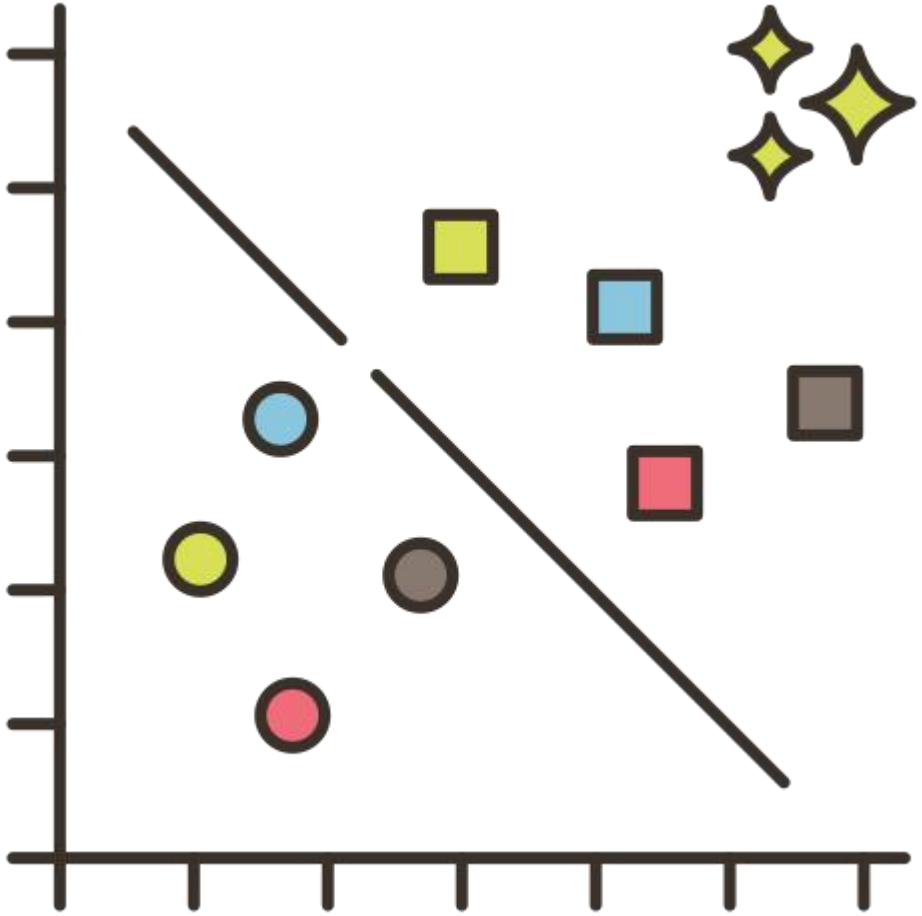
K-Means Clustering Case study

Online Retail Customer Segmentation



Agenda

01 Overview of business problem

02 Approach towards problem

03 Data understanding

04 Data Reading

05 Data Cleansing

06 Data Preparation

07 Outliers Detection

08 Scaling

09 Building the model

10 Cluster profiling

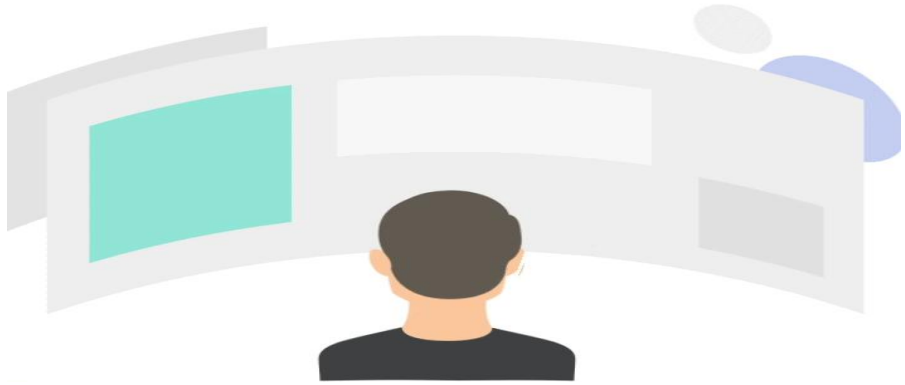
Overview of business problem

Retail giants wants to understand that different segment of customers based on the past transaction. They also want to understand who are the most valued customer in terms value and volume of transaction.

This information will help in effective campaigning and marketing at a later stage.

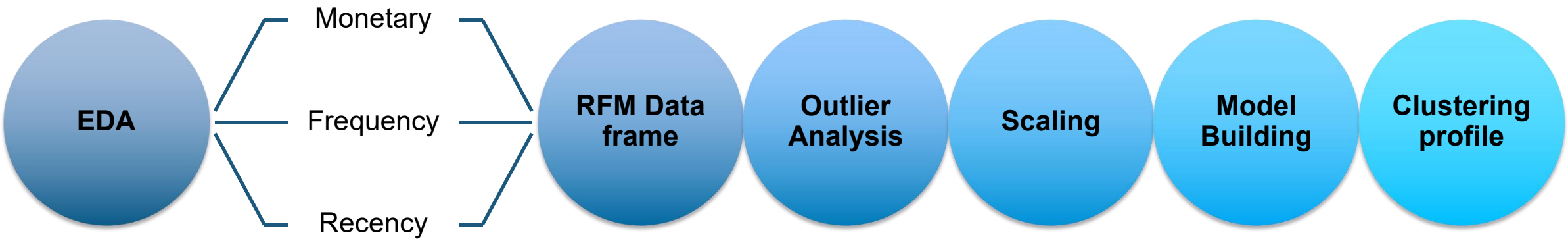
Approach towards problem

The strategy is to understand the recency, frequency and monetary value of the customer, based on their past transaction and we leverage these features as an input to k-means clustering algorithm. We will use these features to identify segments of customers.



After getting segments we will do the profiling of the customer. And identify which segment is basically the most recent transactor, most frequent transactor, and most monitory. And this will be communicated to the campaign marketing department.

Solution Flow



Data Understanding.

Data Reading

Import the dataset to your console and read the data.

Data cleansing

From the dataset estimating the total contribution of null/NAN values ,after estimating removing the all null/NAN values through dropna function.

```
[ ] # Calculating the Missing Values % contribution in DF

    df_null = round(100*(retail.isnull().sum())/len(retail), 2)
    df_null
```

```
InvoiceNo       0.00
StockCode       0.00
Description     0.27
Quantity        0.00
InvoiceDate     0.00
UnitPrice       0.00
CustomerID     24.93
Country         0.00
dtype: float64
```

```
    # Droping rows having missing values

    retail = retail.dropna()
    retail.shape
```

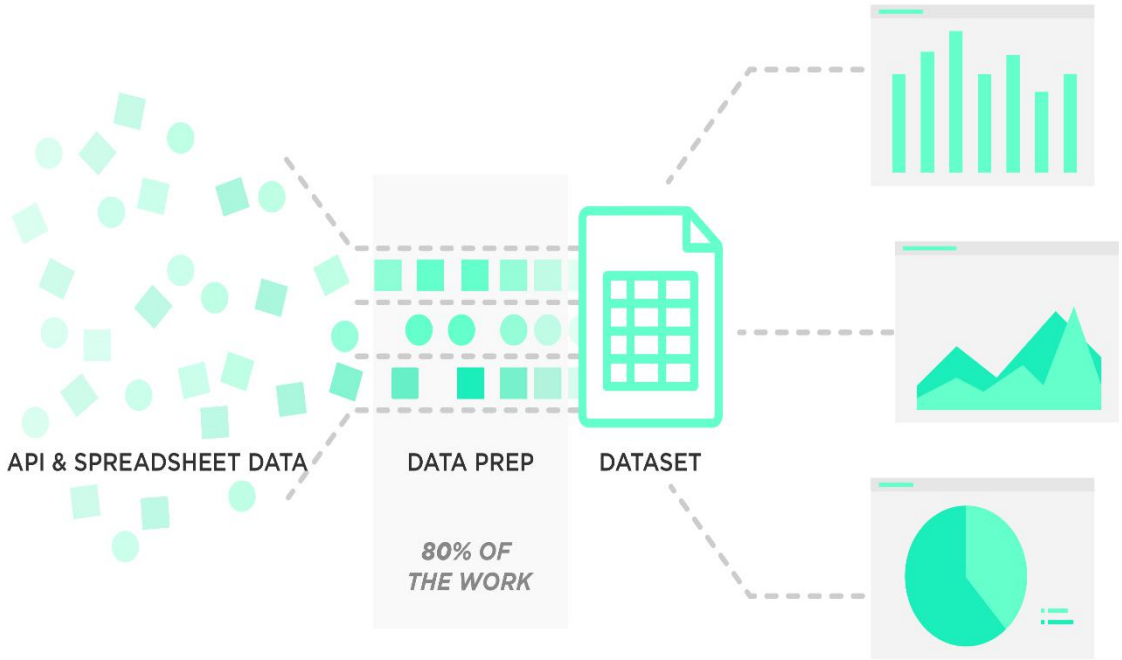
```
(406829, 8)
```

```
[ ] # Changing the datatype of Customer Id as per Business understanding

    retail['CustomerID'] = retail['CustomerID'].astype(str)
```

Data preparation

We are going to analyze the customers based on below 3 factors:
R (Recency): Number of days since last purchase of customer.
F (Frequency): Number of transactions of customers.
M (Monetary): Total amount of transactions (revenue contributed by each customer).



Monetary

- **In monetary, the main goal is to estimate the total amount of transaction. To do so we will calculate the total amount by augmenting quantity and unit price.**

```
# New Attribute : Monetary

retail['Amount'] = retail['Quantity']*retail['UnitPrice']
rfm_m = retail.groupby('CustomerID')['Amount'].sum()
rfm_m = rfm_m.reset_index()
rfm_m.head()
```

Frequency

- **In frequency, the main goal is to estimate the total number of transaction. To do so we will calculate the total transaction by grouping the CustomerId with InvoiceNo.**

```
# New Attribute : Frequency

rfm_f = retail.groupby('CustomerID')['InvoiceNo'].count()
rfm_f = rfm_f.reset_index()
rfm_f.columns = ['CustomerID', 'Frequency']
rfm_f.head()
```

Recency

- In recency, the main goal is to estimate the total number of days since last transaction. To do so we will compute maximum date(last transaction date in dataset) .
- And estimate the difference between maximum date and invoice date, so we will get the count of last transaction date .

```
# New Attribute : Recency

# Convert to datetime to proper datatype

retail['InvoiceDate'] = pd.to_datetime(retail['InvoiceDate'],format='%m/%d/%Y %H:%M')
```

```
# Compute the maximum date to know the last transaction date

max_date = max(retail['InvoiceDate'])
max_date
```

```
Timestamp('2011-12-09 12:50:00')
```

```
# Compute the difference between max date and transaction date

retail['Diff'] = max_date - retail['InvoiceDate']
retail.head()
```

```
# Extract number of days only

rfm_p['Diff'] = rfm_p['Diff'].dt.days
rfm_p.head()
```

	CustomerID	Diff
0	12346.0	325
1	12347.0	1
2	12348.0	74
3	12349.0	18
4	12350.0	309

Data preparation

- **Finally merge Recency ,Frequency ,Monetary tables and create RFM Dataframe.**

```
# Merge tha dataframes to get the final RFM dataframe

rfm = pd.merge(rfm, rfm_p, on='CustomerID', how='inner')
rfm.columns = ['CustomerID', 'Amount', 'Frequency', 'Recency']
rfm.head()
```

	CustomerID	Amount	Frequency	Recency
0	12346.0	0.00	2	325
1	12347.0	4310.00	182	1
2	12348.0	1797.24	31	74
3	12349.0	1757.55	73	18
4	12350.0	334.40	17	309

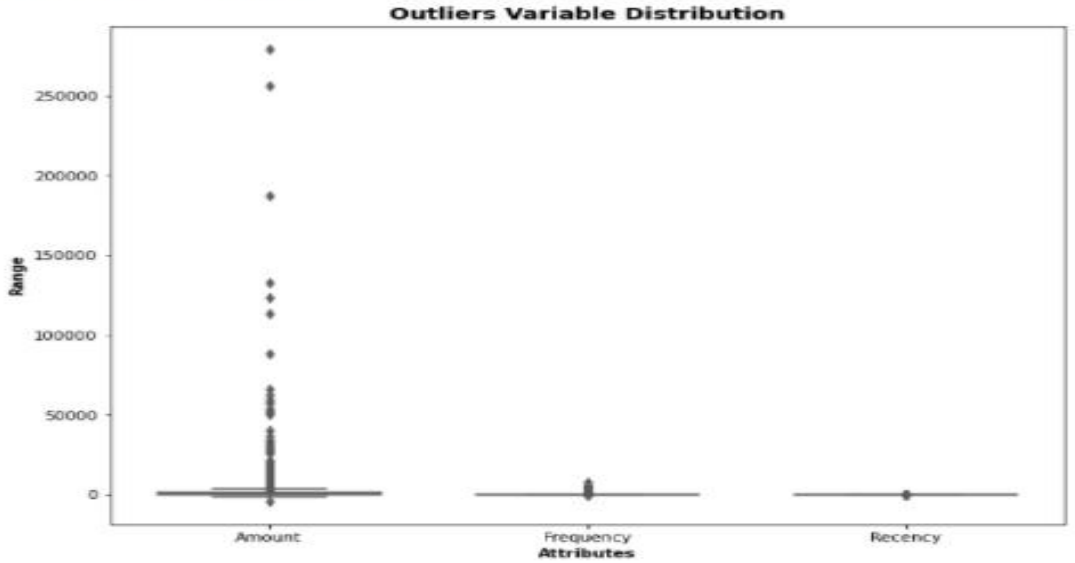
Outliers Analysis

- The process of discovering the abnormal observation in a dataset is referred to as outlier analysis.
- So to find the outliers in RFM Data Frame ,we will plot the Boxplot on RFM Data Frame.

```
# Outlier Analysis of Amount Frequency and Recency

attributes = ['Amount','Frequency','Recency']
plt.rcParams['figure.figsize'] = [10,8]
sns.boxplot(data = rfm[attributes], orient="v", palette="Set2" ,whis=1.5,saturation=1, width=0.7)
plt.title("Outliers Variable Distribution", fontsize = 14, fontweight = 'bold')
plt.ylabel("Range", fontweight = 'bold')
plt.xlabel("Attributes", fontweight = 'bold')
```

```
Text(0.5, 0, 'Attributes')
```



Outliers Analysis

- Outliers can be estimated by finding the Inter Quantile Range(IQR) / difference between 1st quantile and 3rd quantile .
- From IQR value remove the outliers from RFM Data Frame.

```
# Removing (statistical) outliers for Amount
Q1 = rfm.Amount.quantile(0.05)
Q3 = rfm.Amount.quantile(0.95)
IQR = Q3 - Q1
rfm = rfm[(rfm.Amount >= Q1 - 1.5*IQR) & (rfm.Amount <= Q3 + 1.5*IQR)]

# Removing (statistical) outliers for Recency
Q1 = rfm.Recency.quantile(0.05)
Q3 = rfm.Recency.quantile(0.95)
IQR = Q3 - Q1
rfm = rfm[(rfm.Recency >= Q1 - 1.5*IQR) & (rfm.Recency <= Q3 + 1.5*IQR)]

# Removing (statistical) outliers for Frequency
Q1 = rfm.Frequency.quantile(0.05)
Q3 = rfm.Frequency.quantile(0.95)
IQR = Q3 - Q1
rfm = rfm[(rfm.Frequency >= Q1 - 1.5*IQR) & (rfm.Frequency <= Q3 + 1.5*IQR)]
```

Scaling

- Scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.
- Using standardization ,we will rescale the RFM Data Frame.

```
# Rescaling the attributes

rfm_df = rfm[['Amount', 'Frequency', 'Recency']]

# Instantiate
scaler = StandardScaler()

# fit_transform
rfm_df_scaled = scaler.fit_transform(rfm_df)
rfm_df_scaled.shape
```

```
(4293, 3)
```

```
rfm_df_scaled = pd.DataFrame(rfm_df_scaled)
rfm_df_scaled.columns = ['Amount', 'Frequency', 'Recency']
rfm_df_scaled.head()
```

	Amount	Frequency	Recency
0	-0.723738	-0.752888	2.301611
1	1.731617	1.042467	-0.906466
2	0.300128	-0.463636	-0.183658
3	0.277517	-0.044720	-0.738141
4	-0.533235	-0.603275	2.143188

Building the Model

Step 1: Select some arbitrary value of K and fit the rescaled data. According to the value of K ,get the labels for rescaled data.

```
# k-means with some arbitrary k

kmeans = KMeans(n_clusters=4, max_iter=50)
kmeans.fit(rfm_df_scaled)
```

```
KMeans(max_iter=50, n_clusters=4)
```

```
kmeans.labels_
```

```
array([1, 2, 1, ..., 1, 0, 2], dtype=int32)
```

Building the Model

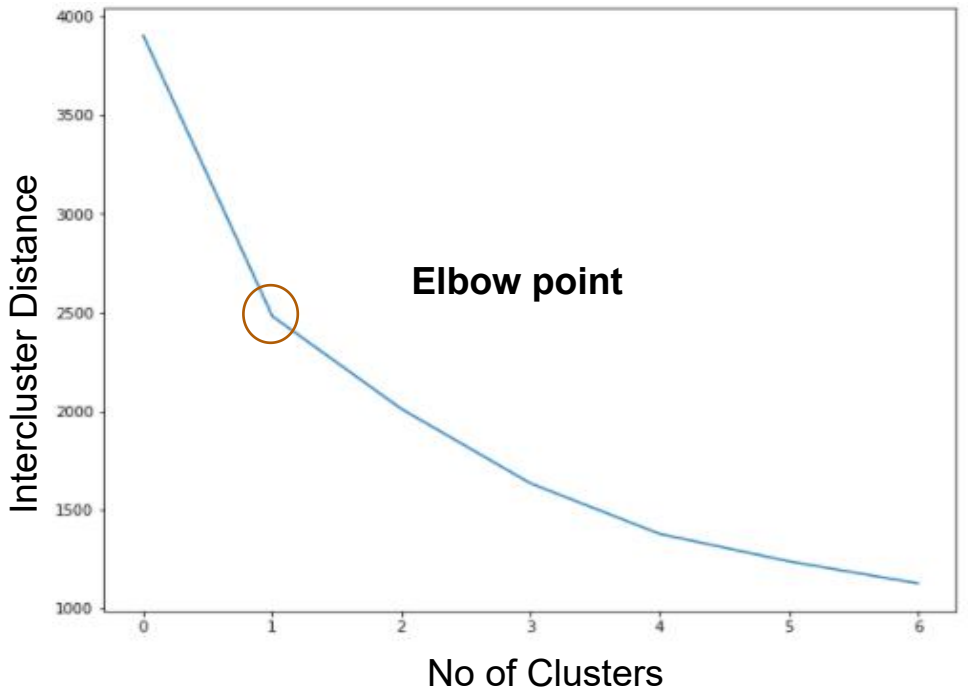
Step 2: Finding optimum number of Clusters , this can be achieved through Elbow method. Calculating the sum of square distance from centroid to estimate a optimal cluster value is known as Elbow method.

```
# Elbow-curve/SSD
# inertia
# Sum of squared distances of samples to their closest cluster center.

ssd = []
range_n_clusters = [2, 3, 4, 5, 6, 7, 8]
for num_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=num_clusters, max_iter=50)
    kmeans.fit(rfm_df_scaled)

    ssd.append(kmeans.inertia_)

# plot the SSDs for each n_clusters
plt.plot(ssd)
```



Building the Model

Step 3: From Optimum Value of K ,Refit the RFM Data Frame and Plot BoxPlot for all 3 Labels (3 Labels will be Recency , Frequency and Monetary) .

```
# Final model with k=3
kmeans = KMeans(n_clusters=3, max_iter=50)
kmeans.fit(rfm_df_scaled)
```

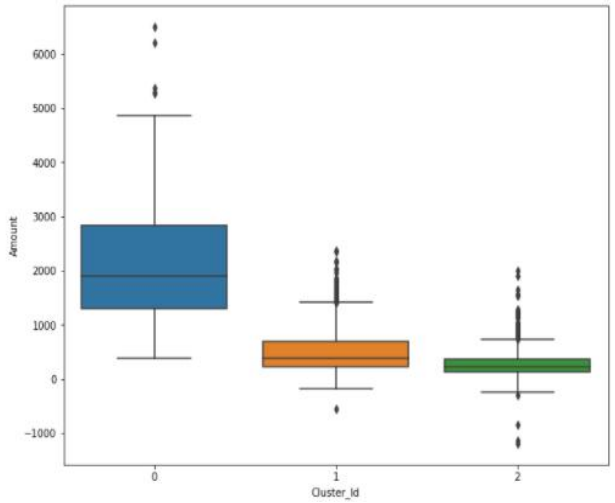
```
KMeans(max_iter=50, n_clusters=3)
```

```
 kmeans.labels_
```

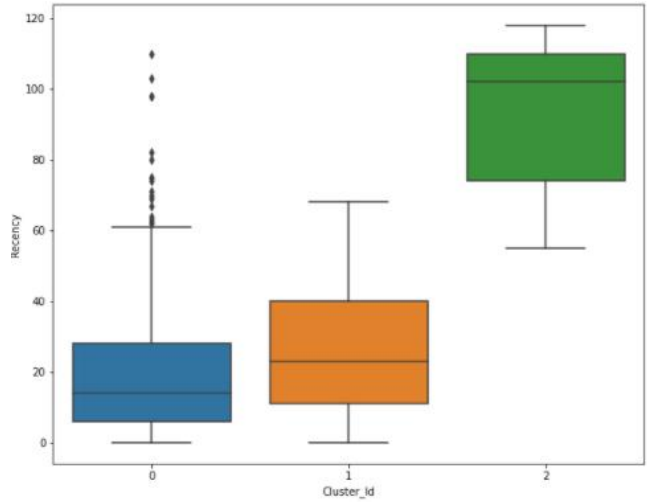
```
array([2, 1, 1, ..., 2, 1, 1], dtype=int32)
```

```
# assign the label
rfm['Cluster_Id'] = kmeans.labels_
rfm.head()
```

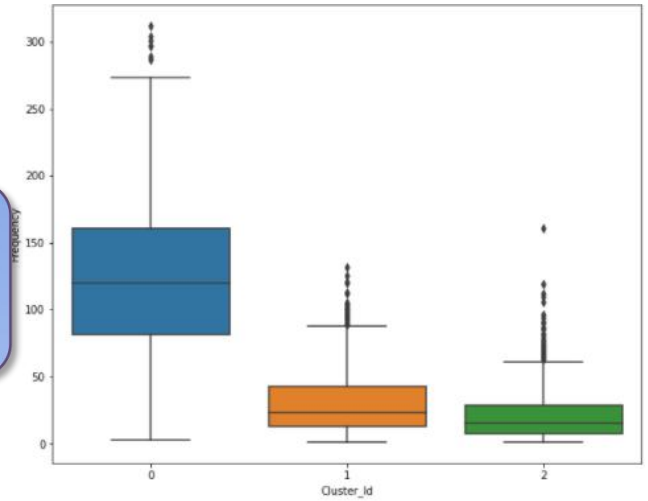
Building the Model



Box plot to visualize Cluster Id vs Amount



Box plot to visualize Cluster Id vs Recency



Box plot to visualize Cluster Id vs Frequency

clustering profile

K-Means Clustering with 3 Cluster Ids :

1. Customers with Cluster Id 0 are the customers with high amount of transactions as compared to other customers.
2. Customers with Cluster Id 1 are frequent buyers.
3. Customers with Cluster Id 2 are recent buyers.

India: +91-7022374614

US: 1-800-216-8930 (TOLL FREE)

support@intellipaat.com

24/ 7 Chat with Our Course Advisor