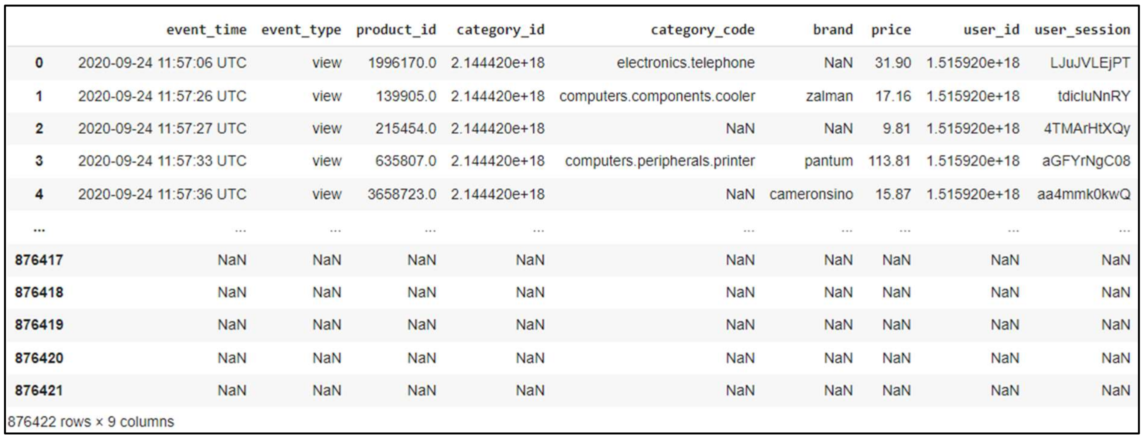
1.0 Introduction

The provided dataset as shown in Figure 1.1, consists of 876,422 rows and 9 columns capturing events that occurred between September and November 2020. The columns include information about the event time, event type, product ID, category ID, category code, brand, price, user ID, and user session.



	event_time	event_type	product_id	category_id	category_code	brand	price	user_id	user_session
0	2020-09-24 11:57:06 UTC	view	1996170.0	2.144420e+18	electronics.telephone	NaN	31.90	1.515920e+18	LJuJVLEjPT
1	2020-09-24 11:57:26 UTC	view	139905.0	2.144420e+18	computers.components.cooler	zalman	17.16	1.515920e+18	tdicluNnRY
2	2020-09-24 11:57:27 UTC	view	215454.0	2.144420e+18	NaN	NaN	9.81	1.515920e+18	4TMArHtXQy
3	2020-09-24 11:57:33 UTC	view	635807.0	2.144420e+18	computers.peripherals.printer	pantum	113.81	1.515920e+18	aGFYrNgC08
4	2020-09-24 11:57:36 UTC	view	3658723.0	2.144420e+18	NaN	cameronsino	15.87	1.515920e+18	aa4mmk0kwQ
...	...	...	...	...	...	...	...	...	...
876417	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
876418	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
876419	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
876420	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
876421	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

876422 rows × 9 columns

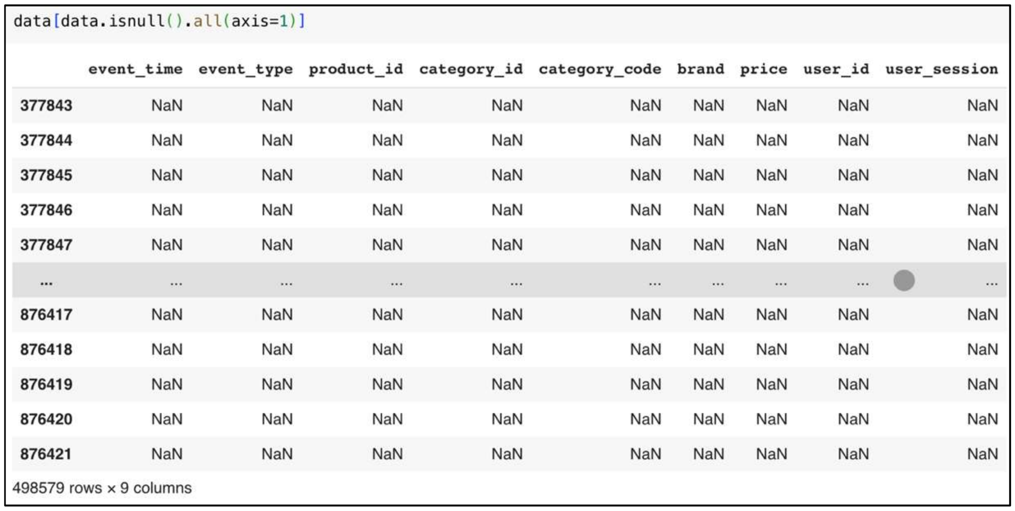
Figure 1.1

The objective of this assignment is to perform clustering on the dataset using two different techniques: partition-based clustering (k-means) and density-based clustering (DBSCAN). By applying these clustering techniques, we aim to identify meaningful groups or clusters of products based on their event types and prices.

2.0 Data Preparation

2.1 Data Cleaning

In the data cleaning process, noticed that there are 498,579 blank rows in the dataset, as indicated in Figure 2.1. These blank rows do not contain any meaningful data and may hinder the analysis and modelling processes. Therefore, these blank rows were removed from the dataset to ensure data integrity and accuracy in subsequent analyses. By removing these blank rows, we can focus on the meaningful data points and avoid any potential biases or inaccuracies caused by the presence of missing values.



```
data[data.isnull().all(axis=1)]
```

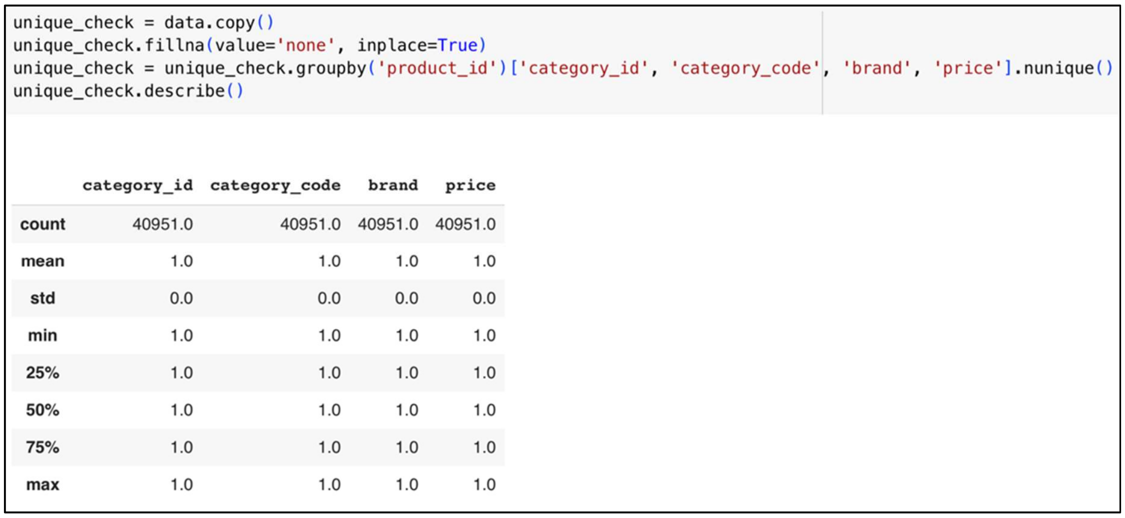
	event_time	event_type	product_id	category_id	category_code	brand	price	user_id	user_session
377843	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
377844	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
377845	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
377846	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
377847	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
876417	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
876418	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
876419	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
876420	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
876421	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

498579 rows × 9 columns

Figure 2.1

2.1 Check Uniqueness of ‘product_id’

By checking the summary statistics as shown in Figure 2.2, the uniqueness of the features ('category_id', 'category_code', 'brand', 'price') within each 'product_id' can be assessed. The count for each column is equal to the count of 'product_id', it indicates that each 'product_id' has a unique combination of features. Since the ‘product_id’ is unique, then it is indexed with its own features.



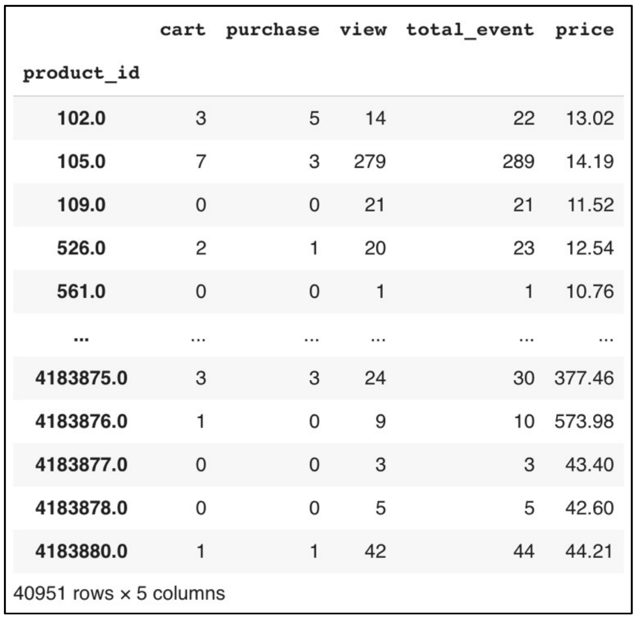
```
unique_check = data.copy()
unique_check.fillna(value='none', inplace=True)
unique_check = unique_check.groupby('product_id')['category_id', 'category_code', 'brand', 'price'].nunique()
unique_check.describe()
```

	category_id	category_code	brand	price
count	40951.0	40951.0	40951.0	40951.0
mean	1.0	1.0	1.0	1.0
std	0.0	0.0	0.0	0.0
min	1.0	1.0	1.0	1.0
25%	1.0	1.0	1.0	1.0
50%	1.0	1.0	1.0	1.0
75%	1.0	1.0	1.0	1.0
max	1.0	1.0	1.0	1.0

Figure 2.2

2.2 Feature Selection and Extraction

Figure 2.3 shows the counts of each 'event_type' (cart, purchase, and view) for each 'product_id' in the dataset. The counts are tabulated into a dataframe, where each row represents a unique 'product_id'. Additionally, a column named 'total_event' is added, which represents the total count of events (cart, purchase, and view) for each 'product_id'. Hence, the dataframe provides valuable information about the event distribution for each product.



	cart	purchase	view	total_event	price
product_id					
102.0	3	5	14	22	13.02
105.0	7	3	279	289	14.19
109.0	0	0	21	21	11.52
526.0	2	1	20	23	12.54
561.0	0	0	1	1	10.76
...	...	...	...	...	...
4183875.0	3	3	24	30	377.46
4183876.0	1	0	9	10	573.98
4183877.0	0	0	3	3	43.40
4183878.0	0	0	5	5	42.60
4183880.0	1	1	42	44	44.21

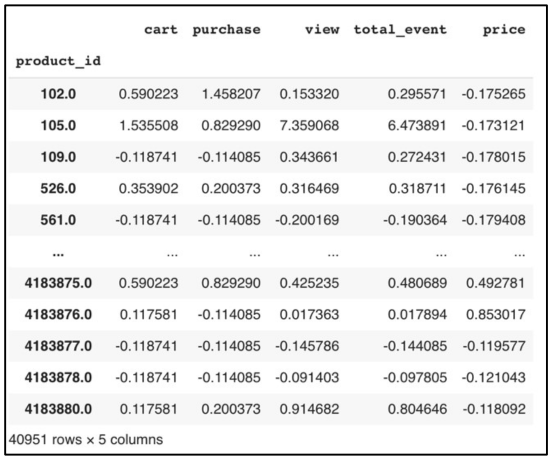
40951 rows × 5 columns

Figure 2.3

2.3 Feature Scaling

Figure 2.4 displays the scaled data after applying the Standardisation method to the selected features. The purpose of feature scaling is to ensure that all features are on a similar scale and avoid any bias during the clustering phase. In this case, the scaling is performed using the StandardScaler, which scales the data to have a range of -1 to 1.

The scaled dataframe consists of the same set of features: 'cart', 'purchase', 'view', 'total_event', and 'price'. By scaling the features, we ensure that all features contribute equally to the clustering process and prevent any feature from dominating the analysis due to differences in their scales. This facilitates a more accurate and unbiased clustering result, enabling meaningful insights to be derived from the data.



product_id	cart	purchase	view	total_event	price
102.0	0.590223	1.458207	0.153320	0.295571	-0.175265
105.0	1.535508	0.829290	7.359068	6.473891	-0.173121
109.0	-0.118741	-0.114085	0.343661	0.272431	-0.178015
526.0	0.353902	0.200373	0.316469	0.318711	-0.176145
561.0	-0.118741	-0.114085	-0.200169	-0.190364	-0.179408
...	...	...	...	...	...
4183875.0	0.590223	0.829290	0.425235	0.480689	0.492781
4183876.0	0.117581	-0.114085	0.017363	0.017894	0.853017
4183877.0	-0.118741	-0.114085	-0.145786	-0.144085	-0.119577
4183878.0	-0.118741	-0.114085	-0.091403	-0.097805	-0.121043
4183880.0	0.117581	0.200373	0.914682	0.804646	-0.118092

40951 rows × 5 columns

Figure 2.4

3.0 Exploratory Data Analysis (EDA)

In Figure 3.1, a bar chart displays the counts of each 'event_type' in the dataset. The chart reveals the following counts for each event type:

- View: 342,410 occurrences
- Cart: 20,576 occurrences
- Purchase: 14,857 occurrences

This bar chart provides an overview of the distribution of event types in the dataset, indicating that the majority of events are categorized as views, followed by a smaller number of cart additions and purchases.

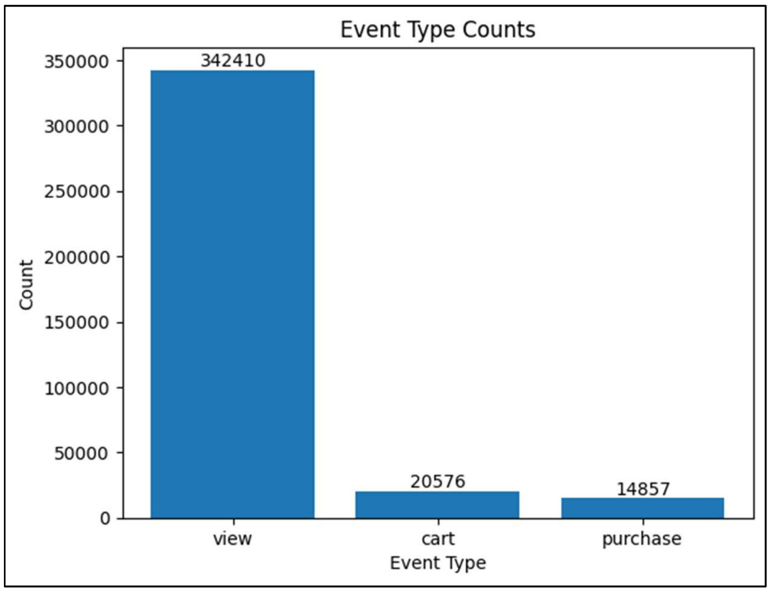


Figure 3.1

Figure 3.2 depicts a three-dimensional scatter plot of the 'product_id' distribution based on the features 'cart', 'view', 'purchase', and 'total_event' after performing Principal Component Analysis (PCA). This PCA allows us to easily visualize the scatter data point distribution before clustering. By reducing the dimensionality of the data to three principal components, we can plot the data points in a three-dimensional space, providing a visual representation of their relationships and patterns. Each data point corresponds to a specific 'product_id', and its position in the plot is determined by its values in the four selected features. The plot helps us gain insights into the distribution of the data and identify any inherent structures or clusters present in the dataset.

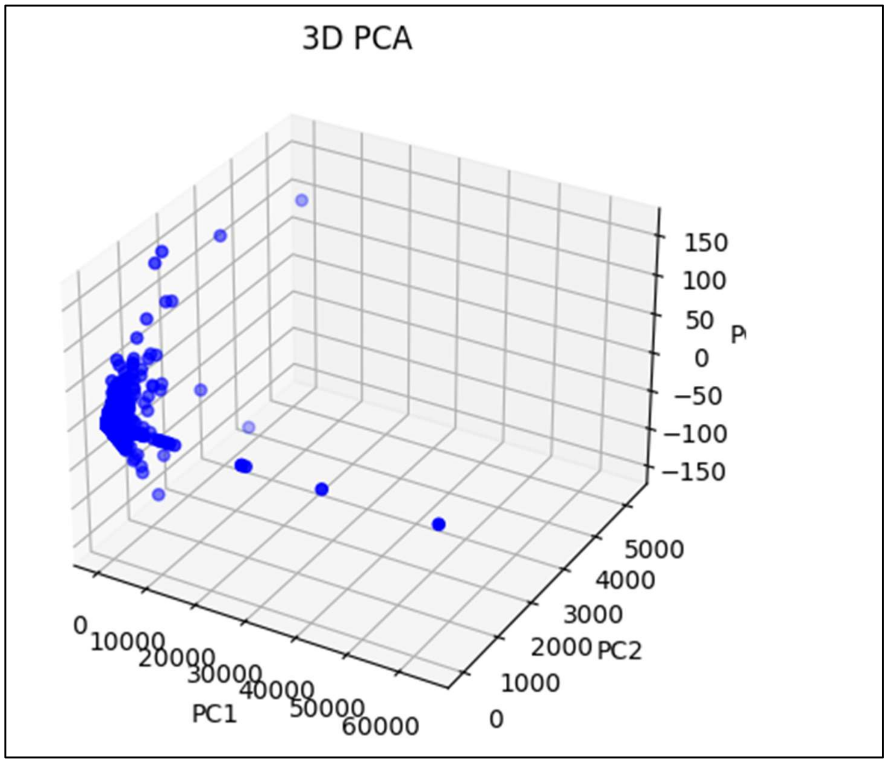
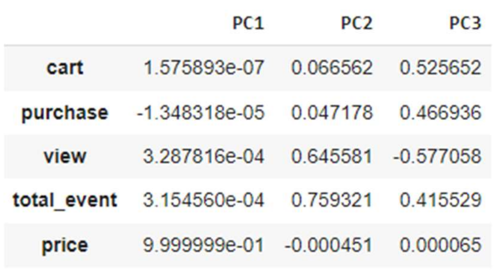


Figure 3.2

In Figure 3.3, the PCA loadings of each feature are presented. The loadings represent the contribution of each feature to the principal components. Specifically, the loadings indicate the direction and magnitude of the feature's influence on each principal component. The loadings are given for PC1, PC2, and PC3. From the loadings, it can be observed how each feature contributes to the overall variation captured by the principal components. This information is valuable for understanding the relative importance of each feature in driving the patterns observed in the scatter plot. For example, PC1 has a loading value of approximately 1 for the feature 'price'. This indicates that 'price' has a strong influence on the variation captured by PC1. A loading value of 1 means that as the 'price' increases, the values along PC1 also increase proportionally. In other words, PC1 is primarily driven by the variations in 'price'.



	PC1	PC2	PC3
cart	1.575893e-07	0.066562	0.525652
purchase	-1.348318e-05	0.047178	0.466936
view	3.287816e-04	0.645581	-0.577058
total_event	3.154560e-04	0.759321	0.415529
price	9.999999e-01	-0.000451	0.000065

Figure 3.2

4.0 Methodology

Two clustering methods are used in this project, k-means (partitioned-based) and DBSCAN (density-based).

4.1 K-Means

K-means algorithm is a partition-based clustering algorithm that aims to divide the dataset into k distinct clusters. It works by iteratively assigning data points to the nearest centroid and then updating the centroids based on the newly assigned points. The algorithm follows these steps:

1. Randomly select k data points from the dataset as the initial centroids.
2. Assign each data point to the nearest centroid based on the Euclidean distance.
3. Calculate the new centroids by taking the mean of all the data points assigned to each centroid.
4. Repeat steps 2 and 3 until convergence or a maximum number of iterations is reached.

4.1.1 Find the optimal K (number of cluster) using elbow method

The key parameter in k-means clustering is the number of clusters (k) that needs to be specified. It determines the desired number of clusters in the dataset. The elbow method is used to determine the optimal k value by plotting the inertia (within-cluster sum of squares) against the number of clusters. The elbow point represents a balance between minimising inertia and avoiding excessive fragmentation of clusters. Inertia measures the compactness of clusters by summing the squared distances between data points and their cluster centroids. The objective is to minimise inertia by creating tight clusters with data points closer to their centroids.

In Figure 4.1, the graph of the elbow method shows the inertia values for different values of k. As k increases, the inertia generally decreases because each cluster becomes more compact. However, there is a point where the rate of decrease in inertia becomes less significant, forming an elbow-like shape in the graph. The optimal value of K is typically chosen at this elbow point. In this case, the graph indicates that the optimal value of k is 4, as it is at the elbow of the graph. Therefore, k-means clustering is conducted with k=4, which means the dataset is divided into 4 clusters based on the features used in the analysis.

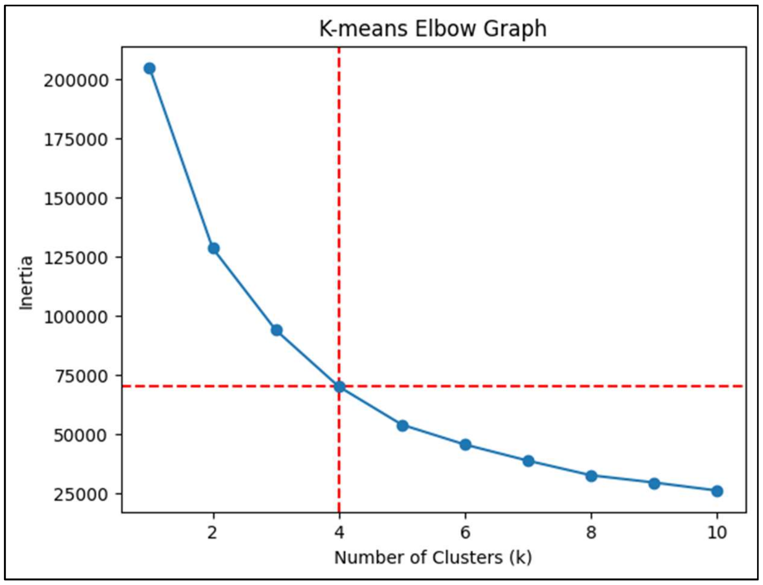
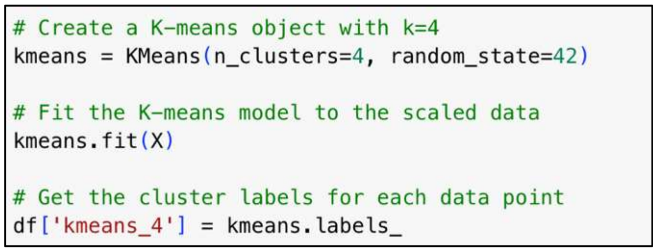


Figure 4.1

4.1.2 Implementation of k-means clustering

The k-means clustering algorithm is applied to the dataset using the code provided in Figure 4.2. A k-means object is created with k=4, representing four clusters. The model is fitted to the scaled data, and the cluster labels for each data point are obtained and stored in the 'kmeans_4' column of the dataframe.



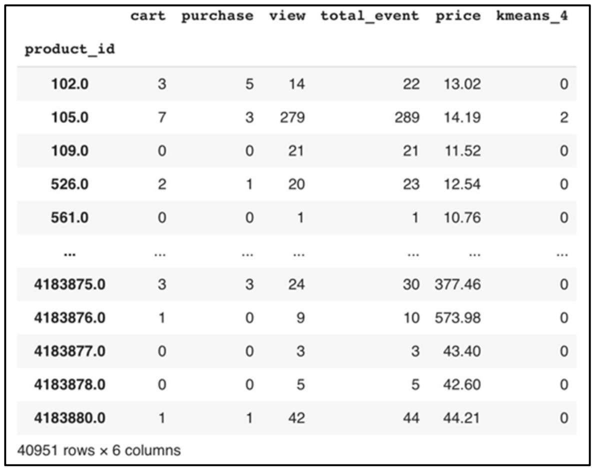
```
# Create a K-means object with k=4
kmeans = KMeans(n_clusters=4, random_state=42)

# Fit the K-means model to the scaled data
kmeans.fit(X)

# Get the cluster labels for each data point
df['kmeans_4'] = kmeans.labels_
```

Figure 4.2

The resulting dataframe, shown in Figure 4.3, displays the original features ('cart', 'purchase', 'view', 'total_event', 'price') along with the assigned cluster labels ('kmeans_4') for each 'product_id'. Each row represents a product, and the corresponding values for the features and cluster label are shown.



product_id	cart	purchase	view	total_event	price	kmeans_4
102.0	3	5	14	22	13.02	0
105.0	7	3	279	289	14.19	2
109.0	0	0	21	21	11.52	0
526.0	2	1	20	23	12.54	0
561.0	0	0	1	1	10.76	0
...	...	...	...	...	...	...
4183875.0	3	3	24	30	377.46	0
4183876.0	1	0	9	10	573.98	0
4183877.0	0	0	3	3	43.40	0
4183878.0	0	0	5	5	42.60	0
4183880.0	1	1	42	44	44.21	0

40951 rows × 6 columns

Figure 4.3

4.2 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that groups data points based on their density. It identifies dense regions as areas with a high concentration of points and expands clusters by connecting neighboring points. Unlike k-means, DBSCAN does not require specifying the number of clusters in advance. It is effective in handling clusters of arbitrary shape and can handle noisy datasets by labelling outliers as noise points.

DBSCAN, being a density-based clustering algorithm, has two important parameters: epsilon (eps) and minimum samples (min_samples).

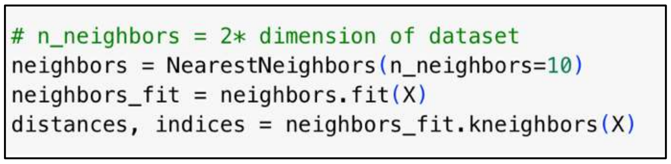
1. Epsilon (eps): It defines the radius within which neighbouring points are considered as part of the same cluster. Points within the epsilon distance are considered as "density-connected" and form clusters. The value of epsilon determines the size of the neighbourhood for each point.
2. Minimum samples (min_samples): It specifies the minimum number of neighbouring points required to form a dense region. If a point has at least min_samples neighbouring points within the epsilon radius, it becomes a core point. Points that do not meet this criteria but are within the epsilon distance of core points are considered as border points.

4.2.1 Find Optimal ‘epsilons’ by using elbow method

In the context of DBSCAN, the choice of the 'min_samples' parameter is important as it determines the minimum number of points required to form a dense region or cluster. According to Sander et al. (1998), a commonly used guideline is to set 'min_samples' to be at least twice the number of dimensions in the dataset.

Since the dataset has 5 dimensions, the 'min_samples' parameter is set to 10 tentatively, following the guideline. This means that a point must have at least 10 neighbouring points within a distance of epsilon (ε) to be considered a core point and to form a cluster. Points with fewer than 10 neighbours but within ε distance of a core point will be classified as border points, and points that have neither enough neighbours nor are within ε distance of a core point will be classified as noise points or outliers.

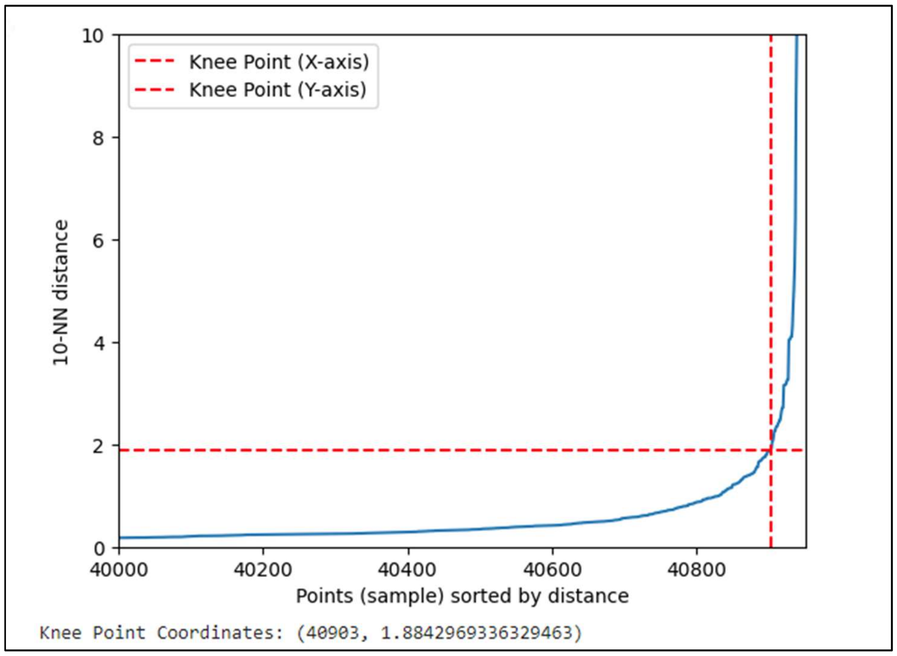
After selecting the 'min_samples' value, we can use the NearestNeighbors algorithm from Scikit-learn to calculate the average distance between each point and its 'n_neighbors'. The 'n_neighbors' parameter is set to the same value as the 'min_samples'. Figure 4.4 shows the codes of finding the k-distances.



```
# n_neighbors = 2* dimension of dataset
neighbors = NearestNeighbors(n_neighbors=10)
neighbors_fit = neighbors.fit(X)
distances, indices = neighbors_fit.kneighbors(X)
```

Figure 4.4

Figure 4.5 shows the k-distance elbow graph to determine the optimal value for epsilon (distance threshold for determining the neighbourhood of a point). The epsilon value is chosen at the point of maximum curvature in the graph, known as the elbow. In this case, the epsilon value found is 1.884 as shown in Figure 4.5.

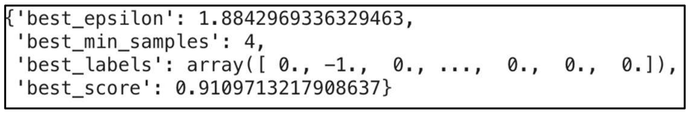


```
Knee Point Coordinates: (40903, 1.8842969336329463)
```

Figure 4.5

4.2.2 Find Optimal 'min_samples' by using Trial and Error DBSCAN Clustering

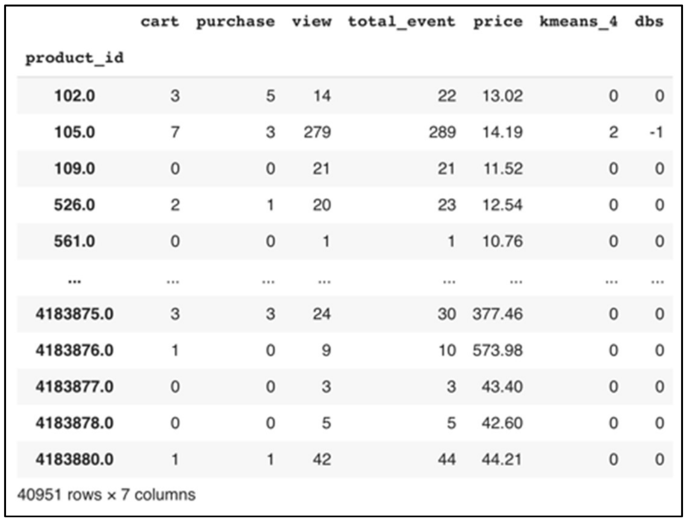
To determine the appropriate 'min_samples' parameter for DBSCAN clustering, a trial and error approach is used. After finding the optimal 'epsilon' value, we proceed to find the best DBSCAN model with the highest silhouette score. The 'min_samples' parameter is varied within a range from 2 to 7, and the silhouette scores are evaluated for each value. The 'min_samples' value that yields the best silhouette score (0.911) is determined to be 4 as shown in Figure 4.6 below.



```
{'best_epsilon': 1.8842969336329463,
 'best_min_samples': 4,
 'best_labels': array([ 0., -1.,  0., ...,  0.,  0.,  0.]),
 'best_score': 0.9109713217908637}
```

Figure 4.6

Finally, the cluster labels are assigned to the 'dbs' column in the data frame as shown in Figure 4.7.



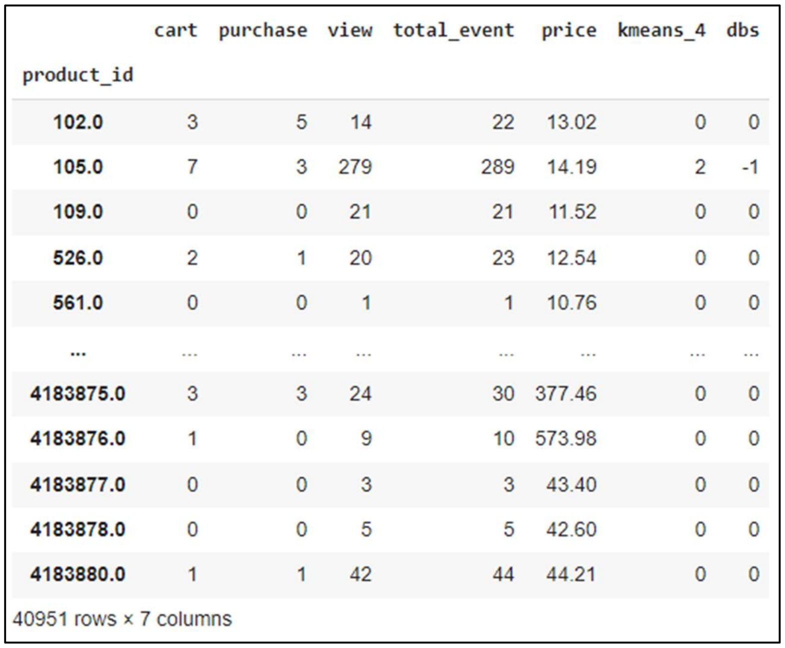
product_id	cart	purchase	view	total_event	price	kmeans_4	dbs
102.0	3	5	14	22	13.02	0	0
105.0	7	3	279	289	14.19	2	-1
109.0	0	0	21	21	11.52	0	0
526.0	2	1	20	23	12.54	0	0
561.0	0	0	1	1	10.76	0	0
...	...	...	...	...	...	...	...
4183875.0	3	3	24	30	377.46	0	0
4183876.0	1	0	9	10	573.98	0	0
4183877.0	0	0	3	3	43.40	0	0
4183878.0	0	0	5	5	42.60	0	0
4183880.0	1	1	42	44	44.21	0	0

40951 rows × 7 columns

Figure 4.7

5.0 Results and Interpretation

The dataframe presented in Figure 5.1 displays the results of applying k-means and DBSCAN clustering algorithms. In the k-means cluster, the labels are assigned as [0, 1, 2, 3]. On the other hand, the DBSCAN cluster consists of three labels, namely [-1, 0, 1].



	cart	purchase	view	total_event	price	kmeans_4	dbs
product_id							
102.0	3	5	14	22	13.02	0	0
105.0	7	3	279	289	14.19	2	-1
109.0	0	0	21	21	11.52	0	0
526.0	2	1	20	23	12.54	0	0
561.0	0	0	1	1	10.76	0	0
...	...	...	...	...	...	...	...
4183875.0	3	3	24	30	377.46	0	0
4183876.0	1	0	9	10	573.98	0	0
4183877.0	0	0	3	3	43.40	0	0
4183878.0	0	0	5	5	42.60	0	0
4183880.0	1	1	42	44	44.21	0	0

40951 rows × 7 columns

Figure 5.1

5.1 Cluster Analysis by Visualisation

Scatter plots were generated for both k-means and DBSCAN clustering techniques, using the 'cart', 'total_event', and 'price' axes based on PCA loadings. These scatter plots provide a clear visualization of the cluster positions along these feature axes. It is evident that the clusters identified by k-means differ from those identified by DBSCAN. The clustering results from both techniques can be further analyzed using centroid methods and silhouette score later. Table 5.1 and 5.2 present the tabulated results of the 2D and 3D scatter plots for both k-means and DBSCAN techniques.

Table 5.1

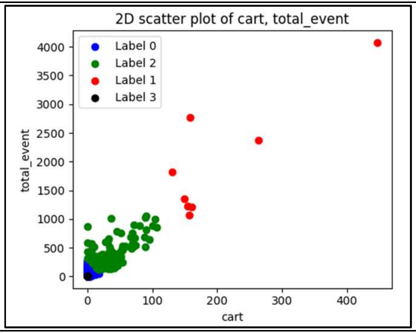
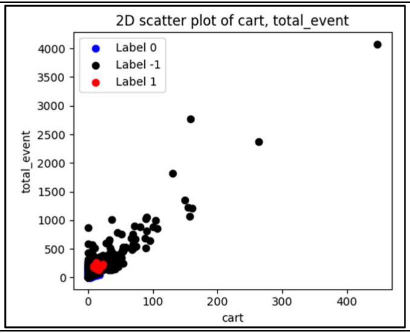
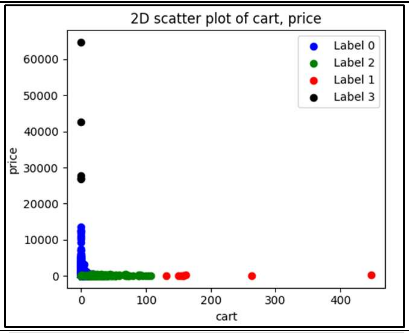
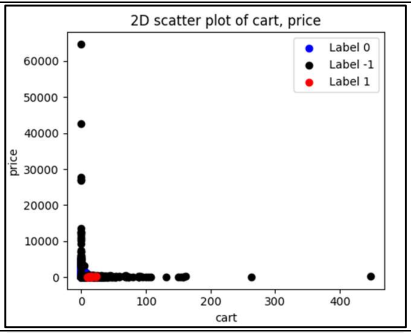
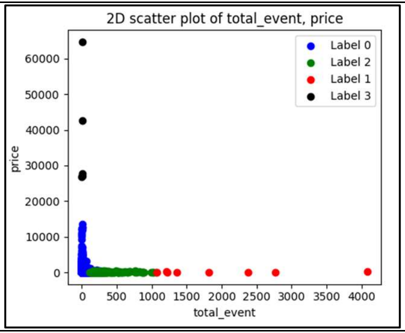
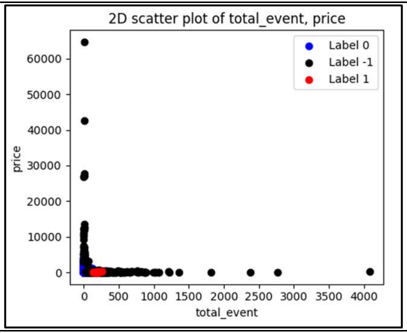
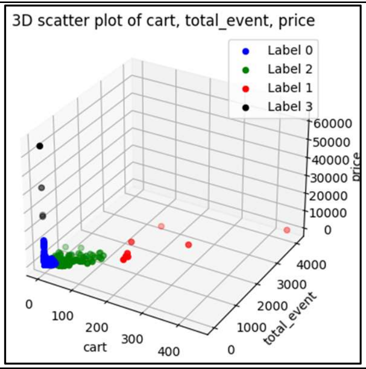
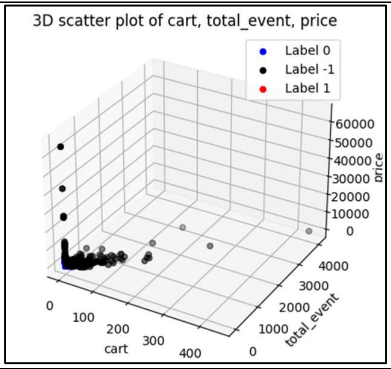
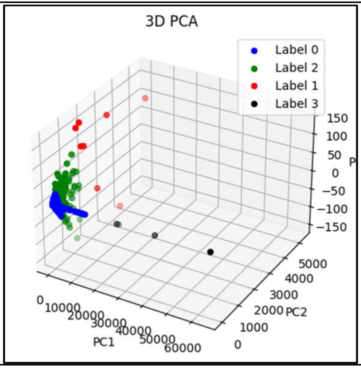
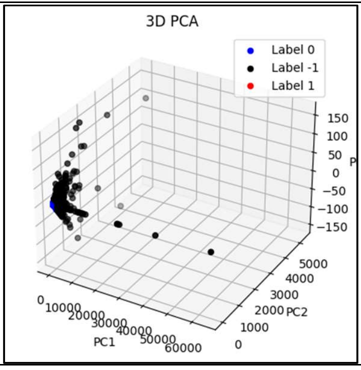
Scatter Plot Axis	K-means	DBSCAN
Cart vs total_event		
Cast vs price		
Total_event vs price		

Table 5.2

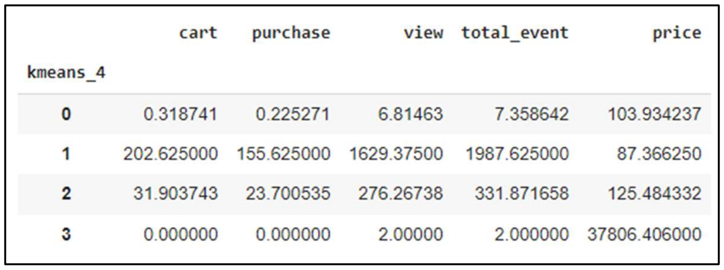
Scatter Plot Axis	**K-means**	**DBSCAN**
Cart, total_event, price		
PC1, PC2, PC3		

5.2 Cluster Analysis by Centroid.

Cluster analysis by centroid involves examining the centroids of each cluster to gain insights into the characteristics and differences among the clusters. The centroid represents the central point or average values of the feature variables within a cluster.

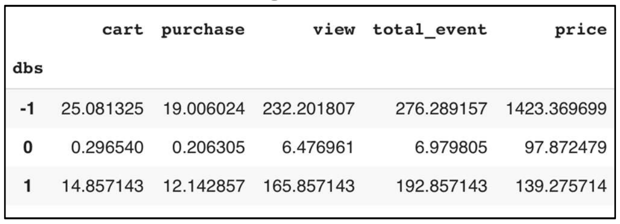
By analyzing the centroids, we can observe the typical feature values that define each cluster and understand how they differ from one another. This analysis can provide valuable information about the distinct characteristics or behaviors exhibited by different clusters.

To perform cluster analysis by centroid, we calculate the mean or average values of the feature variables for each cluster. By comparing these centroid values across clusters, we can identify patterns, trends, and differences in the feature characteristics among the clusters. The feature value of cluster centroid from k-means and DBSCAN clustering are shown in Figure 5.2 and 5.3 below.



kmeans_4	cart	purchase	view	total_event	price
0	0.318741	0.225271	6.81463	7.358642	103.934237
1	202.625000	155.625000	1629.37500	1987.625000	87.366250
2	31.903743	23.700535	276.26738	331.871658	125.484332
3	0.000000	0.000000	2.00000	2.000000	37806.406000

Figure 5.2



dbs	cart	purchase	view	total_event	price
-1	25.081325	19.006024	232.201807	276.289157	1423.369699
0	0.296540	0.206305	6.476961	6.979805	97.872479
1	14.857143	12.142857	165.857143	192.857143	139.275714

Figure 5.3

K-means cluster are analysed based on the centroid of its cluster, and tabulated in Table 5.3 below:

Table 5.3: Description of k-means cluster

Cluster	Description
0	• has relatively low values for all features, indicating a group of products with low engagement and lower prices.
1	• has significantly higher values for all features, particularly in terms of cart, purchase, view, and total_event. • suggests a group of products with high engagement and relatively moderate prices.
2	• has intermediate values for all features. • values for cart, purchase, view, and total_event are higher than Cluster 0 but lower than Cluster 1. • average price is higher than Cluster 0 but lower than Cluster 1.
3	• has very low values for cart, purchase, view, and total_event, indicating a group of products with minimal engagement. • However, the average price is exceptionally high compared to the other clusters.

In k-means clustering, Cluster 1 represents the most engaged group with moderate prices, while Cluster 3 stands out with very high prices with low engagement. Clusters 0 and 2 fall between these extremes with varying levels of engagement and prices.

DBSCAN clusters are also analysed based on the centroid of its cluster, in Table 5.4 below:

Table 5.3: Description of DBSCAN cluster

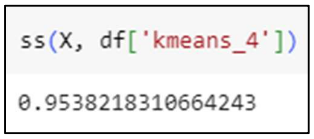
Cluster	Description
-1	• represents the outliers or noise points in the dataset as they are assigned the label -1. The average values indicate relatively higher values for cart, purchase, view, total_event, and price compared to the other clusters. • may represent products that have significantly different engagement levels and prices compared to the majority of the dataset.
0	• has relatively low values for all features, indicating a group of products with low engagement and lower prices. • These points are more densely clustered compared to the outliers.
1	• has intermediate values for all features. • values for cart, purchase, view, and total_event are higher than Cluster 0 but lower than Cluster -1. • average price is also higher than Cluster 0 but lower than Cluster -1.

In DBSCAN clustering, Cluster 0 represents the least engaged group with lower prices, while Cluster 1 represents a group with intermediate engagement levels and prices. Cluster -1 consists of outliers that exhibit significantly different engagement levels and prices compared to the majority of the dataset.

5.3 Silhouette Score

The Silhouette Score is a metric used to evaluate the quality of clustering results. It measures how well each data point fits within its assigned cluster compared to other clusters. The Silhouette Score ranges from -1 to 1, where a score closer to 1 indicates that data points are well-clustered and separated, while a score closer to -1 suggests that data points may have been assigned to the wrong clusters.

In this case, the Silhouette Score of k-means is 0.954 as shown in Figure 5.4, which is close to 1. This indicates that the clustering result of k-means is of high quality, with well-separated and compact clusters. The high Silhouette Score suggests that data points within each cluster are more similar to each other than to data points in other clusters.

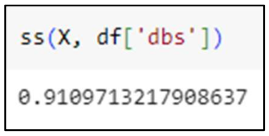


```
ss(X, df['kmeans_4'])

0.9538218310664243
```

Figure 5.4

On the other hand, the Silhouette Score of DBSCAN is 0.911 as shown in Figure 5.5, which is also relatively high but slightly lower than that of k-means. This suggests that the clustering result of DBSCAN is also of good quality, but the degree of separation and compactness of the clusters may be slightly lower compared to k-means.



```
ss(X, df['dbs'])

0.9109713217908637
```

Figure 5.5

In conclusion, the difference in Silhouette Scores between k-means and DBSCAN could be attributed to the inherent differences in the algorithms and their assumptions. K-means aims to minimize the within-cluster sum of squares, resulting in well-defined and compact clusters. DBSCAN, on the other hand, identifies clusters based on density and neighborhood relationships, allowing for more flexibility in cluster shapes and sizes.