High Value Customer Identification

Anuradha Tidke

Priya Johny

Rakshith Reddy Eleti

Venkata Gangadhar Naveen Palaka

Data Mining

Department of Data Science

George Washington University

Table of Content

1. Introduction

A UK-based online retail store has captured the sales data for different products for the period of one year (Nov 2016 to Dec 2017) and updated the dataset (in 2020). The organization sells gifts primarily on the online platform. The customers who make a purchase, consume directly for themselves. There are small businesses that buy in bulk and sell to other customers through the retail outlet channel. The project objective is to find significant customers for the business who make high purchases of their favorite products. The organization wants to roll out a loyalty program to high-value customers after the identification of segments.

The Data mining algorithm used is K-means clustering and it is in standard form (see Appendix A for in depth knowledge). The packages used to implement the network are NumPy, Pandas, Seaborn, Matplotlib, Sklearn and Scipy. For the K-means clustering method, the most common approach for metric evaluation is the so-called elbow method. It involves running the algorithm multiple times over a loop, with an increasing number of cluster choices, and then plotting a clustering score as a function of the number of clusters.

2. Data Description

The dataset used is a trans-national dataset that contains all the transactions occurring between Nov-2016 to Dec-2017 for a UK-based online retail store. It contains over 541909 entries and 9 columns. Attributes = 'InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate', 'UnitPrice', 'CustomerID', 'Country', ‘ ’.

Attribute Description

- InvoiceNo: Invoice number (a 6-digit integral number uniquely assigned to each transaction)
- StockCode: Product (item) code
- Description: Product (item) name
- Quantity: The quantities of each product (item) per transaction
- InvoiceDate: The day when each transaction was generated
- UnitPrice: Product price per unit
- CustomerID: Customer number (Unique ID assigned to each customer)
- Country: Country name (the name of the country where each customer resides)

3. Algorithm Description

K-means clustering is a method of vector quantization, originally from signal processing, that aims to partition ‘n’ observations into ‘k’ clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. k-means clustering minimizes within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimizes squared errors, whereas only the geometric median minimizes Euclidean distances. For instance, better Euclidean solutions can be found using k-medians and k-medoids.

The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both k-means and Gaussian mixture modeling. They both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the Gaussian mixture model allows clusters to have different shapes.

The unsupervised k-means algorithm has a loose relationship to the k-nearest neighbor classifier, a popular supervised machine learning technique for classification that is often confused with k-means due to the name. Applying the 1-nearest neighbor classifier to the cluster centers obtained by k-means classifies new data into the existing clusters. This is known as the nearest centroid classifier or Rocchio algorithm.

Given a set of observations (x_1, x_2, ..., x_n), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k (≤ n) sets S = {S_1, S_2, ..., S_k} so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance).

Formally, the objective is to find:

$$\boxed{\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2 = \underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i}$$

where, μ_i is the mean of points in S_i. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

where μ_i is the mean of points in S_i. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

The equivalence can be deduced from identity

$$\sum_{\mathbf{x}\in S_i} \|\mathbf{x}-\boldsymbol{\mu}_i\|^2 = \sum_{\mathbf{x}\neq\mathbf{y}\in S_i} (\mathbf{x}-\boldsymbol{\mu}_i)^T(\boldsymbol{\mu}_i-\mathbf{y})$$

Because the total variance is constant, this is equivalent to maximizing the sum of squared deviations between points in *different* clusters (between-cluster sum of squares, BCSS), which follows from the law of total variance.

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes.

A cluster refers to a collection of data points aggregated together because of certain similarities.

You’ll define a target number k, which refers to the number of centroids you need in the dataset.

A centroid is the imaginary or real location representing the center of the cluster.

Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares. In other words, the K-means algorithm identifies the k - number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.

The ‘means’ in the K-means refers to averaging of the data; that is, finding the centroid.

4. Working of K-means algorithm

To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids

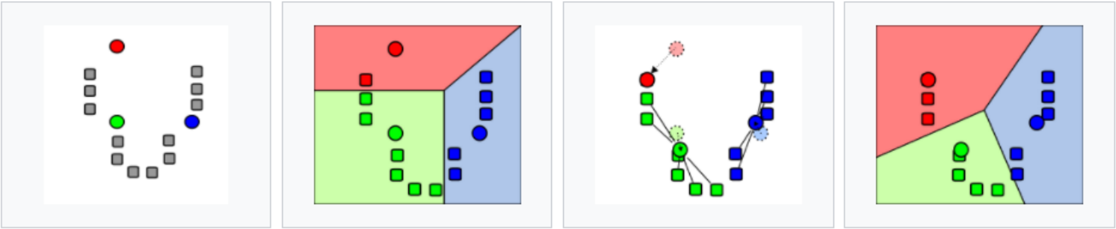
It halts creating and optimizing clusters when either:

- The centroids have stabilized — there is no change in their values because the clustering has been successful.
- The defined number of iterations has been achieved.

Initialization methods:

Commonly used initialization methods are Forgy and Random Partition The Forgy method randomly chooses *k* observations from the dataset and uses these as the initial means. The Random Partition method first randomly assigns a cluster to each observation and then proceeds to the update step, thus computing the initial mean to be the centroid of the cluster's randomly assigned points. The Forgy method tends to spread the initial means out, while Random Partition places all of them close to the center of the data set. According to Hamerly et al., the Random Partition method is generally preferable for algorithms such as the *k*-harmonic means and fuzzy *k*-means. For expectation maximization and standard *k*-means algorithms, the Forgy method of initialization is preferable. A comprehensive study by Celebi et al., however, found that popular initialization methods such as Forgy, Random Partition, and Maximin often perform poorly, whereas Bradley and Fayyad's approach performs "consistently" in "the best group" and *k*-means++ performs "generally well".

Demonstration of the standard algorithm



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

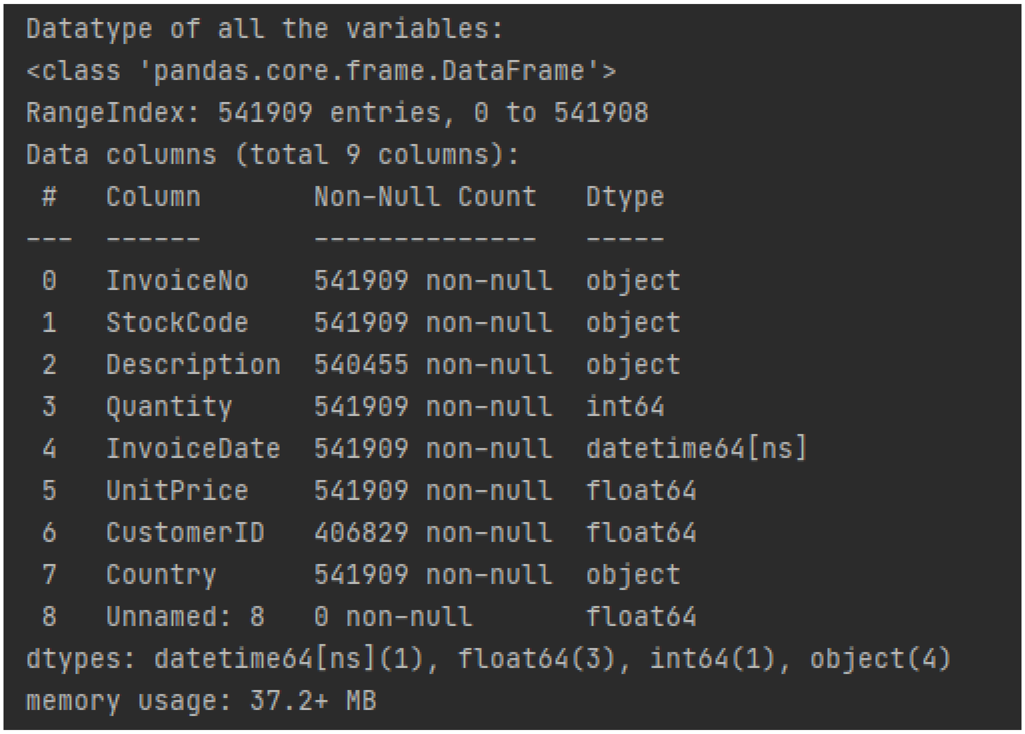
2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

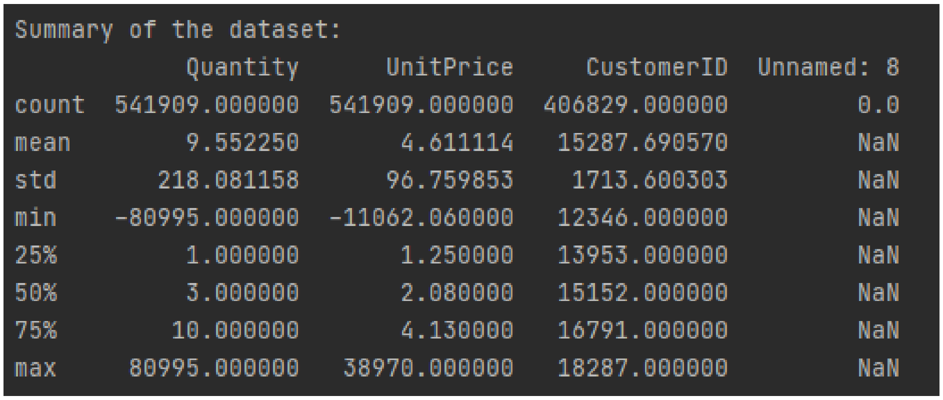
5. Experimental Setup

We have coded using pycharm professional tool.The dataset has 541909 observations and 9 variables. We have read the dataset which is in csv format into a dataframe with data parsing of InvoiceDate and encoding. Then we checked the data types.



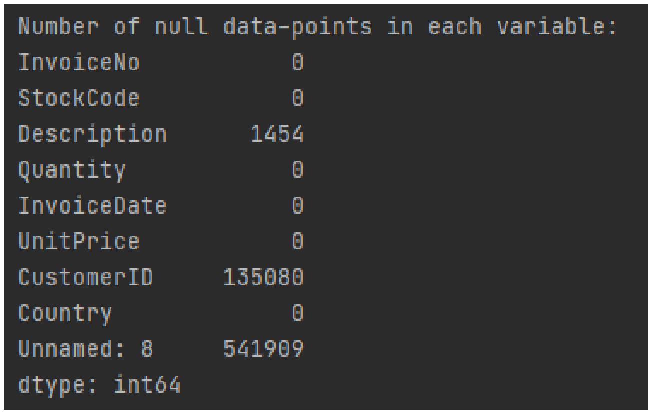
```
Datatype of all the variables:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 9 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
 8   Unnamed: 8   0 non-null       float64
dtypes: datetime64[ns](1), float64(3), int64(1), object(4)
memory usage: 37.2+ MB
```

Summary of the dataset to observe statistical values for int/float dtypes features.



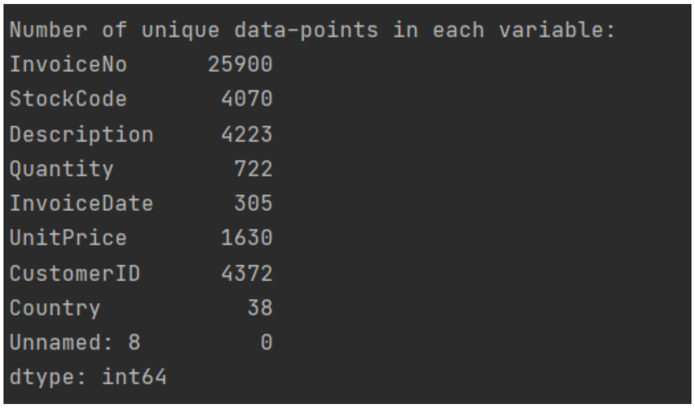
```
Summary of the dataset:
            Quantity      UnitPrice     CustomerID  Unnamed: 8
count  541909.000000  541909.000000  406829.000000         0.0
mean        9.552250       4.611114   15287.690570         NaN
std       218.081158      96.759853    1713.600303         NaN
min    -80995.000000  -11062.060000   12346.000000         NaN
25%         1.000000       1.250000   13953.000000         NaN
50%         3.000000       2.080000   15152.000000         NaN
75%        10.000000       4.130000   16791.000000         NaN
max     80995.000000   38970.000000   18287.000000         NaN
```

We checked for null values count, and found ‘Description’ has 1454, CustomerID has 135080 and Unnamed has 541909 null values



```
Number of null data-points in each variable:
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
Unnamed: 8     541909
dtype: int64
```

Here, we can see the number of unique data points in each variable.

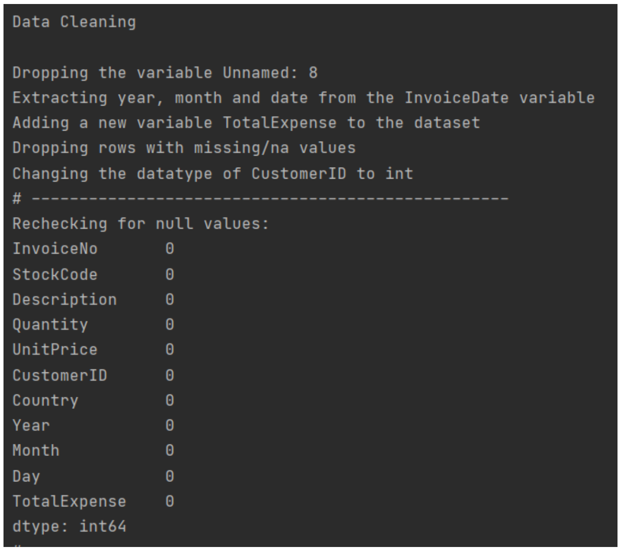


```
Number of unique data-points in each variable:
InvoiceNo      25900
StockCode       4070
Description     4223
Quantity         722
InvoiceDate      305
UnitPrice       1630
CustomerID      4372
Country           38
Unnamed: 8         0
dtype: int64
```

Data Cleaning

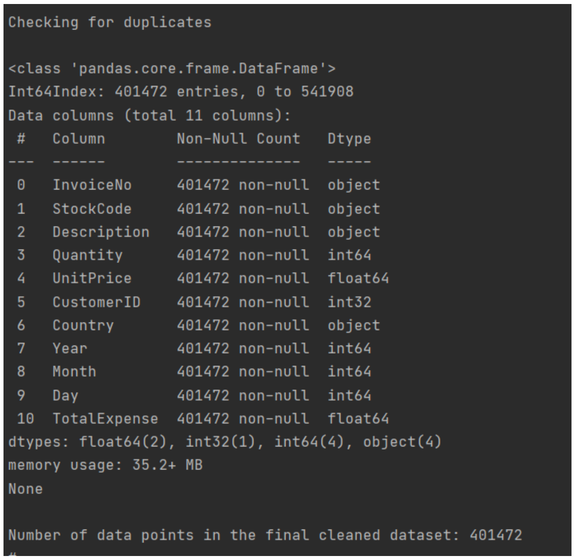
We are dropping unnecessary variables ‘unnamed: 8’, extracting year, month and date from the ‘InvoiceDate’ variable and adding a new variable ‘TotalExpense’ to the dataset. Next we dropped rows with missing/NA values and changed the datatype of ‘CustomerID’ to int.

After data cleaning, there are 11 columns with 401472 observations and no null values



```
Data Cleaning

Dropping the variable Unnamed: 8
Extracting year, month and date from the InvoiceDate variable
Adding a new variable TotalExpense to the dataset
Dropping rows with missing/na values
Changing the datatype of CustomerID to int
# -------------------------------------------------
Rechecking for null values:
InvoiceNo       0
StockCode       0
Description     0
Quantity        0
UnitPrice       0
CustomerID      0
Country         0
Year            0
Month           0
Day             0
TotalExpense    0
dtype: int64
```



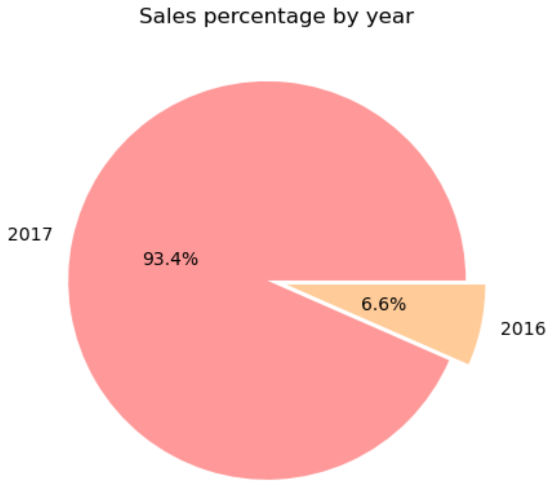
```
Checking for duplicates

<class 'pandas.core.frame.DataFrame'>
Int64Index: 401472 entries, 0 to 541908
Data columns (total 11 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   InvoiceNo     401472 non-null  object 
 1   StockCode     401472 non-null  object 
 2   Description   401472 non-null  object 
 3   Quantity      401472 non-null  int64  
 4   UnitPrice     401472 non-null  float64
 5   CustomerID    401472 non-null  int32  
 6   Country       401472 non-null  object 
 7   Year          401472 non-null  int64  
 8   Month         401472 non-null  int64  
 9   Day           401472 non-null  int64  
 10  TotalExpense  401472 non-null  float64
dtypes: float64(2), int32(1), int64(4), object(4)
memory usage: 35.2+ MB
None

Number of data points in the final cleaned dataset: 401472
```

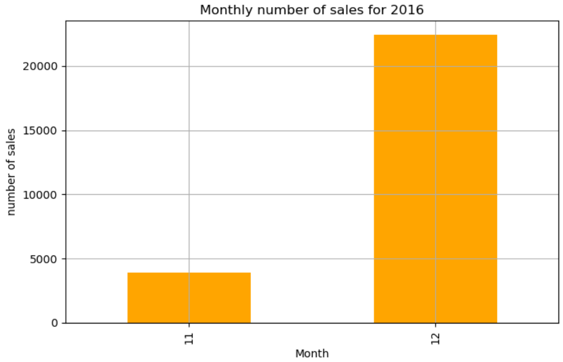
6. Exploratory Data Analysis

- Yearly proportion of number of sales

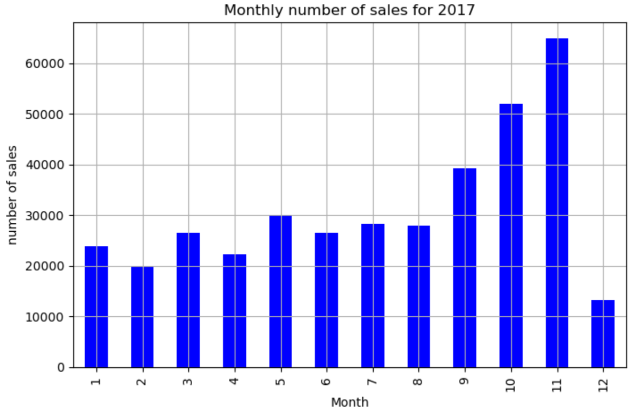


We can observe in the pie chart, in the year 2016 it records 6.6% which is very low as compared to the sales in the year 2017 which is 93.4%

- Monthly sales by year

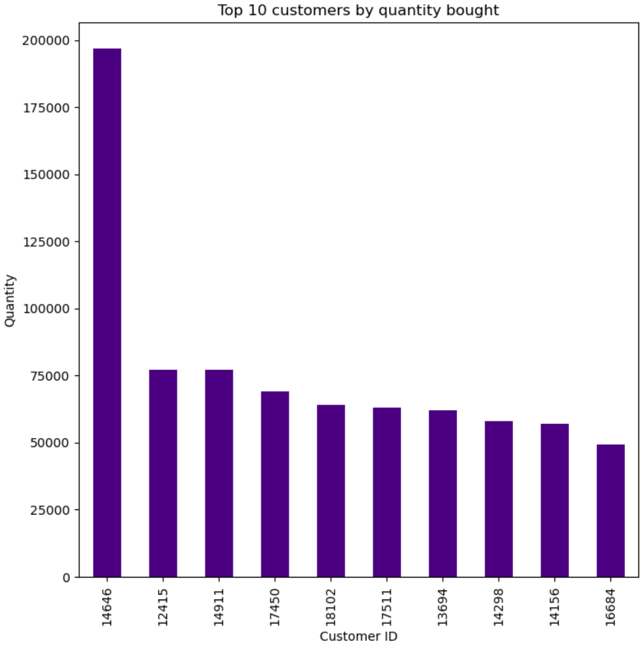


From the above bar graph, we can see in the year 2016, November has less than 5000 sales whereas in December it has more 20000 sales.



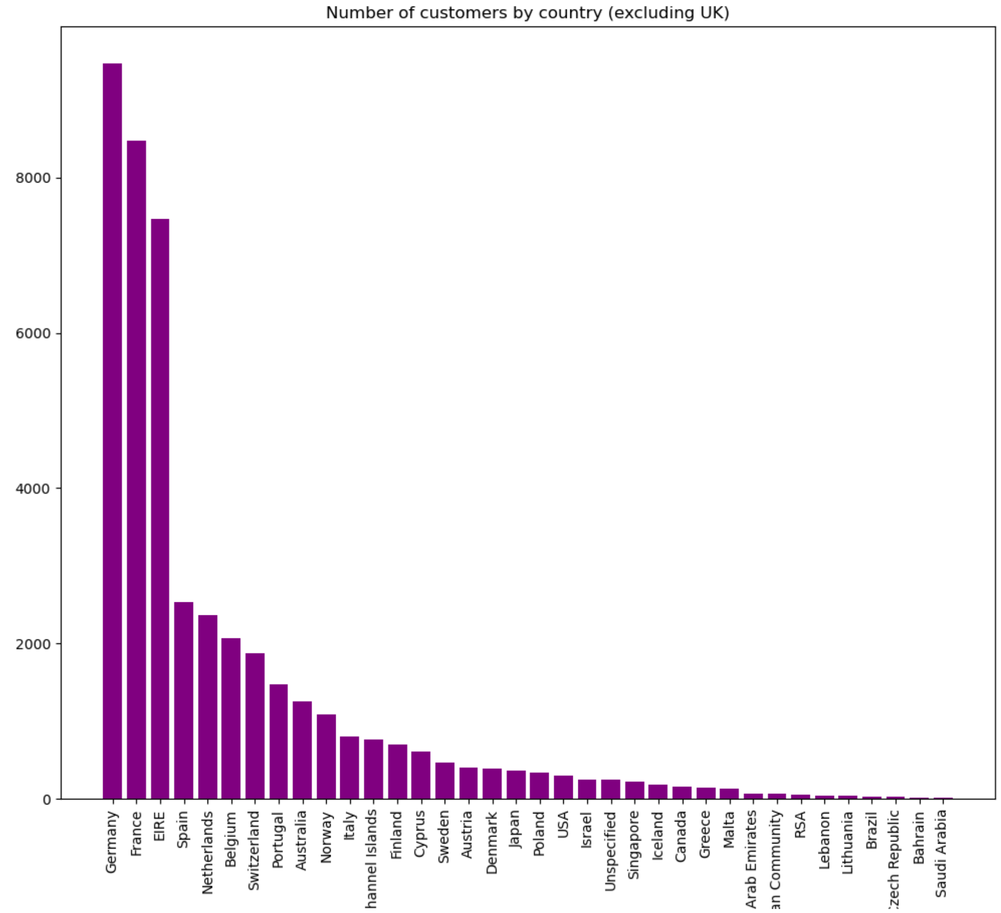
For the year 2017, The first six months sales fluctuate up and down not crossing more than 30k sales, but from July we can observe it starts increasing drastically till November which records highest number of sales of more than 60k, then suddenly sales drop in December to lowest of all.

- Top 10 customers after summing their purchase quantities



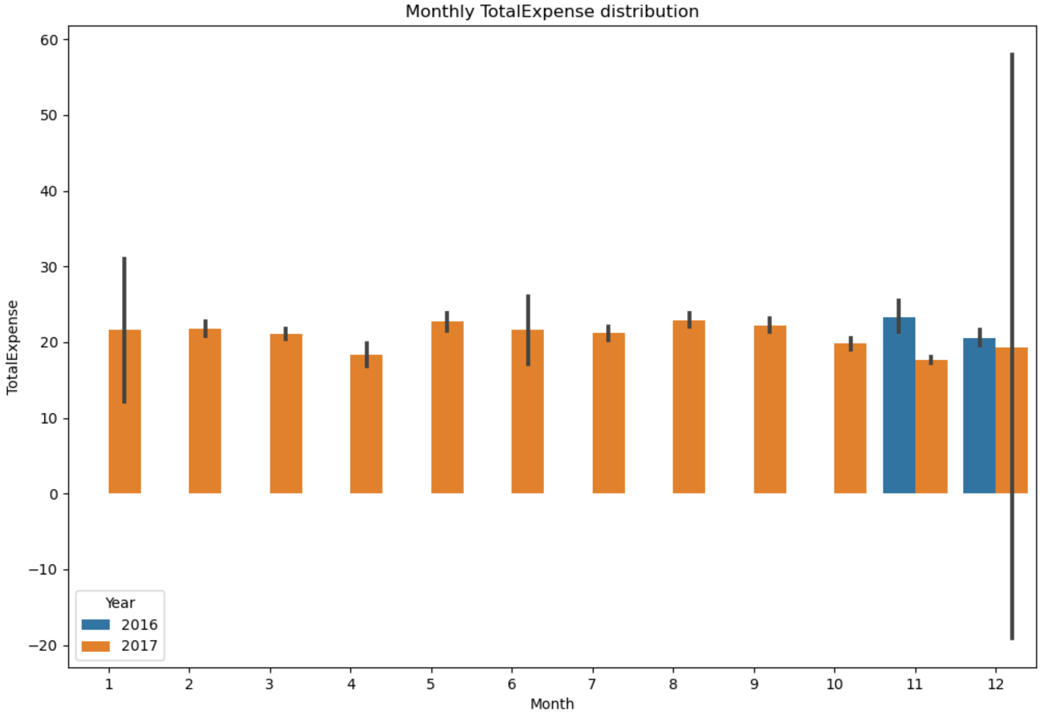
From the above plot, we have taken the top 10 customers with regards to the quantities they purchased. The person with customerID 14646 has bought the highest number of quantities, who can be considered as the highest valued customer.

- Country wise sales (except UK)



We can see in the above bar chart, Germany records second highest sales (9,477) (after UK) followed by France. We already know UK has the first highest sales (3,56,616) with major differences.

- Monthly TotalExpense distribution



7. Clustering

- Tasks to be performed
  - Use the clustering methodology to segment customers into groups
  - Clustering algorithm used: K-means
  - Identify the right number of customer segments (value of k) using elbow method
  - Identify which cluster groups the highly valued customers
- Performance Metrics

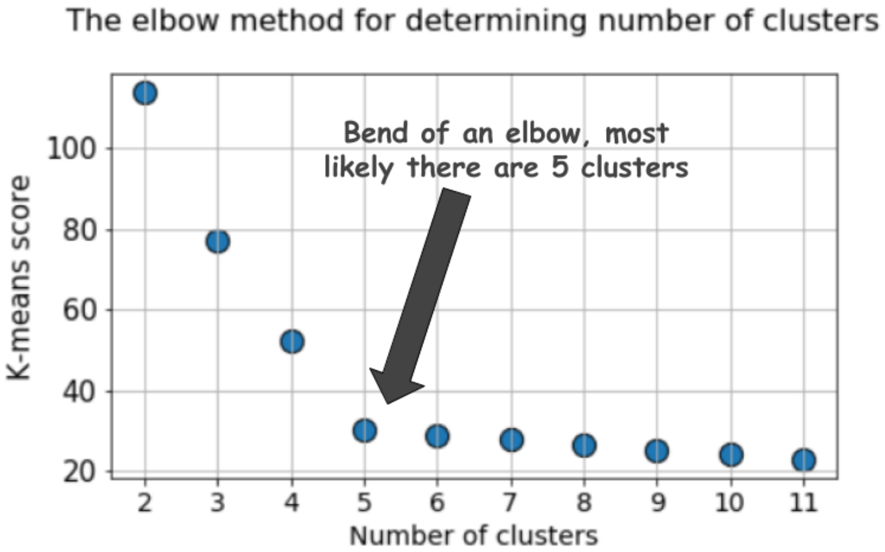
Clustering is often done for such analytics with the goal of market segmentation. It is, therefore, easily conceivable that, depending on the number of clusters, appropriate marketing personnel will be allocated to the problem. Consequently, a wrong assessment of the number of clusters can lead to sub-optimum allocation of precious resources.

- The Elbow Method

What is the score or metric which is being plotted for the elbow method? Why is it called the 'elbow' method?

For the k-means clustering method, the most common approach for answering this question is the so-called elbow method. It involves running the algorithm multiple times over a loop, with an increasing number of cluster choices and then plotting a clustering score as a function of the number of clusters.

A typical plot looks like the following.



The score is, in general, a measure of the input data on the k-means objective function i.e. some form of intra-cluster distance relative to inner-cluster distance. For example, in Scikit-learn's k-means estimator, a score method is readily available for this purpose.

- Silhouette coefficient

The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. The Silhouette Coefficient for a sample is (b - a) / max(a, b). To clarify, b is the distance between a sample and the nearest cluster that the sample is not a part of. We can compute the mean Silhouette Coefficient over all samples and use this as a metric to judge the number of clusters.

8. Results

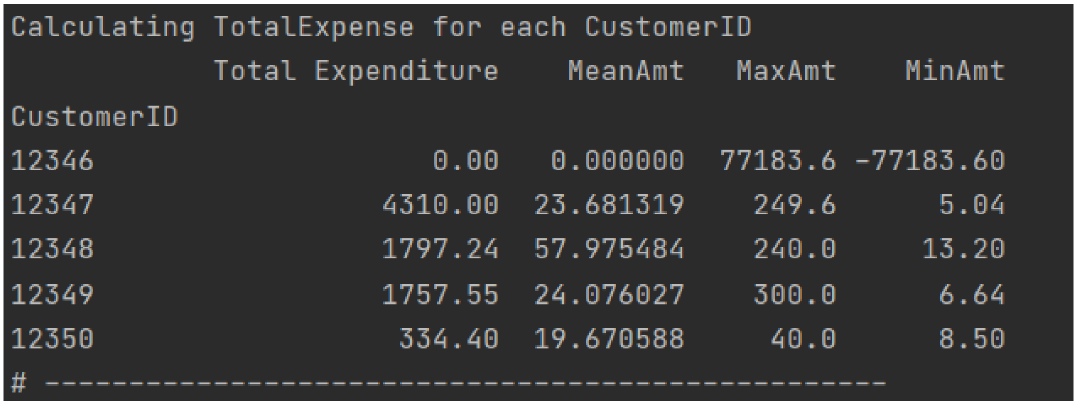
We decided to use the elbow method to determine the best k. But to run the kmean algorithm over a range of K values, we needed to find the best features to model the clustering.

- Features used for clustering

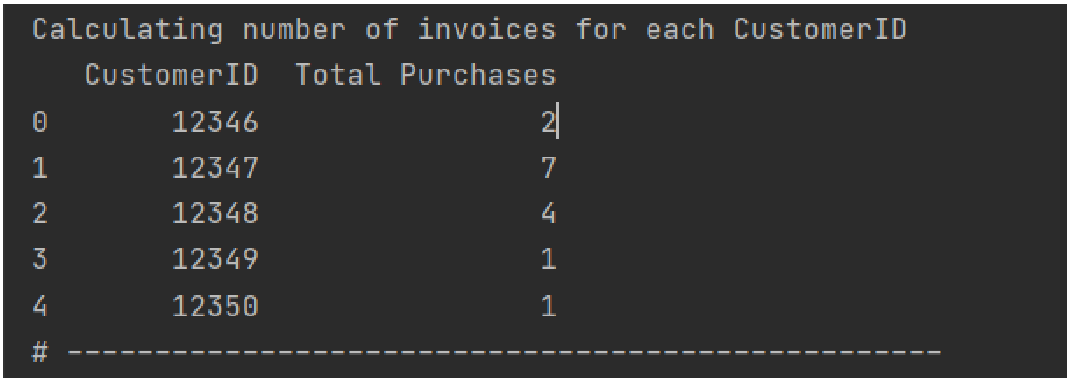
For each customer ID, we calculated:

1. TotalExpense
2. Number of purchases
3. Average amount per purchase
4. Number of unique items bought

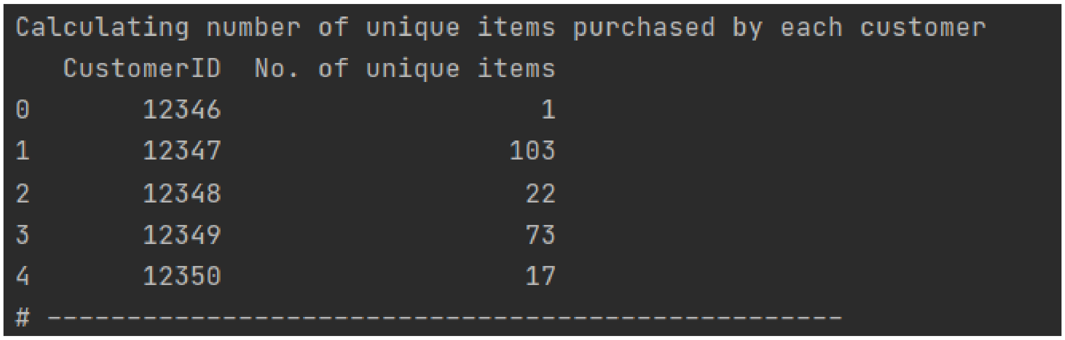
With the help of these metrics we will be able to categorize the customers into groups: High spending customers, regular (loyal) customers, etc.



```
Calculating TotalExpense for each CustomerID
            Total Expenditure    MeanAmt   MaxAmt    MinAmt
CustomerID
12346                    0.00   0.000000  77183.6 -77183.60
12347                 4310.00  23.681319    249.6      5.04
12348                 1797.24  57.975484    240.0     13.20
12349                 1757.55  24.076027    300.0      6.64
12350                  334.40  19.670588     40.0      8.50
# ---------------------------------------------------
```

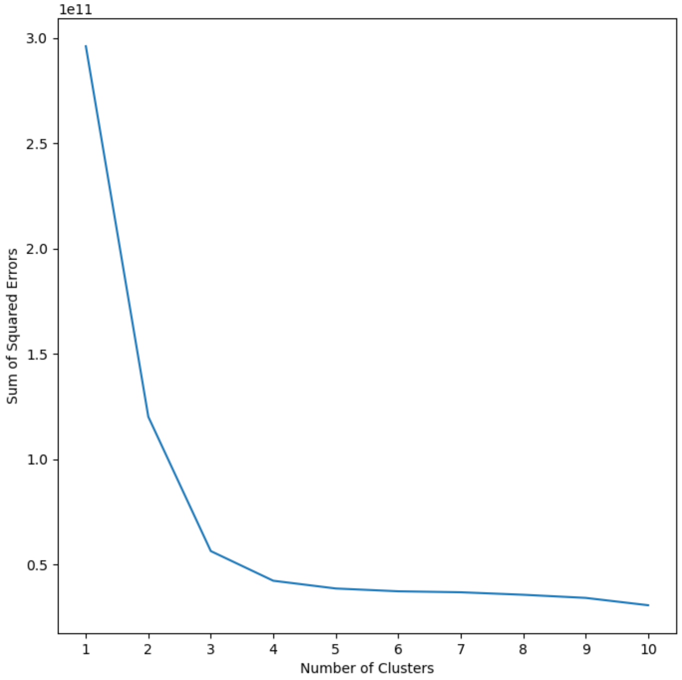


```
Calculating number of invoices for each CustomerID
   CustomerID  Total Purchases
0       12346                2
1       12347                7
2       12348                4
3       12349                1
4       12350                1
# -----------------------------------------------------
```

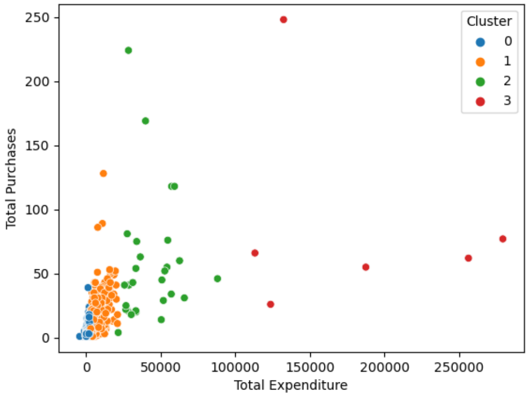


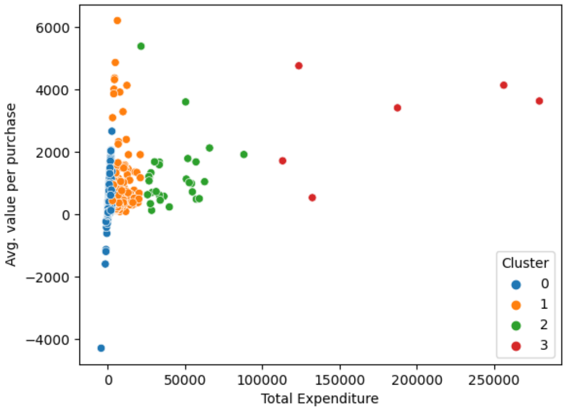
```
Calculating number of unique items purchased by each customer
   CustomerID  No. of unique items
0       12346                    1
1       12347                  103
2       12348                   22
3       12349                   73
4       12350                   17
# --------------------------------------------------
```

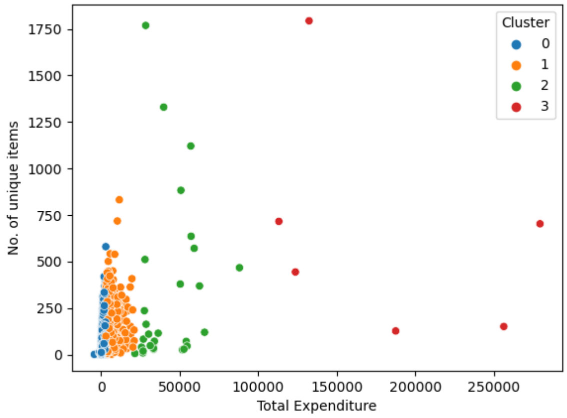
- Elbow method plot

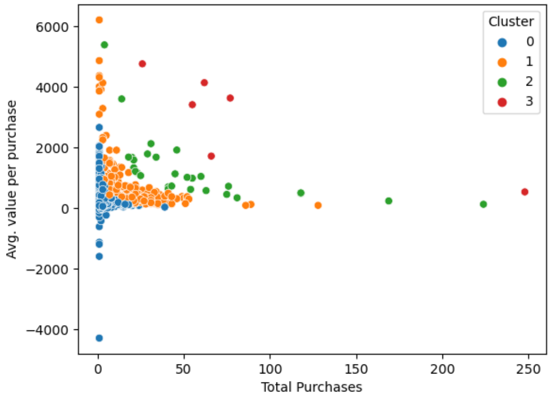


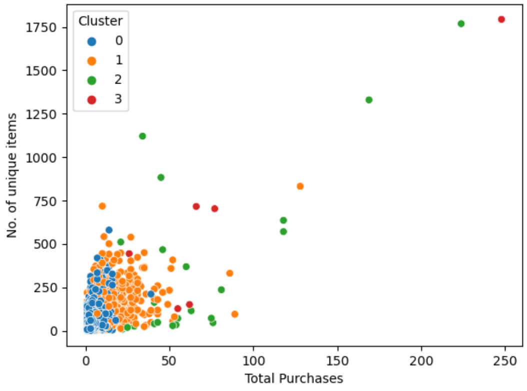
Using the Elbow method, we can decide k = 4 as an appropriate number of clusters.

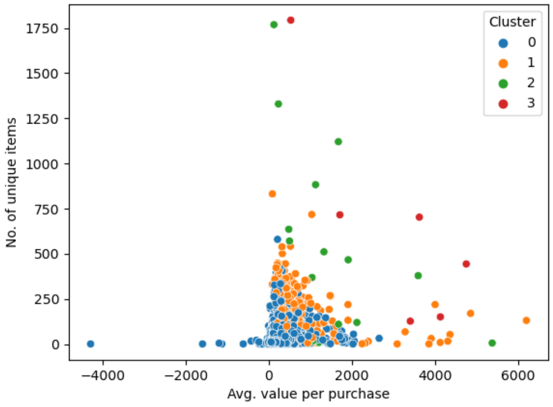












- Interpreting cluster graphs

From the scatter plots showing different clusters, we can conclude a few things and assign some characteristics to the clusters.

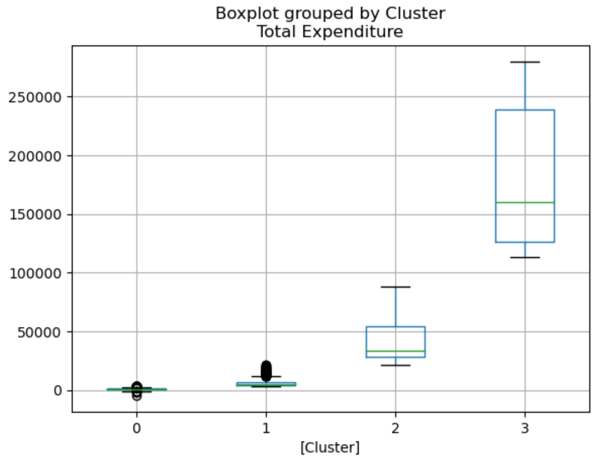
Cluster 0: Total expenditure is the least of all, total purchases are minimal and a minimal number of unique items are bought by them.

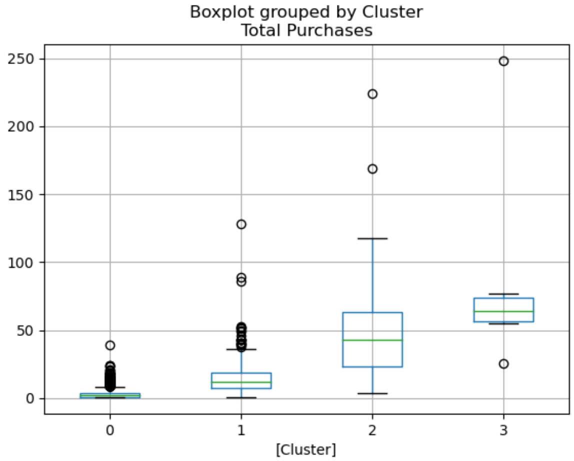
Cluster 1: Least total expenditure is observed. Total purchases are moderate. These customers have bought a large number of unique items.

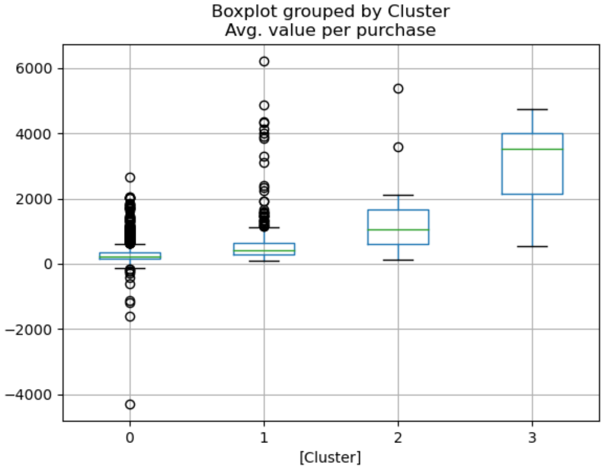
Cluster 2: Total expenditure is more than clusters 0 and 1, number of unique items bought is on a higher side as compared to the first two clusters.

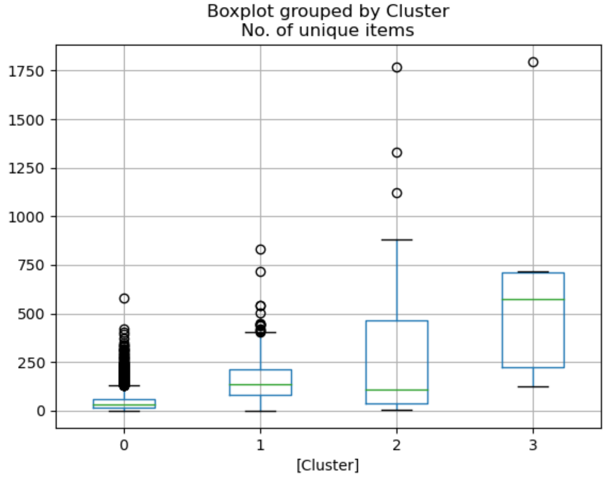
Cluster 3: Total expenditure is the maximum though the total purchases don't vary much.

We plotted boxplots to confirm our observations from the scatter plots.









9. Observations

Based on these results from the cluster feature graphs and the box plots, we can categorize the clusters as follows:

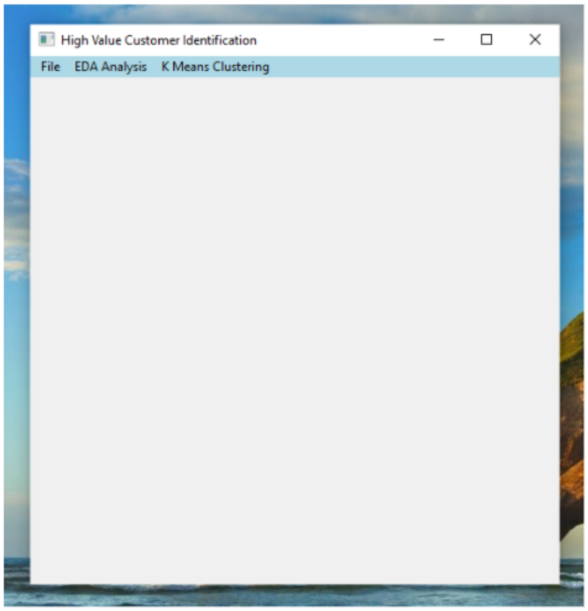
Cluster 0 and **1** need to be focussed **more** on, in terms of **discounts** and other **offers** in order to increase their purchase numbers.

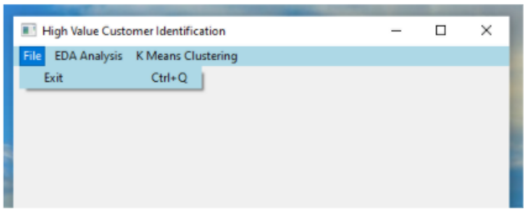
Cluster 2 consists of **regular** value but **loyal** customers who visit the store pretty often.

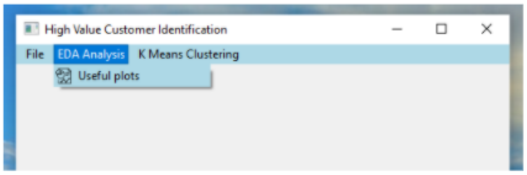
Cluster 3 can be considered as **high-valued customers** for whom a loyalty program should be rolled out. These customers are loyal as well seeing the number of purchases.

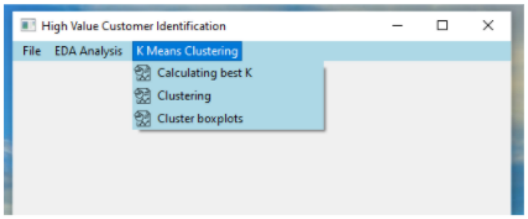
10. Graphical User Interface

The interface is made using PyQt library for Python. The main menu consists of three options: File, EDA Analysis and K Means clustering as follows:

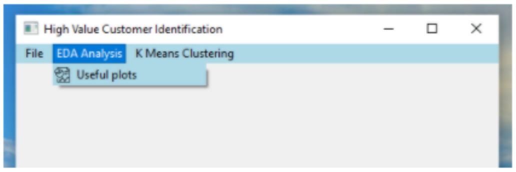


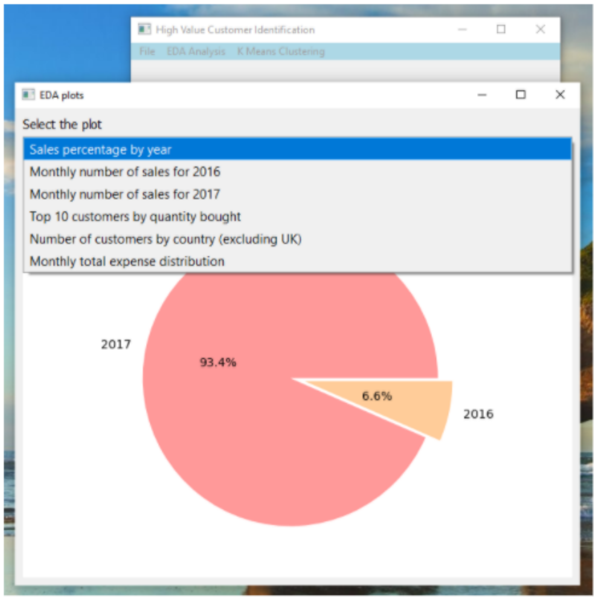




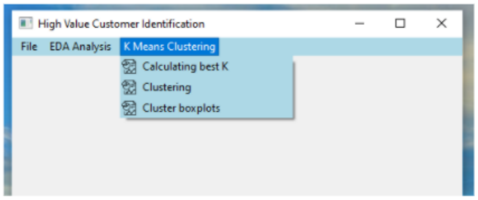


‘EDA Analysis’ button consists of **‘Useful plots’**, which when clicked on, opens a new window with a dropdown of all the graphs plotted in the EDA section.

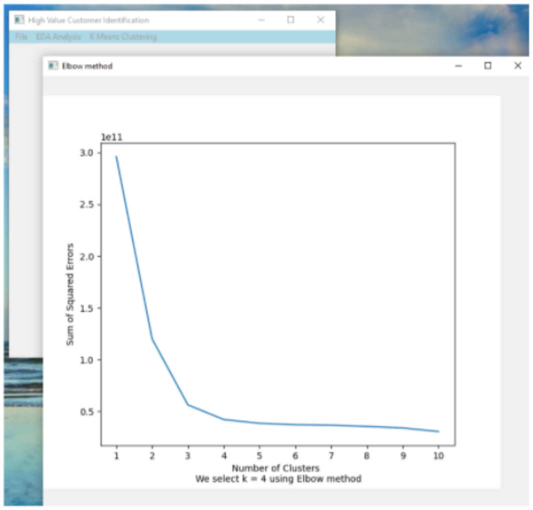



High Value Customer Identification
File
EDA Analysis
K Means Clustering
EDA plots
Select the plot
Sales percentage by year
Monthly number of sales for 2016
Monthly number of sales for 2017
Top 10 customers by quantity bought
Number of customers by country (excluding UK)
Monthly total expense distribution
2017
93.4%
6.6%
2016

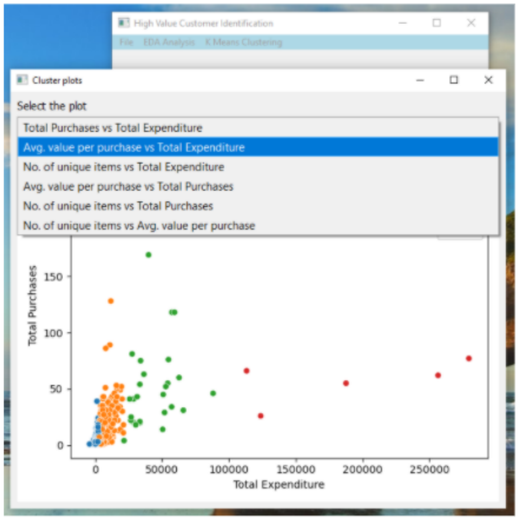
‘K Means Clustering’ button consists of **‘Calculating best K’, ‘Clustering’ and ‘Cluster boxplots’**.



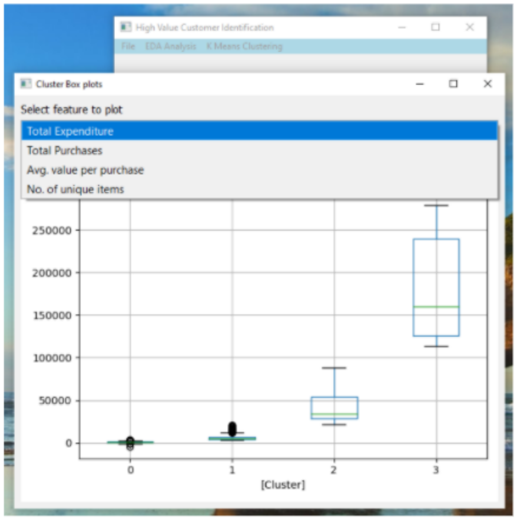
K Means Clustering -> Calculating best K



K Means Clustering -> Clustering


High Value Customer Identification
File
EDA Analysis
K Means Clustering
Cluster plots
Select the plot
Total Purchases vs Total Expenditure
Avg. value per purchase vs Total Expenditure
No. of unique items vs Total Expenditure
Avg. value per purchase vs Total Purchases
No. of unique items vs Total Purchases
No. of unique items vs Avg. value per purchase
Total Purchases
0
50
100
150
0
50000
100000
150000
200000
250000
Total Expenditure

K Means Clustering -> Cluster Boxplots



11. Summary and conclusions

Kmeans clustering is one of the most popular clustering algorithms and usually the first thing practitioners apply when solving clustering tasks to get an idea of the structure of the dataset. The goal of kmeans is to group data points into distinct non-overlapping subgroups. It does a very good job when the clusters have a kind of spherical shapes. However, it suffers as the geometric shapes of clusters deviates from spherical shapes. Moreover, it also doesn’t learn the number of clusters from the data and requires it to be pre-defined.

Kmeans method worked well for our dataset as we obtained some good promising results in the form of the clusters. After interpreting the clusters, we could categorize the customers in 4 different groups which was a success.

12. References

https://www.kaggle.com/vik2012kvs/high-value-customers-identification

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

https://towardsdatascience.com/clustering-metrics-better-than-the-elbow-method-6926e1f723a6

https://www.kdnuggets.com/2019/11/customer-segmentation-using-k-means-clustering.html

https://en.wikipedia.org/wiki/K-means_clustering

https://seaborn.pydata.org/

https://pandas.pydata.org/

https://matplotlib.org/

Appendices

Appendix A

K-Means CLustering

K-means clustering is one of the simplest unsupervised machine learning algorithms. Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes. A cluster refers to a collection of data points aggregated together because of certain similarities. Target number *k*, refers to the number of centroids you need in the dataset. A centroid is the imaginary or real location representing the center of the cluster.