One University. One World. Yours.

Master of Science in Computing and Data Analytics

Data and Text Mining
MCDA 5580

Assignment 1

Submitted by:

Siddhartha Lahkar (A00430620)

Sidharth Bhalla (A00431562)

Divya Chainani (A00432519)

Submitted to:

Dr. Pawan Lingras

Trishla Shah

Table of Contents

Executive Summary

This report gives the detailed explanation of the data processing performed on the given dataset for finding out the customer purchasing behaviour. The data in consideration is the 'Online Retail Store' data, which contains the information of the transactions, products and customer of the concerned online store during the period of 2010-2011.

The attributes which give meaning to the data are as follows:

- CustomerID
- StockCode
- InvoiceNo
- Description
- Quantity
- InvoiceDate
- UnitPrice
- Country
- InvoiceDateTime

The K means clustering was performed on the data to find clusters for customers and products based on selected attributes from the online retail dataset. Upon getting the cluster a detailed analysis was done for profiling the segments. The profiling process included visualisation of clusters, creating rules as per the ideation, segregating clusters into different profiles. On the basis of these profiles achieved for products and customers, different recommendations were suggested. This whole process was to help the business owner get a clear vision of their business, about the market value, so that decision on the marketing strategy could be taken.

Objective

The scope of the assignment is to extract, cleanse and process the "OnlineRetail" data and apply k mean modelling on this data using R, for customer and product clustering. This would allow us to group customers and products based on their purchase and selling behaviors respectively. Post this, cluster profiling would be done which would clearly define each cluster based on behaviour of the attributes. For Example: Customers who visit often to the store, buy more products and generate more revenue would be called "Champions".

The profiling would help the business identify and make meaning of the clusters and come up with marketing strategies to improve their daily sales

Data Summary

The data considered for the assignment was the "Online Retail" data which provides transaction line level details for each customer, product and invoice number. The data specifies which customer bought which set of products and the sales date for that transaction and the Invoice number for the sale. *StockCode* and *Description* columns identify the products bought and the quantity and price of those products can be incurred from the *Quantity* and *UnitPrice* values. The *Country* column provides an idea of the geographic location of the customers and the products that are being bought.

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	InvoiceDateTime
0	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010,8:26	2.55	17850	United Kingdom	2010-12-01 08:26:00
536365	71053	WHITE METAL LANTERN	6	12/1/2010,8:26	3.39	17850	United Kingdom	2010-12-01 08:26:00
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010,8:26	2.75	17850	United Kingdom	2010-12-01 08:26:00
536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010,8:26	3.39	17850	United Kingdom	2010-12-01 08:26:00
536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010,8:26	3.39	17850	United Kingdom	2010-12-01 08:26:00
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010,8:26	7.65	17850	United Kingdom	2010-12-01 08:26:00
536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010,8:26	4.25	17850	United Kingdom	2010-12-01 08:26:00
536366	22633	HAND WARMER UNION JACK	6	12/1/2010,8:28	1.85	17850	United Kingdom	2010-12-01 08:28:00
536366	22632	HAND WARMER RED POLKA DOT	6	12/1/2010,8:28	1.85	17850	United Kingdom	2010-12-01 08:28:00
536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	12/1/2010,8:34	1.69	13047	United Kingdom	2010-12-01 08:34:00
536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	12/1/2010,8:34	2.10	13047	United Kingdom	2010-12-01 08:34:00
536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	12/1/2010,8:34	2.10	13047	United Kingdom	2010-12-01 08:34:00
536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	12/1/2010,8:34	3.75	13047	United Kingdom	2010-12-01 08:34:00
536367	22310	IVORY KNITTED MUG COSY	6	12/1/2010,8:34	1.65	13047	United Kingdom	2010-12-01 08:34:00
536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	6	12/1/2010,8:34	4.25	13047	United Kingdom	2010-12-01 08:34:00
536367	22623	BOX OF VINTAGE JIGSAW BLOCKS	3	12/1/2010,8:34	4.95	13047	United Kingdom	2010-12-01 08:34:00
536367	22622	BOX OF VINTAGE ALPHABET BLOCKS	2	12/1/2010,8:34	9.95	13047	United Kingdom	2010-12-01 08:34:00
536367	21754	HOME BUILDING BLOCK WORD	3	12/1/2010,8:34	5.95	13047	United Kingdom	2010-12-01 08:34:00
536367	21755	LOVE BUILDING BLOCK WORD	3	12/1/2010,8:34	5.95	13047	United Kingdom	2010-12-01 08:34:00
536367	21777	RECIPE BOX WITH METAL HEART	4	12/1/2010,8:34	7.95	13047	United Kingdom	2010-12-01 08:34:00
536367	48187	DOORMAT NEW ENGLAND	4	12/1/2010,8:34	7.95	13047	United Kingdom	2010-12-01 08:34:00
536368	22960	JAM MAKING SET WITH JARS	6	12/1/2010,8:34	4.25	13047	United Kingdom	2010-12-01 08:34:00
536368	22913	RED COAT RACK PARIS FASHION	3	12/1/2010,8:34	4.95	13047	United Kingdom	2010-12-01 08:34:00
536368	22912	YELLOW COAT RACK PARIS FASHION	3	12/1/2010,8:34	4.95	13047	United Kingdom	2010-12-01 08:34:00
536368	22914	BLUE COAT RACK PARIS FASHION	3	12/1/2010,8:34	4.95	13047	United Kingdom	2010-12-01 08:34:00

Figure 1: Online Retail dataset

Restrictions:

For analysis the dataset was limited to 2000 records and these records underwent the process of Data Cleaning

Observations:

The provided dataset contains a year data, from 2010-11.

During the initial analysis, it was seen that there are bad records with *CustomerID* = 0. This record does not follow the usual pattern of the *CustomerID*, further the revenue and the visits numbers for this Customer very high and may lead to forming of additional clusters with a single record which could hamper the overall segmentation.

For some records it was observed that the “Quantity” for some of the items was having negative value which states that the items were purchased in a certain time frame but eventually returned.

Another observation was that that the “Revenue” generated by the product and “PricePerInvoice” column were having value as ‘0’ which means that these products are not generating revenue .

Derived Dataset:

Customer:

The customer dataset was derived from the “OnlineRetail” dataset using a complex query, which performs set of operations on the concerned data and derives meaning full attributes.

```
SELECT CustomerID,
count(DISTINCT StockCode) as  NoOfDistinctProducts,
sum(Quantity) as NoOfProducts,
sum(UnitPrice*Quantity) as Revenue,
count(DISTINCT InvoiceNo) as Visits,
(sum(UnitPrice*Quantity)/count(DISTINCT InvoiceNo)) as AvgBasketValue,
(count(DISTINCT StockCode)/count(DISTINCT InvoiceNo)) as AvgBasketSize,
DATEDIFF('2011-12-11 00:00:00',max(InvoiceDateTime)) as NoOfDaysLastPurchased
FROM dataset04.OnlineRetail
GROUP By CustomerID
```

Figure 2: Initial query to derive customer dataset

The above query generated the below dataset

CustomerID	NoOfDistinctProducts	NoOfProducts	Revenue	Visits	AvgBasketValue	AvgBasketSize	NoOfDaysLastPurchased
0	3681	269562	1402889.59	3526	397.869991	1.0440	2
12346	1	0	0.00	2	0.000000	0.5000	327
12347	103	2458	4268.60	7	609.800000	14.7143	4
12348	22	2341	1527.15	4	381.787500	5.5000	77
12349	73	631	1406.95	1	1406.950000	73.0000	20
12350	17	197	304.39	1	304.390000	17.0000	312
12352	59	470	1321.54	9	146.837778	6.5556	38
12353	4	20	89.00	1	89.000000	4.0000	206
12354	58	530	1052.60	1	1052.600000	58.0000	234
12355	13	240	453.88	1	453.880000	13.0000	216
12356	53	1591	2624.77	3	874.923333	17.6667	24
12357	131	2708	6125.14	1	6125.140000	131.0000	35
12358	13	248	988.00	2	494.000000	6.5000	3
12359	214	1612	4855.22	5	971.044000	42.8000	9
12360	105	1165	2260.93	3	753.643333	35.0000	54
12361	10	91	184.89	1	184.890000	10.0000	289
12362	201	2212	4879.94	11	443.630909	18.2727	5
12363	23	408	552.00	2	276.000000	11.5000	111
12364	70	1506	1276.11	4	319.027500	17.5000	9
12365	22	173	317.93	3	105.976667	7.3333	293
12367	11	173	160.89	1	160.890000	11.0000	6
12370	143	2353	3455.66	4	863.915000	35.7500	53
12371	63	591	1606.83	2	803.415000	31.5000	46
12372	34	794	1255.98	3	418.660000	11.3333	73
12373	14	197	334.59	1	334.590000	14.0000	313

Figure 3: Initial customer dataset before cleaning

Based on the initial analysis and observation, the query was updated as below to remove the record with *CustomerID*=0.

```
SELECT CustomerID,
count(DISTINCT StockCode) as  NoOfDistinctProducts,
sum(Quantity) as NoOfProducts,
sum(UnitPrice*Quantity) as Revenue,
count(DISTINCT InvoiceNo) as Visits,
(sum(UnitPrice*Quantity)/count(DISTINCT InvoiceNo)) as AvgBasketValue,
(count(DISTINCT StockCode)/count(DISTINCT InvoiceNo)) as AvgBasketSize,
DATEDIFF('2011-12-11 00:00:00',max(InvoiceDateTime)) as NoOfDaysLastPurchased
FROM dataset04.OnlineRetail
GROUP By CustomerID having CustomerID <> 0
```

Figure 4: Query to derive clean customer data

The below figure shows a part of the customer dataset, containing the said attributes.

	CustomerID	NoOfDistinctProducts	NoOfProducts	Revenue	Visits	AvgBasketValue	AvgBasketSize	Recency
1	12346	1	0	0.00	2	0.0000	0.5000	327
2	12347	103	2458	4268.60	7	609.8000	14.7143	4
3	12348	22	2341	1527.15	4	381.7875	5.5000	77
4	12349	73	631	1406.95	1	1406.9500	73.0000	20
5	12350	17	197	304.39	1	304.3900	17.0000	312
6	12352	59	470	1321.54	9	146.8378	6.5556	38
7	12353	4	20	89.00	1	89.0000	4.0000	206
8	12354	58	530	1052.60	1	1052.6000	58.0000	234
9	12355	13	240	453.88	1	453.8800	13.0000	216
10	12356	53	1591	2624.77	3	874.9233	17.6667	24
11	12357	131	2708	6125.14	1	6125.1400	131.0000	35
12	12358	13	248	988.00	2	494.0000	6.5000	3
13	12359	214	1612	4855.22	5	971.0440	42.8000	9
14	12360	105	1165	2260.93	3	753.6433	35.0000	54
15	12361	10	91	184.89	1	184.8900	10.0000	289
16	12362	201	2212	4879.94	11	443.6309	18.2727	5
17	12363	23	408	552.00	2	276.0000	11.5000	111
18	12364	70	1506	1276.11	4	319.0275	17.5000	9
19	12365	22	173	317.93	3	105.9767	7.3333	293
20	12367	11	173	160.89	1	160.8900	11.0000	6
21	12370	143	2353	3455.66	4	863.9150	35.7500	53
22	12371	63	591	1606.83	2	803.4150	31.5000	46
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Showing 1 to 23 of 2,000 entries

Figure 5: Metadata of cleaned Customer dataset

Further, the data also consists of some customer records with negative values for revenue and number of products bought. Below figure shows such *CustomerID's*

CustomerID	NoOfDistinctPrc	NoOfProducts	Revenue	Visits	AvgBasketValue	AvgBasketSize	NoOfDays Last Purchased
12503	1	-1	-9.99	1	-9.99	1	339
12505	1	-1	-4.5	1	-4.5	1	303
12605	3	-4	-7.5	1	-7.5	3	367
12666	2	-56	-227.44	1	-227.44	2	361
12870	2	-2	-14.9	1	-14.9	2	368
12943	1	-1	-3.75	1	-3.75	1	303
13154	1	-1	-9.99	1	-9.99	1	146
13672	5	-1	-9.99	3	-3.33	1.6667	303
13693	4	-6	-32	1	-32	4	327
13829	1	-12	-102	1	-102	1	361
13958	5	-23	-94.17	1	-94.17	5	374
14119	1	-2	-19.9	1	-19.9	1	356
14213	5	-244	-1192.2	1	-1192.2	5	373
14627	5	-5	-21.85	1	-21.85	5	313
14679	1	-1	-2.55	1	-2.55	1	373
14777	2	-9	-17.45	1	-17.45	2	6

Figure 6: CustomerID with negative Revenue and No of products bought

While these customers might be valid and might have returned the products they have purchased, they do not add any value. Thus, these records were also removed from the customer dataset.

<u>Product:</u>

The product dataset was derived from the “OnlineRetail” dataset using a complex query, which performs set of operations on the concerned data and derives meaning full attributes.

```sql
SELECT StockCode,
count(DISTINCT CustomerID) as Customers,
sum(UnitPrice*Quantity) as Revenue
,count(InvoiceNo) as Visits,
(sum(UnitPrice*Quantity)/count(DISTINCT InvoiceNo)) as PricePerinvoice,
count(DISTINCT(Country)) as TotalCountries,
sum(Quantity)/count(DISTINCT InvoiceNo) as UnitsSoldPerInvoice
FROM dataset04.`OnlineRetail` GROUP BY StockCode LIMIT 2000;

```

Figure 7: Query to derive product dataset

The below figure shows a part of the product dataset, containing the said attributes. Also , based on the observations above, the records having “Revenue” and “PricePerInvoice” value as ‘0’ was considered noisy data (Figure 2) thus, removed as a part of data cleaning process (Figure 3).

	StockCode	Customers	Revenue	Visits	PricePerInvoice	TotalCountries	UnitsSoldPerInvoice
730	21645	2	-39.60	2	-19.800000	1	-12.5000
163	20703	3	-25.50	4	-12.750000	2	-3.0000
289	20957	1	-1.45	1	-1.450000	1	-1.0000
587	21412	2	-2.52	2	-1.260000	2	-6.0000
990	22034	3	-0.02	9	-0.002222	2	-26.2222
5	10123G	1	0.00	1	0.000000	1	-38.0000
10	10134	1	0.00	1	0.000000	1	-19.0000
43	16053	1	0.00	1	0.000000	1	-102.0000
82	17011A	1	0.00	1	0.000000	1	-61.0000
153	20689	1	0.00	1	0.000000	1	-5.0000
184	20738	1	0.00	1	0.000000	1	-36.0000
232	20825	1	0.00	1	0.000000	1	-5.0000
250	20849	1	0.00	1	0.000000	1	1.0000
251	20850	1	0.00	1	0.000000	1	-3.0000
258	20863	1	0.00	1	0.000000	1	-6.0000
259	20864	1	0.00	1	0.000000	1	-1.0000
270	20896	1	0.00	1	0.000000	1	-5.0000
286	20950	1	0.00	1	0.000000	1	1.0000
397	21134	1	0.00	1	0.000000	1	1.0000

Figure 8: Dataset containing noisy data

	StockCode	Customers	Revenue	Visits	PricePerInvoice	TotalCountries	UnitsSoldPerInvoice
705	21645	2	-39.60	2	-19.800000	1	-12.5000
158	20703	3	-25.50	4	-12.750000	2	-3.0000
276	20957	1	-1.45	1	-1.450000	1	-1.0000
568	21412	2	-2.52	2	-1.260000	2	-6.0000
955	22034	3	-0.02	9	-0.002222	2	-26.2222
59	16202E	4	1.95	4	0.487500	1	3.5000
308	21009	1	1.25	2	0.625000	1	1.0000
570	21414	2	2.10	3	0.700000	1	-8.0000
4	10123C	4	3.25	4	0.812500	1	-3.2500
1114	22206	1	4.17	5	0.834000	1	1.0000
473	21268	1	0.84	1	0.840000	1	2.0000
27	16010	4	3.60	4	0.900000	1	5.7500
620	21491	1	1.95	2	0.975000	1	1.0000
1059	22146	2	1.95	2	0.975000	1	-1.5000
1219	22323	2	1.95	2	0.975000	1	1.0000
1638	22769	5	9.99	11	0.999000	2	-0.5000
247	20860	1	2.10	2	1.050000	1	-3.0000
50	16162M	7	11.76	10	1.176000	1	-2.6000
107	17174	3	3.78	3	1.260000	2	3.0000

Figure 9: Dataset after removal of noisy data

Design, Method and Approach

The approach taken to augment the clustering of the two datasets required the following tasks to be performed: -

1. Select the data for transactions at an Online Retail store
2. Select and engineer the appropriate features to support the analysis
3. Pull out the top 2000 customers and products based on revenue
4. Clean the data and remove outliers
5. Normalize the data
6. Determine appropriate number of clusters and perform clustering
7. De-normalize the data
8. Use metadata from the original dataset to create customer and product profile based on clustering results

Feature Selection

Clustering of the customer dataset was based on the following attributes/ characteristics: -

1. **NoOfProducts:** This provides the information about the total number of products bought by the customer.
2. **NoOfDistinctProducts:** This provides the information about the number of distinct products bought by the customer.
3. **Revenue:** This attribute gives the information about the sales/revenue generated thus, providing the expenses done or wallet size of the customer.
4. **Visits:** This gives the details of the number of visits of each customer in turn providing information about the frequency of the customer.
5. **Average Basket Value:** This attribute describes the average price of the product(s) bought by the customer thus differentiating customers based on the expenses done in each visit.
6. **Average Basket Size:** This attribute describes the average number of product(s) bought by the customer in each visit.
7. **Recency:** This attribute provides the information about how much recent the visit of customer was. Recency was calculated as the difference of the last purchase date/invoice date and the reference date. In this case '2011-12-11' is selected as the reference date.

Following are the attributes that were taken into consideration for the product dataset: -

1. **Customers:** This provides the count of distinct customers buying a product in turn reflecting the demand of the product
2. **Revenue:** This attribute gives the information about the sales/revenue generated thus, providing the profitability from the product
3. **Visits:** This attribute gives the count of distinct invoice(s) in which the product was available consequently depicting the product requirement
4. **PricePerInvoice:** This attribute describes the average price of the product(s) per invoice hence differentiating products based on the selling price
5. **UnitsSoldPerInvoice:** This describes the total units sold per unique invoices of the products
6. **TotalCountries:** This provides the count of distinct countries in which the product is sold which describes the product requirement in various regions

Data Cleansing/ Outlier Removal

For every analysis, the first and the most important step is cleansing the data and removal of any unwanted records which would skew the results. The data cleansing was performed on two separate datasets which were customer and product centric respectively. Below is the analysis performed and the outliers found.

Customer:

To have an overall view of the customer dataset a graph was plotted using GGpairs as shown in the figure below. The graph gives a wholistic picture of the interactions of various attributes and represents each record as points.

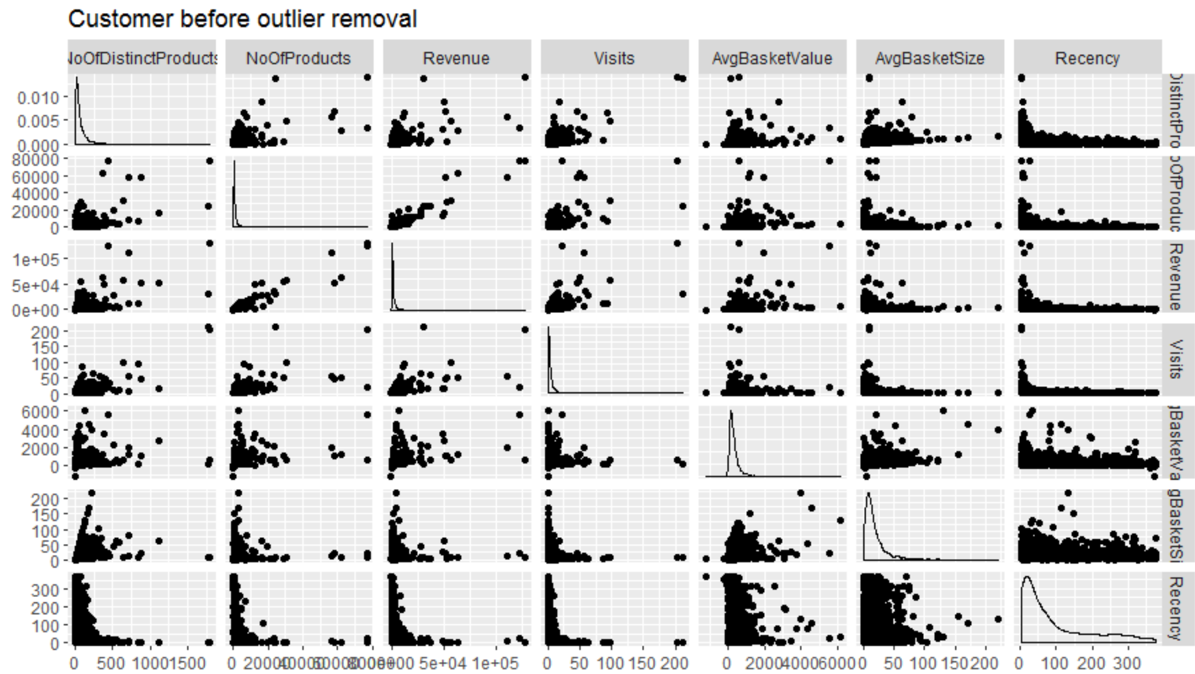

Figure 10: Customer scatterplot before outlier removal

As seen in the above chart, some points were far from the cluster, meaning these points have values which were a lot higher. Based on these we set threshold values for each attribute, these thresholds allows us to set a limit for the values in the attributes, anything beyond these values we would ignore for our clustering.

Attributes threshold values-

- NoOfDistinctProducts: >1500
- NoOfProducts: > 150000
- Revenue: >2e+05 (i.e. 250000)
- Visits: >100
- Average Basket Value: >5000
- Average Basket Size: >200
- NoOfDaysLastPurchased: >400

These points could skew the clustering and hence careful review of each record was done to find these records which are shown below.

CustomerID	NoOfDistinctPrc	NoOfProducts	Revenue	Visits	AvgBasketValue	AvgBasketSize	NoOfDays Last Purchased
12357	131	2708	6125.14	1	6125.14	131	35
12748	1769	24210	30100.71	211	142.657393	8.3839	2
14911	1794	77180	127250.08	202	629.950891	8.8812	3
12415	444	77242	122658.74	22	5575.397273	20.1818	26
12378	219	2529	3953.9	1	3953.9	219	131

Figure 11: Suspected outliers

However, post careful evaluation, it was found that even though some of the attributes had higher values in their group there was not enough evidence to consider these values as outliers. This was mostly concluded as in all the records concerned, there were only one or two high value attributes whereas majority of the attributes remained in the range which was expected.

Thus, no records were removed as outliers for customer dataset.

<u>Product: -</u>

The GGpair function of GGally package was used to plot scatterplot for all combinations of attributes selected for product clustering. Below are the steps followed for product outlier analysis –

- Plotting of scatterplots for identification of threshold (cut-off) values of each attribute
- Searching the outlier for each attribute from the cleaned data based on threshold values identified in the previous step
- Removing outlier from the (cleaned) data for further processing (cleansing the data for k-means clustering)

The following graph shows the scatterplots for all combination of parameters.

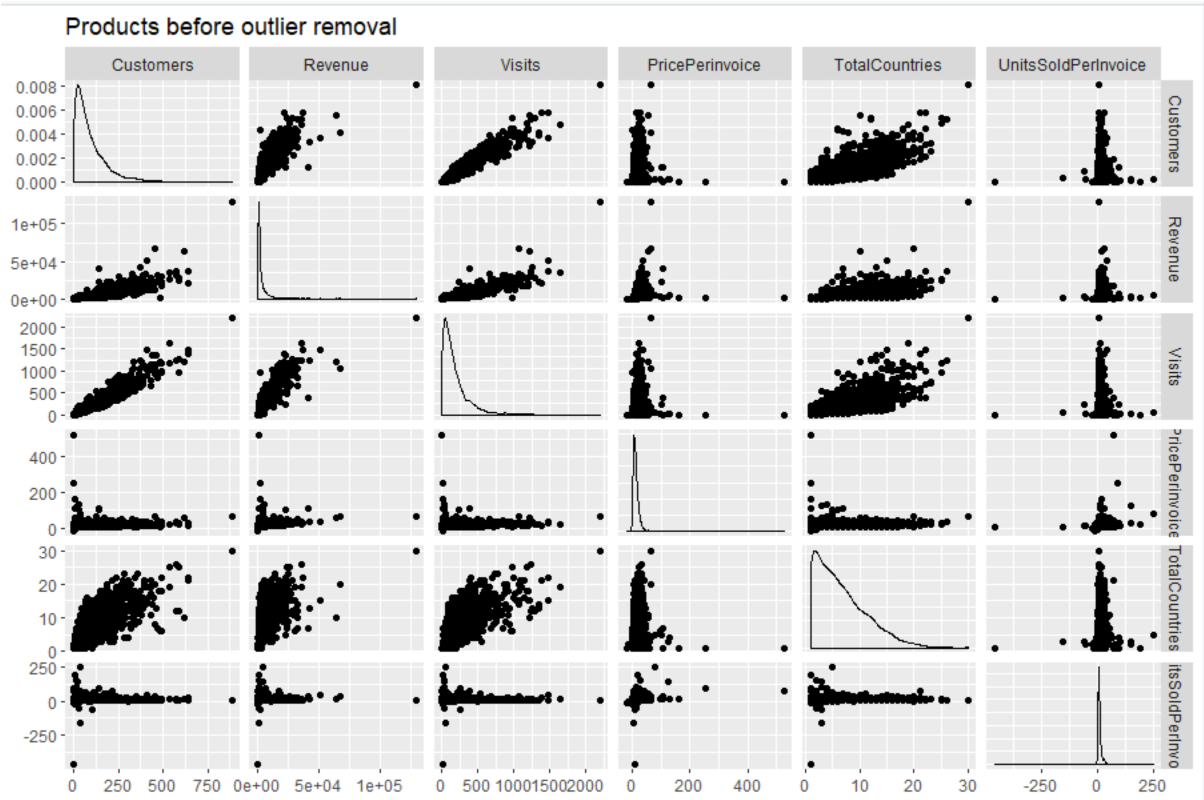

Figure 12: Product scatterplot before outlier removal

From the above graph, the threshold/ cut-off values listed below were identified for removal of product clustering outlier.

Attributes threshold values-
Customers > 750
Revenue >1e * 05 or 1,00,000
Visits > 2000
Price Per Invoice > 400
Total Countries > 30
Units Sold per Invoice < -250

Based on the threshold values, the data was scanned for identification of the outlier for each of the clustering attributes. Product with *Stock Code* **22423** was observed to be the outlier since the record for this item satisfies 3 (highlighted in red) out of 6 attribute threshold values. Hence, the same was removed from the data.

StockCode	Customers	Revenue	Visits	PricePerinvoice	TotalCountries	UnitsSoldPerInvoice
22423	**888**	**129809.6**	2203	65.13278	**30**	6.5128

Figure 13: Outlier from the Product dataset

SAINT MARY'S UNIVERSITY SINCE 1802
One University. One World. Yours.

	StockCode	Customers	Revenue	Visits	PricePerInvoice	TotalCountries	UnitsSoldPerInvoice
		All	All	All	All	All	All
1	10002	41	759.89	73	10.409452	7	14.2055
2	10080	20	119.09	24	4.962083	1	20.6250
3	10120	25	40.53	30	1.397586	2	6.6552
4	10123C	4	3.25	4	0.812500	1	-3.2500
5	10124A	5	6.72	5	1.344000	1	3.2000
6	10124G	4	7.14	4	1.785000	1	4.2500
7	10125	50	994.84	94	10.932308	4	14.2418
8	10133	102	1540.02	200	7.777879	6	14.0152
9	10135	93	2206.14	180	12.534886	6	12.6705
10	11001	46	2152.39	120	18.880614	8	12.5439
11	15030	12	41.47	14	3.190000	1	22.5385
12	15034	71	731.73	142	5.264245	5	37.4532
13	15036	195	18064.16	524	34.738769	10	43.3692
14	15039	59	1957.39	149	13.406781	4	14.1438
15	15044A	54	1453.77	104	14.114272	3	4.4951
16	15044B	38	943.88	62	15.223871	3	4.7258
17	15044C	44	1017.89	91	11.436966	3	3.4944
18	15044D	55	1817.21	87	21.378941	5	7.5647

Figure 14: Product after outlier removal

Upon searching the stock code 22423 in the cleansed data, no matching records are found.

Filter

StockCode	Customers	Revenue	Visits	PricePerInvoice	TotalCountries	UnitsSoldPerInvoice
22423	All	All	All	All	All	All
No matching records found						

Figure 15: Filter to find the Product outlier

The GGpair scatterplot was displayed again for verification and clarity over the current data set that needs to be processed for product clustering analysis.

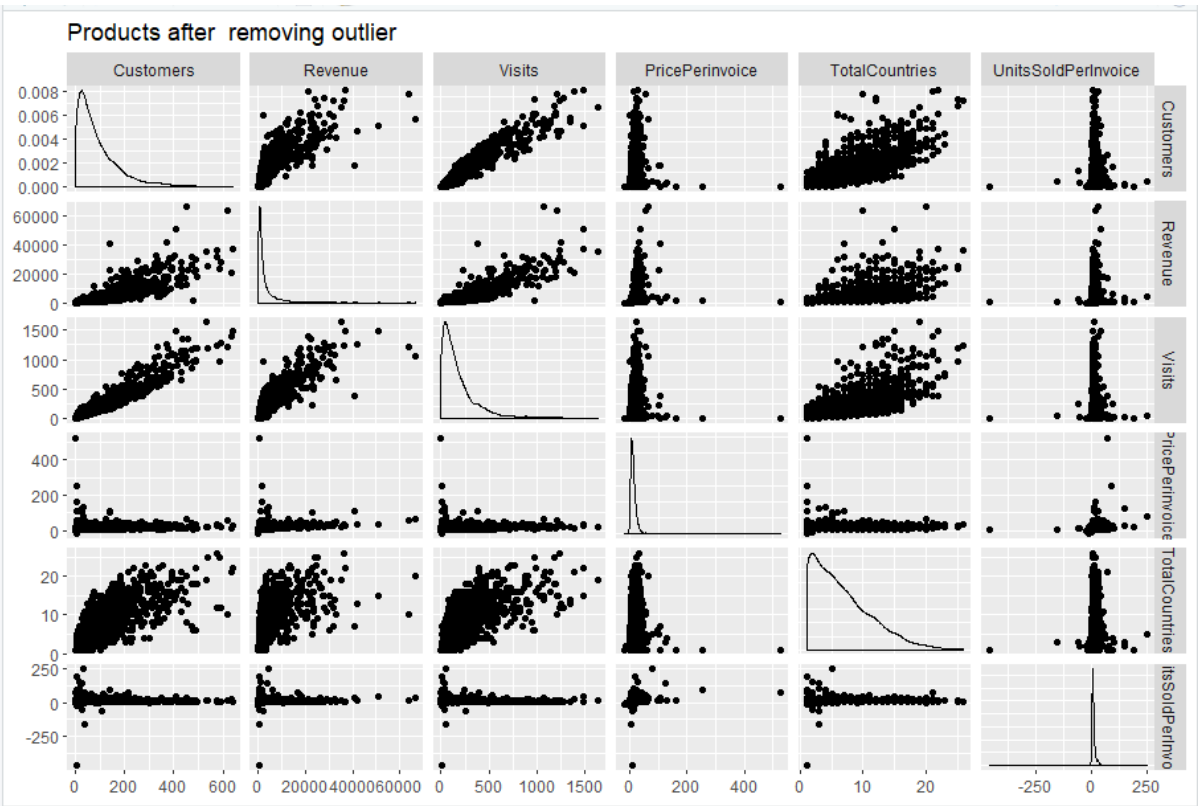

Figure 16: Product scatterplot after outlier removal

Cluster Analysis

The methodology used for clustering the dataset is K-Means . However there are other methods also available which facilitate similar requirement. These methods include :-

1. Partitioning methods
2. Hierarchical clustering
3. Fuzzy clustering
4. Density-based clustering
5. Model-based clustering

Partitioning methods comprise of K-means and K-mediods clustering methods where both of them serve as an alternative to one another . However , K-means is better than K-mediods in terms of time complexity . The time complexity of K-mediods is O(n^2) whereas for K-means is O(n*k*no_of_iteration) which is much lesser than the K-mediods.

K-means Terminologies Observed in R: -

The sum of squares metric was utilized by K-means for finding out the compactness of the clusters and the difference between the clusters.

- Betweenss – It is the sum of squares between the clusters. The higher betweeenss depicts the high heterogenity of the clusters.

- Withinss – It is the sum of squares within the clusters. The distance between the centroid and the cluster data points contribute to the withinss measure. The lower withinss depicts the high compactness of the cluster.
- Tot.withinss – It is the sum of the withinss for all the clusters.

<u>Customer: -</u>

The customer dataset, post removal of outliers was fed into RStudio. Most of the times one column consists of weighted data that takes edge over the other column when clustering is performed. Since the data consisted of attributes with varying scales, scaling was done using the scale() in R to bring the values in the attributes to a common standard scale based on the center and scaled vectors. The *CustomerID* attribute was removed and a new dataset was created which would undergo the k-means function in R.

	Customer.NoOfDistinctProducts	Customer.NoOfProducts	Customer.Revenue	Customer.Visits	Customer.AvgBasketValue	Customer.AvgBasketSize	Customer.Recency
1	-0.644986713	-3.018471e-01	-0.31528790	-0.30437989	-0.875775148	-1.000651444	2.38486291
2	0.406911744	3.279752e-01	0.38219993	0.23646936	0.659374828	-0.136869446	-0.87473665
3	-0.428419384	2.979959e-01	-0.06575209	-0.08804019	0.085361408	-0.696808835	-0.13804696
4	0.097529845	-1.401637e-01	-0.08539273	-0.41254974	2.666171817	3.405066379	-0.71327041
5	-0.479983034	-2.513691e-01	-0.26555067	-0.41254974	-0.109484067	0.002029164	2.23348831
6	-0.046848375	-1.814173e-01	-0.09934870	0.45280906	-0.506116225	-0.632661584	-0.53162090
7	-0.614048523	-2.967224e-01	-0.30074533	-0.41254974	-0.651720789	-0.787961618	1.16377453
8	-0.057161105	-1.660432e-01	-0.14329341	-0.41254974	1.774108199	2.493538553	1.44634043
9	-0.521233954	-2.403510e-01	-0.24112406	-0.41254974	0.266851732	-0.241044923	1.26469092
10	-0.108724755	1.058207e-01	0.11359865	-0.19621004	1.326813461	0.042543537	-0.67290386
11	0.695668184	3.920336e-01	0.68555795	-0.41254974	14.544048609	6.929640637	-0.56189582
12	-0.521233954	-2.383011e-01	-0.15384903	-0.30437989	0.367852416	-0.636040314	-0.88482829
13	1.551624772	1.112016e-01	0.47805344	0.02012966	1.568793848	1.569857024	-0.82427845
14	0.427537204	-3.334883e-03	0.05414731	-0.19621004	1.021495342	1.095862554	-0.37015467
15	-0.552172144	-2.785298e-01	-0.28507693	-0.41254974	-0.410321099	-0.423350488	2.00138061
16	1.417559282	2.649417e-01	0.48209268	0.66914876	0.241050007	0.079369262	-0.86464501
17	-0.418106654	-1.973038e-01	-0.22509128	-0.30437989	-0.180954890	-0.332197705	0.20506878
18	0.066591655	8.404079e-02	-0.10677195	-0.08804019	-0.072634677	0.032413425	-0.82427845
19	-0.428419384	-2.575187e-01	-0.26333824	-0.19621004	-0.608982630	-0.585401905	2.04174716
20	-0.541859414	-2.575187e-01	-0.28899853	-0.41254974	-0.470740252	-0.362581966	-0.85455337
21	0.819420943	3.010707e-01	0.24936580	-0.08804019	1.299100372	1.141438946	-0.38024631
22	-0.005597455	-1.504130e-01	-0.05273241	-0.30437989	1.146793757	0.883172729	-0.45088779
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Showing 1 to 23 of 1,999 entries

Figure 17: The scaled dataset without CustomerID column

For the k means clustering to run, it was required to determine the K value which is the number of clusters. The *withinSSrange* function (described below) takes product data frame, start value, end value and the number of iterations as the input. The output generated is an array (named *withinss*) of tot.withinss for the given range (end - start) calculated using kmeans function to find out the sharp edge or the elbow point i.e. the desired number of clusters for k-means clustering.

```
withinSSrange <- function(data,low,high,maxIter)
{
 withinss = array(0, dim=c(high-low+1));
 for(i in low:high)
 {
  withinss[i-low+1] <- kmeans(data, i, maxIter)$tot.withinss
 }
 withinss
}
```

This withinSSrange was plotted with the help of plot function where range 1-20 along with number of iteration (150) was passed as parameters . This plot results in elbow point which in turn provided the number of clusters .

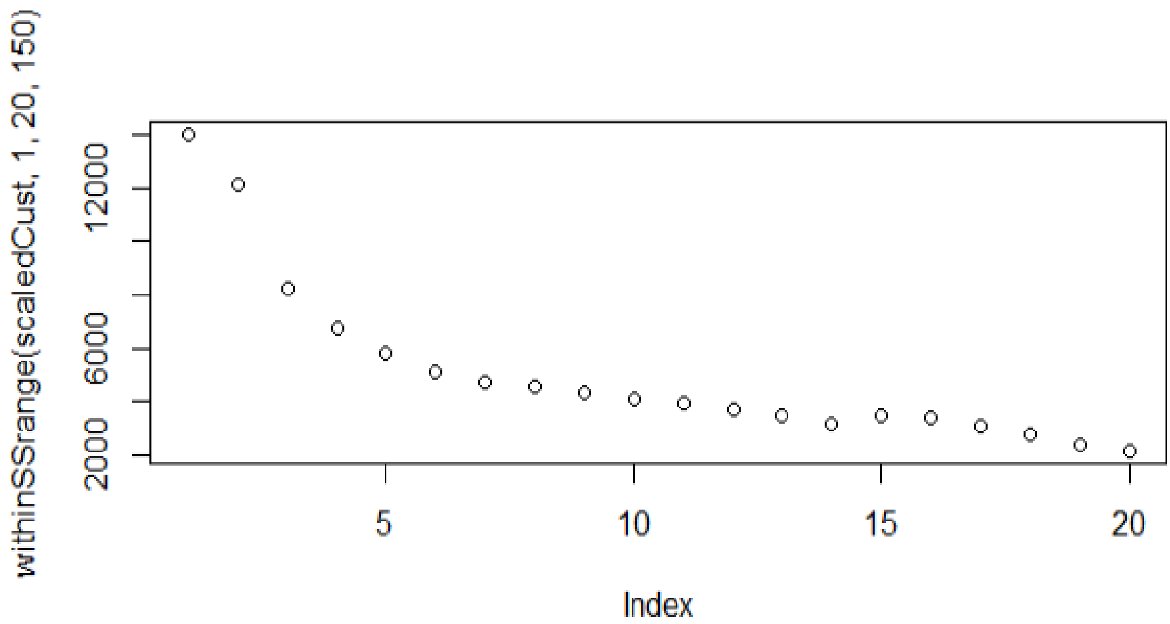

Figure 18: Elbow curve to determine K value

The ***K-means*** function was called on the scaled data with the following parameters as input-

- Scaled product data frame
- Number of Clusters selected from the elbow point graph i.e. **7**
- Number of iterations

>> results <- kmeans(scaledCust,7,150)

Upon clustering, the 7 clusters (of sizes 233, 18, 793, 140, 463, 329, 8) were generated with the scaled centroids displayed in the below figure. The tot.withinss vector is equal to 5239.887 (using the *results$tot.withinss*).

```
K-means clustering with 7 clusters of sizes 233, 18, 793, 140, 463, 329, 8

Cluster means:
  Customer.NoofDistinctProducts Customer.NoofProducts Customer.Revenue Customer.Visits Customer.AvgBasketValue
1                    0.44983574           -0.06655680      -0.07104469     -0.25542697               0.5916462
2                    1.26831034            1.04814350       1.53795647      0.09554661               6.8953020
3                   -0.09284551           -0.06690879      -0.07664953      0.00991085              -0.1845456
4                    1.53835798            0.56528771       0.69528887      1.53538672               0.2790316
5                   -0.41418445           -0.14764497      -0.17056876     -0.26910524              -0.2390514
6                   -0.44633353           -0.17761532      -0.19151921     -0.35583566              -0.2252490
7                    8.65327381           12.16932507      11.78699849      9.58085724               3.7623761
```

Figure 19: Scaled centroids and size of the clusters

The unscaled centroids found after kmeans didn’t give a clear picture of the centre values in the real data. In order to find the real values of centres unscaling of the data was done using the *unscale()* function in R. This provided clear values of the centres of attributes in the same scale as they are in the actual dataset.

>results.realCenters = unscale(results$centers,scaledCust)

	Customer.NoOfDistinctProducts	Customer.NoOfProducts	Customer.Revenue	Customer.Visits	Customer.AvgBasketValue	Customer.AvgBasketSize	Customer.Recency
1	108.54506	894.7339	1466.0731	2.476395	591.5682	48.13186	68.74249
2	188.94444	7452.0556	15473.4511	5.777778	3130.6158	49.12406	74.50000
3	55.23707	892.6633	1417.2795	4.972257	278.9262	12.53490	24.56242
4	215.47143	4611.6143	8137.4939	19.321429	465.6503	15.36398	12.70714
5	23.67171	417.7257	599.6529	2.347732	256.9718	10.92396	103.95464
6	20.51368	241.4225	417.2658	1.531915	262.5313	13.82401	272.01520
7	914.37500	72873.3750	104697.8775	95.000000	1868.7056	11.62192	7.37500

Figure 20: Unscaled centroids

Below figure gives view of the clusters formed in R via a density graph.

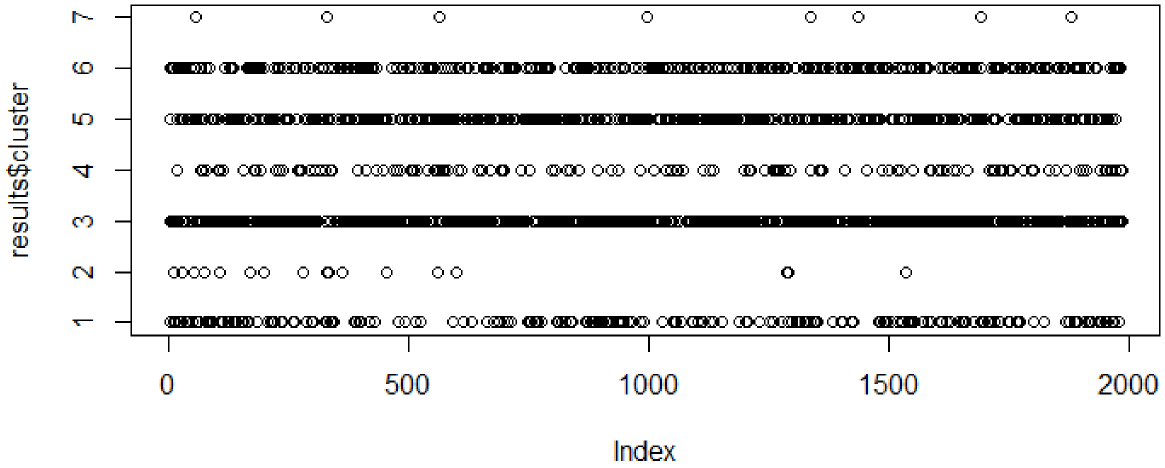

Figure 21: Cluster Density graph

From the figure above, it can be inferred that cluster 3 and 5 are the clusters where most of the customers fall, while cluster 7 has very few customers.

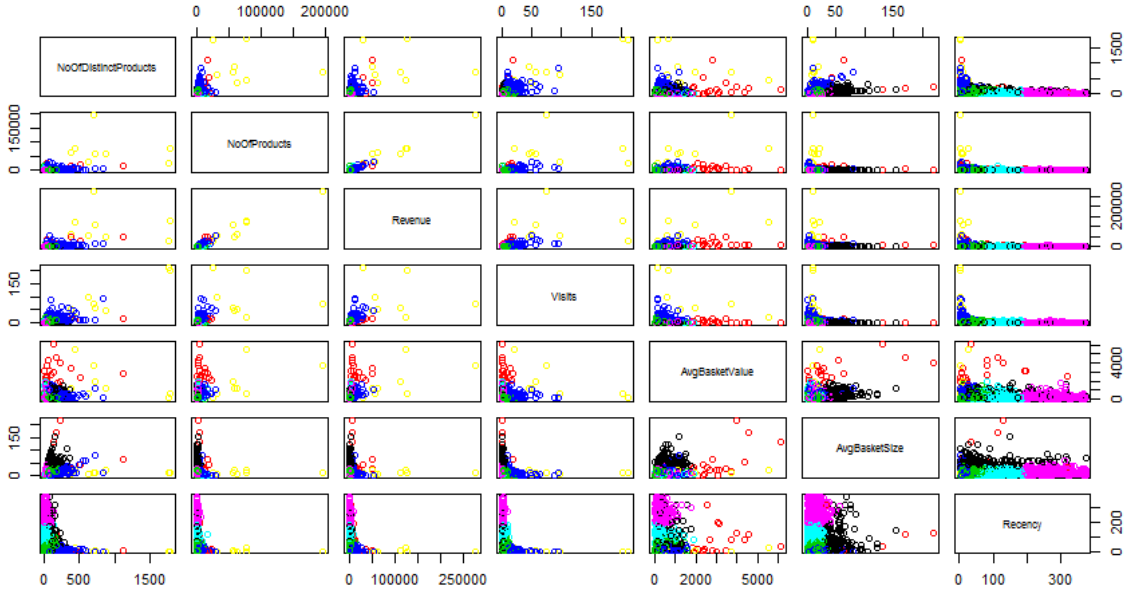

Figure 22: Clusters formed after processing the data

The graph above shows the formation of clusters based on different attributes.

Binding of the clusters was performed with the original dataset to clearly identify which *CustomerID's* are in which dataset. Below R function was used to perform the binding to get the final customer dataset.

> clusteredCustomer = cbind(Customer, results$cluster)

CustomerID	NoOfDistinctProducts	NoOfProducts	Revenue	Visits	AvgBasket	AvgBasket	NoOfDaysLastPurchased	results$cluster
12346	1	0	0	2	0	0.5	327	6
12347	103	2458	4268.6	7	609.8	14.7143	4	3
12348	22	2341	1527.15	4	381.7875	5.5	77	5
12349	73	631	1406.95	1	1406.95	73	20	1
12350	17	197	304.39	1	304.39	17	312	6
12352	59	470	1321.54	9	146.8378	6.5556	38	3
12353	4	20	89	1	89	4	206	6
12354	58	530	1052.6	1	1052.6	58	234	1
12355	13	240	453.88	1	453.88	13	216	6
12356	53	1591	2624.77	3	874.9233	17.6667	24	3
12357	131	2708	6125.14	1	6125.14	131	35	2
12358	13	248	988	2	494	6.5	3	3
12359	214	1612	4855.22	5	971.044	42.8	9	1
12360	105	1165	2260.93	3	753.6433	35	54	1
12361	10	91	184.89	1	184.89	10	289	6
12362	201	2212	4879.94	11	443.6309	18.2727	5	4
12363	23	408	552	2	276	11.5	111	5
12364	70	1506	1276.11	4	319.0275	17.5	9	3
12365	22	173	317.93	3	105.9767	7.3333	293	6
12367	11	173	160.89	1	160.89	11	6	3
12370	143	2353	3455.66	4	863.915	35.75	53	1

Figure 23: Actual customer dataset with clusters

The same set of steps were applied for the product dataset.

Product: -

The cleansed data was scaled before proceeding with the product clustering. The scaling was performed to bring data of all the attributes in the same range with the help of **scale** function in R. Moreover, while scaling the **Stock Code** (Item Id) column was removed since it was not required for product clustering analysis.

Product data frame was ready for K-means clustering after scaling. The following figure displays the scaled product dataset to be further utilized for k-means clustering.

Filter

	Customers	Revenue	Visits	PricePerinvoice	TotalCountries	UnitsSoldPerInvoice
1	-0.54081887	-0.42214047	-0.52948495	-0.215169734	0.13095746	0.35002693
2	-0.76557307	-0.53763239	-0.75995873	-0.536134746	-1.15564213	0.73193766
3	-0.71206017	-0.55179132	-0.73173745	-0.746158856	-0.94120886	-0.09915767
4	-0.93681436	-0.55851033	-0.85402966	-0.780632770	-1.15564213	-0.68844066
5	-0.92611178	-0.55788493	-0.84932611	-0.749316202	-1.15564213	-0.30471542
6	-0.93681436	-0.55780923	-0.85402966	-0.723331994	-1.15564213	-0.24224852
7	-0.44449565	-0.37979524	-0.43071047	-0.184362480	-0.51234234	0.35218650
8	0.11203855	-0.28153700	0.06786546	-0.370224929	-0.08347581	0.33870555
9	0.01571532	-0.16148164	-0.02620547	-0.089936810	-0.08347581	0.25870627
10	-0.48730597	-0.17116904	-0.30841826	0.283960509	0.34539072	0.25117455
11	-0.85119371	-0.55162191	-0.80699419	-0.640547838	-1.15564213	0.84577615
12	-0.21974145	-0.42721577	-0.20494024	-0.518331027	-0.29790907	1.73308574
13	1.10737855	2.69662245	1.59181452	1.218340558	0.77425725	2.08504210
14	-0.34817242	-0.20631406	-0.17201541	-0.038563791	-0.51234234	0.34635626
15	-0.40168533	-0.29708191	-0.38367501	0.003122362	-0.72677560	-0.22766696
16	-0.57292662	-0.38897980	-0.58122396	0.068501163	-0.72677560	-0.21394209
17	-0.50871113	-0.37564092	-0.44482111	-0.154627472	-0.72677560	-0.28720089
18	-0.39098275	-0.23157881	-0.46363530	0.431164702	-0.29790907	-0.04504944
19	0.55084435	2.31639128	0.95213220	1.619553820	1.41755705	-0.06113615

Figure 24: Scaled Product dataset to be used for clustering

The elbow point for the product cluster was plotted using the ***withinSSrange*** custom function (described above).

This withinSSrange was plotted with the help of plot function where range 1-50 along with number of iteration (150) was passed as parameters . This plot results in elbow point which in turn provided the number of clusters .

>> *plot(withinSSrange(prod.scale,1,50,150))*

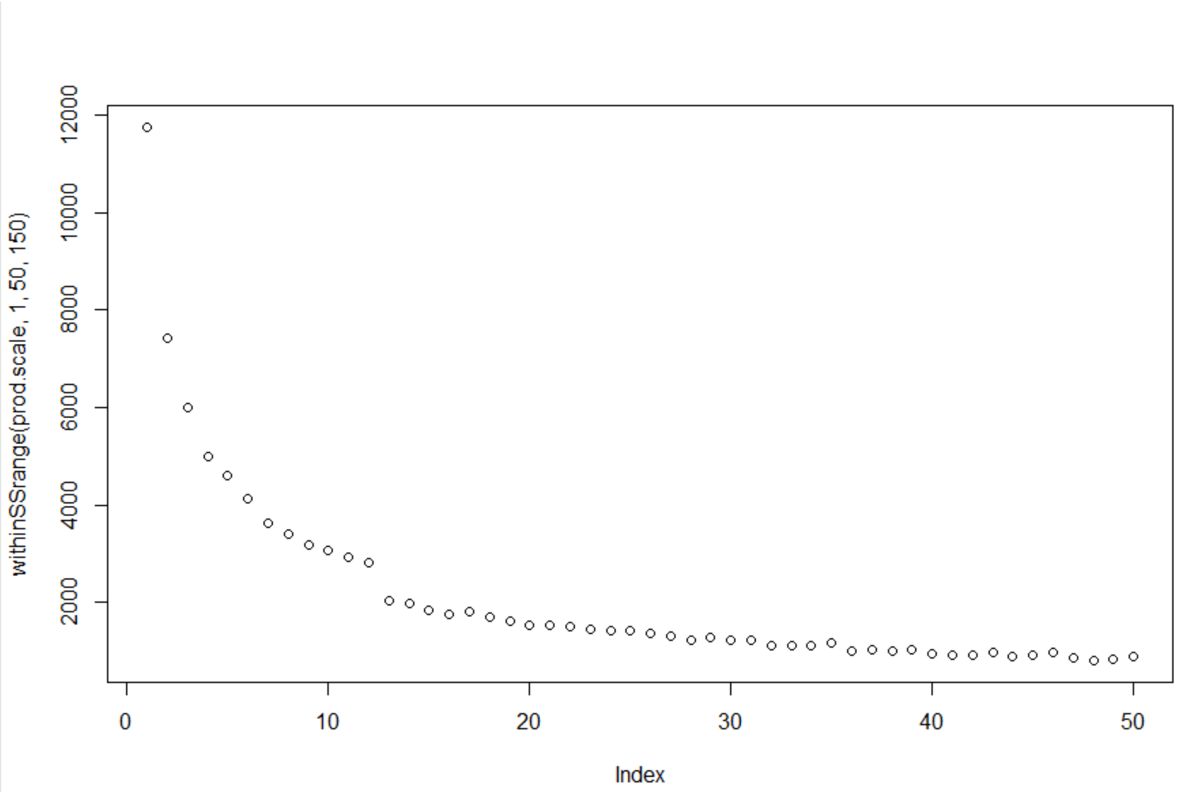

Figure 25: Plot of withinSSrange

The ***Kmeans*** function was called on the scaled data with the following parameters as input-

- Scaled product data frame
- Number of Clusters selected from the elbow point graph i.e. **5**
- Number of iterations

>> pkm = kmeans(prod.scale, 5, 150)

Upon clustering, the 5 clusters (of sizes 644, 5, 74, 958, 279) were generated with the scaled centroids displayed in the below figure. The tot.withinss vector is equal to 4391.122 (pkm$tot.withinss) which is lower than the tot.withinss for 4 number of clusters, thus, greater compactness observed.

```
K-means clustering with 5 clusters of sizes 644, 5, 74, 958, 279

Cluster means:
    Customers    Revenue     Visits PricePerinvoice TotalCountries UnitsSoldPerInvoice
1  0.03796805 -0.1409252 -0.0336698     0.005797902      0.3530491          0.13508054
2 -0.83621010 -0.1609424 -0.7815950    12.719604716     -0.8983222          6.25401280
3  3.13826554  3.9091128  3.5994661     0.843804715      2.0029019          0.25340052
4 -0.64275034 -0.4519937 -0.5788329    -0.230350636     -0.7625891         -0.17196072
5  1.30198139  0.8433553  1.1245619     0.325815369      1.2884359          0.09937205
```

Figure 26: Scaled centroids of the clusters

To obtain the real cluster centroids unscaling of pkm$centers value is required. The same was performed using the unscale function in R. The centroids of the 5 clusters w.r.t to the clustering attributes is displayed in the below figure. These centroids were the final centroids after performing

150 iterations and given each anonymous data point in the dataset an identity by assigning a cluster type to these data points.

>> prod.realCenters = unscale(pkm$centers,prod.scale)

	Customers	Revenue	Visits	PricePerInvoice	TotalCountries	UnitsSoldPerInvoice
1	95.07919	2320.1961	178.41304	14.15968	8.035714	10.592487
2	13.40000	2209.1320	19.40000	229.93646	2.200000	113.445020
3	384.75676	24791.5904	950.83784	28.38220	15.729730	12.581316
4	31.47599	594.2513	62.50835	10.15180	2.832985	5.431461
5	213.18280	7781.4178	424.65950	19.59097	12.397849	9.992267

Figure 27: Unscaled centroids of the clusters

The density of the clusters follows the sequence 4→1→5→3→2 as shown in the below figure.

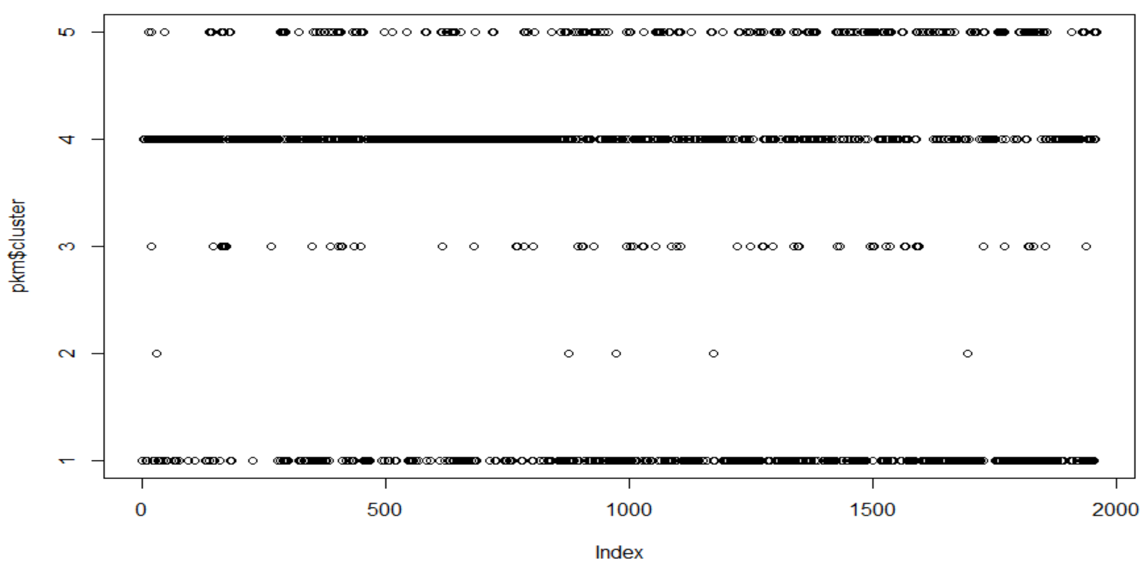

Figure 28: Clusters formed after processing

After clustering, the new data frame was created which consists of product attributes and the cluster column value assigned to all the records in the data frame. The data frame was plotted and coloured based on the cluster value for all the records in the below figure.

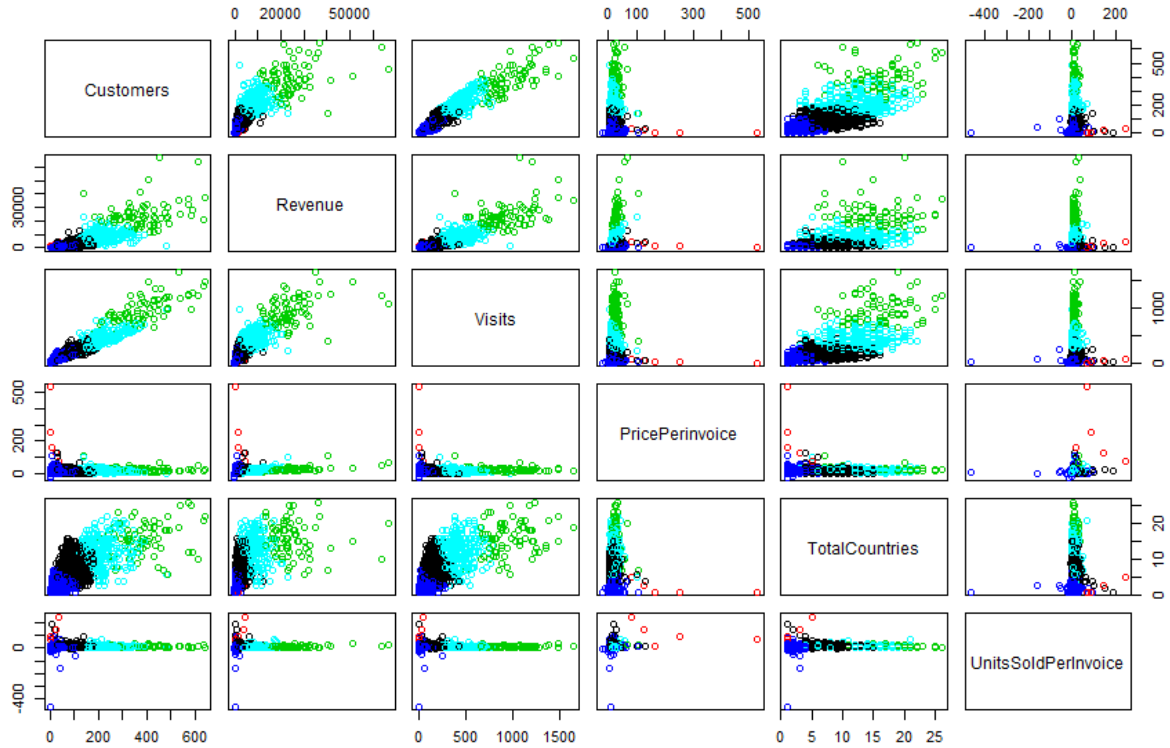

Figure 29: Clusters formed after processing

The data frame containing cluster-value column was downloaded as a csv file – "FinalResults.csv", thus, the below figure displays the above-mentioned csv with pkm$cluster as the column with cluster values.

	StockCode	Customers	Revenue	Visits	PricePerinvoice	TotalCoun	UnitsSoldPerInvoice	pkm$clust
1	10002	41	759.89	73	10.409452	7	14.2055	1
2	10080	20	119.09	24	4.962083	1	20.625	4
3	10120	25	40.53	30	1.397586	2	6.6552	4
4	10123C	4	3.25	4	0.8125	1	-3.25	4
5	10124A	5	6.72	5	1.344	1	3.2	4
6	10124G	4	7.14	4	1.785	1	4.25	4
7	10125	50	994.84	94	10.932308	4	14.2418	4
8	10133	102	1540.02	200	7.777879	6	14.0152	1
9	10135	93	2206.14	180	12.534886	6	12.6705	1
10	11001	46	2152.39	120	18.880614	8	12.5439	1
11	15030	12	41.47	14	3.19	1	22.5385	4
12	15034	71	731.73	142	5.264245	5	37.4532	1
13	15036	195	18064.16	524	34.738769	10	43.3692	5
14	15039	59	1957.39	149	13.406781	4	14.1438	4
15	15044A	54	1453.77	104	14.114272	3	4.4951	4
16	15044B	38	943.88	62	15.223871	3	4.7258	4
17	15044C	44	1017.89	91	11.436966	3	3.4944	4
18	15044D	55	1817.21	87	21.378941	5	7.5647	4
19	15056B	143	15954.47	388	41.548099	13	7.2943	5
20	15056N	185	22831.52	551	42.20244	13	7.4381	3
21	15056P	90	4613.07	176	26.977018	9	4.6842	1

Figure 30: Resultant dataset after clustering

Cluster Profiling

Customer:

Post creation of clusters and binding the same to the original customer dataset, a clear vision of which customers were tagged to which of the clusters was achieved. The next step was to define the clusters which best defines the behavior of customers in that cluster.

To further ease the analysis, the below report was created to facilitate the analysis of different parameters in each cluster

SAINT MARY'S UNIVERSITY SINCE 1802

One University. One World. Yours.

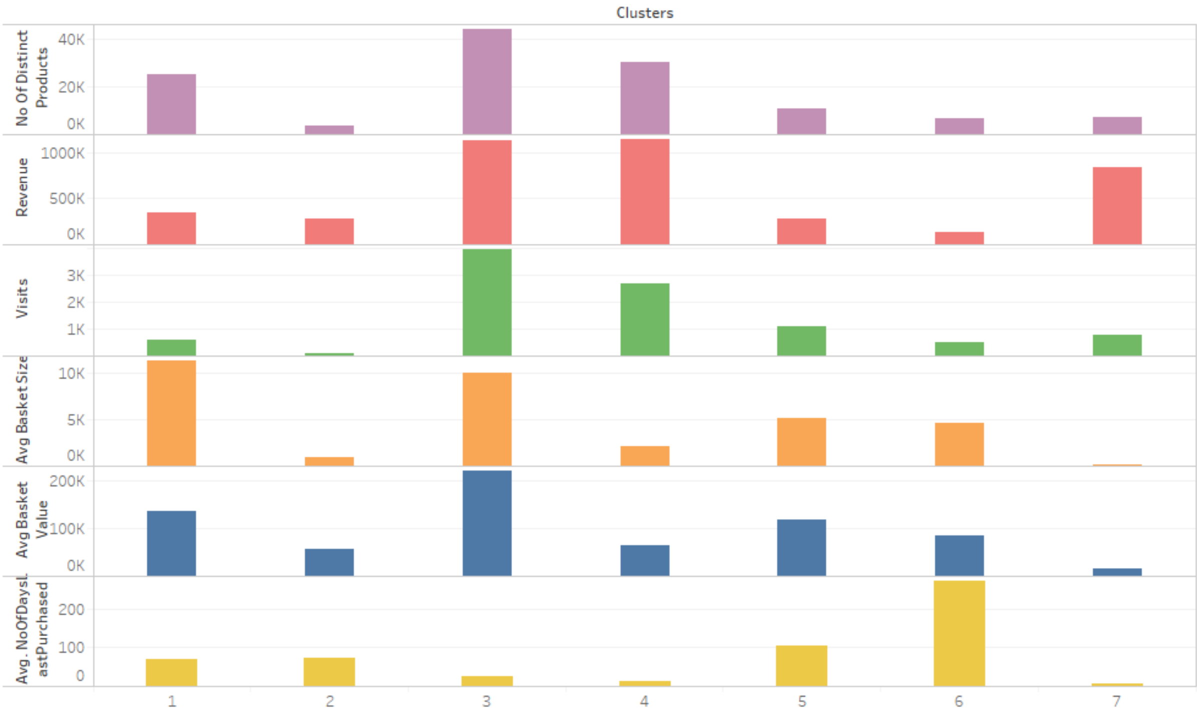

Figure 31:Revenue, No of Products, No of distinct products, Visits, NoOfDaysLastPurchased by Customer Segments

The cluster types were designated to each cluster . These cluster types were defined based on the interaction of the parameters – *NoOfDistinctProducts, NoOfProducts, Revenue, Visits, AvgBasketValue, AvgBasketSize* and *NoOfDaysLastPurchased.* The cluster types are defined below with the rules governing them

Cluster Type	Definition	Action
Champions	They bought recently, visit often and generate more revenue	Can reward such customers
Loyal	They spend good money and visit above average number of times	Engage customers with surveys and offers and make them feel valuable
Need Attention	They have low frequency, spent less money but are moderately recent shoppers	Send offers and try to motivate them with new products
Promising	They are most recent shoppers and have bought large number of products	Send mailers about new products
Seasoned Buyers	They are less frequent, buy low value products but in large quantities. Probably respond to offers	Try to attract with offers and deals to turn them into loyal shoppers

Hibernating	The last purchase done by them was long time back and are low spenders and order less products	Offer new products in market and attract them with deals
About to Sleep	Last purchase was over a year and half back, bought less products and have low visits	Share new products and offers/discounts

Customer Cluster Type	Cluster Number
Seasoned Customer	1
Need Attention	2
Champions	3
Loyal	4
Hibernating	5
About to Sleep	6
Promising	7

In order to further improve the understanding, the flow block chart was created (using Tableau) to show the percentage of customers in each cluster category.

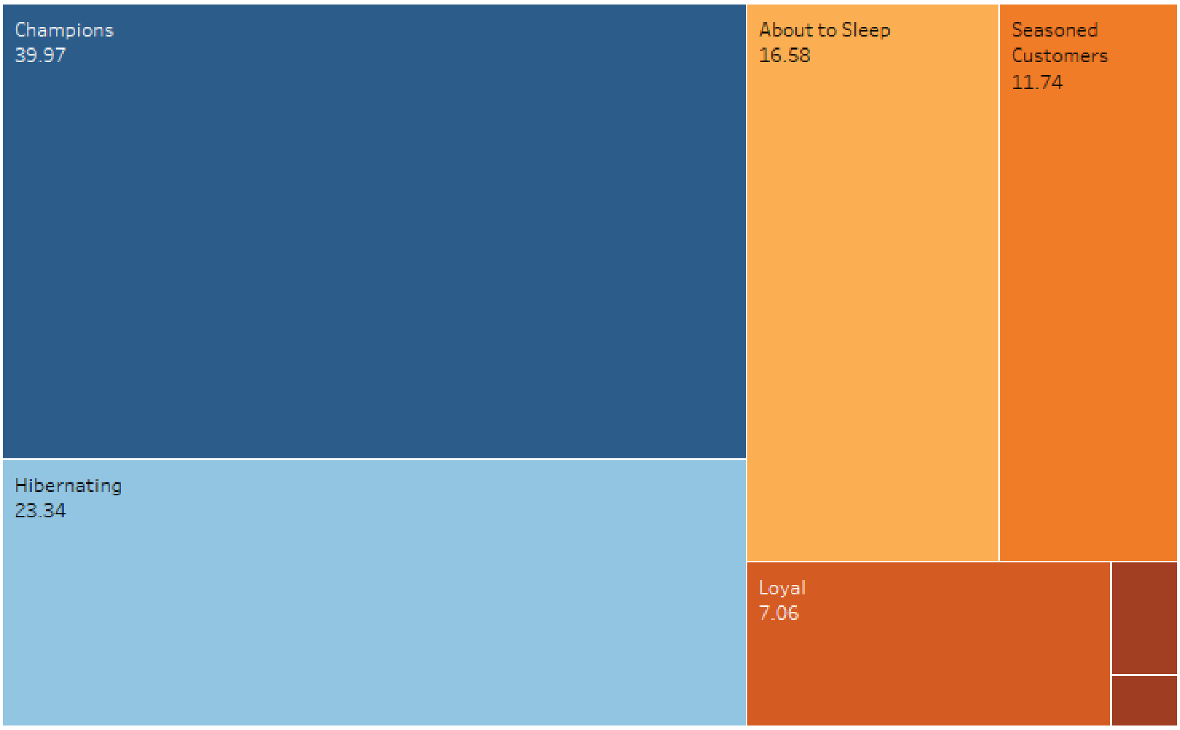

Figure 32: Percentage of Customers by Profile

Product:-

Product profiling was performed based on the features utilized for clustering i.e. Total customers, Revenue, Visits, Total Countries, Price Per Invoice and Units Sold Per Invoice.

The clusters were plotted against all the attributes (measured in average quantities) with the motive of identifying the cluster types for each cluster. Thus, the below bar chart visualization was achieved in Tableau for all the clusters.

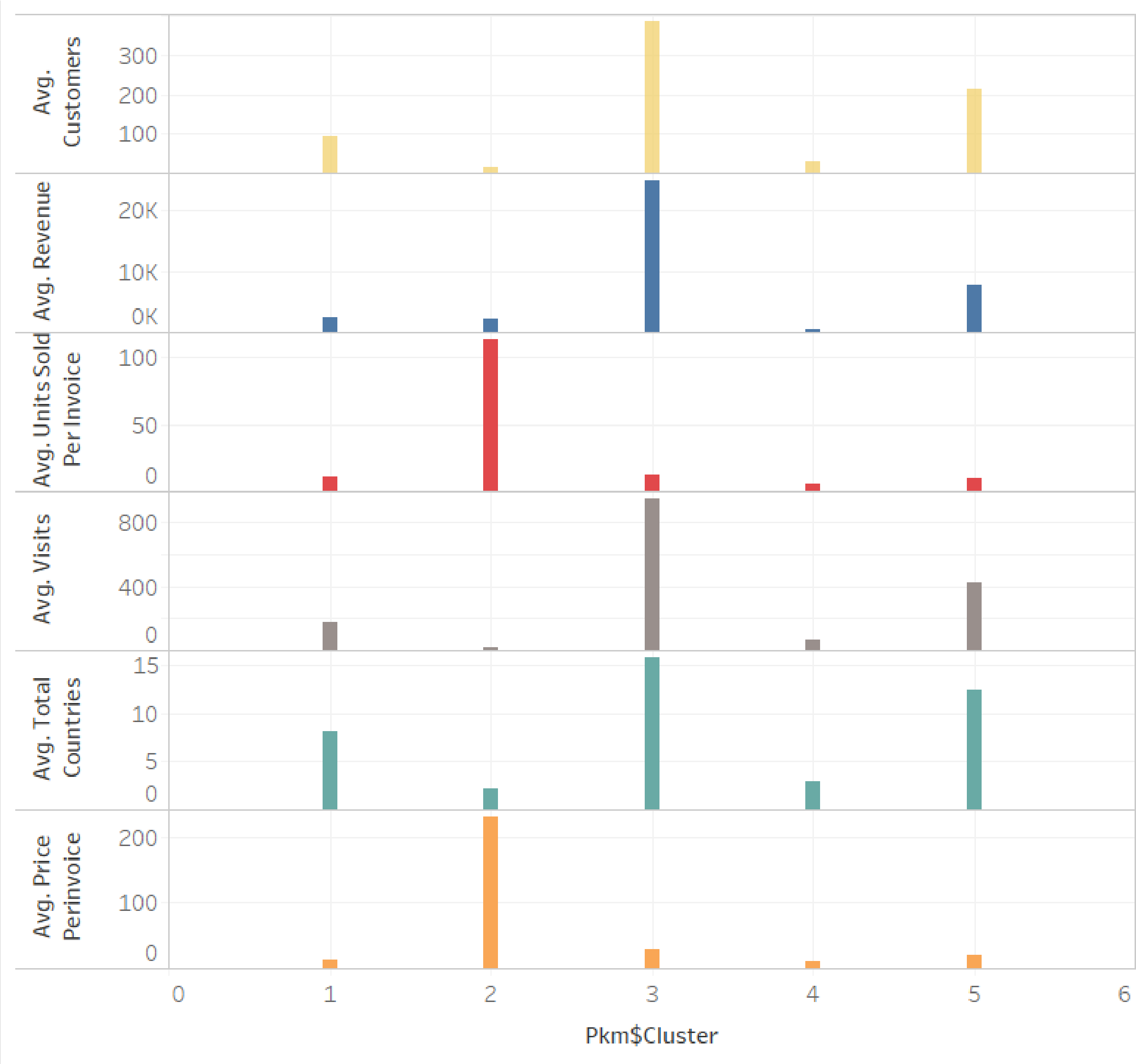

Figure 33: Bar chart representation of the clusters

The following rules define the cluster type or the profile type for the product clusters obtained from the k-means cluster analysis.

Cluster Type	*Definition*	*Recommendation*
Champions	Number of customers buying the product is high, high revenue is generated from the high number of visits, product is bought in high number of countries. However, price per invoice and units sold per invoice is low.	Continue manufacturing these products on the same scale and price.
Maintain	Number of customers buying the product is above average, above average revenue is generated from the above average number of visits, product is bought in high number of countries. However, price per invoice and units sold per invoice is low.	Market value of products is good with high number of customers. So, 'quantity purchase schemes' (1+1) needs to be utilized for increase in units sold per invoice.
High Potential	Number of customers buying the product is below average, below average revenue is generated from the below average number of visits, product is bought in almost medium number of countries. However, price per invoice and units sold per invoice is low.	Marketing strategy needs to be revised for engaging more customers and increasing visits.
Not In Demand	Number of customers buying the product is low, low revenue is generated from the low number of visits, product is bought in low number of countries. Moreover, price per invoice and units sold per invoice is also low.	Carefully control the inventory for these products. Moreover, launch an alternative product range in market if possible.
Low-Price Bids	Number of customers buying the product is low, low revenue is generated from the low number of visits, product is bought in low number of countries. However, price per invoice and units sold per invoice is high.	Customers buying the products are majorly from United Kingdom. Promotions to be done in other countries to increase sales.

Figure 34: Product profiles based on cluster characteristics

As per the above-mentioned rules each cluster was assigned a cluster type on the basis of the calculated measures for each of the attributes. Upon cluster profile analysis, the below product profile results were achieved.

Product Cluster Type	Cluster Number
Not in Demand	4
Champion	3
High Potential	1
Maintain	5
Low-Price Bids	2

In order to further improve the understanding, the flow block chart was created to show the percentage of products in each cluster category.

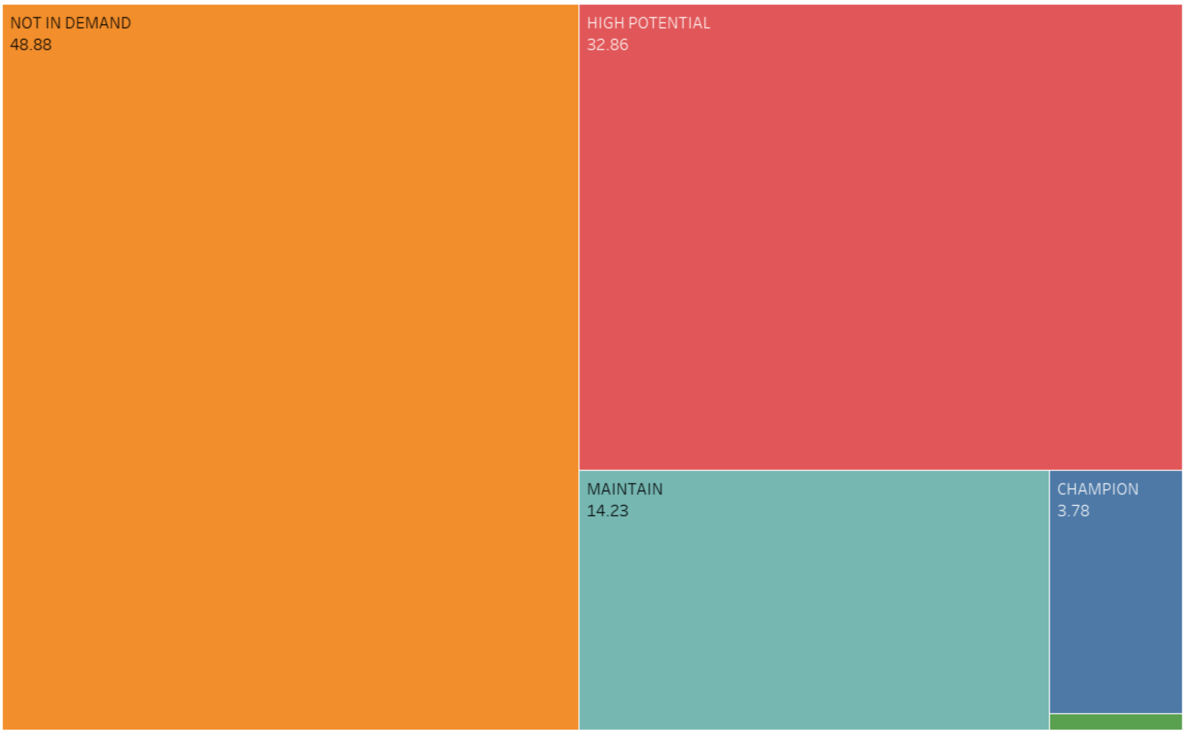

Figure 35: Percentage of Products by Profile

Conclusion

Based on the profiles of customers and products generated , recommendations are provided for the business to take into consideration . These profiles and recommendations will help the business owner to get the clear insight of the selling nature and purchasing behaviour of the products and customers respectively. This will lead them to take decision on the marketing strategy, inventory management and business expansion to be taken into consideration to increase the overall revenue generated.

References

[1] Title -drawbacks-of-k-medoid-pam-algorithm. Retrieved from *URL*: https://stackoverflow.com/questions/46514123/drawbacks-of-k-medoid-pam-algorithm

[2] Title -drawbacks-of-k-medoid-pam-algorithm. Retrieved from *URL*: https://en.wikipedia.org/wiki/K-medoids

[3] Title -drawbacks-of-k-medoid-pam-algorithm. Retrieved from *URL*: https:/0/www.datanovia.com/en/blog/types-of-clustering-methods-overview-and-quick-start-r-code/

[4] Title -drawbacks-of-k-medoid-pam-algorithm. Retrieved from *URL*:http://data-mining.business-intelligence.uoc.edu/k-means

[5] Title – Clustering of customers and products . Retrieved from *URL*: https://www.youtube.com/watch?v=PE_tiO8MXIs&feature=youtu.be

Appendix

SQL Scripts-

Customer:

```
Create table CustomerData

as

SELECT CustomerID,

count(DISTINCT StockCode) as  NoOfDistinctProducts,

sum(Quantity) as NoOfProducts,

sum(UnitPrice*Quantity) as Revenue,

count(DISTINCT InvoiceNo) as Visits,

(sum(UnitPrice*Quantity)/count(DISTINCT InvoiceNo)) as AvgBasketValue,

(count(DISTINCT StockCode)/count(DISTINCT InvoiceNo)) as AvgBasketSize

FROM dataset04.OnlineRetail

GROUP By CustomerID having CustomerID <> 0  LIMIT 2000;
```

Product:

```
Create table ProductClusterNew

as

SELECT StockCode, count(DISTINCT CustomerID) as Customers, sum(UnitPrice*Quantity) as Revenue

,count(InvoiceNo) as Visits, (sum(UnitPrice*Quantity)/count(DISTINCT InvoiceNo)) as
PricePerinvoice, count(DISTINCT(Country)) as TotalCountries,

sum(Quantity)/count(DISTINCT InvoiceNo) as UnitsSoldPerInvoice

FROM `OnlineRetail` GROUP BY StockCode LIMIT 2000;
```

R Script-

Customer:

```
# R Script for K means clustering of Customers

# Import libraries

library(ggplot2)

library(GGally)

library(DMwR)
```

Import Customer dataset

```
Customer <- read.csv("Customer.csv", header = T)
```

Plot to view the data and check outliers

```
ggpairs(Customer[, which(names(Customer) != "CustomerID")], upper = list(continuous = ggally_points),lower = list(continuous = "points"), title = "Customer before outlier removal")
```

Preparing Dataset with only Fact columns for K-means

```
Cust <- data.frame(Customer$NoOfDistinctProducts,Customer$NoOfProducts,Customer$Revenue,Customer$Visits,Customer$AvgBasketValue,Customer$AvgBasketSize,Customer$Recency)
```

Scaling attributes

```
scaledCust <- scale(Cust)
```

Function to determine the K value

```
withinSSrange <- function(data,low,high,maxIter)
{
 withinss = array(0, dim=c(high-low+1));
 for(i in low:high)
 {
  withinss[i-low+1] <- kmeans(data, i, maxIter)$tot.withinss
 }
 withinss
}
```

Passing the parameters in the above function and getting the elbow curve

```
plot(withinSSrange(scaledCust,1,20,150))
```

Set seed value

```
set.seed(5580)

# Considering k =7
results <- kmeans(scaledCust,7,150)

# View the clusters
results

# Plot the clusters
plot(results$cluster)

# Grid view of the customers
plot(clusteredCustomer[,2:8], col=results$cluster)

# Binding the customers to the actual dataset
clusteredCustomer = cbind(Customer, results$cluster)
View(clusteredCustomer)

# Writing the new dataset with clusters into a csv file
write.csv(clusteredCustomer,file ='clusteredcustomer_new.csv')
```

Product:

```
library(ggplot2)
library(GGally)
library(DMwR)
library(dplyr)
#install.packages("dplyr")

set.seed(55)
```

```
prod = read.csv("C://Users//sidha//Documents//SMU//Semester 2//Data Mining//Assignments//Assignment 1//ProductClusterNew.csv",sep = ',')  # Product derived dataset

prod.filter <- prod %>% filter(Revenue!= 0 & PricePerinvoice!= 0) # Data cleaning for Revenue and Price Per Invoice

View(prod.filter)

View(prod)

ggpairs(prod.filter[, which(names(prod.filter) != "StockCode")],

        upper = list(continuous = ggally_points), lower = list(continuous = "points"), title = "Products before outlier removal")   # Scatterplot for Outlier analysis and plotted for all combination of product clustering attributes.


prod.clean <-prod.filter[prod.filter$StockCode != 22423, ]  # Data Cleansing – Outlier Removed

View(prod.clean)


ggpairs(prod.clean[,which(names(prod.clean)!="StockCode")], upper = list(continuous =

                                          ggally_points), lower = list(continuous = "points"), title = "Products after  removing outlier") # Data Cleansing Scatterplots after Outlier Removal

prod.scale = scale(prod.clean[-1])  # Scaling data to normalized data

View(prod.scale)

withinSSrange <-function(data,low,high,maxIter)

{

  withinss = array(0, dim=c(high-low+1));

  for(i in low:high)

  {

    withinss[i-low+1] <-kmeans(data, i, maxIter)$tot.withinss

  }

  withinss

}

plot(withinSSrange(prod.scale,1,50,150))

set.seed(55)

pkm = kmeans(prod.scale, 5, 150)  # K-means clustering performed for cluster size of 5
```

pkm$tot.withinss

View(pkm)

prod.realCenters = unscale(pkm$centers,prod.scale) # Final Unscaled K-means Centroids

View(prod.realCenters)

clusteredprod = cbind(prod.clean, pkm$cluster) # Data frame created with binded cluster number for each record

plot(pkm$cluster)

plot(clusteredprod[,2:7], col=pkm$cluster)

write.csv(clusteredprod, file = "FinalResultsagain.csv", col.names = FALSE) # Final dataset created with cluster numbers for profiling and visualization.