Implementation Report for Question 8

By: Katayoun Kobraei

Instructor: Dr. Farahani

Abstract
In this question from the third series of exercises, we will examine the dataset for supermarket marketing predictive analytics. In this exercise, various clustering techniques are used to best categorize the customers in this sector.

- First, similar to previous exercises, we need to preprocess the data to uncover both visible and hidden information. This preprocessing includes handling null and NaN data, encoding categorical data, and engineering relevant features.
- Next, we apply the K-means clustering model to the dataset to find the optimal value of k for categorization. We will use metrics like silhouette score, etc., for this purpose.
- In the following step, we will visualize the various categories using 2D and 3D plots, utilizing the PCA model.
- In subsequent stages, we will compare different algorithms such as DBSCAN with our model, attempt to reduce data dimensions using PCA, and ultimately review all results for a comprehensive insight into the dataset.

Introduction

The dataset we are working on is a predictive marketing dataset for supermarkets. This dataset was introduced in 2023 and contains over 2 million records from Hunter supermarket. We are tasked with categorizing buyers into suitable groups using these records and determining which category each individual belongs to. Since we are dealing with a clustering problem, it is necessary to apply the well-known K-means model. To better identify these categories, we can also make adjustments to the dataset before training the model. We will then train them with newer clustering methods to find the best categorization. We will explore these methods in more detail below.

Part One: Data Preprocessing

First, we examine the overall structure of the dataset, including all feature variables, total sample count, the structure of each feature, and several random samples. For this, we will load the desired file using the `read_csv()` function and store it in a dataframe. The dataset has 22 columns, including order_id, user_id, order_number, order_dow, order_hour_of_day, days_since_prior_order, product_id, add_to_cart_order, reordered, department_id, department, product_name, etc. Using the `info()` function, we can gather information about these features.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2019501 entries, 0 to 2019500
Data columns (total 12 columns):
 #   Column                  Dtype
---  ------                  -----
 0   order_id                int64
 1   user_id                 int64
 2   order_number            int64
 3   order_dow               int64
 4   order_hour_of_day       int64
 5   days_since_prior_order  float64
 6   product_id              int64
 7   add_to_cart_order       int64
 8   reordered               int64
 9   department_id           int64
 10  department              object
 11  product_name            object
dtypes: float64(1), int64(9), object(2)
memory usage: 184.9+ MB
```

According to the information above, we find that only two features, department and product_name, are not numerical and need decoding. We can gather more information for numerical data using the `describe()` function.

	order_id	user_id	order_number	order_dow	order_hour_of_day	days_since
count	2.019501e+06	2.019501e+06	2.019501e+06	2.019501e+06	2.019501e+06	1
mean	1.707013e+06	1.030673e+05	1.715138e+01	2.735367e+00	1.343948e+01	1
std	9.859832e+05	5.949117e+04	1.752576e+01	2.093882e+00	4.241008e+00	8
min	1.000000e+01	2.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00	0
25%	8.526490e+05	5.158400e+04	5.000000e+00	1.000000e+00	1.000000e+01	5
50%	1.705004e+06	1.026900e+05	1.100000e+01	3.000000e+00	1.300000e+01	8
75%	2.559031e+06	1.546000e+05	2.400000e+01	5.000000e+00	1.600000e+01	1
max	3.421080e+06	2.062090e+05	1.000000e+02	6.000000e+00	2.300000e+01	3

To familiarize ourselves with the types of data in each column, we can use the `unique()` function.

```
order_id                  200000
user_id                   105273
order_number                 100
order_dow                      7
order_hour_of_day             24
days_since_prior_order        31
product_id                   134
add_to_cart_order            137
reordered                      2
department_id                 21
department                    21
product_name                 134
dtype: int64
```

Overall, we see that there are 222,222 records. For each feature, there is a specific number of data types we can work with.

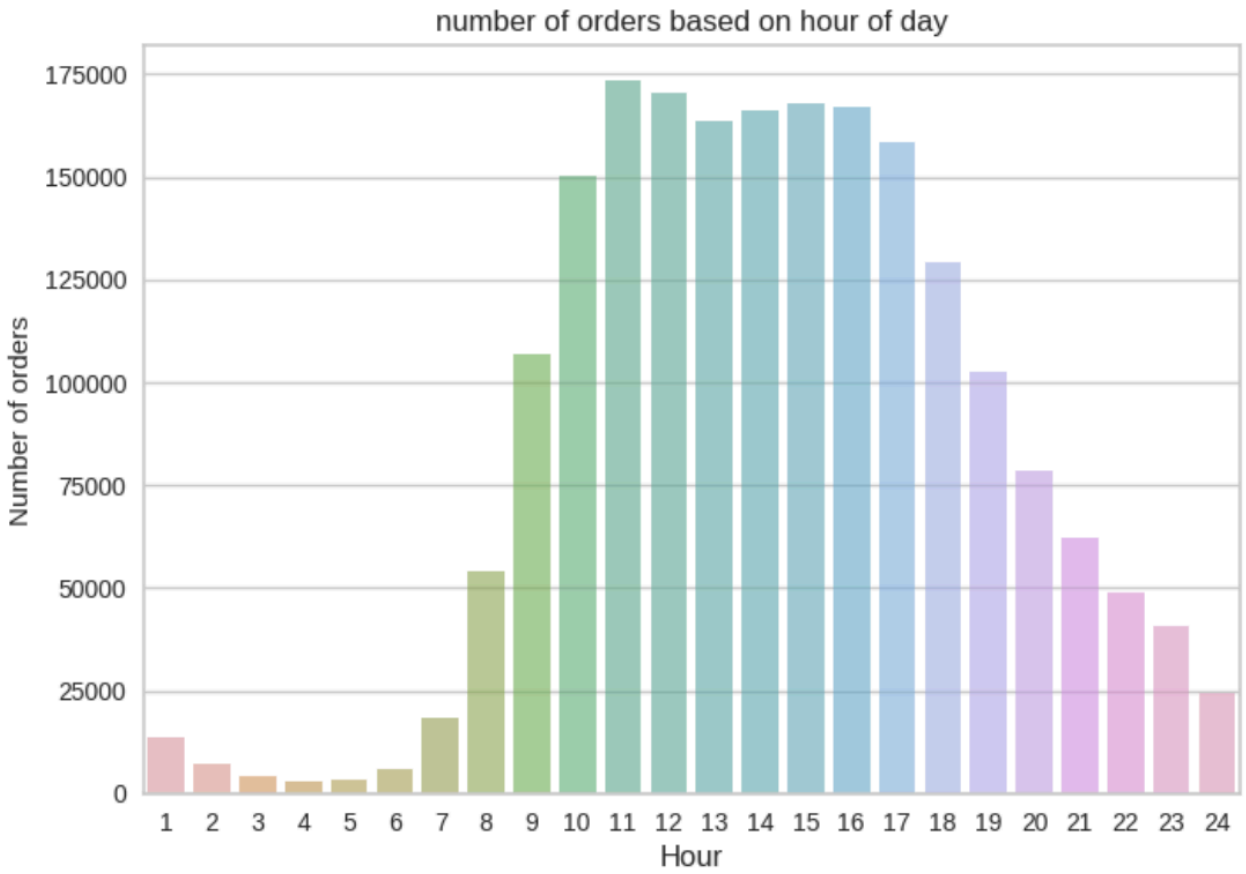
In the next step, we will handle the NaN values in the dataset. There are many ways to handle these values. For example, in the simplest forms, we can assign a specific value to all of them. We first identify the features that have these values using the `isnull()` function, and then call the `sum()` function to obtain the count of these values for each feature.

```
order_id                      0
user_id                       0
order_number                  0
order_dow                     0
order_hour_of_day             0
days_since_prior_order    124342
product_id                    0
add_to_cart_order             0
reordered                     0
department_id                 0
department                    0
product_name                  0
dtype: int64
```

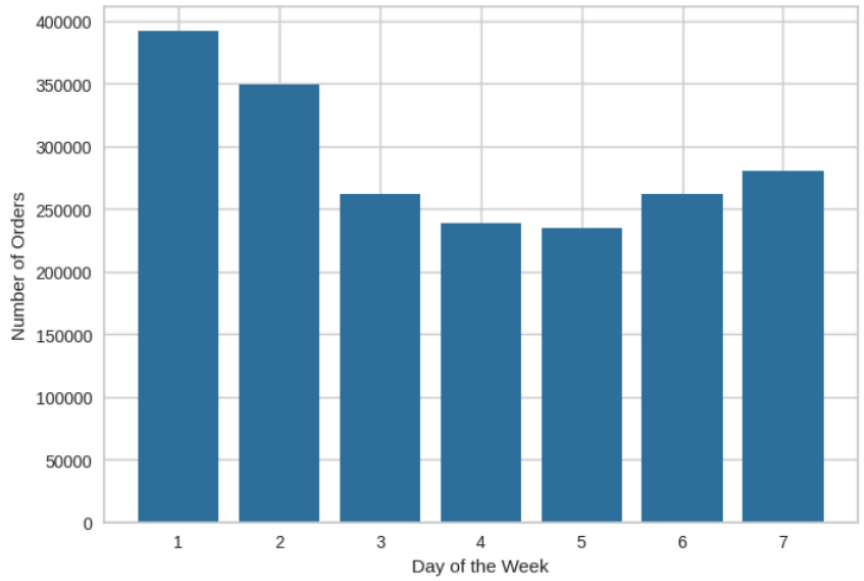
As seen, the feature `days_since_prior_order` has many NaN values. This feature indicates how many days have passed since the customer's last purchase, so each of these NaN values is significant. There are many ways to handle NaN values. The first and simplest way is to assign fixed values to them or the mean of other rows. Since these values range from 2 to 32, we should assign a meaningful value, so we can assign -1 for these values. Better methods include using KNN or regression to predict these NaN values. We used regression for these values. For this, we create an object from the `IterativeImputer` class and select a random forest estimator for it. We then provide a copy of the desired feature to the `fit()` function of this object to handle the values. Finally, we replace the old column with the new one in the main dataframe. Comparing the new column with the previous one, we see that the values have been well predicted:

	Previous Column		New Column with Predicted Values	
			50	6.0
50	6		51	6.0
51	6		52	6.0
52	6		53	6.0
53	6		54	6.0
54	6		55	6.0
55	6		56	6.0
56	6		57	6.0
57	6		58	6.0
58	6		59	6.0
59	6		60	6.0
60	6		61	6.0
61	6		62	6.0
62	6		63	7.0
63	7		64	7.0
64	7		65	7.0
65	7		66	7.0
66	7		67	7.0
67	7		68	7.0
68	7		69	NaN
69	11			

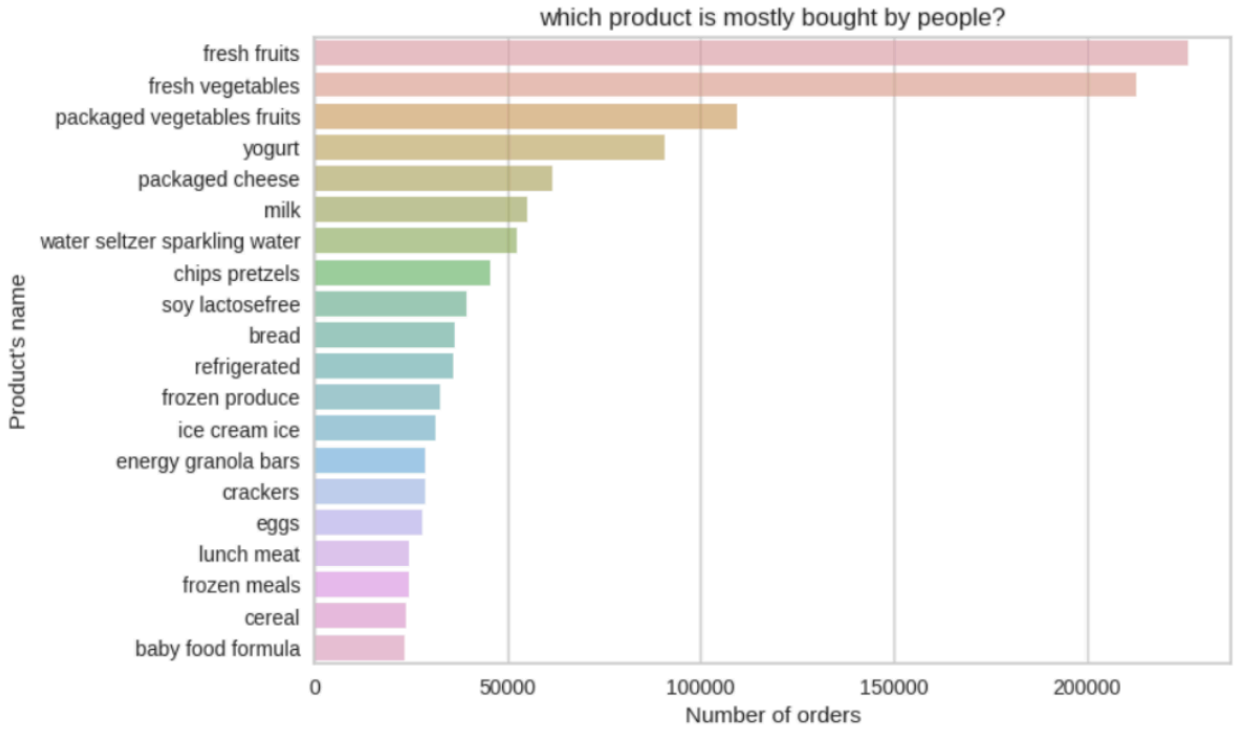
In the next step, we will assess the status of each feature using various charts. First, we will see, based on the meaning of each feature, during which hour of the day the most purchases occur:



From this chart, we observe that the highest amount of purchases happens at 22:00 and overall, from 22:00 to 21:00, the purchase amount is significantly higher. If we want to see on which day of the week people purchase more, we will also find that:
According to the chart, the highest purchases occur in the first two days of the week, and this amount decreases in the following days until the fifth day, then slightly increases on the last two days of the week.



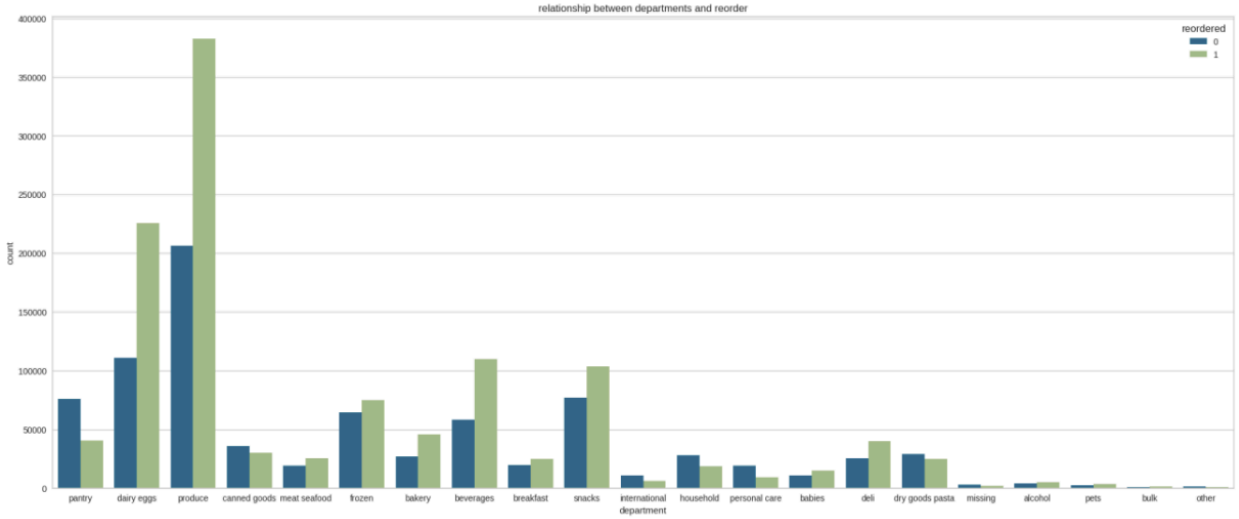
It’s also good to know the most popular products among people:



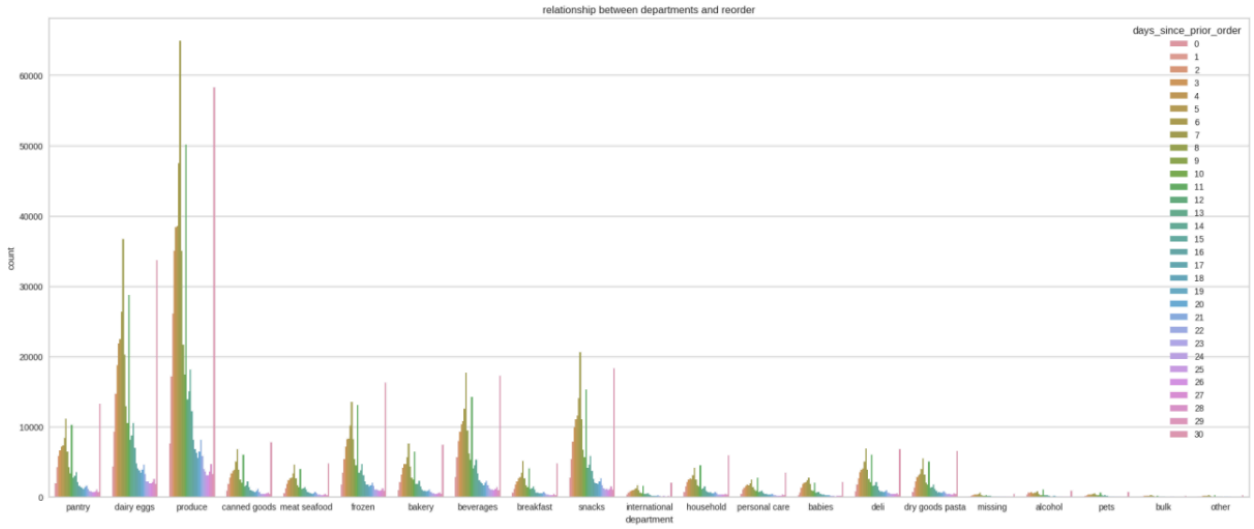
As we see, the most purchased product from this store is fresh fruit, followed by fresh vegetables and packaged fruits. After that, yogurt and packaged cheese come next.
Now, if we check what the best-selling product is in each category, we will have the following values:

department	
alcohol	beers coolers
babies	baby food formula
bakery	bread
beverages	water seltzer sparkling water
breakfast	cereal
bulk	bulk grains rice dried goods
canned goods	soup broth bouillon
dairy eggs	yogurt
deli	lunch meat
dry goods pasta	dry pasta
frozen	frozen produce
household	paper goods
international	asian foods
meat seafood	hot dogs bacon sausage
missing	missing
other	other
pantry	baking ingredients
personal care	oral hygiene
pets	cat food care
produce	fresh fruits
snacks	chips pretzels

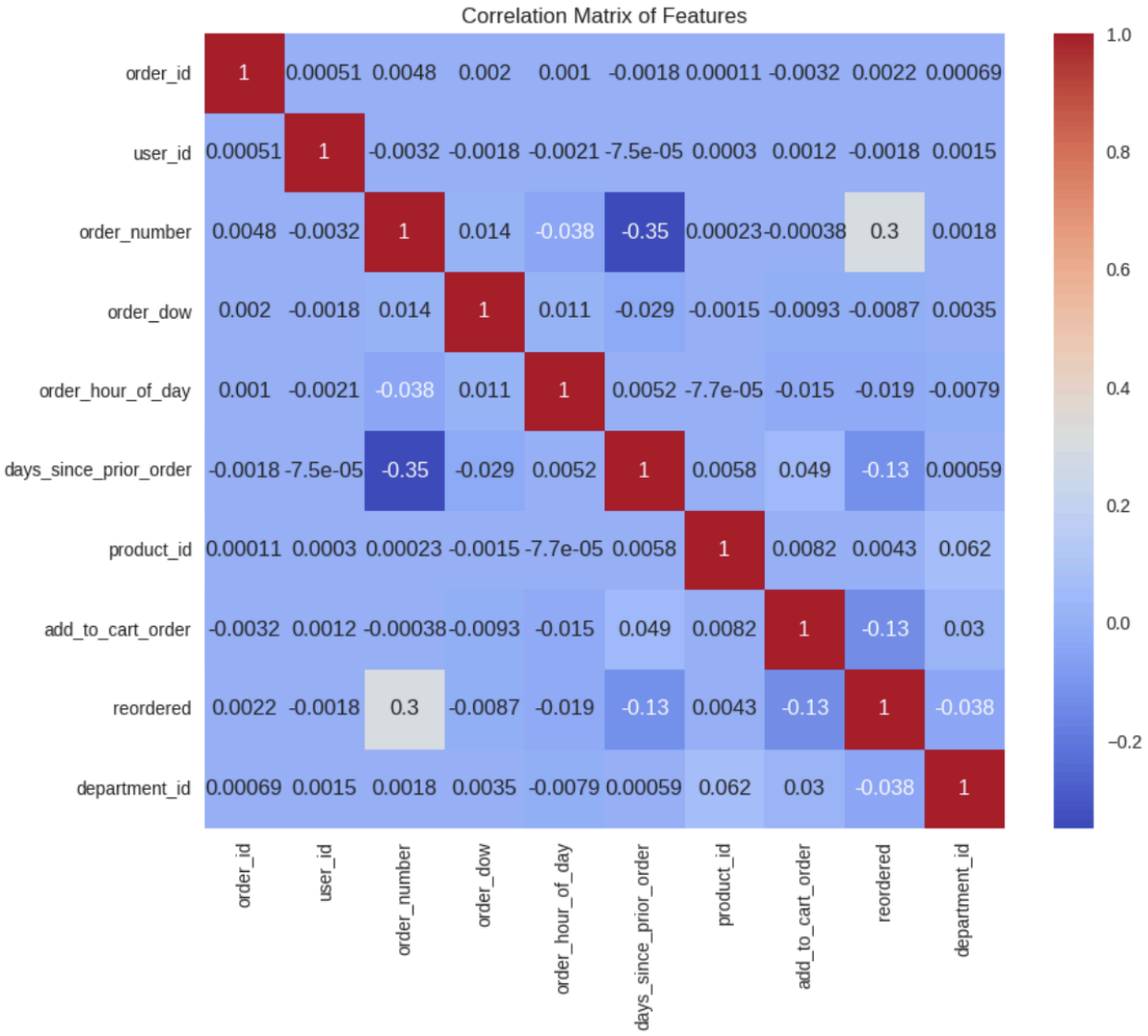
In the next step, we need to explore the explicit and hidden relationships between features. The first thing we examine is whether pre-purchasing a product by someone is related to categories or not.



Another relationship we can investigate is the correlation between the number of days since the last purchase and the category of the product.



Finally, we plot the correlation matrix for each of these features:

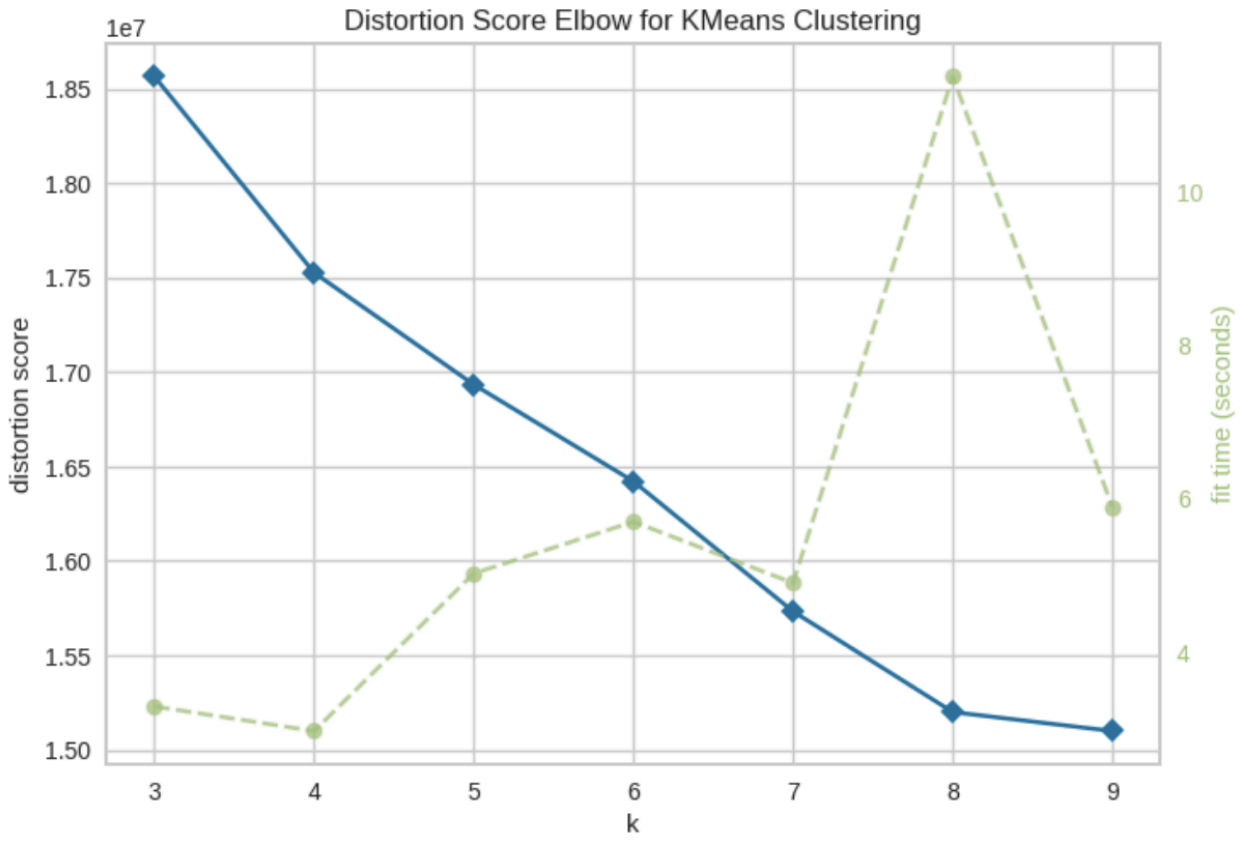


In the next preprocessing step, we need to encode categorical features.

Since the data in each feature has different ranges, it is necessary to scale the data as well. Various scalers can be used, such as MinMaxScaler or StandardScaler; here, we will use StandardScaler. For this, we create an object of the StandardScaler class. Then, we call this object on the dataframe using the `transform_fit()` function to scale all values in the dataframe.

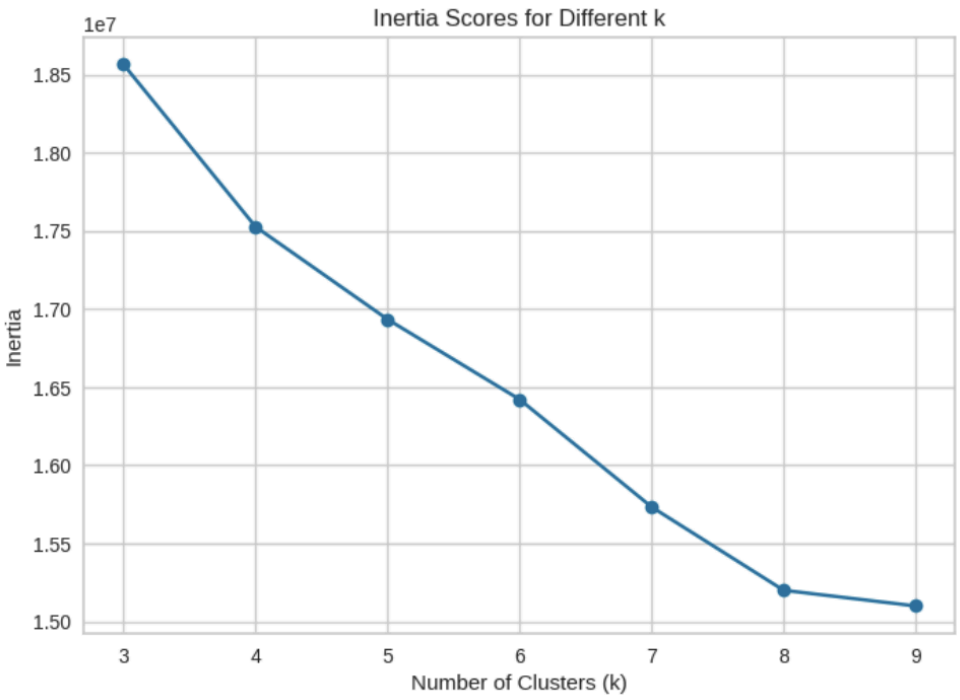
Part Two: Implementing KNN Model

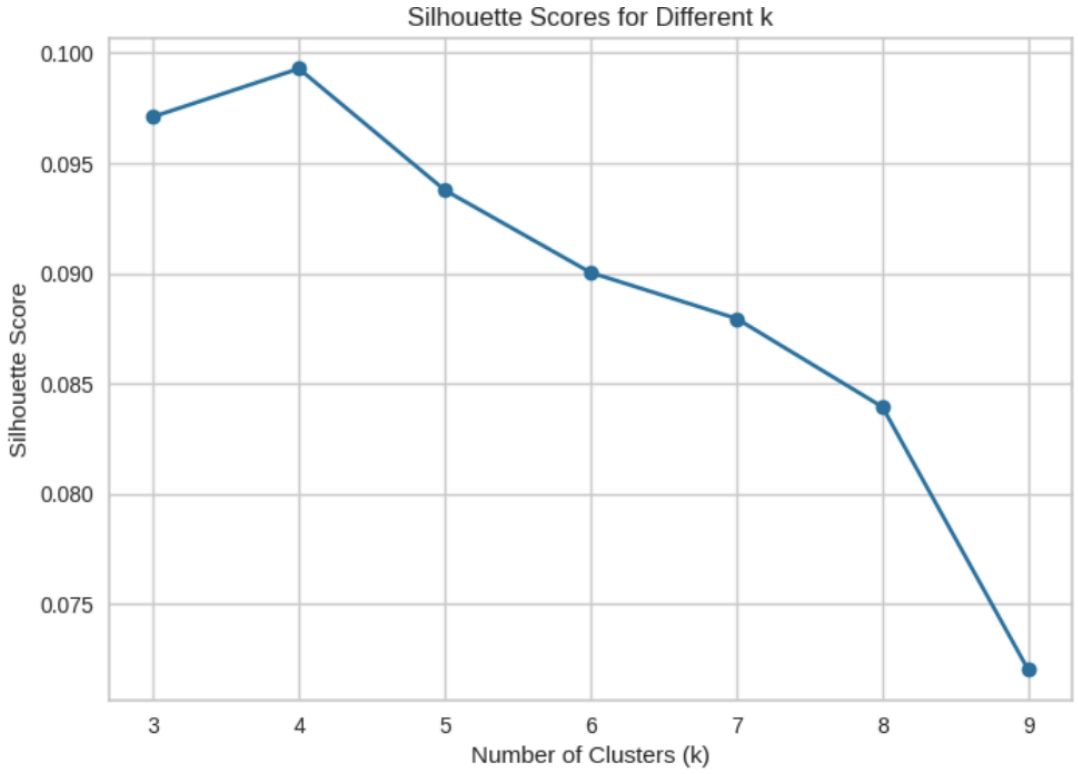
In this part, we need to create an object from the KMeans class. Then, to find the best value of k for this dataset, we will use the KElbowVisualizer class. For this, we provide the K-means model as input and the range of k values we want it to test. We then call the `fit()` function for the scaled data from the previous step.
The resulting chart will be:



We see that this class determines that the best amount for us is approximately 6; this is because we are looking for a point in the chart where the decrease in inertia starts to level off, indicating diminishing returns from adding more clusters. This point is often referred to as the "elbow" and indicates a good balance between capturing meaningful clusters and avoiding overfitting. However, we will see that the best result for us will be k=5.

Now we need to test various k values ourselves to find the best k based on the inertia and silhouette scores.





From the charts, we see that the best k equals 5. So, we solve the model again with this k.

Once clustering is completed, we can check the cluster number for each record to ensure the model has performed correctly:

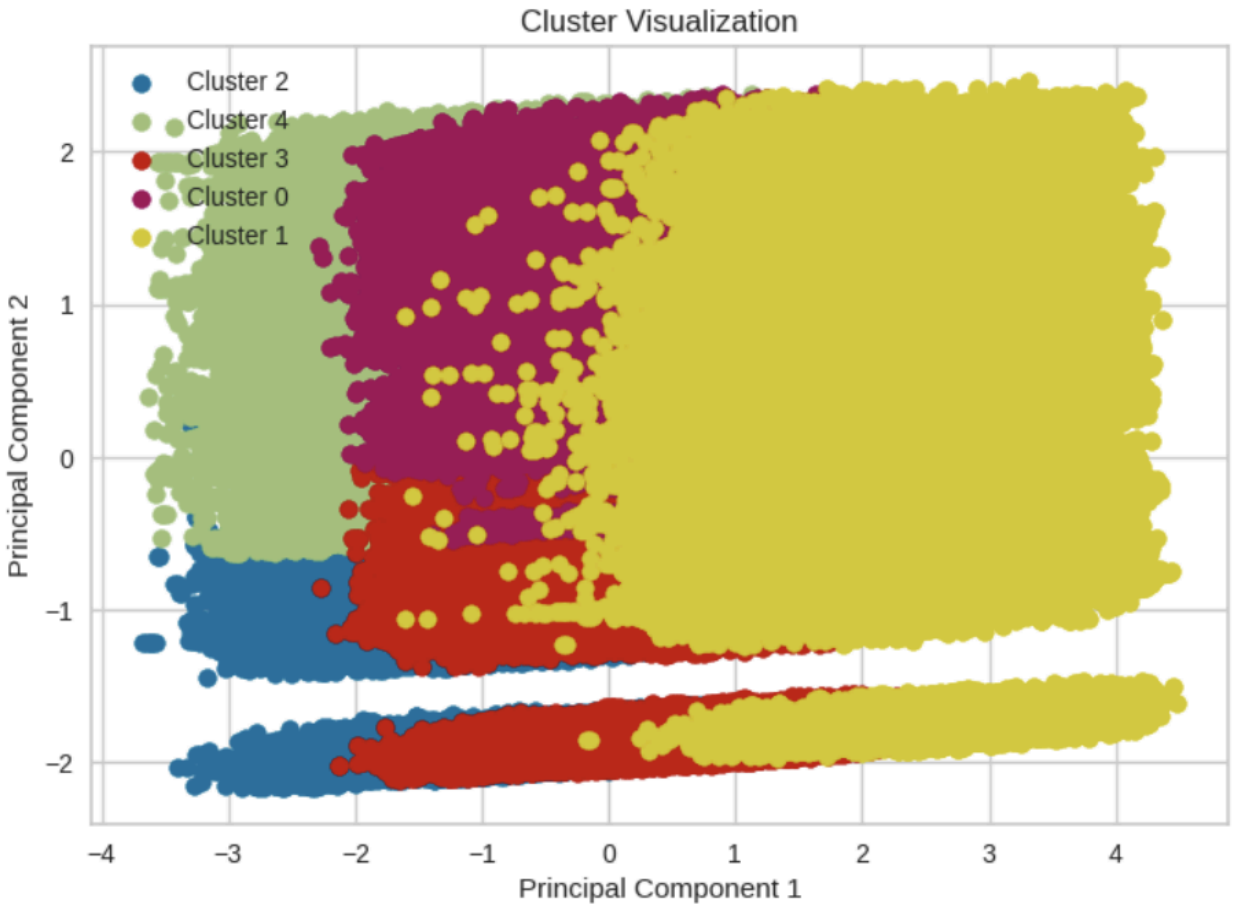
_order	product_id	add_to_cart_order	reordered	department_id	department	product_name	cluster
11	17	1	0	13	16	6	2
11	91	2	0	16	7	119	4
11	36	3	0	16	7	17	4
11	83	4	0	4	19	53	2
11	83	5	0	4	19	53	2
11	91	6	0	16	7	119	4
11	120	7	0	16	7	133	4
11	59	8	0	15	6	21	4
11	35	9	0	12	13	104	2
11	37	1	0	1	10	71	2
11	24	2	0	4	19	50	2
11	83	3	0	4	19	53	2
11	84	4	0	16	7	83	4
11	91	5	0	16	7	119	4
11	24	6	0	4	19	50	2
11	24	7	0	4	19	50	2
11	24	8	0	4	19	50	2
11	21	9	0	16	7	93	4
11	112	10	0	3	2	11	4

Then, we can check how many of our data are in each cluster:

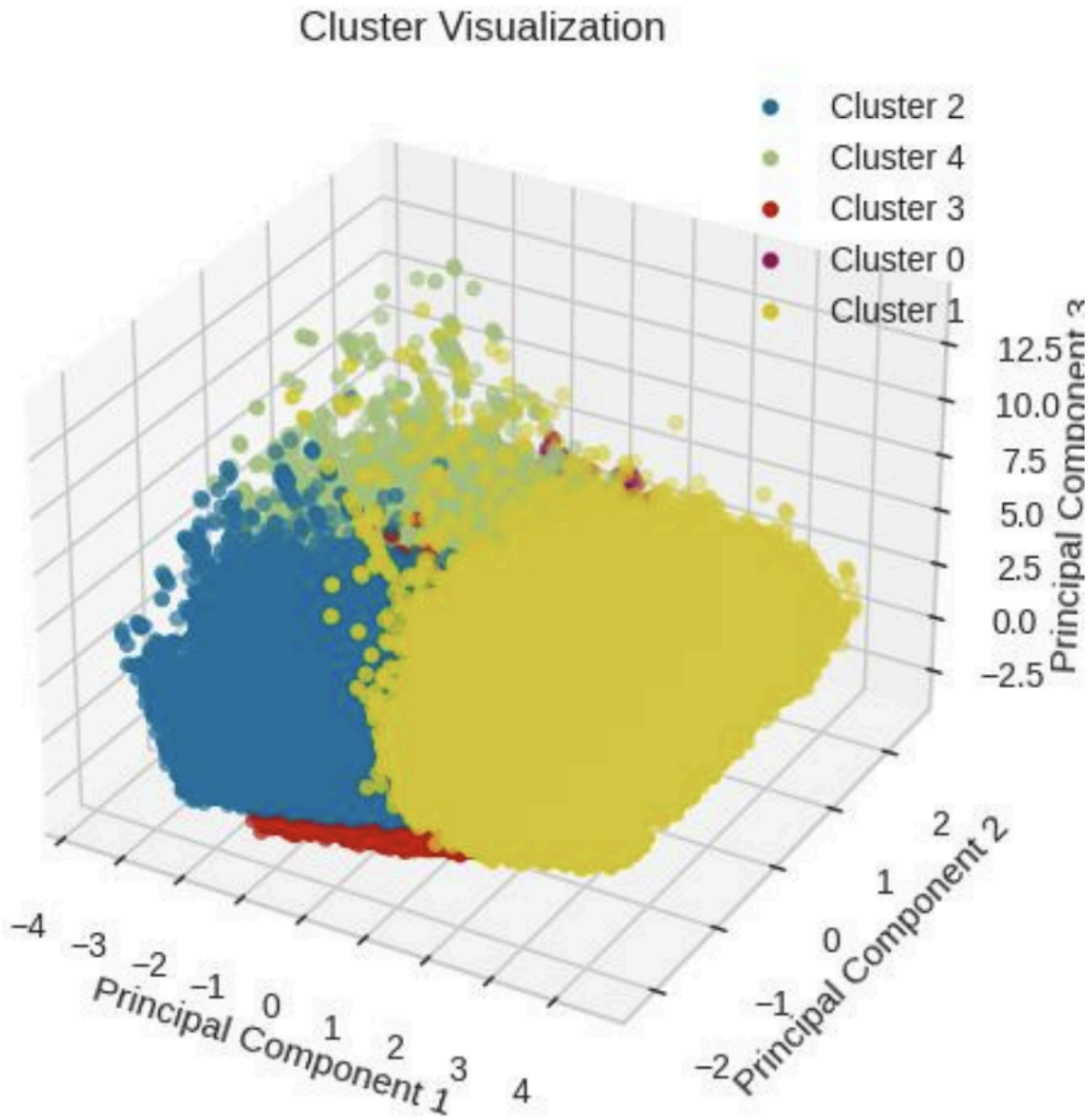
```
0    516607
3    475786
4    435354
2    359056
1    232698
Name: cluster, dtype: int64
```

Part Three: Visualization

In this part of the exercise, we will use models like PCA to display the categorized dataset in 2D and 3D.
First, we create a PCA object and set the number of components to 2. Then we send the `data_scaled` to the `transform_fit` function to apply a linear combination and convert the data to a two-dimensional space. Next, we store the transformed data in a new dataframe named `pca_df` and add two columns named '1PC' and '2PC'. Using the unique cluster values, we create a loop for these values. Finally, we display the result as follows:



And for the 3D visualization, we will do it similarly, just with three components:



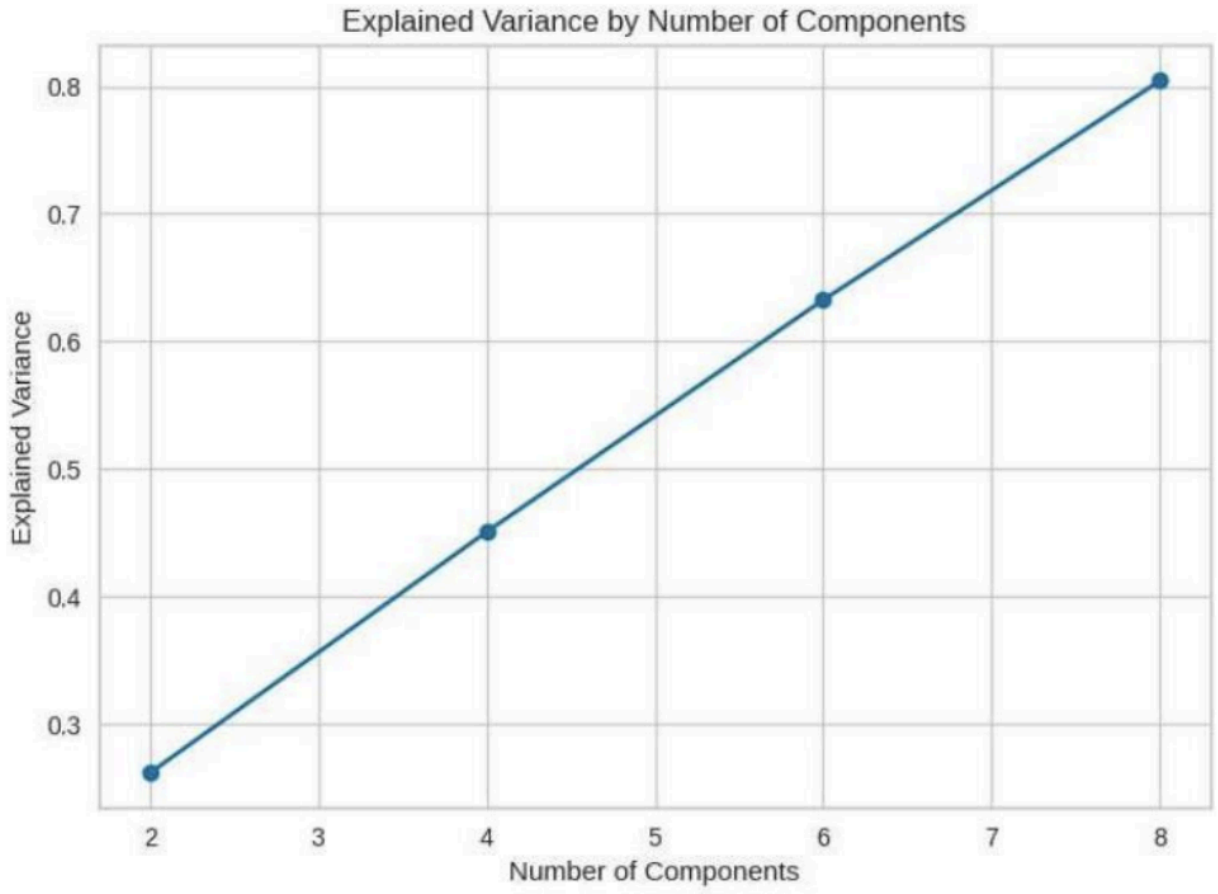
Part Four: Experimenting with Other Clustering Algorithms

In this part of the exercise, we will test other clustering algorithms on the dataset. The algorithms we will try are DBSCAN and hierarchical clustering, and we will finally compare the implementations using silhouette scores with the K-means model.

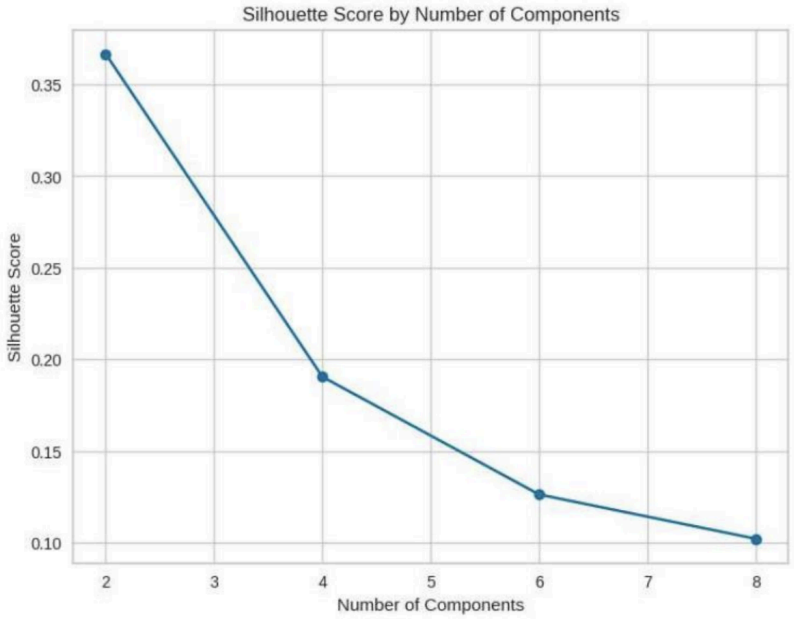
```
K-means: 0.09236717828273076
DBSCAN: DBSCAN identified only one cluster.
Hierarchical Clustering: 0.07368430608775553
```

Part Five: Different Number of Components

In this section, we will use different numbers of components to reduce dimensions to find the best number. For this, we will obtain and print the explained variance for each number of components.



If we print the silhouette score for each component, we will have:

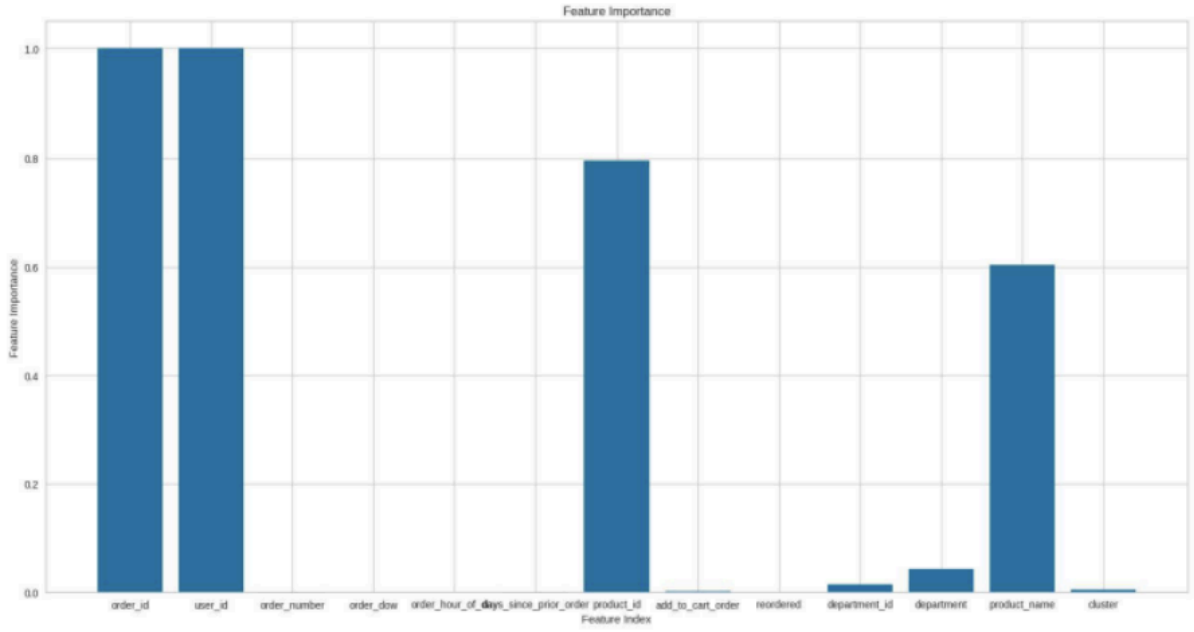


Since this value should be low, and given their graphs, it appears that component 5 will be the most suitable.

Part Six: Feature Engineering

In this section, we first need to identify the most important features of the dataset. For this, we use PCA to sequentially find the most impactful features, which will be:

```
Feature 0 -> order_id : 1.0000311749976503%
Feature 1 -> user_id : 1.0000315609391568%
Feature 2 -> order_number : 0.00047113592124569%
Feature 3 -> order_dow : 6.573607663956962e-05%
Feature 4 -> order_hour_of_day : 8.935691911234986e-05%
Feature 5 -> days_since_prior_order : 0.0012356737713178404%
Feature 6 -> product_id : 0.7954423061179605%
Feature 7 -> add_to_cart_order : 0.0026929381934968142%
Feature 8 -> reordered : 0.0002331313072859106 4%
Feature 9 -> department_id : 0.014241212976986349%
Feature 10 -> department : 0.042582343030184426%
Feature 11 -> product_name : 0.604333346844525%
Feature 12 -> cluster : 0.005334460928142816%
```



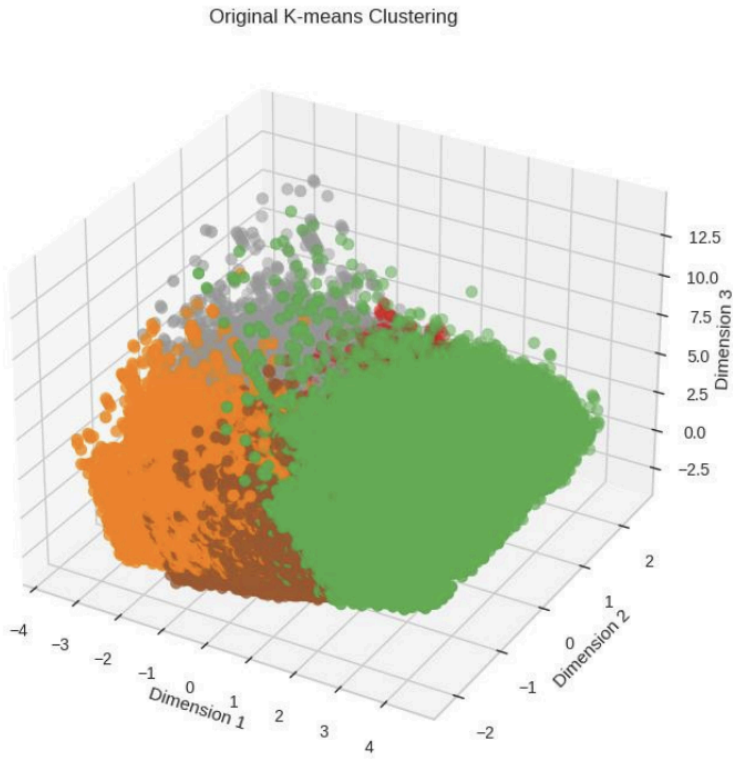
Now we need to remove the non-essential features and keep only the most important ones. Thus, we will remove the features 'order_dow', 'number_order', 'order_hour_of_day', 'days_since_prior_order', 'add_to_cart_order', 'reordered', 'department_id'.
For the remaining features, we will run the model again and measure it:

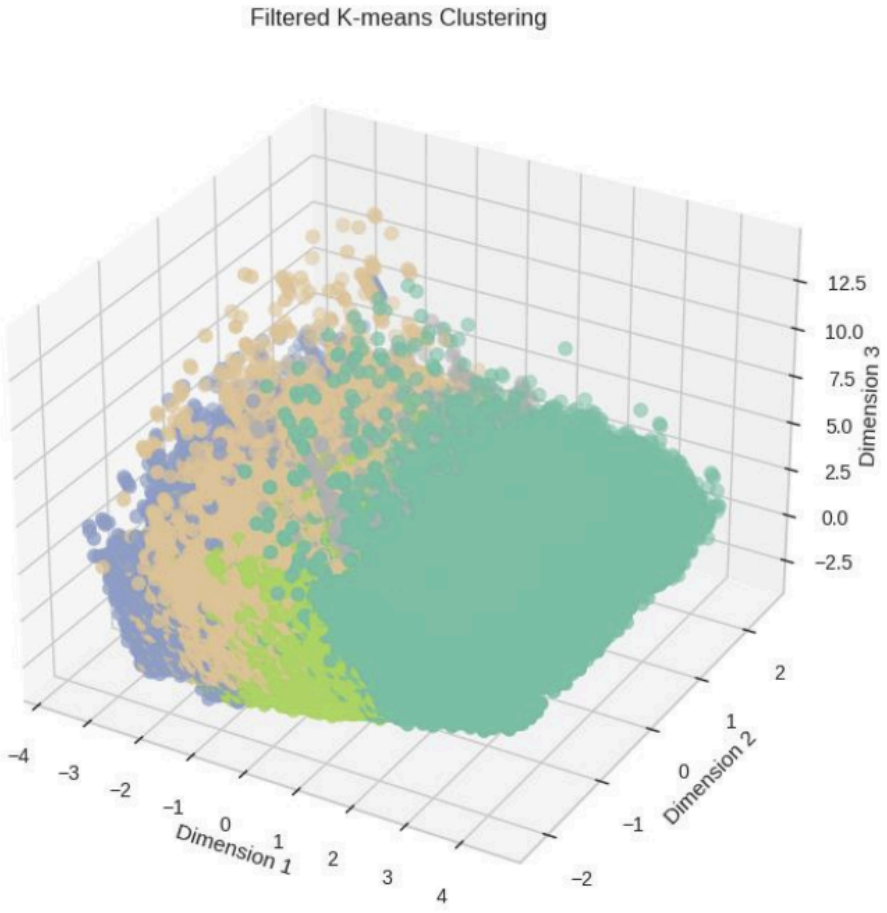
```
Original K-means Silhouette Score: 0.093380473977278 99
Original K-means Inertia: 16934521.355000548
```

```
Filtered K-means Silhouette Score: 0.5298672062884408
Filtered K-means Inertia: 2130411261927395.5
```

As observed, the model's performance has improved.
In the next step, we can add new features. We can create new features named `order_dow_category` and `avg_days_since_prior_order` and evaluate the model with these two new features:

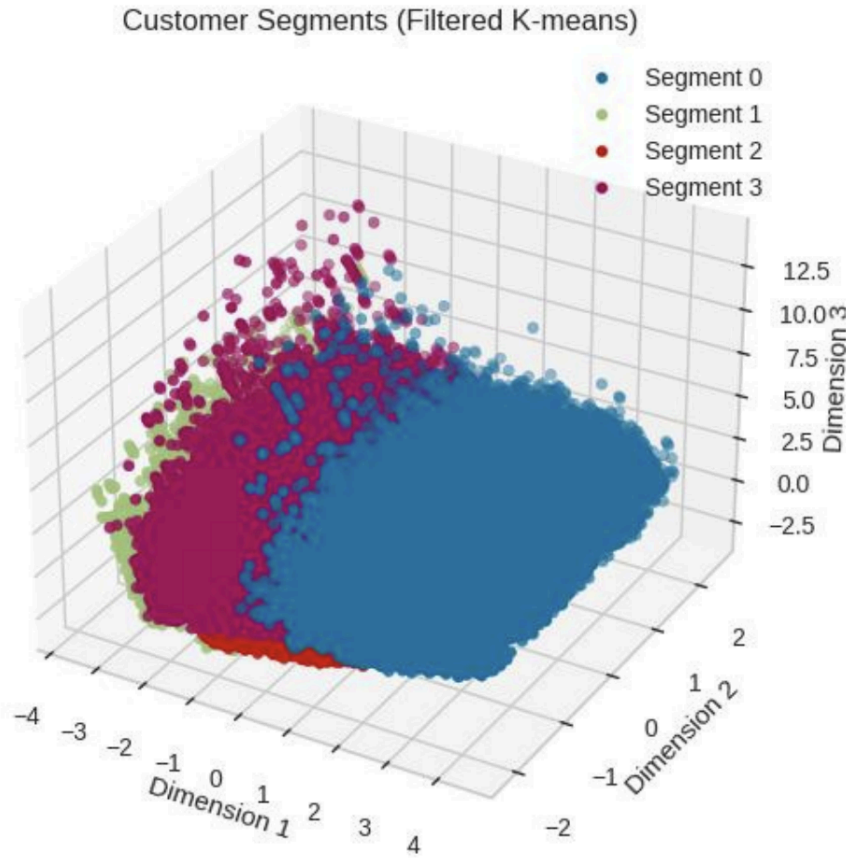
```
Original K-means Silhouette Score: 0.093380473977278 99
Original K-means Inertia: 16934521.355000548
New K-means Silhouette Score: 0.15871459493003096
New K-means Inertia: 8905079.397437796
```




Filtered K-means Clustering
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
Dimension 3
−4
−3
−2
−1
0
1
2
3
4
Dimension 1
2
1
0
−1
−2
Dimension 2

Part Seven: Insights

Finally, in the last part of the exercise, we can ultimately reach a better understanding and insight into the entire dataset. We select our best model and finally plot the dataset with 5 clusters in both 2D and 3D.


Customer Segments (Filtered K-means)
Segment 0
Segment 1
Segment 2
Segment 3
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
Dimension 3
−4
−3
−2
−1
0
1
2
3
4
Dimension 1
2
1
0
−1
−2
Dimension 2