# SuperMarket Clustering For Predictive Marketing

Mohammad Rouintan

Computer Science, Shahid Beheshti University

**Abstract**

*The supermarket dataset for predictive marketing 2023 consists of over 2 million purchase records at a renowned Hunter's supermarket. Hunter's Supermarket is a well-known French brand based in Brittany that offers a new generation of e-grocery and lifestyle products. With a presence in ten countries, Hunter's e-grocery continuously seeks new ways to anticipate and meet customer demands, building strong loyalty among its customers. However, recent black swan events such as Covid-19, the Ukraine crisis, and gas shortages have impacted purchasing behavior, making it necessary to develop a business value proposition to increase customer lifetime value. We perform various analyzes on this dataset and run different clustering models on it so that we can compare the models and reach the best results.*

## Introduction

Our task is to use clustering algorithms to segment the customers into distinct groups based on their shopping behavior and demographics. We are going to work with the supermarket dataset. We will implement multiple clustering models using the Scikit-Learn package to cluster data points, based on about 12 features. This dataset has a very large number of samples (about 2 million samples), which may lead to runtime problems and hardware problems to run different algorithms. Therefore, we have to use different ways to implement different models, which we will discuss further.

As we said, this dataset has a large number of samples, so to run some models, we need to sample from the dataset. In this project, this sampling is about 50,000 data or less, which was done in the form of random sampling.

In this project, we use different ways to evaluate the performance of the clustering model, for example:

- ***Elbow Method*** (Inertia)
- ***Silhouette score***
- ***t-SNE***

The following methods are used to reduce dimensions, feature engineering and visualize the clusters to analyze their characteristics:

- ***Trimming Outlier***
- ***Scaling Methods***
- ***PCA***
- ***Kernel PCA***

The clustering models that we have implemented in this project are:

- ***K-Means***
- ***DBSCAN***
- ***SpectralClustering***
- ***GaussianMixture***

## Methodologies

### Exploratory Data Analysis

first, for data analysis we should read description of dataset and know features of dataset in detail. This dataset contains:

- ***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)

According to the columns of the dataset, we understand that this dataset has 10 numerical features and the rest of the features are categorical. After checking the dataset to find missing values, we see that the ***days_since_prior_order*** has 124,342 missing values. Therefore, we must either fill them with a special method or delete them. We used computer iterative algorithm to fill these missing values. If we want to explain this algorithm, The ***IterativeImputer*** is an imputation algorithm in scikit-learn, a Python machine learning library. It is used to fill in missing values in a dataset by predicting them from the other observed values. In particular, the Iterative Imputer uses a machine learning model to impute missing values iteratively. It starts by filling in missing values with initial estimates, such as the mean or me-

dian of the observed values. Then it uses a machine learning model, such as a decision tree or a random forest, to predict the missing values based on the other observed values in the same row. This process is repeated multiple times, with the imputed values from each iteration being used as the input for the next iteration, until convergence is achieved. The Iterative Imputer algorithm can handle missing values in both numeric and categorical data and simultaneously handle missing values in multiple columns. It is a useful tool for data preprocessing and can improve the performance of machine-learning models by reducing the impact of missing data.

***order_id and user_id:*** The order_id has 200,000 unique values, that is, the order_id of 2 million data is duplicated from these 200,000. Also, user_id has 105273 unique values. This means that all these 2 million orders were made by 105,273 people.

***order_number:*** According to the plot, we can see that the order number is skewed to the left and almost 50% of the order numbers are between 1 and 11 (Figure 1).

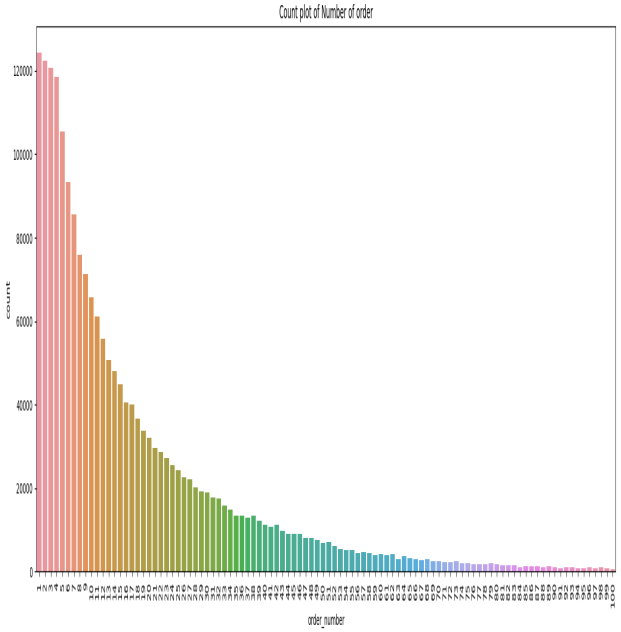


**Figure 1:** *Count Plot of Order Number*

***order_dow:*** According to the count plot and pie plot, we see that the top 4 for order number are the first two days of the week and the two days of the weekend, that is, Monday, Tuesday and Saturday, Sunday (Figure 2). According to the heatmap, We see that most of the orders were on the first and last days of week between 10:00 and 16:00 (Figure 3).

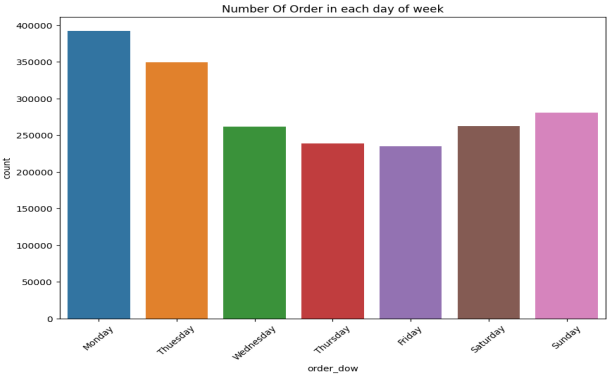


**Figure 2:** *Number of Order in each Day of week*

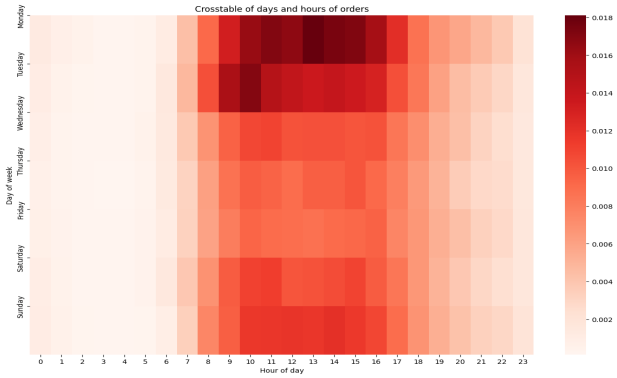


**Figure 3:** *Heatmap of day, hour and order_number*

***order_hour_of_day:*** According to the Count plot and Box plot, We see that most of the orders were between 9 and 17 hours (Figure 4 & 5).

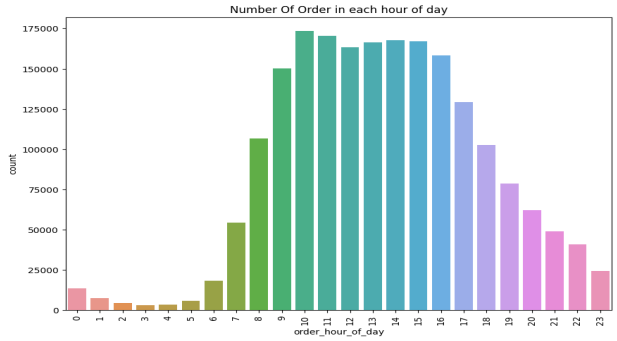


**Figure 4:** *Numebr of Order in each hour of day*

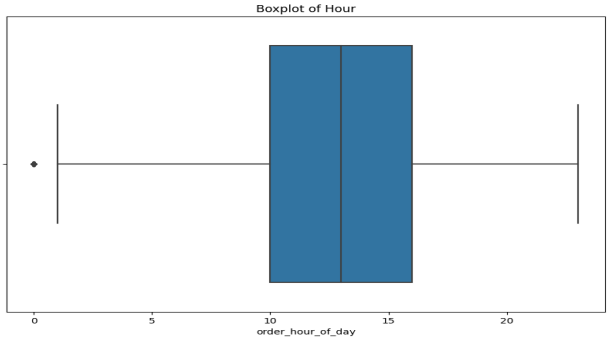


**Figure 5:** *Boxplot of hour*

***add_to_cart_order:*** According to the plot and its

description, we can see that in almost 75% of the orders, the number of items in the shopping cart was between 1 and 11 (Figure 6).

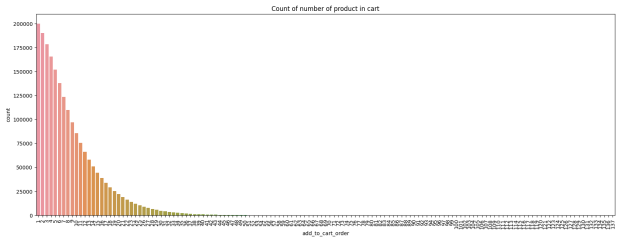


**Figure 6:** *Count of add to cart order*

***reordered:*** Almost 59% of purchases have been re-ordered (Figure 7).

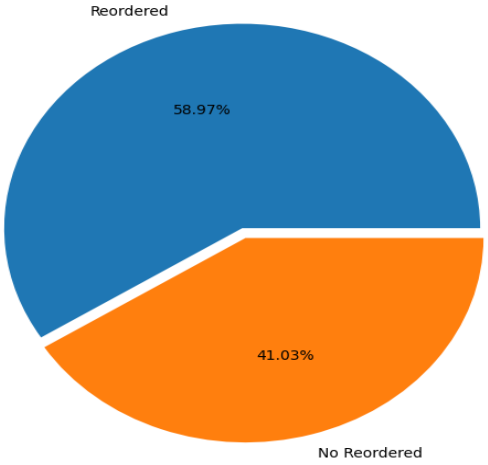


**Figure 7:** *Pieplot of reordered purchases*

***department:***

Top 2 Department is (Figure 8):

- produce
- dairy eggs

***product_name:***

Top 3 Product (Figure 9):

- fresh fruits
- fresh vegetables
- packaged vegetables fruits

Bottom 3 Product (Figure 10):

- frozen juice
- beauty
- baby accessories

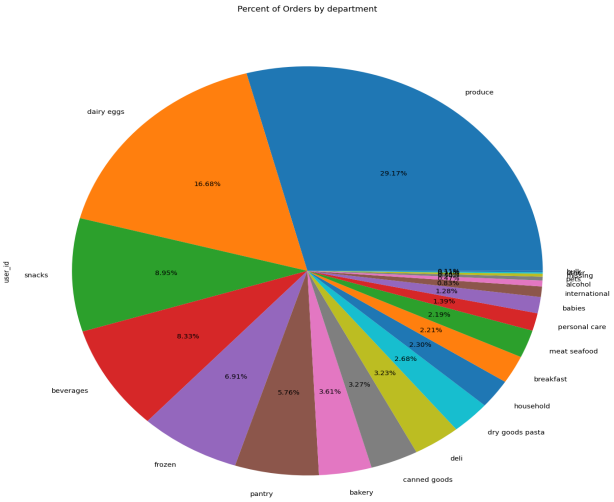


**Figure 8:** *Pieplot of departments*

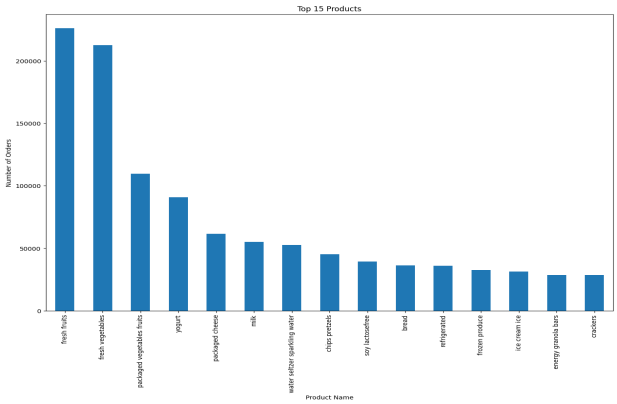


**Figure 9:** *Barplot of Top 15 product*

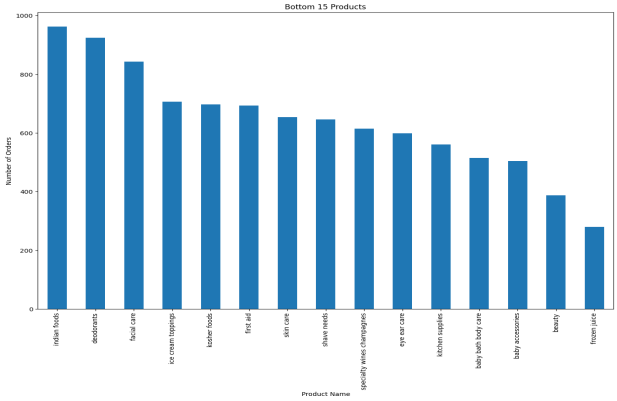


**Figure 10:** *Barplot of Bottom 15 product*

***Other Analysis:*** Almost 80% of the orders were in the morning and afternoon. Top 3 range of order number:

- 1-10 , 48.67%
- 11-20 , 21.96%
- 21-30 , 11.79%

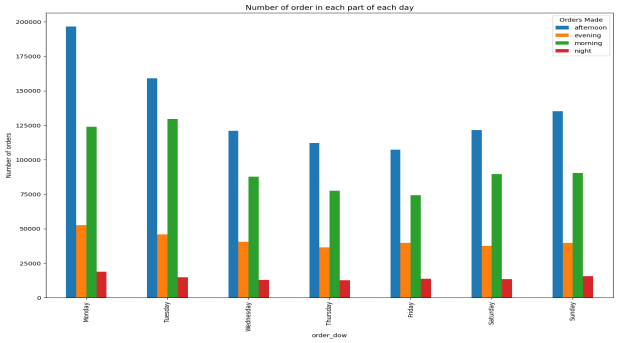


**Figure 11:** *Barplot of Number of order in each day and each part of day*

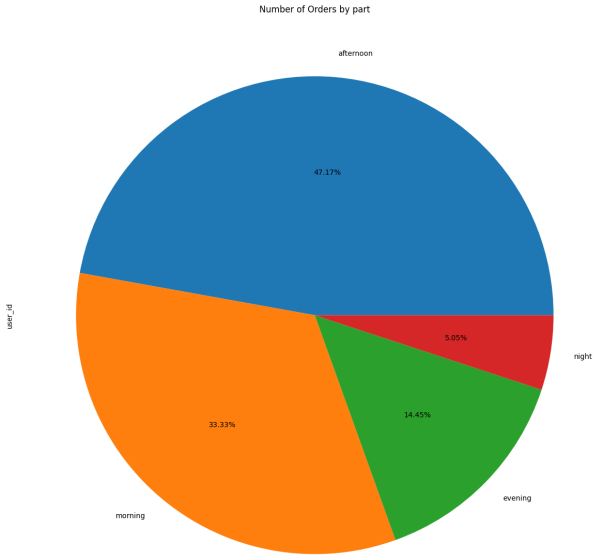


**Figure 12:** *Pieplot of each part of day*

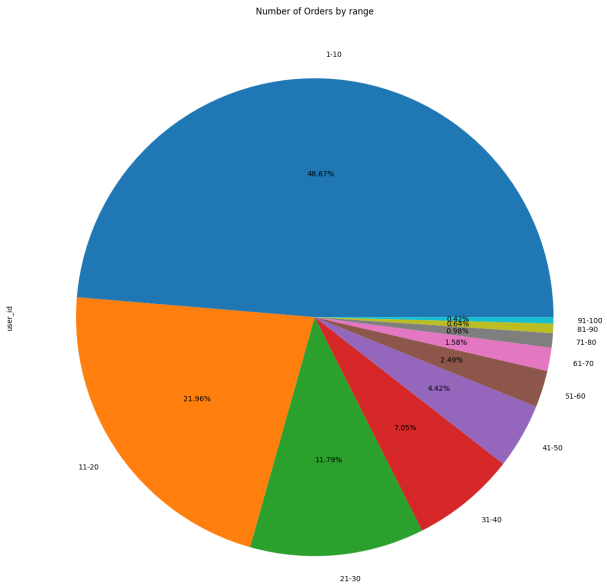


**Figure 13:** *Pieplot of order number range*

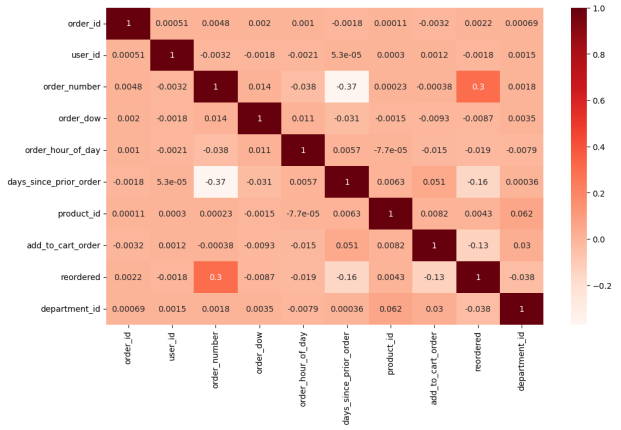


**Figure 14:** *Correlation Matrix*

## Preprocessing Data

In data preprocessing, we first remove the two categorical features because their equivalent IDs are in the table. Second, we use a method for trimming outlier that trim samples that have number less than 1st percentile of current column or more than 99th percentile of current column.

In the next step, we take 50,000 random samples and train the models on them. This sampling may be changed in the Training Models section according to the time complexity and space complexity of the model.

## Training Models

In this section, we check the best result of each model and the rest of the results are checked in the notebook.

**K-Means:** In this section, we run the K-Means model on the sampled dataset that contains 50,000 samples without performing any scaling method and observe the results. Here we consider the number of clusters as 4.

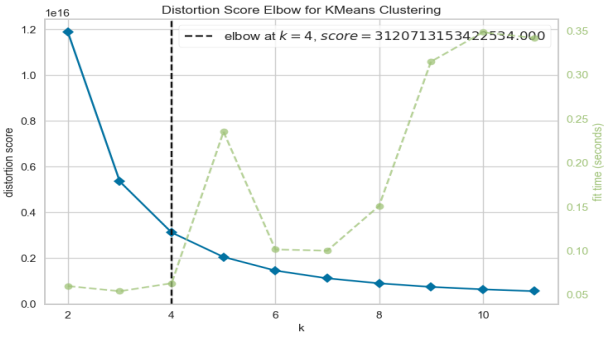


**Figure 15:** *Elbow Plot*

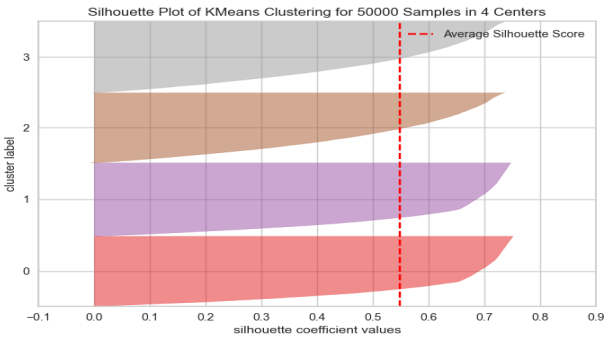


**Figure 16:** *Silhoutte Plot , silhoutte score = 0.55*

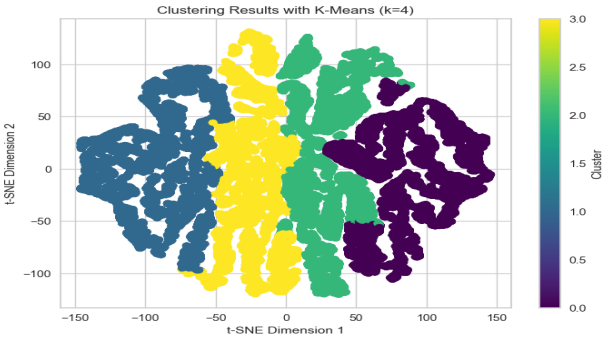


**Figure 17:** *t-SNE with 2 components of Kmeans*

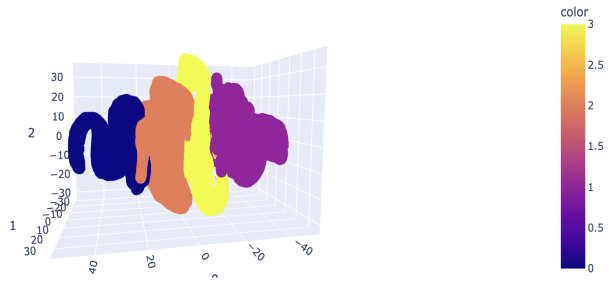


**Figure 18:** *t-SNE with 3 components of Kmeans*

***Using PCA:*** In this section, we run the K-Means model on the sampled dataset that contains 50,000 samples with applied PCA (2 components) on this and observe the results. Here we consider the number of clusters as 3.

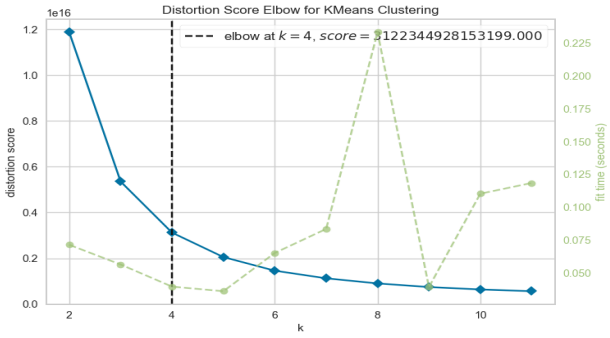


**Figure 19:** *Elbow Plot*

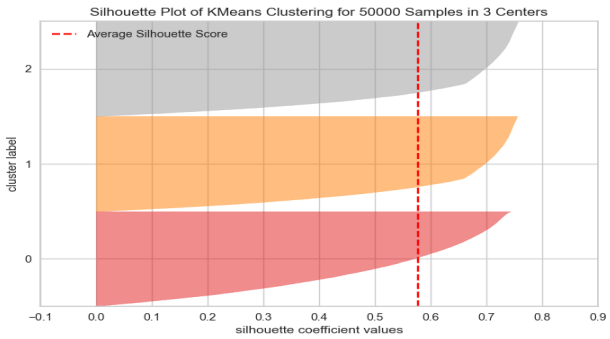


**Figure 20:** *Silhoutte Plot, silhoutte score = 0.58*

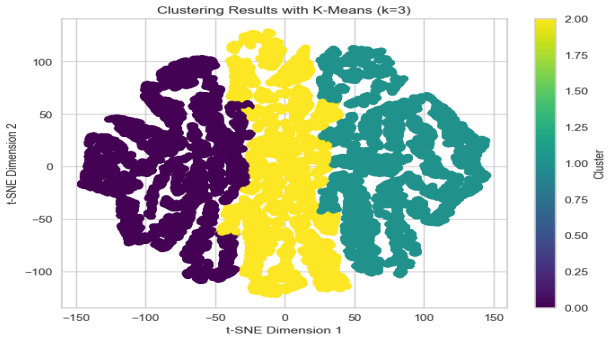


**Figure 21:** *t-SNE with 2 components of Kmeans*

***Using KernelPCA:*** In this part of the time complexity of the KernelPCA, we take a 20,000 sample from the dataset and first apply the MinMaxScaler method on it, then change its dimensions using the KernelPCA (2 components, poly kernel, 3 degree), and finally run K-Means on it. Here we consider the number of clusters as 2.

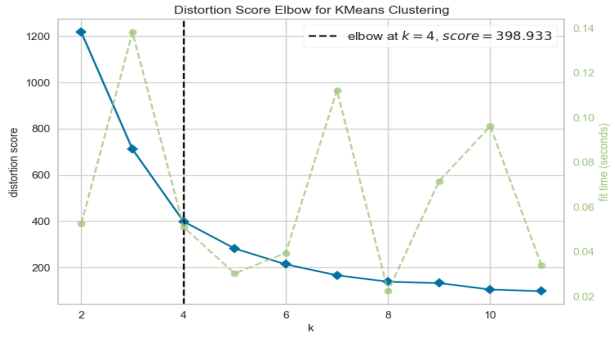


**Figure 22:** *Elbow Plot*

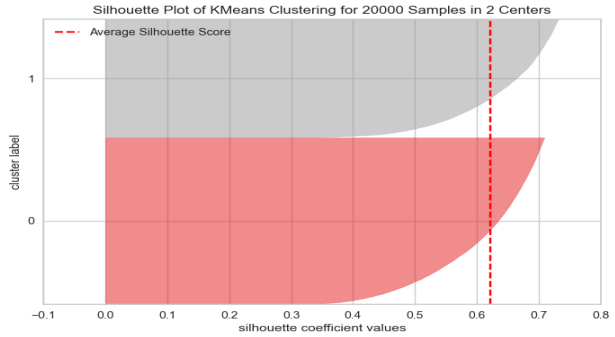


**Figure 23:** *Silhoutte Plot, silhoutte score = 0.63*

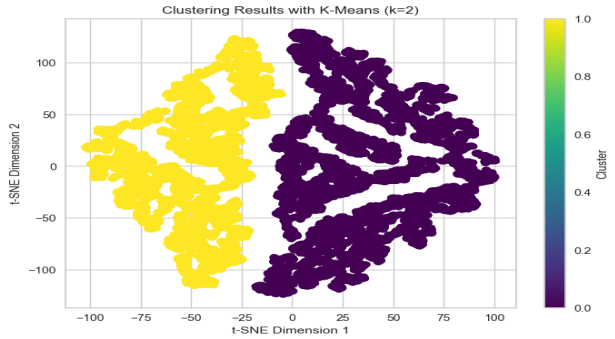


**Figure 24:** *t-SNE with 2 components of Kmeans*

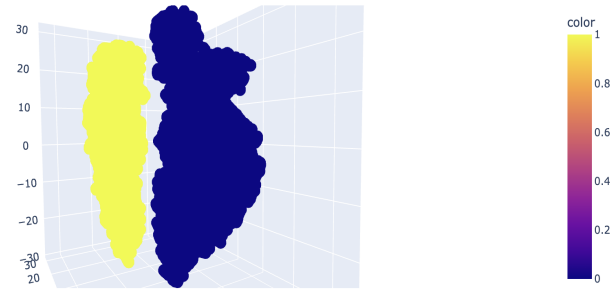


**Figure 25:** *t-SNE with 3 components of Kmeans : KernelPCA (3 components, poly kernel, 7 degree), silhoutte score = 0.45*

**DBSCAN:** ***Using PCA:*** In this section, we first applied the standard scaler method on the dataset, then reduced it to 2 dimensions with the PCA algorithm. The additional work we have done here is that we have used the nearest neighbor method so that we can approximately find a suitable value for the epsilon parameter. The number of clusters obtained in this section is 3, one of which is considered an outlier cluster (cluster = -1).

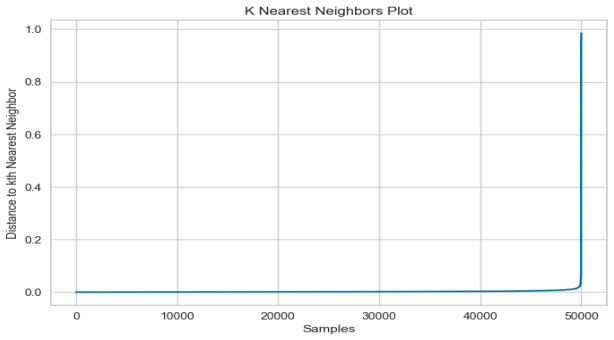


**Figure 26:** *K Nearest Neighbors Plot*

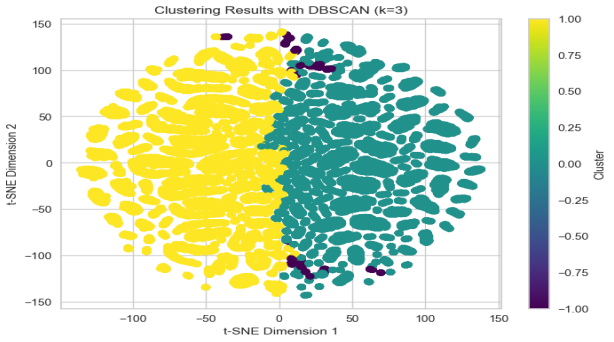


**Figure 27:** *t-SNE with 2 components of DBSCAN, silhoutte score = 0.54*

***Using KernelPCA:*** In this part of the time complexity of the KernelPCA, we take a 20,000 sample from the dataset and first apply the MinMaxScaler method on it, then change its dimensions using the KernelPCA (2 components, poly kernel, 3 degree), and finally run DBSCAN on it. Here we consider the number of clusters as 2.

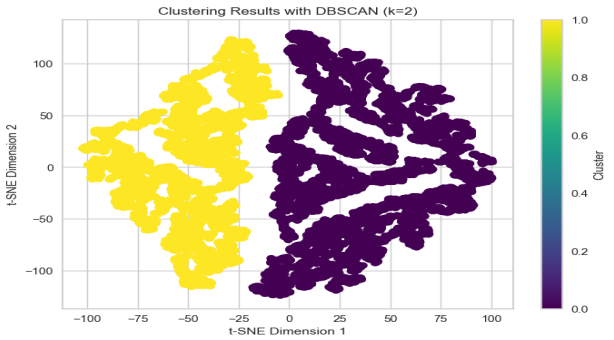


**Figure 28:** *t-SNE with 2 components of DBSCAN, silhoute score = 0.63*

**SpectralClustering:** In this section, due to the time complexity of spectral clustering, we have taken a sample of 20,000 from the dataset and applied the MinMax Scaler method on it. Then we run the spectral clustering model on it and see the results. Here we consider the number of clusters as 2.

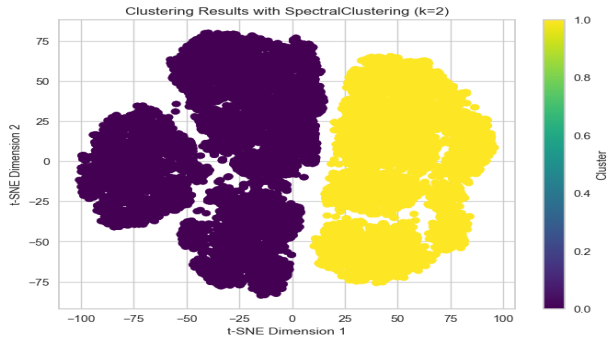


**Figure 29:** *t-SNE with 2 components of SpectralClustering, silhoutte score = 0.26*

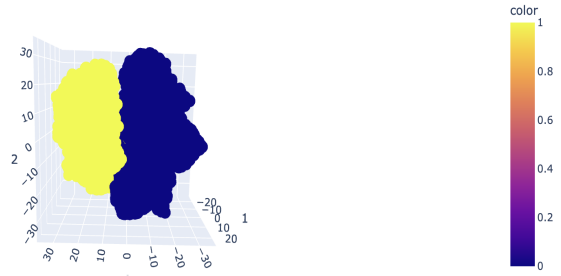


**Figure 30:** *t-SNE with 3 components of SpectralClustering, silhoutte score = 0.26*

***Using PCA:*** In this section, in addition to what we did in the previous section, we reduced it to 2 dimensions using PCA and then ran the algorithm. Here we consider the number of clusters as 2.

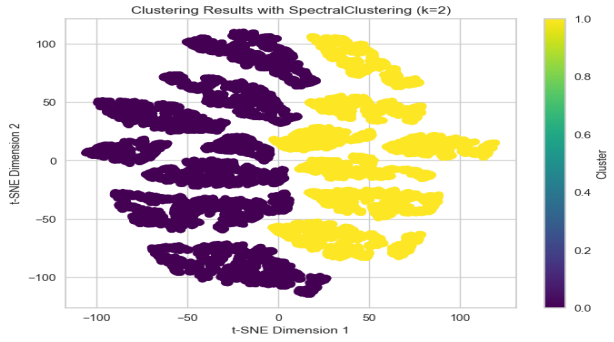


**Figure 31:** *t-SNE with 2 components of SpectralClustering, silhoutte score = 0.62*

***Using KernelPCA:*** In this part of the time complexity of the KernelPCA, we take a 20,000 sample from the dataset and first apply the MinMaxScaler method on it, then change its dimensions using the KernelPCA (2 components, poly kernel, 3 degree), and finally run SpectralClustering on it. Here we consider the number of clusters as 2.

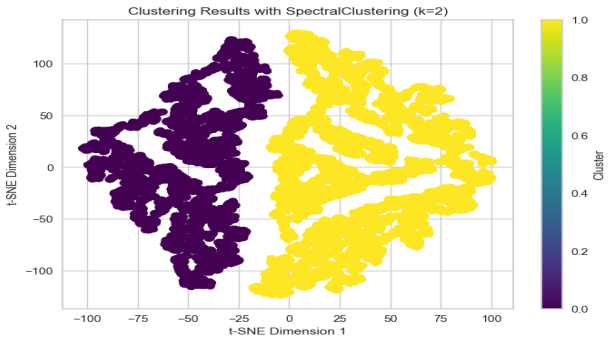


**Figure 32:** *t-SNE with 2 components of SpectralClustering, silhoutte score = 0.63*

**GaussianMixture:** In this section, we run the GaussianMixture model on the sampled dataset that contains 50,000 samples without performing any scaling method and observe the results. Here we consider the number of clusters as 2.

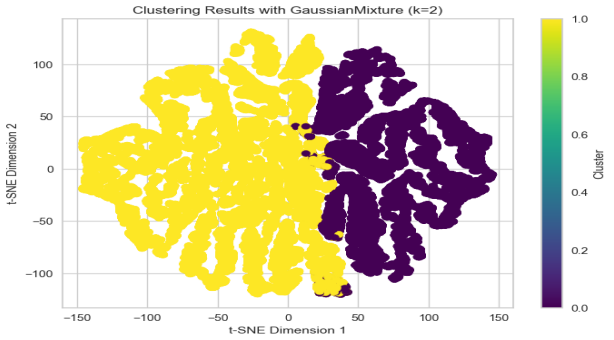


**Figure 33:** *t-SNE with 2 components of GaussianMixture, silhoutte score = 0.58*

In this section, we have taken a sample of 20,000 from the dataset and applied the MinMaxScaler method on it. Then we run the GaussianMixture model on it and see the results. Here we consider the number of clusters as 2.

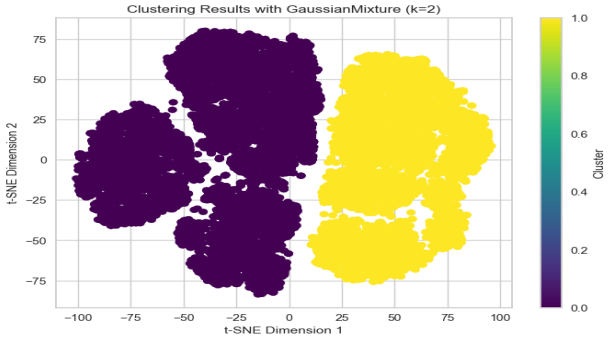


**Figure 34:** *t-SNE with 2 components of GaussianMixture, silhoutte score = 0.26*

***Using KernelPCA:*** In this part of the time complexity of the KernelPCA, we take a 20,000 sample from the dataset and first apply the MinMaxScaler

method on it, then change its dimensions using the KernelPCA (2 components, poly kernel, 3 degree), and finally run GaussianMixture on it. Here we consider the number of clusters as 2.

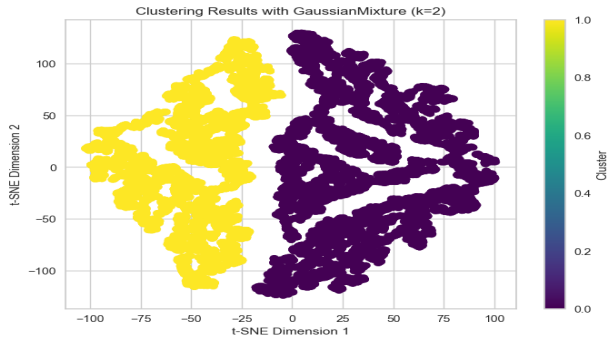


**Figure 35:** *t-SNE with 2 components of GaussianMixture, silhoutte score = 0.63*

## Results

The main purpose of the project is to segment the customers into distinct groups based on their shopping behavior and demographics.

As I can see in the Training Model section, we have used 4 different clustering models. For each model, we used various methods for data preprocessing, data scaling, and dimensionality reduction to be able to separate the clusters well.

For each model, we used various methods for data preprocessing, data scaling, and dimensionality reduction to be able to separate the clusters well. But I see that when we give the data itself to the models without any pre-processing, the clusters are not well separated and we can take all the data as one cluster. But by using Min Max Scaler and Kernel PCA, we were able to separate the cluster to a good extent. When we separated the clusters, all the models used on the dataset performed well. But the point is that according to the more tests we did by increasing the size of the training dataset, Kernel and other methods also cannot separate the data well, and if we consider the entire dataset, maybe We can say that it is placed in a cluster.

We can also look at this issue from another point of view and add other features according to the features of the dataset or create a more suitable dataset from this dataset so that we can perform the clustering process better.

## References

[1] Preprocessing

[2] Visualization

[3] Clustering

[4] Dataset codes in kaggle