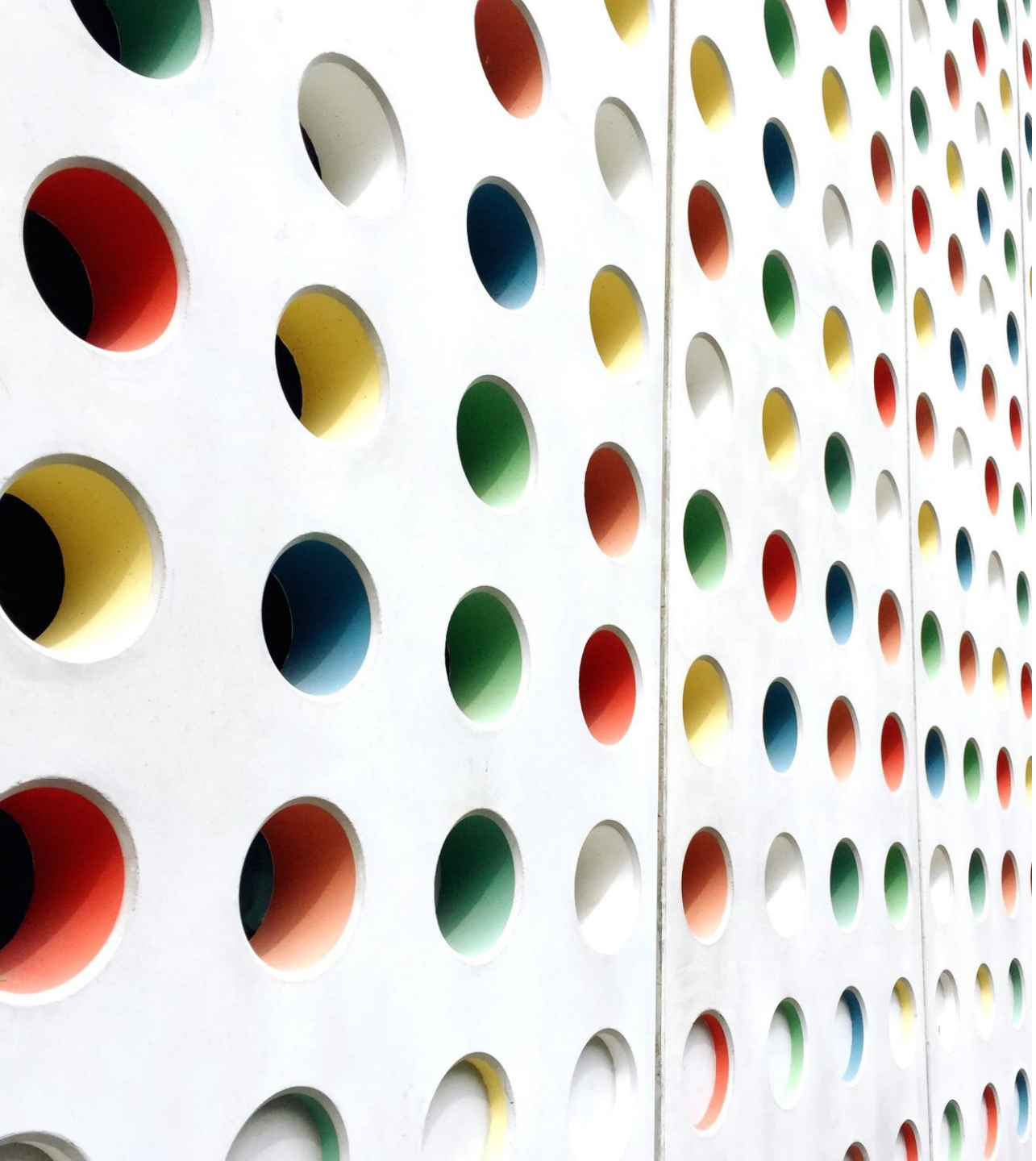


Homework 07

Product Recommendation

MS.SARANYA THINSOOK

ID NO. 6220422002

BADS 7105 CUSTOMER RELATIONSHIP MANAGEMENT ANALYTICS AND INTELLIGENCE

OBJECTIVE

PRODUCT RECOMMENDATION

CLUSTERING FOR CUSTOMER SEGMENTATION

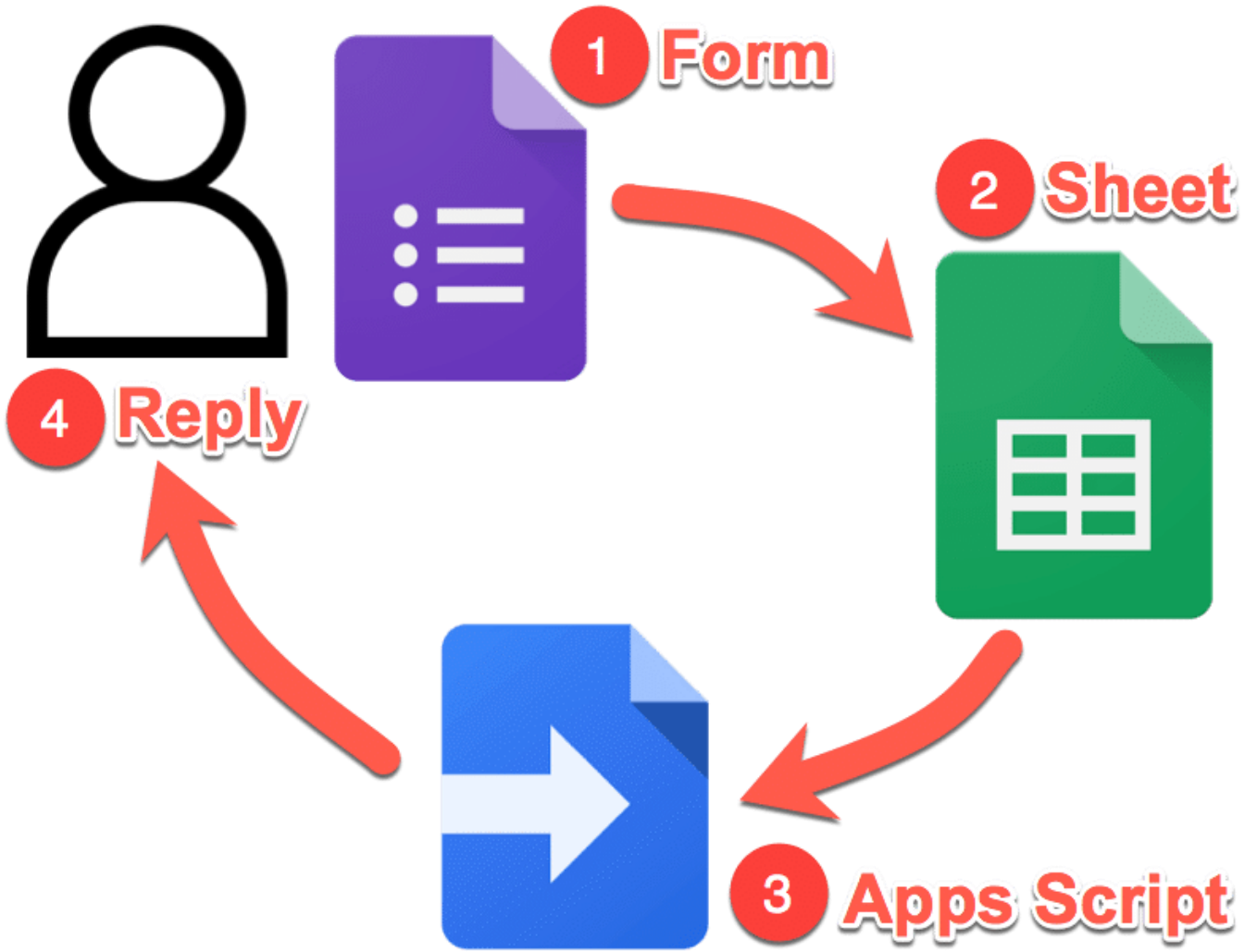
SAMPLE SIZE AND SURVEY

✍ Online Survey by Google Forms Questionnaire for 60 items question with

2 choices ✍

☑ YES or ☒ NO

✍ Sample size is 44 participants.



DATA

PRODUCT ITEMS

1 Moutain bike	16 SangSom	31 Document grinder	46 Roti
2 Collagen	17 Curry puff	32 Bag	47 Pickled Shrimp Paste
3 Green Brownie	18 Lego	33 Shabu	48 Dog
4 Mekhong	19 Fan	34 Surfskate	49 Thai travel
5 Botox	20 Pickle drink	35 Board game	50 FIFA Online
6 Cannabis	21 Nightdress	36 Dog shirt	51 Mango
7 Hemp	22 Dumbell	37 Omakase	52 Cat auto toilet
8 Pressure gauge	23 Robot cleaner	38 Cashew nuts	53 bag (Chanel)
9 LED RGB	24 Wa grill	39 AI LED	54 Kaki
10 Bikini wax	25 Pork stick	40 Hair clippers	55 Cordyceps
11 Electric massage chair	26 UAV	41 Dyson	56 Healthy pillow for insomnia
12 Ornamental plants	27 Food waste shredder	42 Music Discs	57 Clean food for cat
13 Bolster	28 C2-shirt	43 Nose hair scissors	58 Dido
14 Scales	29 Yipsee card	44 PS5	59 Software
15 Jaw botox	30 BBQ	45 Blood Glucose Meter	60 Sunscreen

DATA ANALYTICS

1.Product recommendation

* Using Market Basket Analysis

* Using Collaborative Filtering

2. Clustering for Customer Segmentation

Product recommendation

* Using Market Basket Analysis

Prepare data

```
[ ] import pandas as pd
```

```
[ ] df = pd.read_excel("Customer Survey.xlsx")
    df.drop(labels=df.columns[0:2].tolist(), axis=1, inplace=True)
    df.head(5)
```

	Moutain bike	Collagen	Green Brownie	Mekhong	Botox	Cannabis	Hemp	Pressure gauge	LED RGB	Bikini wax	Electric massage chair	Ornamental plants	Bolster	Scales	Jaw botox	SangSom	Curry puff
0	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	เคย	ไม่เคย	ไม่เคย	ไม่เคย	เคย
1	ไม่เคย	เคย	ไม่เคย	เคย	ไม่เคย	ไม่เคย	ไม่เคย	เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	เคย	เคย	ไม่เคย	เคย	เคย
2	เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	เคย	เคย	ไม่เคย	ไม่เคย	เคย	เคย	เคย	ไม่เคย	ไม่เคย	เคย
3	ไม่เคย	ไม่เคย	ไม่เคย	เคย	ไม่เคย	เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	เคย	เคย	เคย	เคย	ไม่เคย	เคย	เคย
4	ไม่เคย	เคย	ไม่เคย	ไม่เคย	เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	เคย	ไม่เคย	ไม่เคย	เคย	ไม่เคย	เคย

```
[ ] for i in df.columns.tolist():
        for index, j in enumerate(df[i]):
            if j == "ไม่เคย":
                df[i][index] = 0
            else:
                df[i][index] = 1
    df.head(5)
```

	Moutain bike	Collagen	Green Brownie	Mekhong	Botox	Cannabis	Hemp	Pressure gauge	LED RGB	Bikini wax	Electric massage chair	Ornamental plants	Bolster
0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	1	0	1	0	0	0	1	0	0	0	0	1
2	1	0	0	0	0	0	0	1	1	0	0	1	1
3	0	0	0	1	0	1	0	0	0	0	1	1	1
4	0	1	0	0	1	0	0	0	0	0	0	1	0

Product recommendation

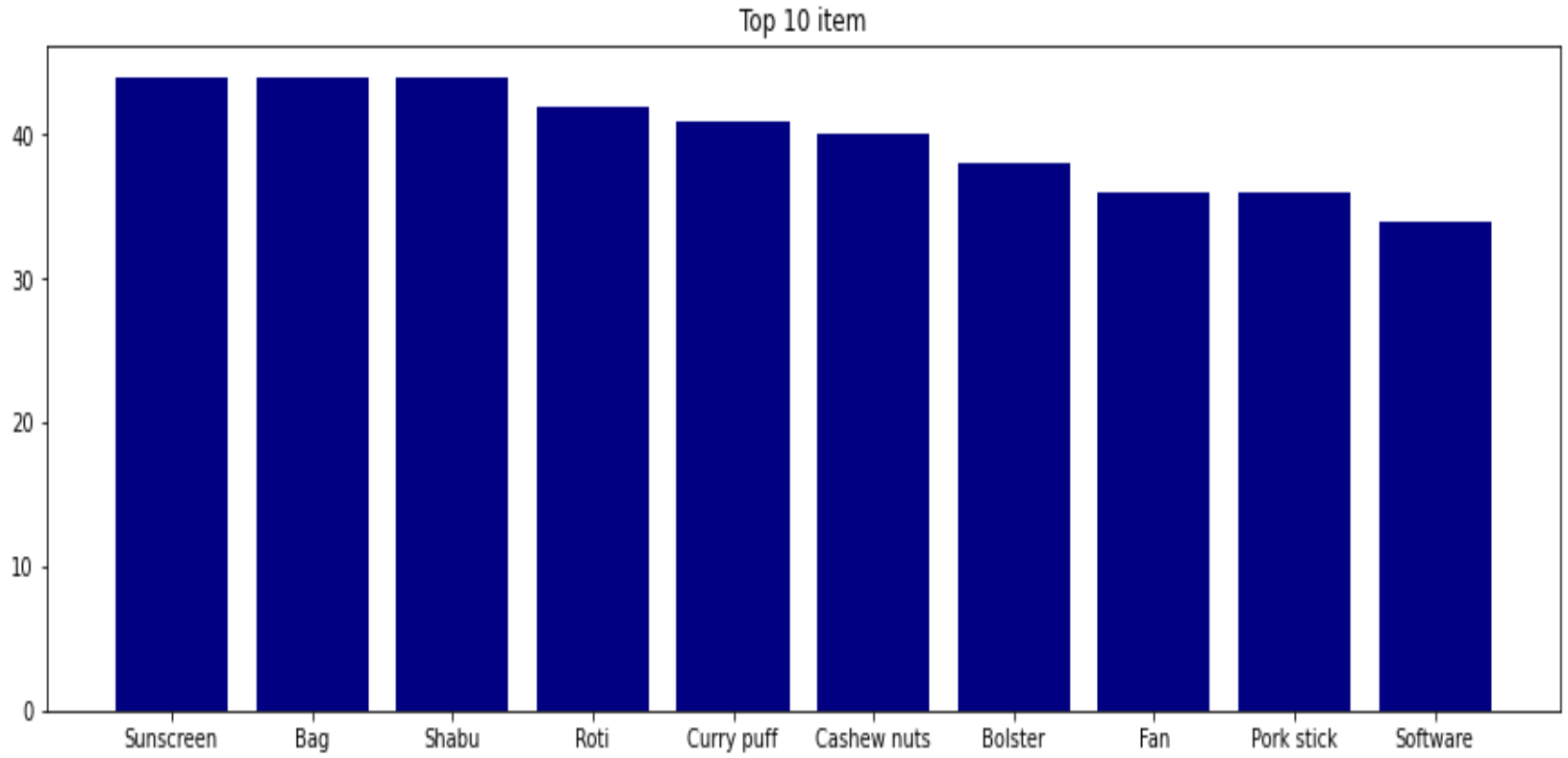
Plot top 10 items

```
[ ]  top_10_item = df.sum(axis=0).sort_values(ascending=False)[:10]
     Item_name = top_10_item.keys().tolist()

[ ]  import numpy as np
     import matplotlib.pyplot as plt

     Item_array = np.arange(len(top_10_item))

     plt.figure(figsize=(15,5))
     plt.bar(Item_array, top_10_item.iloc[:], color = "navy")
     plt.xticks(Item_array, Item_name)
     plt.title('Top 10 item')
     plt.show()
```



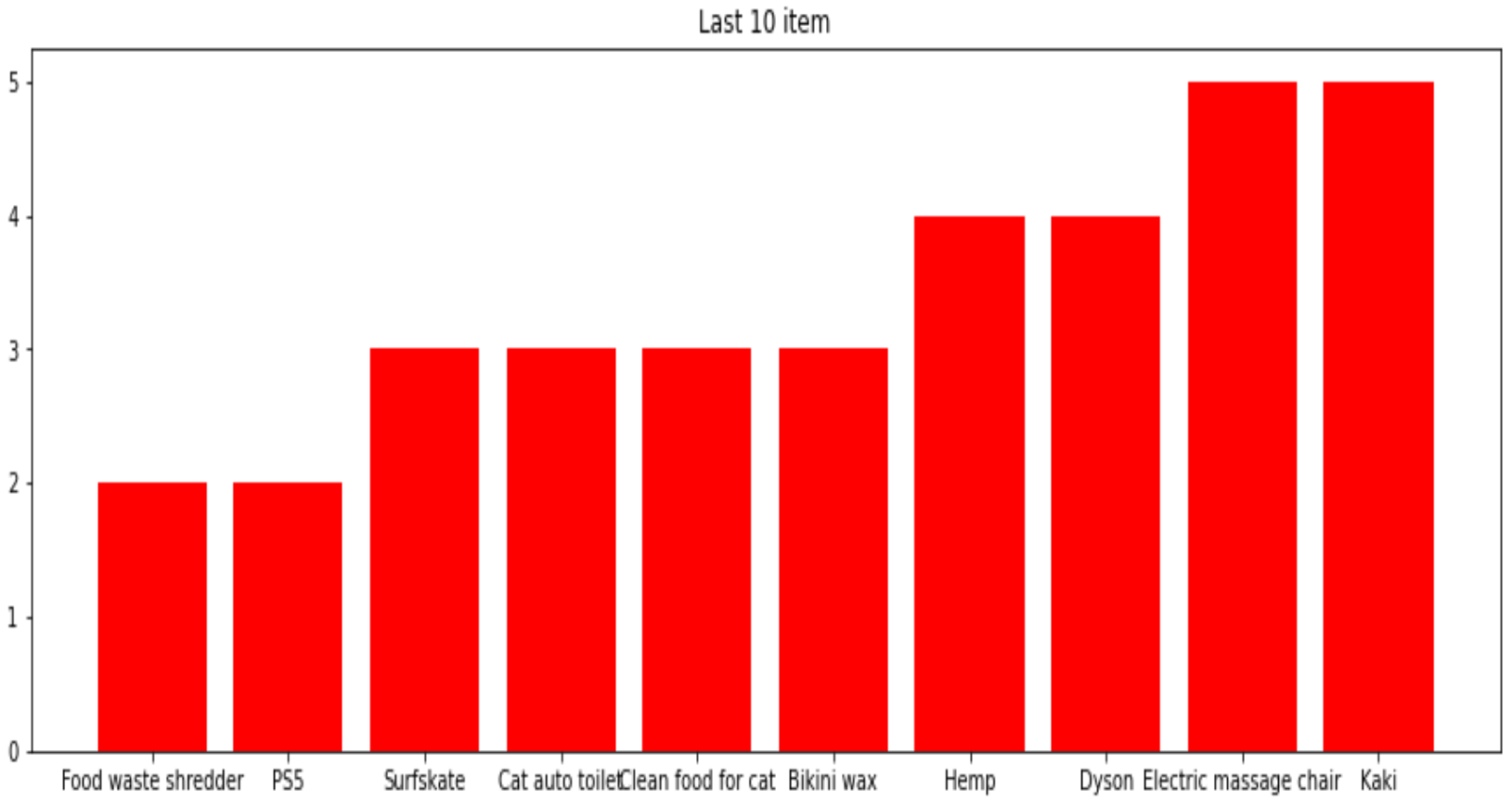
Product recommendation

Plot last 10 items

```
[ ]  last_10_item = df.sum(axis=0).sort_values(ascending=True)[:10]
     Item_name1 = last_10_item.keys().tolist()
```

```
[ ]  Item_array1 = np.arange(len(last_10_item))

     plt.figure(figsize=(15,5))
     plt.bar(Item_array1, last_10_item.iloc[:], color = "red")
     plt.xticks(Item_array1, Item_name1)
     plt.title('Last 10 item')
     plt.show()
```



Product recommendation

- Prepare data for Apriori using one hot encoding
- Turn float into integer

```
hot_encoded_df = df_Item.groupby(["Score", "Item"])['Item'].count().unstack().reset_index().fillna(0).set_index("Score")
hot_encoded_df.head(5)
```

Item	AI LED	BBQ	Bag	Bikini wax	Blood Glucose Meter	Board game	Bolster	Botox	C2-shirt	Cannabis	Cashew nuts	Cat auto toilet	Clean food for cat	Collagen	Cordyceps
Score															
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
[ ] def encode_units(x):
        if x <= 0:
            return 0
        if x >= 1:
            return 1
    hot_encoded_df = hot_encoded_df.applymap(encode_units)
```

Product recommendation

- Determine product association using apriori
- List out rules
- Filter rules

```
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

frequent_itemsets = apriori(hot_encoded_df, min_support=0.01, use_colnames=True)
```

```
[ ] rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
    rules.head(10)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(AI LED)	(Mekhong)	0.034483	0.034483	0.034483	1.0	29.0	0.033294
1	(Mekhong)	(AI LED)	0.034483	0.034483	0.034483	1.0	29.0	0.033294
2	(BBQ)	(Ornamental plants)	0.034483	0.034483	0.034483	1.0	29.0	0.033294
3	(Ornamental plants)	(BBQ)	0.034483	0.034483	0.034483	1.0	29.0	0.033294
4	(Bag)	(Shabu)	0.034483	0.034483	0.034483	1.0	29.0	0.033294

```
[ ] rules[ (rules['lift'] >= 1) &
           (rules['confidence'] >= 0.5) ]
```

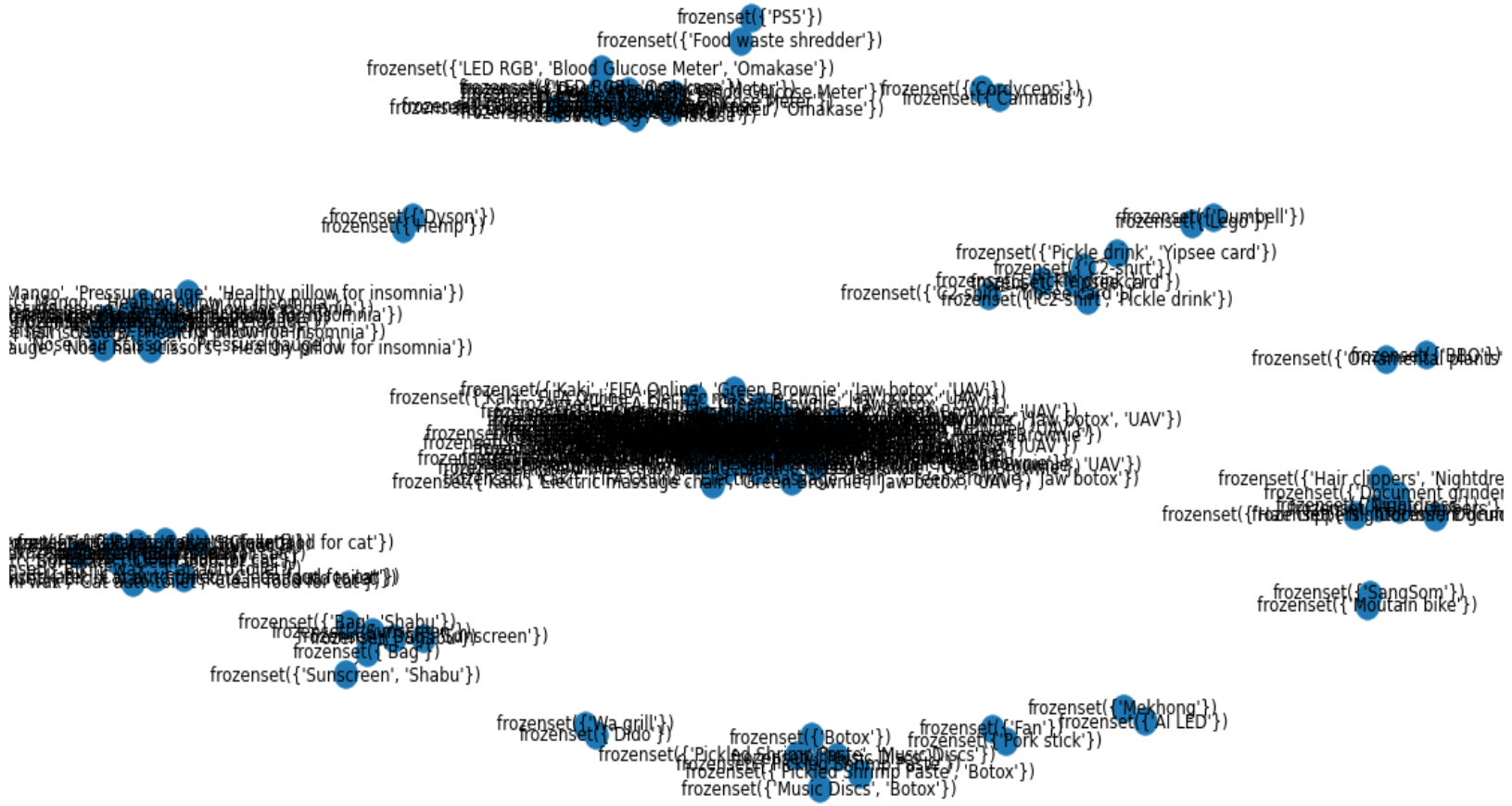
	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(AI LED)	(Mekhong)	0.034483	0.034483	0.034483	1.0	29.0	0.033294
1	(Mekhong)	(AI LED)	0.034483	0.034483	0.034483	1.0	29.0	0.033294
2	(BBQ)	(Ornamental plants)	0.034483	0.034483	0.034483	1.0	29.0	0.033294
3	(Ornamental plants)	(BBQ)	0.034483	0.034483	0.034483	1.0	29.0	0.033294
4	(Bag)	(Shabu)	0.034483	0.034483	0.034483	1.0	29.0	0.033294

Product recommendation

Visualize rules

```
[ ] import matplotlib.pyplot as plt
    import networkx as nx

    fig, ax=plt.subplots(figsize=(20, 10))
    GA=nx.from_pandas_edgelist(rules,source='antecedents',target='consequents')
    nx.draw(GA,with_labels=True)
    plt.show()
```



Product recommendation

* * Using Collaborative Filtering

- Prepare data
Calculate Cosine Similarity
- Convert results into list of rules
- Filter rules

```
[ ]  item_item_matrix = pd.DataFrame(index=hot_encoded_df.columns,columns=hot_encoded_df.columns)
```

```
[ ]  from scipy.spatial.distance import cosine

     for i in range(0,len(item_item_matrix.columns)) :
         # Loop through the columns for each column
         for j in range(0,len(item_item_matrix.columns)) :
           # Fill in placeholder with cosine similarities
           item_item_matrix.iloc[i,j] = 1 -cosine(hot_encoded_df.iloc[:,i],hot_encoded_df.iloc[:,j])
```

```
[ ]  links = item_item_matrix.rename_axis('related item',
                                         axis='columns').stack().reset_index()
     links.columns = ['item', 'related item','value']
```

```
[ ]  links_filtered=links.loc[ (links['value'] > 0.1) &
                               (links['item'] != links['related item']) ]
```

Product recommendation

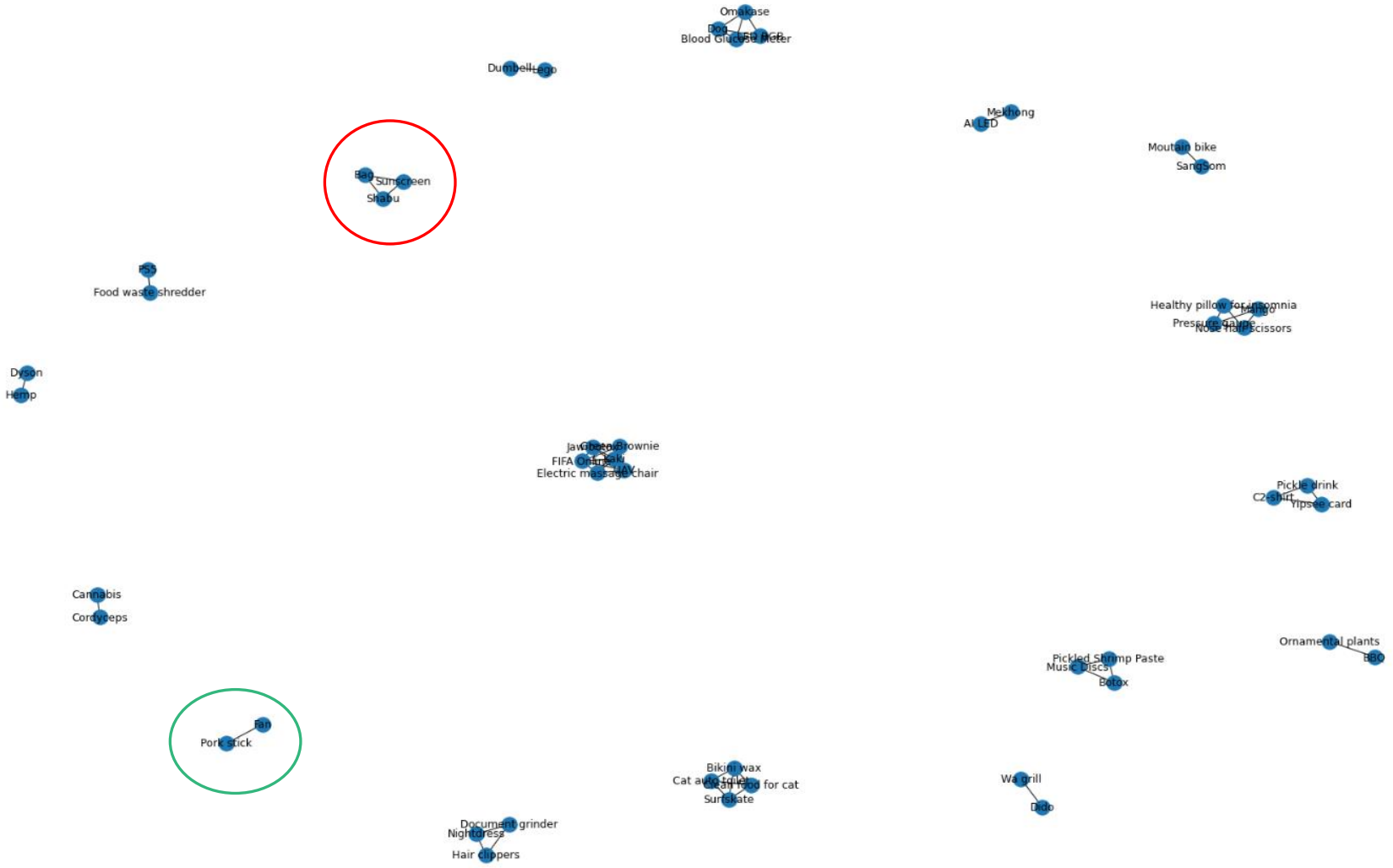
* * Using Collaborative Filtering

Visualize rules

	Score	Item
0	44	Sunscreen
1	44	Bag
2	44	Shabu
3	42	Roti
4	41	Curry puff
5	40	Cashew nuts
6	38	Bolster
7	36	Fan
8	36	Pork stick

```
[ ]  import matplotlib.pyplot as plt
     import networkx as nx

     fig, ax=plt.subplots(figsize=(30,20))
     GA=nx.from_pandas_edgelist(links_filtered, source='item',target='related item')
     nx.draw(GA,with_labels=True)
     plt.show()
```



Clustering for Customer Segmentation

Prepare data
Visualize distribution of variables

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
```

```
df = pd.read_excel("Customer Survey.xlsx")
df.drop(labels=df.columns[0:2].tolist(), axis=1, inplace=True)
df.head(5)
```

	Moutain bike	Collagen	Green Brownie	Mekhong	Botox	Cannabis	Hemp	Pressure gauge	LED RGB	Bikini wax	Electric massage chair	Ornamental plants	Bolster
0	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	เคย
1	ไม่เคย	เคย	ไม่เคย	เคย	ไม่เคย	ไม่เคย	ไม่เคย	เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	เคย
2	เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	เคย	เคย	ไม่เคย	ไม่เคย	เคย	เคย
3	ไม่เคย	ไม่เคย	ไม่เคย	เคย	ไม่เคย	เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	เคย	เคย	เคย
4	ไม่เคย	เคย	ไม่เคย	ไม่เคย	เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	ไม่เคย	เคย	ไม่เคย

```
for i in df.columns.tolist():
    for index, j in enumerate(df[i]):
        if j == "ไม่เคย":
            df[i][index] = 0
        else:
            df[i][index] = 1
df.head(5)
```

	Moutain bike	Collagen	Green Brownie	Mekhong	Botox	Cannabis	Hemp	Pressure gauge	LED RGB	Bikini wax	Electric massage chair	Ornamental plants	Bolster
0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	1	0	1	0	0	0	1	0	0	0	0	1
2	1	0	0	0	0	0	0	1	1	0	0	1	1
3	0	0	0	1	0	1	0	0	0	0	1	1	1
4	0	1	0	0	1	0	0	0	0	0	0	1	0

Clustering for Customer Segmentation

Find optimum k using Silhouette Index

```
[ ] from sklearn.cluster import KMeans
```

```
[ ] cols = df.columns.tolist()
```

```
[ ] from sklearn.metrics import silhouette_samples, silhouette_score
    from yellowbrick.cluster import SilhouetteVisualizer
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning: The sklearn.metrics.classific
  warnings.warn(message, FutureWarning)
```

```
[ ] def sil_score(X, from_k=2, to_k=6):
        #calculate silhouette score for k clusters
        sils=[]
        for k in range(from_k, to_k + 1):
            m = KMeans(n_clusters=k, random_state=1)
            m.fit(X)
            # The silhouette_score gives the average value for all the samples
            silhouette_avg = silhouette_score(X, m.labels_).round(4)
            sils.append([silhouette_avg, k])

        return sils
```

```
[ ] ss = sil_score(df[cols], 2, 5)
    print(f'scores = {ss}')
    print(f'optimal number of clusters = {max(ss)[1]}')
```

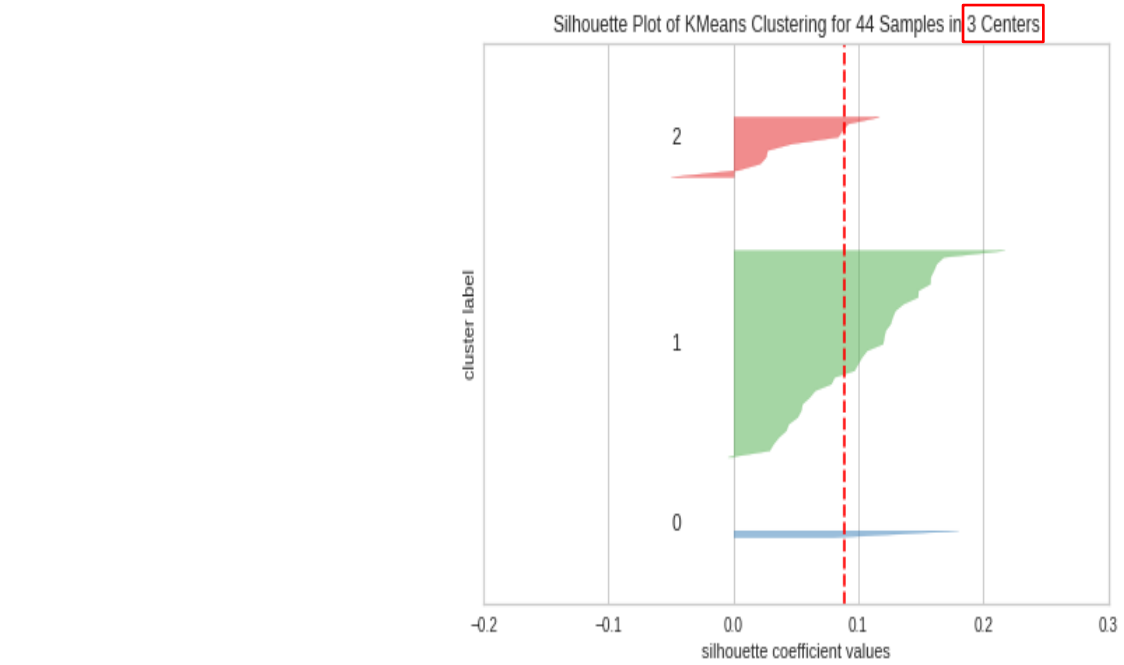
```
scores = [[0.0873, 2], [0.0892, 3], [0.0755, 4], [0.0591, 5]]
optimal number of clusters = 3
```

```
[ ] def silhouette_plot(X, from_k, to_k):
        sil_scores=[]
        for k in range(from_k, to_k + 1):
            m = KMeans(n_clusters=k, random_state=1)
            visualizer = SilhouetteVisualizer(m)
            visualizer.fit(X)
            visualizer.poof()
            sil_scores.append([visualizer.silhouette_score_, k])
        return sil_scores
```

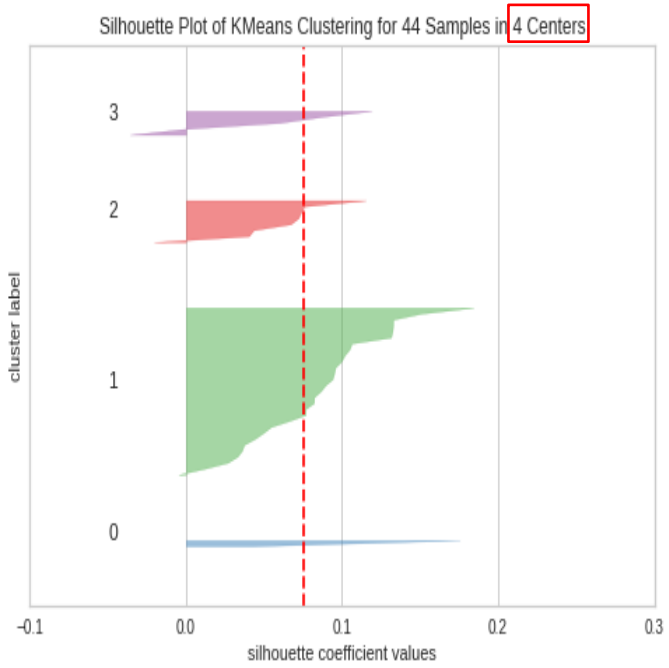
```
[ ] scores = silhouette_plot(df[cols], 3, 5)
```

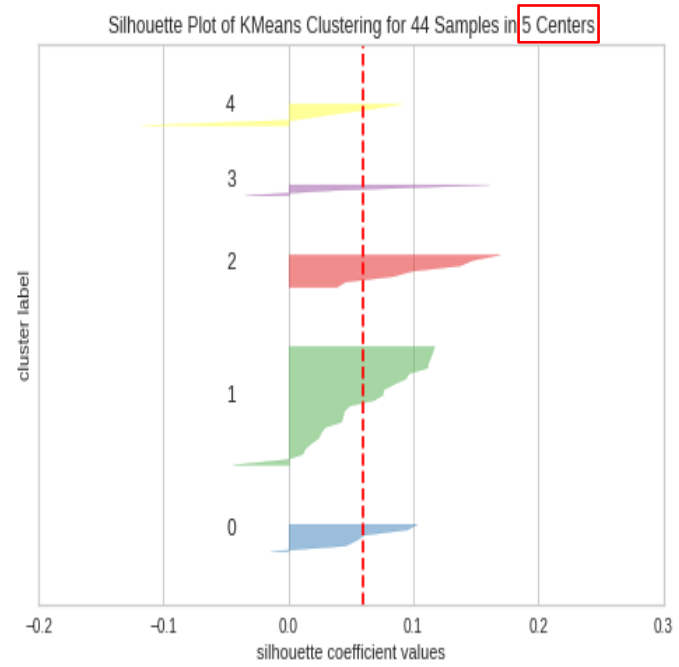
Clustering for Customer Segmentation

K-means clustering



k = 3





Clustering for Customer Segmentation

Determine average behaviors of each cluster

```
[11] model = KMeans(n_clusters=3)
     model.fit(df[cols])
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

```
[12] df["cluster"]=model.labels_
     df.head(5)
```

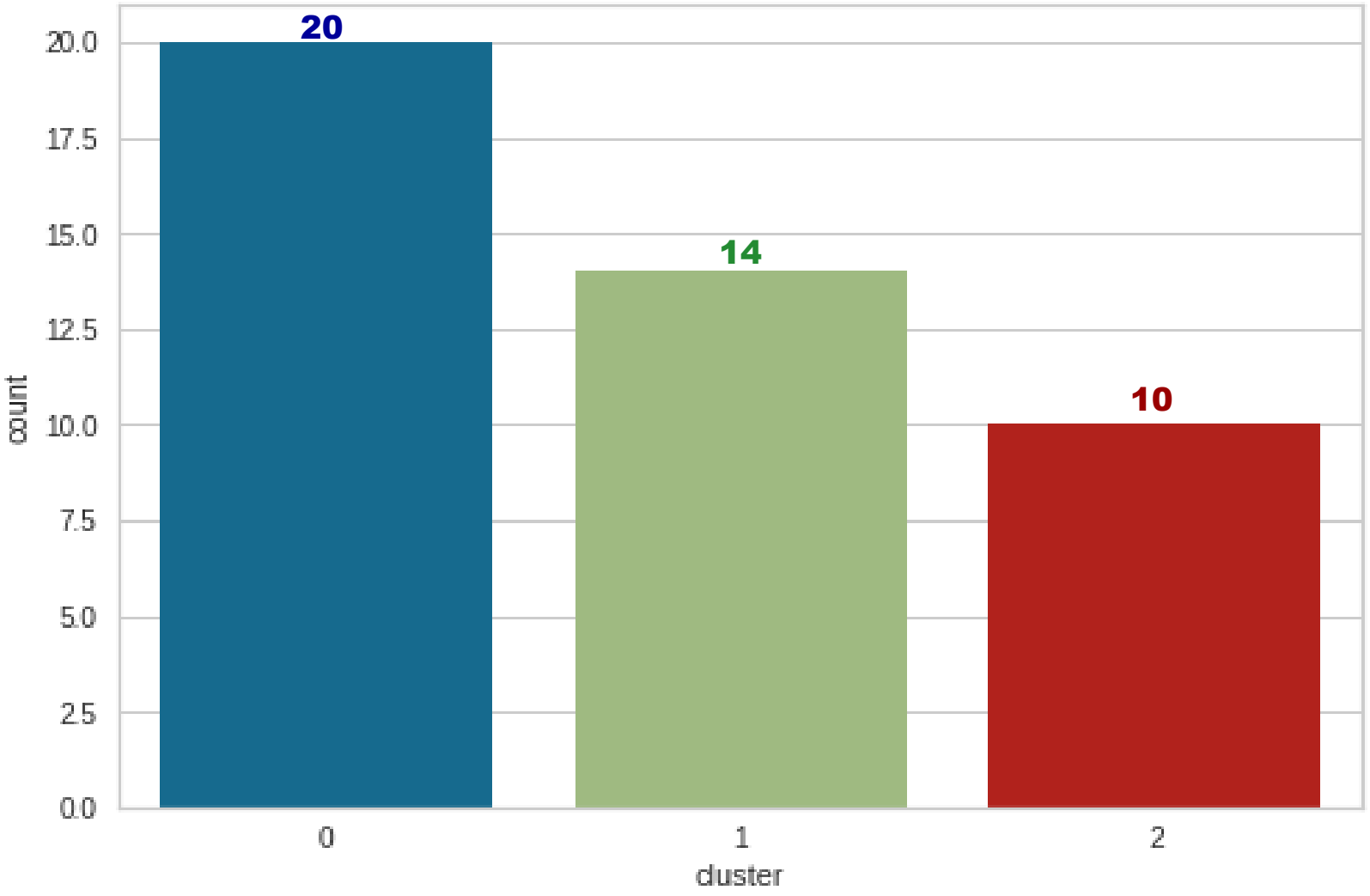
	Moutain bike	Collagen	Green Brownie	Mekhong	Botox	Cannabis	Hemp	Pressure gauge	LED RGB	Bikini wax	Electric massage chair	Ornamental plants	Bolster	Scales	Jaw botox
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	0	1	0	1	0	0	0	1	0	0	0	0	1	1	0
2	1	0	0	0	0	0	0	1	1	0	0	1	1	1	0
3	0	0	0	1	0	1	0	0	0	0	1	1	1	1	0
4	0	1	0	0	1	0	0	0	0	0	0	1	0	0	1

Clustering for Customer Segmentation

Visualize distribution of variables – Bar chart

```
[13] sns.countplot(x="cluster", data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff0893e6610>
```



Sample size = 44

Clustering for Customer Segmentation

Visualize distribution of variables - Cluster 2

```
[14] df.sort_values('cluster', inplace=True)
```

```
df_cluster_1 = df.copy().loc[df["cluster"] == 0]
df_cluster_2 = df.copy().loc[df["cluster"] == 1]
df_cluster_3 = df.copy().loc[df["cluster"] == 2]
```

```
[16] df_cluster_2
```

	Moutain bike	Collagen	Green Brownie	Mekhong	Botox	Cannabis	Hemp	Pressure gauge	LED RGB	Bikini wax	Electric massage chair	Ornamental plants	Bolster	Scales	Jaw botox
33	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
38	1	1	1	1	0	1	0	0	0	0	0	1	1	1	0
29	1	0	1	1	1	1	1	0	0	0	0	1	1	1	0
40	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0
41	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0
26	0	0	0	0	0	1	0	1	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	1	0	1	0	1	0	0	0	1	1	1	0
3	0	0	0	1	0	1	0	0	0	0	1	1	1	1	0
6	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0
9	0	1	0	0	0	0	0	1	0	0	0	1	1	0	0
25	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
43	0	0	1	1	0	1	1	0	0	0	1	1	1	0	0

Clustering for Customer Segmentation

Visualize distribution of variables - Show top 10 in each cluster

```
[17] #Let's pick top 10 for each groups
```

```
[18] df_cluster_1_top10 = df_cluster_1.copy().sum(axis=0).sort_values(ascending=False)[1:11]
     df_cluster_1_top10
```

```
Bag            20
Shabu          20
Roti           20
Bolster        19
Cashew nuts    18
Curry puff     18
Fan            17
Software       17
Dumbell        17
BBQ            16
dtype: int64
```

```
[19] df_cluster_2_top10 = df_cluster_2.copy().sum(axis=0).sort_values(ascending=False)[1:11]
     df_cluster_2_top10
```

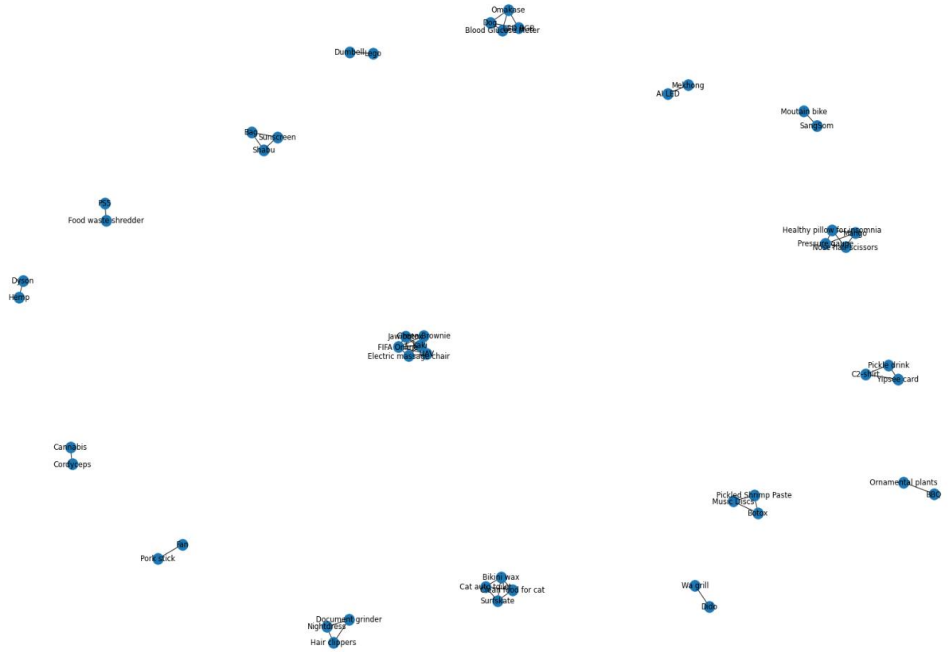
```
Cashew nuts          14
Roti                 14
Curry puff           14
Shabu                14
Bag                  14
Sunscreen            14
Pork stick           14
BBQ                  13
Wa grill             13
Ornamental plants    13
dtype: int64
```

```
[20] df_cluster_3_top10 = df_cluster_3.copy().sum(axis=0).sort_values(ascending=False)[1:11]
     df_cluster_3_top10
```

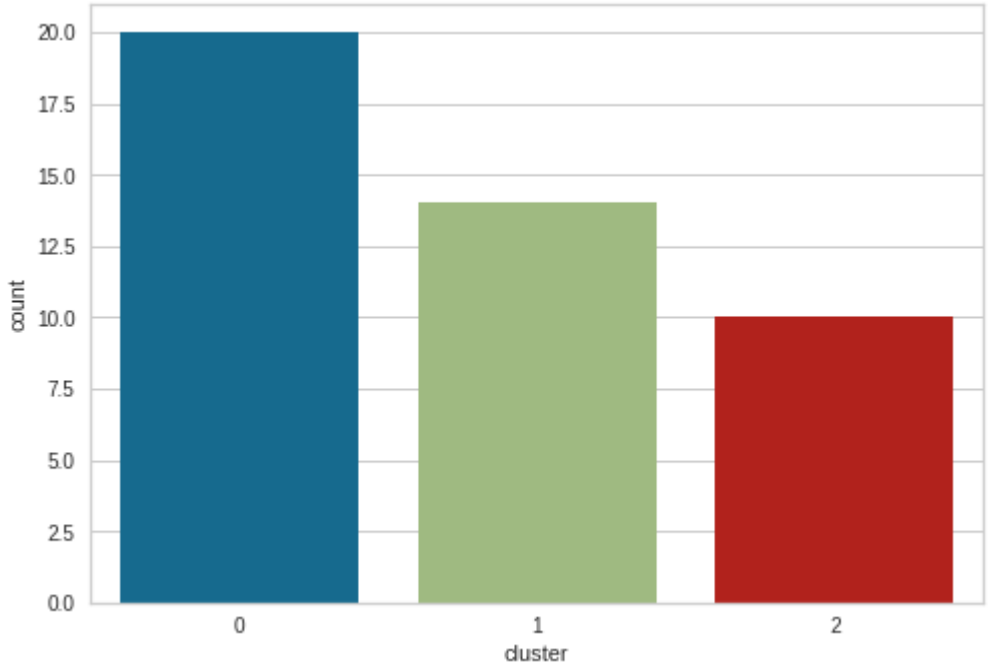
```
Sunscreen      10
Bag            10
Shabu          10
Dido            9
Curry puff      9
Roti            8
Pork stick      8
Board game      8
Cashew nuts     8
Software        7
dtype: int64
```

RESULT AND CONCLUSION

- Product recommendation



- Clustering for Customer Segmentation



THANK YOU