



CLUSTERING REPORT

Of Facebook Live Sellers in Thailand



NOVEMBER 27, 2024

TMU – CS 803

Mithil Joshua Peeris - 501158175

Contents

Background of Dataset.....	2
Methods.....	2
Understanding the Dataset	2
Cleaning and pre-processing the Dataset	3
Results and derived Conclusions	7
Results with Status Type Column	7
Results without Status Type Column	9
References	10
Table of Figures	10

Background of Dataset.

The dataset I am working with is a collection of Facebook pages from 10 Thai fashion and cosmetics retail sellers. It contains data on various types of posts—videos, photos, statuses, and links—across 7051 instances, with 11 features. The engagement metrics tracked include comments, shares, and reactions, which provide valuable insights into how audiences interact with different post types.

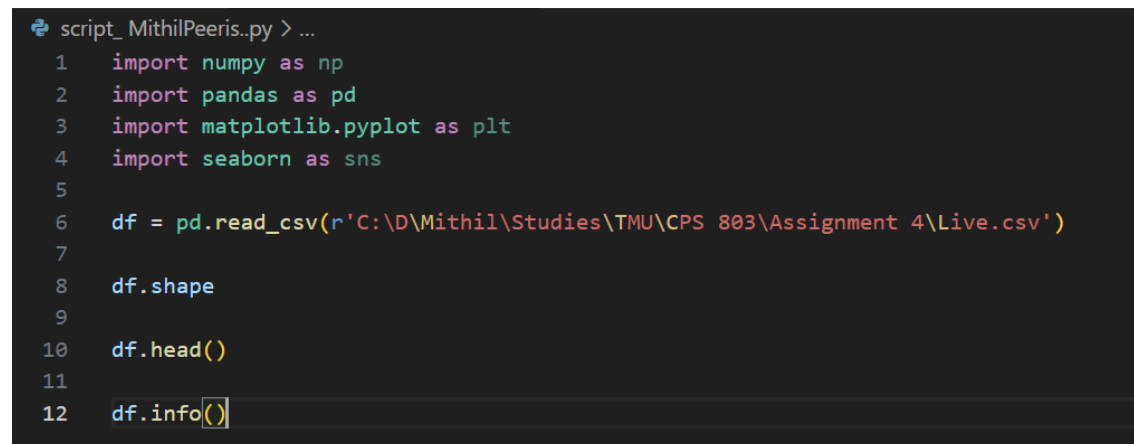
I obtained this dataset from the [UCI Machine Learning Repository](#). It is particularly interesting because it provides a real-world example of consumer behavior on social media platforms, which is essential for businesses aiming to optimize their digital marketing strategies. The dataset focuses on the engagement patterns of retail sellers in Thailand, offering a unique perspective on how businesses use Facebook Live, posts, and other content to engage their customers. This makes it especially relevant for analyzing the effectiveness of different types of content and understanding the factors driving engagement in the fashion and cosmetics sectors.

The dataset lends itself to clustering tasks, where the goal is to group similar types of posts based on engagement metrics. This can provide insights into what types of content attract the most engagement, and whether there are seasonal or time-based trends in consumer behavior.

Overall, this dataset is a rich resource for studying consumer engagement in a social media context and provides an opportunity to apply clustering and other data analysis techniques to uncover patterns and trends in digital marketing.

Methods

Understanding the Dataset



```
script_ MithilPeeris..py > ...
1  import numpy as np
2  import pandas as pd
3  import matplotlib.pyplot as plt
4  import seaborn as sns
5
6  df = pd.read_csv(r'C:\D\Mithil\Studies\TMU\CPS 803\Assignment 4\Live.csv')
7
8  df.shape
9
10 df.head()
11
12 df.info()
```

Figure 1- Importing the CSV

To begin the analysis, the dataset was loaded using Pandas (imported as pd). The CSV file was read into a Data Frame, and exploratory functions such as shape, head(), and info() were utilized to summarize the dataset. This preliminary examination revealed that the data contains 7050 entries and 16 attributes of varying types, including object, int64, and float64. Additionally, it was observed that the last four columns are redundant, likely serving no analytical purpose, and may need to be removed during the pre-processing stage.

```

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)

```

Figure 2 - Summary of the Data Set

Cleaning and pre-processing the Dataset

As observed below the last 4 Columns have no values in them thus, they will be removed from the dataset.

```
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
```

Figure 3- Null Values in each Column

```
1 status_id,status_type,status_published,num_reactions,num_comments,num_shares,num_likes,num_loves,num_wows,num_hahas,num_sads,num_angrys,Column1,Column2,Column3,Column4
2 246675545449582_1649696485147474,video,4/22/2018 6:00,529,512,262,432,92,3,1,1,0,,,,,
3 246675545449582_16494269885087757,photo,4/21/2018 22:45,150,0,0,150,0,0,0,0,0,,
4 246675545449582_1648730588577397,video,4/21/2018 6:17,227,236,57,204,21,1,1,0,0,,
5 246675545449582_1648576705259452,photo,4/21/2018 2:29,111,0,0,111,0,0,0,0,0,,
6 246675545449582_1645700502213739,photo,4/18/2018 3:22,213,0,0,204,0,0,0,0,0,,
7 246675545449582_1645650162218773,photo,4/18/2018 2:14,217,0,0,211,5,1,0,0,0,,
8 246675545449582_1645564175560705,video,4/18/2018 0:24,503,614,72,418,70,10,2,0,3,,
```

Figure 4 - CSV File

The code below was run to clean the columns from the data frame.

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

df.info()

df.describe()
```

Figure 5 - Dropping Columns

Resulting Data Frame.

```
RangeIndex: 7050 entries, 0 to 7049
Data columns (total 12 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
```

Figure 6 - Data after dropped Columns

The dataset includes three columns: status_type, status_id, and status_published, which contain highly unique codes. These unique values could pose challenges for clustering algorithms, as they may hinder the ability to group similar instances effectively. To address this, the unique values in each of these columns were analyzed using the code below, and columns with a significantly large number of unique values were identified and removed to improve the clustering process.

```
print("The number of different status id", len(df['status_id'].unique()))

print("The number of different published id", len(df['status_published'].unique()))

print("The number of different status type", len(df['status_type'].unique()))

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

df.info()

df.head()
```

Figure 7 - Unique Values of First Three Columns

Resulting Data Frame is shown below.

```

The number of different status id 6997
The number of different published id 6913
The number of different status type 4
<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)

```

Figure 8 - Summary of Data after Dropping Columns

Next a target variable was declared to check accuracy later and converted categorical variables to continuous variables.

```

# Separate features (X) and target labels (y)
target_labels = df['status_type']
X = df

# Min-Max scaling
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

```

Figure 9 - Converting Categorical Variables to Continuous Variables

And the data was scaled for better accuracy

```

45 #Scaled the data to make it easier for the clustering model
46 scale_col = X.columns
47 ms = MinMaxScaler()
48 X = ms.fit_transform(X)
49 X = pd.DataFrame(X, columns=[scale_col])
50 print(X)

```

	status_type	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys
0	1.000000	0.112314	0.024393	0.076519	0.091720	0.140030	0.010791	0.006369	0.019608	0.0
1	0.333333	0.031847	0.000000	0.000000	0.031847	0.000000	0.000000	0.000000	0.000000	0.0
2	1.000000	0.048195	0.011243	0.016647	0.043312	0.031963	0.003597	0.006369	0.000000	0.0
3	0.333333	0.023567	0.000000	0.000000	0.023567	0.000000	0.000000	0.000000	0.000000	0.0
4	0.333333	0.045223	0.000000	0.000000	0.043312	0.013699	0.000000	0.000000	0.000000	0.0
...
7045	0.333333	0.018896	0.000000	0.000000	0.018896	0.000000	0.000000	0.000000	0.000000	0.0
7046	0.333333	0.003397	0.000000	0.000000	0.002972	0.001522	0.000000	0.006369	0.000000	0.0
7047	0.333333	0.000425	0.000000	0.000000	0.000212	0.001522	0.000000	0.000000	0.000000	0.0
7048	0.333333	0.074522	0.000572	0.006425	0.074098	0.003044	0.000000	0.000000	0.000000	0.0
7049	0.333333	0.003609	0.000000	0.000000	0.003609	0.000000	0.000000	0.000000	0.000000	0.0

Figure 10 - Scaling the Variables

Next a graph of SSE was made to find the best number of clusters.

```
# Elbow Method to find the optimal number of clusters (SSE)
sse = []
k_range = range(1, 11) # You can adjust the range as needed

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(X_scaled)
    sse.append(kmeans.inertia_) # Inertia (SSE) for current k

# Plot SSE to visualize the "elbow"
plt.figure(figsize=(8, 6))
plt.plot(k_range, sse, marker='o', color='b')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Sum of Squared Errors (SSE)')
plt.grid(True)
plt.show()
```

Figure 11 - Forming an SSE Graph

The graph below was obtained which shows that the optimal point is 2, because the gradient has a large dip at this point.

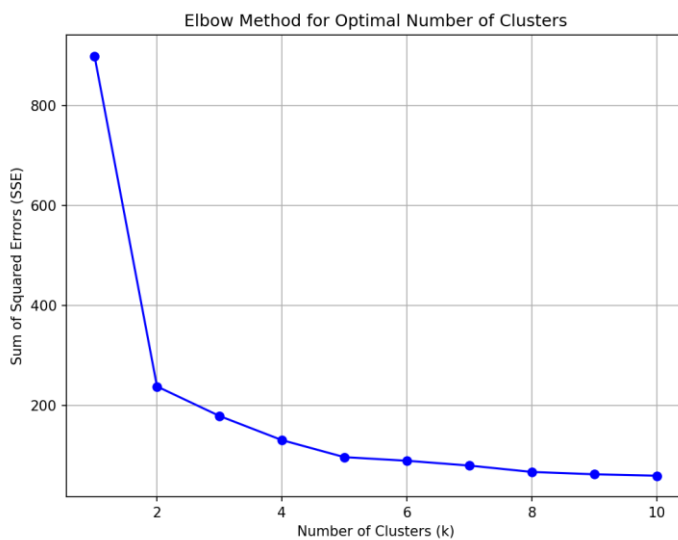


Figure 12 - SSE Graph

Next, the code below was used to perform K-Means clustering on the dataset with the optimal number of clusters ($k = 2$), which was determined using the Elbow method above. After fitting the model, the predicted cluster labels were compared with the true labels to calculate accuracy, providing insight into how well the clustering aligns with known groupings. To facilitate visualization, the high-dimensional data was reduced to two dimensions using Principal Component Analysis (PCA). Finally, a scatter plot

was generated to display the clusters, highlighting the separation of data points in the reduced feature space.

```
# Based on the Elbow method, choose the optimal k
optimal_k = 2 # Change this to the optimal k from the plot

# Apply KMeans clustering with the optimal k
kmeans = KMeans(n_clusters=optimal_k, random_state=0)
kmeans.fit(X_scaled)
labels = kmeans.labels_

# Calculate accuracy (comparison between true labels and predicted labels)
correct_labels = sum(target_labels == labels)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, target_labels.size))
print('Accuracy score: {0:0.2f}'.format(correct_labels / float(target_labels.size)))

# Dimensionality reduction using PCA for visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Plotting the clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=labels, palette='viridis', marker='o')
plt.title('KMeans Clustering of Facebook Live Posts')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title='Cluster', loc='upper right')
plt.show()
```

Figure 13 - Forming the K means Cluster

Results and derived Conclusions

Results with Status Type Column

The clustering results indicate that the model achieved an accuracy of 61% when the number of clusters was set to 2. However, when the number of clusters was increased to 3 and 4, the accuracy decreased to 58%, suggesting that the clustering model struggled to accurately match the four categories of `status_type`. This drop in accuracy implies that the structure of the data may not align well with the clustering produced by K-means, or the categories may not be well-separated in the feature space.

The four distinct lines observed in the PCA plot could be attributed to the `status_type` categories, as the data is divided into four main categories, which could be causing the separation of the data into these four distinct groups. Each cluster might represent a different type of content or engagement behavior, and the separation in the plot suggests that the clusters, while not perfectly aligning with the actual categories, might still be capturing meaningful distinctions based on the type of content posted.

Overall, while the 2-cluster solution provided the highest accuracy, it still reveals that the clustering model is not aligning well with the Status ID column, thus more tweaking needs to be performed to the code to find the hidden pattern.

Additional Model Metrics:

- **Accuracy Score:** 61% (4288 out of 7050 samples were correctly labeled)
- **Silhouette Score:** 0.7641, indicating that the clustering solution has a relatively good separation between clusters.

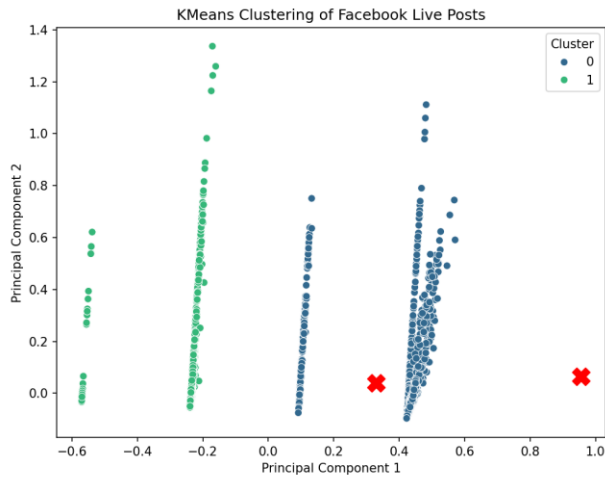


Figure 14 - Clustering Graph with 2 Clusters – With Status Type

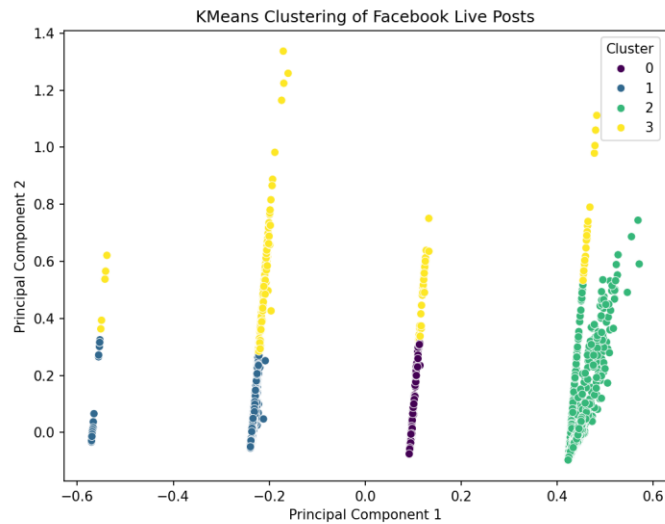


Figure 15- Clustering Graph with 4 Clusters - With Status Type

Results without Status Type Column

After removing the `status_type` column and reapplying the K-means clustering (with 3 clusters as suggested by the elbow method), the accuracy dropped significantly to 7%, but the silhouette score increased to 0.82. This suggests that grouping based on `status_type` wasn't effective, but removing it led to clusters with better separation.

Metrics:

- **Accuracy:** 7% (which indicates poor match with actual labels)
- **Silhouette Score:** 0.82 (indicating better cluster separation)

Insights from Bar Graph:

A bar graph was created with the mean values of each engagement metric vs each cluster, this showed that:

- **Cluster 0:** High engagement
- **Cluster 1:** Mid engagement
- **Cluster 2:** Low engagement

This analysis suggests that the clusters now represent different levels of engagement, with the removal of `status_type` leading to a more meaningful separation of the data based on engagement metrics alone.

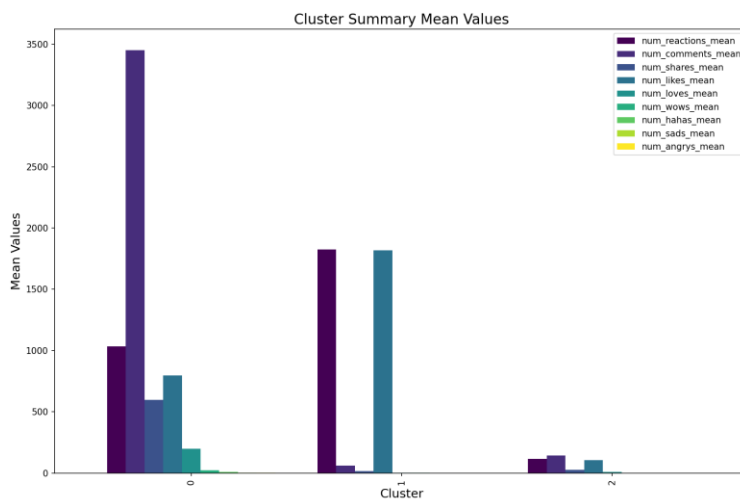


Figure 16 - Levels of Engagement after Removing Status Type Column

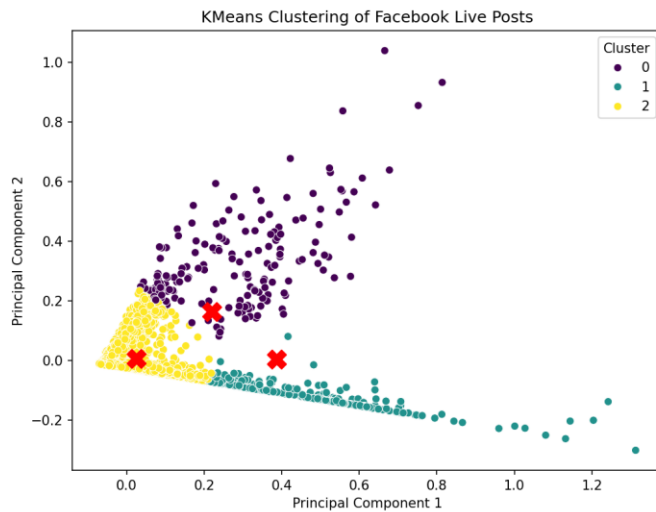


Figure 17 - New Cluster Diagram without Status Type

References

- Lugez, E. (2024). Machine learning: Unsupervised LearningClustering.
- Scikit-learn developers. (2024). *Clustering*. Retrieved from scikit-learn.org: <https://scikit-learn.org/1.5/modules/clustering.html>
- Sérgio Moro, P. R. (2016, August 4). Facebook Live Sellers in Thailand. Retrieved from <https://archive.ics.uci.edu/dataset/488/facebook+live+sellers+in+thailandw>

Table of Figures

Figure 1- Importing the CSV	2
Figure 2 - Summary of the Data Set.....	3
Figure 3- Null Values in each Column	3
Figure 4 - CSV File.....	3
Figure 5 - Dropping Columns	4
Figure 6 - Data after dropped Columns.....	4
Figure 7 - Unique Values of First Three Columns.....	4
Figure 8 - Summary of Data after Dropping Columns	5
Figure 9 - Converting Categorical Variables to Continuous Variables	5
Figure 10 - Scaling the Variables	5
Figure 11 - Forming an SSE Graph.....	6
Figure 12 - SSE Graph.....	6
Figure 13 - Forming the K means Cluster.....	7
Figure 14 - Clustering Graph with 2 Clusters – With Status Type.....	8
Figure 15- Clustering Graph with 4 Clusters - With Status Type	8

Figure 16 - Levels of Engagement after Removing Status Type Column.....	9
Figure 17 - New Cluster Diagram without Status Type.....	10