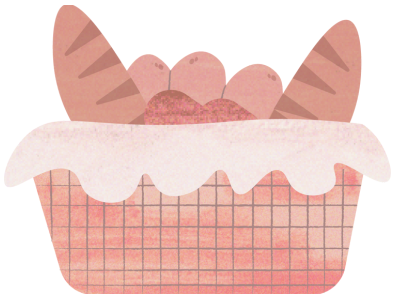
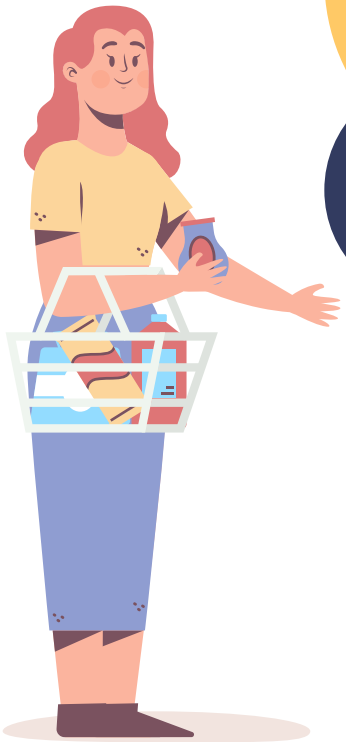
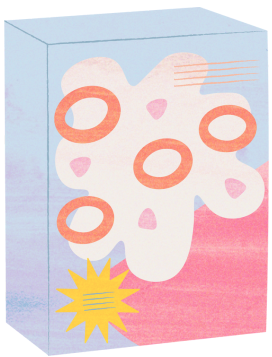
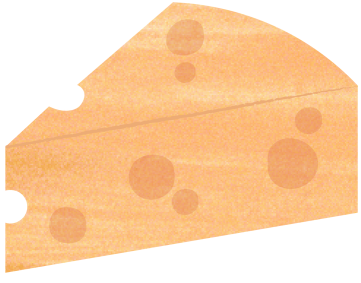




CUSTOMER PERSONALITY CLUSTERING

with Kmeans & PCA

May.2023 Ashley Bao
ashley_bao@outlook.com


Introduction

Customer Personality Analysis is a detailed analysis of a company’s ideal customers. It helps a business to better understand its customers and makes it easier for them to modify products and services according to the specific needs, behaviors, and concerns of different types of customers.

The objective of this project is to divide the existing customers into groups that reflect similarities among customers in each cluster with an unsupervised clustering machine learning method K-means, and the Principal Component Analysis to reduce the dimensionality which helps increase interpretability while preserving the maximum amount of information.

Data Attributes

- People

ID: Customer's unique identifier

Year_Birth: Customer's birth year

Education: Customer's education level

Marital_Status: Customer's marital status

Income: Customer's yearly household income

Kidhome: Number of children in customer's household

Teenhome: Number of teenagers in customer's household

Dt_Customer: Date of customer's enrollment with the company

Recency: Number of days since customer's last purchase

Complain: 1 if the customer complained in the last 2 years, 0 otherwise

- Products

MntWines: Amount spent on wine in last 2 years

MntFruits: Amount spent on fruits in last 2 years

MntMeatProducts: Amount spent on meat in last 2 years

MntFishProducts: Amount spent on fish in last 2 years

MntSweetProducts: Amount spent on sweets in last 2 years

MntGoldProds: Amount spent on gold in last 2 years

- Promotion

NumDealsPurchases: Number of purchases made with a discount

AcceptedCmp1: 1 if customer accepted the offer in the 1st campaign, 0 otherwise

AcceptedCmp2: 1 if customer accepted the offer in the 2nd campaign, 0 otherwise

AcceptedCmp3: 1 if customer accepted the offer in the 3rd campaign, 0 otherwise

AcceptedCmp4: 1 if customer accepted the offer in the 4th campaign, 0 otherwise

AcceptedCmp5: 1 if customer accepted the offer in the 5th campaign, 0 otherwise

Response: 1 if customer accepted the offer in the last campaign, 0 otherwise

- Place

NumWebPurchases: Number of purchases made through the company's website

NumCatalogPurchases: Number of purchases made using a catalogue

NumStorePurchases: Number of purchases made directly in stores

NumWebVisitsMonth: Number of visits to company's website in the last month

Contents

1. Loading Libraries and Data

The first step is to load the necessary packages and import the dataset into pandas.

```
import numpy as np
import pandas as pd
pd.set_option('display.max_columns', 500)

import seaborn as sns
import matplotlib.pyplot as plt

from datetime import date
from datetime import datetime
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

import warnings
warnings.filterwarnings('ignore')

plt.rcParams["axes.prop_cycle"] = plt.cycler("color", plt.cm.Set2.colors)
```

```
df = pd.read_csv("../Customer Personality/marketing_campaign.csv",sep="\t")
```

```
df.head()
```

]:

	Education	Income	Kidhome	Teenhome	Recency	Wines	Fruits	Meat	Fish	Sweets	Gold	NumDealsPurchases	NumWebPurchases	NumCatalogPu
0	Graduate	58138.0	0	0	58	635	88	546	172	88	88	3	8	
1	Graduate	46344.0	1	1	38	11	1	6	2	1	6	2	1	
2	Graduate	71613.0	0	0	26	426	49	127	111	21	42	1	8	
3	Graduate	26646.0	1	0	26	11	4	20	10	3	5	2	2	
4	Postgraduate	58293.0	1	0	94	173	43	118	46	27	15	5	5	

2. Understanding the Data

Before cleaning the dataset, we need a full understanding of the data structure.

```
df.shape
```

]: (2240, 29)

```
df.columns
```

```
]: Index(['ID', 'Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidhome',
       'Teenhome', 'Dt_Customer', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Z_CostContact', 'Z_Revenue', 'Response'],
      dtype='object')
```

This dataset contains 2240 rows and 29 columns. Here we break the features into four parts.

Customers' Information	Products (Amount in last 2 years)	Promotion	Place (Purchase Channel)
· ID · Year_Birth · Educatio · Marital_Status · Income · Kidhome · Teenhome · Dt_Customer · Recency · Complain	· MntWines · MntFruits · MntMeatProducts · MntFishProducts · MntSweetProducts · MntGoldProds	· NumDealsPurchases · AcceptedCmp1 · AcceptedCmp2 · AcceptedCmp3 · AcceptedCmp4 · AcceptedCmp5 · Response	· NumWebPurchases · NumCatalogPurchases · NumStorePurchases · NumWebVisitsMonth

By further exploring the dataset, we can understand the data type, unique values, and number of these unique values of each column, which are listed below.

```
def report(df):
    col = []
    d_type = []
    uniques = []
    n_uniques = []

    for i in df.columns:
        col.append(i)
        d_type.append(df[i].dtypes)
        uniques.append(df[i].unique()[:3])
        n_uniques.append(df[i].nunique())

    return pd.DataFrame({'Column': col, 'd_type': d_type, 'unique_sample': uniques, 'n_uniques': n_uniques})


pd.set_option('max.colwidth', 100)

report(df)
```

]:

	Column	d_type	unique_sample	n_uniques
0	ID	int64	[5524, 2174, 4141]	2240
1	Year_Birth	int64	[1957, 1954, 1965]	59
2	Education	object	[Graduation, PhD, Master]	5
3	Marital_Status	object	[Single, Together, Married]	8
4	Income	float64	[58138.0, 46344.0, 71613.0]	1974
5	Kidhome	int64	[0, 1, 2]	3
6	Teenhome	int64	[0, 1, 2]	3
7	Dt_Customer	object	[04-09-2012, 08-03-2014, 21-08-2013]	663
8	Recency	int64	[58, 38, 26]	100
9	MntWines	int64	[635, 11, 426]	776
10	MntFruits	int64	[88, 1, 49]	158
11	MntMeatProducts	int64	[546, 6, 127]	558
12	MntFishProducts	int64	[172, 2, 111]	182
13	MntSweetProducts	int64	[88, 1, 21]	177
14	MntGoldProds	int64	[88, 6, 42]	213
15	NumDealsPurchases	int64	[3, 2, 1]	15
16	NumWebPurchases	int64	[8, 1, 2]	15
17	NumCatalogPurchases	int64	[10, 1, 2]	14
18	NumStorePurchases	int64	[4, 2, 10]	14
19	NumWebVisitsMonth	int64	[7, 5, 4]	16
20	AcceptedCmp3	int64	[0, 1]	2
21	AcceptedCmp4	int64	[0, 1]	2
22	AcceptedCmp5	int64	[0, 1]	2
23	AcceptedCmp1	int64	[0, 1]	2
24	AcceptedCmp2	int64	[0, 1]	2
25	Complain	int64	[0, 1]	2
26	Z_CostContact	int64	[3]	1
27	Z_Revenue	int64	[11]	1
28	Response	int64	[1, 0]	2

3. Data Cleaning & Feature Engineering

In this part, we are going to create some new features based on the existing information for better understanding of the customers' personalities in further analysis.

First of all, for demographic information, we created 'Age' based on the 'Year_Brith'; 'Education' to indicate the education background; 'Living_With' to show if they live with partner or alone; 'Children', 'Family_Size' and 'Is_Parent' to show the details of the family structure.

```
df['Age'] = (pd.Timestamp('now').year - df['Year_Birth'])
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'])
```

```
months = []
for i in df['Dt_Customer']:
    customermonth = i.month
    customeryear = i.year
    monthcount = pd.Timestamp('now').month - customermonth + 12*(pd.Timestamp('now').year - i.year)
    months.append(monthcount)

df['months_customer'] = months
df['months_customer'] = pd.to_numeric(df['months_customer'], errors="coerce")
```

```
df['Living_With']=df['Marital_Status'].replace({'Married':'Partner', 'Together':'Partner', 'Absurd':'Alone',
                                                'Widow':'Alone', 'YOLO':'Alone', 'Divorced':'Alone', 'Single':'Alone',})

df["Children"]=df["Kidhome"]+df["Teenhome"]

df["Family_Size"] = df["Living_With"].replace({"Alone": 1, "Partner":2})+ df["Children"]

df["Is_Parent"] = np.where(df.Children> 0, 1, 0)

df["Education"]=df["Education"].replace({"Basic":"Undergraduate","2n Cycle":"Undergraduate", "Graduation":"Graduate",
                                         "Master":"Postgraduate", "PhD":"Postgraduate"})
```

For features related to the store and shipping habits, we created 'months_customer' to show how many months being the customer; 'total_expenses' to show the total amount spent in the last two years, 'pct_essentials' is the percentage of the amount on essentials items, 'num_perchases' is the total number of the purchase of all channels.

```
df['total_expenses'] = df['MntWines'] + df['MntFruits'] + df[
    'MntMeatProducts'] + df['MntFishProducts'] + df['MntSweetProducts'] + df['MntGoldProds']

df['pct_essentials'] = (df['MntWines'] + df['MntFruits'] + df[
    'MntMeatProducts'] + df['MntFishProducts'] + df['MntSweetProducts'])/df['total_expenses']

df['num_purchases'] = df['NumWebPurchases'] + df['NumCatalogPurchases'] + df['NumStorePurchases']

df=df.rename(columns={"MntWines": "Wines","MntFruits":"Fruits","MntMeatProducts":"Meat",
                      "MntFishProducts":"Fish","MntSweetProducts":"Sweets","MntGoldProds":"Gold"})
```

After adding the new features, let us drop the unnecessary columns and then review the data structure and check the missing value as well.

```
df.drop(columns=["Marital_Status", "Dt_Customer", "Z_CostContact", "Z_Revenue", "Year_Birth", "ID"], inplace=True)
```

```
df.describe()
```

]:

	Income	Kidhome	Teenhome	Recency	Wines	Fruits	Meat	Fish	Sweets	Gold	NumD
count	2216.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	
mean	52247.251354	0.444196	0.506250	49.109375	303.935714	26.302232	166.950000	37.525446	27.062946	44.021875	
std	25173.076661	0.538398	0.544538	28.962453	336.597393	39.773434	225.715373	54.628979	41.280498	52.167439	
min	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	35303.000000	0.000000	0.000000	24.000000	23.750000	1.000000	16.000000	3.000000	1.000000	9.000000	
50%	51381.500000	0.000000	0.000000	49.000000	173.500000	8.000000	67.000000	12.000000	8.000000	24.000000	
75%	68522.000000	1.000000	1.000000	74.000000	504.250000	33.000000	232.000000	50.000000	33.000000	56.000000	
max	666666.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000	263.000000	362.000000	

```
df.isnull().sum()
```

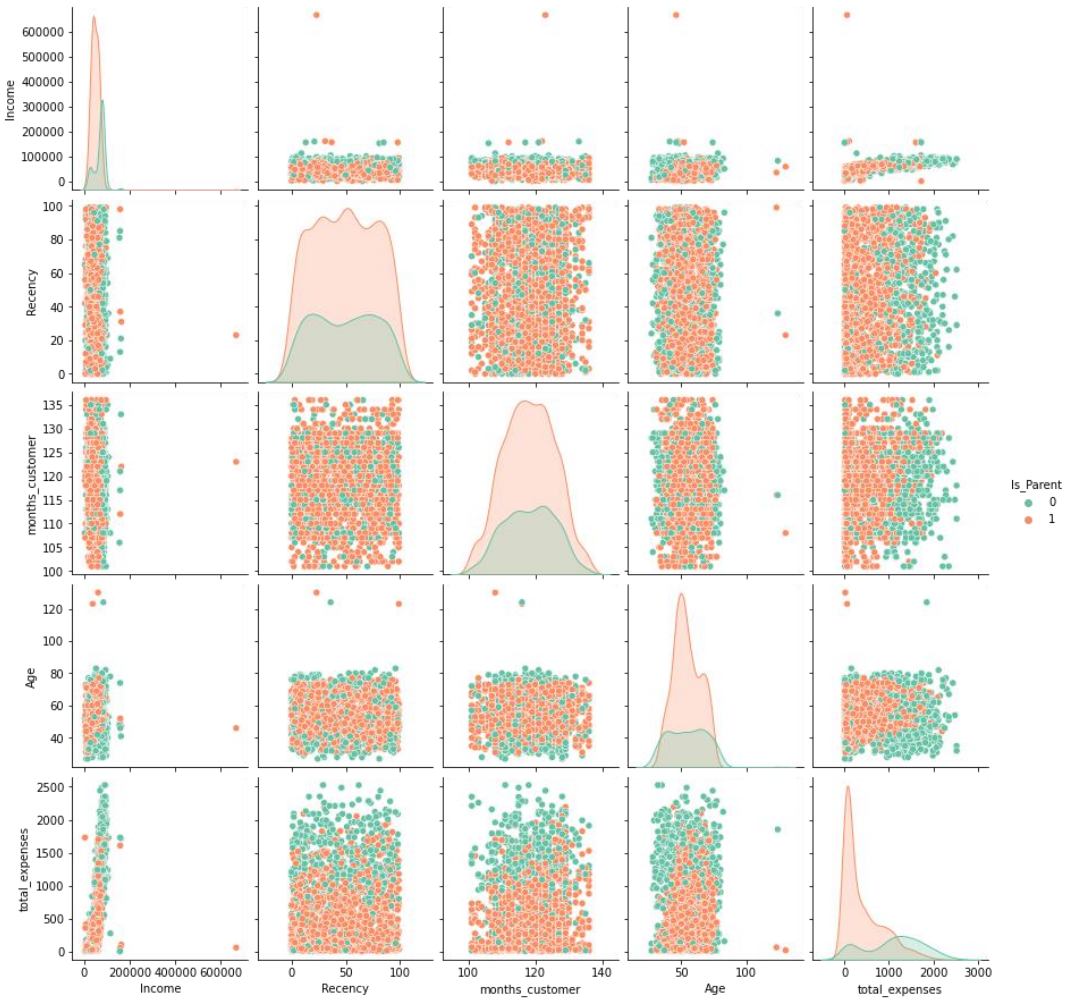
```
Education              0
Income                24
Kidhome                0
Teenhome               0
Recency                0
Wines                  0
Fruits                 0
Meat                   0
Fish                   0
Sweets                 0
Gold                   0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
AcceptedCmp3           0
AcceptedCmp4           0
AcceptedCmp5           0
AcceptedCmp1           0
AcceptedCmp2           0
Complain               0
Response               0
Age                    0
months_customer        0
Living_With            0
Children               0
Family_Size            0
Is_Parent              0
total_expenses         0
pct_essentials         0
num_purchases          0
dtype: int64
```

It seems 'Income' has 24 missing values, and we will drop these records.

```
df.dropna(subset=['Income'],inplace=True)
```

Next, we will check the outliers by using the pairplot from seaborn.

```
pairplot = ["Income", "Recency", "months_customer", "Age", "total_expenses", "Is_Parent"]
sns.pairplot(df[pairplot],hue= "Is_Parent")
```



From the chart above, there are some outliers in 'Age' and 'Income', so we will drop these records as well.

After adding new features and cleaning the data, we have a dataset with 2212 rows and 32 columns.

```
df = df[(df['Age'] < 90) & (df['Income'] < 600000)].reset_index(drop = True)
```

```
df.shape
```

```
]: (2212, 32)
```

4. Label Encoding & Standardization

Before entering clustering, we need to preprocess the data first, which includes three major steps: label encoding, standardization, and dimensionality reduction. In this part, we will conduct the first two steps.

Since K-means can only process numerical data, label encoding can help to convert categorical columns into numerical ones. In the dataset, 'Education' and 'Living_With' are categorical, so they will be transferred to numerical data.

```
df_tobescaled = df.copy()

s = (df_tobescaled.dtypes == 'object')
object_cols = list(s[s].index)

for i in object_cols:
    df_tobescaled[i]=df_tobescaled[[i]].apply(LabelEncoder().fit_transform)
```

After checking the dataset, we found that the columns of 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1', 'AcceptedCmp2', 'Complain', 'Response' are not much helpful in providing insights for the analysis, we will drop them as well.

Next, we will perform the standardization with StandardScaler, which is the process of putting different variables on the same scale. This process allows you to compare scores between different types of variables.

```
cols_del = ['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1','AcceptedCmp2', 'Complain', 'Response']

df_tobescaled = df_tobescaled.drop(cols_del, axis=1)

scaler = StandardScaler()
scaler.fit(df_tobescaled)
df_scaled = pd.DataFrame(scaler.transform(df_tobescaled),columns= df_tobescaled.columns)
```

```
df_scaled.head()
```

	Education	Income	Kidhome	Teenhome	Recency	Wines	Fruits	Meat	Fish	Sweets	Gold	NumDealsPurchases	NumWebPu
0	-0.893586	0.287105	-0.822754	-0.929699	0.310353	0.977660	1.552041	1.690293	2.453472	1.483713	0.852576	0.351030	1
1	-0.893586	-0.260882	1.040021	0.908097	-0.380813	-0.872618	-0.637461	-0.718230	-0.651004	-0.634019	-0.733642	-0.168701	-1
2	-0.893586	0.913196	-0.822754	-0.929699	-0.795514	0.357935	0.570540	-0.178542	1.339513	-0.147184	-0.037254	-0.688432	1
3	-0.893586	-1.176114	1.040021	-0.929699	-0.795514	-0.872618	-0.561961	-0.655787	-0.504911	-0.585335	-0.752987	-0.168701	-0
4	0.571657	0.294307	1.040021	-0.929699	1.554453	-0.392257	0.419540	-0.218684	0.152508	-0.001133	-0.559545	1.390492	0

5. Reducing Dimensionality with PCA

Since the dataset has a high number of features, we need Principal component analysis(PCA) to reduce the dimensionality in order to increase the interpretability of data while preserving the maximum amount of information.

In this project, we will set the number of components as 3, which means it will transform our original data into a three-dimensional space.

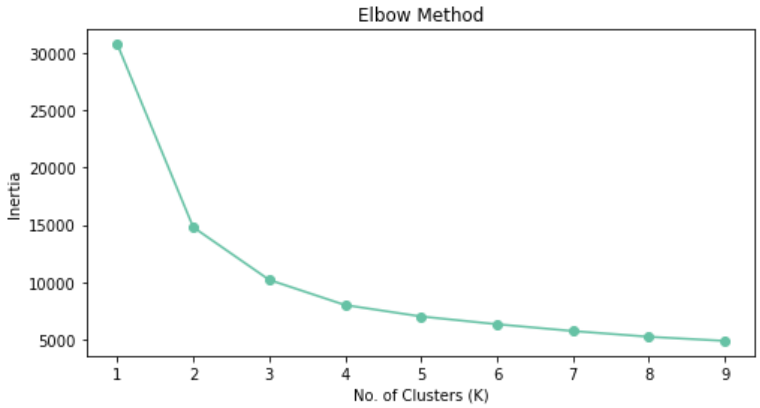
```
pca = PCA(n_components=3)
pca.fit(df_scaled)
PCA_dfs = pd.DataFrame(pca.transform(df_scaled), columns=(["col1","col2", "col3"]))
PCA_dfs.describe().T
```

	count	mean	std	min	25%	50%	75%	max
col1	2212.0	2.399126e-16	3.029910	-6.017648	-2.766027	-0.745733	2.663469	7.739259
col2	2212.0	1.816911e-17	1.773255	-4.751027	-1.380253	-0.196702	1.301324	6.148118
col3	2212.0	2.604908e-17	1.261479	-3.448128	-0.859644	-0.023439	0.839163	5.398811

6. Clustering with K-means

Next, we are going to run the K-means analysis. Before that, there is a fundamental step for any unsupervised algorithm, which is to determine the optimal number of clusters into which the data may be clustered. Here, the Elbow Method, which is the most commonly used method, will be implemented to determine the number of clusters.

```
wcss = []
for i in range(1, 10):
    km = KMeans(n_clusters = i, random_state = 42)
    km.fit(PCA_dfs)
    wcss.append(km.inertia_)

fig, ax = plt.subplots(figsize=(8, 4))

ax.plot(range(1,10), wcss,'-o')

ax.set_title("Elbow Method")
ax.set_xlabel('No. of Clusters (K)')
ax.set_ylabel('Inertia')
```

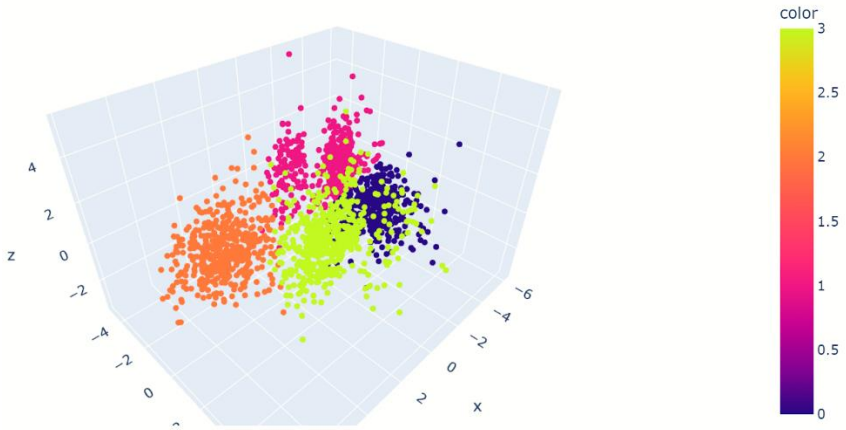


The chart above indicates that 4 should be the optimal number of clusters for this dataset. Therefore, we will apply "n_cluster = 4" to run the k-means model to generate the final result of clustering.

```
k = 4
kmeans = KMeans(n_clusters = 4, init = 'k-means++', random_state = 42)

y = kmeans.fit_predict(PCA_dfs)
#y = array([0, 1, 2, ..., 0, 2, 1])
PCA_dfs['Cluster'] = y
df['Cluster'] = y
```

In the picture below, we have plotted the 3D chart to visualize the distribution of each cluster by using Plotly.

```
x =PCA_dfs["col1"]
y =PCA_dfs["col2"]
z =PCA_dfs["col3"]

import plotly.express as px

fig = px.scatter_3d(df, x, y, z, color=PCA_dfs["Cluster"])
fig.update_traces(marker_size = 3)

fig.show()
```

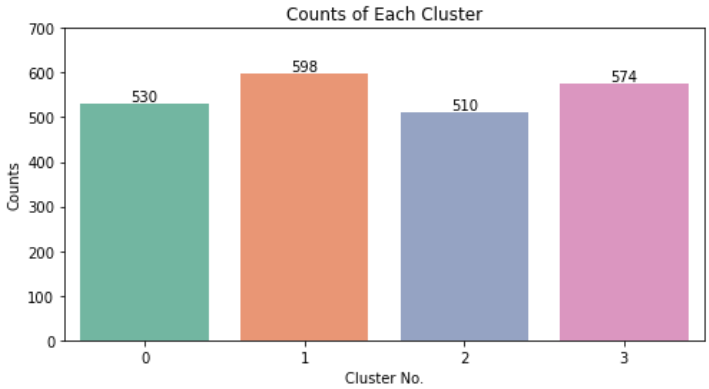


7. Evaluating the Model

So far, we have finished the clustering, let us explore the patterns and details of the results that the model provided.

First of all, we need to know the number of each cluster. As indicated below, we have 530, 598 510 and 574 for each cluster. The numbers are quite evenly distributed as they are all in the range of 500 to 600.

```
fig, ax = plt.subplots(figsize=(8, 4))

ax = sns.barplot(y=df['Cluster'].value_counts().values,
            x=df['Cluster'].value_counts().index
           )

for i in ax.containers:
    ax.bar_label(i,)

ax.set_title("Counts of Each Cluster")
ax.set_xlabel('Cluster No.')
ax.set_ylabel('Counts')
ax.set_ylim(0, 700)
```

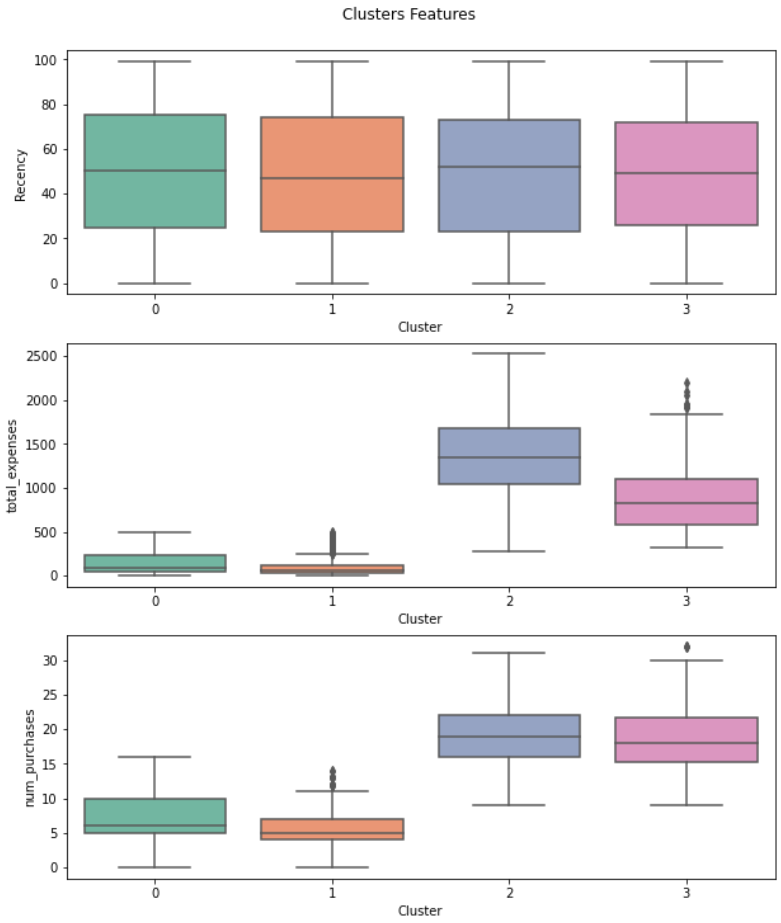


Next, let us check the customers' performance based on the RFM matrix. RFM stands for Recency, Frequency, and Monetary value, each corresponding to some key customer trait. In this case, we will use recency, the number of purchases and total expenses to show the customers' characteristics.

Below is the RFM of each cluster. We can see that the model clearly differentiates each cluster based on the expenses and the number of purchases. Cluster 2 shows the highest expenses and number of purchases, followed by Cluster 3. Cluster 0 shows low expenses and number of purchases and Cluster 1 has the lowest in both indicators.

However, the recency does not show impressive differences.

```
fig, axes = plt.subplots(3, 1, figsize=(10,12))

sns.boxplot(ax=axes[0], data=df, x='Cluster', y = 'Recency')
sns.boxplot(ax=axes[1], data=df, x='Cluster', y = 'total_expenses')
sns.boxplot(ax=axes[2], data=df, x='Cluster', y = 'num_purchases')

fig.suptitle('Clusters Features',y =0.92)
```



We will continue by visualizing the income level of each cluster. The plot below shows obvious relationships between expenses and income levels that the high income, the high expenses. To be more detailed, we can divide the data into four categories, among them it is quite clear that Cluster 1 and 3 are our most valuable customers.

a) Cluster 2: highest income, highest expenses.

b) Cluster 3: high income, average expenses

c) Cluster 1: low income, less expenses

d) Cluster 0: lowest income, lowest expenses

```
fig, ax = plt.subplots(figsize=(8, 4))
sns.color_palette("Set2")

sns.scatterplot(data=df, x='Income', y='total_expenses', hue='Cluster',palette='Set2')

ax.set_title('Relation Between Expenses and Income Level for Each Cluster')
ax.set_xlabel('Income')
ax.set_ylabel('Total Expenses')
```

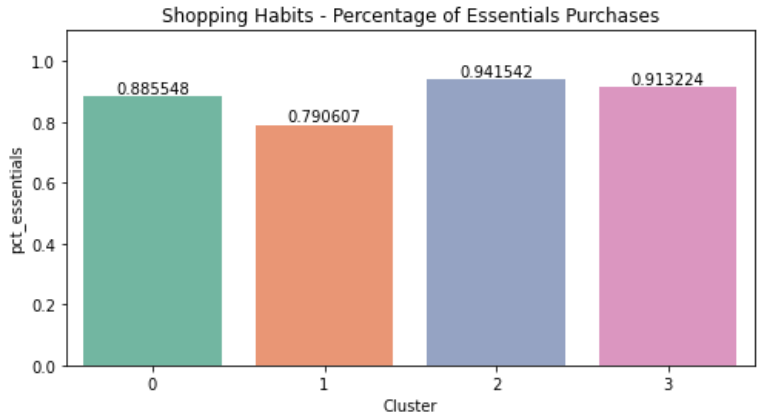
8. Profiling

After generating the general ideas of each cluster, let us dive into the details of the shopping behaviors and demographic information of each cluster.

1) Shopping Behaviors

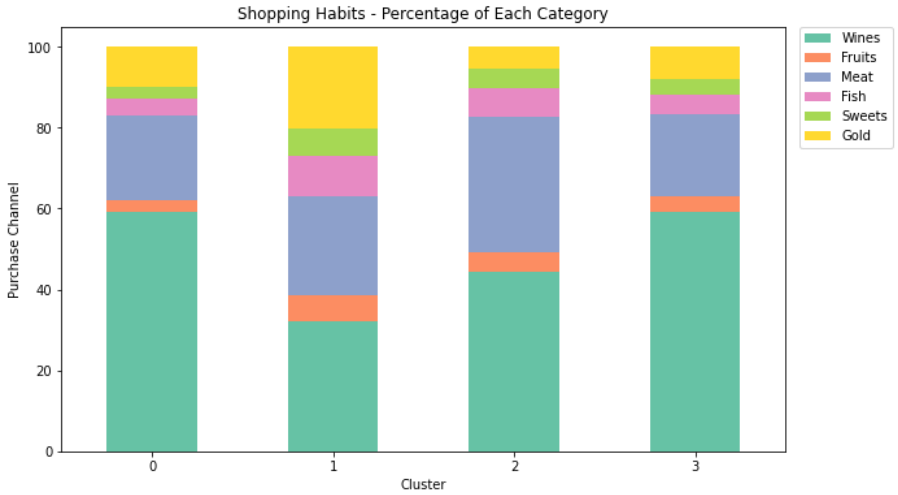
First, we will explore the expenses for each product. For the essentials (product of Wines, Fruits, Meat, Fish, Sweets), compared to other clusters, Cluster 1 shows lower percentage in the essentials. Cluster 2 and 3 shows a higher percentage in the essentials.

```
fig, ax = plt.subplots(figsize=(8, 4))

sns.barplot(x=df['Cluster'], y=df['pct_essentials'],ci=None)

for i in ax.containers:
    ax.bar_label(i,)

ax.set_title('Shopping Habits - Percentage of Essentials Purchases')
ax.set_xlabel('Cluster')
ax.set_ylabel('pct_essentials')

ax.set_ylim(0, 1.1)
```



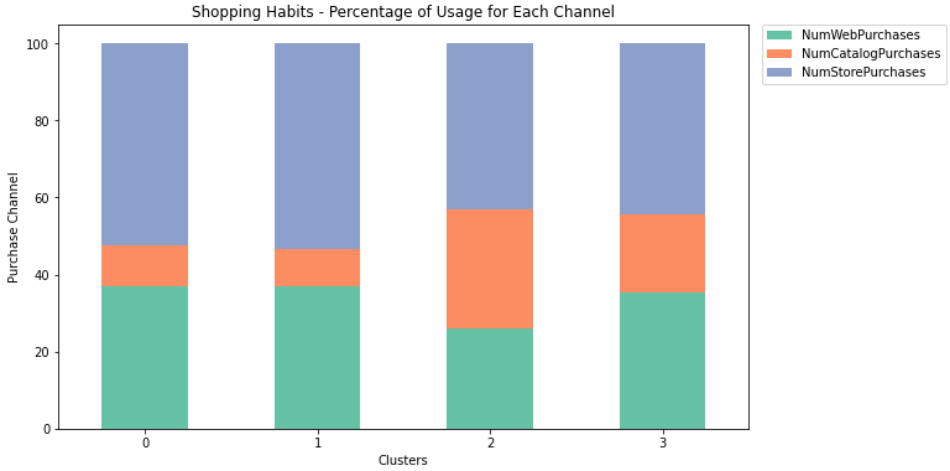
The chart below shows the detailed proportion of each category in terms of spending. Overall, all clusters spend the most on Wine, followed by Meat. Cluster 2 tends to spend more on Meat compared to other Clusters.

```
expenses = df.groupby('Cluster')[['Wines','Fruits','Meat',
                                  'Fish','Sweets','Gold']].sum()

expenses = expenses.apply(lambda x: x * 100 / sum(x), axis=1)

expenses.plot(kind='bar', stacked=True, figsize=(10,6))

plt.title('Shopping Habits - Percentage of Each Category')
plt.xlabel('Cluster')
plt.ylabel('Purchase Channel');
plt.xticks(rotation=0, ha='center')
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```



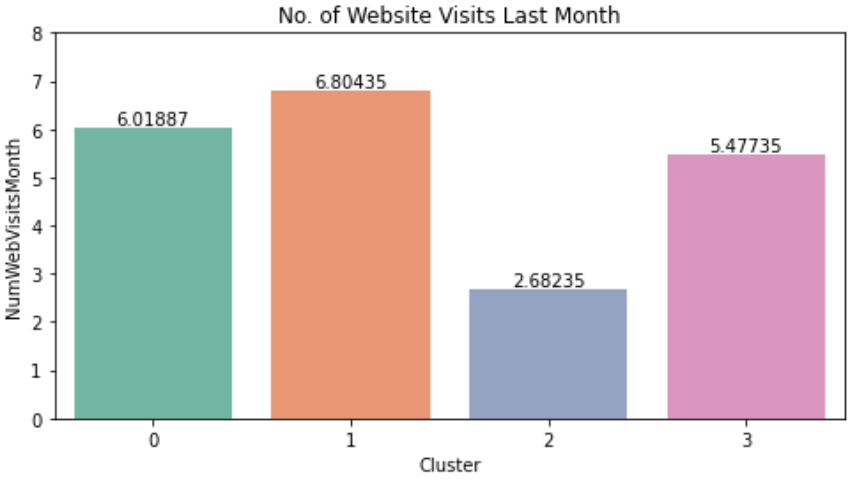
Next, we will look at the behavior of purchase channels for each cluster. Overall, in store purchase is the most popular one for all clusters. For Cluster 1,2 and 4, online purchase is the secondary. Cluster 3 shows a higher tendency to purchase via catalog as opposed to purchase via online.

```
channel = df.groupby('Cluster')[['NumWebPurchases','NumCatalogPurchases',
                                 'NumStorePurchases']].sum()

channel = channel.apply(lambda x: x * 100 / sum(x), axis=1)

channel.plot(kind='bar', stacked=True, figsize=(10,6))

plt.title('Shopping Habits - Percentage of Usage for Each Channel')
plt.xlabel('Clusters')
plt.ylabel('Purchase Channel');
plt.xticks(rotation=0, ha='center')
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```



The next chart exhibits the number of website visits in the last month. The number of 2.68 times of Cluster 2 shows this group visits the website least frequently among all clusters, followed by Cluster 3. This result is consistent with the Shopping Habits - Percentage of Usage of Each Channel that Cluster 2 and 3 do not utilize online shopping as frequently as Cluster 0 and 1.

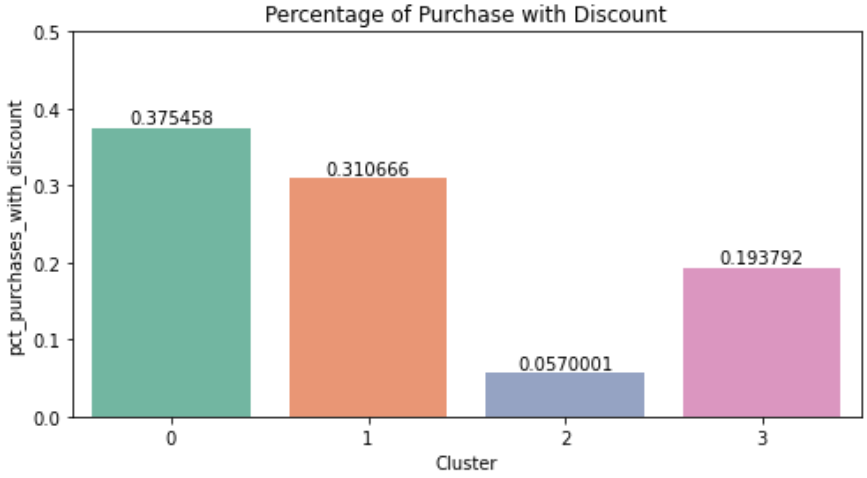
```
fig, ax = plt.subplots(figsize=(8, 4))

sns.barplot(x=df['Cluster'], y=df['NumWebVisitsMonth'],ci=None)

for i in ax.containers:
    ax.bar_label(i,)

ax.set_title('No. of Website Visits Last Month')
ax.set_xlabel('Cluster')
ax.set_ylabel('NumWebVisitsMonth')
ax.set_ylim(0, 8)
```



The next chart is about the proportion of the purchase with discount against without discount. Cluster 0 uses discount most frequently followed by Cluster 1. These are two groups that with lower income and expenses. The most valuable customer groups, Cluster 2 and 3, use less discount when they shop.

```
deal = df.groupby('Cluster')[['NumDealsPurchases','num_purchases']].sum()
deal['pct_purchases_with_discount'] = deal['NumDealsPurchases']/deal['num_purchases']
```

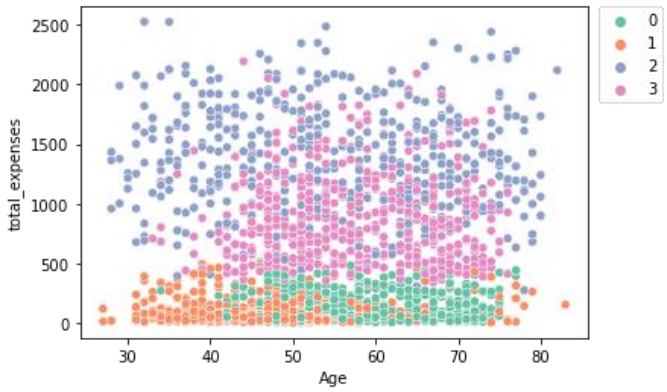
```
fig, ax = plt.subplots(figsize=(8, 4))

sns.barplot(x=deal.index, y=deal['pct_purchases_with_discount'],ci=None)
for i in ax.containers:
    ax.bar_label(i,)
ax.set_title('Percentage of Purchase with Discount')
ax.set_xlabel('Cluster')
ax.set_ylabel('pct_purchases_with_discount')
ax.set_ylim(0, 0.5)
```



2) Demographic Information

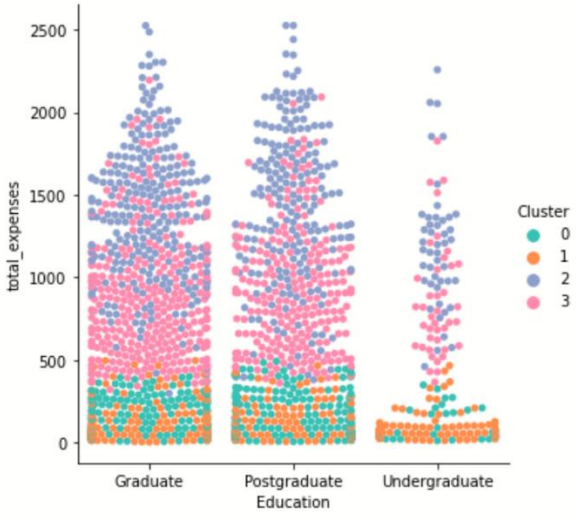
Let us move on to the demographic characteristics of each cluster. First, we plot the age distribution of total dataset. Cluster 2 spreads relatively evenly from 20s to 70s. Cluster 3 and Cluster 0 are located more among 40s to 70s. Cluster 1 represents the younger generation scatters more among 20s to 50s.

```
sns.scatterplot(data = df, x='Age', y='total_expenses',hue='Cluster',
                palette='Set2' )
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
```



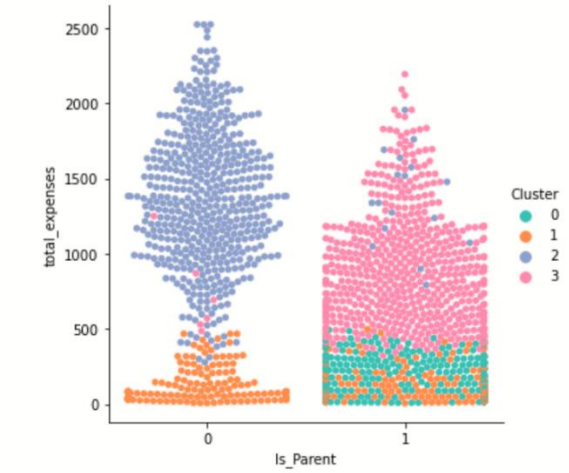
The plot below regarding education level of each cluster does not provide overwhelming differences.

```
sns.catplot(data = df, x='Education', y='total_expenses',hue='Cluster', kind="swarm");
```

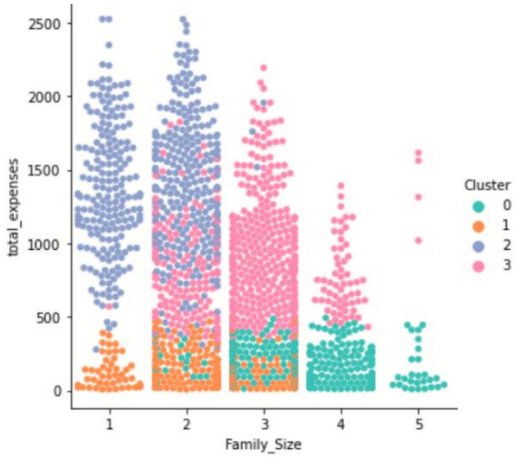


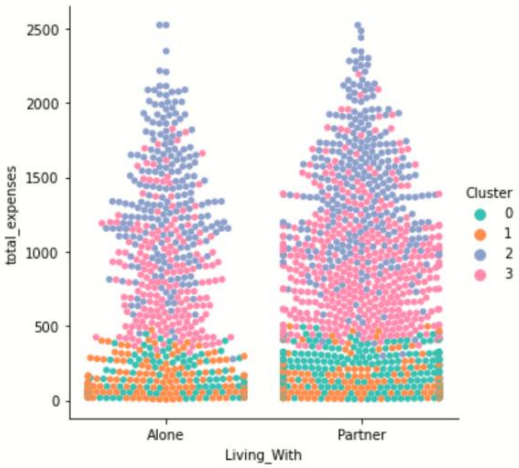
The chart below shows if the customer is the parent for each cluster. It is quite clear that most of the customers in Cluster 2 is not parent and most of Cluster 3 and Cluster 0 is parent.

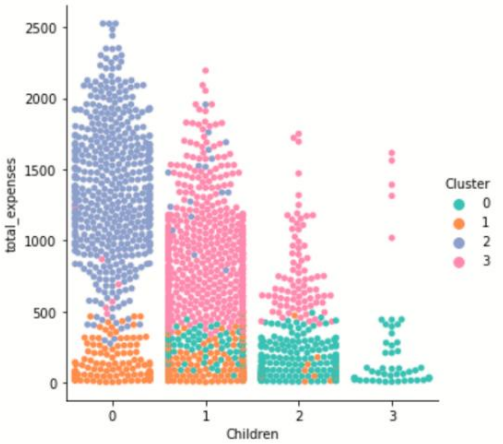
```
sns.catplot(data = df, x='Is_Parent', y='total_expenses',hue='Cluster',kind="swarm")
```

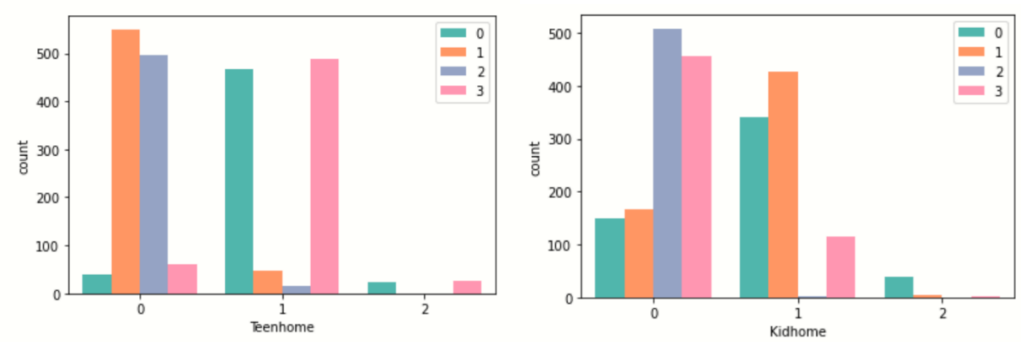


The family size, if lives with partner and the number of children information below indicates that Cluster 2 lives alone or with a partner without child, Cluster 3 lives with one child or two children (mostly teenager) with or without a partner, Cluster 1 tends to have 0 or one child (mostly kid) and Cluster 0 tends to have more than one child.









9. Segmentation Summary

Now, based on all the information generated in the previous section, we are going to summarize the characteristics of each cluster which can help us better understand the customer and lead to effective business decision-making.

a) Cluster 2: highest income, highest expenses

- Mostly not a parent
- Max only 2 members in the family
- Span all ages
- Consume relatively more meat
- Purchase frequently more via catalog and less via online

b) Cluster 3: high income, average expenses

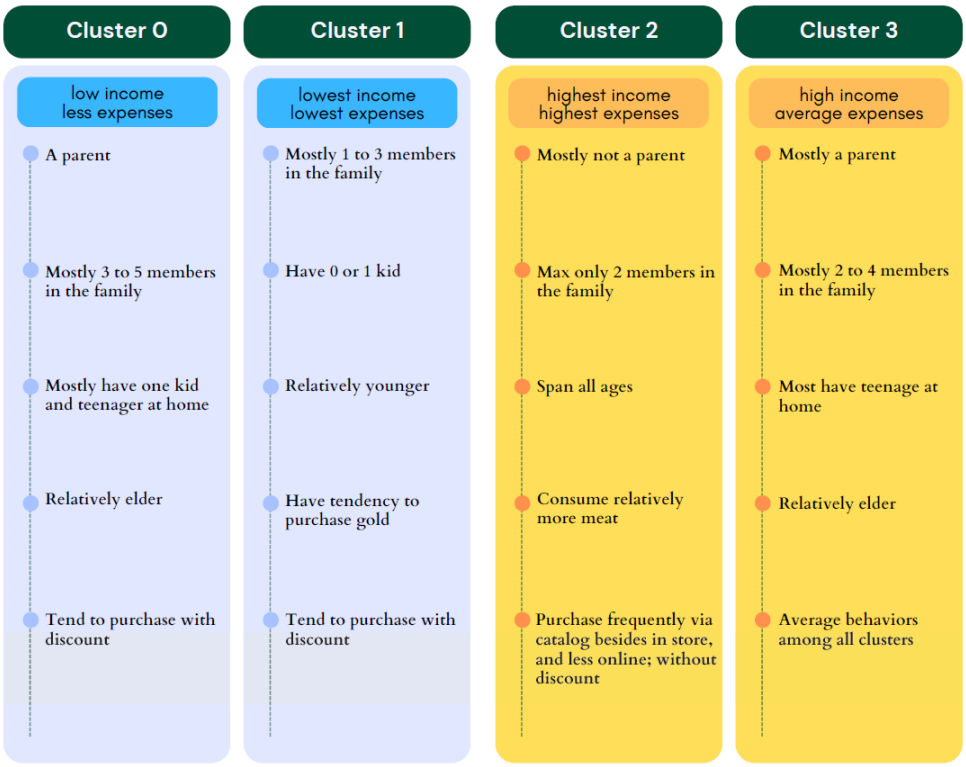
- Mostly a parent
- Mostly 2 to 4 members in the family
- Most have teenage at home
- Relatively elder

c) Cluster 0: low income, less expenses

- A parent
- Mostly 3 to 5 members in the family
- Mostly have one kid and teenager at home
- Relatively elder
- Tend to purchase with discount

d) Cluster 1: lowest income, lowest expenses

- Mostly 1 to 3 members in the family
- Have 0 or 1 kid
- Relatively younger
- Tend to purchase with discount



10. Conclusion

In this project, in order to better understand the different characteristics of our customer, we have performed the k-means analysis along with PCA to reduce the dimensionality. As a result, the analysis successfully segmented the whole customer set into four clusters each with distinct characteristics in terms of shopping behaviors, family structures and income.

Based on the information generated, the company can develop customized strategies such as marketing campaigns, product pricing and distribution strategy.