CAPSTONE PROJECT - 4

Online Retail Customer Segmentation

Unsupervised Machine Learning

BY

Swapnil Bhagawat

❖ Problem Statement:

- To identify major customer segments on a transnational data set.
- Data set contains all the transactions occurring between 1st December 2010 and 9th December 2011 for a UK-based and registered non-store online retail.
- The company mainly sells unique all-occasion gifts.
- Many customers of the company are wholesalers.

❖ Data Description:

Total Rows= 541909
Total features=8

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

Data Wrangling

Information of the data

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 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  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

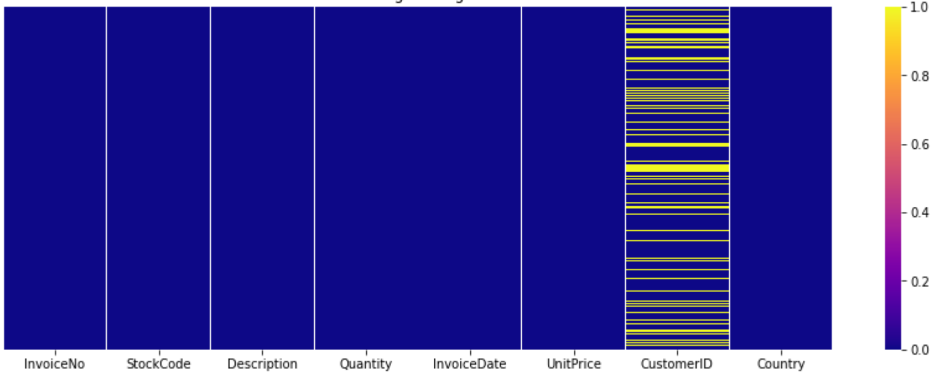
- Invoicedate to datetime.
- If InvoiceNo starts with C means it's a cancellation.
- Shape of data after dropping entries=397884

Null values

```
# Let's check the null values count.
retail_df.isnull().sum().sort_values(ascending=False)
```

```
CustomerID     135080
Description      1454
InvoiceNo           0
StockCode           0
Quantity            0
InvoiceDate         0
UnitPrice           0
Country             0
dtype: int64
```

Visualising Missing Values



❖ Data Wrangling :

```
retail_df[retail_df['Quantity']<0]
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
141	C536379	D	Discount	-1	01-12-2010 09:41	27.50	14527.0	United Kingdom
154	C536383	35004C	SET OF 3 COLOURED FLYING DUCKS	-1	01-12-2010 09:49	4.65	15311.0	United Kingdom
235	C536391	22556	PLASTERS IN TIN CIRCUS PARADE	-12	01-12-2010 10:24	1.65	17548.0	United Kingdom
236	C536391	21984	PACK OF 12 PINK PAISLEY TISSUES	-24	01-12-2010 10:24	0.29	17548.0	United Kingdom
237	C536391	21983	PACK OF 12 BLUE PAISLEY TISSUES	-24	01-12-2010 10:24	0.29	17548.0	United Kingdom
...	...	...	...	...	...	...	...	...
540449	C581490	23144	ZINC T-LIGHT HOLDER STARS SMALL	-11	09-12-2011 09:57	0.83	14397.0	United Kingdom
541541	C581499	M	Manual	-1	09-12-2011 10:28	224.69	15498.0	United Kingdom
541715	C581568	21258	VICTORIAN SEWING BOX LARGE	-5	09-12-2011 11:57	10.95	15311.0	United Kingdom
541716	C581569	84978	HANGING HEART JAR T-LIGHT HOLDER	-1	09-12-2011 11:58	1.25	17315.0	United Kingdom
541717	C581569	20979	36 PENCILS TUBE RED RETROSPOT	-5	09-12-2011 11:58	1.25	17315.0	United Kingdom

- Invoice No starting with C had negative entries in the quantity column means negative values in quantity column indicates cancellations.

Feature Engineering:

- Changed the datatype of Invoice Date column into datetime .

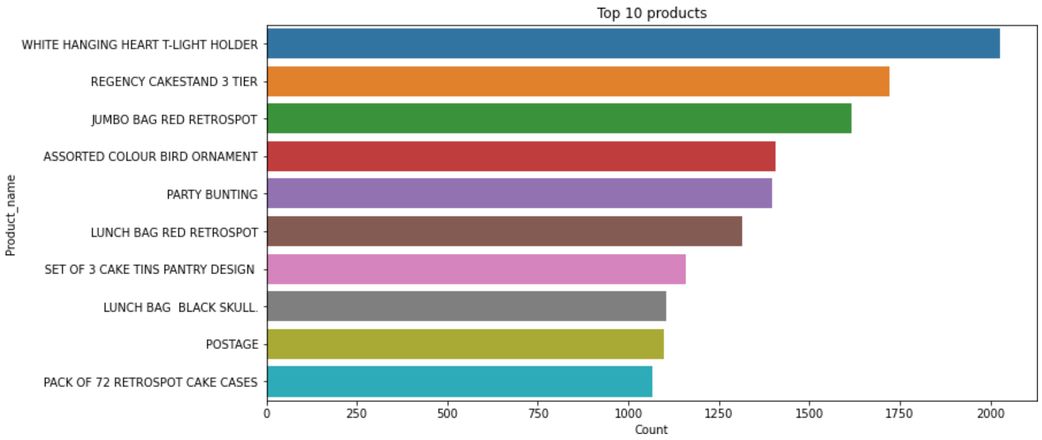
```
retail_df["year"] = retail_df["InvoiceDate"].apply(lambda x: x.year)
retail_df["month_num"] = retail_df["InvoiceDate"].apply(lambda x: x.month)
retail_df["day_num"] = retail_df["InvoiceDate"].apply(lambda x: x.day)
retail_df["hour"] = retail_df["InvoiceDate"].apply(lambda x: x.hour)
retail_df["minute"] = retail_df["InvoiceDate"].apply(lambda x: x.minute)
```

```
retail_df['TotalAmount']=retail_df['Quantity']*retail_df['UnitPrice']
```

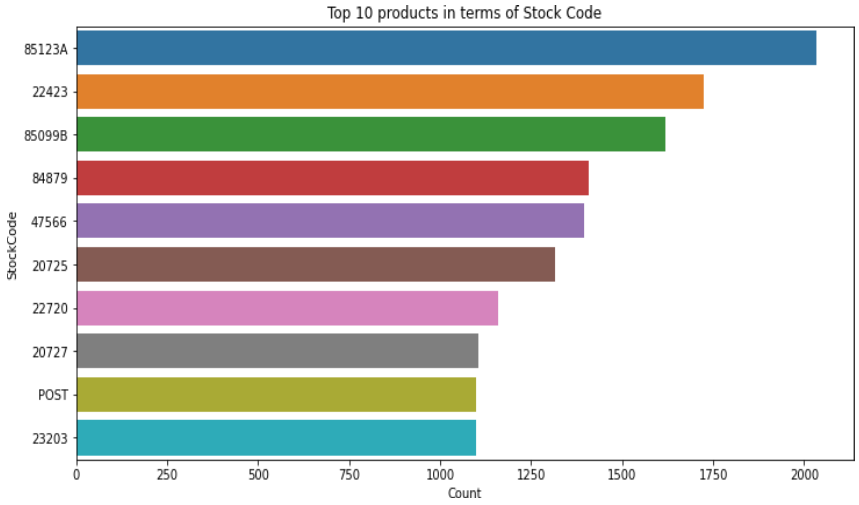
```
def time(time):
  if (time==6 or time==7 or time==8 or time==9 or time==10 or time==11) :
    return'Morning'
  elif (time==12 or time==13 or time==14 or time==15 or time==16 or time==17):
    return 'Afternoon'
  else:
    return 'Evening'
```

```
retail_df['Day_time_type']=retail_df['hour'].apply(time)
```

❖EDA(Exploratory Data Analysis):



	Product_name	Count
0	WHITE HANGING HEART T-LIGHT HOLDER	2028
1	REGENCY CAKESTAND 3 TIER	1723
2	JUMBO BAG RED RETROSPOT	1618
3	ASSORTED COLOUR BIRD ORNAMENT	1408
4	PARTY BUNTING	1396
5	LUNCH BAG RED RETROSPOT	1316
6	SET OF 3 CAKE TINS PANTRY DESIGN	1159
7	LUNCH BAG BLACK SKULL.	1105
8	POSTAGE	1099
9	PACK OF 72 RETROSPOT CAKE CASES	1068



	StockCode	Count
0	85123A	2035
1	22423	1723
2	85099B	1618
3	84879	1408
4	47566	1396
5	20725	1317
6	22720	1159
7	20727	1105
8	POST	1099
9	23203	1098

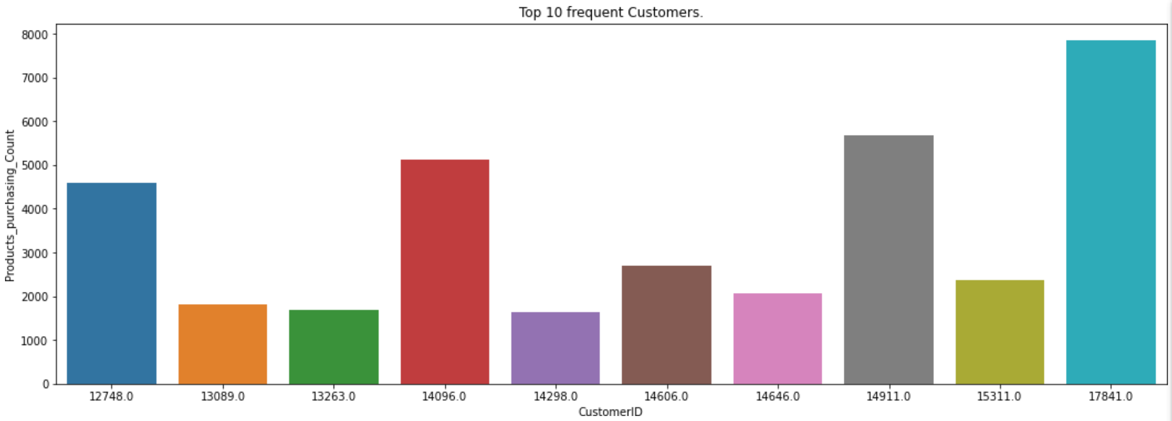
Top 10 Products(Description wise)

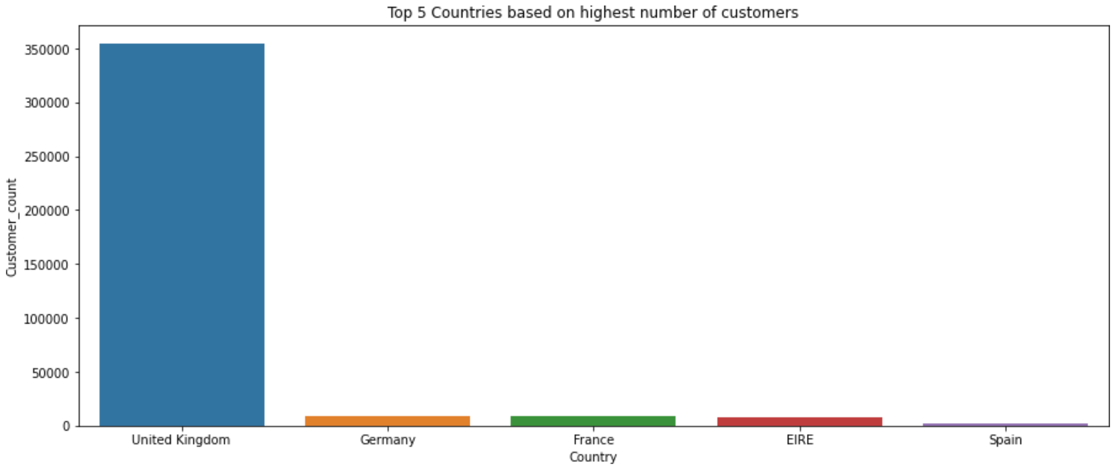
- **WHITE HANGING HEART T- LIGHT HOLDER is the highest selling product almost 2028 units were sold.**
- **REGENCY CAKESTAND 3 TIER is the 2nd highest selling product almost 1723 units were sold.**

Top 10 products(Stock Code wise)

- **StockCode-85123A is the first highest selling product.**
- **StockCode-22423 is the 2nd highest selling product.**

❖EDA(Exploratory Data Analysis):





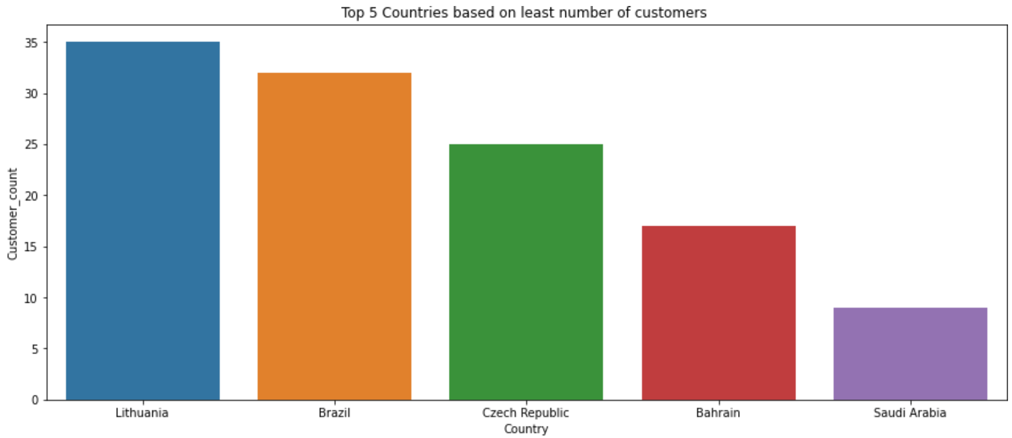
TOP 10 Customers

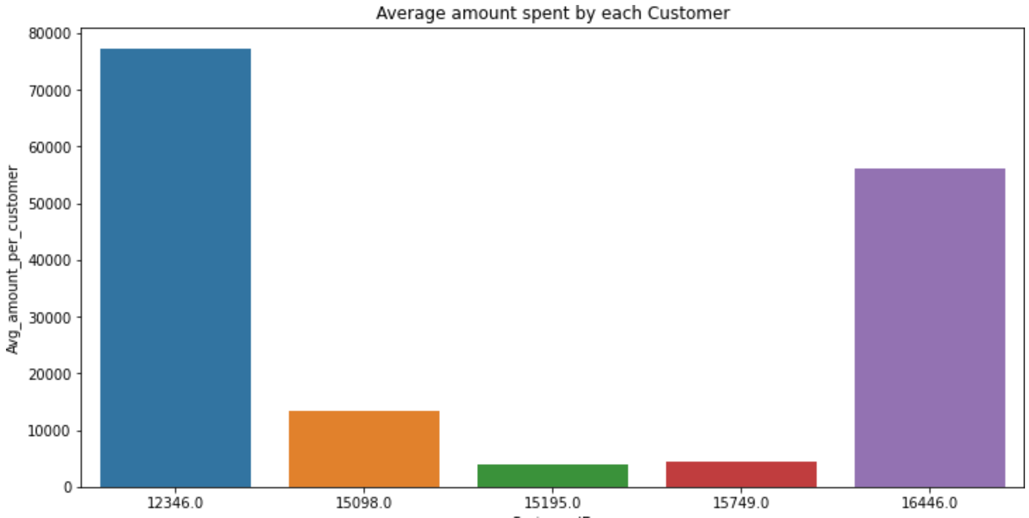
- **CustomerID-17841 had purchased highest number of products.**
- **CustomerID-14911 is the 2nd highest customer who purchased the most the products.**

Top 5 Countries(Based on number of Customers)

- **UK has highest number of customers.**
- **Germany, France and Ireland has almost equal number of customers.**

❖EDA(Exploratory Data Analysis):





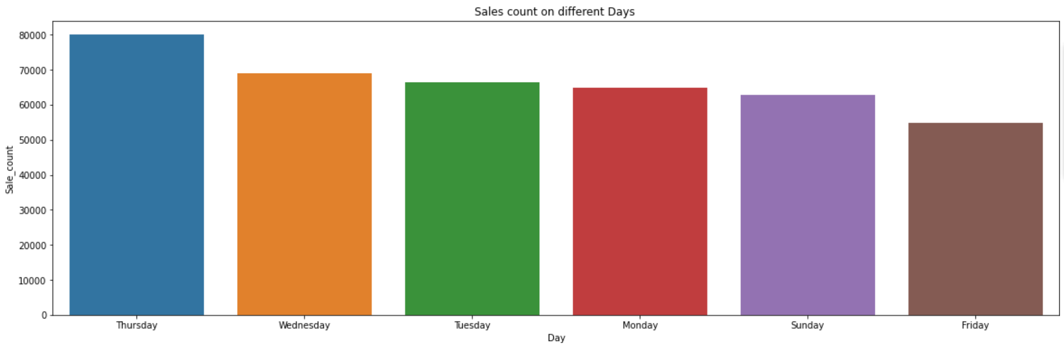
Top 5 Countries(Based on Least number of Customers)

- **There are very less customers from Saudi Arabia.**
- **Bahrain is the 2nd Country having least number of customers.**

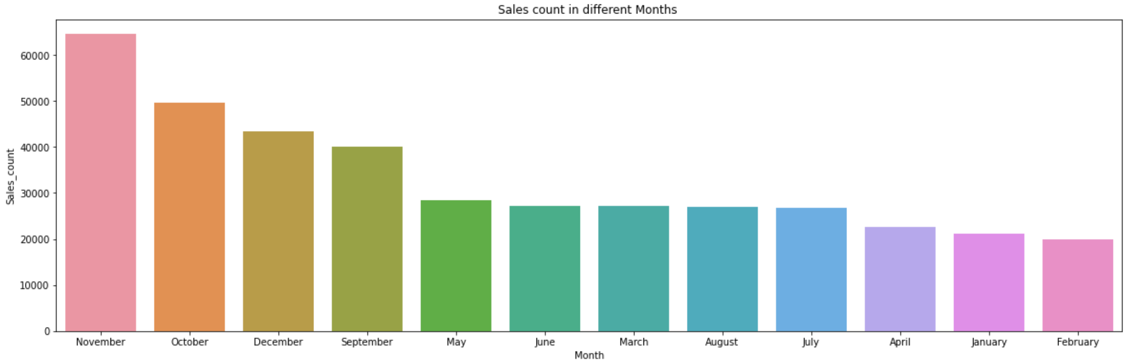
TOP 10 Customers(Avg amount spent by customers)

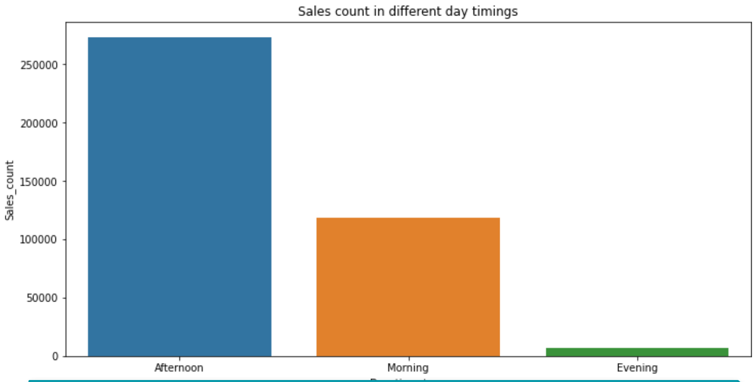
- **77183 (Pounds) is the highest average amount spent by the CustomerID-12346.**
- **56157 (Pounds) is the 2nd highest average amount spent by the CustomerID-16446.**

❖EDA(Exploratory Data Analysis):



- Sales On Thursdays are very high.
- Sales On Fridays are very less.





•Most of the sales happened in November month.
•February Month had least sales.

- Most of the sales happens in the afternoon.
- Least sales happens in the evening.

Model Building:

RFM Model Analysis:

What is RFM?

•RFM is a method used to analyze customer value. RFM stands for RECENCY, Frequency, and Monetary.
•RECENCY: How recently did the customer visit our website or how recently did a customer purchase?
•Frequency: How often do they visit or how often do they purchase?
•Monetary: How much revenue we get from their visit or how much do they spend when they purchase?

Why it is Needed?

RFM Analysis is a marketing framework that is used to understand and analyze customer behavior based on the above three factors RECENCY, Frequency, and Monetary.
The RFM Analysis will help the businesses to segment their customer base into different homogenous groups so that they can engage with each group with different targeted marketing strategies.

Model Building:

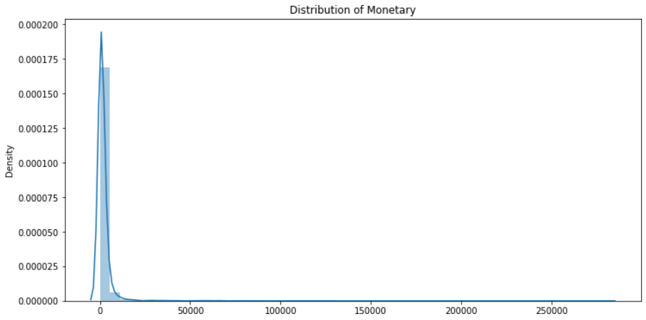
RFM Model Analysis:

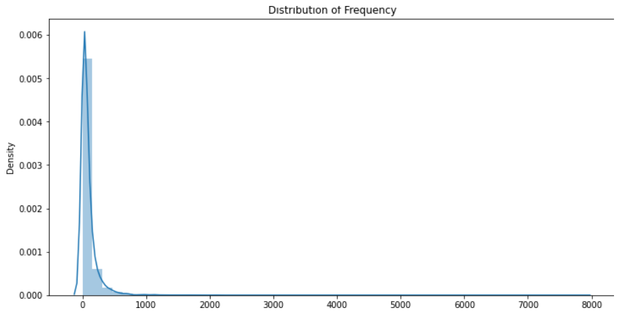
- Recency = Latest Date - Last Invoice Data.
- Frequency = Count of invoice no. of transaction(s).
- Monetary = Sum of Total Amount for each customer.

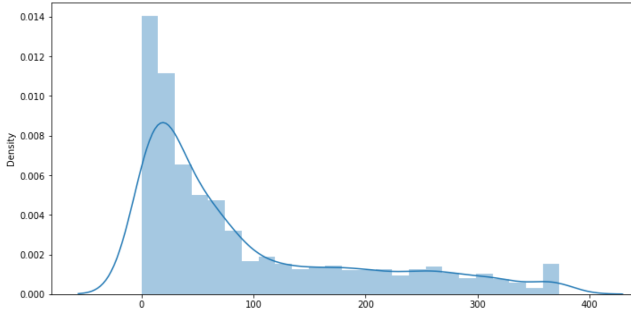
```
quantile
```

```
{'Frequency': {0.25: 17.0, 0.5: 41.0, 0.75: 100.0},
 'Monetary': {0.25: 307.41499999999996,
  0.5: 674.4849999999999,
  0.75: 1661.7400000000002},
 'Recency': {0.25: 17.0, 0.5: 50.0, 0.75: 141.75}}
```

	CustomerID	Recency	Frequency	Monetary	R	F	M	RFM_Group	RFM_Score	RFM_Loyalty_Level
0	14646.0	1	2076	280206.02	1	1	1	111	3	Platinaum
1	18102.0	0	431	259657.30	1	1	1	111	3	Platinaum
2	17450.0	8	337	194550.79	1	1	1	111	3	Platinaum
3	14911.0	1	5675	143825.06	1	1	1	111	3	Platinaum
4	12415.0	24	714	124914.53	2	1	1	211	4	Platinaum
5	14156.0	9	1400	117379.63	1	1	1	111	3	Platinaum
6	17511.0	2	963	91062.38	1	1	1	111	3	Platinaum
7	16029.0	38	242	81024.84	2	1	1	211	4	Platinaum
8	16684.0	4	277	66653.56	1	1	1	111	3	Platinaum
9	14096.0	4	5111	65164.79	1	1	1	111	3	Platinaum





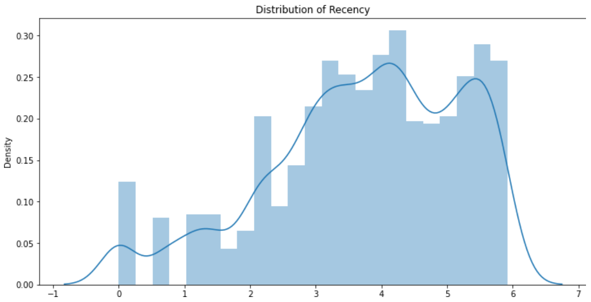


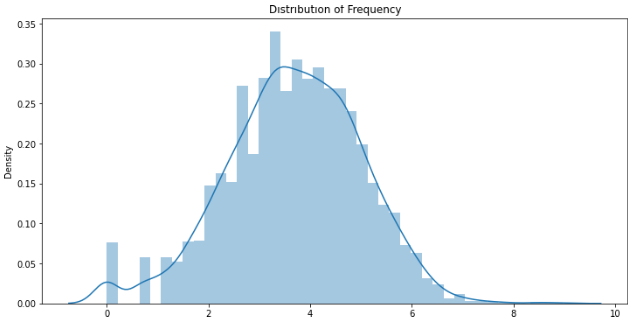
Model Building:

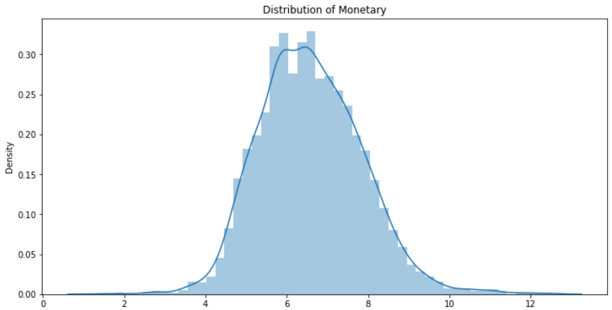
RFM Model Analysis:

.

- Log transformation on Frequency, Recency and Monetary.







Recency

Frequency

Monetary

Model Building:

RFM Model Analysis:

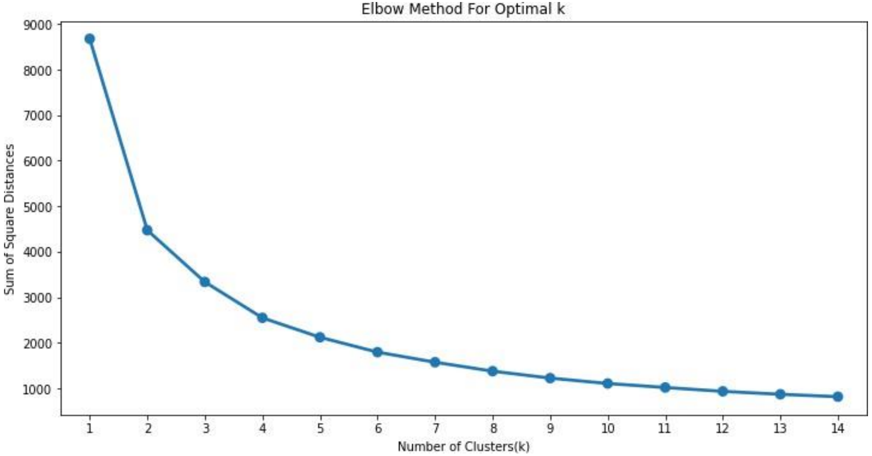
- So just using RFM Model analysis we created 4 clusters namely Platinum, Gold, Silver and Bronze.

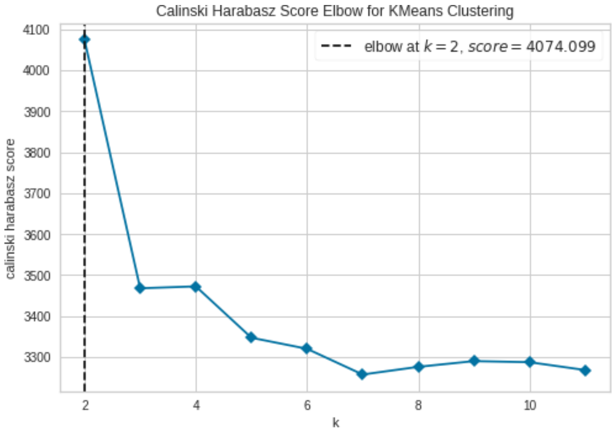
count
1263
1324
981
770

❖ Model Building:

K-means Clustering: (Recency and Monetary)

- Finding the Optimal value of cluster using Elbow method and Silhouette Score.



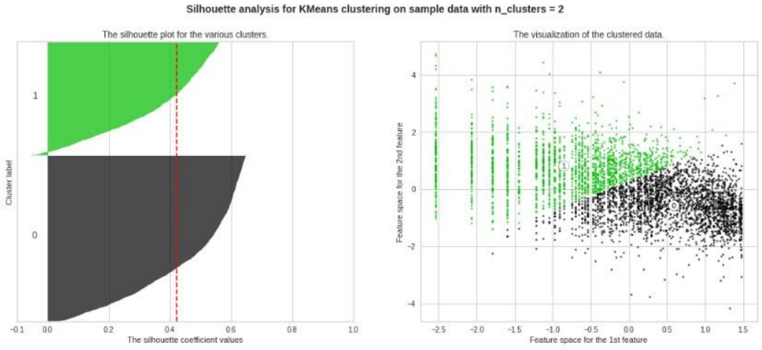


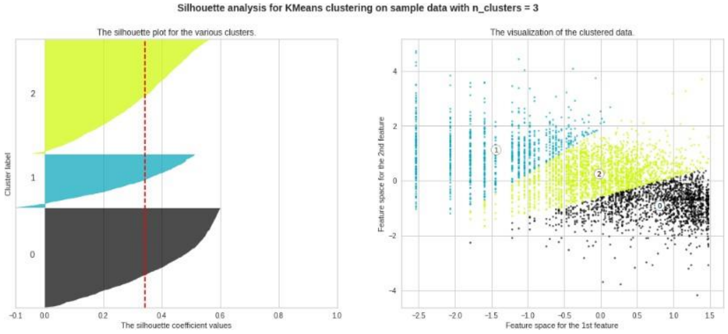

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd94e5ed90>
```

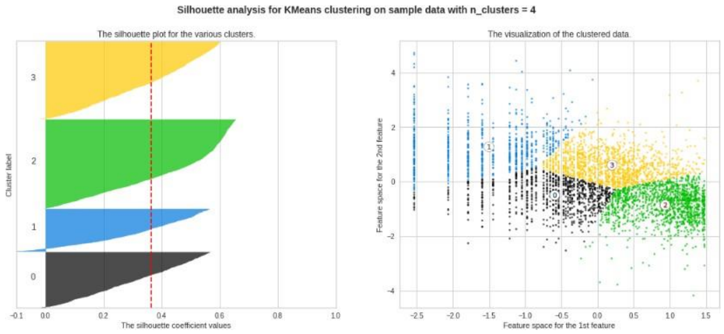
```
For n_clusters = 2 The average silhouette_score is : 0.421461308316105
For n_clusters = 3 The average silhouette_score is : 0.3433470120059089
For n_clusters = 4 The average silhouette_score is : 0.3649058771514865
For n_clusters = 5 The average silhouette_score is : 0.3395250404488943
For n_clusters = 6 The average silhouette_score is : 0.3422201212043055
For n_clusters = 7 The average silhouette_score is : 0.3478708635683099З
For n_clusters = 8 The average silhouette_score is : 0.3377453526486669S
For n_clusters = 9 The average silhouette_score is : 0.345960478941957S
For n_clusters = 10 The average silhouette_score is : 0.3479066146663346
```

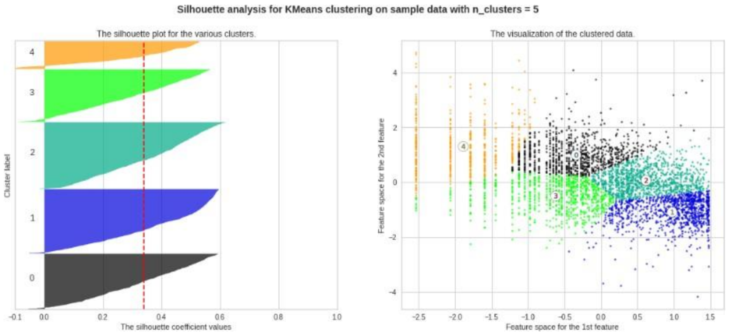
❖ Model Building:

K-means Clustering: (Recency and Monetary)



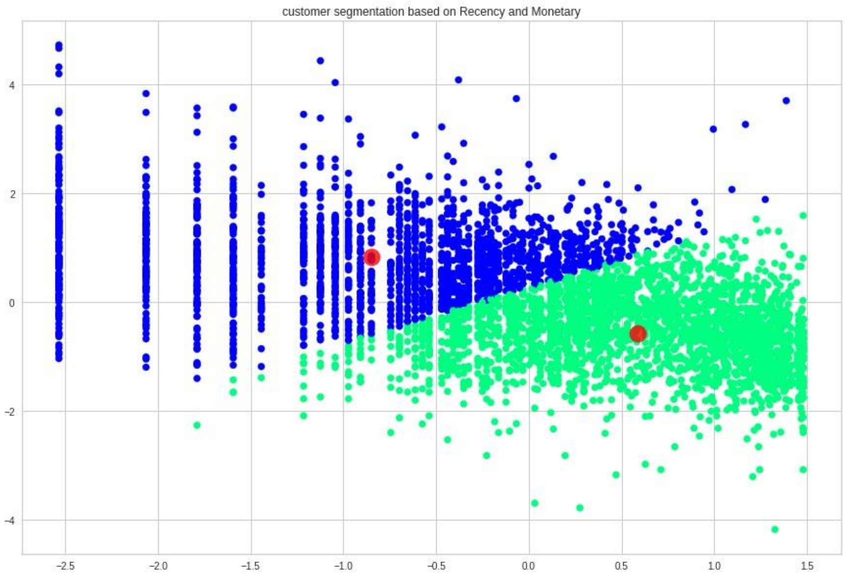




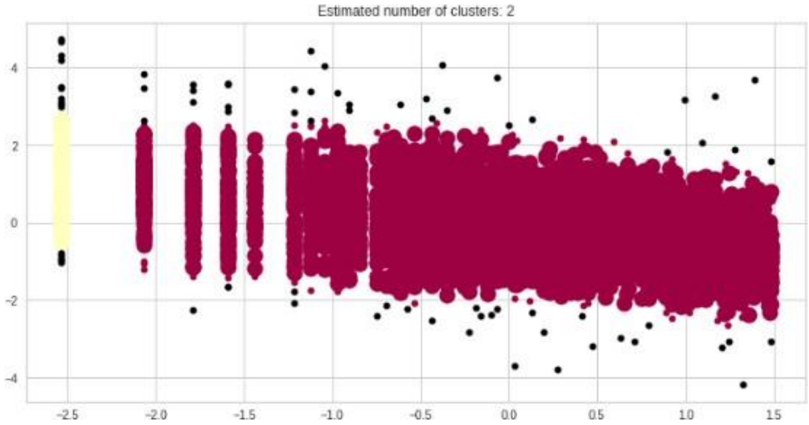


❖ Model Building:

K-means Clustering: (Recency and Monetary)



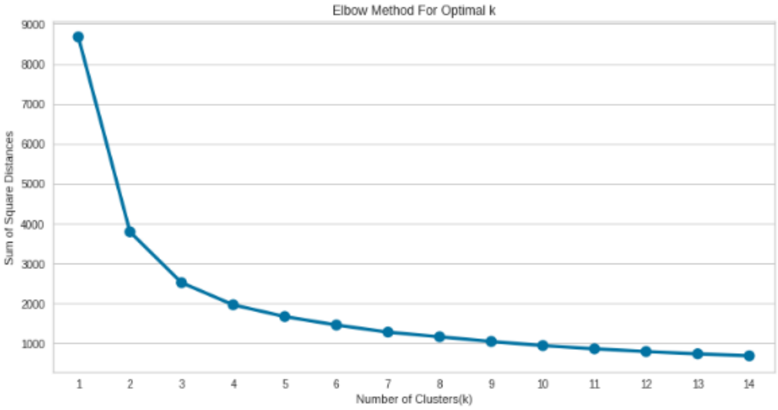
DBSCAN Algorithm (Recency and Monetary)

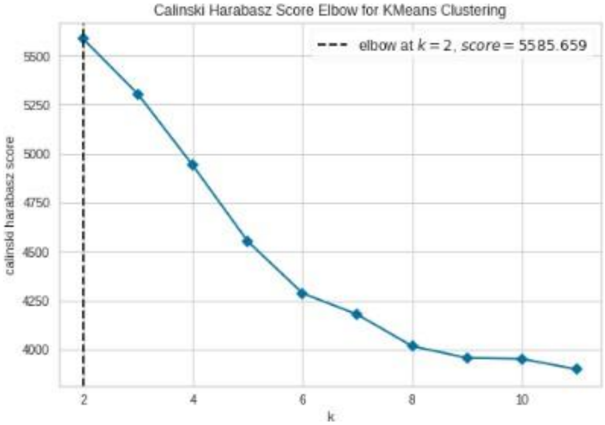


Model Building:

K-means Clustering: (Frequency and Monetary)

- Finding the Optimal value of cluster using Elbow method and Silhouette Score.

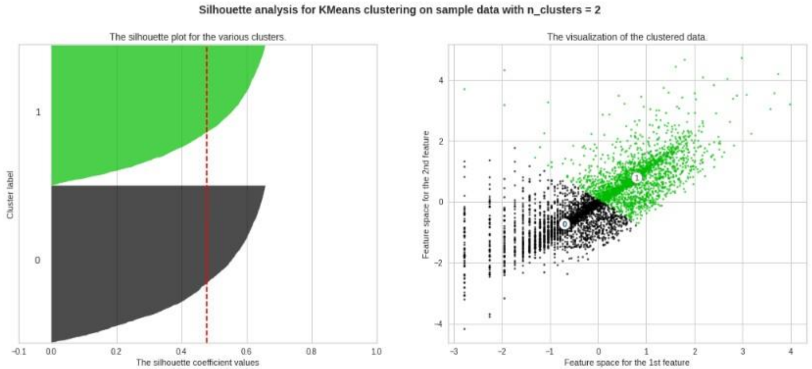


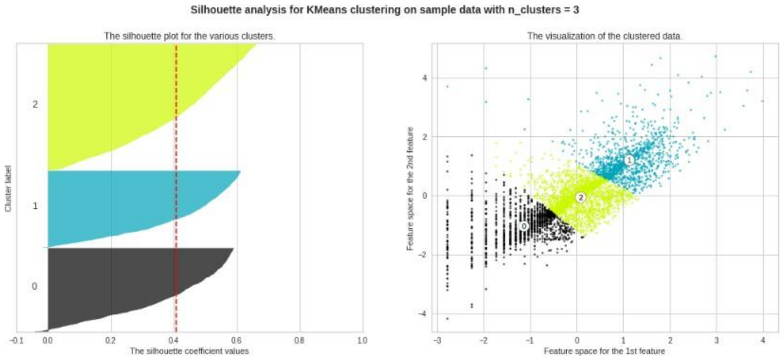


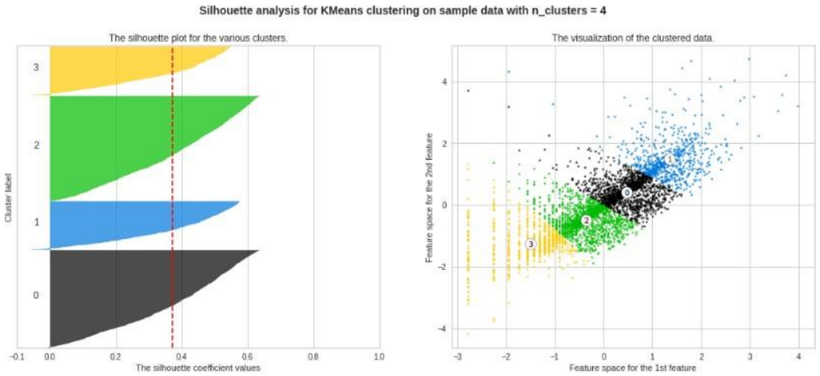

```
For n_clusters = 2, silhouette score is 0.478535709506603
For n_clusters = 3, silhouette score is 0.40764120562174455
For n_clusters = 4, silhouette score is 0.3715810384601166
For n_clusters = 5, silhouette score is 0.3442965607959301
For n_clusters = 6, silhouette score is 0.3586829219947334
For n_clusters = 7, silhouette score is 0.34342098057749704
For n_clusters = 8, silhouette score is 0.3500546906243836
For n_clusters = 9, silhouette score is 0.34419928062567495
For n_clusters = 10, silhouette score is 0.36238664926507114
For n_clusters = 11, silhouette score is 0.3682455762844025
For n_clusters = 12, silhouette score is 0.3534862139672636
For n_clusters = 13, silhouette score is 0.36139542577471895
For n_clusters = 14, silhouette score is 0.3486849890768239
For n_clusters = 15, silhouette score is 0.3628225939841498
```

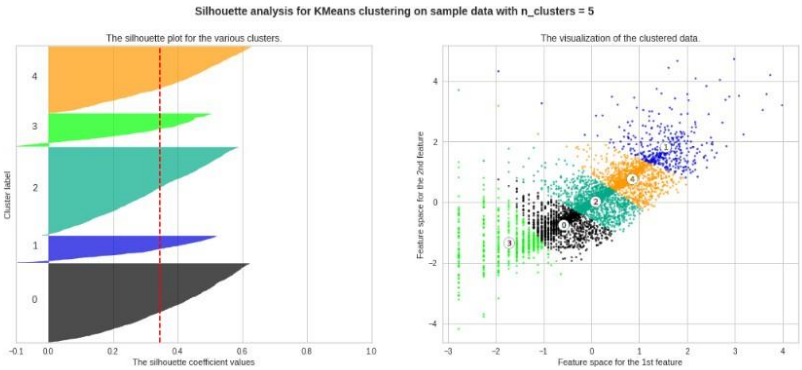
Model Building:

K-means Clustering: (Frequency and Monetary)



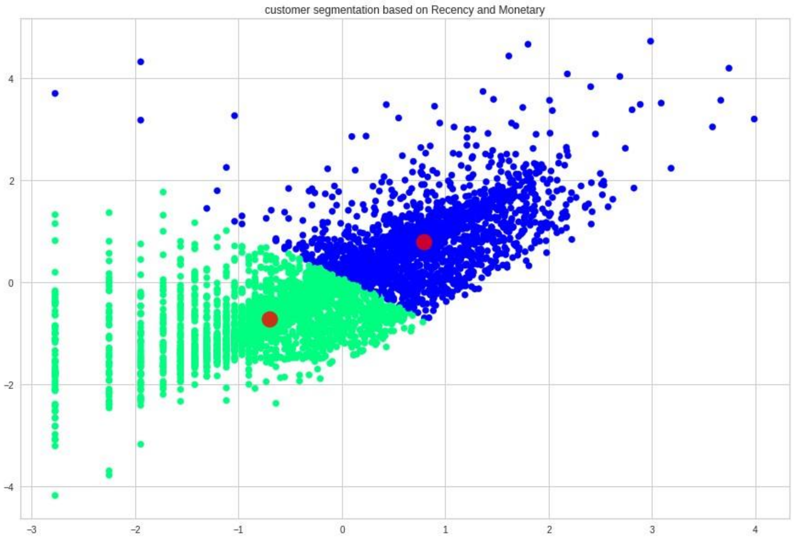




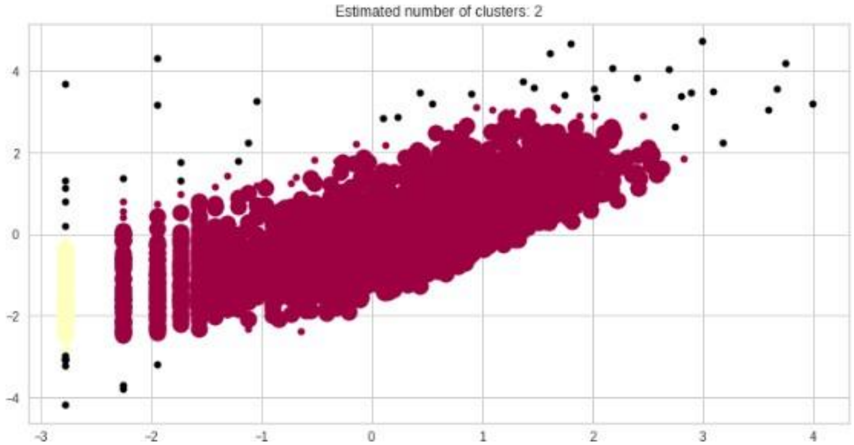


❖ Model Building:

K-means Clustering: (Frequency and Monetary)



DBSCAN Algorithm (Frequency and Monetary)

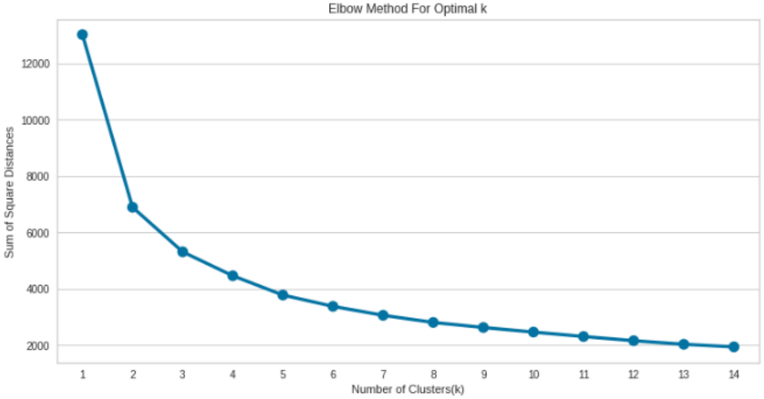


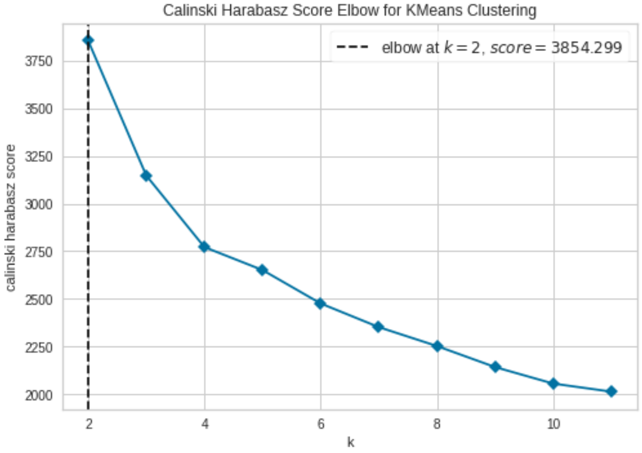
❖ Model Building:

AI

K-means Clustering: (Recency, Frequency and Monetary)

- Finding the Optimal value of cluster using Elbow method and Silhouette Score.



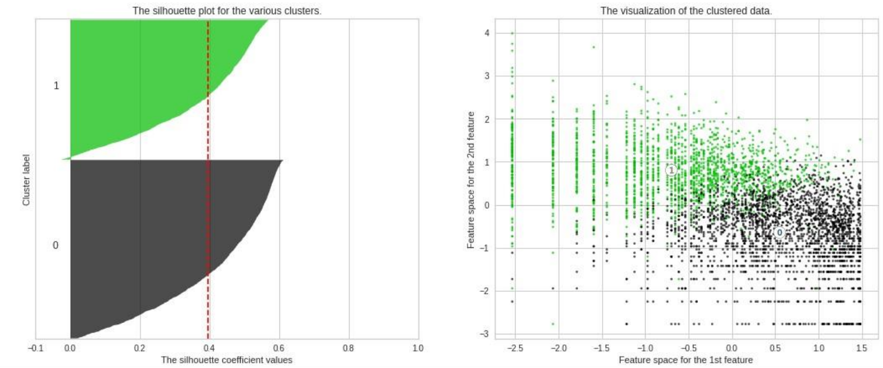



```
For n_clusters = 2, silhouette score is 0.39597280345877467
For n_clusters = 3, silhouette score is 0.30306623428198437
For n_clusters = 4, silhouette score is 0.30188282895683416
For n_clusters = 5, silhouette score is 0.2787783127811271
For n_clusters = 6, silhouette score is 0.2789560652501828
For n_clusters = 7, silhouette score is 0.26251570956441783
For n_clusters = 8, silhouette score is 0.26604516508252274
For n_clusters = 9, silhouette score is 0.25334399829461035
For n_clusters = 10, silhouette score is 0.2594507943913136
For n_clusters = 11, silhouette score is 0.261084644577631
For n_clusters = 12, silhouette score is 0.2630954807140074
For n_clusters = 13, silhouette score is 0.2629821003752366
For n_clusters = 14, silhouette score is 0.26165526187324323
For n_clusters = 15, silhouette score is 0.2561927031281945
```

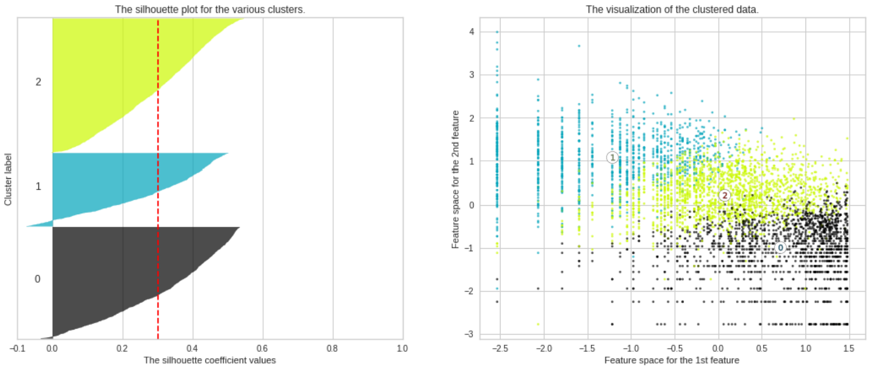
Model Building:

K-means Clustering: (Recency, Frequency and Monetary)

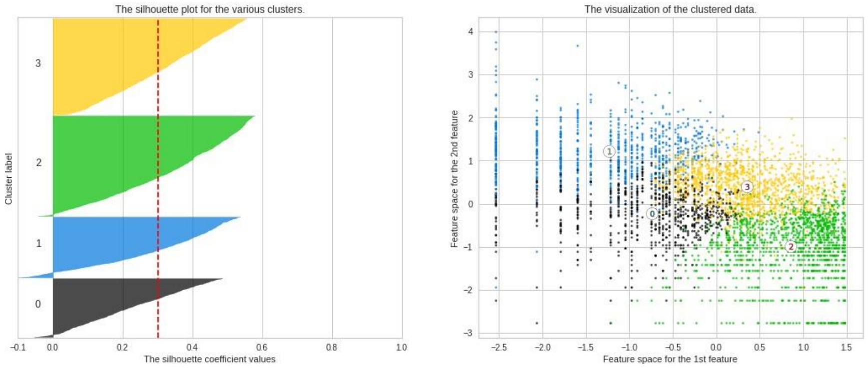
Silhouette analysis for KMeans clustering on sample data with n_clusters = 2



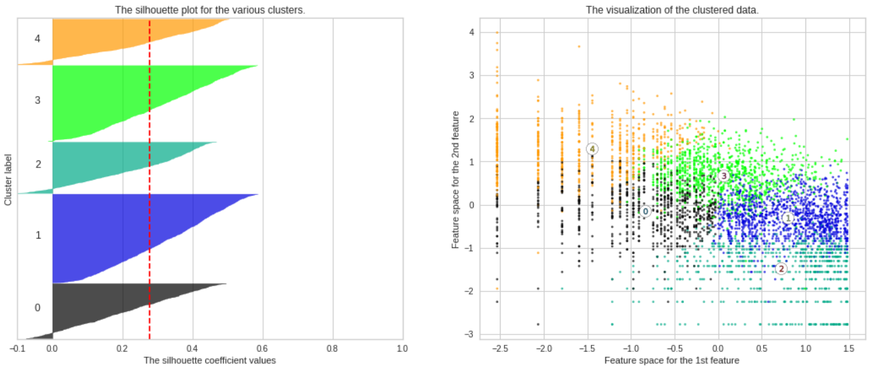
Silhouette analysis for KMeans clustering on sample data with n_clusters = 3



Silhouette analysis for KMeans clustering on sample data with n_clusters = 4

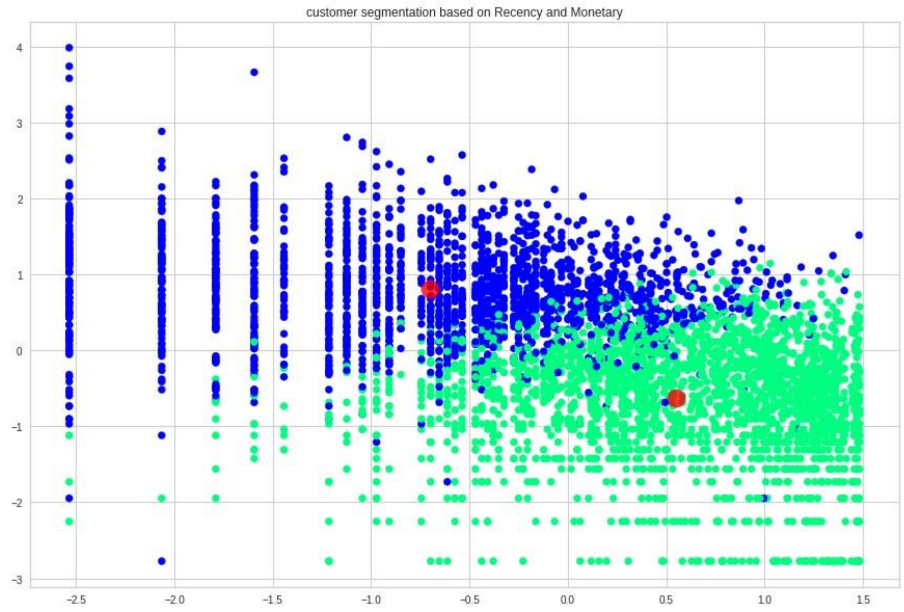


Silhouette analysis for KMeans clustering on sample data with n_clusters = 5

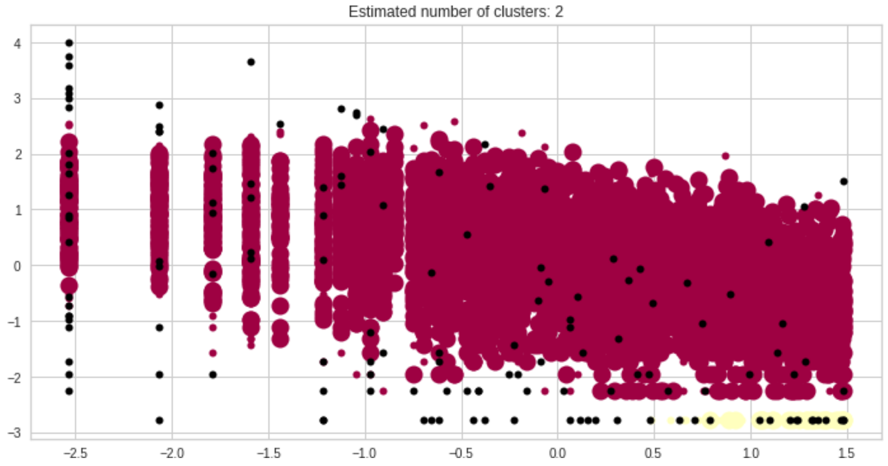


❖ Model Building:

K-means Clustering: (Recency,Frequency and Monetary)

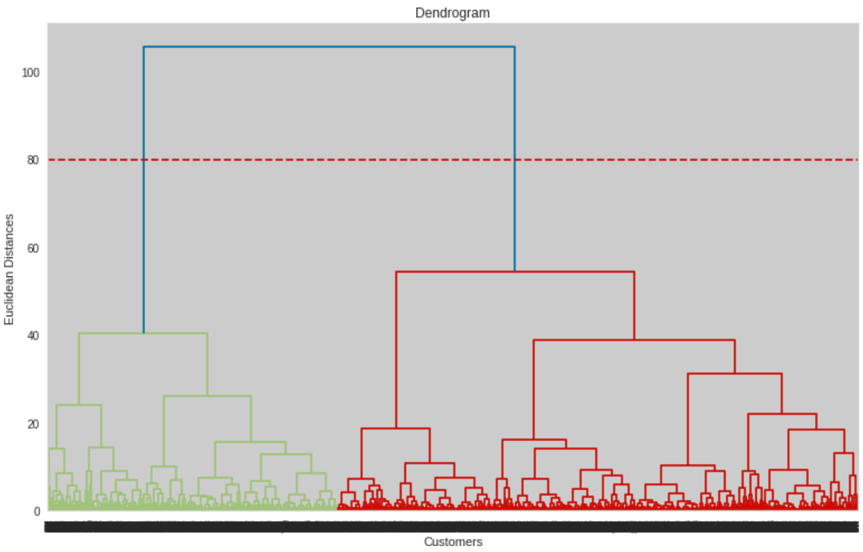


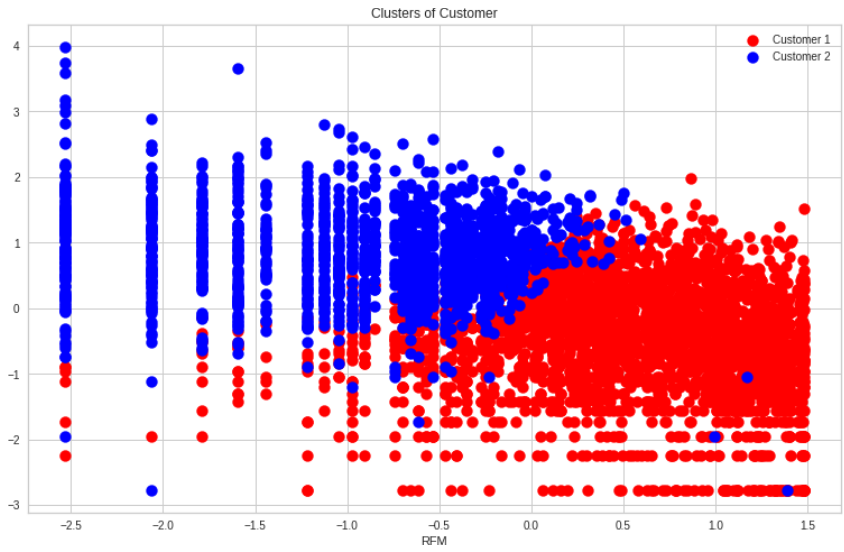
DBSCAN Algorithm (Recency, Frequency and Monetary)



Model Building:

Hierarchical Clustering(Recency, Frequency and Monetary)





Optimal Number of clusters using Dendogram.(Optimal Clusters=2)

❖Summary and Conclusion:

- Firstly we did clustering based on RFM analysis. We had 4 clusters/Segmentation of customers based on RFM score.

	Recency			Frequency			Monetary			
	mean	min	max	mean	min	max	mean	min	max	count
RFM_Loyalty_Level										
Platinaum	19.412510	0	140	228.559778	20	7847	5255.277617	360.93	280206.02	1263
Gold	63.376133	0	372	57.959970	1	543	1169.031202	114.34	168472.50	1324
Silver	126.029562	1	373	24.503568	1	99	583.936944	6.90	77183.60	981
Bronz	217.261039	51	373	10.955844	1	41	199.159506	3.75	660.00	770

- Platinum customers=1263 (less recency but high frequency and heavy spending)
- Gold customers=1324 (good recency, frequency and monetary)
- Silver customers=981(high recency, low frequency and low spending)
- Bronze customers=770 (very high recency but very less frequency and spending)

- *Later we implemented the machine learning algorithms to cluster the customers.*

SL.No	Model Name	Data	Optimal Number of Clusters
1	Kmeans with Elbow method(Elbow Visualizer)	Recency and Monetary	2
2	Kmeans withSilhouette Score method	Recency and Monetary	2
3	DBSCAN	Recency and Monetary	2
4	Kmeans with Elbow method(Elbow Visualizer)	Frequency and Monetary	2
5	Kmeans withSilhouette Score method	Frequency and Monetary	2
6	DBSCAN	Frequency and Monetary	2
7	Kmeans with Elbow method(Elbow Visualizer)	Recency ,Frequency and Monetary	2
8	Kmeans withSilhouette Score method	Recency ,Frequency and Monetary	2
9	DBSCAN	Recency ,Frequency and Monetary	2
10	Hierarchical clustering	Recency ,Frequency and Monetary	2

❖ Summary and Conclusion:

	Recency			Frequency			Monetary			
	mean	min	max	mean	min	max	mean	min	max	count
Cluster_based_on_freq_mon_rec										
0	140.818973	1	373	24.930406	1	168	470.256981	3.75	77183.60	2414
1	30.900208	1	372	175.520790	1	7847	4041.687917	161.03	280206.02	1924

.

- **Above clustering is done with recency, frequency and monetary data(Kmeans Clustering) as all 3 together will provide more information.**
- **Cluster 0 has high recency rate but very low frequency and monetary. Cluster 0 contains 2414 customers.**
- **Cluster 1 has low recency rate but they are frequent buyers and spends very high money than other customers as mean monetary value is very high. Thus generates more revenue to the retail business.**

- ***With this, we are done. Also, we can use more robust analysis for the clustering, using not only RFM but other metrics such as demographics or product features.***

WE ARE DONE HERE !

- **THANK YOU……**