ONLINE RETAIL CUSTOMER SEGMENTATION

SUBMITTED BY:

MUKUND KUMAR

INSTITUTE: INTELLIPAAT

DATE:30TH MAY 2023

Problem statement:

An online retail store is trying to understand the various customer purchase patterns for their firm, you are required to give enough evidence based insights to provide the same.

Dataset Information:

The online_retail.csv contains 387961 rows and 8 columns.

Feature Name	Description
Invoice	Invoice number
StockCode	Product ID
Description	Product Description
Quantity	Quantity of the product
InvoiceDate	Date of the invoice
Price	Price of the product per unit
CustomerID	Customer ID
Country	Region of Purchase

1. Using the above data, find useful insights about the customer purchasing history that can be an added advantage for the online retailer.

2. Segment the customers based on their purchasing behavior

INTRODUCTION:

In this article we are going to made a project on **Online Retail Customer Segmentation or Market Segmentation** in Python by data pre-processing and KMeans Clustering technique ,we will divide the whole data of customers on the basis of RMF i.e. Recency, Monetary and Frequency and we will also visualize these groups on the basis of these 3 terms.

But Before jumping in making the Project directly first know some basic terms-

Customer Segmentation or Market Segmentation

- We can say that Customer Segmentation or Market Segmentation is methodology or marketing practice through which we divide our customer group into various similar sub groups such as on the basis of spending amount, frequency of visit,behaviour , age, gender,e.t.c.
- This helps the companies to know :-

1. Which group of customers are loyal.
2. Which group can spend more money.
3. Which group visit them infrequency.
4. Which group of customers they are loosing.

- Through this companies tries to target the sub groups of customers in retaining them on the basis of their needs and desires by executing various marketing campaigns such as providing special offers, discounts, e.t.c.

In this project we are dividing our customers on the basis of 3 factors

1. **Recency**:- It represents how recently a customer purchased a product.
2. **Frequency**:- It represents how often a customer purchased a product. The more frequent will be the better score.
3. **Monetary**:- It represents how much an customer spends.

Now Lets start building our project

Supervised Machine Learning:

Supervised learning is a machine learning method in which models are trained using labeled data. In supervised learning, models need to find the mapping function to map the input variable (X) with the output variable (Y).

$$Y = f(X)$$

Supervised learning needs supervision to train the model, which is similar to as a student learns things in the presence of a teacher. Supervised learning can be used for two types of problems: **Classification** and **Regression**.

Learn more Supervised Machine Learning

Example: Suppose we have an image of different types of fruits. The task of our supervised learning model is to identify the fruits and classify them accordingly. So to identify the image in supervised learning, we will give the input data as well as output for that, which means we will train the model by the shape, size, color, and taste of each fruit. Once the training is completed, we will test the model by giving the new set of fruit. The model will identify the fruit and predict the output using a suitable algorithm.

Unsupervised Machine Learning:

Unsupervised learning is another machine learning method in which patterns inferred from the unlabeled input data. The goal of unsupervised learning is to find the structure and patterns from the input data. Unsupervised learning does not need any supervision. Instead, it finds patterns from the data by its own.

Learn more Unsupervised Machine Learning

Unsupervised learning can be used for two types of problems: **Clustering** and **Association**.

Example: To understand the unsupervised learning, we will use the example given above. So unlike supervised learning, here we will not provide any supervision to the model. We will just provide the input dataset to the model and allow the model to find the patterns from the data. With the help of a suitable algorithm, the model will train itself and divide the fruits into different groups according to the most similar features between them.

Supervised Learning	Unsupervised Learning
Supervised learning algorithms are trained using labeled data.	Unsupervised learning algorithms are trained using unlabeled data.
Supervised learning model takes direct feedback to check if it is predicting correct output or not.	Unsupervised learning model does not take any feedback.
Supervised learning model predicts the output.	Unsupervised learning model finds the hidden patterns in data.
In supervised learning, input data is provided to the model along with the output.	In unsupervised learning, only input data is provided to the model.
The goal of supervised learning is to train the model so that it can predict the output when it is given new data.	The goal of unsupervised learning is to find the hidden patterns and useful insights from the unknown dataset.
Supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.
Supervised learning can be categorized in **Classification** and **Regression** problems.	Unsupervised Learning can be classified in **Clustering** and **Associations** problems.
Supervised learning can be used for those cases where we know the input as well as corresponding outputs.	Unsupervised learning can be used for those cases where we have only input data and no corresponding output data.
Supervised learning model produces an accurate result.	Unsupervised learning model may give less accurate result as compared to supervised learning.
Supervised learning is not close to true Artificial intelligence as in this, we first train the model for each data, and then only it can predict the correct output.	Unsupervised learning is more close to the true Artificial Intelligence as it learns similarly as a child learns daily routine things by his experiences.
It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.	It includes various algorithms such as Clustering, KNN, and Apriori algorithm.

Hence the problem statement which we have been given is of unsupervised learning.

So will use Kmeans algorithm to get the behaviour of customer and will use python lib like pandas, numpy, matplotlib to extract the useful insights.

```
In [11]: 1 df.isnull().sum()

Out[11]: InvoiceNo          0
         StockCode          0
         Description     1454
         Quantity           0
         InvoiceDate        0
         UnitPrice          0
         CustomerID    135080
         Country            0
         dtype: int64

In [13]: 1 df.isnull().mean()*100

Out[13]: InvoiceNo       0.000000
         StockCode       0.000000
         Description     0.268311
         Quantity        0.000000
         InvoiceDate     0.000000
         UnitPrice       0.000000
         CustomerID     24.926694
         Country         0.000000
         dtype: float64

In [16]: 1 df.keys()

Out[16]: Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
                'UnitPrice', 'CustomerID', 'Country'],
               dtype='object')
```

```
In [17]: 1 df.describe()
```

Out[17]:

	Quantity	UnitPrice	CustomerID
count	541909.000000	541909.000000	406829.000000
mean	9.552250	4.611114	15287.690570
std	218.081158	96.759853	1713.600303
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13953.000000
50%	3.000000	2.080000	15152.000000
75%	10.000000	4.130000	16791.000000
max	80995.000000	38970.000000	18287.000000

Here we have null values in two columns Description and customer_id.

By using the df.describe() function we came to know that our minimum quantity is in negative and we all know that a quantity will never be in Negative. So, we have to remove this redundancy in order to get better accuracy because redundancy can cause miss grouping of data.

DATA PRE-PROCESSING:

#removing the redundancy

```
max    80995.000000    38970.000000    18287.000000
```

```
In [24]:  1  df=df.loc[df["Quantity"]>0]
```

```
In [25]:  1  df.shape
Out[25]: (531285, 8)
```

```
In [26]:  1  df.describe()
```

Out[26]:

	Quantity	UnitPrice	CustomerID
count	531285.000000	531285.000000	397924.000000
mean	10.655262	3.857296	15294.315171
std	156.830323	41.810047	1713.169877
min	1.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13969.000000
50%	3.000000	2.080000	15159.000000
75%	10.000000	4.130000	16795.000000
max	80995.000000	13541.330000	18287.000000

Now here we can see that Invoice date is object type now we have to convert this into datetime for calculating all the values.

```
In [15]:  1  df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 531285 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    531285 non-null  object
 1   StockCode    531285 non-null  object
 2   Description  530693 non-null  object
 3   Quantity     531285 non-null  int64
 4   InvoiceDate  531285 non-null  object
 5   UnitPrice    531285 non-null  float64
 6   CustomerID   397924 non-null  float64
 7   Country      531285 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 36.5+ MB

In [17]:  1  df['InvoiceDate']=pd.to_datetime(df['InvoiceDate'])

In [18]:  1  df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 531285 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    531285 non-null  object
 1   StockCode    531285 non-null  object
 2   Description  530693 non-null  object
 3   Quantity     531285 non-null  int64
 4   InvoiceDate  531285 non-null  datetime64[ns]
 5   UnitPrice    531285 non-null  float64
 6   CustomerID   397924 non-null  float64
 7   Country      531285 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 36.5+ MB
```

from here we now calculate our **Monetary Value**

#calculating our monetary value

df["Sale"] =df.Quantity * df.UnitPrice
#created a column of sale

df.head()

monetary =df.groupby("CustomerID").Sale.sum().reset_index()
"""
Here we are getting our monetary value by grouping customer with their customer id and total no. of sales.

"""

#resetting our index,our monetary has multiindex so we are removing it

In [19]:
```
df.head()
```

Out[19]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

In [20]:
```
df["Sale"]=df.Quantity * df.UnitPrice
```

In [22]:
```
df.head()
```

Out[22]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Sale
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	15.30
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34

```
In [25]: 1 monetary =df.groupby("CustomerID").Sale.sum().reset_index()

In [26]: 1 monetary
```

Out[26]:

	CustomerID	Sale
0	12346.0	77183.60
1	12347.0	4310.00
2	12348.0	1797.24
3	12349.0	1757.55
4	12350.0	334.40
...	...	...
4334	18280.0	180.60
4335	18281.0	80.82
4336	18282.0	178.05
4337	18283.0	2094.88
4338	18287.0	1837.28

4339 rows × 2 columns

Now we will calculate **frequency** of our dataset:

This will tell us, How frequent a customer is Purchasing products.

```
In [36]: 1 frequency=df.groupby("CustomerID").InvoiceNo.count().reset_index()
```

```
In [37]: 1 frequency
```

Out[37]:

	CustomerID	InvoiceNo
0	12346.0	1
1	12347.0	182
2	12348.0	31
3	12349.0	73
4	12350.0	17
...	...	...
4334	18280.0	10
4335	18281.0	7
4336	18282.0	12
4337	18283.0	756
4338	18287.0	70

4339 rows × 2 columns

Now we will calculate our **recency** value

```
#calculating our recency value

LastDate=max(df.InvoiceDate) #calculating the last date of InvoiceDate


LastDate


LastDate = LastDate + pd.DateOffset(days=1)
#adding one to LastDate


LastDate

df["Diff"] = LastDate - df.InvoiceDate

#Diff is the difference between our Lastate and InvoiceData
recency = df.groupby("CustomerID").Diff.min()
"""
here we get our recency value using group by
"""
```

recency = recency.reset_index()

```
In [38]: LastDate=max(df.InvoiceDate)

In [39]: LastDate
Out[39]: Timestamp('2011-12-09 12:50:00')

In [40]: LastDate = LastDate + pd.DateOffset(days=1)

In [41]: LastDate
Out[41]: Timestamp('2011-12-10 12:50:00')

In [42]: df["Diff"] = LastDate - df.InvoiceDate

In [43]: df
```

Out[43]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Sale	Diff
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	15.30	374 days 04:24:00
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34	374 days 04:24:00
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	22.00	374 days 04:24:00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34	374 days 04:24:00
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34	374 days 04:24:00
...	...	...	...	...	...	...	...	...	...	...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France	10.20	1 days 00:00:00
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France	12.60	1 days 00:00:00
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09	4.15	12680.0	France	16.60	1 days 00:00:00

```
In [52]: recency = df.groupby("CustomerID").Diff.min().reset_index()

In [53]: recency
```

Out[53]:

	CustomerID	Diff
0	12346.0	326 days 02:49:00
1	12347.0	2 days 20:58:00
2	12348.0	75 days 23:37:00
3	12349.0	19 days 02:59:00
4	12350.0	310 days 20:49:00
...	...	...
4334	18280.0	278 days 02:58:00
4335	18281.0	181 days 01:57:00
4336	18282.0	8 days 01:07:00
4337	18283.0	4 days 00:48:00
4338	18287.0	43 days 03:21:00

4339 rows × 2 columns

```
In [ ]:
```

Combining all dataframes:

rmf = monetary.merge(frequency, on = "CustomerID")

rmf = rmf.merge(recency, on = "CustomerID")

rmf.columns = ["CustomerID", "Monetary", "Frequence", "Recency"]

rmf

RMF1 = rmf.drop("CustomerID",axis =1)

#dropping customer id and storing it into RMF1

RMF1.Recency = RMF1.Recency.dt.days

In [108]:
```
rmf = monetary.merge(frequency, on = "CustomerID")
rmf
```

Out[108]:

	CustomerID	Sale	InvoiceNo
0	12346.0	77183.60	1
1	12347.0	4310.00	182
2	12348.0	1797.24	31
3	12349.0	1757.55	73
4	12350.0	334.40	17
...	...	...	...
4334	18280.0	180.60	10
4335	18281.0	80.82	7
4336	18282.0	178.05	12
4337	18283.0	2094.88	756
4338	18287.0	1837.28	70

4339 rows × 3 columns

In [109]:
```
rmf=rmf.merge(recency , on="CustomerID")
```

In [113]:
```
rmf.columns = ["CustomerID", "Monetary", "Frequence", "Recency"]
```

In [114]:
```
rmf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4339 entries, 0 to 4338
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   CustomerID  4339 non-null   float64
 1   Monetary    4339 non-null   float64
 2   Frequence   4339 non-null   int64
 3   Recency     4339 non-null   timedelta64[ns]
dtypes: float64(2), int64(1), timedelta64[ns](1)
memory usage: 169.5 KB
```

In [118]:
```
RMF1 = rmf.drop("CustomerID",axis =1)

#dropping customer id and storing it into RMF1


RMF1.Recency = RMF1['Recency'].dt.days
```

In [119]:
```
RMF1
```

Out[119]:

	Monetary	Frequence	Recency
0	77183.60	1	326
1	4310.00	182	2
2	1797.24	31	75
3	1757.55	73	19
4	334.40	17	310
...	...	...	...
4334	180.60	10	278
4335	80.82	7	181
4336	178.05	12	8
4337	2094.88	756	4
4338	1837.28	70	43

4339 rows × 3 columns

Our Data pre-processing part ends here now we will perform the analysis of or data.

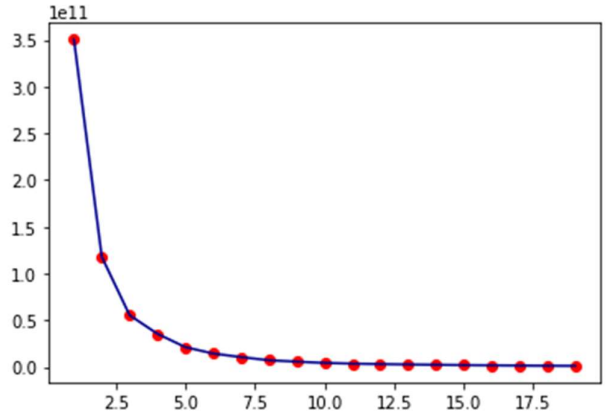
we will perform analysis of data using KMeans algorithm.

In [134]:
```
from sklearn.cluster import KMeans
```

In [135]:
```
ssd = []
for k in range(1,20):


    km = KMeans(n_clusters=k)

    km.fit(RMF1)

    ssd.append(km.inertia_)
```

In [136]:
```
plt.plot(np.arange(1,20), ssd,color="darkblue")


plt.scatter(np.arange(1,20), ssd,color="red")
```

Out[136]: <matplotlib.collections.PathCollection at 0x1fcf1df8f40>



In the KMean algo we are using elbow method to find the no. of clustering groups.

We will perform clustering now onwards

In [137]:
```
model = KMeans(n_clusters=5)


ClusterID = model.fit_predict(RMF1)


ClusterID


RMF1["ClusterID"] = ClusterID
```

In [138]:
```
RMF1
```

Out[138]:

	Monetary	Frequence	Recency	ClusterID
0	77183.60	1	326	2
1	4310.00	182	2	0
2	1797.24	31	75	0
3	1757.55	73	19	0
4	334.40	17	310	0
...	...	...	...	...
4334	180.60	10	278	0
4335	80.82	7	181	0
4336	178.05	12	8	0
4337	2094.88	756	4	0
4338	1837.28	70	43	0

4339 rows × 4 columns

VISUALISATION:

In [140]:
```
km_cluster_sale =RMF1.groupby("ClusterID").Monetary.mean()


km_cluster_Recency =RMF1.groupby("ClusterID").Recency.mean()


km_cluster_Frequence =RMF1.groupby("ClusterID").Frequence.mean()
```

In [141]:
```
km_cluster_sale

```

Out[141]:
```
ClusterID
0      1049.274575
1    149828.502000
2     51858.727500
3    269931.660000
4     10022.790242
Name: Monetary, dtype: float64
```

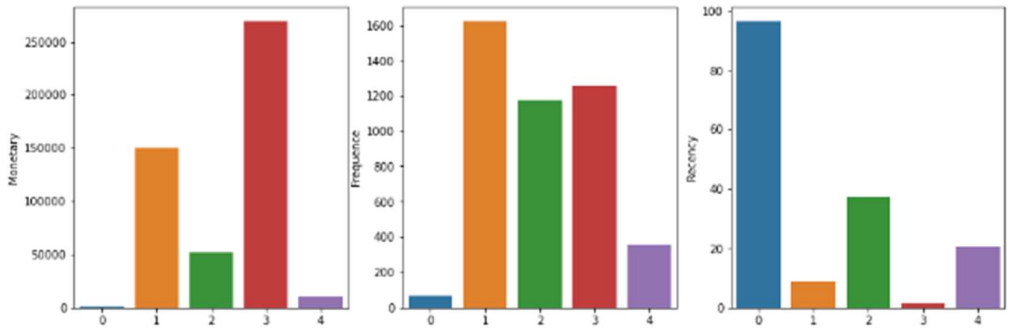
In [142]:
```
km_cluster_Recency
```

Out[142]:
```
ClusterID
0    96.622043
1     8.800000
2    37.250000
3     1.500000
4    20.526570
Name: Recency, dtype: float64
```

In [146]:

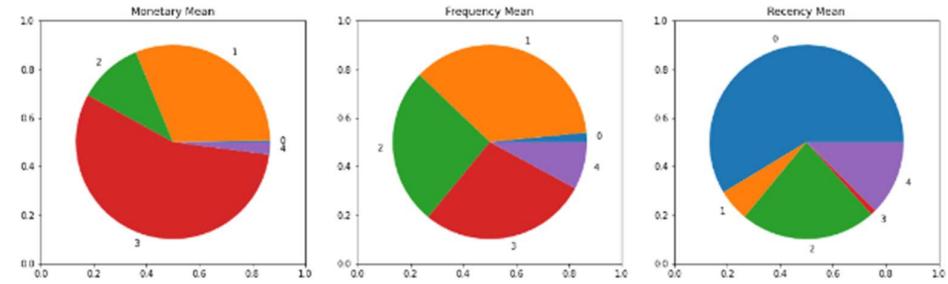
```
import seaborn as sns

#first we are plotting bar chart

fig, axs = plt.subplots(1,3, figsize = (15, 5))


sns.barplot(x = [0,1,2,3,4],  y = km_cluster_sale , ax = axs[0])


sns.barplot(x = [0,1,2,3,4],  y = km_cluster_Frequence , ax = axs[1])
sns.barplot(x = [0,1,2,3,4],  y = km_cluster_Recency , ax = axs[2])
```

Out[146]: <AxesSubplot:ylabel='Recency'>



In [147]:

```
fig,axis = plt.subplots(1,3, figsize =(18,5))
ax1 =fig.add_subplot(1,3,1)
plt.title("Monetary Mean")
ax1.pie(km_cluster_sale, labels =[0,1,2,3,4])
ax1 =fig.add_subplot(1,3,2)
plt.title("Frequency Mean")
ax1.pie(km_cluster_Frequence, labels =[0,1,2,3,4])
ax1 =fig.add_subplot(1,3,3)
plt.title("Recency Mean")
ax1.pie(km_cluster_Recency, labels =[0,1,2,3,4])



#ax1.axis("off")

plt.show()
```



from the above pie chart we can easily understand our 5 groups according to Recency mean,Frequency mean and Monetary mean.

Group 1 is the group of customer who spends maximum amount of money and also has a good frequency and low recency rate.Group 4 are the customers whose frequency rate is maximum and monetary value is also good and recency rate is also quite good, whereas Group 0 is the group of customers who has a very high recency rate means they have not purchased anything from the past.

Thank you!!