Customer Segmentation of Online Retail Dataset

(Using Clustering Models)

COSC2670 Practical Data Science Project

(Assignment 2)

Submitted by: Vijeta Tulsiyan

Student ID: (s3398979)

Email:s3398979@student.rmit.edu.au

Date: 15.05.2018

1. Executive Summary

The main purpose of this project is to segment the customers based on their buying behaviour. The underlying dataset is a real dataset. Typically, E-commerce datasets are proprietary and consequently hard to find among publicly available data. However, The UCI Machine Learning Repository has made this dataset containing actual transactions from 2010 and 2011.

Customer segmentation process involves defining project's goal, understanding, preparing & exploring data, and data modelling.

The emphasis is on applying different clustering models on the dataset and then comparing their performance to select the one that is the best.

2. Introduction

The e-commerce data are actual transactional dataset which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based & registered non-store online retail. The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers. We found a customer with customer id 12346 with the highest total price 77,183.60 pounds for just one transaction (invoice no. 541431). This suggests wholesale transaction.

<u>Tool Used:</u>

I used Python as a tool for analysing and interpreting data. I worked on iPython notebook.

<u>Project Aim:</u>

To group or cluster the customers based on similar behaviours such as number of days to the most recent purchase, frequency of transactions and total monetary value of all the transactions.

<u>Scope of the project:</u>

The scope of the project is limited to analysing and modelling data for customers located in UK. The scope doesn't cover analysing cancelled orders, customer located in other countries, and description of items bought by the customers.

<u>Attributes Information:</u>

- InvoiceNo: Invoice number. Nominal, a 6-digit integral number uniquely assigned to each transaction.
- StockCode: Product (item) code. Nominal, a 5-digit integral number uniquely assigned to each distinct product. If this code starts with letter 'c', it indicates a cancellation.
- Description: Product (item) name. Nominal.

- Quantity: The quantities of each product (item) per transaction. Numeric.
- InvoiceDate: Invoice Date and time. Numeric, the day and time when each transaction was generated.
- UnitPrice: Unit price. Numeric, Product price per unit in sterling.
- CustomerID: Customer number. Nominal, a 5-digit integral number uniquely assigned to each customer
- Country: Country name. Nominal, the name of the country where each customer resides.

This is an unsupervised data where we don't have target variable. The purpose is to cluster the customers with similar attributes and add the respective cluster by adding a new column "Cluster" to the original dataset.

3. Methodology

Methology includes following steps:

A. Data Exploration and Preparation
B. Data Visualisation and Transformation
C. Data Modelling (KMeans, DBSCAN, Hierarchical)

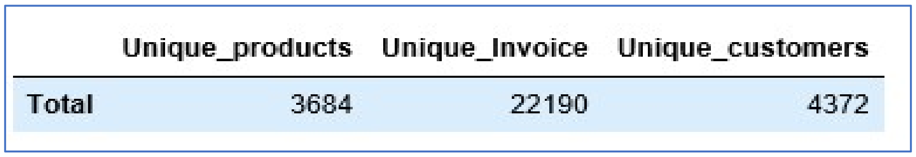
A. Data Exploration and Preparation:
The data downloaded from UCI Repository was in CSV format. The data was loaded in a dataframe and named 'Retail'. Appropriate python libraries were also loaded.

'Retail' dataframe originally had 541,909 rows and 8 variables. The issues with the raw data were:

- Missing data: Column 'Variable' had 1454 missing values and column 'CustomerID' had 135,080 missing values.
- Cancelled Orders: There were 9288 rows with 'InvoiceNo' beginning with letter 'C', negative value for 'Quantity' and missing 'CustomerID'.
- Unclear item Description: There were 82 rows with items 'Description' containing '?' and missing 'CustomerID'.

The above issues were resolved by removing missing observations/rows from the dataset. This alone cleaned the dataset. The dataset after removing missing data rows had 406,829 rows and 8 variables. We calculated the Total Price paid by multiplying 'UnitPrice' with 'Quantity' column and added a new column 'TotalPrice' to the Retail dataframe.

The number of unique levels at this stage in the dataset were:

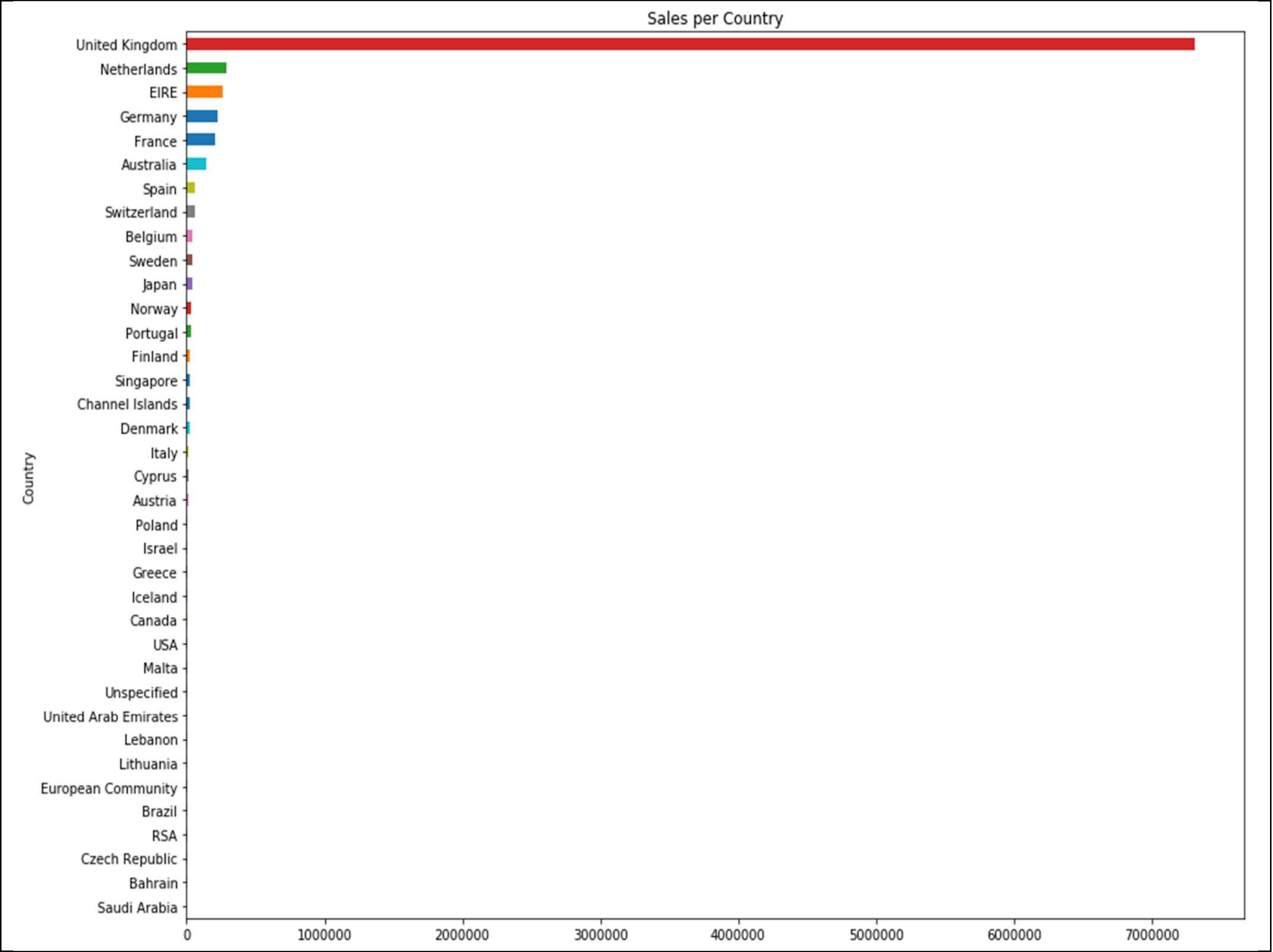


	Unique_products	Unique_Invoice	Unique_customers
Total	3684	22190	4372

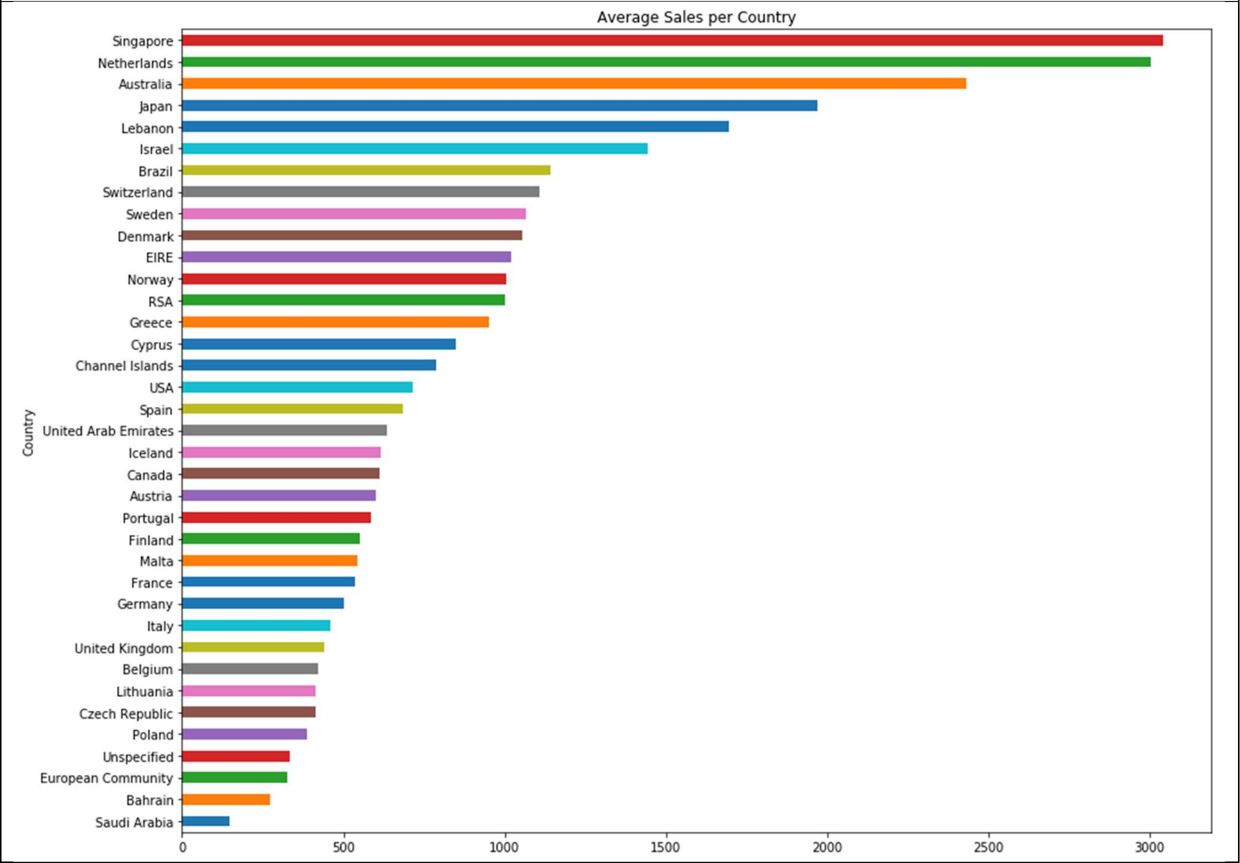
The cleaned dataset was visualized to check for Total sales (in pounds) grouped by countries. The graph showed that UK had highest total sales in pounds. This does not mean that other countries contributes insignificantly. High sales in UK can be attributed to few single wholesale transactions. To deal with this issue I visualized the average sales in pounds per transaction in each country and found a totally different story. Customers from 'Singapore', 'Netherland' and 'Australia' had the highest average sales. This shows that its important to segment the customers from each country separately.

I chose to analyse the behaviour of customers based in UK. There are 3921 unique customers in UK.

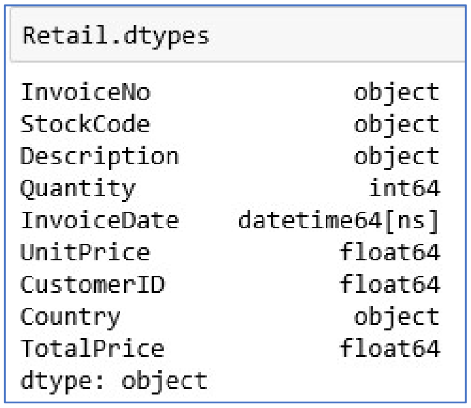
Total Sales per Country (in pounds)



Average Sales per Country (in pounds)



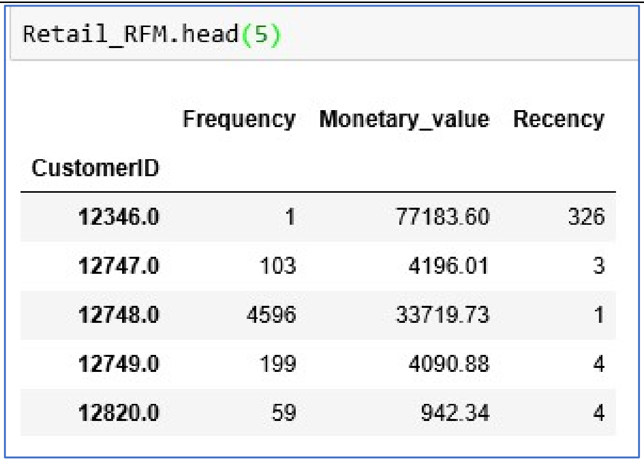
The dataset was filtered to select country UK. The resultant dataset had 354,345 rows and 9 variables. CustomerID is appearing as data type ‘float’. So, it was changed to type ‘object’.



```
Retail.dtypes

InvoiceNo               object
StockCode               object
Description             object
Quantity                 int64
InvoiceDate     datetime64[ns]
UnitPrice              float64
CustomerID             float64
Country                 object
TotalPrice             float64
dtype: object
```

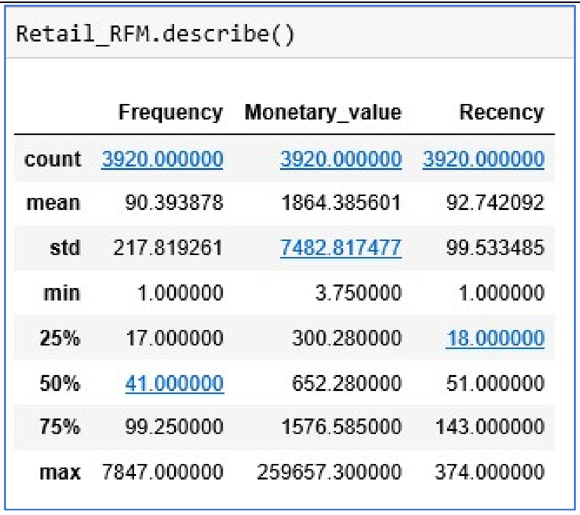
Retail data was further processed to calculate Frequency, Monetary Value and Recency grouped by customers. These 3 variables help to study the customers' behaviour in terms of how frequency they buy, how much they buy and when did they last buy. This will help us group the customers such as 'Very High Value' for those who buy much frequently and spend the most, etc.

The new dataframe called 'Retail_RFM' had 3921 rows and 3 columns namely 'Frequency', 'Monetary_value' and 'Recency'.



```
Retail_RFM.head(5)
```

CustomerID	Frequency	Monetary_value	Recency
12346.0	1	77183.60	326
12747.0	103	4196.01	3
12748.0	4596	33719.73	1
12749.0	199	4090.88	4
12820.0	59	942.34	4

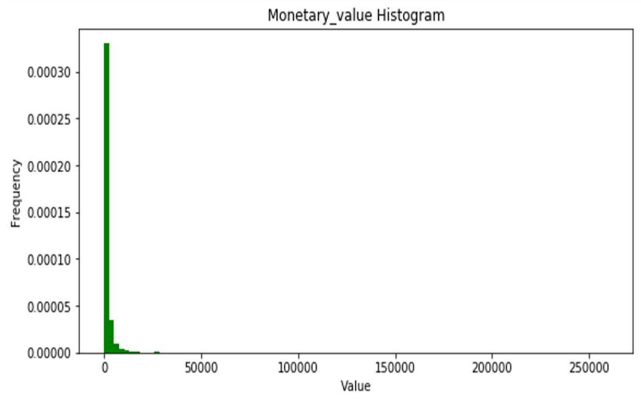


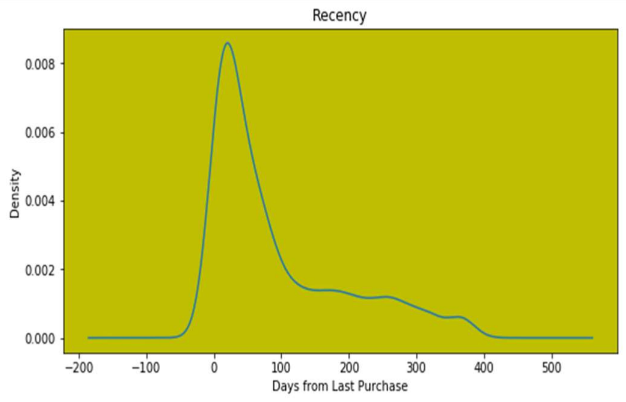
```
Retail_RFM.describe()
```

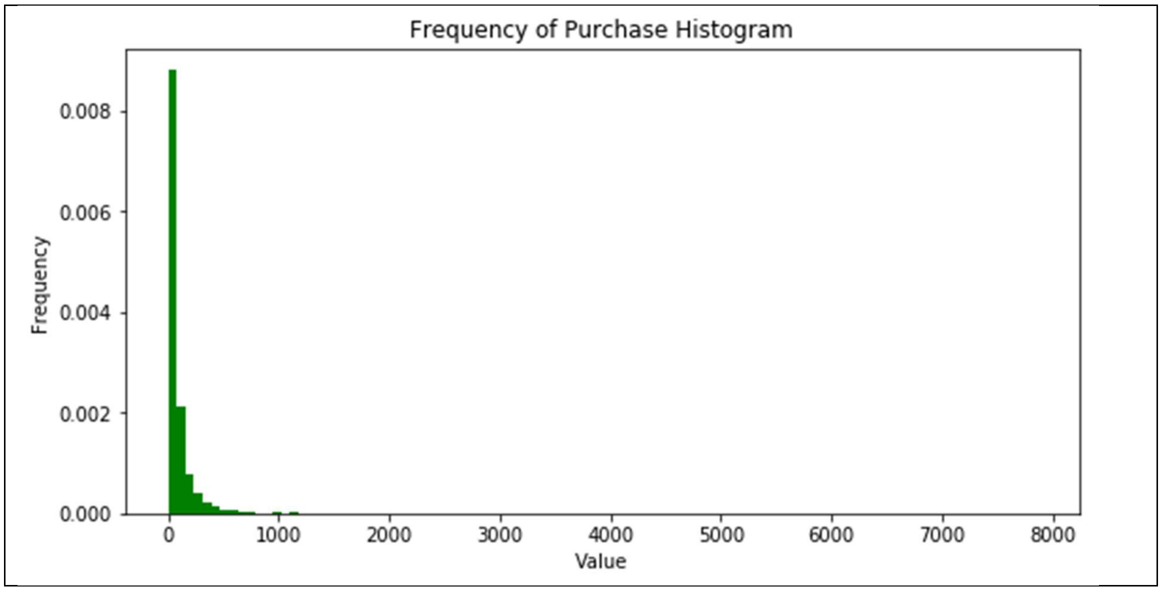
	Frequency	Monetary_value	Recency
count	3920.000000	3920.000000	3920.000000
mean	90.393878	1864.385601	92.742092
std	217.819261	7482.817477	99.533485
min	1.000000	3.750000	1.000000
25%	17.000000	300.280000	18.000000
50%	41.000000	652.280000	51.000000
75%	99.250000	1576.585000	143.000000
max	7847.000000	259657.300000	374.000000

B. <u>Data Visualisation and Transformation</u>

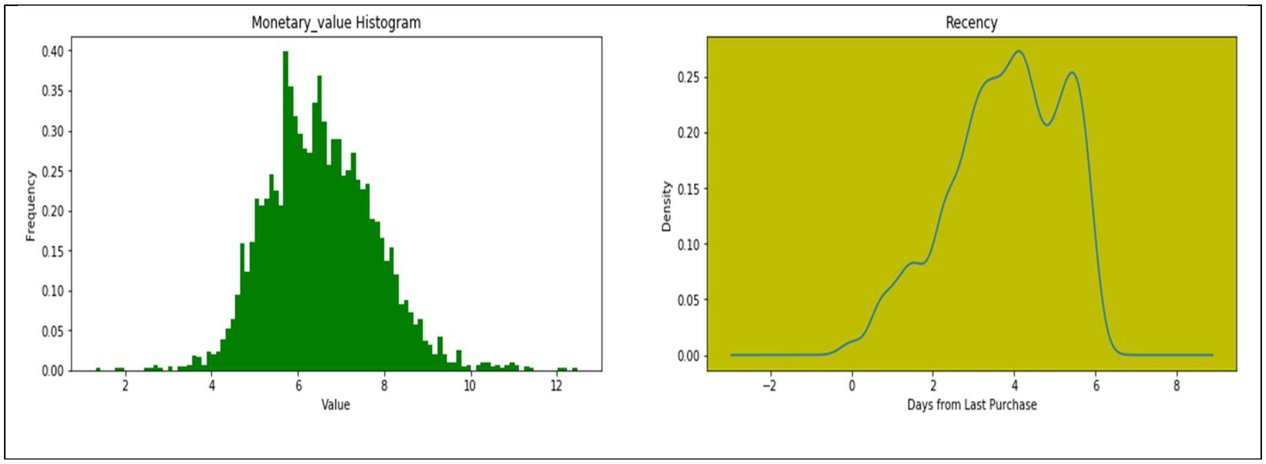
This step involves visualising each variable separately to study their distribution. It was found that the distribution is skewed to the right for all 3 variables Frequency, Monetary_value and Recency.

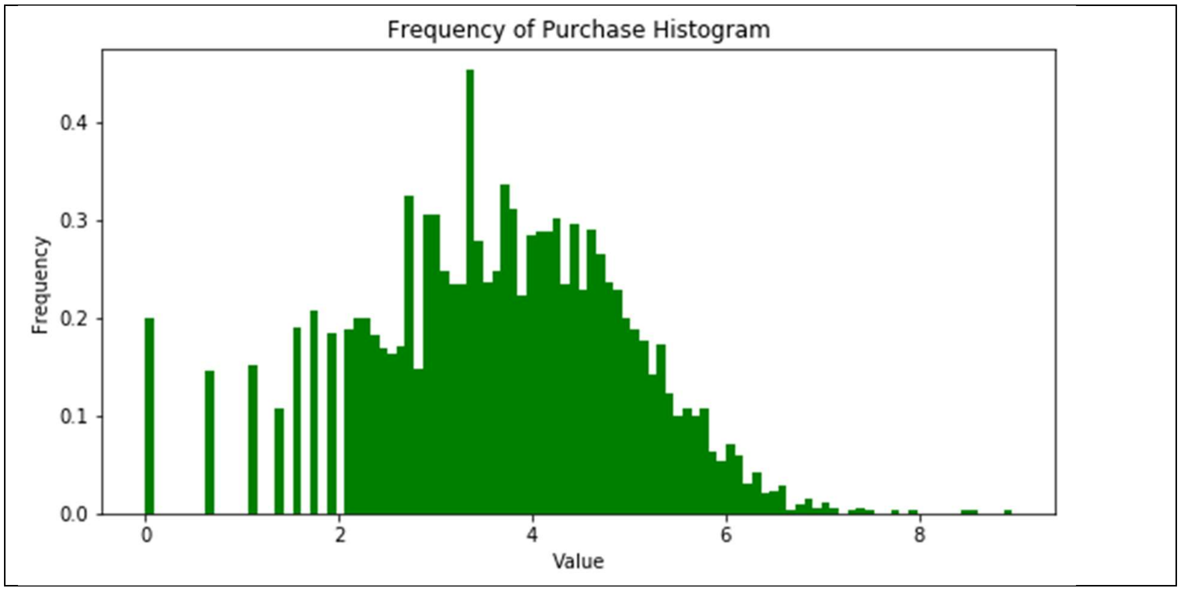


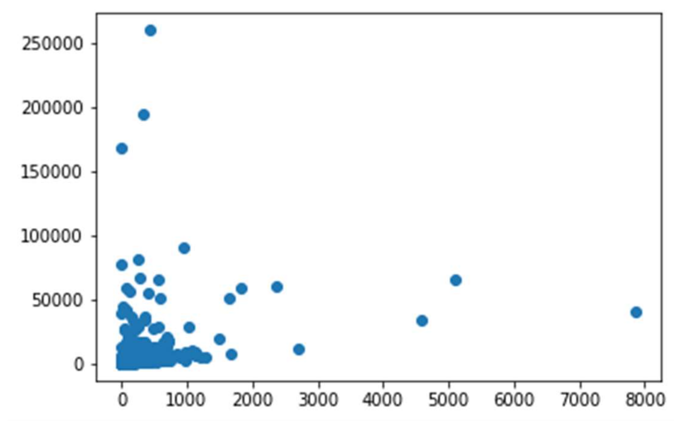
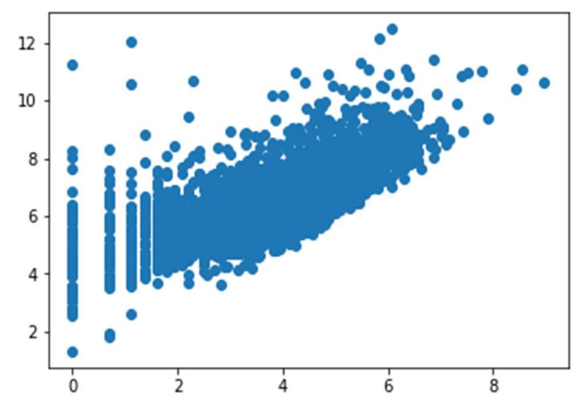




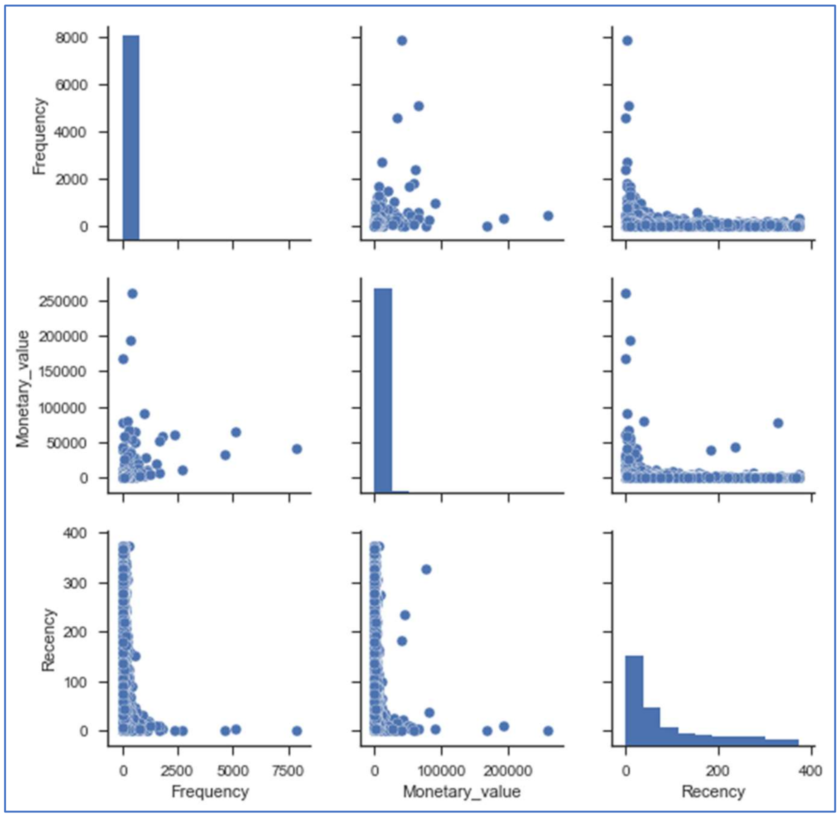
The purpose of this step is to prepare the data to apply Clustering Model. However, data is highly skewed and the scale of the variables are different. We apply log transform to normalize the data. The transformed dataframe has almost normal distribution. Below is the scatter plot of Frequency versus Monetary value before and after log transformation.



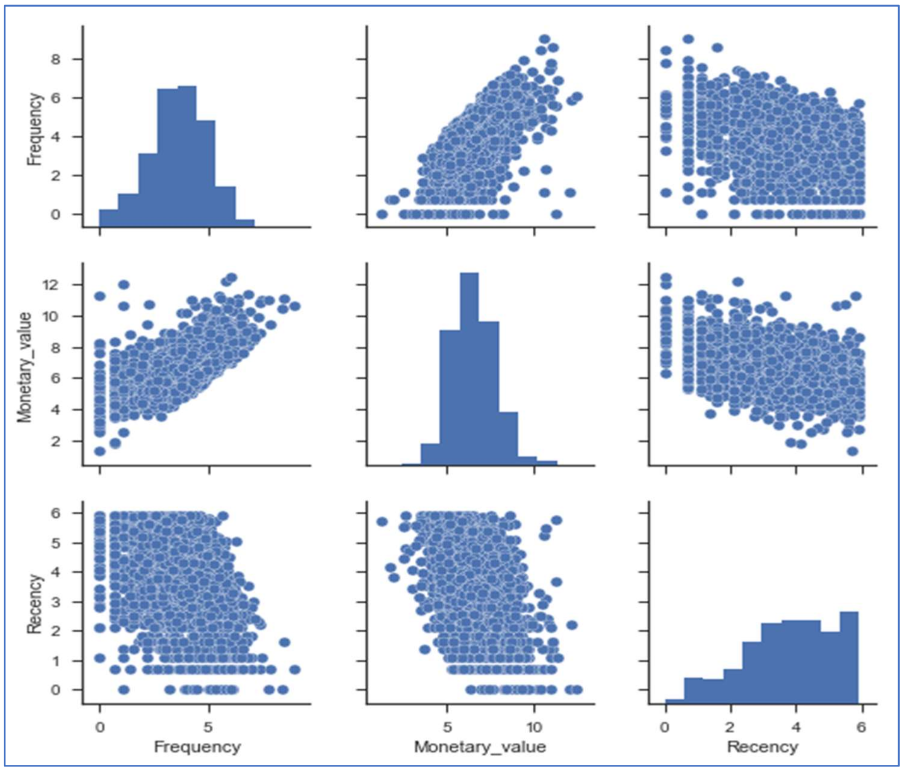


Before Log Transformation	After Log Transformation
	

Pair-plot before data transformation:



Pair-plot after data transformation:

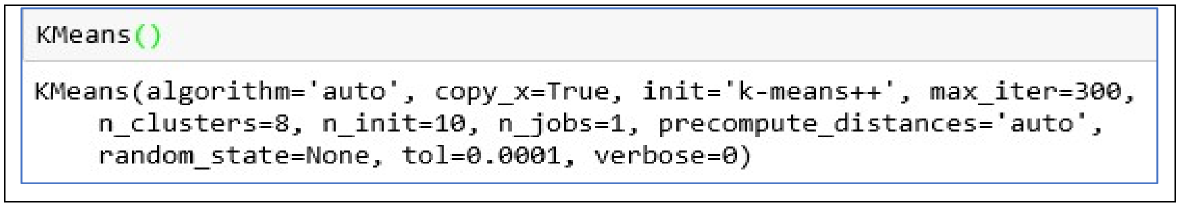


C. Data Modelling

KMeans() Clustering:

KMeans clustering was applied on the Retail recency, frequency and monetary dataframe names 'Retail_RFM' to find the different segments of customers differentiated by Frequency, Monetary_value and Recency.

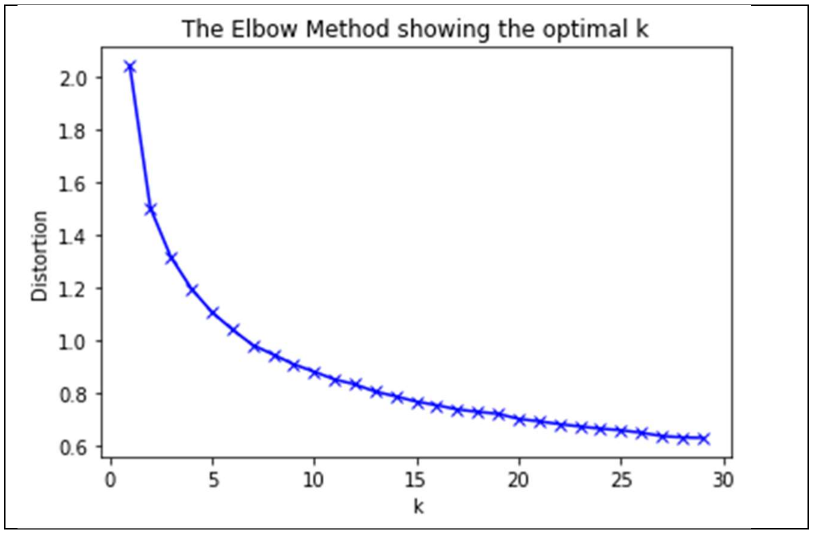
The default parameter values for the KMeans() algorithm are:



```
KMeans()
```

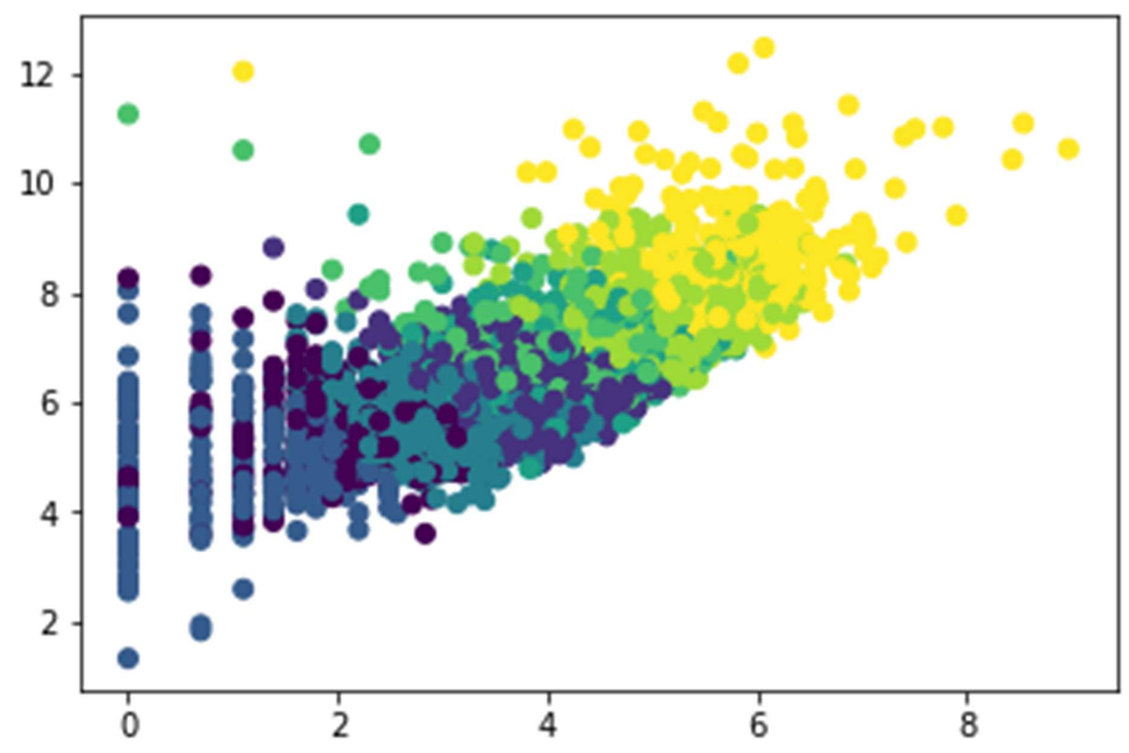
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=8, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

I needed to find the optimal value for clusters. The Elbow method was used to find the optimal value of n_clusters for the KMeans algorithm. **The below graph shows that the optimal value is 6.**

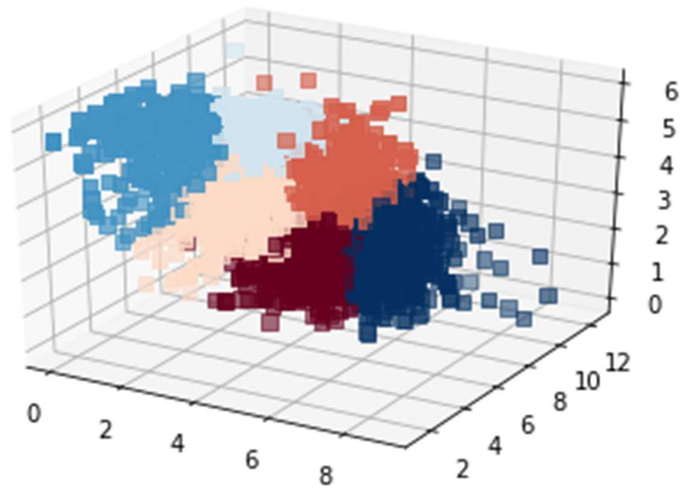


.

Scatterplot Frequency vs Monetary_value (KMeans)

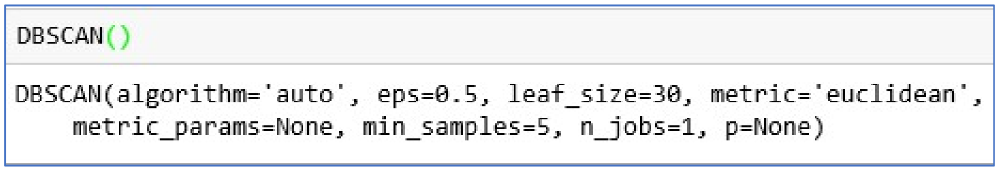


3D Plot for all 3 variables Frequency, Monetary_value and Recency for KMeans



DBSCAN:

Density-based spatial clustering of applications with noise (DBSCAN) groups together points that are close to each other based on a distance measurement (default Euclidean distance) and a minimum number of points.



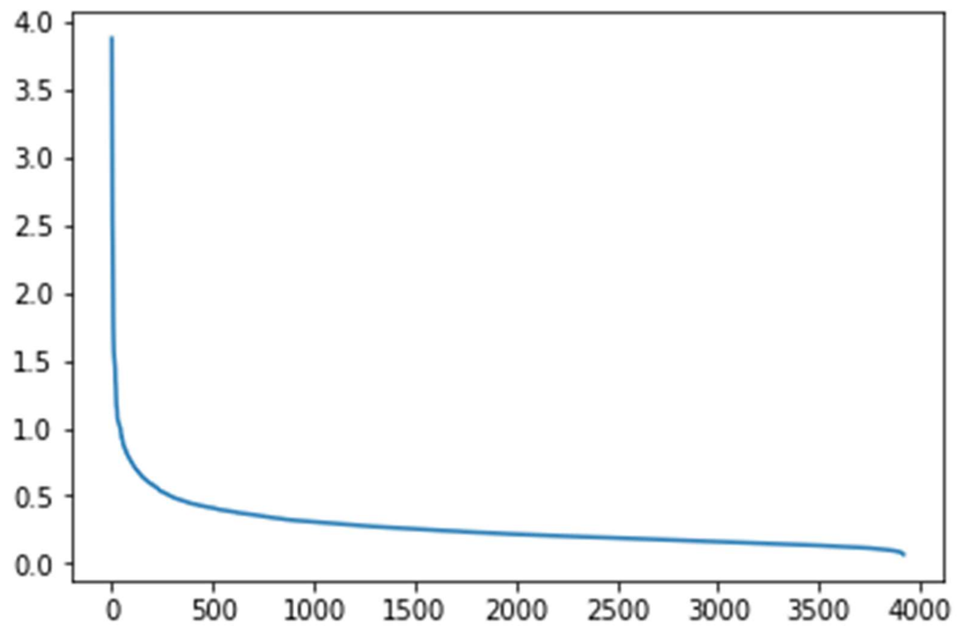
```
DBSCAN()
```

```
DBSCAN(algorithm='auto', eps=0.5, leaf_size=30, metric='euclidean',
    metric_params=None, min_samples=5, n_jobs=1, p=None)
```

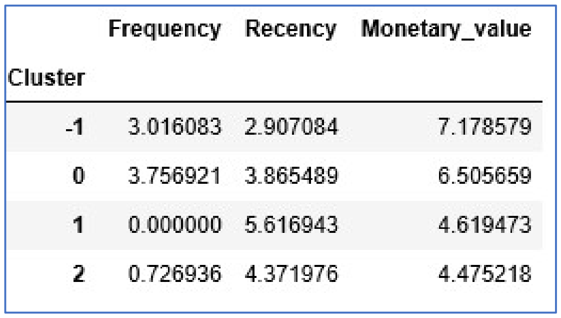
The two main parameters of the DBSCAN() algorithms are eps and min_samples.

I determined the eps using KNN (K nearest neighbour) algorithm. The plot using this was:

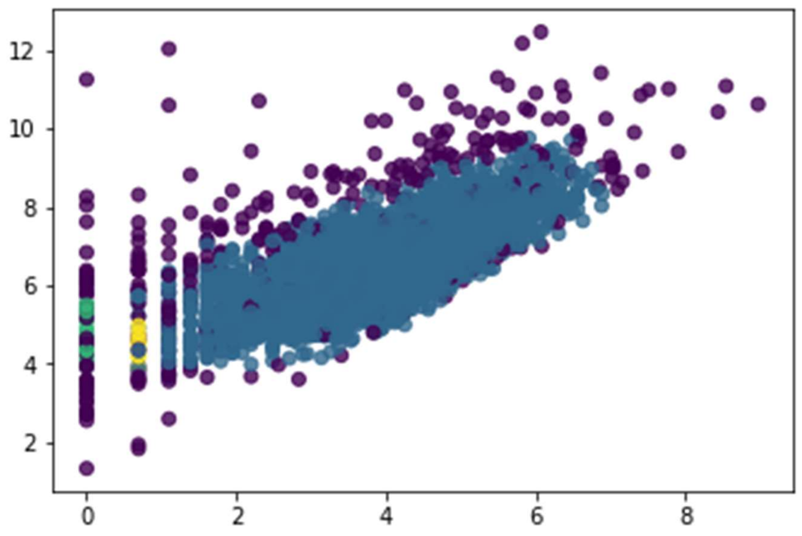
Plot using KNN to determine the value for Eps



The above graph suggests optimal eps = 0.5. I took min_samples = 10 and found 4 clusters as below:



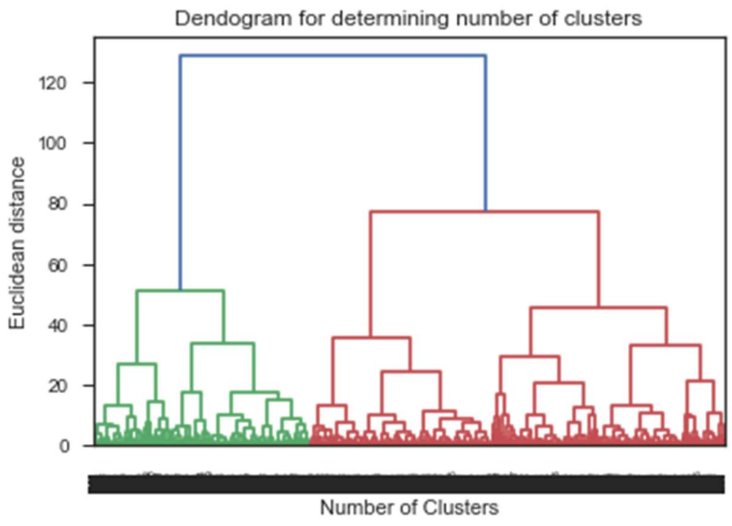
Cluster	Frequency	Recency	Monetary_value
-1	3.016083	2.907084	7.178579
0	3.756921	3.865489	6.505659
1	0.000000	5.616943	4.619473
2	0.726936	4.371976	4.475218



Hierarchical Clustering (HCA)

Agglomerative clustering was applied to the dataframe ‘Retail_RFM’. Compared to K-means, agglomerative algorithms are more cumbersome and do not scale well to large datasets. Agglomerative algorithms are more suitable for statistical studies.

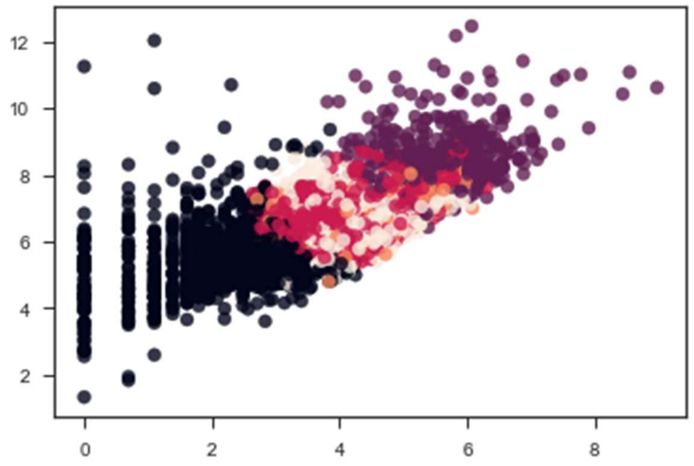
Before applying agglomerative hierarchical clustering, its important to plot dendrogram to determine the number of clusters.



From the above plot, number of clusters can be determined by drawing a horizontal line cutting the vertical lines at different levels of Euclidean distances. I tried at level 40, 40, 30 and visually determined number of clusters as 4, 5, 8.

```
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
```

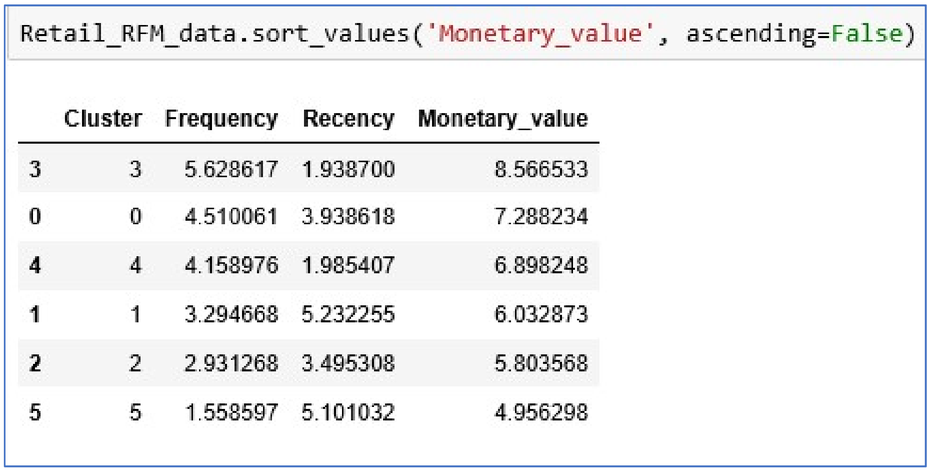
Plot of Frequency vs Monetary Value for HCA



4. Results

KMeans():

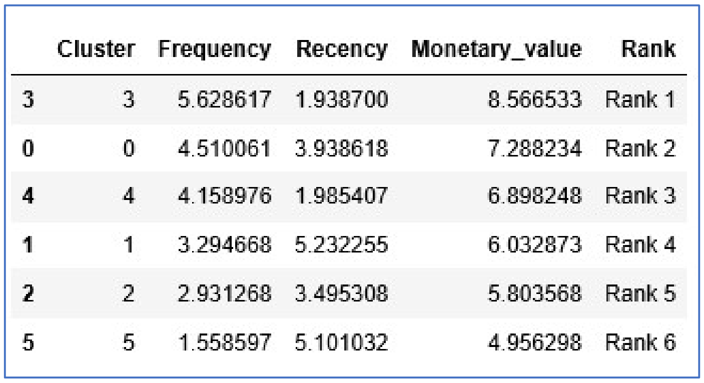
The table for the clusters sorted in descending order of Monetary value:



```
Retail_RFM_data.sort_values('Monetary_value', ascending=False)
```

	Cluster	Frequency	Recency	Monetary_value
3	3	5.628617	1.938700	8.566533
0	0	4.510061	3.938618	7.288234
4	4	4.158976	1.985407	6.898248
1	1	3.294668	5.232255	6.032873
2	2	2.931268	3.495308	5.803568
5	5	1.558597	5.101032	4.956298

The table shows that cluster 3 has highest frequency, highest monetary value and lowest recency, whereas, cluster 5 has lowest frequency, lowest monetary value and highest recency. We rank the clusters with cluster 3 as high valued customer with 'Rank 1' and cluster 5 as lowest value customer with "Rank 6".

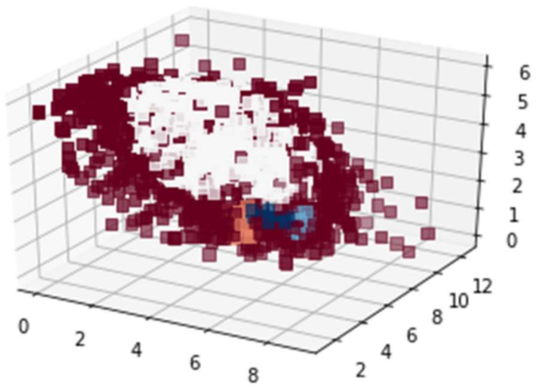


	Cluster	Frequency	Recency	Monetary_value	Rank
3	3	5.628617	1.938700	8.566533	Rank 1
0	0	4.510061	3.938618	7.288234	Rank 2
4	4	4.158976	1.985407	6.898248	Rank 3
1	1	3.294668	5.232255	6.032873	Rank 4
2	2	2.931268	3.495308	5.803568	Rank 5
5	5	1.558597	5.101032	4.956298	Rank 6

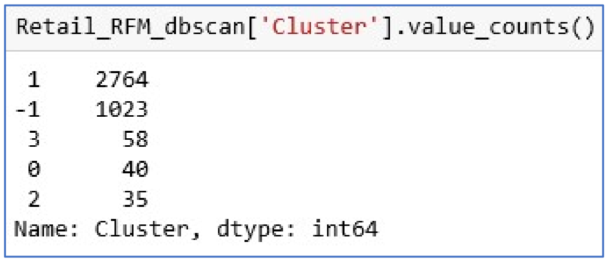
Lets suppose the marketing department are targeting very high valued customer and wanted to know of customer with CustomerID = 17850.0 is worth sending that offer. We can check the rank of the customer to suggest that it’s not “Rank 1” so doesn’t fit the offer.

DBSCAN():

Most of the customers are clustered in -1 and 1. Upon visualizing the scatter plot of monetary value and frequency, we found that the clusters are not very distinct. This is due to the fact DBSCAN forms clusters based on density and therefore doesn’t work well on this dataset.



Number of Customers Grouped in each cluster (1, -1, 3, 0, 2) using DBSCAN

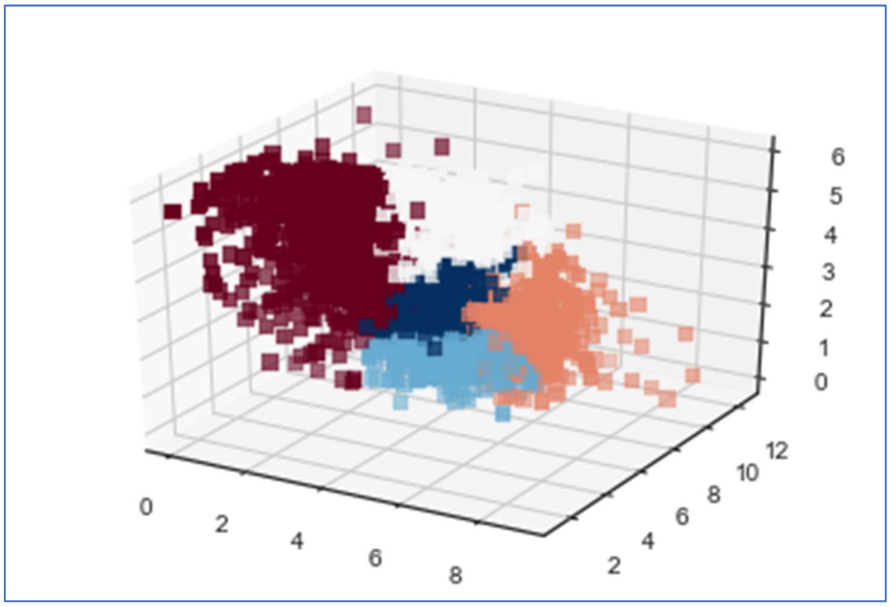


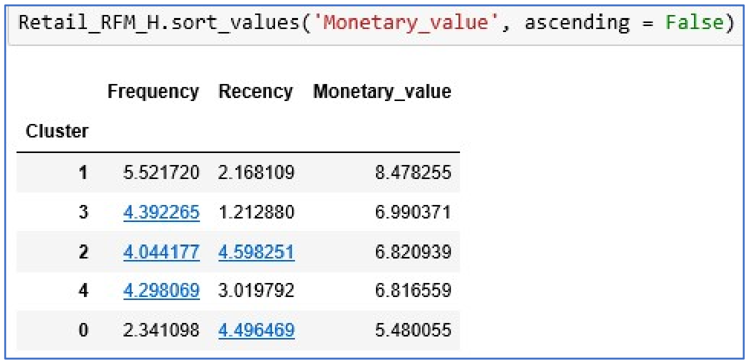
```
Retail_RFM_dbscan['Cluster'].value_counts()

 1     2764
-1     1023
 3       58
 0       40
 2       35
Name: Cluster, dtype: int64
```

Hierarchical Clustering (HCA):

I observed that 5 clusters worked well with the dataset and formed distinct clusters for the 3 variables frequency, monetary value and recency. But Agglomerative algorithm took longer time than KMeans and DBSCAN to compute the results.

3D Plot for all 3 variables Frequency, Monetary_value and Recency





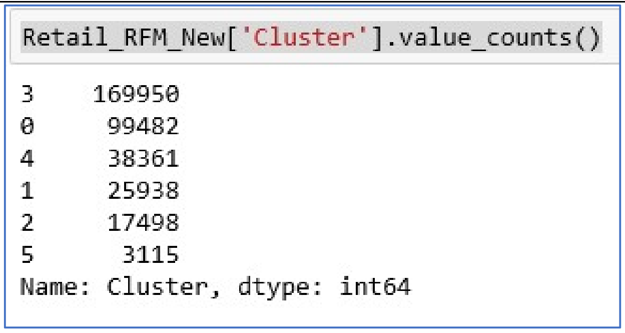
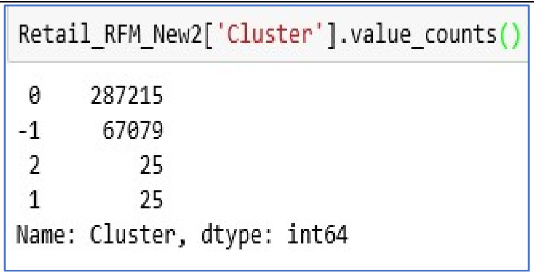
```
Retail_RFM_H.sort_values('Monetary_value', ascending = False)
```

Cluster	Frequency	Recency	Monetary_value
1	5.521720	2.168109	8.478255
3	4.392265	1.212880	6.990371
2	4.044177	4.598251	6.820939
4	4.298069	3.019792	6.816559
0	2.341098	4.496469	5.480055

5. Discussion

The results of KMeans model shows well defined clusters segmenting customers in the various 6 groups. This can be verified by randomly picking CustomerID and studying their buying behaviour and comparing it to its Rank.
The total number of rows counts in various clusters for KMeans and DBSCAN models are given in table below for comparison.

Number of Observations in each cluster from Retail data (KMeans)	**Number of Observations in each cluster from Retail data (DBSCAN**

 `Retail_RFM_New['Cluster'].value_counts()`
`3    169950`
`0     99482`
`4     38361`
`1     25938`
`2     17498`
`5      3115`
`Name: Cluster, dtype: int64`
	 `Retail_RFM_New2['Cluster'].value_counts()`
	` 0    287215`
	`-1     67079`
	` 2        25`
	` 1        25`
	`Name: Cluster, dtype: int64`

As evident from the above table, KMeans works very well. DBSCAN buckets most of the observations from Retail data in just 2 categories; this is not very helpful in determining the correct behaviour.

Also, compared to hierarchical clustering, KMeans clustering is computationally more efficient.

6. Conclusion

This is an unsupervised dataset, so in absence of target variable, the performance cannot be compared using confusion matrix. The clusters are ranked to classify the behaviours of customers in each cluster.

For this Ecommerce dataset, KMeans() clustering with k = 6 is the best model of the two. There is still lot of information that can be explored by carrying out similar analysis on other countries to group the customers. A study of the 'Description' of items in each cluster may provide more information on which items are most frequently bought. Also, cancelled orders can be analysed to find which items led to cancellation and in which customer category.

7. References

- https://archive.ics.uci.edu/ml/datasets/Online%20Retail
- RMIT University Lecture and Tute notes for course COSC2670
- https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a
- https://www.kaggle.com/carrie1/ecommerce-data