7-Jan-2024

Capstone Project: Decision Making in Data Science – IIT Roorkee

Comparative Study on various methods used for Customer Segmentation via estimating their Customer Lifetime Value

Nathan Aditya Ecka

DEFINITION

5 trillion U.S. dollars in a year,
15.6 billion U.S dollars in a day.

This was the estimated total sum of e-commerce purchases done in 2022, in the United States of America.

It is safe to say that in the last decade, the world has adopted and adapted to e-commerce aka online shopping due to its many benefits, if not convenience alone. And this trend is only going to rise. While USA is one of the biggest economies in the world with the largest userbase of e-commerce shoppers, there are other major players too, like UK and India which has shown steep affinity towards e-shopping over the past few years.

Over the past few years, what e-shopping was able to achieve which brick & mortar shops were not was the collection and retention of customer data. And great data brings great insights. Data has given insight to companies about their user purchase behavior, frequency, likeability, feedback loop has become faster and stronger.

This data has also helped companies to take out the guess work in Sales & Marketing. Now they are much more aware of their user base and are able to far better polish their marketing campaigns.

However, with the changing world, with changing needs, changing consumption patterns, and increasing costs, there is still much work needed for optimization in reducing costs and increasing profits. Consumption pattern changes across countries, states, localities etc. As there is no one size fit all, there is no single marketing technique to fit all.

Data Science, without a doubt, has been a boon in this area. The ability to see within the mega landscape of data and find intricate details, or to build prediction engines to predict customer behavior, the world is now getting warmed up to leverage the potential of using Data Science, even in marketing.

Customer Segmentation is one such area which hold massive potential in increasing revenue, reducing the guess work with the hit and trial approach in marketing. As they say, 80% of revenue comes from 20% of customers, it is very crucial to know who are these 20% of user. Moreover, who are the other 20% persuadable who can bring in more revenue to the organization.

This project is focused on understanding how Customer Segmentation can be achieved using a robust methodology called CLV i.e. Customer Lifetime Value.

PROJECT OVERVIEW

This project is focused on understanding, exploring and comparing various ways of generating Customer Lifetime Value (hereafter CLV).

Over the years, there have emerged various ways to estimate CLV, like –

1. *Excel based formulas*
2. *Manual computation in python or R*
3. *Python & R library-based computation*
4. *Supervised Learning like Predictive modeling using a target variable (mostly Total Sales)*
5. *Unsupervised Learning like Clustering*

In the world of Data Science, there are endless ways to learn, infer, predict, analyze anything. Also, there are many ways to generate accuracy, test score, prediction scores. And yes, seeing a 96% score looks good and satisfactory, but how do we know whether that a high score means high usability or higher sales or higher profit?

So, rather than just generating a one-off CLV using one of the methods, in this project we use using various techniques to generate CLV and then we will compare them to understand –

1. *methods which were productive and gave consistent results*
2. *methods which were complex*
3. *methods which were easier to compute yet provided appreciable result.*

Important things to note –

Dataset used – UkdataFinal.csv

This data was derived from the online retail master dataset from the UCI Machine Learning Repository. This is a transactional data set which contains all the actual transactions for a UK-based and registered ecommerce online retail store. The company mainly sells unique all-occasion gifts. This dataset has several features which includes the Invoice Number, Stock Code, Product Description, Product Quantity, Invoice Date, Unit Price, Customer ID, etc.

Techniques used for CLV generation –

1. *Linear Regression*
2. *Gradient Boosting Machines*
3. *K-Means Clustering*
4. *Simulation based on Poisson distribution*
5. *Lifetimes Fitters - ModifiedBetaGeo*
6. *Lifetimes plotting*
7. *Simple coding in Python*
8. *MinMaxScalar*

ABSTRACT

Before we move ahead, let's first understand what is CLV.

CLV is the present value of the future cash flow or the value of business decisions about sales, marketing, product development, and customer support.

CLV tell marketers, how much revenue they can expect from one customer over the course of the business relationship. The longer a customer continues to purchase from a company, the greater their lifetime value becomes.

By applying CLV, marketing managers can easily arrive at the dollar/rupee value associated with the long-term relationship with any customer.

It is an important metric as it costs less to keep existing customers than it does to acquire new ones, so increasing the value of your existing customers is a great way to drive growth.

Ideally, lifetime value should be greater than the cost of acquiring a customer i.e. ***CPA***

CLV > CPA

CPA is a bad method to evaluate customer value because we can easily get obsessed with costs and we strive to bring the cost down because costs are tangible.

CPA is the floor/bottom. VPA is the ceiling.

Great business, focuses more on bringing the profits up, rather than bringing the costs down. Apple could be a good example for this. Their products are not designed with a perspective of cost-cutting, rather than providing immense value which ultimately overshadows the higher acquisition of the product in the first place.

There are basically two types of business context mentioned below regards to the relationship and purchase opportunities.

a) Contractual - *Contractual business refers to the business where there is a definite time when the customer is going to churn or we can say we know when the customer is going to be dropped. This type of customer relationship known as contractual and the customers called the subscription customers. For Ex - Hotstar, Netflix, Amazon Prime Subscription*

b) Non-Contractual - *In the non-contractual world, customers do go away, but they do so silently; they have no need to tell us they are leaving. This makes for a much trickier CLV calculation. For Ex- Retail/E-Commerce*

In this project we will be focusing on the ***non-contractual transactional data****.*

Steps Involved in this Project: *Data Importing | Data Cleaning | Exploratory Data Analysis | Feature Engineering/Extraction | Cross Validation | Different Predictive Models Building | CLV Based Customer Segmentation | Model Evaluation | Outcome Comparison and Understanding*

Libraries, Packages & Fitters Used

Scikit Learn
Lifetimes
Plotly, Matplotlib, Seaborn
Numpy
Pandas
Datetime
Math
Warnings
Sklearn
MinMaxScaler
LinearRegression
GradientBoostingRegressor
train_test_split
r2_score
median_absolute_error
mean_squared_error
KMeans
ModifiedBetaGeoFitter
plot_probability_alive_matrix
calibration_and_holdout_data
summary_data_from_transaction_data
conditional_expected_number_of_purchases_up_to_time
conditional_probability_alive
customer_lifetime_value
scipy
poisson

METHODOLOGY & WORKFLOW

Methodology –

The analysis is done on RFM values
R – Recency
F- Frequency
M – Monetary Value

RFM stands for frequency, recency & monetary, is a marketing technique which is used to find the best customers by analyzing their past purchasing behavior. It includes how frequently they have purchased, what is the total amount that they have spent so far, what was the last time they have purchased from our online store so on & so forth.

R F M definition from lifetimes website
https://lifetimes.readthedocs.io/en/latest/Quickstart.html

- `frequency` represents the number of *repeats* purchases the customer has made. This means that it's one less than the total number of purchases. This is actually slightly wrong. It's the count of time periods the customer had a purchase in. So, if using days as units, then it's the count of days the customer had a purchase on.
- `T` represents the age of the customer in whatever time units chosen (weekly, in the above dataset). This is equal to the duration between a customer's first purchase and the end of the period under study.
- `recency` represents the age of the customer when they made their most recent purchases. This is equal to the duration between a customer's first purchase and their latest purchase. (Thus, if they have made only 1 purchase, the recency is 0.)
- `monetary_value` represents the average value of a given customer's purchases. This is equal to the sum of all a customer's purchases divided by the total number of purchases. Note that the denominator here is different than the `frequency` described above.

Workflow –

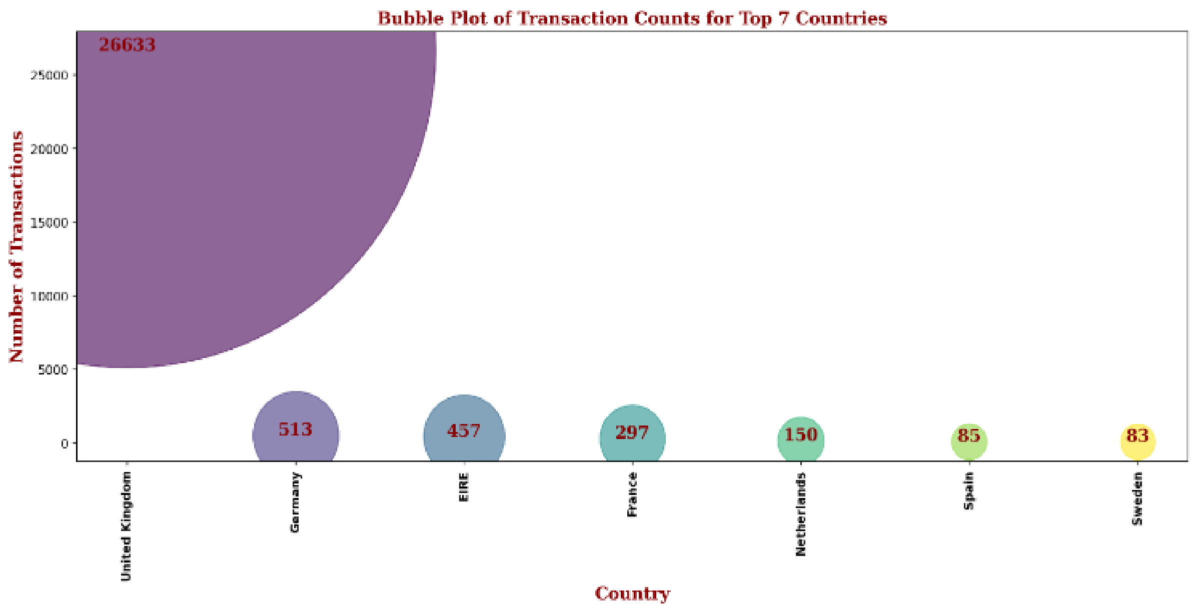
Below methods were used for our analysis.

- *Generating RFM from Quantiles & CLV generation using Python Code*

- *Generating RFM from lifetimes & CLV generation using Python Code*
- *CLV sales prediction using Linear Regression*
- *CLV sales prediction using Gradient Boosting Machine along with RFM data*
- *CLV generation using Lifetimes package: ModifiedBetaGeo fitter + GammaGamm fitter + K-Means*
- *Simulations on ModifiedBetaGeo data using Poisson package + K-Means on the simulation data*

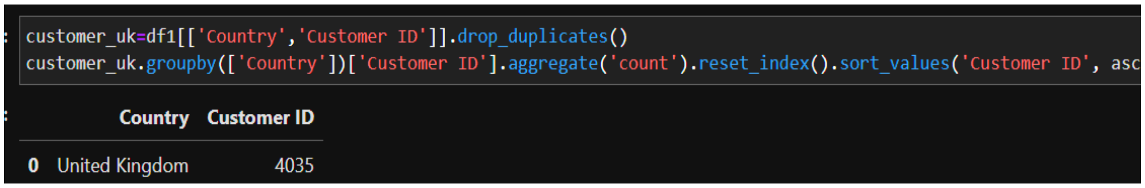
DATA EXPLORATION & PREPARATION

1. *Below are the top 7 countries with highest transaction.*



Since UK has the highest number of transactions, we will only work on UK data.

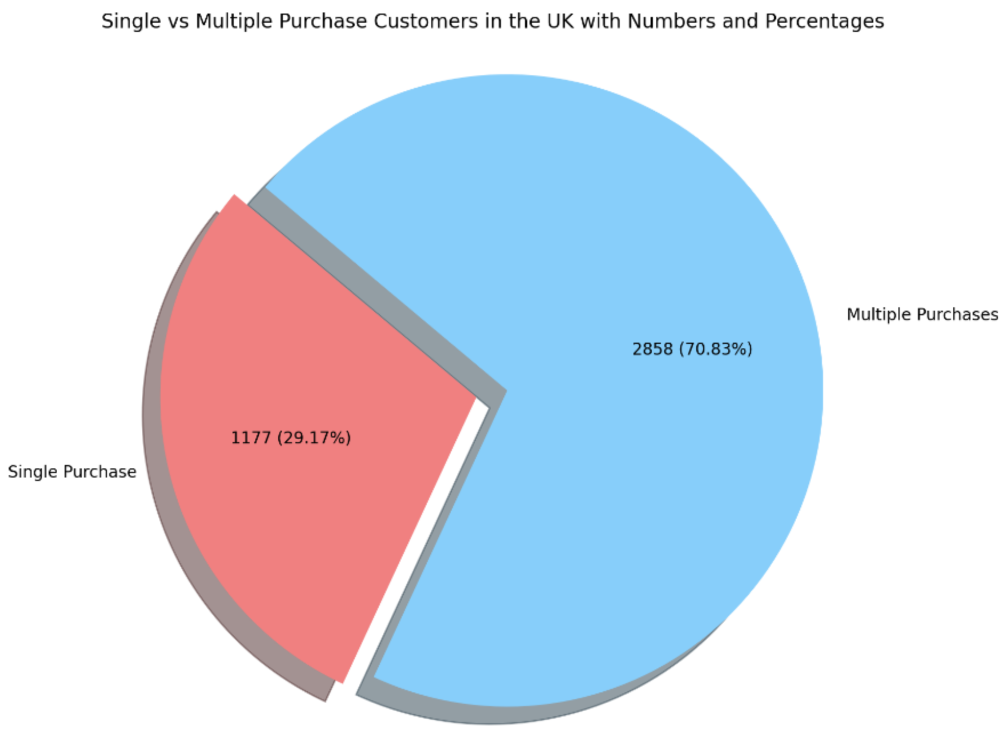
2. *Total count of dataset:* 525461
3. *Total count of UK data:* 485852
4. *Total number of unique customers who purchased in UK:* 4035
5. *Total customers after removing null customer ids, duplicate customer ids, Quantity only greater than 1(which means there was at least single purchase): 4035*



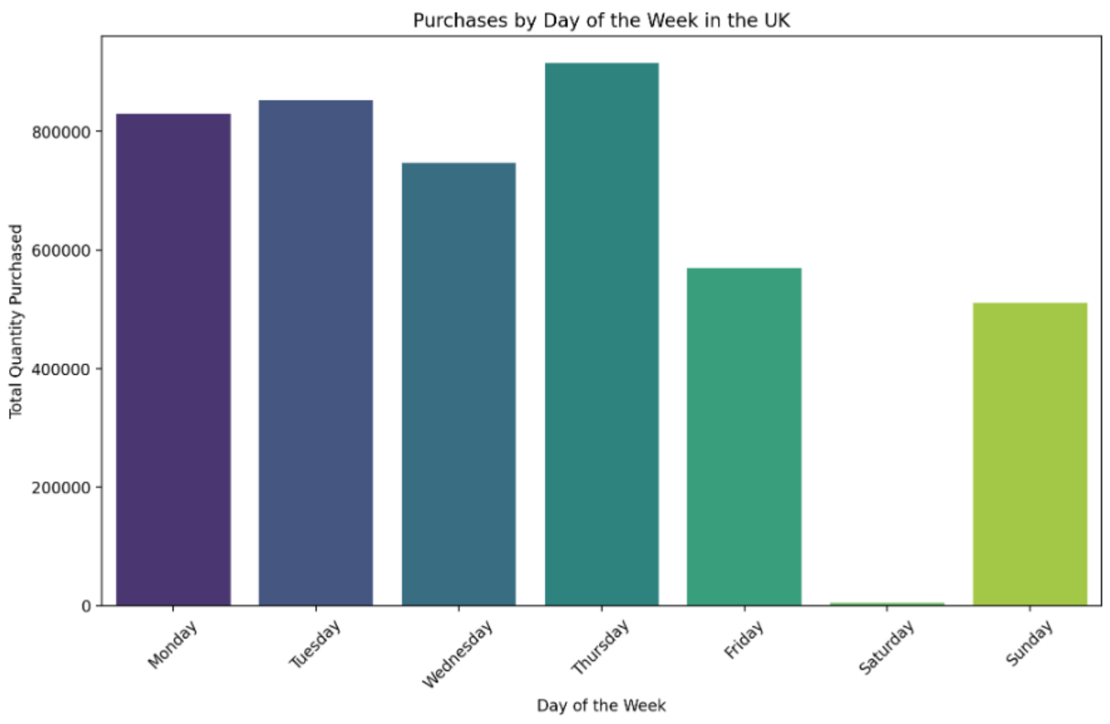
```
customer_uk=df1[['Country','Customer ID']].drop_duplicates()
customer_uk.groupby(['Country'])['Customer ID'].aggregate('count').reset_index().sort_values('Customer ID', asc
```

```
          Country   Customer ID
0  United Kingdom          4035
```

6. *Single vs Multiple purchases*



7. *Most Transactions happened in the month of November which is evident due to festive seasons.*
8. *2010 is the year in which we have the most transactions followed by the 2011*
9. *Q4 being the highest when it comes transactions.*
10. *It also observed that in the end of the 1st week and starting of the 3rd week, people tend to buy more.*
11. *People loves to shop on Thursday followed by Monday and Tuesday. Saturday gets the least transactions.*



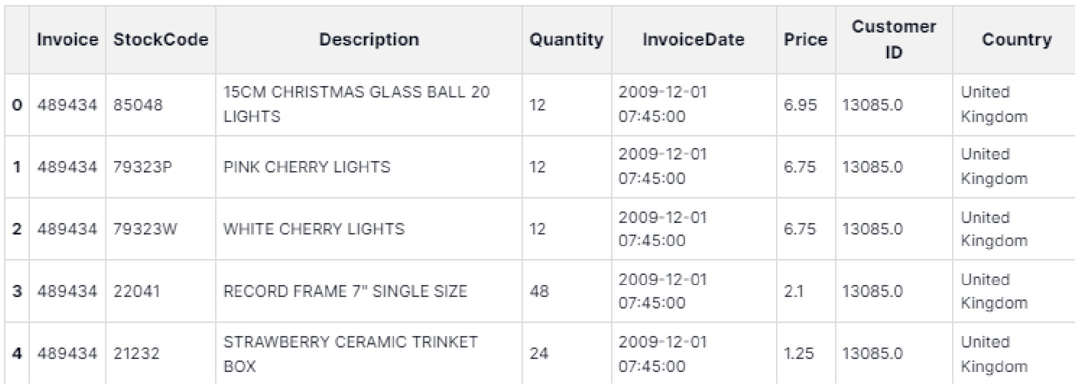
12. *Only Quantity above 0 are selected. Once done, we are left with 3971 unique customers.*

df1 = df1[(df1['Quantity']>0)]

13. *A new column for Total Sales is created –*

*df1['TotalPrice'] = df1['Quantity'] * df1['Price']*

Glimpse of our main Dataframe –



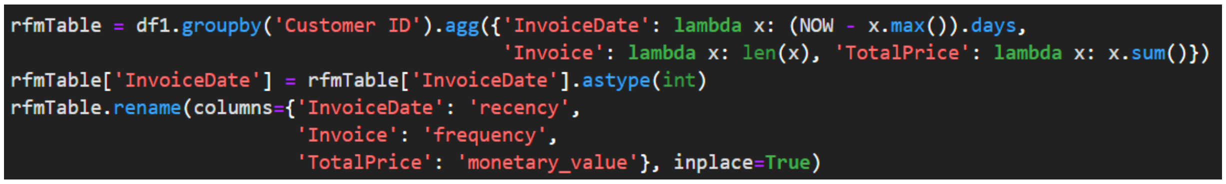
	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.1	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom

ANALYSIS

Though there are many great things which can be learned and explored in our dataset but since this project focuses more on the techniques used for RFM and CLV calculation and evaluation, we will rather straight dive into the techniques, analysis and evaluation.

Method - 1
Generating RFM from Quantiles & CLV generation using Python Code

Step 1: Calculate the RFM metrics for each customer.



```
rfmTable = df1.groupby('Customer ID').agg({'InvoiceDate': lambda x: (NOW - x.max()).days,
                                           'Invoice': lambda x: len(x), 'TotalPrice': lambda x: x.sum()})
rfmTable['InvoiceDate'] = rfmTable['InvoiceDate'].astype(int)
rfmTable.rename(columns={'InvoiceDate': 'recency',
                         'Invoice': 'frequency',
                         'TotalPrice': 'monetary_value'}, inplace=True)
```

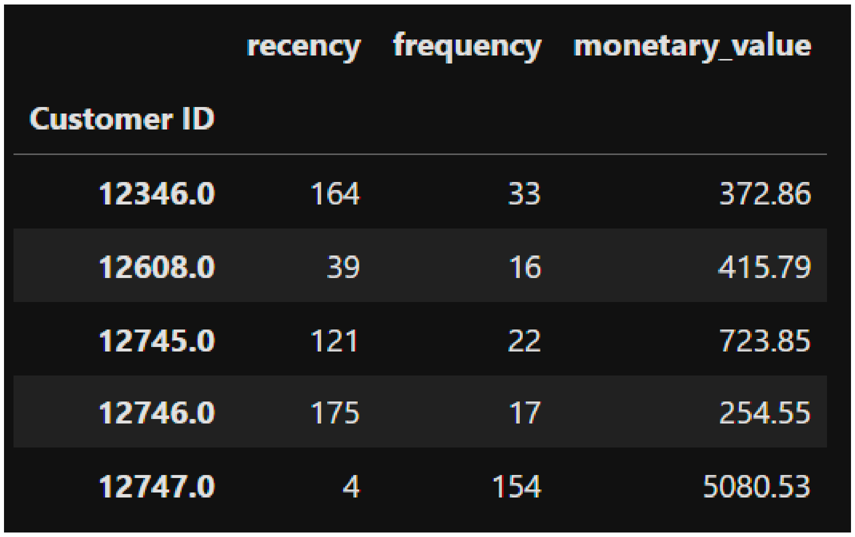
rfmTable = df1.groupby('Customer ID').agg({'InvoiceDate': lambda x: (NOW - x.max()).days, 'Invoice': lambda x: len(x), 'TotalPrice': lambda x: x.sum()})
rfmTable['InvoiceDate'] = rfmTable['InvoiceDate'].astype(int)
rfmTable.rename(columns={'InvoiceDate': 'recency',
'Invoice': 'frequency',
'TotalPrice': 'monetary_value'}, inplace=True)

F : frequency : calculated by counting unique orders(not the number of items inside a transaction)

R : Recency : calculated using subtracting latest order date with NOW = dt.datetime(2010,12,10)

M : Monetary_Value : total price of all orders per customer

Output –

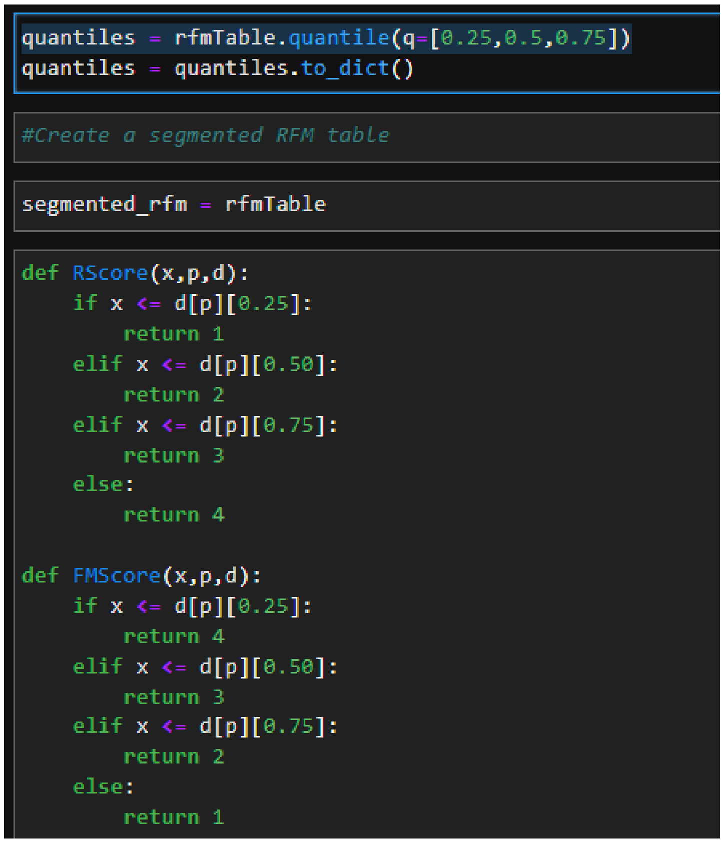


	recency	frequency	monetary_value
Customer ID			
12346.0	164	33	372.86
12608.0	39	16	415.79
12745.0	121	22	723.85
12746.0	175	17	254.55
12747.0	4	154	5080.53

Step 2: Add segment numbers to RFM table.

we use the Quantiles to generate a score between 1 to 4 or each of the R, F, M values.

Code used –



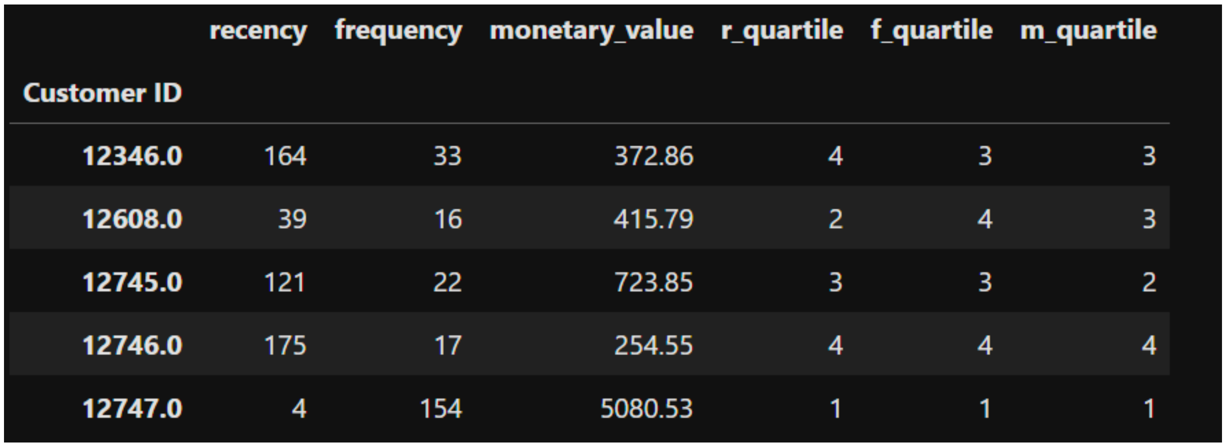
```
quantiles = rfmTable.quantile(q=[0.25,0.5,0.75])
quantiles = quantiles.to_dict()
```

```
#Create a segmented RFM table
```

```
segmented_rfm = rfmTable
```

```
def RScore(x,p,d):
    if x <= d[p][0.25]:
        return 1
    elif x <= d[p][0.50]:
        return 2
    elif x <= d[p][0.75]:
        return 3
    else:
        return 4

def FMScore(x,p,d):
    if x <= d[p][0.25]:
        return 4
    elif x <= d[p][0.50]:
        return 3
    elif x <= d[p][0.75]:
        return 2
    else:
        return 1
```

Output -



Customer ID	recency	frequency	monetary_value	r_quartile	f_quartile	m_quartile
12346.0	164	33	372.86	4	3	3
12608.0	39	16	415.79	2	4	3
12745.0	121	22	723.85	3	3	2
12746.0	175	17	254.55	4	4	4
12747.0	4	154	5080.53	1	1	1

Step 3: Sort according to the RFM scores from the best customers.

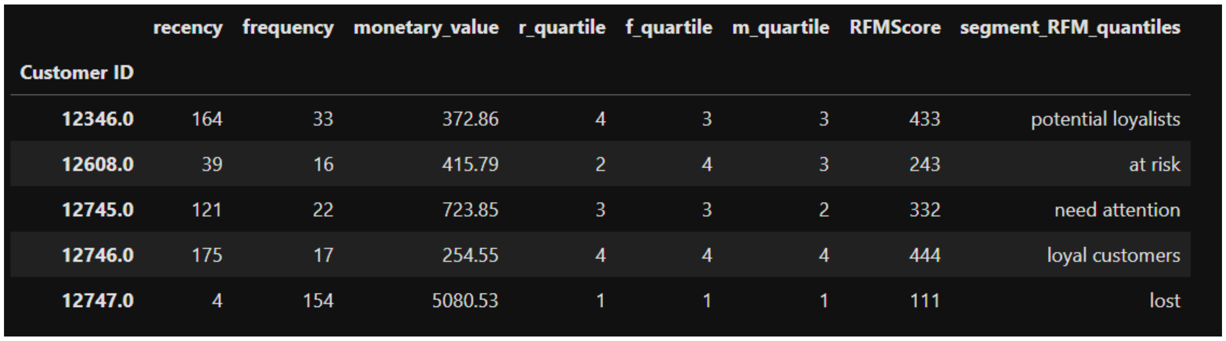
score 111 = worst
score 444 = best

Step 4: Create customer segment map

Code & Map used –

```
r'22': 'hibernating',
r'[1-2][1-2]': 'lost',
r'15': 'can\'t lose',
r'[1-2][3-5]': 'at risk',
r'3[1-2]': 'about to sleep',
r'33': 'need attention',
r'55': 'champions',
r'[3-5][4-5]': 'loyal customers',
r'41': 'promising',
r'51': 'new customers',
r'[4-5][2-3]': 'potential loyalists'
```

Output after mapping –



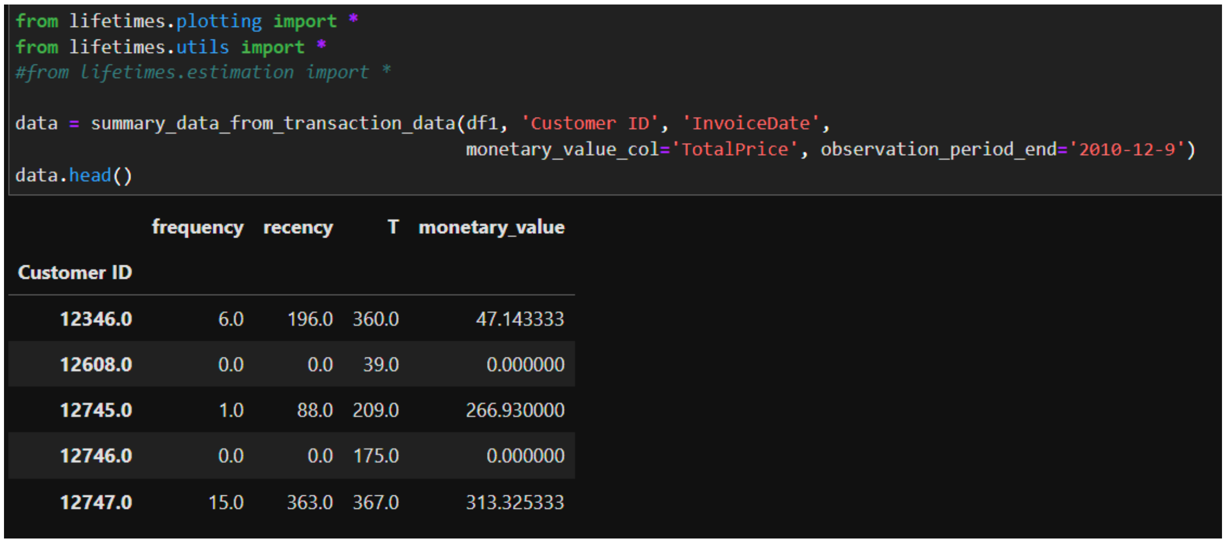
Customer ID	recency	frequency	monetary_value	r_quartile	f_quartile	m_quartile	RFMScore	segment_RFM_quantiles
12346.0	164	33	372.86	4	3	3	433	potential loyalists
12608.0	39	16	415.79	2	4	3	243	at risk
12745.0	121	22	723.85	3	3	2	332	need attention
12746.0	175	17	254.55	4	4	4	444	loyal customers
12747.0	4	154	5080.53	1	1	1	111	lost

Method - 2
Generating RFM from lifetimes package & CLV generation using Python Code

This method is similar to the previous one with the exception that in this method we use Lifetimes package to generate the RFM matrix.

Step 1: Generate RFM matrix using lifetimes package

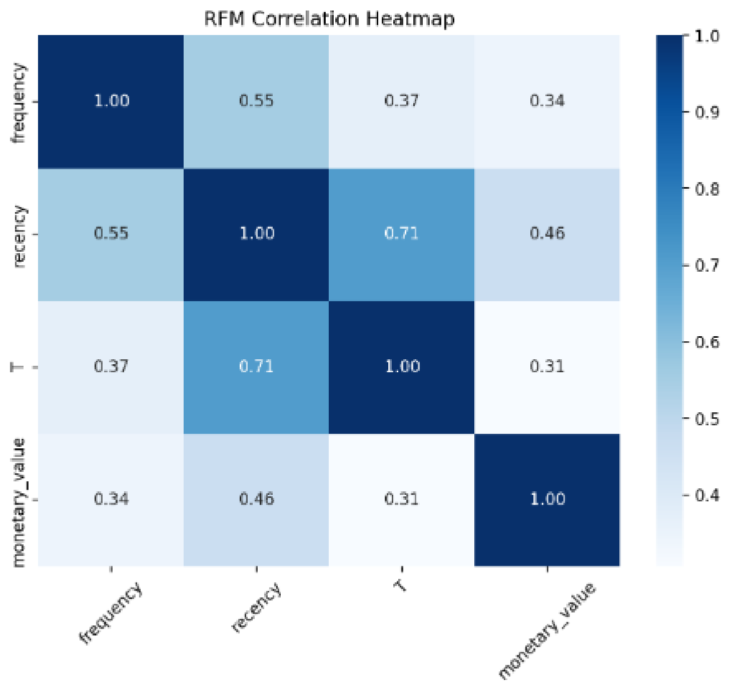
Package lifetimes.utils.summary_data_from_transaction_data() *takes transaction data as input and provides R,F,M,T as output.*



```
from lifetimes.plotting import *
from lifetimes.utils import *
#from lifetimes.estimation import *

data = summary_data_from_transaction_data(df1, 'Customer ID', 'InvoiceDate',
                                          monetary_value_col='TotalPrice', observation_period_end='2010-12-9')
data.head()
```

```
             frequency  recency      T  monetary_value
Customer ID
    12346.0        6.0    196.0  360.0       47.143333
    12608.0        0.0      0.0   39.0        0.000000
    12745.0        1.0     88.0  209.0      266.930000
    12746.0        0.0      0.0  175.0        0.000000
    12747.0       15.0    363.0  367.0      313.325333
```

Heatmap Correlation is intuitive.

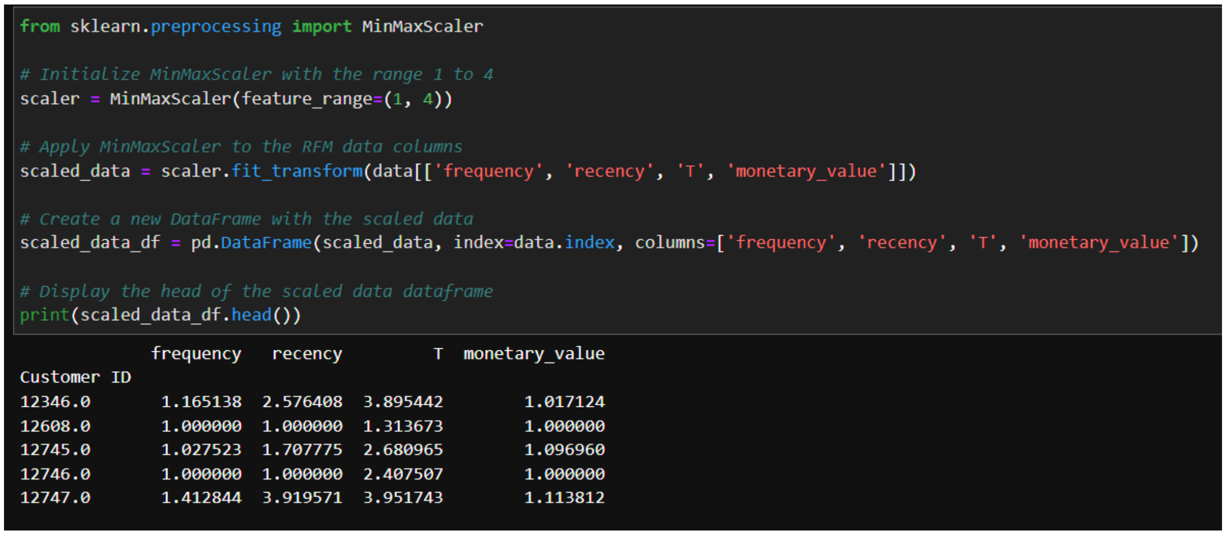


Step 2: Using MinMaxScalar with range 1 to 4 on the RFM matrix

The data in previous Quantiles strategy was in whole numerals between 1 to 4. Whereas, if you see the screenshot below, the RFMT values generated using Lifetimes package are –

a. Not in the range between 1 to 4
b. Not in whole numbers i.e. 1, 2, 3,4

*We used **minmaxscalar** with range 1 to 4 to transform the RFMT values between the range 1 to 4. This will enable us to compare it directly with RFM quantiles values.*



```
from sklearn.preprocessing import MinMaxScaler

# Initialize MinMaxScaler with the range 1 to 4
scaler = MinMaxScaler(feature_range=(1, 4))

# Apply MinMaxScaler to the RFM data columns
scaled_data = scaler.fit_transform(data[['frequency', 'recency', 'T', 'monetary_value']])

# Create a new DataFrame with the scaled data
scaled_data_df = pd.DataFrame(scaled_data, index=data.index, columns=['frequency', 'recency', 'T', 'monetary_value'])

# Display the head of the scaled data dataframe
print(scaled_data_df.head())
```

```
             frequency   recency         T  monetary_value
Customer ID
12346.0       1.165138  2.576408  3.895442        1.017124
12608.0       1.000000  1.000000  1.313673        1.000000
12745.0       1.027523  1.707775  2.680965        1.096960
12746.0       1.000000  1.000000  2.407507        1.000000
12747.0       1.412844  3.919571  3.951743        1.113812
```

Step 3: Use custom map to transform values between 1 to 4

Output from MinMaxScalar is in range between 1 to 4 but they are still decimal values. This is still not usable for direct comparison.

We first used round() function in numpy but we realized that a lot of values were lost in conversion. For example, it is debatable whether 1.4 should lie in 1 or 2, or 1.6 should lie in 1 or 2.

We used the below custom map to convert decimal values to whole numbers -

1.0 to 1.75 = 1
1.76 to 2.67 = 2
2.68 to 3.4 = 3
3.5 to 4 = 4

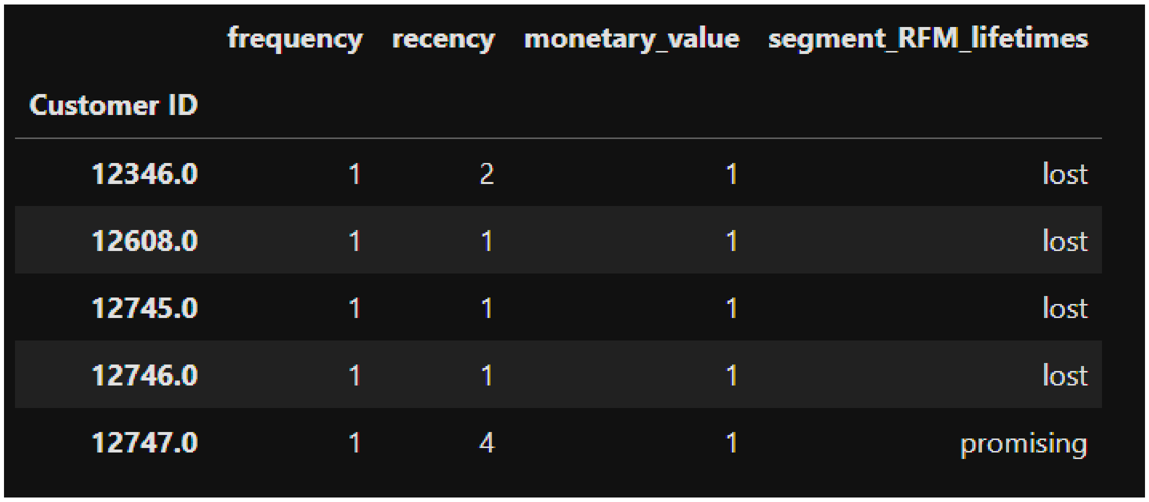
Step 4: Create customer segment map

Code & Map used –

r'22': 'hibernating',
r'[1-2][1-2]': 'lost',
r'15': 'can\'t lose',
r'[1-2][3-5]': 'at risk',

```
    r'3[1-2]': 'about to sleep',
    r'33': 'need attention',
    r'55': 'champions',
    r'[3-5][4-5]': 'loyal customers',
    r'41': 'promising',
    r'51': 'new customers',
    r'[4-5][2-3]': 'potential loyalists'
```

Output after mapping –



	frequency	recency	monetary_value	segment_RFM_lifetimes
Customer ID				
12346.0	1	2	1	lost
12608.0	1	1	1	lost
12745.0	1	1	1	lost
12746.0	1	1	1	lost
12747.0	1	4	1	promising

Method - 3
CLV sales prediction using Linear Regression

Important things to note –

14. *This is Supervised Learning; hence a target variable is required.*
15. *Sales average from the past data is used as target variable to predict the value of a customer.*
16. *Recency & Frequency is calculated from data & not passing it through lifetimes package.*
17. *Custom python queries are used to create summary data which will be used for regression.*
18. *R2 score is used for evaluation.*

Step 1: Create custom summary data using python code –

this code (mentioned below) clubs all the transaction per user and then generates min, max, total, average sales, total sales count.

Also, minimum purchase date & maximum purchase date is extracted. Recency is the difference between min & max.

*The definition of **Recency** is flexible and can change based on business and problem statement, but for our analysis we are taking Recency as the difference between a customer's **First** purchase & **Latest** purchase.*

Recency: Customer's First purchase – Latest purchase

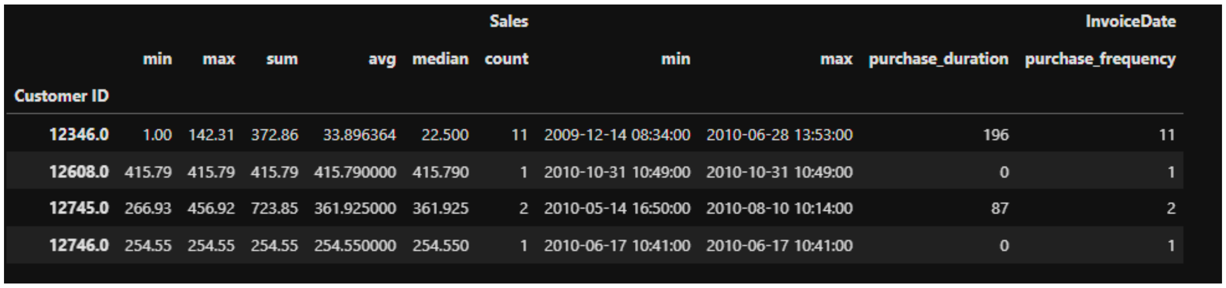
Frequency is most always the number of times a customer has done purchases on the platform.

Frequency: total number of purchases from a customer

Other definitions or methods of calculation frequency can also be –

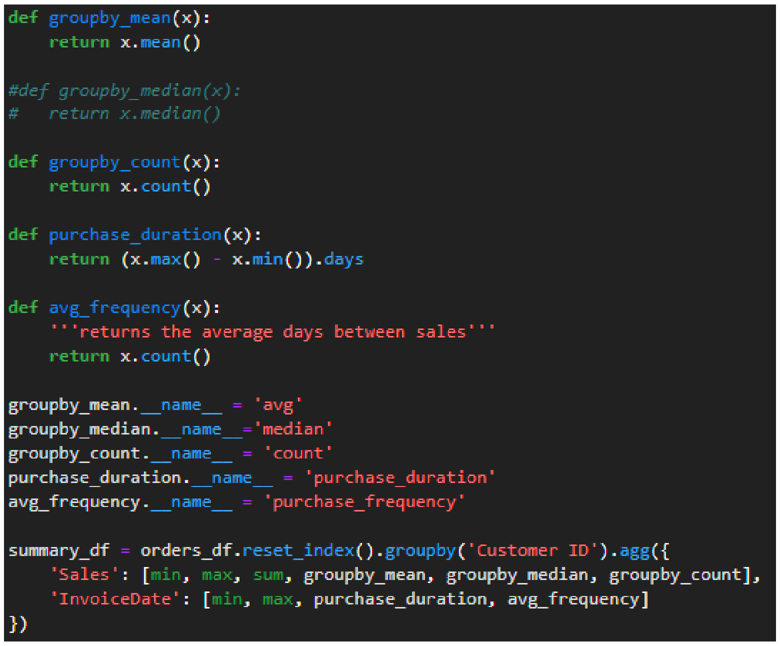
a. *the number of purchases within the observed time period*
b. *count of time periods when the customer repurchases*

For simplicity, we will be using 'total number of purchases from a customer' as the value of frequency in this entire project.



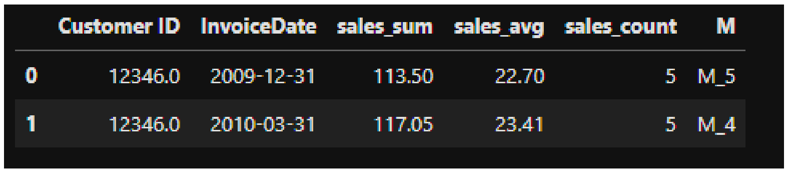
	Sales						InvoiceDate			
	min	max	sum	avg	median	count	min	max	purchase_duration	purchase_frequency
Customer ID										
12346.0	1.00	142.31	372.86	33.896364	22.500	11	2009-12-14 08:34:00	2010-06-28 13:53:00	196	11
12608.0	415.79	415.79	415.79	415.790000	415.790	1	2010-10-31 10:49:00	2010-10-31 10:49:00	0	1
12745.0	266.93	456.92	723.85	361.925000	361.925	2	2010-05-14 16:50:00	2010-08-10 10:14:00	87	2
12746.0	254.55	254.55	254.55	254.550000	254.550	1	2010-06-17 10:41:00	2010-06-17 10:41:00	0	1

Code used –



```python
def groupby_mean(x):
    return x.mean()

#def groupby_median(x):
#   return x.median()

def groupby_count(x):
    return x.count()

def purchase_duration(x):
    return (x.max() - x.min()).days

def avg_frequency(x):
    '''returns the average days between sales'''
    return x.count()

groupby_mean.__name__ = 'avg'
groupby_median.__name__='median'
groupby_count.__name__ = 'count'
purchase_duration.__name__ = 'purchase_duration'
avg_frequency.__name__ = 'purchase_frequency'

summary_df = orders_df.reset_index().groupby('Customer ID').agg({
    'Sales': [min, max, sum, groupby_mean, groupby_median, groupby_count],
    'InvoiceDate': [min, max, purchase_duration, avg_frequency]
})
```

Step 2: Generate data for user shopping done on 3 months intervals -

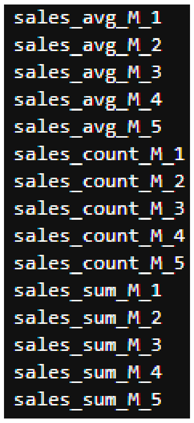


	Customer ID	InvoiceDate	sales_sum	sales_avg	sales_count	M
0	12346.0	2009-12-31	113.50	22.70	5	M_5
1	12346.0	2010-03-31	117.05	23.41	5	M_4

Above data is for customer id 12346, showing that they have done 5 purchases in Month interval 5 and 4 purchases in month interval 4.

Step 3: *Create custom summary data for 3-months intervals –*

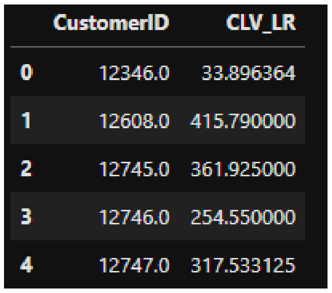
Data is generated in the below column, using pivot tables.



```
sales_avg_M_1
sales_avg_M_2
sales_avg_M_3
sales_avg_M_4
sales_avg_M_5
sales_count_M_1
sales_count_M_2
sales_count_M_3
sales_count_M_4
sales_count_M_5
sales_sum_M_1
sales_sum_M_2
sales_sum_M_3
sales_sum_M_4
sales_sum_M_5
```

Step 4: *Generating Feature & Target variable*

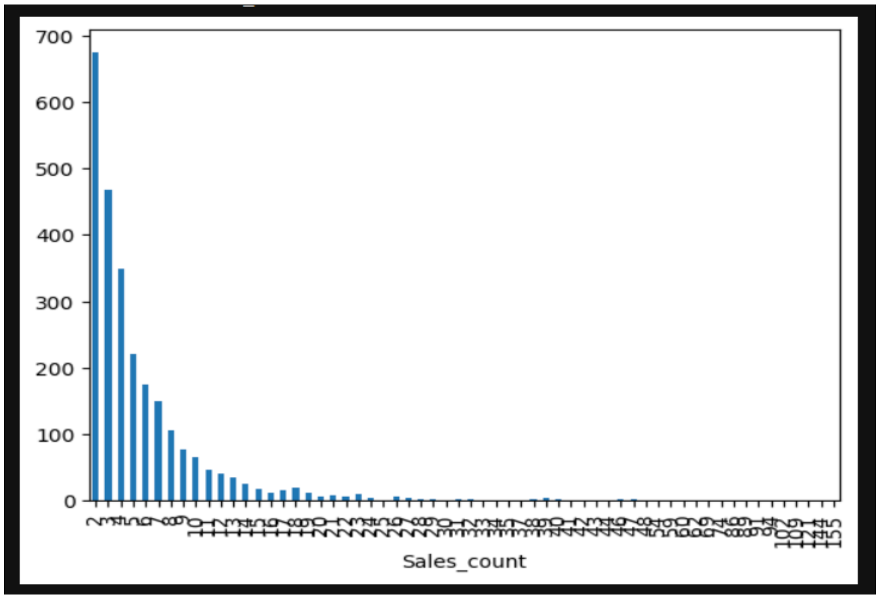
As mentioned earlier, the supervised learning technique requires a target variable, and so we are going to use sales average of each user as the target variable. Target column is names CLV_LR.



	CustomerID	CLV_LR
0	12346.0	33.896364
1	12608.0	415.790000
2	12745.0	361.925000
3	12746.0	254.550000
4	12747.0	317.533125

Rest all of the columns in step 3 are used as feature variables.
Note: Recency & Frequency is not used as features.

The drawback is that there are 30% customers with only single purchases. So, their sales average is the only purchase they did. Also, their recency and frequency is not the real determent for their purchasing behavior.



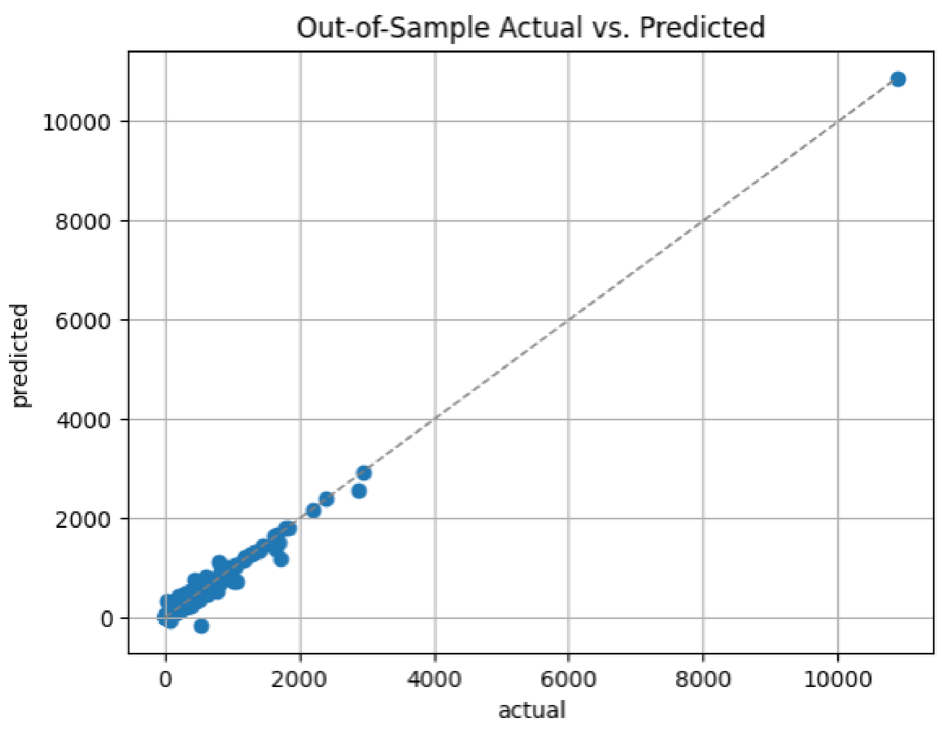
Step 5: Linear Regression

Sklearn.linear_model LinearRegression is used with train_test split of 70:30 and random_state=11

Below is the result –

R2_score on training data: 97.24%
R2_score on test data: 98.511%

Scatter plot using sales_avg as target -



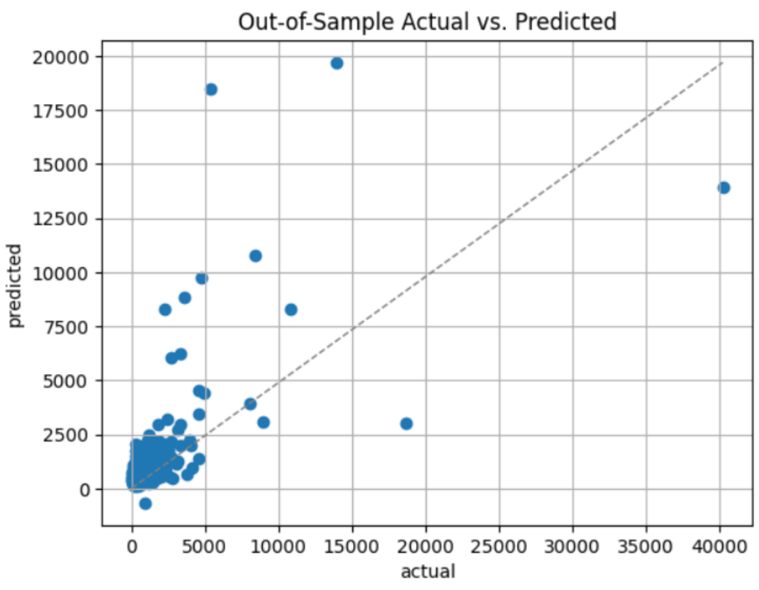
Actual values and predicted values have fitted beautifully across the 45% best fit line. Which indicates that our model has worked very well.

Before choosing salves-average as target variable, we also built the model with sales-sum as target variable. But got the r2-score for

Training data: 82.98%
Test Data: 43.83%

Higher training score over test score indicates overfitted model.

Scatter plot using sales_sum as target –



The plot above depicts that the model has not fitted the data as good as compared to *taking sales_avg as target.*

We also worked with sales_median as target variable but the r2 score was not very attractive. The reason for using median over average is that in non-contractual transactions, some transaction can be of very high value and some of very low value. This can affect the mean of sales average drastically when the total number of transactions of a user is very low to begin with. Median could have provided a middle point of total sales value rather than mean.

NOTE: The complete dataset was again passed through the model and the result were store in a new column 'predicted_CLV_LR' which will be used later for analysis.

Method - 4
CLV sales prediction using Gradient Boosting Machine along with RFM data

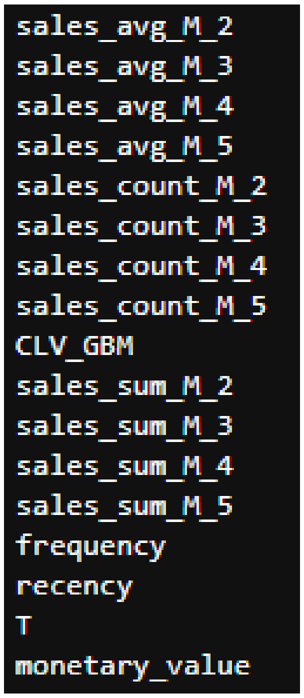
Step 1: Choosing feature and target variables

Feature variables:
same as used in Linear Regression except the M1 i.e. the first 3 month of average, total and count of sales data.

We also added Recency, Frequency, T, Monetary value data for each customer using lifetimes package summary_data_from_transaction_data.

Target variable:
we choose sales sum of M1 per user. Target column is named CLV_GBM.

Dataframe columns –



```
sales_avg_M_2
sales_avg_M_3
sales_avg_M_4
sales_avg_M_5
sales_count_M_2
sales_count_M_3
sales_count_M_4
sales_count_M_5
CLV_GBM
sales_sum_M_2
sales_sum_M_3
sales_sum_M_4
sales_sum_M_5
frequency
recency
T
monetary_value
```

*Instead of running the whole regression exercise with just another model i.e. **Gradient Boosting Models,** we are adding 2 more features i.e. recency & frequency. Reason for adding 2 more features and not keeping the feature set same as previous study is that we are more interested in learning how we can achieve results using different techniques rather than just focusing on comparing 2 different models.*

*We choose **Gradient Boosting Models** since it is often more flexible and can capture more complex nonlinear relationships between the features and the output variable.*

Step 2: running the GBM regressor

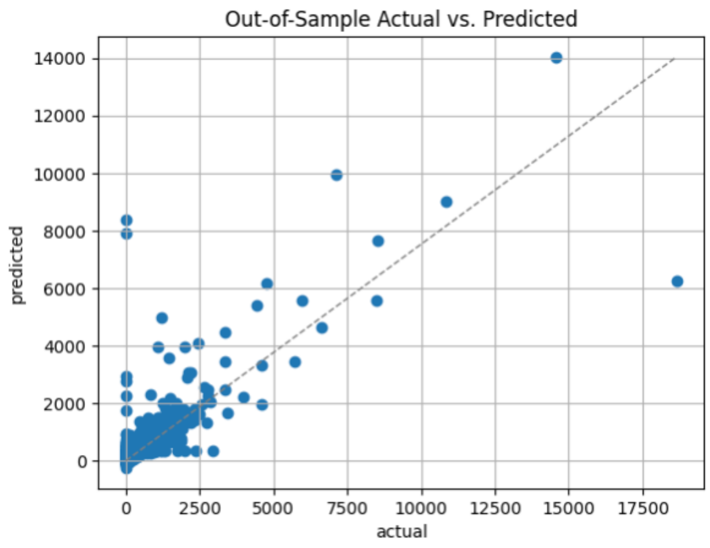
sklearn.ensembl GradientBoostingRegressoris used with train_test split of 70:30 and random_state=11

Below is the result –

R2_score on training data: 97.78%
R2_score on test data: 63.82%

A higher training score over test score depicts overfitting.

Scatter plot –



Even though we would have loved to see a higher R2 score on test data, 63% is decent number to begin with. Scatter plot shows fairly good actual vs predictions along the best fit line.

NOTE: The complete dataset was again passed through the model and the result were store in a new column 'predicted_CLV_GBM' which will be used later for analysis.

Method - 5
CLV generation using Lifetimes package: ModifiedBetaGeo fitter + GammaGamm fitter + K-Means

The following analysis is conducted in Python using Lifetimes package developed by Cameron Davidson-Pilon, data scientist at Shopify, and the code was largely borrowed from Lifetimes documentation.

This analysis is drawn upon Dr. Peter Fader of Wharton. Going into the math of CLV researched and present by Dr. Peter Fader is beyond the scope of this project. However, I will link the Paper in the end of this project.

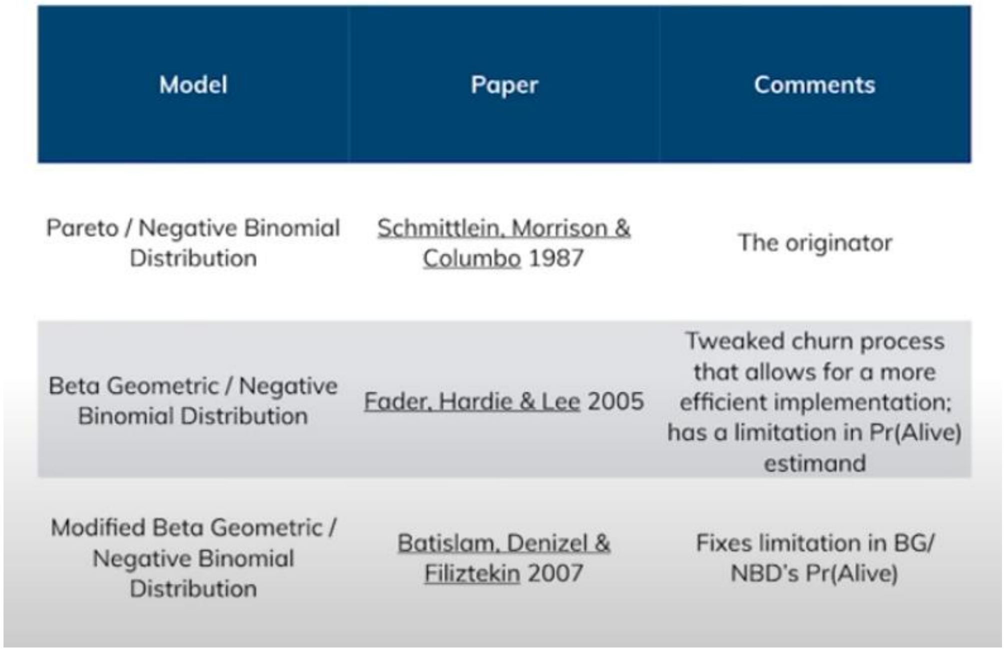
But to give a brief, there are 4 model/fitters which are widely used –

1. *Pareto Negative-Binomial Distribution (NBD hereafter)*
2. *Beta Geometric Distribution*

3. *Modified Beta Geometric/ NBD*
4. *Beta Geometric Beta Binomial Distribution*

Pareto-NDB was the pioneering model first published in 1987 by Dr. Schmittlein. Later in 2005, Dr. Peter Fader of Wharton came up with Beta Geometric/ NBD model which improves upon Pareto-NDB, though pareto still is a very successful and useful model to begin with which can give excellent results.

Below is a short summary of timelines –



Model	Paper	Comments
Pareto / Negative Binomial Distribution	Schmittlein, Morrison & Columbo 1987	The originator
Beta Geometric / Negative Binomial Distribution	Fader, Hardie & Lee 2005	Tweaked churn process that allows for a more efficient implementation; has a limitation in Pr(Alive) estimand
Modified Beta Geometric / Negative Binomial Distribution	Batislam, Denizel & Filiztekin 2007	Fixes limitation in BG/ NBD's Pr(Alive)

The ***BG/NBD is an "easy" alternative to the Pareto/NBD*** *model that is easier to implement. The BG/NBD model slight adapts the behavioral "story" associated with the Pareto/NBD model in order to simplify the implementation. The BG/NBD model uses a beta-geometric and exponential gamma mixture distributions to model customer behavior.* ***The key difference to the Pareto/NBD*** *model is that a customer can only churn right after a transaction. This simplifies computations significantly, however has the drawback that a customer cannot churn until he/she makes a transaction. The Pareto/NBD model assumes that a customer can churn at any time.*

This can also be used to understand customer churn.

Beta Geo Fitter is used to estimate the model parameters using Maximum Likelihood.
Pareto/NBD model is generally concerned with repeat transactions—that is, the first transaction is ignored. This is convenient for firms using the model in practice, since it is easier to keep track of all customers who have made at least one transaction (as opposed to trying to account for those who have not made any transactions at all)..

***BG** model is a simple way to model customer retention, based on two assumptions:*

- *Customers have a fixed probability of making a purchase in any given time period, which follows a **geometric distribution**.*
- *Customers have a fixed probability of becoming inactive (or "dying") after any purchase, which follows a **beta distribution**.*

The BG model can estimate the expected number of purchases a customer will make in a given time period, as well as the probability that they are still active (or "alive"). However, the BG model has some limitations:

- *It assumes that customers are either active or inactive, and does not account for varying levels of engagement or loyalty.*
- *It assumes that the purchase and dropout probabilities are constant over time, and do not depend on the recency or frequency of previous purchases.*
- *It does not incorporate any information about the monetary value of purchases, which can vary significantly across customers and time periods.*

*The **MBG/NBD** model is an extension of the BG model that addresses some of these limitations. The MBG/NBD model makes the following modifications:*

- *Instead of assuming a geometric distribution for the purchase probability, it assumes a **negative binomial distribution**, which allows for more flexibility and heterogeneity in customer behaviour.*
- *Instead of assuming a constant dropout probability, it assumes that the dropout probability depends on the number of previous purchases, which captures the idea that customers become more loyal as they purchase more frequently.*
- *It also allows for the possibility that some customers never make a purchase,*

which is a realistic scenario in many businesses.

The MBG/NBD model can also estimate the expected number of purchases and the probability of being alive, but with more accuracy and realism than the BG model. Additionally, the MBG/NBD model can be combined with other models to estimate the monetary value of purchases, such as the gamma-gamma model or the Pareto/NBD model. This way, the MBG/NBD model can provide a more comprehensive and reliable estimate of CLV.

To summarize, the MBG/NBD model is better than the BG model in predicting CLV because it can account for more variation and dynamics in customer behaviour, and it can incorporate information about the monetary value of purchases.

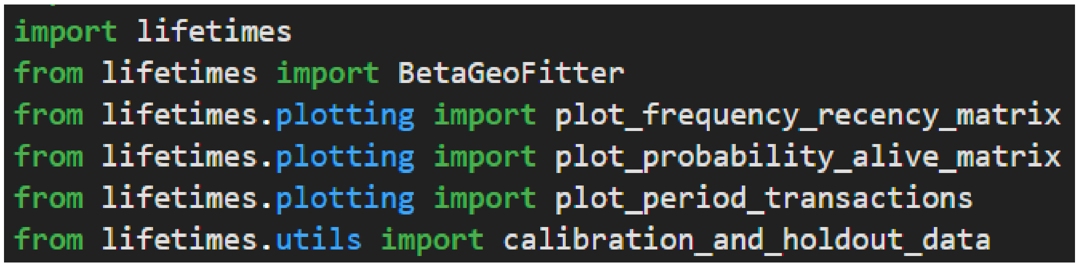
To select which model to use really depends on the use case and dataset. In our case, we are going to use MBG/NBD, not only because it is a powerful model, we also have customer with single purchases.

Once MBG/NBD is applied to our dataset, we will pass our data through Gamma-Gamm fitter to take into account the economic value of each transaction.

Gamma-Gamma fitter can predict likely spend per transaction in the future for each customer.

Once Done, we will use K-Means to create cluster the type of customer we have into our dataset and to map them accordingly.

Following packages and utilities were used –



```
import lifetimes
from lifetimes import BetaGeoFitter
from lifetimes.plotting import plot_frequency_recency_matrix
from lifetimes.plotting import plot_probability_alive_matrix
from lifetimes.plotting import plot_period_transactions
from lifetimes.utils import calibration_and_holdout_data
```

Following utilities from Lifetimes packages were used –

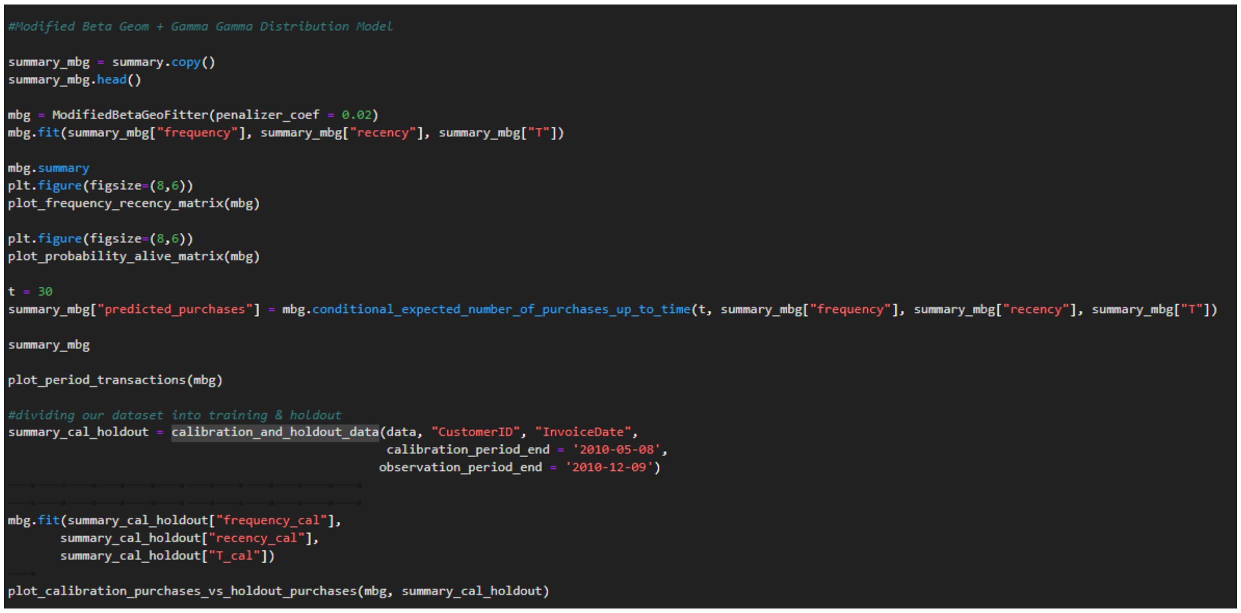
lifetimes.utils.summary_data_from_transaction_data
– to generate RFM matrix.

conditional_expected_number_of_purchases_up_to_time
– to generate future purchases by each customer.

calibration_and_holdout_data –
for train and test
conditional_probability_alive –
to understand whether the customer is 'alive' i.e. active in the platform. This can be used for customer churn.

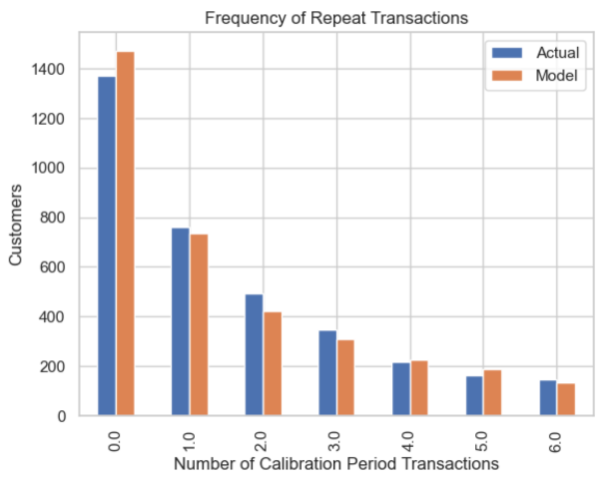
conditional_expected_average_profit –
to generate expected future sales by a customer in the given time period.

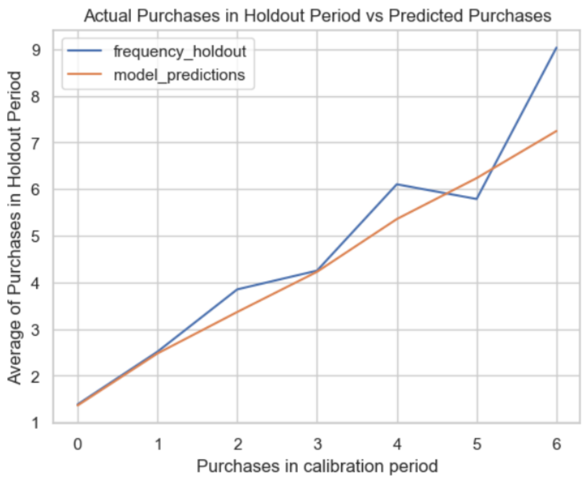
customer_lifetime_value –
to generate CLV of the customer.



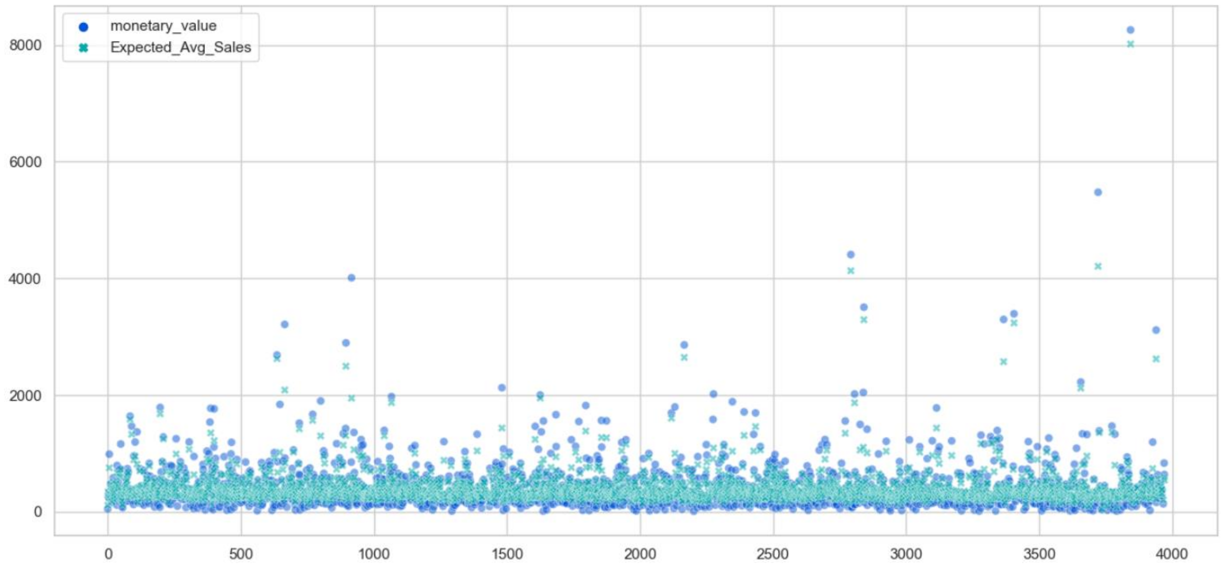
```
#Modified Beta Geom + Gamma Gamma Distribution Model

summary_mbg = summary.copy()
summary_mbg.head()

mbg = ModifiedBetaGeoFitter(penalizer_coef = 0.02)
mbg.fit(summary_mbg["frequency"], summary_mbg["recency"], summary_mbg["T"])

mbg.summary
plt.figure(figsize=(8,6))
plot_frequency_recency_matrix(mbg)

plt.figure(figsize=(8,6))
plot_probability_alive_matrix(mbg)

t = 30
summary_mbg["predicted_purchases"] = mbg.conditional_expected_number_of_purchases_up_to_time(t, summary_mbg["frequency"], summary_mbg["recency"], summary_mbg["T"])

summary_mbg

plot_period_transactions(mbg)

#dividing our dataset into training & holdout
summary_cal_holdout = calibration_and_holdout_data(data, "CustomerID", "InvoiceDate",
                                                   calibration_period_end = '2010-05-08',
                                                  observation_period_end = '2010-12-09')


mbg.fit(summary_cal_holdout["frequency_cal"],
       summary_cal_holdout["recency_cal"],
       summary_cal_holdout["T_cal"])

plot_calibration_purchases_vs_holdout_purchases(mbg, summary_cal_holdout)
```

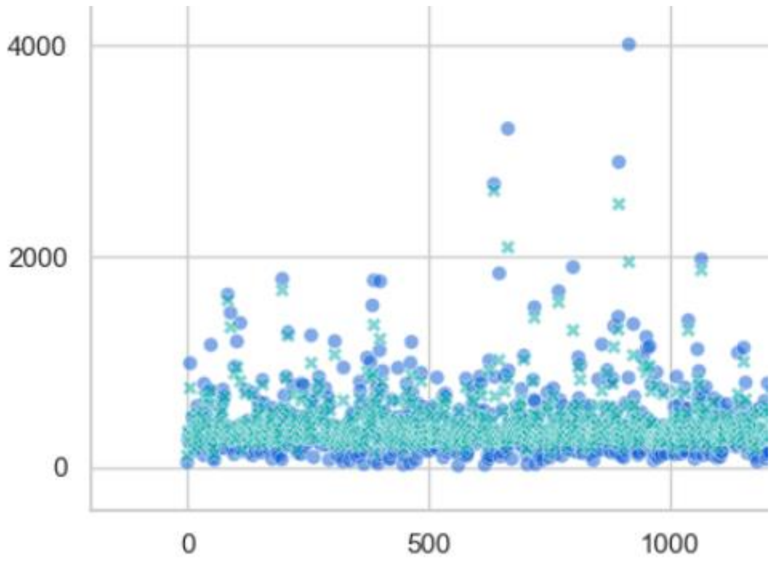
Actual vs Model prediction is satisfactory –



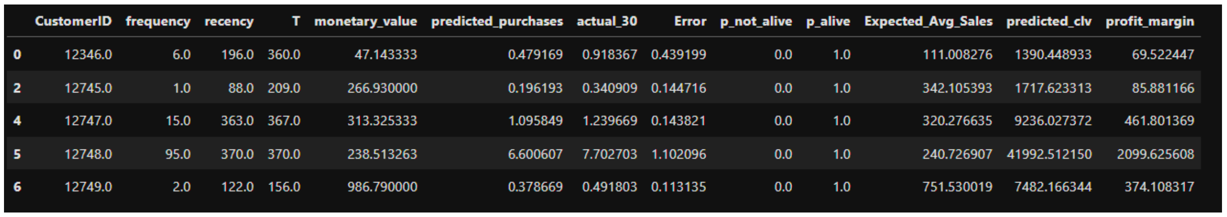


Expected Average sales vs Monetary Value is converging nicely –





This is the dataframe output after passing the data through lifetimes fitters mentioned above –



	CustomerID	frequency	recency	T	monetary_value	predicted_purchases	actual_30	Error	p_not_alive	p_alive	Expected_Avg_Sales	predicted_clv	profit_margin
0	12346.0	6.0	196.0	360.0	47.143333	0.479169	0.918367	0.439199	0.0	1.0	111.008276	1390.448933	69.522447
2	12745.0	1.0	88.0	209.0	266.930000	0.196193	0.340909	0.144716	0.0	1.0	342.105393	1717.623313	85.881166
4	12747.0	15.0	363.0	367.0	313.325333	1.095849	1.239669	0.143821	0.0	1.0	320.276635	9236.027372	461.801369
5	12748.0	95.0	370.0	370.0	238.513263	6.600607	7.702703	1.102096	0.0	1.0	240.726907	41992.512150	2099.625608
6	12749.0	2.0	122.0	156.0	986.790000	0.378669	0.491803	0.113135	0.0	1.0	751.530019	7482.166344	374.108317

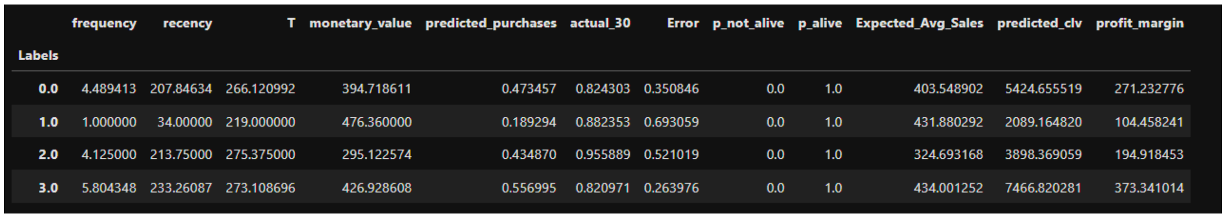
Apart from R, F, M, T, predicted-purchases, predicted-clv, Expected-Average-Sales, p-alive vales which were mentioned above, we also calculated a few more values –

p_not_alive: which is (1 – p_alive)
Actual_30: actual number of purchases by user. Calculated using

*(Frequency/recency) * T*

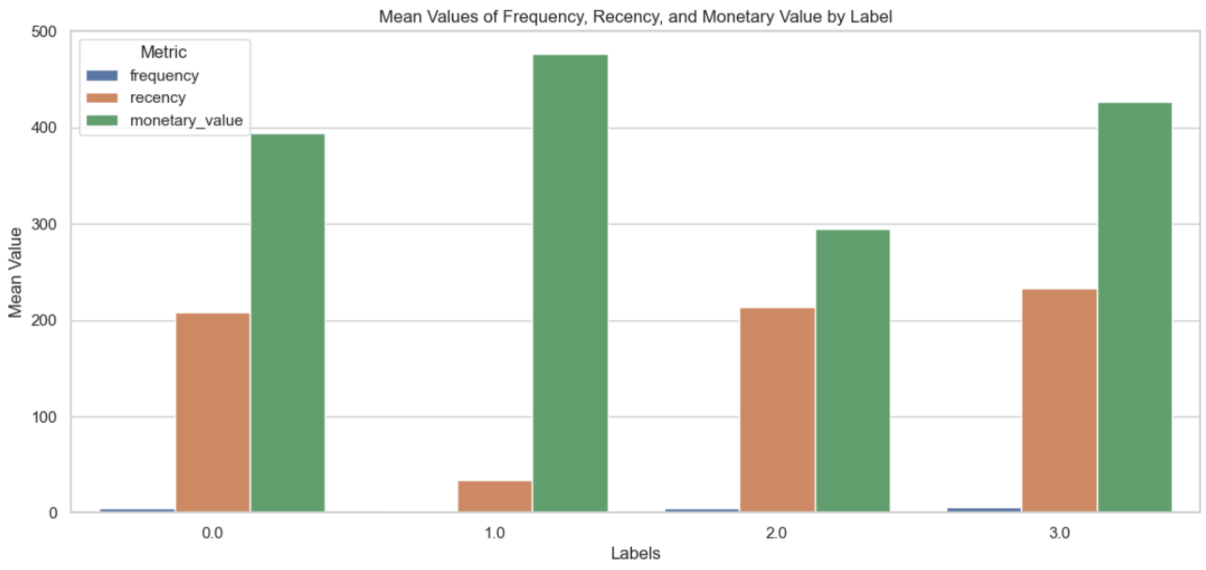
where T is the observation time period i.e. 30 in our case.
Note this is not number of days, it is the number of observation period.

On the output data from lifetimes package, we have applied K-Means clustering. 4 clusters were created. Below is the mean summary of labels -



Labels	frequency	recency	T	monetary_value	predicted_purchases	actual_30	Error	p_not_alive	p_alive	Expected_Avg_Sales	predicted_clv	profit_margin
0.0	4.489413	207.84634	266.120992	394.718611	0.473457	0.824303	0.350846	0.0	1.0	403.548902	5424.655519	271.232776
1.0	1.000000	34.00000	219.000000	476.360000	0.189294	0.882353	0.693059	0.0	1.0	431.880292	2089.164820	104.458241
2.0	4.125000	213.75000	275.375000	295.122574	0.434870	0.955889	0.521019	0.0	1.0	324.693168	3898.369059	194.918453
3.0	5.804348	233.26087	273.108696	426.928608	0.556995	0.820971	0.263976	0.0	1.0	434.001252	7466.820281	373.341014

Below is the RFM distribution of the labels –



Based on the above data, we can summarize the customer characteristics for each label. This will help us to understand the types of customers represented by each label -

Label 0: These customers have a moderate frequency of purchases and their recent activity is quite high. Their monetary value is lower than Labels 1 and 3, which might indicate they are regular but not high-spending customers.

Label 1: This label represents a unique group with a high frequency of purchases and the *highest monetary value, suggesting these are premium customers, possibly bulk buyers or loyal customers with high engagement.*

Label 2: These customers have a low average frequency of purchases and their recent activity varies widely. They have the lowest average monetary value among all labels, indicating they may be occasional shoppers or new customers.

Label 3: Customers in this label also show a high frequency of purchases and a high monetary value, but with a slightly lower average than Label 1. They could be considered as regular customers with significant spending.

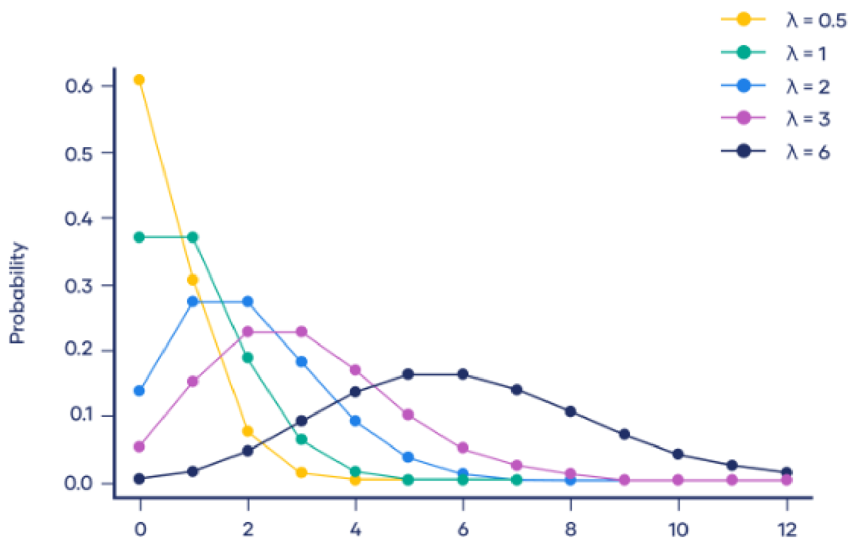
From this data alone, we can infer that 'Label 0' represent the customer best for marketing considering we are looking for 'Persuadable' segment of customers for launching our marketing campaign.

Simulations on ModifiedBetaGeo data using Poisson package + K-Means on the simulation data

The lifetimes package has already provided with future predictions on purchases, average sales, clv, and we have already done customer segmentation using k-means. But whether it is enough or not will always be debatable and the acceptance of output data depends on business case needs.

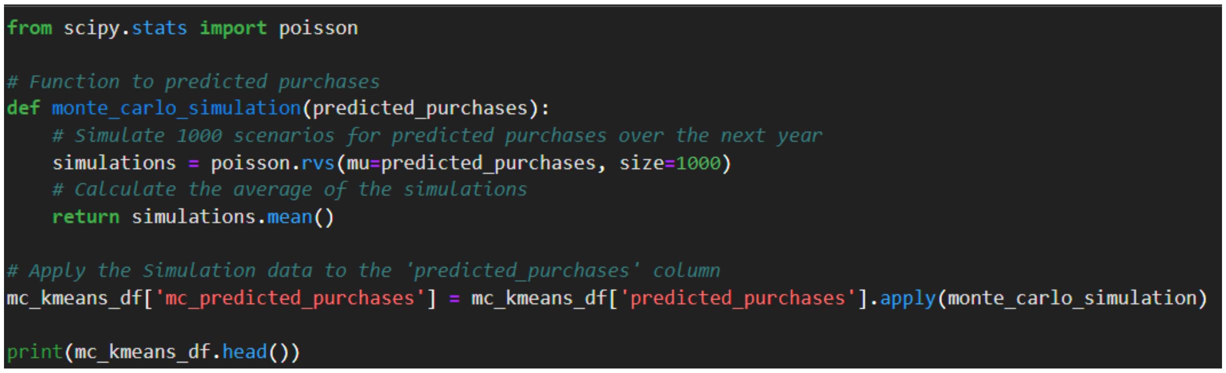
Even though this field of customer lifetime value evaluation is still under study, there has already been enough study been done on this topic. And we can use some of the accepted industry truths to further our analysis and evaluation.

Studies have shown that customer purchase lifecycle always follows a Poisson distribution curve.



In our scenario, Poisson distribution can be used to model the number of purchases made by a customer in a given time period, such as a day, a week, or a month. This can help to identify different types of customers, such as frequent buyers, occasional buyers, or inactive buyers.

To run the simulation, we have used scipy's Poisson package.



```python
from scipy.stats import poisson

# Function to predicted purchases
def monte_carlo_simulation(predicted_purchases):
    # Simulate 1000 scenarios for predicted purchases over the next year
    simulations = poisson.rvs(mu=predicted_purchases, size=1000)
    # Calculate the average of the simulations
    return simulations.mean()

# Apply the Simulation data to the 'predicted_purchases' column
mc_kmeans_df['mc_predicted_purchases'] = mc_kmeans_df['predicted_purchases'].apply(monte_carlo_simulation)

print(mc_kmeans_df.head())
```

Predicted output is stored under a new column mc_predicted_purchases which will be used for analysis.

One this data is collected, k-means clustering is again applied on it.

Only these columns were used for clustering –

'frequency',

'recency',

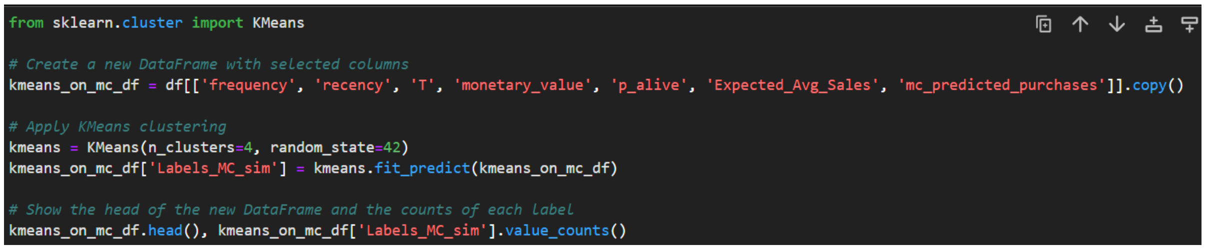
'T',

'monetary_value',

'p_alive',

'Expected_Avg_Sales',

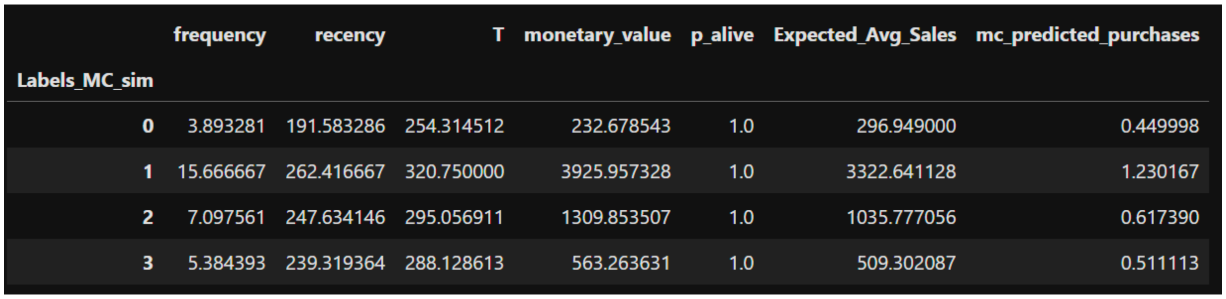
'mc_predicted_purchases'



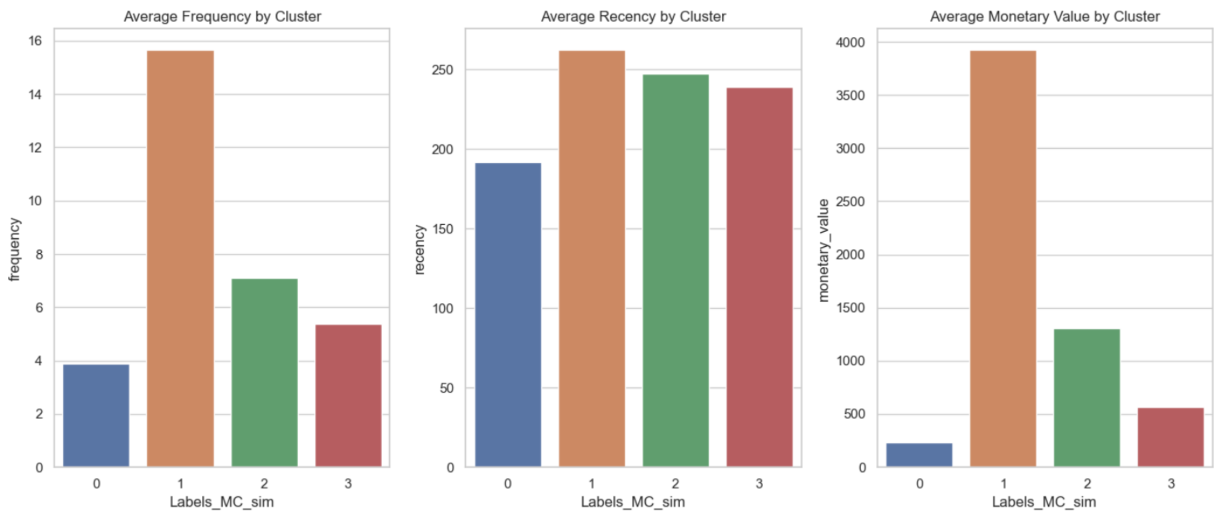
```python
from sklearn.cluster import KMeans

# Create a new DataFrame with selected columns
kmeans_on_mc_df = df[['frequency', 'recency', 'T', 'monetary_value', 'p_alive', 'Expected_Avg_Sales', 'mc_predicted_purchases']].copy()

# Apply KMeans clustering
kmeans = KMeans(n_clusters=4, random_state=42)
kmeans_on_mc_df['Labels_MC_sim'] = kmeans.fit_predict(kmeans_on_mc_df)

# Show the head of the new DataFrame and the counts of each label
kmeans_on_mc_df.head(), kmeans_on_mc_df['Labels_MC_sim'].value_counts()
```

Below is the result of clustering –

mean() value per label –



Labels_MC_sim	frequency	recency	T	monetary_value	p_alive	Expected_Avg_Sales	mc_predicted_purchases
0	3.893281	191.583286	254.314512	232.678543	1.0	296.949000	0.449998
1	15.666667	262.416667	320.750000	3925.957328	1.0	3322.641128	1.230167
2	7.097561	247.634146	295.056911	1309.853507	1.0	1035.777056	0.617390
3	5.384393	239.319364	288.128613	563.263631	1.0	509.302087	0.511113



Label Characteristics based on the visualizations:

Label 0: customers have a moderate frequency and recency of purchases with lower monetary value. They could represent occasional shoppers who spend less.

Label 1: customers have the highest frequency and monetary value, indicating they are likely to be premium customers with consistent and high-value purchases.

Label 2: customers have an above-moderate frequency and high monetary value, suggesting they are valuable customers with significant spending but less frequent purchases than Label 1.

Label 3: customers have moderate values across frequency, recency, and monetary value, which might indicate regular customers with average spending habits.

Here we can clearly see that Label 2 represents customers which are –

1. *Active*
2. *Above average frequency i.e. they buy often*
3. *Average monetary value i.e. they are not spending much even though they are buying regularly*
4. *They have been in the system from a long time i.e. they are loyal.*

Targeting our marketing potential on these users could be beneficial.

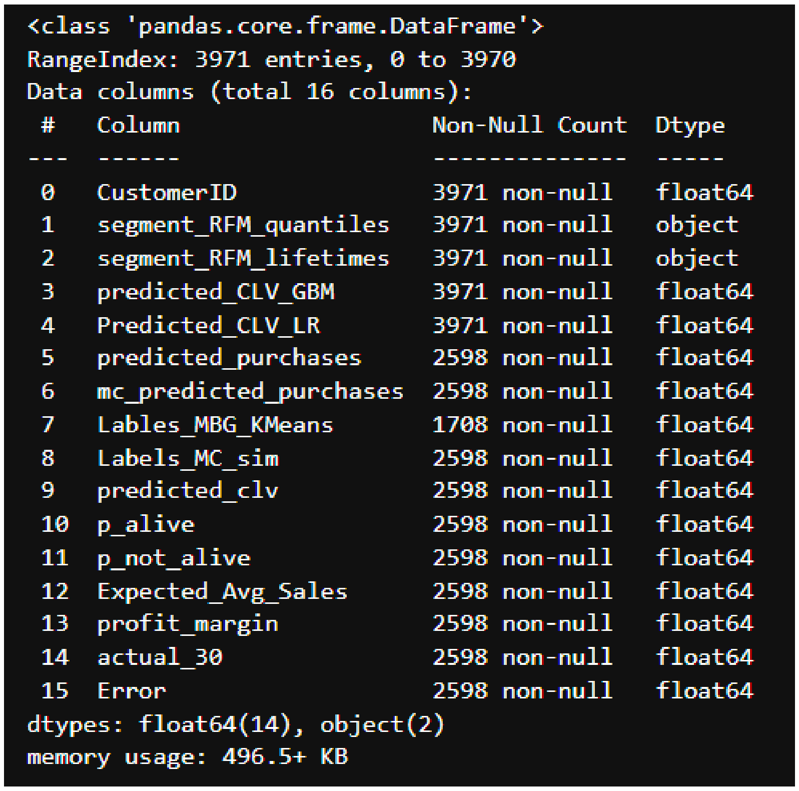
RESULTS

Let's dive into the results as we are done with the data analysis using various techniques.

A super dataframe was crated which contained the output from all the techniques used (just a refresher)–

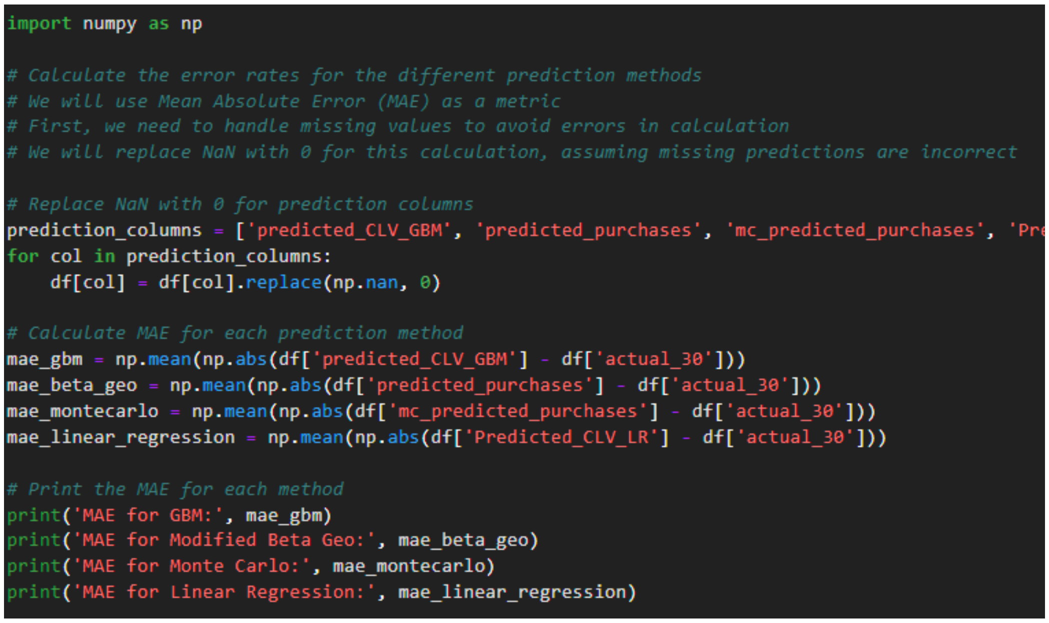
1. *RFM segments using Quantiles*
2. *RFM segments using Lifetimes*
3. *Linear Regression output*
4. *Gradient Boosting Machine output*
5. *MBG/NBD output*
6. *K-Means clustering labels on MBG/MBD data*
7. *Purchase Prediction via Poisson.rvs*
8. *K-means clustering labels on poisson.rvs output along with some data from MBG/NBD*

Below is the super_df dataframe –



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3971 entries, 0 to 3970
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   CustomerID             3971 non-null   float64
 1   segment_RFM_quantiles  3971 non-null   object 
 2   segment_RFM_lifetimes  3971 non-null   object 
 3   predicted_CLV_GBM      3971 non-null   float64
 4   Predicted_CLV_LR       3971 non-null   float64
 5   predicted_purchases    2598 non-null   float64
 6   mc_predicted_purchases 2598 non-null   float64
 7   Lables_MBG_KMeans      1708 non-null   float64
 8   Labels_MC_sim          2598 non-null   float64
 9   predicted_clv          2598 non-null   float64
 10  p_alive                2598 non-null   float64
 11  p_not_alive            2598 non-null   float64
 12  Expected_Avg_Sales     2598 non-null   float64
 13  profit_margin          2598 non-null   float64
 14  actual_30              2598 non-null   float64
 15  Error                  2598 non-null   float64
dtypes: float64(14), object(2)
memory usage: 496.5+ KB
```

1. *Analyzing predicted purchases*

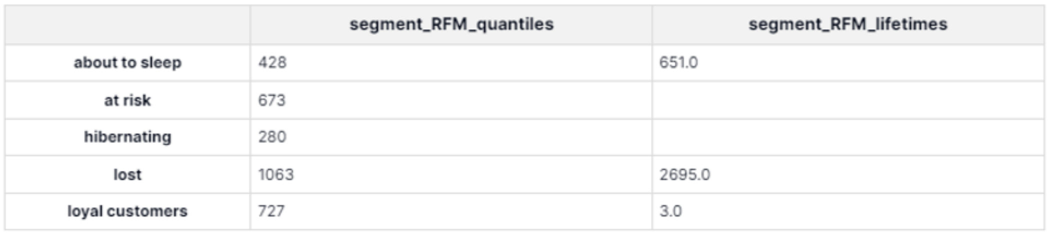


```python
import numpy as np

# Calculate the error rates for the different prediction methods
# We will use Mean Absolute Error (MAE) as a metric
# First, we need to handle missing values to avoid errors in calculation
# We will replace NaN with 0 for this calculation, assuming missing predictions are incorrect

# Replace NaN with 0 for prediction columns
prediction_columns = ['predicted_CLV_GBM', 'predicted_purchases', 'mc_predicted_purchases', 'Pre
for col in prediction_columns:
    df[col] = df[col].replace(np.nan, 0)

# Calculate MAE for each prediction method
mae_gbm = np.mean(np.abs(df['predicted_CLV_GBM'] - df['actual_30']))
mae_beta_geo = np.mean(np.abs(df['predicted_purchases'] - df['actual_30']))
mae_montecarlo = np.mean(np.abs(df['mc_predicted_purchases'] - df['actual_30']))
mae_linear_regression = np.mean(np.abs(df['Predicted_CLV_LR'] - df['actual_30']))

# Print the MAE for each method
print('MAE for GBM:', mae_gbm)
print('MAE for Modified Beta Geo:', mae_beta_geo)
print('MAE for Monte Carlo:', mae_montecarlo)
print('MAE for Linear Regression:', mae_linear_regression)
```

The Modified Beta Geo and Monte Carlo methods have very similar and the lowest MAE, indicating they are better at predicting the actual purchases in the next 30 days compared to GBM and Linear Regression.

2. *RFM segments vs Predicted_CLV_LR*

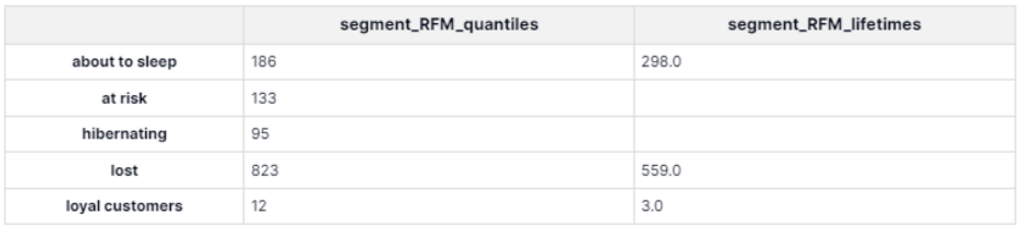
Here we are checking the count of lables in column 'segment_RFM_quantiles', 'segment_RFM_lifetimes' -



	segment_RFM_quantiles	segment_RFM_lifetimes
about to sleep	428	651.0
at risk	673	
hibernating	280	
lost	1063	2695.0
loyal customers	727	3.0

Looks like lifetimes have put more people in 'lost' compared to quantiles. So, quantiles RFM worked better in estimating labels on Linear Regression data.

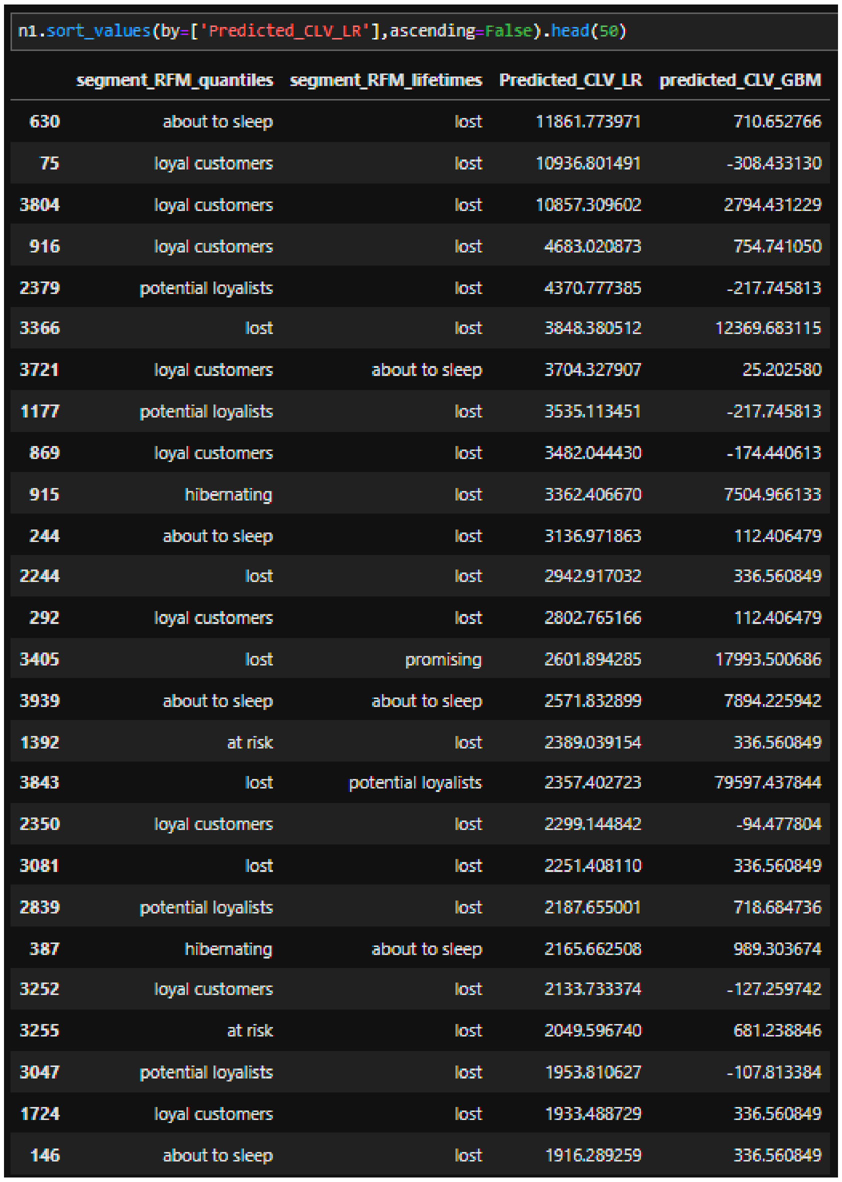
3. *RFM segments vs Predicted_CLV_GBM*
   We did the same with GBM predicted data



	segment_RFM_quantiles	segment_RFM_lifetimes
about to sleep	186	298.0
at risk	133	
hibernating	95	
lost	823	559.0
loyal customers	12	3.0

Here it looks like lifetimes has put less people under 'lost'. So, lifetimes RFM worked better in estimating labels on GBM data.

4. *Descending order of predicted_CLV_LR data*

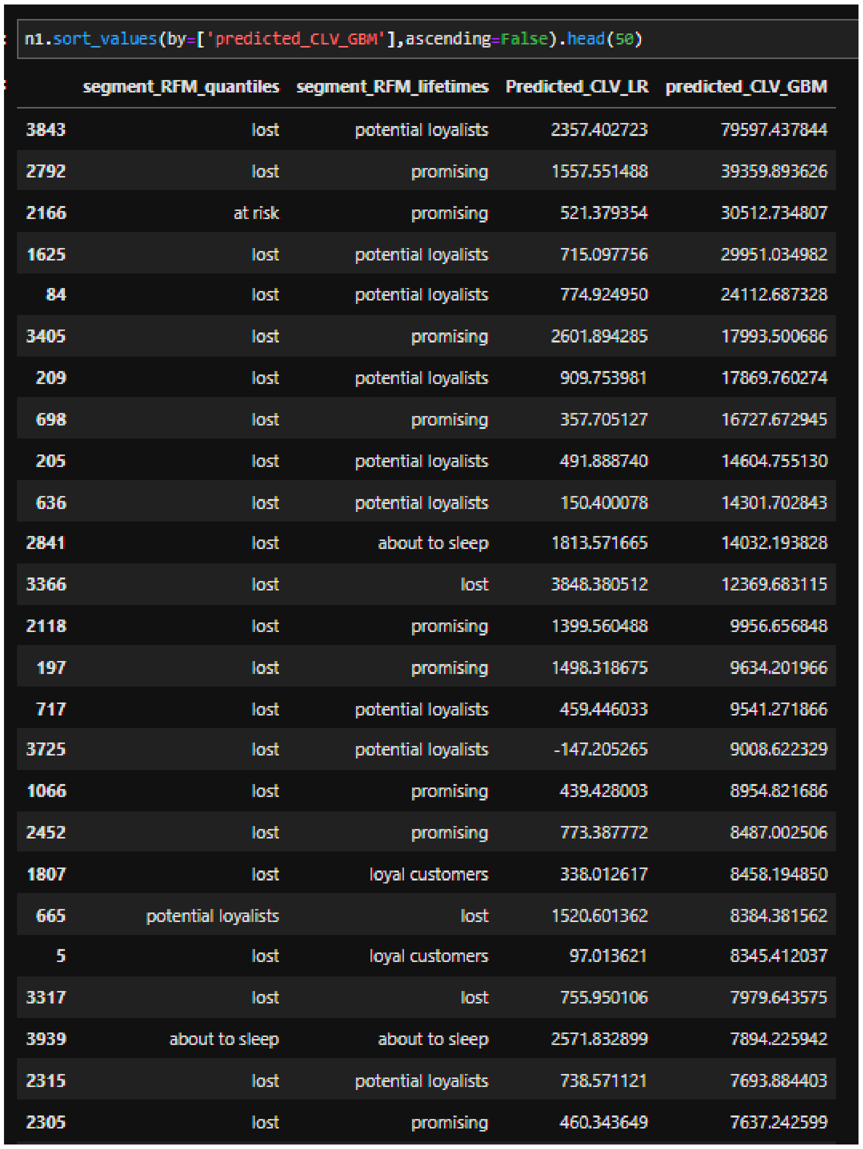


```
n1.sort_values(by=['Predicted_CLV_LR'],ascending=False).head(50)
```

	segment_RFM_quantiles	segment_RFM_lifetimes	Predicted_CLV_LR	predicted_CLV_GBM
630	about to sleep	lost	11861.773971	710.652766
75	loyal customers	lost	10936.801491	-308.433130
3804	loyal customers	lost	10857.309602	2794.431229
916	loyal customers	lost	4683.020873	754.741050
2379	potential loyalists	lost	4370.777385	-217.745813
3366	lost	lost	3848.380512	12369.683115
3721	loyal customers	about to sleep	3704.327907	25.202580
1177	potential loyalists	lost	3535.113451	-217.745813
869	loyal customers	lost	3482.044430	-174.440613
915	hibernating	lost	3362.406670	7504.966133
244	about to sleep	lost	3136.971863	112.406479
2244	lost	lost	2942.917032	336.560849
292	loyal customers	lost	2802.765166	112.406479
3405	lost	promising	2601.894285	17993.500686
3939	about to sleep	about to sleep	2571.832899	7894.225942
1392	at risk	lost	2389.039154	336.560849
3843	lost	potential loyalists	2357.402723	79597.437844
2350	loyal customers	lost	2299.144842	-94.477804
3081	lost	lost	2251.408110	336.560849
2839	potential loyalists	lost	2187.655001	718.684736
387	hibernating	about to sleep	2165.662508	989.303674
3252	loyal customers	lost	2133.733374	-127.259742
3255	at risk	lost	2049.596740	681.238846
3047	potential loyalists	lost	1953.810627	-107.813384
1724	loyal customers	lost	1933.488729	336.560849
146	about to sleep	lost	1916.289259	336.560849

When we check the CLV_LR score in descending order, it appears that quantiles RFM is doing better in predicting customer value.

5. *Descending order of predicted_CLV_GBM data*



```
n1.sort_values(by=['predicted_CLV_GBM'],ascending=False).head(50)
```

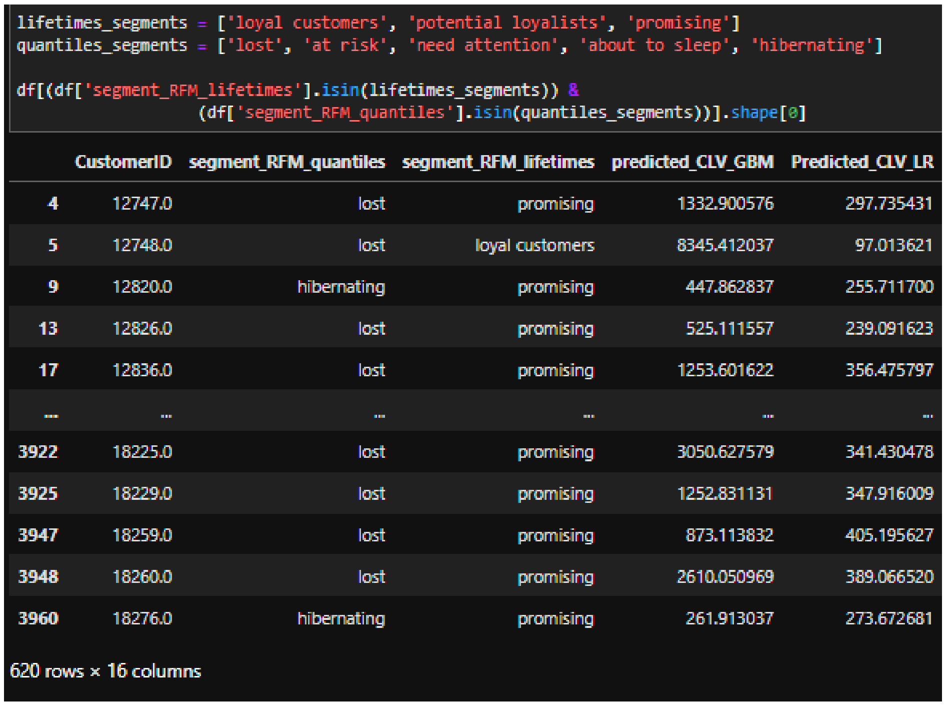
	segment_RFM_quantiles	segment_RFM_lifetimes	Predicted_CLV_LR	predicted_CLV_GBM
3843	lost	potential loyalists	2357.402723	79597.437844
2792	lost	promising	1557.551488	39359.893626
2166	at risk	promising	521.379354	30512.734807
1625	lost	potential loyalists	715.097756	29951.034982
84	lost	potential loyalists	774.924950	24112.687328
3405	lost	promising	2601.894285	17993.500686
209	lost	potential loyalists	909.753981	17869.760274
698	lost	promising	357.705127	16727.672945
205	lost	potential loyalists	491.888740	14604.755130
636	lost	potential loyalists	150.400078	14301.702843
2841	lost	about to sleep	1813.571665	14032.193828
3366	lost	lost	3848.380512	12369.683115
2118	lost	promising	1399.560488	9956.656848
197	lost	promising	1498.318675	9634.201966
717	lost	potential loyalists	459.446033	9541.271866
3725	lost	potential loyalists	-147.205265	9008.622329
1066	lost	promising	439.428003	8954.821686
2452	lost	promising	773.387772	8487.002506
1807	lost	loyal customers	338.012617	8458.194850
665	potential loyalists	lost	1520.601362	8384.381562
5	lost	loyal customers	97.013621	8345.412037
3317	lost	lost	755.950106	7979.643575
3939	about to sleep	about to sleep	2571.832899	7894.225942
2315	lost	potential loyalists	738.571121	7693.884403
2305	lost	promising	460.343649	7637.242599

When we check the CLV_GBM score in descending order, it appears that lifetimes RFM is doing better in predicting customer value.

This is intuitive since we used RFMT data from lifetimes package also as feature variables, it can be safe to say that GBM model has worked well to predict the future purchase since it is coinciding with the lables created using lieftimes RFM alone.

However, the r2_scre for GBM was lower than LR i.e. 63% vs 98%
So, the takeaway is that just simple score are not valuable in one go and also using single model may not be enough to come into conclusion.
We need to look further to understand the depth of the data.

6. *There were 620 customers which were marked as either 'lost, at risk, need attention, about to sleep, hibernating ' in segment_RFM_quantiles but were marked as either 'loyal customers, potential loyalists, promising' in segment_RFM_lifetimes.*

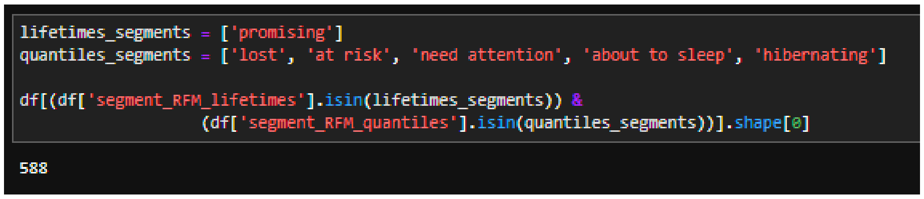


```
lifetimes_segments = ['loyal customers', 'potential loyalists', 'promising']
quantiles_segments = ['lost', 'at risk', 'need attention', 'about to sleep', 'hibernating']

df[(df['segment_RFM_lifetimes'].isin(lifetimes_segments)) &
                  (df['segment_RFM_quantiles'].isin(quantiles_segments))].shape[0]
```

	CustomerID	segment_RFM_quantiles	segment_RFM_lifetimes	predicted_CLV_GBM	Predicted_CLV_LR
4	12747.0	lost	promising	1332.900576	297.735431
5	12748.0	lost	loyal customers	8345.412037	97.013621
9	12820.0	hibernating	promising	447.862837	255.711700
13	12826.0	lost	promising	525.111557	239.091623
17	12836.0	lost	promising	1253.601622	356.475797
...	...	...	...	...	...
3922	18225.0	lost	promising	3050.627579	341.430478
3925	18229.0	lost	promising	1252.831131	347.916009
3947	18259.0	lost	promising	873.113832	405.195627
3948	18260.0	lost	promising	2610.050969	389.066520
3960	18276.0	hibernating	promising	261.913037	273.672681

620 rows × 16 columns

7. *588 'promising' customers in LIFETIMES were marked as either 'lost', 'at risk', 'need attention', 'about to sleep', 'hibernating' in Quantiles*



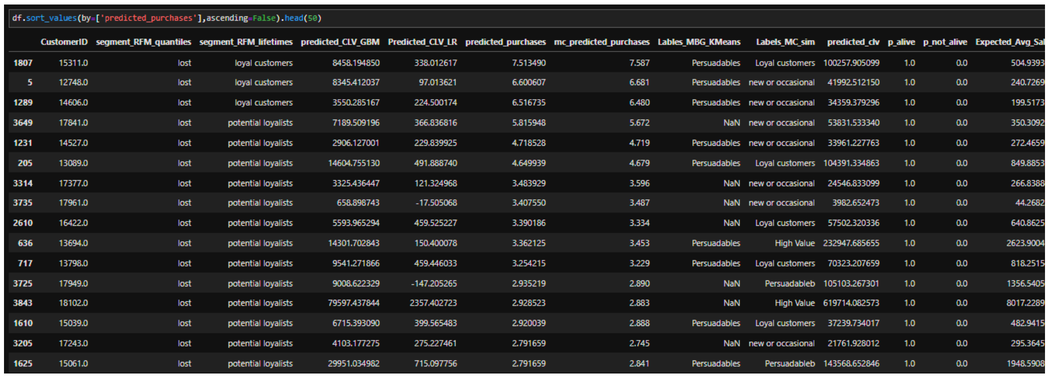
```
lifetimes_segments = ['promising']
quantiles_segments = ['lost', 'at risk', 'need attention', 'about to sleep', 'hibernating']

df[(df['segment_RFM_lifetimes'].isin(lifetimes_segments)) &
                  (df['segment_RFM_quantiles'].isin(quantiles_segments))].shape[0]
```

```
588
```

8. *For predicted_purchases from MBG/NBD - rfm_lifetimes is performing better,*

Lables_MBG_KMeans is acceptable,

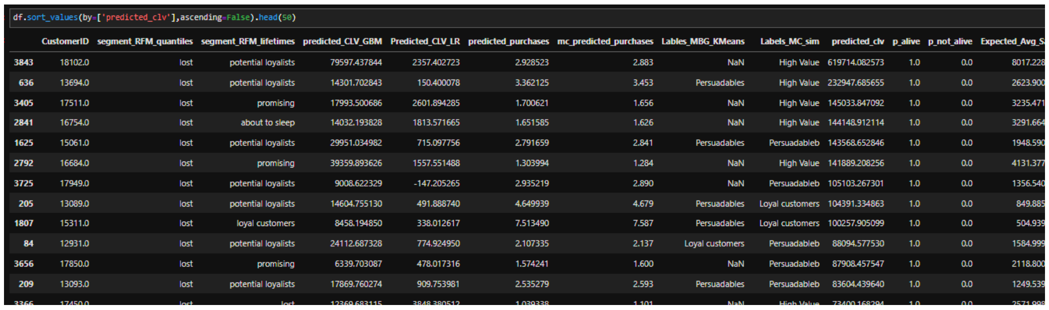
Labels_MC_sim is not promising.



```
df.sort_values(by=['predicted_purchases'],ascending=False).head(50)
```

	CustomerID	segment_RFM_quantiles	segment_RFM_lifetimes	predicted_CLV_GBM	Predicted_CLV_LR	predicted_purchases	mc_predicted_purchases	Lables_MBG_KMeans	Labels_MC_sim	predicted_clv	p_alive	p_not_alive	Expected_Avg_Sal
1807	15311.0	lost	loyal customers	8458.194850	338.012617	7.513490	7.587	Persuadables	Loyal customers	100257.905099	1.0	0.0	504.9393
5	12748.0	lost	loyal customers	8345.412037	97.013621	6.600607	6.681	Persuadables	new or occasional	41992.512150	1.0	0.0	240.7269
1289	14606.0	lost	loyal customers	3550.285167	224.500174	6.516735	6.480	Persuadables	new or occasional	34359.379296	1.0	0.0	199.5173
3649	17841.0	lost	potential loyalists	7189.509196	366.836816	5.815948	5.672	NaN	new or occasional	53831.533340	1.0	0.0	350.3092
1231	14527.0	lost	potential loyalists	2906.127001	229.839925	4.718528	4.719	Persuadables	new or occasional	33961.227763	1.0	0.0	272.4659
205	13089.0	lost	potential loyalists	14604.755130	491.888740	4.649939	4.679	Persuadables	Loyal customers	104391.334863	1.0	0.0	849.8853
3314	17377.0	lost	potential loyalists	3325.436447	121.324968	3.483929	3.596	NaN	new or occasional	24546.833099	1.0	0.0	266.8388
3735	17961.0	lost	potential loyalists	658.898743	-17.505068	3.407550	3.487	NaN	new or occasional	3982.652473	1.0	0.0	44.2682
2610	16422.0	lost	potential loyalists	5593.965294	459.525227	3.390186	3.334	NaN	Loyal customers	57502.320336	1.0	0.0	640.8625
636	13694.0	lost	potential loyalists	14301.702843	150.400078	3.362125	3.453	Persuadables	High Value	232947.685655	1.0	0.0	2623.9004
717	13798.0	lost	potential loyalists	9541.271866	459.446033	3.254215	3.229	Persuadables	Loyal customers	70323.207659	1.0	0.0	818.2515
3725	17949.0	lost	potential loyalists	9008.622329	-147.205265	2.935219	2.890	NaN	Persuadableb	105103.267301	1.0	0.0	1356.5405
3843	18102.0	lost	potential loyalists	79597.437844	2357.402723	2.928523	2.883	NaN	High Value	619714.082573	1.0	0.0	8017.2289
1610	15039.0	lost	potential loyalists	6715.393090	399.565483	2.920039	2.888	Persuadables	Loyal customers	37239.734017	1.0	0.0	482.9415
3205	17243.0	lost	potential loyalists	4103.177275	275.227461	2.791659	2.745	NaN	new or occasional	21761.928012	1.0	0.0	295.3645
1625	15061.0	lost	potential loyalists	29951.034982	715.097756	2.791659	2.841	Persuadables	Persuadableb	143568.652846	1.0	0.0	1948.5908

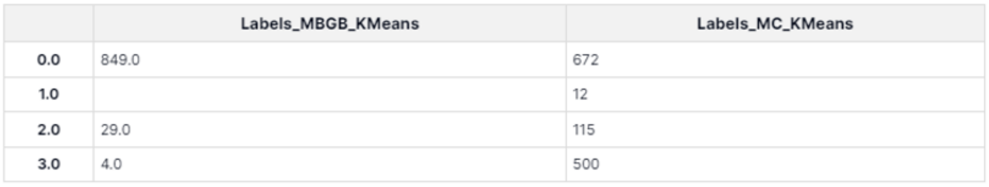
9. *For predcted_clv from MBG/NBD -*
   rfm_quantiles is performed poorly,
   rfm_lifetimes if performing better,
   Labels from K-means on poisson simulation is far better than k-means labels from MBG,
   K-Means labels from MBG is showing persudables, which is good too,
   Expected_Avg_Sales is starting from max values and showing above mean/median and near max value,
   profit_margin is starting from max values and showing above mean/median and near max value,
   Error is around mean value.



```
df.sort_values(by=['predicted_clv'],ascending=False).head(50)
```

	CustomerID	segment_RFM_quantiles	segment_RFM_lifetimes	predicted_CLV_GBM	Predicted_CLV_LR	predicted_purchases	mc_predicted_purchases	Lables_MBG_KMeans	Labels_MC_sim	predicted_clv	p_alive	p_not_alive	Expected_Avg_S
3843	18102.0	lost	potential loyalists	79597.437844	2357.402723	2.928523	2.883	NaN	High Value	619714.082573	1.0	0.0	8017.228
636	13694.0	lost	potential loyalists	14301.702843	150.400078	3.362125	3.453	Persuadables	High Value	232947.685655	1.0	0.0	2623.900
3405	17511.0	lost	promising	17993.500686	2601.894285	1.700621	1.656	NaN	High Value	145033.847092	1.0	0.0	3235.471
2841	16754.0	lost	about to sleep	14032.193828	1813.571665	1.651585	1.626	NaN	High Value	144148.912114	1.0	0.0	3291.664
1625	15061.0	lost	potential loyalists	29951.034982	715.097756	2.791659	2.841	Persuadables	Persuadableb	143568.652846	1.0	0.0	1948.590
2792	16684.0	lost	promising	39359.893626	1557.551488	1.303994	1.284	NaN	High Value	141889.208256	1.0	0.0	4131.377
3725	17949.0	lost	potential loyalists	9008.622329	-147.205265	2.935219	2.890	NaN	Persuadableb	105103.267301	1.0	0.0	1356.540
205	13089.0	lost	potential loyalists	14604.755130	491.888740	4.649939	4.679	Persuadables	Loyal customers	104391.334863	1.0	0.0	849.885
1807	15311.0	lost	loyal customers	8458.194850	338.012617	7.513490	7.587	Persuadables	Loyal customers	100257.905099	1.0	0.0	504.939
84	12931.0	lost	potential loyalists	24112.687328	774.924950	2.107335	2.137	Loyal customers	Persuadableb	88094.577530	1.0	0.0	1584.999
3656	17850.0	lost	promising	6339.703087	478.017316	1.574241	1.600	NaN	Persuadableb	87908.457547	1.0	0.0	2118.800
209	13093.0	lost	potential loyalists	17869.760274	909.753981	2.535279	2.593	Persuadables	Persuadableb	83604.439640	1.0	0.0	1249.539
3366	17450.0	lost	lost	12369.683115	3848.380512	1.039338	1.101	NaN	High Value	73400.168294	1.0	0.0	2571.998

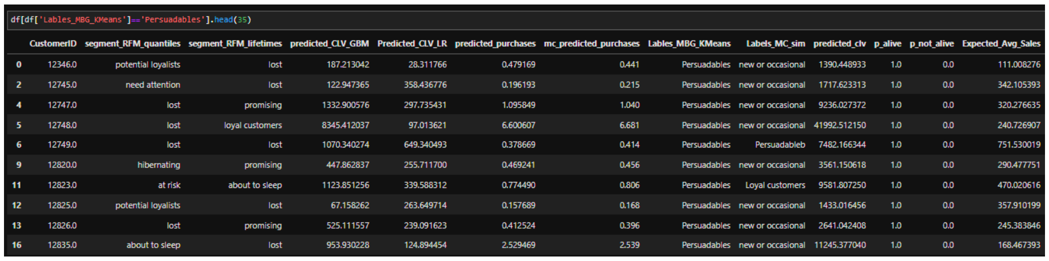
10. *Count of labels by K-means on MBG data and K-means on Poisson simulation data for customers with a 'predicted_clv' value greater than or equal to the median.*



	Labels_MBGB_KMeans	Labels_MC_KMeans
0.0	849.0	672
1.0		12
2.0	29.0	115
3.0	4.0	500

Labels are mostly consistent in recognizing patterns but there are more 'persuadable' in k-means labels on MBG data rather than Poisson simulation data. Let's see further.

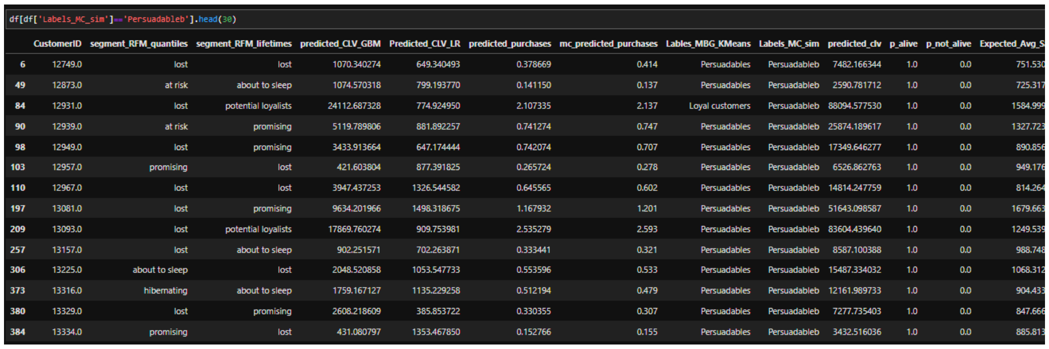
11. *For MBG k-means labels as Persuadable -*
    predicted_clv is in the range 25% to 75%
    Expected_Avg_Sales is in the range 25% to 75%
    segment_RFM_quantiles shows 'lost' & 'at risk'
    segment_RFM_lifetimes is performing well with mix of customer from 'about to sleep' to 'loyal' with some 'lost'



```
df[df['Lables_MBG_KMeans']=='Persuadables'].head(35)
```

	CustomerID	segment_RFM_quantiles	segment_RFM_lifetimes	predicted_CLV_GBM	Predicted_CLV_LR	predicted_purchases	mc_predicted_purchases	Lables_MBG_KMeans	Labels_MC_sim	predicted_clv	p_alive	p_not_alive	Expected_Avg_Sales
0	12346.0	potential loyalists	lost	187.213042	28.311766	0.479169	0.441	Persuadables	new or occasional	1390.448933	1.0	0.0	111.008276
2	12745.0	need attention	lost	122.947365	358.436776	0.196193	0.215	Persuadables	new or occasional	1717.623313	1.0	0.0	342.105393
4	12747.0	lost	promising	1332.900576	297.735431	1.095849	1.040	Persuadables	new or occasional	9236.027372	1.0	0.0	320.276635
5	12748.0	lost	loyal customers	8345.412037	97.013621	6.600607	6.681	Persuadables	new or occasional	41992.512150	1.0	0.0	240.726907
6	12749.0	lost	lost	1070.340274	649.340493	0.378669	0.414	Persuadables	Persuadableb	7482.166344	1.0	0.0	751.530019
9	12820.0	hibernating	promising	447.862837	255.711700	0.469241	0.456	Persuadables	new or occasional	3561.150618	1.0	0.0	290.477751
11	12823.0	at risk	about to sleep	1123.851256	339.588312	0.774490	0.806	Persuadables	Loyal customers	9581.807250	1.0	0.0	470.020616
12	12825.0	potential loyalists	lost	67.158262	263.649714	0.157689	0.168	Persuadables	new or occasional	1433.016456	1.0	0.0	357.910199
13	12826.0	lost	promising	525.111557	239.091623	0.412524	0.396	Persuadables	new or occasional	2641.042408	1.0	0.0	245.383846
16	12835.0	about to sleep	lost	953.930228	124.894454	2.529469	2.539	Persuadables	new or occasional	11245.377040	1.0	0.0	168.467393

12. *For K-Means Lables on Possion Simulation data -*
    segment_RFM_lifetimes shows 'loyal' and 'loyalist'
    Expected_Avg_Sales is in 50% to 75% mark



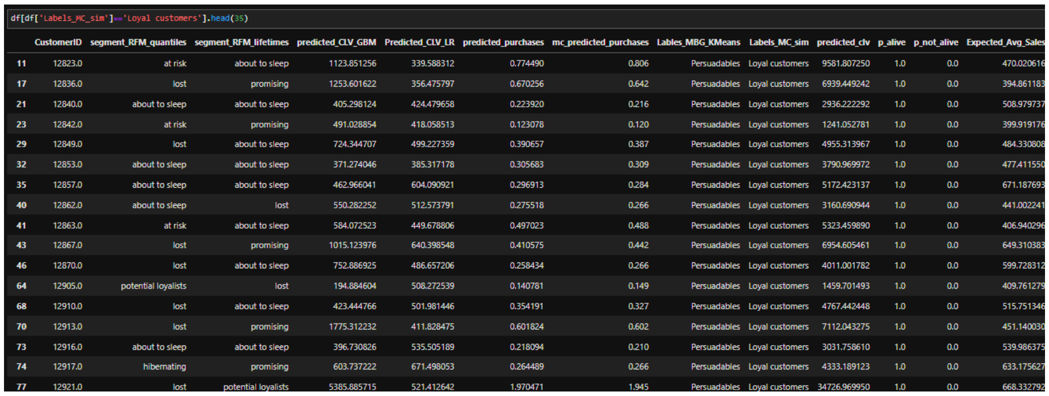
```
df[df['Labels_MC_sim']=='Persuadableb'].head(30)
```

	CustomerID	segment_RFM_quantiles	segment_RFM_lifetimes	predicted_CLV_GBM	Predicted_CLV_LR	predicted_purchases	mc_predicted_purchases	Lables_MBG_KMeans	Labels_MC_sim	predicted_clv	p_alive	p_not_alive	Expected_Avg_S
6	12749.0	lost	lost	1070.340274	649.340493	0.378669	0.414	Persuadables	Persuadableb	7482.166344	1.0	0.0	751.530
49	12873.0	at risk	about to sleep	1074.570318	799.193770	0.141150	0.137	Persuadables	Persuadableb	2590.781712	1.0	0.0	725.317
84	12931.0	lost	potential loyalists	24112.687328	774.924950	2.107335	2.137	Loyal customers	Persuadableb	88094.577530	1.0	0.0	1584.999
90	12939.0	at risk	promising	5119.789806	881.892257	0.741274	0.747	Persuadables	Persuadableb	25874.189617	1.0	0.0	1327.723
98	12949.0	lost	promising	3433.913664	647.174444	0.742074	0.707	Persuadables	Persuadableb	17349.646277	1.0	0.0	890.856
103	12957.0	promising	lost	421.603804	877.391825	0.265724	0.278	Persuadables	Persuadableb	6526.862763	1.0	0.0	949.176
110	12967.0	lost	lost	3947.437253	1326.544582	0.645565	0.602	Persuadables	Persuadableb	14814.247759	1.0	0.0	814.264
197	13081.0	lost	promising	9634.201966	1498.318675	1.167932	1.201	Persuadables	Persuadableb	51643.098587	1.0	0.0	1679.663
209	13093.0	lost	potential loyalists	17869.760274	909.753981	2.535279	2.593	Persuadables	Persuadableb	83604.439640	1.0	0.0	1249.539
257	13157.0	lost	about to sleep	902.251571	702.263871	0.333441	0.321	Persuadables	Persuadableb	8587.100388	1.0	0.0	988.748
306	13225.0	about to sleep	lost	2048.520858	1053.547733	0.553596	0.533	Persuadables	Persuadableb	15487.334032	1.0	0.0	1068.312
373	13316.0	hibernating	about to sleep	1759.167127	1135.229258	0.512194	0.479	Persuadables	Persuadableb	12161.989733	1.0	0.0	904.433
380	13329.0	lost	promising	2608.218609	385.853722	0.330355	0.307	Persuadables	Persuadableb	7277.735403	1.0	0.0	847.666
384	13334.0	promising	lost	431.080797	1353.467850	0.152766	0.155	Persuadables	Persuadableb	3432.516036	1.0	0.0	885.813

13. *Upon checking values for 'Loyal Customers' in K-Means on Poisson simulation data, we see that it is consistent with -*

segment_RFM_lifetimes

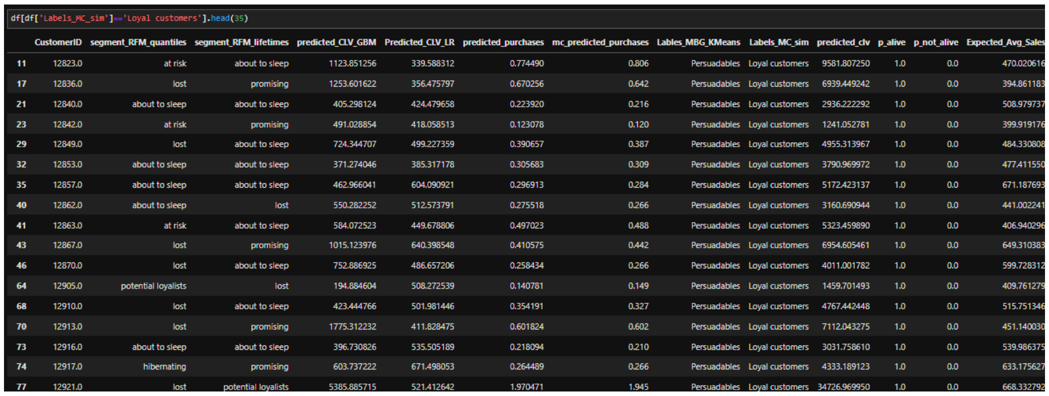
Expected_Avg_Sales is within 50% to 75% i.e. around mean & above



```
df[df['Labels_MC_sim']=='Loyal customers'].head(35)
```

	CustomerID	segment_RFM_quantiles	segment_RFM_lifetimes	predicted_CLV_GBM	Predicted_CLV_LR	predicted_purchases	mc_predicted_purchases	Lables_MBG_KMeans	Labels_MC_sim	predicted_clv	p_alive	p_not_alive	Expected_Avg_Sales
11	12823.0	at risk	about to sleep	1123.851256	339.588312	0.774490	0.806	Persuadables	Loyal customers	9581.807250	1.0	0.0	470.020616
17	12836.0	lost	promising	1253.601622	356.475797	0.670256	0.642	Persuadables	Loyal customers	6939.449242	1.0	0.0	394.861183
21	12840.0	about to sleep	about to sleep	405.298124	424.479658	0.223920	0.216	Persuadables	Loyal customers	2936.222292	1.0	0.0	508.979737
23	12842.0	at risk	promising	491.028854	418.058513	0.123078	0.120	Persuadables	Loyal customers	1241.052781	1.0	0.0	399.919176
29	12849.0	lost	about to sleep	724.344707	499.227359	0.390657	0.387	Persuadables	Loyal customers	4955.313967	1.0	0.0	484.330808
32	12853.0	about to sleep	about to sleep	371.274046	385.317178	0.305683	0.309	Persuadables	Loyal customers	3790.969972	1.0	0.0	477.411550
35	12857.0	about to sleep	about to sleep	462.966041	604.090921	0.296913	0.284	Persuadables	Loyal customers	5172.423137	1.0	0.0	671.187693
40	12862.0	about to sleep	lost	550.282252	512.573791	0.275518	0.266	Persuadables	Loyal customers	3160.690944	1.0	0.0	441.002241
41	12863.0	at risk	about to sleep	584.072523	449.678806	0.497023	0.488	Persuadables	Loyal customers	5323.459890	1.0	0.0	406.940296
43	12867.0	lost	promising	1015.123976	640.398548	0.410575	0.442	Persuadables	Loyal customers	6954.605461	1.0	0.0	649.310383
46	12870.0	lost	about to sleep	752.886925	486.657206	0.258434	0.266	Persuadables	Loyal customers	4011.001782	1.0	0.0	599.728312
64	12905.0	potential loyalists	lost	194.884604	508.272539	0.140781	0.149	Persuadables	Loyal customers	1459.701493	1.0	0.0	409.761279
68	12910.0	lost	about to sleep	423.444766	501.981446	0.354191	0.327	Persuadables	Loyal customers	4767.442448	1.0	0.0	515.751346
70	12913.0	lost	promising	1775.312232	411.828475	0.601824	0.602	Persuadables	Loyal customers	7112.043275	1.0	0.0	451.140030
73	12916.0	about to sleep	about to sleep	396.730826	535.505189	0.218094	0.210	Persuadables	Loyal customers	3031.758610	1.0	0.0	539.986375
74	12917.0	hibernating	promising	603.737222	671.498053	0.264489	0.266	Persuadables	Loyal customers	4333.189123	1.0	0.0	633.175627
77	12921.0	lost	potential loyalists	5385.885715	521.412642	1.970471	1.945	Persuadables	Loyal customers	34726.969950	1.0	0.0	668.332792

14. *Upon checkingExpected_Avg_Sales data, we see that -*
    it is consistent with k-means label on Poisson simulated data
    consistent with k-means labels on MBG data
    consistent with predicted_clv showing the max values in descending order
    segement_RFM_lifetimes lables are acceptable
    segment_RFM_quantiles labels are not acceptable as they are marked as lost
    predicted_CLV_GBM is showing max values in descending order.



```
df[df['Labels_MC_sim']=='Loyal customers'].head(35)
```

	CustomerID	segment_RFM_quantiles	segment_RFM_lifetimes	predicted_CLV_GBM	Predicted_CLV_LR	predicted_purchases	mc_predicted_purchases	Lables_MBG_KMeans	Labels_MC_sim	predicted_clv	p_alive	p_not_alive	Expected_Avg_Sales
11	12823.0	at risk	about to sleep	1123.851256	339.588312	0.774490	0.806	Persuadables	Loyal customers	9581.807250	1.0	0.0	470.020616
17	12836.0	lost	promising	1253.601622	356.475797	0.670256	0.642	Persuadables	Loyal customers	6939.449242	1.0	0.0	394.861183
21	12840.0	about to sleep	about to sleep	405.298124	424.479658	0.223920	0.216	Persuadables	Loyal customers	2936.222292	1.0	0.0	508.979737
23	12842.0	at risk	promising	491.028854	418.058513	0.123078	0.120	Persuadables	Loyal customers	1241.052781	1.0	0.0	399.919176
29	12849.0	lost	about to sleep	724.344707	499.227359	0.390657	0.387	Persuadables	Loyal customers	4955.313967	1.0	0.0	484.330808
32	12853.0	about to sleep	about to sleep	371.274046	385.317178	0.305683	0.309	Persuadables	Loyal customers	3790.969972	1.0	0.0	477.411550
35	12857.0	about to sleep	about to sleep	462.966041	604.090921	0.296913	0.284	Persuadables	Loyal customers	5172.423137	1.0	0.0	671.187693
40	12862.0	about to sleep	lost	550.282252	512.573791	0.275518	0.266	Persuadables	Loyal customers	3160.690944	1.0	0.0	441.002241
41	12863.0	at risk	about to sleep	584.072523	449.678806	0.497023	0.488	Persuadables	Loyal customers	5323.459890	1.0	0.0	406.940296
43	12867.0	lost	promising	1015.123976	640.398548	0.410575	0.442	Persuadables	Loyal customers	6954.605461	1.0	0.0	649.310383
46	12870.0	lost	about to sleep	752.886925	486.657206	0.258434	0.266	Persuadables	Loyal customers	4011.001782	1.0	0.0	599.728312
64	12905.0	potential loyalists	lost	194.884604	508.272539	0.140781	0.149	Persuadables	Loyal customers	1459.701493	1.0	0.0	409.761279
68	12910.0	lost	about to sleep	423.444766	501.981446	0.354191	0.327	Persuadables	Loyal customers	4767.442448	1.0	0.0	515.751346
70	12913.0	lost	promising	1775.312232	411.828475	0.601824	0.602	Persuadables	Loyal customers	7112.043275	1.0	0.0	451.140030
73	12916.0	about to sleep	about to sleep	396.730826	535.505189	0.218094	0.210	Persuadables	Loyal customers	3031.758610	1.0	0.0	539.986375
74	12917.0	hibernating	promising	603.737222	671.498053	0.264489	0.266	Persuadables	Loyal customers	4333.189123	1.0	0.0	633.175627
77	12921.0	lost	potential loyalists	5385.885715	521.412642	1.970471	1.945	Persuadables	Loyal customers	34726.969950	1.0	0.0	668.332792

CONCLUSIONS - 1

Looking at the various results in the above section we can conclude that –

1. *there are various ways to approach a problem.*
2. *there can be various methods to arrive to a result/score/accuracy*
3. *no method is 100% fool-proof*
4. *it is always beneficial to not solely rely on one method but always compare or cross verify with other methods.*
5. *there is no one size fit all. One method may work well is some scenario but perform poorly in another scenario.*

As we can see that RFM segments created with quantiles gave very poor results. There are high chances that it happened due to skewness of data as there were some high value transactions which and since quantiles are just based on 25% - 50% - 75%(in our scenario) distribution of the population, it heavily skewed the results too.

RFM segments created using lifetime package worked really well since the Recency, Frequency, Monetary Value were calculated on individual customer level and not on the complete distribution of the dataset as opposed to using quantiles. This overcomes the major drawback of using quantiles.

Linear Regression gave 98% score on train and test data and had the near perfect best fit line. The score and scatter plot are enough to make anyone happy that they their analysis, feature engineering, model building is sufficient. However, while analyzing the data we saw that LR outcomes were also not useful. When we compare the LR score with the output data of all the other methods, the places where LR data was highest, the other methods unanimously gave the opposite results.

Gradient Boosting Model has shown promising results. It has shown favorable results when compared with both the k-means labels, RFM from lifetimes package, predicted clv and predicted purchases from lifetimes package. This is an area where we can improve dramatically if we choose the segment map more judiciously and *also perform feature engineering better. This way of predicting holds potential in quickly and easily find out an approximate CLV for customers only to improve upon later.*

Using MBG/NBD + Gamma-Gamma Fitter was also a quick and easy to future predict purchases, clv, aliveness of the customer. The result when cross verifies with GBM results and RFM lifetimes segments were also consistent. Implementing K-Means over this has given us data on microscopic level, telling us labels their characteristics which makes it easier for us to evaluate CLV for customer segmentation.

Simulations using Poisson distribution has further magnified the results which were given by MBG/NBD model and further applying K-Means on it has given us the clearest of Labels. These labels were the easiest to read and interpret. Furthermore, when we analyzed the customers within different labels, their average purchases, total purchase, max and min purchase value, frequency, recency, along with GBM predicted data, this is the most beneficial result in our analysis.

CONCLUSIONS - 2

Considering the summary of conclusion from outcomes of various techniques mentioned above, we can quickly conclude –

Easiest & quickest with least code: by Generating RFM from lifetimes & CLV generation using Python Code

Most difficult with maximum code: by Simulations on ModifiedBetaGeo data using Poisson package + K-Means on the simulation data

Method to avoid: by Generating RFM from Quantiles & CLV generation using Python Code

Promising method but needs manual fitting & trial: CLV sales prediction using Gradient Boosting Machine along with RFM data

Easiest & very reliable way but with some coding: CLV generation using Lifetimes *package: ModifiedBetaGeo fitter + GammaGamm fitter + K-Means*

FUTURE WORK

For the future work and implementation, we consider the inclusion of other models like Hierarchical Fragmentation-Coagulation Processes (HFCP). This is a model which has shown promising results and gained popularity since it can –

1. *automatically determine the groups required to model diverse customer behavior*
2. *capture the changes such as split and merge of customer groups over time, which can easily happen on customer segmentation on non-contractual businesses.*
3. *discover behavior patterns shared among products and identify products with similar or different purchase behavior impacted by promotion, brand choice and change of seasons.*
4. *overcome overfitting problems and outperform previous customer segmentation models on estimating behavior for unseen customers.*

Once the segmentation is done, we can use ***calusalML*** *package to design an uplift solution. This package allows us to make causal inference like ATE, CATE, THE etc*

Also, it would be interesting to further polish the regression method to understand CLV. Once done optimally, it holds the power to quickly create insights on customer segmentation.

END OF REPORT

REFERENCES

1. *Dr. Abishek Samantray. Codes for Linear Regression showcased on Module-10: Hands-On, workbook 2: CLV_Regression.*
2. *Lifetimes package - https://lifetimes.readthedocs.io/en/latest/lifetimes.html*
3. *Dataset - https://archive.ics.uci.edu/dataset/352/online+retail*
4. *Presentation By Dr. Peter fader on CLV - https://www.youtube.com/watch?v=guj2gVEEx4s*
5. *Deriving the Pareto/NBD Model by Dr. Peter Fader, Dr. Bruce Hardie - https://brucehardie.com/notes/009/pareto_nbd_derivations_2005-11-05.pdf*
6. *Presentation on "What's a customer worth?" y Riberto Medri. Case study on Etsy - https://cdn.oreillystatic.com/en/assets/1/event/85/Case%20Study_%20What_s%20a%20Customer%20Worth_%20Presentation.pdf*
7. *R's package "Bye 'Til You Die' – A Walkthrough - https://cran.r-project.org/web/packages/BTYD/vignettes/BTYD-walkthrough.pdf*
8. *"Counting Your Customers" the Easy Way: An alternative to the pareto/NBD Model - https://brucehardie.com/papers/018/fader_et_al_mksc_05.pdf*
9. *Customer-Base Analysis in a Discrete-Time Noncontractual Setting – Dr. peter Fader, Dr, Bruce Hardie - https://www.brucehardie.com/papers/020/*
10. *BYTD package repository - https://github.com/theofilos/BTYD/blob/master/pnbd.R*