Customer Life Time Value Analysis

Submitted by

Priyansh Srivastava[RA2011047010022]
Roshan Upadhyay[RA2011047010015]

Under the Guidance of

DR.N.ARIVAZHAGAN

Assistant Professor, Department of Computational Intelligence

In partial satisfaction of the requirements for the degree of

BACHELORS OF TECHNOLOGY
in
ARTIFICIAL INTELLIGENCE

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

May 2023

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that the Course 18AIE328T-E-Marketing Analytics Project Report titled **"CUSTOMER LIFETIME VALUE ANALYSIS"** is the bonafide work done by **Priyansh Srivastava [RA2011047010022], Roshan Upadhyay [RA2011047010015]** who carried out under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

SIGNATURE
Faculty In-Charge
Dr.N.Arivazhagan
Assistant Professor,
Department of Computational Intelligence,
SRM Institute of Science and Technology,
Kattankulathur Campus, Chennai

HEAD OF THE DEPARTMENT
Dr. R Annie Uthra
Professor and Head ,
Department of Computational Intelligence,
SRM Institute of Science and Technology,
Kattankulathur Campus, Chennai

1. Question

Analyze Customer Lifetime Value in a Customer Description Dataset

2. Rubrics for Case Study Evaluation

(Change as per the question)

Level of Achievement					
		Good (5)	**Average (4)**	**Poor (3)**	**Score**
a	**Able to import it in any platform and showing its features**	Data set imported in any platform and its features displayed	Data set imported and only few features displayed	Data set imported	
b	**Able to identify dependent and independent variables**	Dependent and Independent variables identified correctly and listed with justification	Dependent and Independent variables identified and listed	Dependent and Independent variables identified	
c	**Able to form regression model**	Linear regression model formed with Explanation of each and every variables in the model	Linear regression model formed with Explanation of some variables in the model	Linear regression model formed	
d	**Able to find coefficient and intercept and predict the mileage for the given values**	Coefficient and Intercept values found using any programming language and results shown as per requirement	Coefficient and Intercept values found using any programming language and results shown.	Coefficient and Intercept values found using any programming language.	
				Total	

Maximum Mark: 20

Index

Introduction

Customer lifetime value analysis is a method of quantifying the value of a customer over the course of their relationship with a business. It involves calculating the total revenue a customer is expected to generate for the company, minus the cost of acquiring and servicing that customer. By understanding the lifetime value of their customers, businesses can make informed decisions about how much to spend on marketing and customer retention efforts, and how to allocate resources effectively.

Customer lifetime value analysis has become increasingly important in recent years, as companies have realized the importance of building long-term relationships with customers. With competition in many industries at an all-time high, it's becoming more and more difficult to attract and retain customers. By understanding the lifetime value of their customers, businesses can develop strategies to increase customer loyalty and satisfaction, and ultimately drive revenue growth.

In this article, we will explore the importance of customer lifetime value analysis in today's business environment, and provide practical guidance on how to calculate and use this metric to drive business success. We will examine the key factors that contribute to customer lifetime value, and discuss best practices for maximizing this value through effective customer acquisition and retention strategies.

Abstract

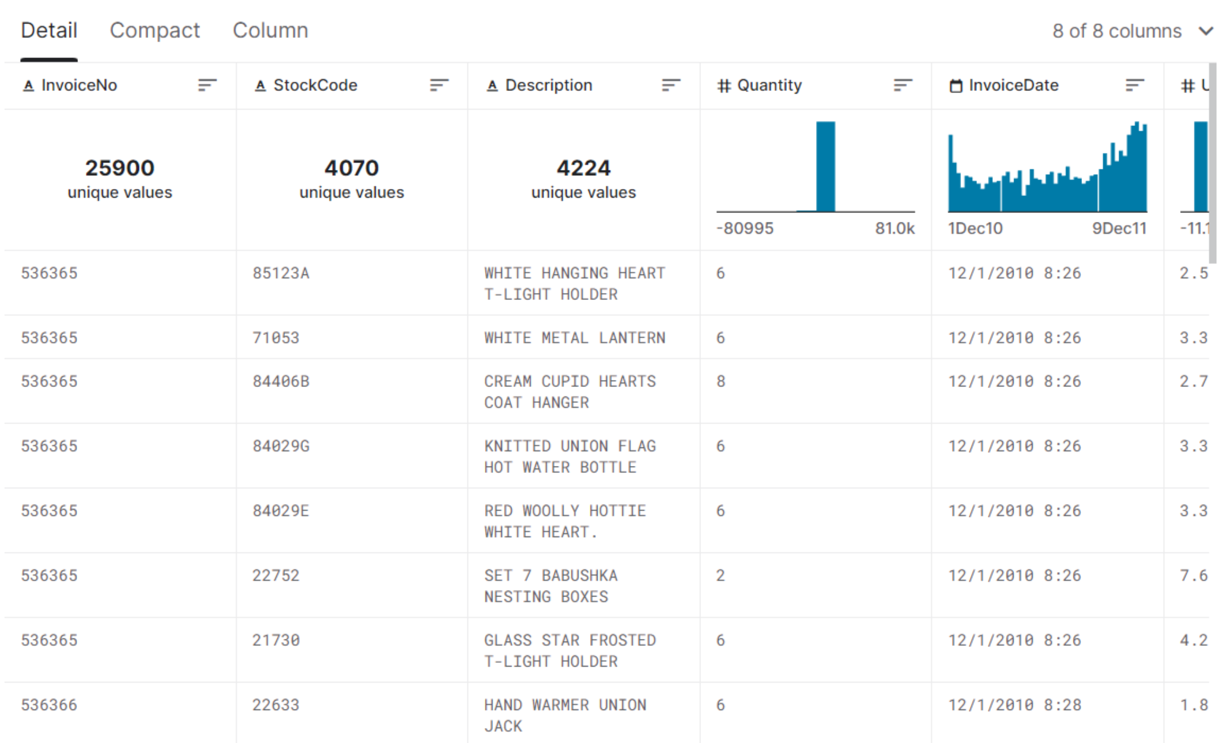
Customer lifetime value analysis is a crucial tool that helps businesses to identify the value of their customers over the entire duration of their relationship with the company. This analysis provides insights into the profitability of customer segments, allowing businesses to tailor their marketing strategies to retain high-value customers and attract new ones. By calculating the customer lifetime value, companies can make informed decisions regarding investments in customer acquisition and retention, and allocate resources more effectively. This abstract provides an overview of the importance and benefits of customer lifetime value analysis for businesses of all sizes and industries.

Dataset Description

The dataset has the following columns:

- InvoiceNo: Unique identifier for each transaction.
- StockCode: Unique identifier for each product.
- Description: Description of each product.
- Quantity: Number of units purchased for each product.
- InvoiceDate: Date and time of each transaction.
- UnitPrice: Price of each unit for each product.
- CustomerID: Unique identifier for each customer.
- Country: Country where the transaction was made.

This dataset is commonly used for retail analytics, as it provides information about the sales of products, customers, and countries. The dataset can be used to analyze sales patterns, identify popular products, understand customer behavior, and optimize pricing and promotions.


Detail Compact Column 8 of 8 columns

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	# U
25900 unique values	4070 unique values	4224 unique values	-80995 81.0k	1Dec10 9Dec11	-11.
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.5
536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.3
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.7
536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.3
536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.3
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.6
536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.2
536366	22633	HAND WARMER UNION JACK	6	12/1/2010 8:28	1.8

Detail Compact **Column** 8 of 8 columns

25900
unique values

Mismatched	0	0%
Missing	0	0%
Unique	25.9k	
Most Common	573585	0%

StockCode

4070
unique values

Valid	542k	100%
Mismatched	0	0%
Missing	0	0%
Unique	4070	
Most Common	85123A	0%

Description

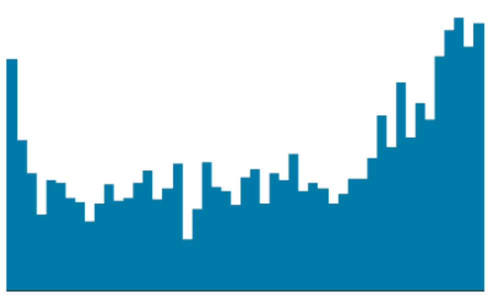
4224
unique values

Valid	540k	100%
Mismatched	0	0%
Missing	1454	0%
Unique	4223	
Most Common	WHITE HAN...	0%

Quantity

Valid	542k	100%
Mismatched	0	0%
Missing	0	0%
Mean	9.55	
Std. Deviation	218	
Quantiles	-80995	Min
	1	25%
	3	50%
	10	75%
	81.0k	Max

InvoiceDate

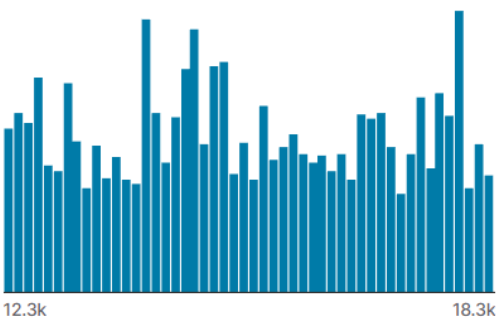


Valid	542k	100%
Mismatched	0	0%
Missing	0	0%
Minimum	1Dec10	
Mean	4Jul11	
Maximum	9Dec11	

UnitPrice

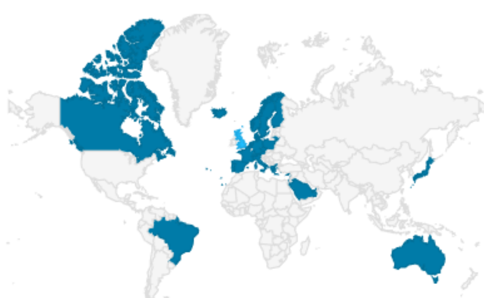
Valid	542k	100%
Mismatched	0	0%
Missing	0	0%
Mean	4.61	
Std. Deviation	96.8	
Quantiles	-11.1k	Min
	1.25	25%
	2.08	50%
	4.13	75%
	39k	Max

CustomerID



Valid	407k	75%
Mismatched	0	0%
Missing	135k	25%
Mean	15.3k	
Std. Deviation	1.71k	
Quantiles	12.3k	Min
	14.0k	25%
	15.2k	50%
	16.8k	75%
	18.3k	Max

Country



Valid	542k	100%
Mismatched	0	0%
Missing	0	0%
Unique	38	
Most Common	United King...	91%

Hardware and Software Requirements

Hardware Requirements:

1. Computer: You will need a computer that meets the minimum requirements for running Python, machine learning libraries, and other software tools required for the analysis.
2. Data Storage: You will need enough data storage capacity to store the customer data you will be analyzing. This can be in the form of local storage, cloud storage, or a combination of both.
3. Internet Connection: A stable and reliable internet connection is required to download and install software packages, access data sources, and perform analysis tasks.

Software Requirements:

1. Python: Python is the primary programming language used for customer lifetime value analysis. You will need to download and install Python on your computer.
2. Machine Learning Libraries: You will need to install machine learning libraries such as Scikit-learn, TensorFlow, and Keras. These libraries are used to build predictive models for customer lifetime value analysis.
3. Data Analytics Tools: You will need to install data analytics tools such as pandas, NumPy, and Matplotlib. These tools are used to manipulate, analyze, and visualize customer data.
4. Database Management Systems: You will need to install a database management system such as MySQL or PostgreSQL to store and manage customer data.
5. Other Software: You may also need other software tools such as Jupyter Notebook, PyCharm, or Visual Studio Code, depending on your preferred development environment and analysis workflow.

Technologies Used

1.)Python

Python is a powerful programming language that has become a popular choice for data science and machine learning projects. Its simplicity, readability, and flexibility make it easy to learn and use, even for those without a background in programming. Python has a vast ecosystem of libraries and tools for data analysis, machine learning, and visualization, including NumPy, Pandas, Scikit-learn, TensorFlow, and Keras. These libraries enable developers to build sophisticated models and extract insights from complex data sets. Python is also highly customizable, allowing developers to create and share their own libraries and tools to address specific needs.

2.)Machine Learning

Machine learning is a subfield of artificial intelligence that involves building predictive models from data. Machine learning algorithms are used to identify patterns in large and complex data sets, and make predictions or decisions based on those patterns. Machine learning has applications in a variety of industries, including finance, healthcare, retail, and marketing. Python is a popular language for machine learning due to its rich ecosystem of libraries and tools, including Scikit-learn, TensorFlow, and Keras. These libraries provide a wide range of algorithms and techniques for machine learning, including classification, regression, clustering, and deep learning.

3.)Data Analysis

Data analysis involves examining and interpreting large and complex data sets to extract insights and make informed decisions. Data analysis can help businesses identify trends, patterns, and relationships in their data, and use that information to improve their operations and profitability. Python is a popular language for data analysis due to its ease of use and powerful libraries, including NumPy and Pandas. These libraries provide powerful tools for manipulating and analyzing data, such as filtering, aggregating, and visualizing data. With Python, businesses can easily extract insights from their data and make informed decisions based on that information.

4.)Jupyter Notebook

Jupyter Notebook is an open-source web application that allows developers to create and share documents that combine live code, visualizations, and explanatory text. Jupyter Notebook supports a variety of programming languages, including Python, R, and Julia. It is a popular tool for data analysis and machine learning projects because it allows developers to write and execute code in an interactive environment, visualize and explore data, and document their work in a single, shareable document. Jupyter Notebook also supports the use of markdown, a simple markup language that allows developers to include formatted text, equations, and images in their documents. With Jupyter Notebook, developers can easily collaborate and share their work with others, making it a powerful tool for data science teams.

5.)Pandas

Pandas is an open-source library for data manipulation and analysis in Python. It provides a powerful data structure called a DataFrame, which is similar to a table in a relational database. Pandas is used extensively in data analysis and machine learning projects due to its ability to handle large data sets and its rich set of tools for data manipulation, filtering, and aggregation. With Pandas, developers can easily load data from various sources, clean and transform data, and analyze data using a variety of statistical techniques.

6.)Matplotlib

Matplotlib is a popular data visualization library in Python. It provides a wide range of customizable plots, including line charts, scatter plots, bar charts, histograms, and more. Matplotlib is used in data analysis and machine learning projects to visualize data and gain insights into patterns and relationships. With Matplotlib, developers can easily create and customize plots, including adding titles, labels, legends, and annotations.

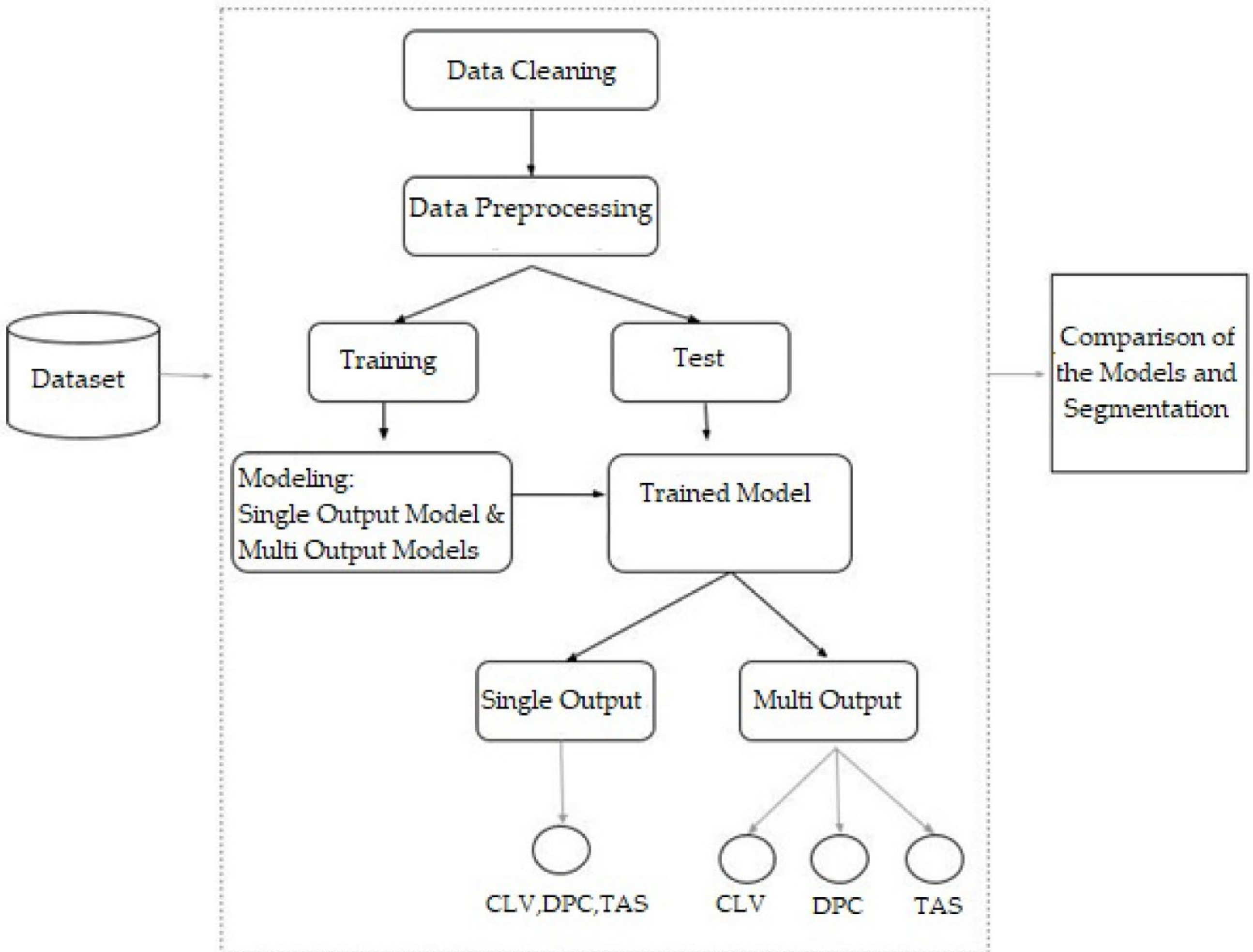
7.)Scikit-Learn

Scikit-learn is a popular machine learning library in Python. It provides a wide range of machine learning algorithms, including classification, regression, clustering, and dimensionality reduction. Scikit-learn is used in data analysis and machine learning projects to build predictive models and extract insights from complex data sets. With Scikit-learn, developers can easily train and test models, evaluate model performance, and optimize model parameters.

8.)Linear Regression

Linear regression is a statistical technique used to model the relationship between a dependent variable and one or more independent variables. In machine learning, linear regression is used for predicting continuous values. The goal of linear regression is to find the line of best fit that describes the relationship between the variables. This line is represented by a linear equation in the form of $y = mx + b$, where y is the dependent variable, x is the independent variable, m is the slope of the line, and b is the y-intercept. Linear regression is a simple and powerful technique for modeling relationships between variables and is widely used in data analysis and machine learning projects.

Workflow



The workflow for customer lifetime value analysis involves several stages, as described below:

1. Data collection: The first step is to gather relevant data about customers, including their purchase history, demographic information, and other relevant variables. This data can be obtained from various sources, such as customer databases, sales records, and marketing campaigns.

2. Data preparation: Once the data is collected, it must be cleaned, organized, and prepared for analysis. This involves removing any duplicates, missing values, or errors, and transforming the data into a suitable format for analysis.
3. Customer segmentation: The next step is to segment customers based on various criteria, such as their purchasing behavior, demographics, or other relevant variables. This allows businesses to identify groups of customers with similar characteristics and tailor their marketing efforts accordingly.
4. Customer lifetime value calculation: Once the customers are segmented, the next step is to calculate their lifetime value. This involves using statistical techniques and models to estimate the total value of a customer to the business over their entire lifetime.
5. Interpretation and action: After the customer lifetime value is calculated, the results must be interpreted and acted upon. This involves identifying high-value customers and developing strategies to retain them, as well as identifying low-value customers and developing strategies to increase their value or minimize their impact on the business.
6. Monitoring and evaluation: The final step is to monitor the effectiveness of the strategies implemented and evaluate the results. This allows businesses to make adjustments and refine their strategies over time, based on actual customer behavior and outcomes.

Overall, the workflow for customer lifetime value analysis involves a combination of data collection, segmentation, modeling, and action, with an emphasis on using data to drive strategic decision-making and improve business outcomes.

Result

```python
#import libraries
from __future__ import division

from datetime import datetime, timedelta,date
import pandas as pd
%matplotlib inline
from sklearn.metrics import classification_report,confusion_matrix
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.cluster import KMeans


import plotly as py
import plotly.offline as pyoff
import plotly.graph_objs as go

import xgboost as xgb
from sklearn.model_selection import KFold, cross_val_score, train_test_split

import xgboost as xgb
```

In [4]:
```
!pip install kaggle
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: kaggle in /usr/local/lib/python3.9/dist-packages (1.5.13)
Requirement already satisfied: certifi in /usr/local/lib/python3.9/dist-packages (from kaggle) (2022.12.7)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from kaggle) (2.27.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.9/dist-packages (from kaggle) (1.26.15)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages (from kaggle) (4.65.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.9/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.9/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.9/dist-packages (from kaggle) (8.0.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.9/dist-packages (from python-slugify->kaggle) (1.
3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->kaggle) (2.
0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->kaggle) (3.4)
```

In [3]:
```python
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
```

In [6]:
```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

In [7]:
```
!kaggle datasets download -d sergeymedvedev/customer_segmentation
```

In [9]:
```
!unzip customer_segmentation.zip
```

```
Archive:  customer_segmentation.zip
replace customer_segmentation.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: customer_segmentation.csv
```

n [12]:
```
tx_data = pd.read_csv('/content/customer_segmentation.csv', encoding='cp1252')
tx_data
```

ut[12]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
...	...	...	...	...	...	...	...	...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	12/9/2011 12:50	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	12/9/2011 12:50	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	12/9/2011 12:50	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	12/9/2011 12:50	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	12/9/2011 12:50	4.95	12680.0	France

541909 rows × 8 columns

Feature Engineering

In [13]:
```
#converting the type of Invoice Date Field from string to datetime.
tx_data['InvoiceDate'] = pd.to_datetime(tx_data['InvoiceDate'])
```

In [14]:
```
tx_data['InvoiceYearMonth'] = tx_data['InvoiceDate'].map(lambda date: 100*date.year + date.month)
```

In [15]:
```
tx_data.describe()
```

Out[15]:

	Quantity	UnitPrice	CustomerID	InvoiceYearMonth
count	541909.000000	541909.000000	406829.000000	541909.000000
mean	9.552250	4.611114	15287.690570	201099.713989
std	218.081158	96.759853	1713.600303	25.788703
min	-80995.000000	-11062.060000	12346.000000	201012.000000
25%	1.000000	1.250000	13953.000000	201103.000000
50%	3.000000	2.080000	15152.000000	201107.000000
75%	10.000000	4.130000	16791.000000	201110.000000
max	80995.000000	38970.000000	18287.000000	201112.000000

```
In [16]: tx_data['Country'].value_counts()

Out[16]: United Kingdom          495478
         Germany                   9495
         France                    8557
         EIRE                      8196
         Spain                     2533
         Netherlands               2371
         Belgium                   2069
         Switzerland               2002
         Portugal                  1519
         Australia                 1259
         Norway                    1086
         Italy                      803
         Channel Islands            758
         Finland                    695
         Cyprus                     622
         Sweden                     462
         Unspecified                446
         Austria                    401
         Denmark                    389
         Japan                      358
         Poland                     341
         Israel                     297
         USA                        291
         Hong Kong                  288
         Singapore                  229
         Iceland                    182
         Canada                     151
         Greece                     146
         Malta                      127
         United Arab Emirates        68
         European Community          61
         RSA                         58
         Lebanon                     45
         Lithuania                   35
         Brazil                      32
         Czech Republic              30
         Bahrain                     19
         Saudi Arabia                10
         Name: Country, dtype: int64
```

In [17]:
```
tx_uk = tx_data.query("Country=='United Kingdom'").reset_index(drop=True)
```

#Recency

In [18]:
```
tx_user = pd.DataFrame(tx_data['CustomerID'].unique())
tx_user.columns = ['CustomerID']
tx_user.head()
```

Out[18]:

	CustomerID
0	17850.0
1	13047.0
2	12583.0
3	13748.0
4	15100.0

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

Out[19]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	InvoiceYearMonth
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	201012
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	201012
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	201012
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	201012
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	201012

In [20]:
```
#max purchase date for each customer and create a dataframe with it
tx_max_purchase = tx_uk.groupby('CustomerID').InvoiceDate.max().reset_index()
tx_max_purchase.columns = ['CustomerID','MaxPurchaseDate']
tx_max_purchase.head()
```

Out[20]:

	CustomerID	MaxPurchaseDate
0	12346.0	2011-01-18 10:17:00
1	12747.0	2011-12-07 14:34:00
2	12748.0	2011-12-09 12:20:00
3	12749.0	2011-12-06 09:56:00
4	12820.0	2011-12-06 15:12:00

In [21]:
```
# Comparing the last transaction of the dataset with last transaction dates of the individual customer IDs.
tx_max_purchase['Recency'] = (tx_max_purchase['MaxPurchaseDate'].max() - tx_max_purchase['MaxPurchaseDate']).dt.days
tx_max_purchase.head()
```

Out[21]:

	CustomerID	MaxPurchaseDate	Recency
0	12346.0	2011-01-18 10:17:00	325
1	12747.0	2011-12-07 14:34:00	1
2	12748.0	2011-12-09 12:20:00	0
3	12749.0	2011-12-06 09:56:00	3
4	12820.0	2011-12-06 15:12:00	2

In [22]:
```
#merge this dataframe to our new user dataframe
tx_user = pd.merge(tx_user, tx_max_purchase[['CustomerID','Recency']], on='CustomerID')
tx_user.head()
```

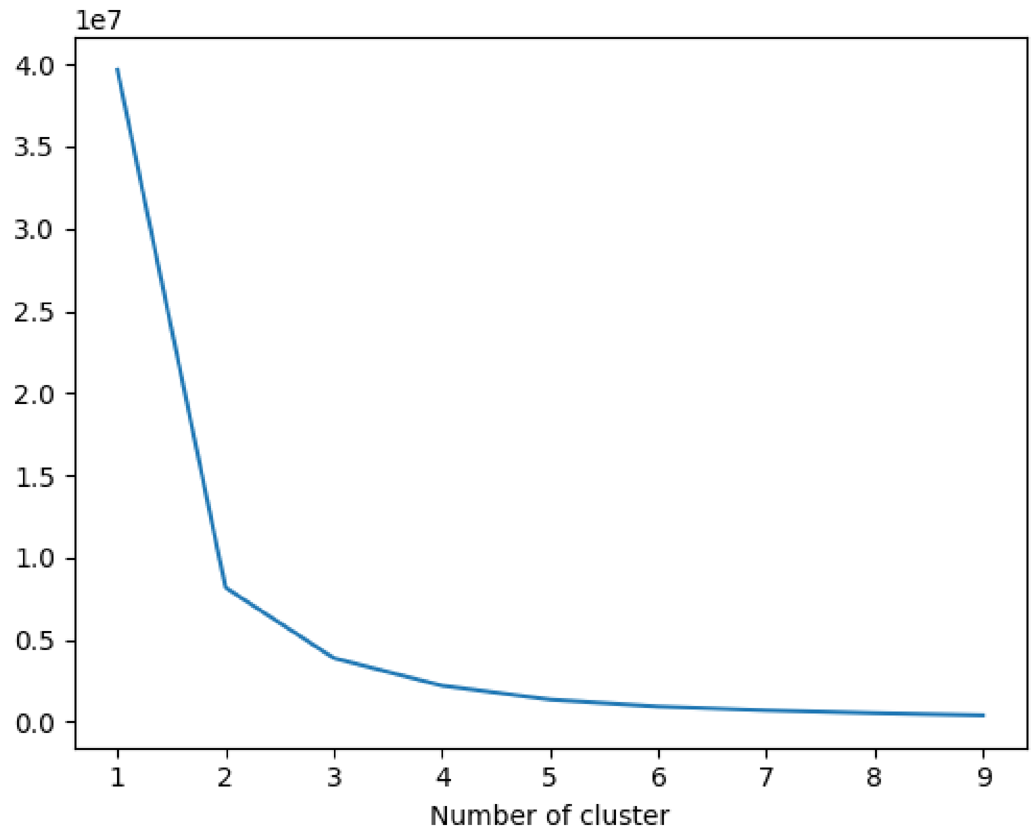
Out[22]:

	CustomerID	Recency
0	17850.0	301
1	13047.0	31
2	13748.0	95
3	15100.0	329
4	15291.0	25

Assigning recency score

In [23]:
```
from sklearn.cluster import KMeans

sse={} # error
tx_recency = tx_user[['Recency']]
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(tx_recency)
    tx_recency["clusters"] = kmeans.labels_  #cluster names corresponding to recency values
    sse[k] = kmeans.inertia_ #sse corresponding to clusters
plt.figure()
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of cluster")
plt.show()
```



In [24]:
```python
kmeans = KMeans(n_clusters=4)
tx_user['RecencyCluster'] = kmeans.fit_predict(tx_user[['Recency']])
```

In [25]:
```python
tx_user.head()
```

Out[25]:

	CustomerID	Recency	RecencyCluster
0	17850.0	301	1
1	13047.0	31	0
2	13748.0	95	3
3	15100.0	329	1
4	15291.0	25	0

In [26]:
```python
tx_user.groupby('RecencyCluster')['Recency'].describe()
```

Out[26]:

RecencyCluster	count	mean	std	min	25%	50%	75%	max
0	1950.0	17.488205	13.237058	0.0	6.00	16.0	28.00	47.0
1	478.0	304.393305	41.183489	245.0	266.25	300.0	336.00	373.0
2	568.0	184.625000	31.753602	132.0	156.75	184.0	211.25	244.0
3	954.0	77.679245	22.850898	48.0	59.00	72.5	93.00	131.0

Ordering clusters

In [27]:
```python
#function for ordering cluster numbers
def order_cluster(cluster_field_name, target_field_name,df,ascending):
    new_cluster_field_name = 'new_' + cluster_field_name
    df_new = df.groupby(cluster_field_name)[target_field_name].mean().reset_index()
    df_new = df_new.sort_values(by=target_field_name,ascending=ascending).reset_index(drop=True)
    df_new['index'] = df_new.index
    df_final = pd.merge(df,df_new[[cluster_field_name,'index']], on=cluster_field_name)
    df_final = df_final.drop([cluster_field_name],axis=1)
    df_final = df_final.rename(columns={"index":cluster_field_name})
    return df_final

tx_user = order_cluster('RecencyCluster', 'Recency',tx_user,False)
```

In [28]:
```python
tx_user.head()
```

Out[28]:

	CustomerID	Recency	RecencyCluster
0	17850.0	301	0
1	15100.0	329	0
2	18074.0	373	0
3	16250.0	260	0
4	13747.0	373	0

```
In [29]: tx_user.groupby('RecencyCluster')['Recency'].describe()
```

Out[29]:

RecencyCluster	count	mean	std	min	25%	50%	75%	max
0	478.0	304.393305	41.183489	245.0	266.25	300.0	336.00	373.0
1	568.0	184.625000	31.753602	132.0	156.75	184.0	211.25	244.0
2	954.0	77.679245	22.850898	48.0	59.00	72.5	93.00	131.0
3	1950.0	17.488205	13.237058	0.0	6.00	16.0	28.00	47.0

#Frequency

```
In [30]: #getting order counts for each user and creating a dataframe with it
tx_frequency = tx_uk.groupby('CustomerID').InvoiceDate.count().reset_index()
tx_frequency.columns = ['CustomerID','Frequency']
```

```
In [31]: tx_frequency.head() #how many orders does a customer have
```

Out[31]:

	CustomerID	Frequency
0	12346.0	2
1	12747.0	103
2	12748.0	4642
3	12749.0	231
4	12820.0	59

```
In [22]: #merge this dataframe to our new user dataframe
tx_user = pd.merge(tx_user, tx_max_purchase[['CustomerID','Recency']], on='CustomerID')
tx_user.head()
```

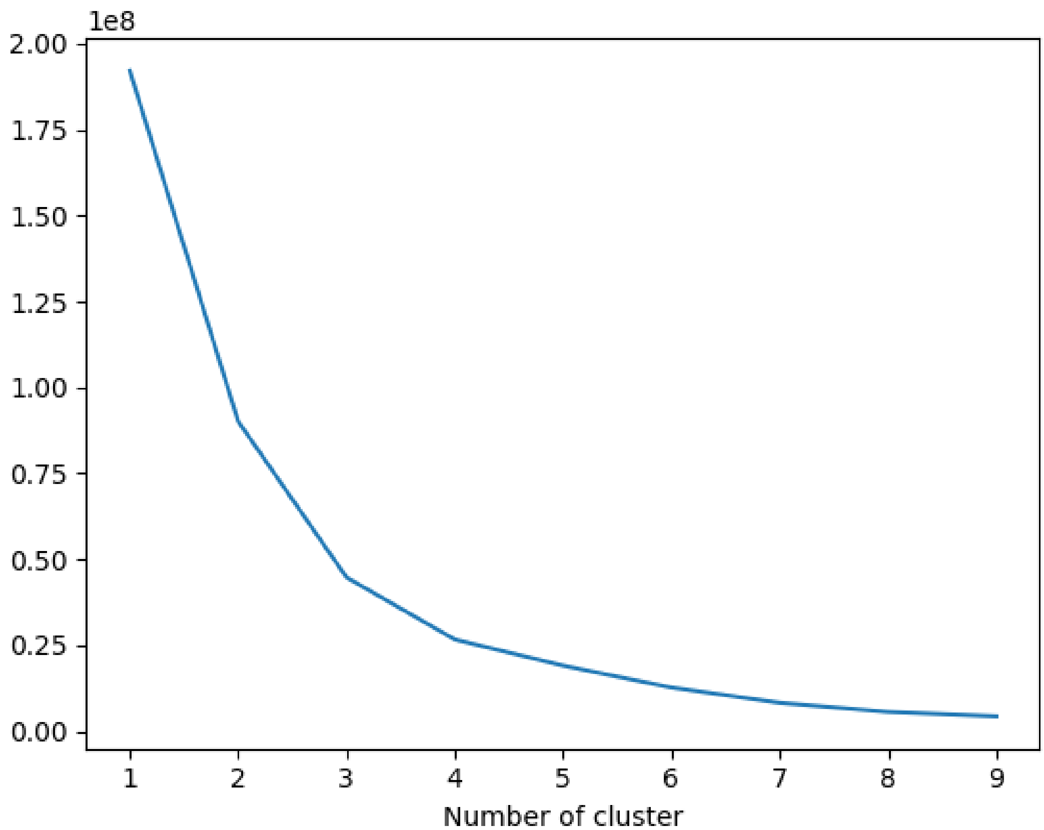
Out[22]:

	CustomerID	Recency
0	17850.0	301
1	13047.0	31
2	13748.0	95
3	15100.0	329
4	15291.0	25

Frequency clusters

In [33]:
```
from sklearn.cluster import KMeans

sse={} # error
tx_recency = tx_user[['Frequency']]
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(tx_recency)
    tx_recency["clusters"] = kmeans.labels_  #cluster names corresponding to recency values
    sse[k] = kmeans.inertia_ #sse corresponding to clusters
plt.figure()
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of cluster")
plt.show()
```



In [34]:
```python
# Applying k-Means
kmeans=KMeans(n_clusters=4)
tx_user['FrequencyCluster']=kmeans.fit_predict(tx_user[['Frequency']])

#order the frequency cluster
tx_user = order_cluster('FrequencyCluster', 'Frequency', tx_user, True )
tx_user.groupby('FrequencyCluster')['Frequency'].describe()
```

Out[34]:

FrequencyCluster	count	mean	std	min	25%	50%	75%	max
0	3496.0	49.525744	44.954212	1.0	15.0	33.0	73.0	190.0
1	429.0	331.221445	133.856510	191.0	228.0	287.0	399.0	803.0
2	22.0	1313.136364	505.934524	872.0	988.5	1140.0	1452.0	2782.0
3	3.0	5917.666667	1805.062418	4642.0	4885.0	5128.0	6555.5	7983.0

Clustering based on Revenue

In [35]:
```python
#calculate revenue for each customer
tx_uk['Revenue'] = tx_uk['UnitPrice'] * tx_uk['Quantity']
tx_revenue = tx_uk.groupby('CustomerID').Revenue.sum().reset_index()
```

In [36]:
```python
tx_revenue.head()
```

Out[36]:

	CustomerID	Revenue
0	12346.0	0.00
1	12747.0	4196.01
2	12748.0	29072.10
3	12749.0	3868.20
4	12820.0	942.34

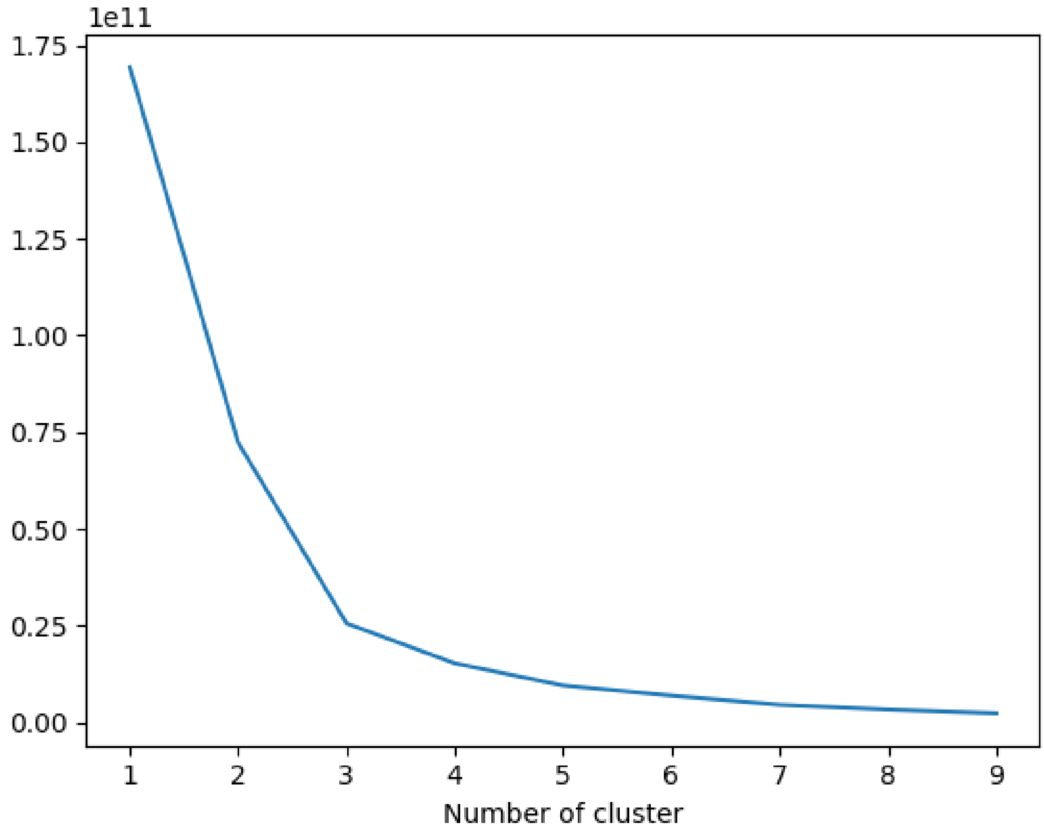
In [37]:
```python
#merge it with our main dataframe
tx_user = pd.merge(tx_user, tx_revenue, on='CustomerID')
tx_user.head()
```

Out[37]:

	CustomerID	Recency	RecencyCluster	Frequency	FrequencyCluster	Revenue
0	17850.0	301	0	312	1	5288.63
1	15808.0	305	0	210	1	3724.77
2	13047.0	31	3	196	1	3079.10
3	14688.0	7	3	359	1	5107.38
4	16029.0	38	3	274	1	50992.61

In [38]:
```
from sklearn.cluster import KMeans

sse={} # error
tx_recency = tx_user[['Revenue']]
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(tx_recency)
    tx_recency["clusters"] = kmeans.labels_  #cluster names corresponding to recency values
    sse[k] = kmeans.inertia_ #sse corresponding to clusters
plt.figure()
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of cluster")
plt.show()
```



Revenue clusters

In [39]:
```
#apply clustering
kmeans = KMeans(n_clusters=4)
tx_user['RevenueCluster'] = kmeans.fit_predict(tx_user[['Revenue']])

#order the cluster numbers
tx_user = order_cluster('RevenueCluster', 'Revenue',tx_user,True)

#show details of the dataframe
tx_user.groupby('RevenueCluster')['Revenue'].describe()
```

Out[39]:

RevenueCluster	count	mean	std	min	25%	50%	75%	max
0	3687.0	907.254414	921.910820	-4287.63	263.115	572.56	1258.220	4314.72
1	234.0	7760.699530	3637.173671	4330.67	5161.485	6549.38	9142.305	21535.90
2	27.0	43070.445185	15939.249588	25748.35	28865.490	36351.42	53489.790	88125.38
3	2.0	221960.330000	48759.481478	187482.17	204721.250	221960.33	239199.410	256438.49

RFM Score

In [40]:
```
#calculate overall score and use mean() to see details
tx_user['OverallScore'] = tx_user['RecencyCluster'] + tx_user['FrequencyCluster'] + tx_user['RevenueCluster']
tx_user.groupby('OverallScore')['Recency','Frequency','Revenue'].mean()
```

Out[40]:

OverallScore	Recency	Frequency	Revenue
0	304.584388	21.995781	303.339705
1	185.362989	32.596085	498.087546
2	78.991304	46.963043	868.082991
3	20.689610	68.419590	1091.416414
4	14.892617	271.755034	3607.097114
5	9.662162	373.290541	9136.946014
6	7.740741	876.037037	22777.914815
7	1.857143	1272.714286	103954.025714
8	1.333333	5917.666667	42177.930000

In [42]:
```
tx_user
```

Out[42]:

	CustomerID	Recency	RecencyCluster	Frequency	FrequencyCluster	Revenue	RevenueCluster	OverallScore	Segment
0	17850.0	301	0	312	1	5288.63	1	2	Low-Value
1	14688.0	7	3	359	1	5107.38	1	5	High-Value
2	13767.0	1	3	399	1	16945.71	1	5	High-Value
3	15513.0	30	3	314	1	14520.08	1	5	High-Value
4	14849.0	21	3	392	1	7904.28	1	5	High-Value
...	...	...	...	...	...	...	...	...	...
3945	12748.0	0	3	4642	3	29072.10	2	8	High-Value
3946	17841.0	1	3	7983	3	40340.78	2	8	High-Value
3947	14096.0	3	3	5128	3	57120.91	2	8	High-Value
3948	17450.0	7	3	351	1	187482.17	3	7	High-Value
3949	18102.0	0	3	433	1	256438.49	3	7	High-Value

3950 rows × 9 columns

Customer Lifetime Value for six months

In [43]:
```
tx_uk.head()
```

Out[43]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	InvoiceYearMonth	Revenue
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	201012	15.30
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	201012	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	201012	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	201012	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	201012	20.34

In [44]:
```
tx_uk['InvoiceDate'].describe()
```

Out[44]:
```
count                  495478
unique                  21220
top       2011-10-31 14:41:00
freq                     1114
first     2010-12-01 08:26:00
last      2011-12-09 12:49:00
Name: InvoiceDate, dtype: object
```

In [50]:
```
from datetime import datetime

tx_3m = tx_uk[(tx_uk.InvoiceDate < datetime(2011,6,1)) & (tx_uk.InvoiceDate >= datetime(2011,3,1))].reset_index(drop=True) #3 mor
tx_6m = tx_uk[(tx_uk.InvoiceDate >= datetime(2011,6,1)) & (tx_uk.InvoiceDate < datetime(2011,12,1))].reset_index(drop=True) # 6 m
```

In [51]:
```
#calculate revenue and create a new dataframe for it
tx_6m['Revenue'] = tx_6m['UnitPrice'] * tx_6m['Quantity']
tx_user_6m = tx_6m.groupby('CustomerID')['Revenue'].sum().reset_index()
tx_user_6m.columns = ['CustomerID','m6_Revenue']
```

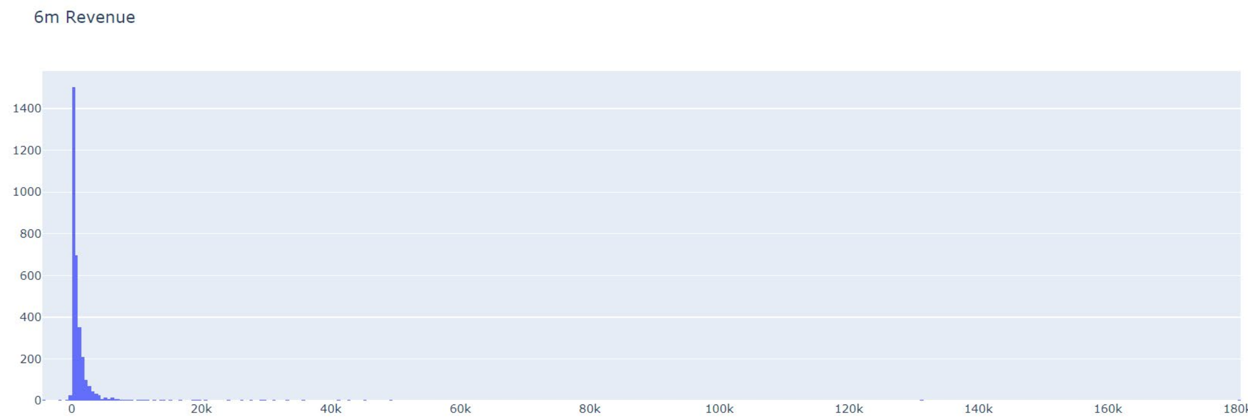
In [52]:
```
tx_user_6m.head()
```

Out[52]:

	CustomerID	m6_Revenue
0	12747.0	1666.11
1	12748.0	18679.01
2	12749.0	2323.04
3	12820.0	561.53
4	12822.0	918.98

```
In [53]: #plot LTV histogram
plot_data = [
    go.Histogram(
        x=tx_user_6m['m6_Revenue']
    )
]

plot_layout = go.Layout(
        title='6m Revenue'
    )
fig = go.Figure(data=plot_data, layout=plot_layout)
pyoff.iplot(fig)
```




```
In [54]: tx_user.head()
```

Out[54]:

	CustomerID	Recency	RecencyCluster	Frequency	FrequencyCluster	Revenue	RevenueCluster	OverallScore	Segment
0	17850.0	301	0	312	1	5288.63	1	2	Low-Value
1	14688.0	7	3	359	1	5107.38	1	5	High-Value
2	13767.0	1	3	399	1	16945.71	1	5	High-Value
3	15513.0	30	3	314	1	14520.08	1	5	High-Value
4	14849.0	21	3	392	1	7904.28	1	5	High-Value

```
In [55]: tx_uk.head()
```

Out[55]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	InvoiceYearMonth	Revenue
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	201012	15.30
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	201012	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	201012	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	201012	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	201012	20.34

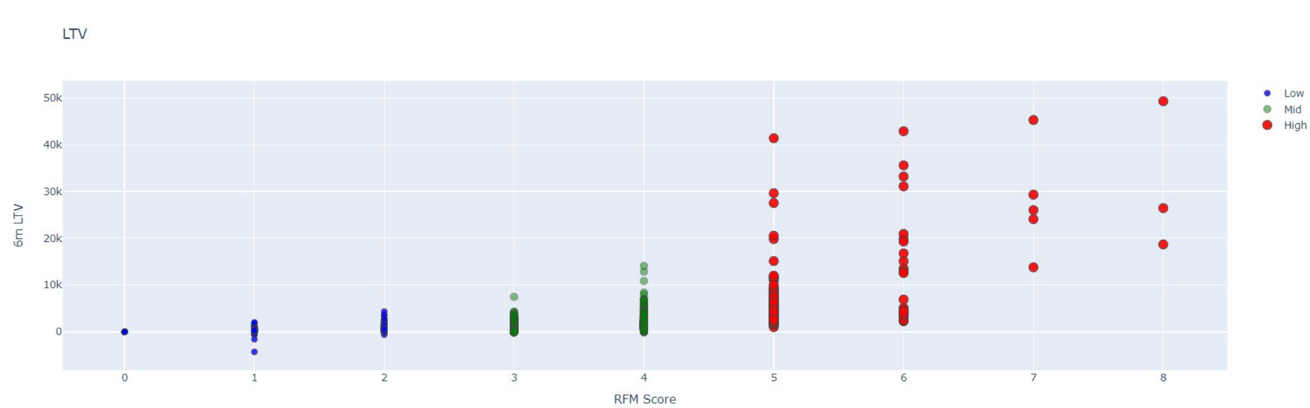
```
In [56]: tx_merge = pd.merge(tx_user, tx_user_6m, on='CustomerID', how='left') #Only people who are in the timeline of tx_user_6m
```

In [57]:
```
tx_merge = tx_merge.fillna(0)
```

In [58]:
```
tx_graph = tx_merge.query("m6_Revenue < 50000") #because max values are ending at 50,000 as seen in graph above

plot_data = [
    go.Scatter(
        x=tx_graph.query("Segment == 'Low-Value'")['OverallScore'],
        y=tx_graph.query("Segment == 'Low-Value'")['m6_Revenue'],
        mode='markers',
        name='Low',
        marker= dict(size= 7,
            line= dict(width=1),
            color= 'blue',
            opacity= 0.8
           )
    ),
        go.Scatter(
        x=tx_graph.query("Segment == 'Mid-Value'")['OverallScore'],
        y=tx_graph.query("Segment == 'Mid-Value'")['m6_Revenue'],
        mode='markers',
        name='Mid',
        marker= dict(size= 9,
            line= dict(width=1),
            color= 'green',
            opacity= 0.5
           )
    ),
        go.Scatter(
        x=tx_graph.query("Segment == 'High-Value'")['OverallScore'],
        y=tx_graph.query("Segment == 'High-Value'")['m6_Revenue'],
        mode='markers',
        name='High',
        marker= dict(size= 11,
            line= dict(width=1),
            color= 'red',
            opacity= 0.9
           )
    ),
]

plot_layout = go.Layout(
        yaxis= {'title': "6m LTV"},
        xaxis= {'title': "RFM Score"},
        title='LTV'
    )
fig = go.Figure(data=plot_data, layout=plot_layout)
pyoff.iplot(fig)
```



In [59]:
```
#remove outliers
tx_merge = tx_merge[tx_merge['m6_Revenue']<tx_merge['m6_Revenue'].quantile(0.99)]
```

In [60]:
```
tx_merge.head()
```

Out[60]:

	CustomerID	Recency	RecencyCluster	Frequency	FrequencyCluster	Revenue	RevenueCluster	OverallScore	Segment	m6_Revenue
0	17850.0	301	0	312	1	5288.63	1	2	Low-Value	0.00
1	14688.0	7	3	359	1	5107.38	1	5	High-Value	1702.06
4	14849.0	21	3	392	1	7904.28	1	5	High-Value	5498.07
6	13468.0	1	3	306	1	5656.75	1	5	High-Value	1813.09
7	17690.0	29	3	258	1	4748.45	1	5	High-Value	2616.15

In [61]:
```
#creating 3 clusters
kmeans = KMeans(n_clusters=3)
tx_merge['LTVCluster'] = kmeans.fit_predict(tx_merge[['m6_Revenue']])

tx_merge.head()
```

Out[61]:

	CustomerID	Recency	RecencyCluster	Frequency	FrequencyCluster	Revenue	RevenueCluster	OverallScore	Segment	m6_Revenue	LTVCluster
0	17850.0	301	0	312	1	5288.63	1	2	Low-Value	0.00	0
1	14688.0	7	3	359	1	5107.38	1	5	High-Value	1702.06	2
4	14849.0	21	3	392	1	7904.28	1	5	High-Value	5498.07	1
6	13468.0	1	3	306	1	5656.75	1	5	High-Value	1813.09	2
7	17690.0	29	3	258	1	4748.45	1	5	High-Value	2616.15	2

In [62]:
```
#order cluster number based on LTV
tx_merge = order_cluster('LTVCluster', 'm6_Revenue',tx_merge,True)

#creatinga new cluster dataframe
tx_cluster = tx_merge.copy()

#see details of the clusters
tx_cluster.groupby('LTVCluster')['m6_Revenue'].describe()
```

Out[62]:

	count	mean	std	min	25%	50%	75%	max
LTVCluster								
0	2955.0	276.176333	280.962101	-4287.63	0.0000	228.910	449.425	937.60
1	799.0	1605.393279	550.258391	938.20	1143.6650	1479.840	1933.495	3113.70
2	156.0	4645.661795	1345.674897	3129.27	3537.7325	4256.115	5497.980	8432.68

In [63]:
```
tx_cluster.head()
```

Out[63]:

	CustomerID	Recency	RecencyCluster	Frequency	FrequencyCluster	Revenue	RevenueCluster	OverallScore	Segment	m6_Revenue	LTVCluster
0	17850.0	301	0	312	1	5288.63	1	2	Low-Value	0.0	0
1	13093.0	266	0	170	0	7741.47	1	1	Low-Value	0.0	0
2	15032.0	255	0	55	0	4464.10	1	1	Low-Value	0.0	0
3	16000.0	2	3	9	0	12393.70	1	4	Mid-Value	0.0	0
4	15749.0	234	1	15	0	21535.90	1	2	Low-Value	0.0	0

#Feature Engineering

In [64]:
```
#convert categorical columns to numerical
tx_class = pd.get_dummies(tx_cluster) #There is only one categorical variable segment
tx_class.head()
```

Out[64]:

	CustomerID	Recency	RecencyCluster	Frequency	FrequencyCluster	Revenue	RevenueCluster	OverallScore	m6_Revenue	LTVCluster	Segment_High-Value	Seg
0	17850.0	301	0	312	1	5288.63	1	2	0.0	0	0	
1	13093.0	266	0	170	0	7741.47	1	1	0.0	0	0	
2	15032.0	255	0	55	0	4464.10	1	1	0.0	0	0	
3	16000.0	2	3	9	0	12393.70	1	4	0.0	0	0	
4	15749.0	234	1	15	0	21535.90	1	2	0.0	0	0	

In [65]:
```python
#calculate and show correlations
corr_matrix = tx_class.corr()
corr_matrix['LTVCluster'].sort_values(ascending=False)
```

Out[65]:
```
LTVCluster           1.000000
m6_Revenue           0.877508
Revenue              0.774841
RevenueCluster       0.604372
Frequency            0.569080
OverallScore         0.542623
FrequencyCluster     0.515422
Segment_High-Value   0.495977
RecencyCluster       0.359110
Segment_Mid-Value    0.190712
CustomerID          -0.030966
Recency             -0.351114
Segment_Low-Value   -0.379459
Name: LTVCluster, dtype: float64
```

In [66]:
```python
#create X and y, X will be feature set and y is the label - LTV
X = tx_class.drop(['LTVCluster','m6_Revenue'],axis=1)
y = tx_class['LTVCluster']

#split training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.05, random_state=56)
```

XGBOOST Model for Customer Lifetime Value Prediction

In [67]:
```python
#XGBoost Multiclassification Model
ltv_xgb_model = xgb.XGBClassifier(max_depth=5, learning_rate=0.1,n_jobs=-1).fit(X_train, y_train)

print('Accuracy of XGB classifier on training set: {:.2f}'
       .format(ltv_xgb_model.score(X_train, y_train)))
print('Accuracy of XGB classifier on test set: {:.2f}'
       .format(ltv_xgb_model.score(X_test[X_train.columns], y_test)))

y_pred = ltv_xgb_model.predict(X_test)
```

```
Accuracy of XGB classifier on training set: 0.95
Accuracy of XGB classifier on test set: 0.91
```

In [68]:
```python
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.93      0.95       144
           1       0.76      0.89      0.82        44
           2       0.86      0.75      0.80         8

    accuracy                           0.91       196
   macro avg       0.86      0.86      0.86       196
weighted avg       0.92      0.91      0.92       196
```

Conclusion

In conclusion, customer lifetime value analysis is a crucial tool for businesses looking to understand the long-term value of their customers and develop strategies to maximize their revenue and profitability. By calculating customer lifetime value, businesses can identify their most valuable customers, develop personalized marketing strategies to retain them, and optimize their acquisition and retention efforts to drive growth and increase profitability. Through the use of advanced data analysis and machine learning techniques, businesses can gain deeper insights into customer behavior and preferences, allowing them to tailor their marketing efforts and provide a more personalized customer experience. Ultimately, customer lifetime value analysis is a powerful tool for businesses looking to stay competitive and succeed in today's data-driven marketplace.

Teamwork Plan & Execution

Team Member Name	Work Planned	Work Completed	Remarks
			-
			-
			-

Team Member 1 Sign **Team Member 2 Sign** **Team Member 3 Sign** **Team Member 4 Sign**