
```

```
<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
```

In [30]:
```
df.isnull().sum() #checking for missing values in dataset
```

Out[30]:
```
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
dtype: int64
```

We can see that there are 4 redundant columns in the dataset. We should drop them before proceeding further.

In [32]:
```
df.drop(['Column1', 'Column2', 'Column3', 'Column4'], axis=1, inplace=
```

...

In [33]:
```
df.info() #summary of dataset
```

```
<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
```

We can see that, there are 3 character variables (data type = object) and remaining 9 numerical variables (data type = int64).

In [34]:
```
df.describe() #view statistical summary of data
```

Out[34]:

	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

There are 3 categorical variables in the dataset. I will explore them one by one.

Explore all 3 categorical variables

In [35]:
```
# view the labels in the variable

df['status_id'].unique()
```

Out[35]:
```
array([   1,    2,    3, ..., 7048, 7049, 7050])
```

In [36]:
```
# view how many different types of variables are there

len(df['status_id'].unique())
```

Out[36]:
```
7050
```

We can see that there are 6997 unique labels in the `status_id` variable. The total number of instances in the dataset is 7050. So, it is approximately a unique identifier for each of the instances. Thus this is not a variable that we can use. Hence, I will drop it.

In [37]:
```
# view the labels in the variable

df['status_published'].unique()
```

Out[37]:
```
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)
```

```
In [38]: # view how many different types of variables are there

         len(df['status_published'].unique())

Out[38]: 6913
```

Again, we can see that there are 6913 unique labels in the `status_published` variable. The total number of instances in the dataset is 7050. So, it is also a approximately a unique identifier for each of the instances. Thus this is not a variable that we can use. Hence, I will drop it also.

```
In [39]: # view the labels in the variable

         df['status_type'].unique()

Out[39]: array(['video', 'photo', 'link', 'status'], dtype=object)
```

```
In [40]: # view how many different types of variables are there

         len(df['status_type'].unique())

Out[40]: 4
```

We can see that there are 4 categories of labels in the `status_type` variable.

Drop `status_id` and `status_published` variable from the dataset

```
In [41]: df.drop(['status_id', 'status_published'], axis=1, inplace=True)
```

View the summary of dataset again

In [42]:
```
df.info() ##summary
```

```
<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
```

Preview the dataset again

In [43]:
```
df.head()
```

Out[43]:

	status_type	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows
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	

We can see that there is 1 non-numeric column `status_type` in the dataset. I will convert it into integer equivalents.

Declare feature vector and target variable

In [44]:
```
##Declare feature vector and target variable
X = df

y = df['status_type']
```

Convert categorical variable into integers

In [45]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X['status_type'] = le.fit_transform(X['status_type'])

y = le.transform(y)
```

View the summary of X

In [46]:
```
X.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   int64
 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(10)
memory usage: 550.9 KB
```

Preview the dataset X

In [47]:
```
X.head()
```

Out[47]:

	status_type	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows
0	3	529	512	262	432	92	3
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	(

Feature Scaling

In [48]:
```
cols = X.columns
```

In [49]:
```
from sklearn.preprocessing import MinMaxScaler

ms = MinMaxScaler()

X = ms.fit_transform(X)
```

In [50]:
```
X = pd.DataFrame(X, columns=[cols])
```

In [51]:
```
X.head()
```

Out[51]:

	status_type	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows
0	1.000000	0.112314	0.024393	0.076519	0.091720	0.140030	0.01079
1	0.333333	0.031847	0.000000	0.000000	0.031847	0.000000	0.00000
2	1.000000	0.048195	0.011243	0.016647	0.043312	0.031963	0.00359
3	0.333333	0.023567	0.000000	0.000000	0.023567	0.000000	0.00000
4	0.333333	0.045223	0.000000	0.000000	0.043312	0.013699	0.00000

K-Means model with two clusters

In [52]:
```
from sklearn.cluster import KMeans

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

kmeans.fit(X)
```

Out[52]:
```
KMeans(n_clusters=2, random_state=0)
```

K-Means model parameters study

In [53]:
```
kmeans.cluster_centers_
```

Out[53]:
```
array([[3.28506857e-01, 3.90710874e-02, 7.54854864e-04, 7.53667113e-0
4,
        3.85438884e-02, 2.17448568e-03, 2.43721364e-03, 1.20039760e-0
3,
        2.75348016e-03, 1.45313276e-03],
       [9.54921576e-01, 6.46330441e-02, 2.67028654e-02, 2.93171709e-0
2,
        5.71231462e-02, 4.71007076e-02, 8.18581889e-03, 9.65207685e-0
3,
        8.04219428e-03, 7.19501847e-03]])
```

- The KMeans algorithm clusters data by trying to separate samples in n groups of equal variances, minimizing a criterion known as **inertia**, can be recognized as a measure of how internally coherent clusters are.

- The k-means algorithm divides a set of N samples X into K disjoint clusters C, each described by the mean j of the samples in the cluster. The means are commonly called the cluster **centroids**.

- The K-means algorithm aims to choose centroids that minimize the inertia, or within-cluster sum of squared criterion.

In [54]:
```
kmeans.inertia_
```

Out[54]:
```
237.7572640441955
```

- The lesser the model inertia, the better the model fit.
- We can see that the model has very high inertia. So, this is not a good model fit to the data.

Check quality of weak classification by the model

In [55]:
```
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." % (correc
```

```
Result: 63 out of 7050 samples were correctly labeled.
```

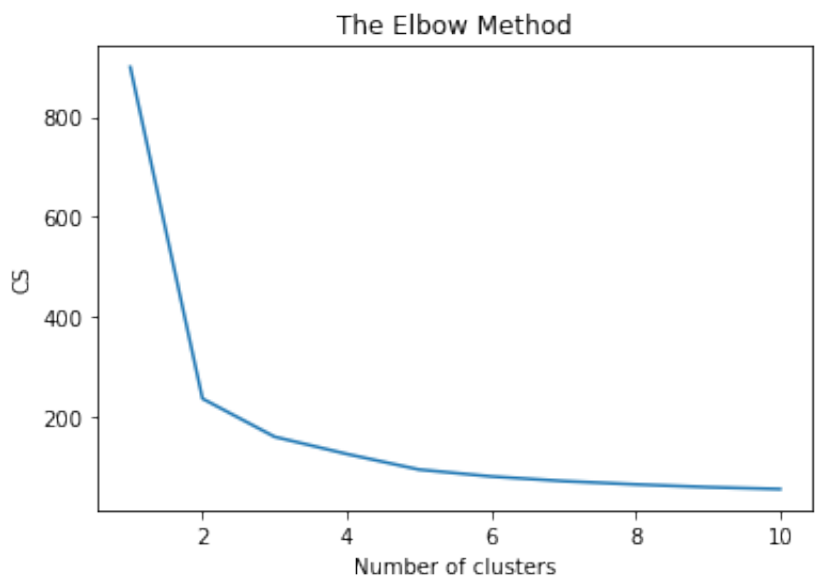
In [56]:
```
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size))
```

```
Accuracy score: 0.01
```

We have achieved a weak classification accuracy of 1% by our unsupervised model.

Using elbow method to find optimal number of clusters

In [57]:
```
from sklearn.cluster import KMeans
cs = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300
    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()
```



- By the above plot, we can see that there is a kink at k=2.
- Hence k=2 can be considered a good number of the cluster to cluster this data.
- But, we have seen that I have achieved a weak classification accuracy of 1% with k=2.
- I will check the model accuracy with different number of clusters.

5. Results

K-Means model with different clusters

In [58]:
```
##K-Means model with 3 clusters
kmeans = KMeans(n_clusters=3, 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." % (correc
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size))
```

```
Result: 138 out of 7050 samples were correctly labeled.
Accuracy score: 0.02
```

In [59]:
```
##K-Means model with 4 clusters
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." % (correc
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size))
```

```
Result: 4340 out of 7050 samples were correctly labeled.
Accuracy score: 0.62
```

We have achieved a relatively high accuracy of 62% with k=4.

6. Discussion and Conclusion

1. One of the central aims of this analysis was to use machine learning models to better understand the underlying patterns of facebook posts,comments and reactions.
2. I was able to load data, preprocess it accordingly, do a little bit of feature engineering and finally were able to make a K-Means model and see it in action.
3. I have applied the elbow method and find that k=2 can be considered a good number of cluster to cluster this data.
4. I have find that the model has very high inertia of 237.7572. So, this is not a good model fit to the data.
5. I have achieved a weak classification accuracy of 1% with k=2 by our unsupervised model.
6. So, I have changed the value of k and find relatively higher classification accuracy of 62% with k=4.
7. Hence, we can conclude that k=4 being the optimal number of clusters.

Limitations

- K-means clustering is a very simple and fast algorithm. Furthermore, it can efficiently deal with very large data sets. However, there are some weaknesses of the k-means approach.
- One potential disadvantage of K-means clustering is that it requires us to pre-specify the number of clusters. Hierarchical clustering is an alternative approach which does not require that we commit to a particular choice of clusters.

7.References

1. Udemy course – Machine Learning – A Z by Kirill Eremenko and Hadelin de Ponteves
2. https://en.wikipedia.org/wiki/K-means_clustering (https://en.wikipedia.org/wiki/K-means_clustering)
3. https://www.datacamp.com/community/tutorials/k-means-clustering-python (https://www.datacamp.com/community/tutorials/k-means-clustering-python)
4. https://archive.ics.uci.edu/ml/datasets/Facebook+Live+Sellers+in+Thailand# (https://archive.ics.uci.edu/ml/datasets/Facebook+Live+Sellers+in+Thailand)