Applying Data Mining for Consumer Behavior Analysis in E-commerce

Group 4

Irving Covarrubias Medina (0278032), Chang Li (0275645), Fiorella Maza (0260804), Darsini Unnikrishnan (0269697), Abhirami Pradeep (0269518), and Di Fei (0271759)

Shannon School of Business, Cape Breton University

MGSC-5126-11 2023 Fall Data Mining

Instructor: Ebrahim Sharifi

December 8th, 2023

Table of Contents

Applying Data Mining for Consumer Behavior Analysis in E-commerce

1. Introduction

Hunter's e-grocery is a popular French brand known for its online grocery and lifestyle products. It operates in ten countries and is committed to building strong customer loyalty through anticipating and meeting customer demands. However, a series of black swan events such as the Covid-19 pandemic, the Ukraine crisis, and recently Isarel-Palestine conflict have changed the typical pattern of consumer behaviour in the past.

With the existing value proposition being challenged, in response, the objective of our study is to assist Hunter's e-grocery in verifying new patterns of consumer habits to keep customer retention and satisfaction. For this study, we'll use the Apriori association and K-means clustering models. These models are customized to identify frequent purchasing patterns and unknown classification of consumer characteristics. The goal is to make Hunter's e-grocery more adaptable and customer-focused, ensuring long-term success.

The dataset was extracted from Kaggle (Rupesh Kumar, 2023), comprising 2,019,501 Rows & 12 Columns with the following attributes:

order_id – (A unique number to identity the order)

user_id - (A unique number to identify the user)

order_number – (Number of the order)

order_dow – (Day of the Week the order was made)

order_hour_of_day – (Time of the order)

days_since_prior_order - (History of the order)

product_id – (Id of the product)

add_to_cart_order – (Number of items added to cart)

reordered – (If the reorder took place)

department_id - (Unique number allocated to each department)

department – (Names of the departments)

product_name – (Name of the products)

2. Association: Apriori approach

2.1. Procedures of Apriroi Approach

- **Step 1: Loading data**

 # Read the CSV file downloaded from Kaggle into a data frame with headers

```
your_data <- read.csv("E:\\Dropbox\\Academies\\CBU\\MGSC-5126-11 Data Mining\\Grp Ass\\Data\\Ecommerce_consumer behaviour.csv", header = TRUE)
```

- **Step 2: Data transformation**

Before applying Apriori model with 'arules' pacakge in R, the dataset needs to be transformed into a transaction format with unique transaction ID in each row and the other column with item names separated by comma.

 # Keep only columns "order_id" and "product_name" relevant to Apriori analysis

```
your_data <- your_data[, c("order_id", "product_name")]
```

 # Load the packages

```
library(arules)
library(arulesViz)
library(RColorBrewer)
```

 # Combine rows with the same 'order_id'

```
combined_data <- aggregate (product_name ~ order_id, data = your_data, paste, collapse = ",")
```

 # Convert the combined data to transaction format

```
transactions <- read.transactions(textConnection(combined_data$product_name), format = "basket", sep = ",")
```

 # Inspect the transactions

```
inspect(transactions)
```

- **Step 3: Applying Apriori() function**

 # Run Apriori function with support and confidence = 0.01 and 0.2

```
rules <- apriori(transactions, parameter = list(support = 0.01, confidence = 0.2))
```

- **Step 4: Applying insepct() function**

 # Sort rules by lift

```
rules <- sort(rules, by = "lift", decreasing = TRUE)
```

 # Inspect the top 5 rules with the highest strength measured by lift

```
top_5_rules <- head(rules, n = 5)

inspect(top_5_rules)
```

The top 5 rules ranked by its lift are listed below

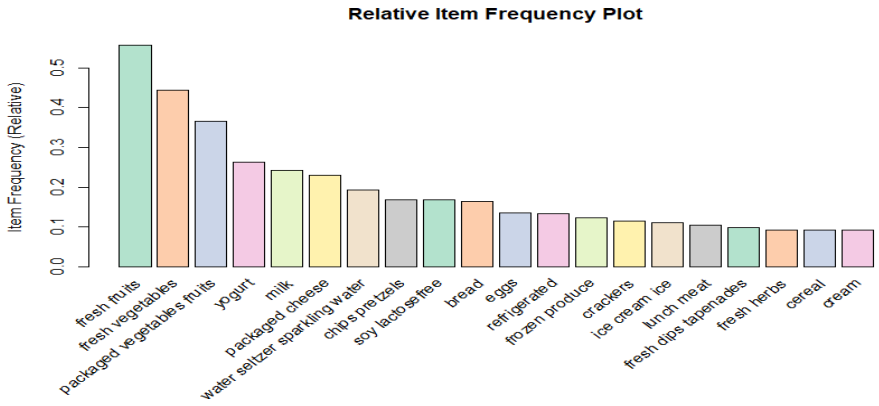
```
> inspect(top_5_rules)
    lhs                                rhs             support confidence coverage     lift count
[1] {dry pasta,
     packaged vegetables fruits}    => {pasta sauce} 0.010855  0.2978869 0.036440 4.754021  2171
[2] {packaged vegetables fruits,
     pasta sauce}                   => {dry pasta}   0.010855  0.3330265 0.032595 4.753787  2171
[3] {fresh fruits,
     fresh vegetables,
     pasta sauce}                   => {dry pasta}   0.010145  0.3227808 0.031430 4.607534  2029
[4] {fresh vegetables,
     pasta sauce}                   => {dry pasta}   0.013080  0.3189855 0.041005 4.553358  2616
[5] {fresh fruits,
     pasta sauce}                   => {dry pasta}   0.013610  0.3176937 0.042840 4.534919  2722
```

- **Step 5: Applying itemFrequencyPlot() function**

 # Applying itemFrequencyPlot() function to list the top 20 items with the highest frequency

```
arules::itemFrequencyPlot(transactions, topN = 20, col = brewer.pal(8, 'Pastel2'), main = 'Relative Item Frequency Plot', type = "relative", ylab = "Item Frequency (Relative)")
```

Products bought with the highest frequencies of transactions are listed below



2.2 Results and Insights from Apriroi Approach

The dataset utilized in this analysis is derived from Hunter's e-grocery, comprising a total of 2,019,501 transactions. These transactions involve various items purchased together from the store. Employing the Apriori algorithm with a minimum support threshold of 0.01 and a minimum confidence of 0.2, we identified robust associations based on lift. The initial five transactions are presented below, accompanied by a bar plot illustrating the top 20 items with the highest relative item frequency. After this analysis, several noteworthy association rules have been discerned:

- If **dry pasta** and **packaged vegetables and fruits** are bought, then **pasta sauce** is also bought.
- If **packaged vegetables and fruits** and **pasta sauce** are bought, then **dry pasta** is also bought.
- If **fresh fruits, fresh vegetables** and **pasta sauce** are bought, then **dry pasta** is also bought.

- If **fresh vegetables** and **pasta sauce** are bought, then **dry pasta** is also bought.
- If **fresh fruits** and **pasta sauce** are bought, then **dry pasta** is also bought.

3. Clustering: K-means Method

3.1 Procedures of K-means Method

- **Step 1: Loading data**

```
# upload the dataset
df=pd.read_csv('data_mining_prj.csv')
df.head()
```

- **Step 2: Data transformation**

In the data preparation phase, several steps were taken to ensure the dataset is ready for analysis:

Handling Missing Values: The days_since_prior_order column had missing values, which were filled with a value of -1 for simplicity.

```
# Detecting null values and sort them in descending order
df.isnull().sum().sort_values(ascending=False)
```

```
days_since_prior_order    64721
order_id                      0
user_id                       0
order_number                  0
order_dow                     0
order_hour_of_day             0
product_id                    0
add_to_cart_order             0
reordered                     0
department_id                 0
department                    0
product_name                  0
dtype: int64
```

```
# fill NAN values in "days_since_prior_order" column with -1
df['days_since_prior_order'] = df['days_since_prior_order'].fillna(-1)
```

Outlier Removal: Outliers in the order_number and add_to_cart_order columns were identified and removed to enhance the robustness of subsequent analyses.

```
# remove outliers of order_number
Q1, Q3 = np.percentile(df_clean['order_number'], [25, 75])

IQR= Q3 - Q1
lower_threshold = Q1 - 1.5 * IQR
upper_threshold = Q3 + 1.5 * IQR

outliers1 = (df_clean['order_number'] < lower_threshold)
outliers2 = (df_clean['order_number'] > upper_threshold)

print('number of small outlier: ',outliers1.value_counts())
print('number of big outliers: ', outliers2.value_counts())
```

```
number of small outlier:  False    1048575
Name: order_number, dtype: int64
number of big outliers:  False    990996
True      57579
Name: order_number, dtype: int64
```

```
# remove outliers of add_to_cart_order
Q1, Q3 = np.percentile(df_clean['add_to_cart_order'], [25, 75])

IQR= Q3 - Q1
lower_threshold = Q1 - 1.5 * IQR
upper_threshold = Q3 + 1.5 * IQR

outliers3 = (df_clean['add_to_cart_order'] < lower_threshold)
outliers4 = (df_clean['add_to_cart_order'] > upper_threshold)

print('number of small outlier: ',outliers3.value_counts())
print('number of big outliers: ', outliers4.value_counts())
```

```
number of small outlier:  False    990996
Name: add_to_cart_order, dtype: int64
number of big outliers:  False    960446
True      30550
Name: add_to_cart_order, dtype: int64
```

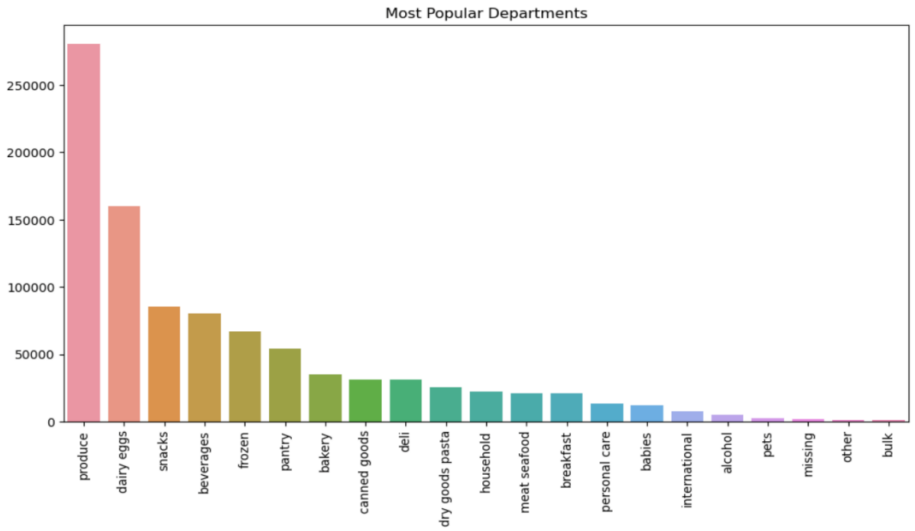
```
df_clean = df_clean[~(outliers3 | outliers4)]
```

Data Types Adjustment: Some columns' data types were adjusted for better representation, particularly converting numeric columns like order_id and user_id to the category and reordered as Boolean.

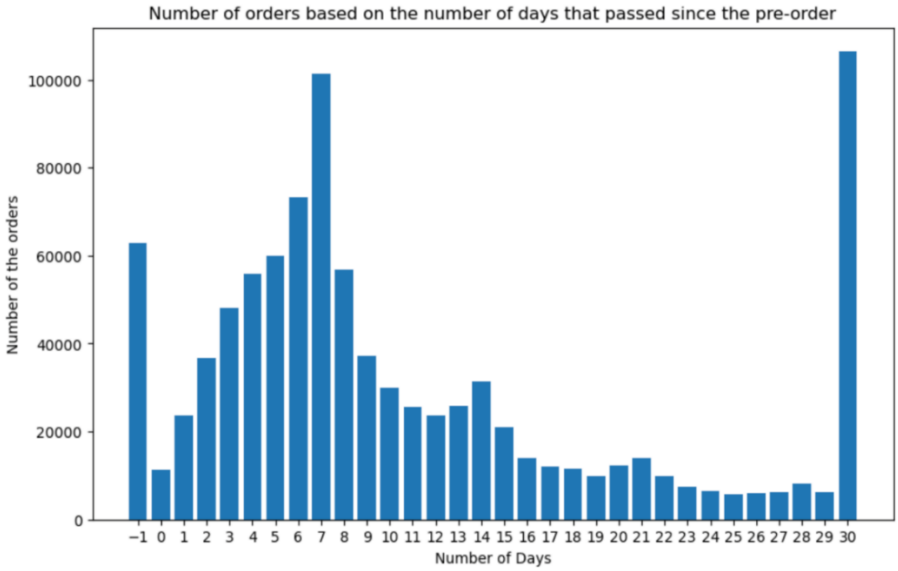
```
#Changing dtype from numeric to category
df_clean.loc[:, 'department_id'] = df_clean['department_id'].astype('category')
df_clean.loc[:, 'product_id'] = df_clean['product_id'].astype('category')
df_clean.loc[:, 'order_id'] = df_clean['order_id'].astype('category')
df_clean.loc[:, 'user_id'] = df_clean['user_id'].astype('category')
df_clean.loc[:, 'reordered'] = df_clean['reordered'].astype('bool')
df_clean.loc[:, 'days_since_prior_order'] = df_clean['days_since_prior_order'].astype('int64')
```

- **Step 3: Applying K-means Model:**

Exploratory Data Analysis: Basic visualizations were performed to explore the distribution of orders across days of the week, hours of the day, and the popularity of product departments.



We can see that the high range of people placing theirs orders are between 10 and 17 hrs.



Cluster Analysis with K-Means:

The optimal number of clusters (k=6) was determined using the Elbow method and Silhouette Score. The data was then scaled, and K-Means clustering was applied.

```
# scaling the data with standardizing the features
scalar = StandardScaler()
scaled_data = scalar.fit_transform(df_clean.iloc[:, :-4])
```

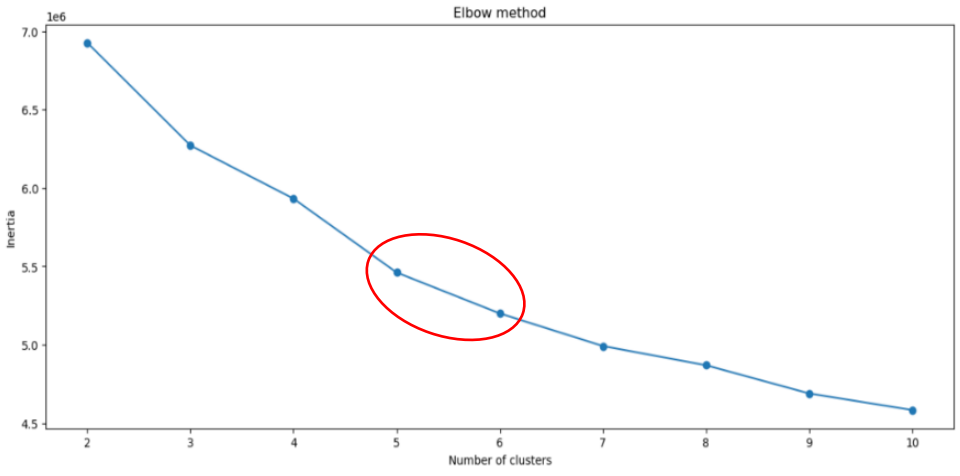
```
# finding the optimal number of clusters in k-means

inertia = []
silhouette = []

for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, n_init='auto')
    kmeans.fit(scaled_data)
    k_in = kmeans.inertia_
    print('check with {} clusters | Inertia : {}'.format(k, k_in))
    inertia.append(k_in)
    labels = kmeans.fit_predict(scaled_data)
    silhouette.append(silhouette_score(scaled_data, labels, sample_size=10000))
```

```
check with 2 clusters | Inertia : 6881806.791591982
check with 3 clusters | Inertia : 6271348.566531798
check with 4 clusters | Inertia : 5890372.378027501
check with 5 clusters | Inertia : 5454104.4279096965
check with 6 clusters | Inertia : 5290650.713679924
check with 7 clusters | Inertia : 4989892.836522333
check with 8 clusters | Inertia : 4830368.097422329
check with 9 clusters | Inertia : 4723269.450268554
check with 10 clusters | Inertia : 4565994.833204016
```

```
# ploting the Elbow method
plt.figure(figsize=(15,6))
plt.plot(range(2, 11), inertia, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.title('Elbow method')
plt.show()
```

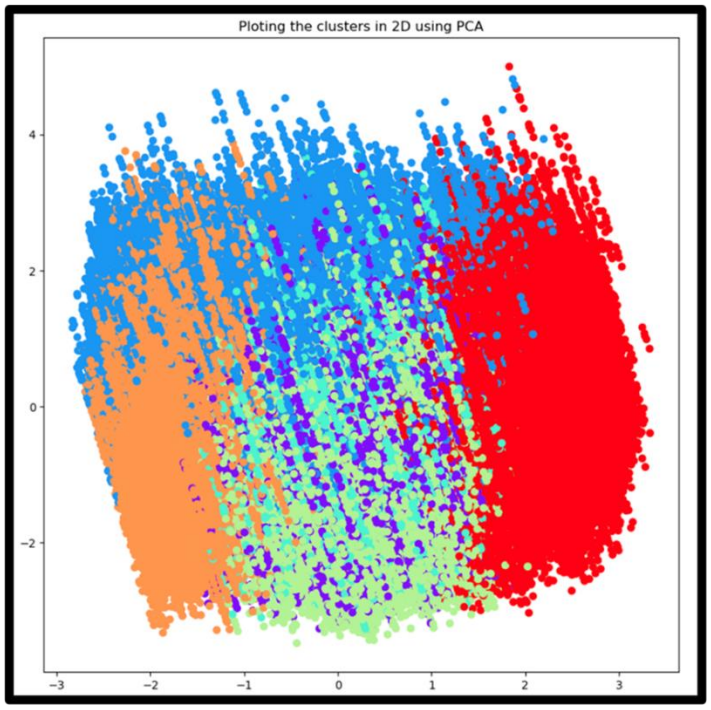


```
# Plotting the silhouette score
plt.plot(range(2, 11), silhouette, marker='o')
plt.xlabel('k')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score')
plt.show()
```

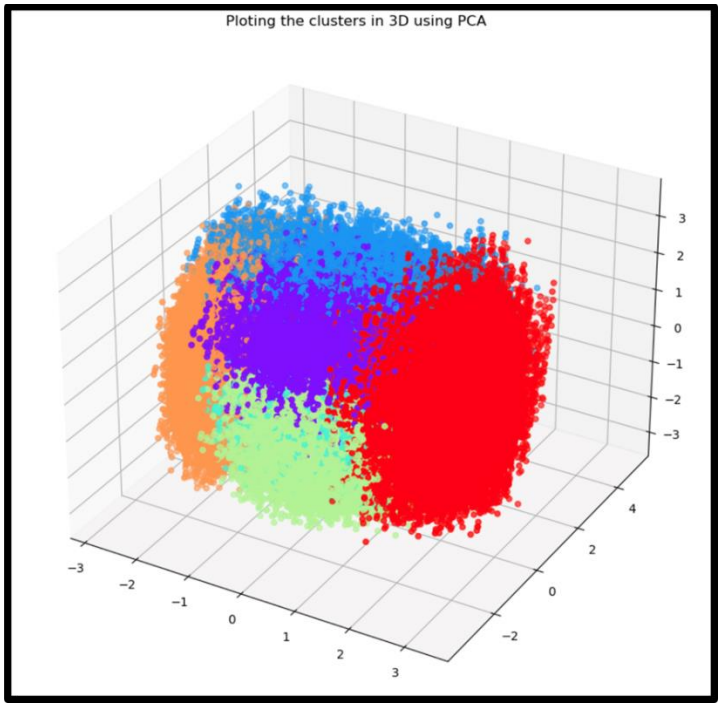

```
# Chossing k = 6 as the optimal number of clusters
kmeans = KMeans(n_clusters=6, n_init='auto')
kmeans.fit(scaled_data)
kmeans_labels = kmeans.fit_predict(scaled_data)
```

Clusters were visualized in both 2D and 3D using PCA.

```
# Ploting the clusters in 2D using PCA
pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)
```



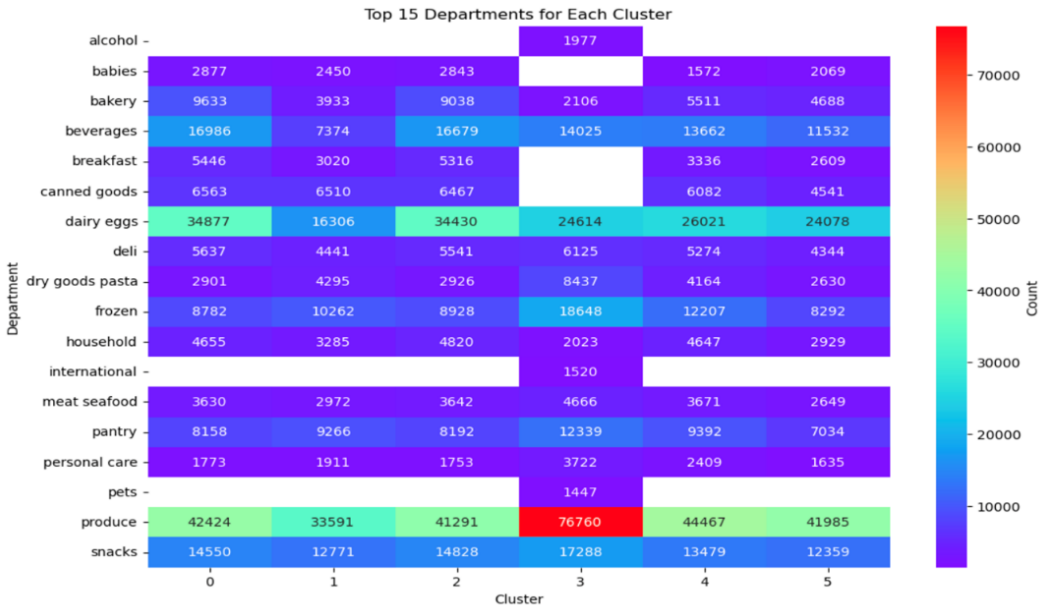
```
# Ploting the clusters in 3D using PCA
pca = PCA(n_components=3, random_state=0)
pca_data = pca.fit_transform(scaled_data)
```




```
# 'cluster' and 'product_name' column names
cluster_product_counts = df_clean.groupby('department')['cluster'].value_counts().unstack(fill_value=0)

# Get the top 15 products for each cluster
top_products_by_cluster = cluster_product_counts.apply(lambda x: x.sort_values(ascending=False).head(20))

# Plotting a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(top_products_by_cluster, cmap='rainbow', annot=True, fmt='.0f', cbar_kws={'label': 'Count'})
plt.title('Top 15 Products for Each Cluster')
plt.xlabel('Product')
plt.ylabel('Cluster')
plt.show()
```



3.2 Results and Insights from K-means Method

Exploratory Data Analysis: The visualizations revealed insights into order patterns, showing peak order times during certain hours and days. The most popular departments were identified, providing a snapshot of customer preferences.

K-Means Clustering: The dataset was successfully clustered into six groups using K-Means. The clusters were visualized in 2D and 3D, providing a clear representation of the grouping patterns.

Top store departments by Cluster: The report includes a heat map showcasing the top 15 store departments for each cluster, helping to identify product preferences within each cluster.

Conclusion: The application of K-Means clustering revealed distinct patterns in customer behaviour, allowing for targeted strategies based on cluster preferences. The pre-processing steps ensured a clean dataset for meaningful analysis. The insights gained from the exploratory analysis and clustering can guide strategic decisions to enhance customer satisfaction, also, optimize marketing and sales strategies.

4. Recommendations

4.1 Recommendations based on Apriori Approach

Based on the findings and insights from Apriori Approach, the following recommendations are proposed to the business for its optimal value proposition and profitability:

a. Product Recommendations: Utilize Apriori analysis insights for personalized product suggestions. On the web page of pasta sauce, fresh fruits and vegetables and packaged fruits and vegetables the ‘other recommended items’ section at the bottom of web page shall be displayed with ‘dry pasta’.
b. Implement intelligent product reminder systems based on frequent itemsets to align with customer purchasing patterns while increasing potential sales. Before navigating the customer to the ‘checkout’ page, if the combination of items in the shopping cart such as dry pasta and packaged vegetables and fruits; packaged vegetables and fruits and pasta sauce; fresh fruits, fresh vegetables and pasta sauce; fresh vegetables and pasta sauce or fresh fruits and pasta sauce are detected, then a popped-up reminder ‘Did you miss this items?’ shall display with the item dry pasta.
c. When considering promotional campaigns such as ‘buy this to get a free product’, dry pasta can be considered as a free gift of low value and of high association with other products such as pasta sauce and fruits and vegetables, minimizing the cost-efficiency of the campaign.
d. Ensure popular items are well-stocked, reducing stock out and enhancing overall customer satisfaction.

4.2 Recommendations based on K-means Method.

a. Promotional Strategies: Tailor promotions and discounts based on K-Means cluster analysis. Design targeted campaigns for specific customer groups to increase engagement and loyalty.
b. Supply Chain Optimization: Optimize inventory management with top products from each cluster.
c. Tailor Ads to our customers: Use K-Means clustering to understand what your customers like. Create ads that match each group's interests, making your ads more effective.
d. Customer Communication: Establish communication strategies based on insights from both analyses. Keep customers informed about new arrivals, promotions, and updates aligned with their cluster preferences.

Integrating these recommendations will help Hunter's e-grocery enhance the current customer experience and adapt swiftly to changing market conditions, ensuring sustained satisfaction and loyalty.

5. References

Rupesh Kumar (2023). *Supermarket dataset for predictive marketing 2023.* Kaggle. https://www.kaggle.com/datasets/hunter0007/ecommerce-dataset-for-predictive-marketing-2023