Assignment 2

CBD 2214: Big Data Fundamentals

Lambton College, Toronto

Group 6

Graham Wall

April 19, 2022

Which dataset we are using

The dataset is called Facebook Live Sellers in Thailand.

The dataset is taken from Facebook pages of 10 Thai fashion and cosmetics retail sellers' Posts of a different nature (video, photos, statuses, and links). Engagement metrics consist of comments, shares, and reactions.

Data Set Characteristics: Multivariate, Number of Instances: 7051, Attribute Characteristics: Integer, Number of Attributes: 12

Specification of your dataset

The variability of consumer engagement is analyzed through a Principal Component Analysis, highlighting the changes induced by the use of Facebook Live. The seasonal component is analyzed through a study of the averages of the different engagement metrics for different time frames (hourly, daily and monthly). Finally, we identify statistical outlier posts, that are qualitatively analyzed further, in terms of their selling approach and activities

.

Attribute Information:

1. status_id
2. status_type
3. status_published
4. num_reactions
5. num_comments
6. num_shares

7. num_likes
8. num_loves
9. num_wows
10. num_hahas
11. num_sads
12. num_angrys

Pre-processing

Importing 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 warnings

warnings.filterwarnings('ignore')
```

Reading the dataset

```
data = 'Live.csv'

df = pd.read_csv(data)
```

Exploratory data analysis

The shape and preview of the dataset

In [4]:
```
df.shape
```

Out[4]:
```
(7050, 16)
```

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

Out[5]:

	status_id	status_type	status_published	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_ha
0	246675545449582_1649696485147474	video	4/22/2018 6:00	529	512	262	432	92	3	
1	246675545449582_1649426988507757	photo	4/21/2018 22:45	150	0	0	150	0	0	
2	246675545449582_1648730588577397	video	4/21/2018 6:17	227	236	57	204	21	1	
3	246675545449582_1648576705259452	photo	4/21/2018 2:29	111	0	0	111	0	0	
4	246675545449582_1645700502213739	photo	4/18/2018 3:22	213	0	0	204	9	0	

Summary of the dataset

In [6]:
```
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
```

Checking missing values

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
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
```

There are 4 redundant columns in the dataset that should be dropped as following:

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

```
df.info()
```

```
<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   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 
dtypes: int64(9), object(3)
memory usage: 661.1+ KB
```

.

```
df.describe()
```

	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys
count	7050.000000	7050.000000	7050.000000	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	0.696454	0.243688	0.113191
std	462.625309	889.636820	131.599965	449.472357	39.972930	8.719650	3.957183	1.597156	0.726812
min	0.000000	0.000000	0.000000	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	0.000000	0.000000	0.000000
50%	59.500000	4.000000	0.000000	58.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	219.000000	23.000000	4.000000	184.750000	3.000000	0.000000	0.000000	0.000000	0.000000
max	4710.000000	20990.000000	3424.000000	4710.000000	657.000000	278.000000	157.000000	51.000000	31.000000

There is 3 categorical variables in the dataset that will be explored one by one.

```
In [11]: df['status_id'].unique()
```

```
Out[11]: array(['246675545449582_1649696485147474',
                 '246675545449582_1649426988507757',
                 '246675545449582_1648730588577397', ...,
                 '1050855161656896_1060126464063099',
                 '1050855161656896_1058663487542730',
                 '1050855161656896_1050858841656528'], dtype=object)
```

```
In [12]: len(df['status_id'].unique())
```

```
Out[12]: 6997
```

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.

```
In [13]: df['status_published'].unique()
Out[13]: 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 [14]: len(df['status_published'].unique())
Out[14]: 6913
```

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 approximately a unique identifier for each of the instances.

```
In [15]: df['status_type'].unique()
Out[15]: array(['video', 'photo', 'link', 'status'], dtype=object)

In [16]: len(df['status_type'].unique())
Out[16]: 4
```

There are 4 categories of labels in the status_type variable.

Dropping down status_id and status_published variable from the dataset and view of the dataset:

```
In [17]: df.drop(['status_id', 'status_published'], axis=1, inplace=True)

In [18]: 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

In [19]: df.head()
```

Out[19]:

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

What is Unsupervised Learning?

It applies to the dataset, which has no specific label or class attributes. Here the algorithm learns independently and tries to make clusters of data without any human interference.

K-Means Clustering

It is a prevalent clustering method where data points are divided into K groups. K indicates the number of clusters and the distance between the centroids of each group. The data points closer to the given centroid are clustered under a similar group. It is commonly used for image segmentation document clustering.

How K-means clustering algorithm work

K-Means clustering is an unsupervised learning algorithm. There is no labeled data for this clustering, unlike in supervised learning. K-Means performs the division of objects into clusters that share similarities and are dissimilar to the objects belonging to another cluster.

The term ‘K’ is a number. You need to tell the system how many clusters you need to create. For example, K = 2 refers to two clusters. K-means clustering uses “centroids”, K different randomly-initiated points in the data, and assigns every data point to the nearest centroid. After every point has been assigned, the centroid is moved to the average of all of the points assigned to it.

Result of implementation

Declare feature vector and target variable

```
X = df

y = df['status_type']
```

Converting Status_Type into integers

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

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

y = le.transform(y)
```

View of X

```
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   int32
 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: int32(1), int64(9)
memory usage: 523.4 KB
```

```
X.head()
```

	status_type	num_reactions	num_comments	num_shares	num_likes	num_loves	num_wows	num_hahas	num_sads	num_angrys
0	3	529	512	262	432	92	3	1	1	0
1	1	150	0	0	150	0	0	0	0	0
2	3	227	236	57	204	21	1	1	0	0
3	1	111	0	0	111	0	0	0	0	0
4	1	213	0	0	204	9	0	0	0	0

Feature scaling

```
cols = X.columns
```

```
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_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

K-mean with two clusters

```
from sklearn.cluster import KMeans

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

kmeans.fit(X)
```

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

K-Means model parameters study

```
kmeans.cluster_centers_
```

```
array([[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],
       [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]])
```

- The KMeans algorithm clusters data by trying to separate samples in n groups of equal variances, minimizing a criterion known as inertia, or within-cluster sum-of-squares Inertia, or the within-cluster sum of squares criterion, 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
- Inertia is not a normalized metric.
- The lower values of inertia are better and zero is optimal.
- But in very high-dimensional spaces, euclidean distances tend to become inflated (this is an instance of curse of dimensionality).
- Running a dimensionality reduction algorithm such as PCA prior to k-means clustering can alleviate this problem and speed up the computations.
- We can calculate model inertia as follows:-

```
kmeans.inertia_
```

```
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.

Clusters Accuracy:

Two Clusters Accuracy

```
from sklearn.cluster import KMeans

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

#clusters=kmeans.fit(X)
label=kmeans.fit_predict(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: 63 out of 7050 samples were correctly labeled.
Accuracy score: 0.01
```

3 Clusters

```
kmeans = KMeans(n_clusters=3, random_state=0)

#kmeans.fit(X)
label=kmeans.fit_predict(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: 138 out of 7050 samples were correctly labeled.
Accuracy score: 0.02
```

4 Clusters

```
kmeans = KMeans(n_clusters=4, random_state=0)

#kmeans.fit(X)
label=kmeans.fit_predict(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: 4340 out of 7050 samples were correctly labeled.
Accuracy score: 0.62
```

Conclusion

1. In this project, we have implemented the most popular unsupervised clustering technique called K-Means Clustering.
2. we have find that the model has very high inertia of 237.7572. So, this is not a good model fit to the data.
3. we have achieved a weak classification accuracy of 1% with k=2 by our unsupervised model.
4. So, we have changed the value of k and find relatively higher classification accuracy of 62% with k=4.
5. Hence, we can conclude that k=4 being the optimal number of clusters.

References

Arvai, Kevin., (2020), K-Means Clustering in Python: A Practical Guide,

https://realpython.com/k-means-clustering-python/

Brownlee, J. (August, 2020), **https://machinelearningmastery.com/clustering-algorithms-with-python/**

Facebook Live Sellers in Thailand Data Set, (2019, April 22), UCI,

https://archive.ics.uci.edu/ml/datasets/Facebook+Live+Sellers+in+Thailand

What is K-Clustering, (2022), Learn By Market,

https://www.learnbymarketing.com/methods/k-means-clustering/#:~:text=K%2Dmeans%20clustering%20uses%20%E2%80%9Ccentroids,the%20points%20assigned%20to%20it.

Paul, S., (2018, July 5), K-Means Clustering in Python with scikit-learn,

https://www.datacamp.com/community/tutorials/k-means-clustering-python