**A CASE STUDY REPORT ON ONLINE RETAIL DATA SET**



**Data Mining (CIS4015-N-FJI-2020)**

**A Mahikanth Reddy (W9332307)**

**Kalyana Ramakrishna Karri(W9316286 )**

**Manjunath Chandrakanth Yalala (W9290876)**

**Preetham Chikkamagalur Shylendhra (W9300889 )**

Table of Contents

1. **Abstract2**
2. **Introduction2**
3. **Data pre-processing3**
4. **Data Analysis3**

4.1 Addition of Total column4

4.2 Overperforming countries4

4.3 Underperforming countries4

4.4 Top customers order wise5

4.5 Top customers spending wise5

4.6 Most sold products by quantity6

4.7 Monthly sales analysis7

4.8 Daily sales analysis7

4.9 Hourly sales analysis8

4.10 Sales of a product over the year9

4.11 Some other basic statistics11

4.12 Analysis of an individual country10

1. **Apriori algorithm12**

5.1 Step by step evaluation of the programme12

**6. RFM k-means clustering model15**

6.1 Libraries, file read, data frame16

6.2 Dataset cleansing16

6.3 Data preparation17

6.4 k-means implementation20

6.5 Rescaling the attributes23

6.6 Building the k-means clustering model23

**7. Results, Discussions and Future work**28

**8. Conclusion**32

9. **Individual Peer Marks**…………………………………………………………………………….32

**1. Abstract**

Online retailers need to analyze the customer requirements and their purchase patterns to understand them better and keep them attracted to shop on their website often. Local stores have the opportunity to connect with their customers utilizing direct interaction and respond to their needs spontaneously. However, online retailers do not have that liberty. On the brighter side, online retailers have got the customer metadata, their buying habits, and the preferences of their products. In this report, a case study has been performed using data mining techniques to extract business intelligence. The main purpose of this report is to provide online retailers with an opportunity to better understand their customers. Data analysis technique; apriori has been performed on the dataset to find the buying patterns of customers and also implemented the RFM model and K-means Clustering to divide the customers into meaningful ranked categories based on their importance to the business.

**2. Introduction**

In this particular case study, we will be analyzing an online retail data set which sells gift products, it is a UK based company but has a franchise in over 38 countries. We found the data set in UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets/Online+Retail>), and it holds the records of purchases and sales of products over a year that is from 2010-2011, we will be analyzing the dataset with two possible approaches and try to answer the descriptive and predictive question for this case study. Apriori and K-means approaches are suitable for this particular data set because Apriori helps us in finding a pattern in the purchase behaviour of the customers, this pattern tells us about the different items which are brought together often. Based on this analysis the retailers can understand how to manage their stocks, organize things and plan marketing strategies. Along with that, the retailer must know his most valuable customers or the different types of customers he has. This requirement is sufficed using RFM model and clustering using K-means clustering. Here the attributes of customers; Recency, Frequency and Monetary are taken from their buying pattern, these values are used in K-means to group customers.

The online retail data set contains over 500,000 records with different attributes such as customer id, name of the item, date and time of item purchase, cost of each unit, and invoice number of the purchase. These attributes give us different ways to explore the data, but the objective of this case study is to find out some insights which will be helpful for the company. First, the descriptive analysis of the data will be performed to find out what has happened in the past, and then based on the insights that are gathered, we will be proceeding forward to make predictions. First, we answer a question which is very common in most of the businesses, What is a customer interested in purchasing? because customer preferences change accordingly over a period of time but we can draw a pattern out of his purchase behaviour to find out what he might be interested in purchasing.

Most of the companies which are into online retail business should keep attracting customers because there are a lot of options out there and the customer has the freedom to choose which particular product to buy from which website, so it’s the company’s main objective to find out what a customer would prefer to buy the next time he purchases. The happier the customer, the more the sales of the company. Also, companies can make vital decisions such as increasing or decreasing the price of the product, running offers on products that are preferred by customers by giving special discounts and recommending products according to the customer preference.

This whole process would fit if we are analyzing a small group of customer data, but what if the company is big and it has its ventures over 38 countries? It would be a difficult task to analyze that big data. But with the help of data mining techniques, we will be able to perform this task. The data-mining techniques will be discussed further.

**3. Data pre-processing**

To perform the apriori and k-means clustering analysis, the original dataset needs to be processed and cleaned. Below are the steps involved in the data preparation:

1. Re-order the columns as per convenience. In our case, eight variables are chosen: ‘StockCode’, ‘Description’, ‘Quantity’, ‘InvoiceDate’, ‘UnitPrice’, ‘CustomerID’, ‘Country’, ‘InvoiceNo’
2. Find empty values in the rows and drop all the null values
3. CustomerID has floating values, convert them into a string
4. Filter out all the rows which have empty invoice numbers
5. Delete all the transactions where a transaction has been done using credit card (in our case, invoice number starts with C for credit)
6. Formatting the InvoiceDate column into separate entities such as year, month, day, hour and add them as separate columns

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Datatype** | **Description** |
| StockCode | object | A five-digit unique number assigned to each product |
| Description | object | Product name |
| Quantity | integer | Number of items per transaction |
| InvoiceDate | date | Date and time when the transaction was made |
| UnitPrice | float | Price of a single product |
| CustomerID | float | A unique id is given to each customer |
| Country | object | The country in which the order was made |
| InvoiceNo | object | A six-digit unique number attached to each transaction |
| Year | integer | Year represented in four-digit (obtained from the invoice date) |
| Month | Integer | Month represented in two-digit (obtained from the invoice date) |
| Day | Integer | Day represented in single-digit (obtained from the invoice date) |
| hour | Integer | Hour represented in single-digit (obtained from the invoice date) |

*Table 3.1: Dataset description*

The processed dataset for analysis is obtained and the dataset is forwarded to the next stage for data analysis.

**4. Data analysis**

Once data is cleaned and brought to the final version, there is some basic statistical analysis done on the dataset. Also, the dataset is formatted according to the requirement. Steps and code involved are documented below:

import pandas as pd

from mlxtend.frequent\_patterns import apriori

from mlxtend.frequent\_patterns import association\_rules

from matplotlib import pyplot as plt

df\_real = pd.read\_excel('C:\\OnlineRetail.xlsx',encoding = 'ISO-8859-1')

Libraries required for the analysis have been imported as written in the code above.

4.1 Addition of Total column

This column tells us about the money spent on that particular product in a transaction.

Total = df['Quantity'] \* df['UnitPrice']

df.insert(loc=7,column='Total',value=Total)

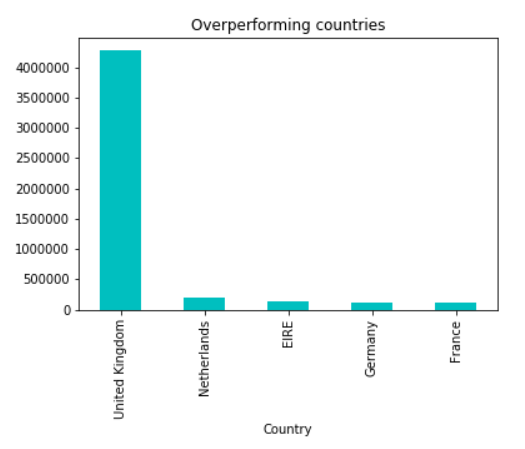
4.2 Overperforming countries

In the above code, a new column is added by multiplying the quantity and the unit price of the product. The new entity is labelled as ‘Total’ and added as a separate column. The objective here is to find the monetary value transaction done by a customer on a specific product.

salesort = df.groupby(['Country'])['Quantity'].sum().sort\_values(ascending = False)

salesort[:5].plot(kind = 'bar', title = 'Overperforming countries', color='c')

The first statistical analysis of dataset is to plot the top five countries as per the number of items purchased.

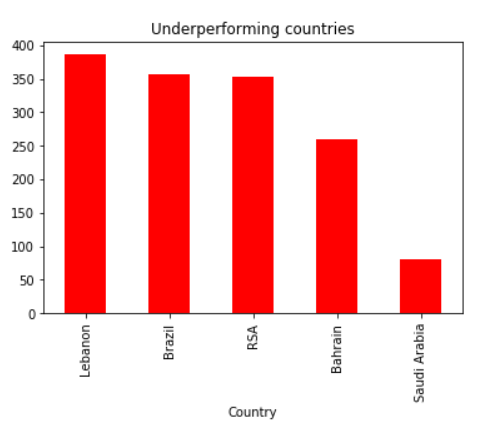


*Figure 4.1: Overperforming countries*

4.3 Underperforming countries

The bottom five countries as per the number of items purchased.

salesort[32:].plot(kind = 'bar', title = 'Underperforming countries', color='red')

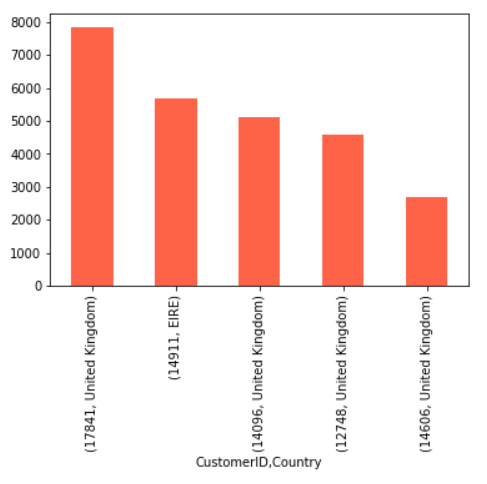


*Figure 4.2: Underperforming countries*

4.4 Top customers order wise

In this section, the top five customers from the entire dataset have been found and plotted. Grouped by Customer ID and Country.

df.groupby(['CustomerID','Country'])['InvoiceNo'].count().sort\_values(ascending = False).head().plot(kind = 'bar', color='tomato')

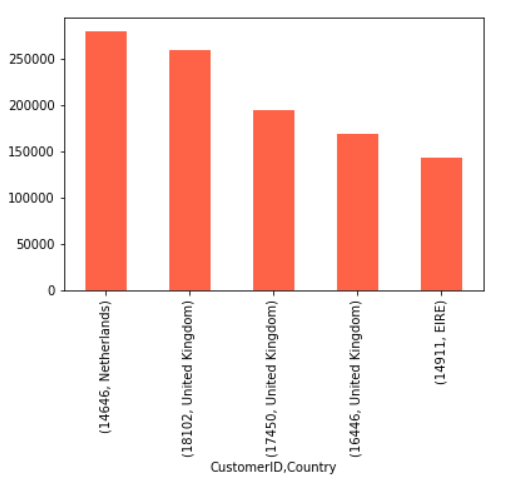


*Figure 4.3: Top five customers as per orders*

4.5 Top customers spending-wise

Top five customers who have spent the most amount to purchase products

df.groupby(['CustomerID','Country'])['Total'].sum().sort\_values(ascending = False).head().plot(kind = 'bar', color='tomato')



*Figure 4.4: Top five customers based on money spent*

4.6 Most sold products by quantity

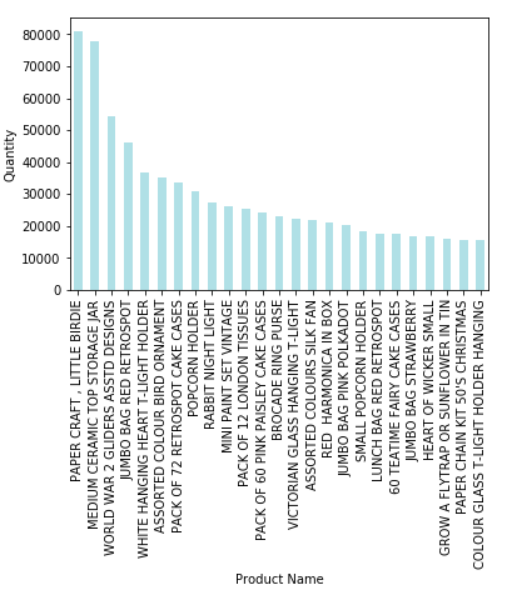
Below is the code to extract the top 25 products ordered by quantity and graph to be plotted.

most\_sold = df.groupby(['Description'])['Quantity'].sum().sort\_values(ascending = False)

top25 = most\_sold[0:25].plot(kind = 'bar', color='powderblue')

top25.set\_xlabel("Product Name")

top25.set\_ylabel("Quantity")



*Figure 4.5: Top 25 products sold*

4.7 Monthly sales analysis

The below code is written to analyze the monthly sales of the retail store over the whole dataset.

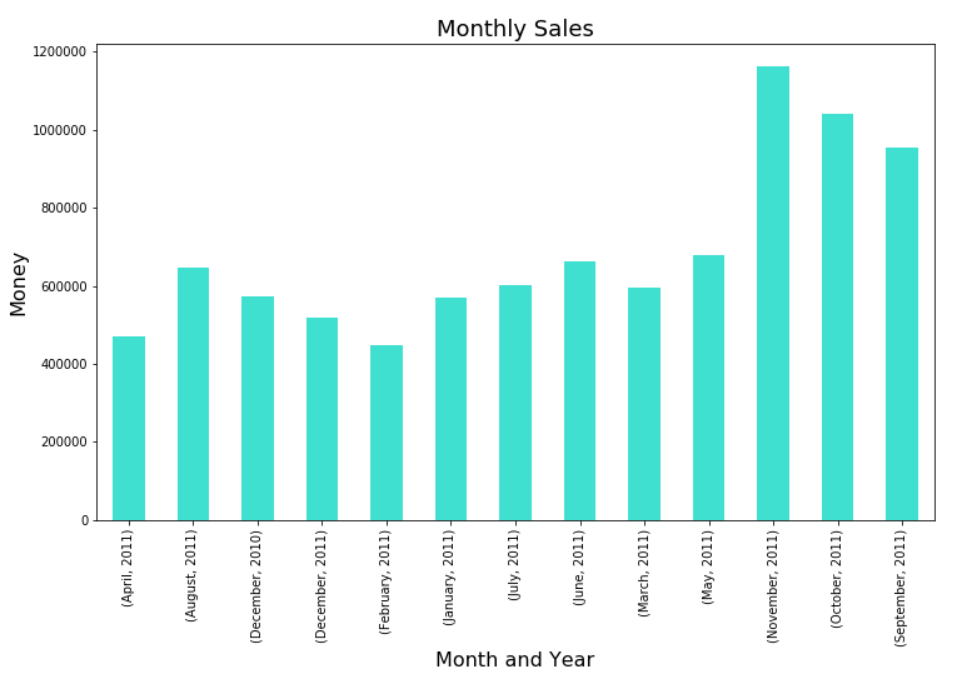
df['monthname'] = df['InvoiceDate'].dt.month\_name()

monthly = df.groupby(['monthname','year'])['Total'].sum().plot(kind = 'bar', color = 'turquoise', figsize=(12,7))

monthly.set\_xlabel("Month and Year",size=16)

monthly.set\_ylabel("Money",size=16)

monthly.set\_title("Monthly Sales",size=18



*Figure 4.6: Monthly sales*

4.8 Daily sales analysis

The below code is written to analyze the sales according to the day of the week. This is implemented in the retail store over the whole dataset.

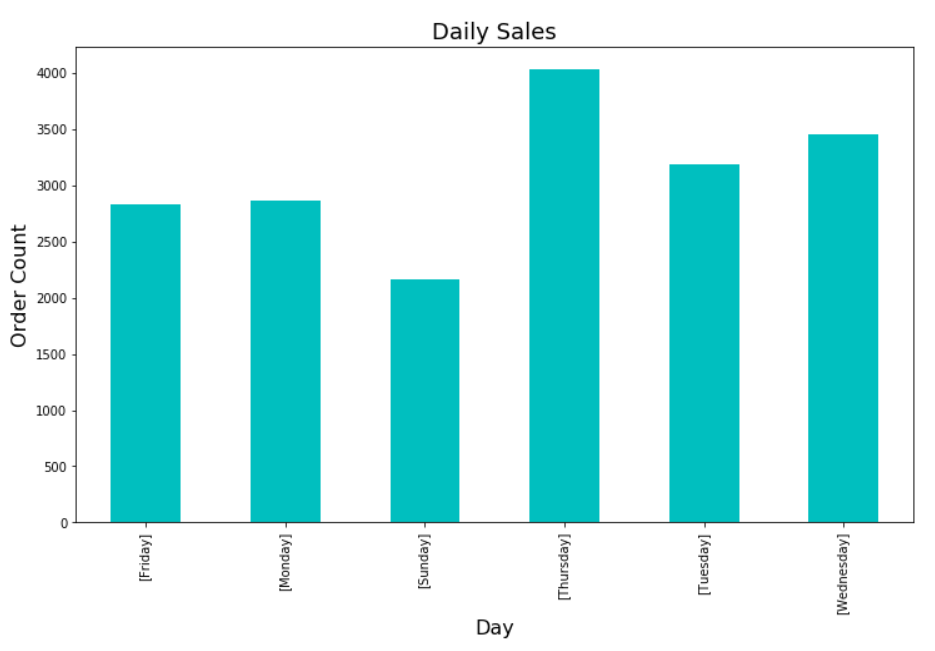
df['dayname'] = df['InvoiceDate'].dt.day\_name()

daily = df.groupby('InvoiceNo')['dayname'].unique().value\_counts().sort\_index().plot(kind = 'bar',color = 'c', figsize=(12,7))

daily.set\_xlabel("Day",size=16)

daily.set\_ylabel("Order Count",size=16)

daily.set\_title("Daily Sales",size=18)



*Figure 4.7: Day of the week sales*

4.9 Hourly sales analysis

The below code is written to analyze the sales according to the hour of the day. This is implemented on the complete dataset.

hourly = df.groupby('InvoiceNo')['hour'].unique().value\_counts().iloc[:-1].sort\_index().plot(kind = 'bar',color = 'aqua', figsize=(12,7))

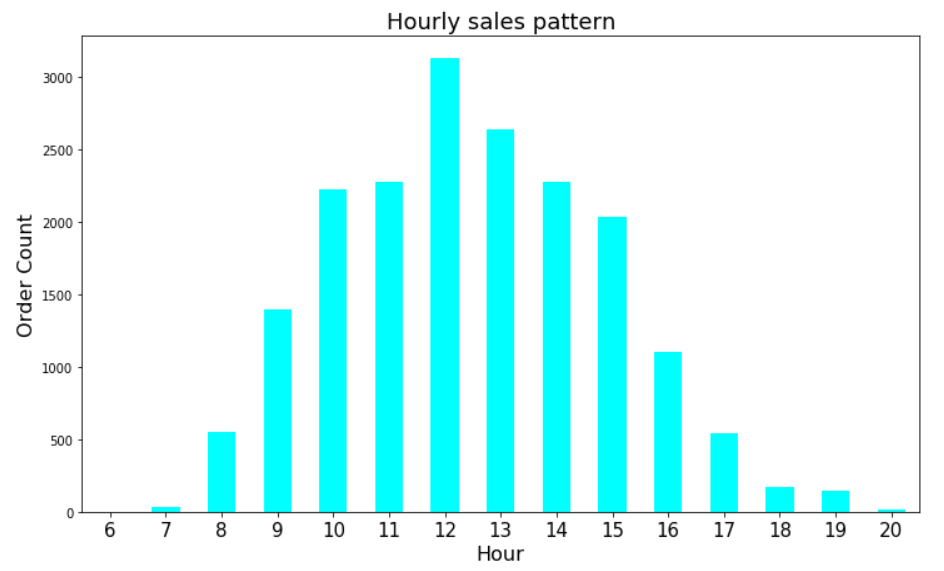
hourly.set\_xlabel("Hour",size=16)

hourly.set\_ylabel("Order Count",size=16)

hourly.set\_title("Hourly sales pattern",size=18)

hourly.set\_xticklabels(range(6,21), rotation='horizontal', fontsize=15)

plt.show()

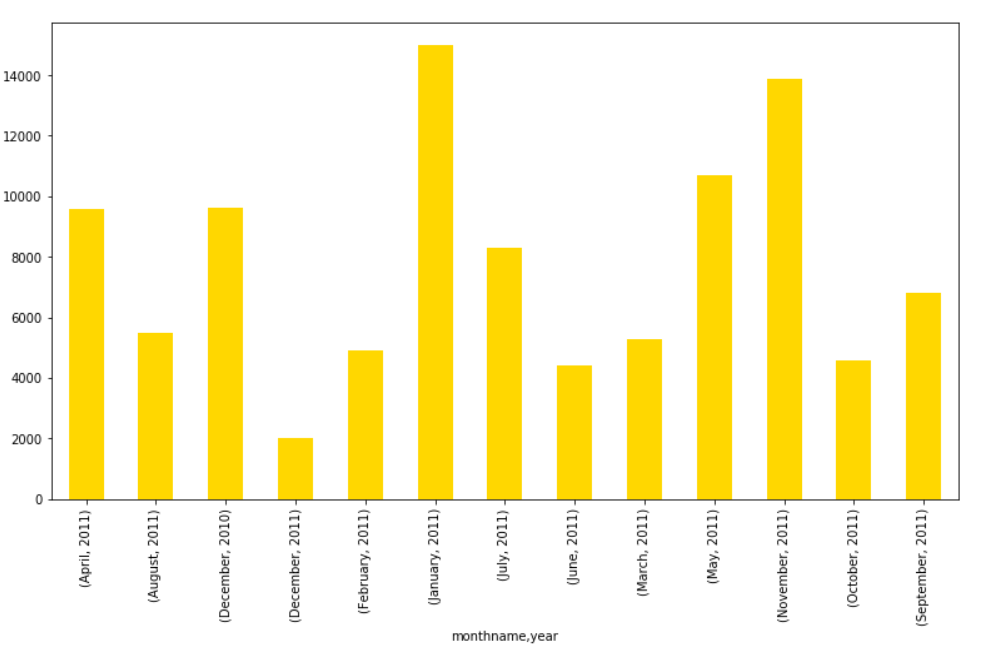


*Figure 4.8: Hour in the day sales*

4.10 Sales of a product over the year

By using the below code, the sale of a specific product could be plotted over the year.

df[df['Description'] == "WHITE HANGING HEART T-LIGHT HOLDER"].groupby(['monthname','year'])['Total'].sum().plot(kind = 'bar',color = 'gold', figsize=(13,7))



*Figure 4.9: Sale of a product over the dataset time period over months*

4.11 Some of the other basic statistics

To get the idea of the dataset, some basic statistics have been extracted in this section.

Code to get the top five customers from the countries.

df.groupby(['Country'])['CustomerID'].count().sort\_values(ascending = False).head()

|  |  |
| --- | --- |
| **Country** | **Customers** |
| United Kingdom | 354345 |
| Germany | 9042 |
| France | 8342 |
| EIRE | 7238 |
| Spain | 2485 |

*Table 4.1: Top five country and customers count*

Code to get the average sales of a product based on stock code.

df.groupby(['Description'])['Quantity','Total'].mean().sort\_values(by = 'Quantity',ascending = False)



*Figure 4.10: Average sales of a product*



*Figure 4.11: Average sales by countries*

4.12 Analysis of an individual country

Below is the code to plot the monthly sales in the UK.

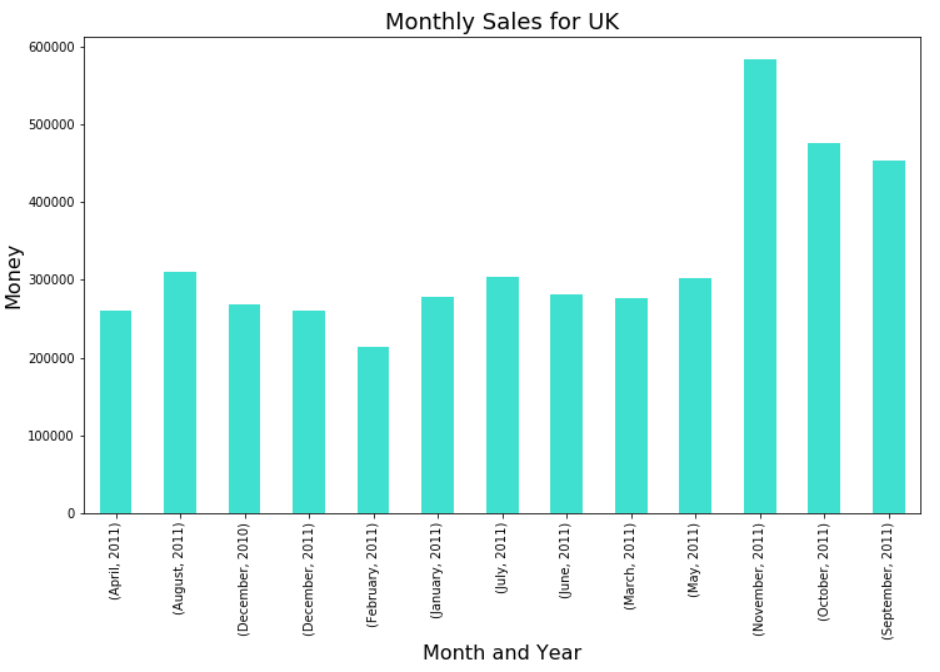
df['monthname'] = df['InvoiceDate'].dt.month\_name()

monthly1 = df[df['Country'] == 'United Kingdom'].groupby(['monthname','year'])['Quantity'].sum().plot(kind = 'bar', color = 'turquoise', figsize=(12,7))

monthly1.set\_xlabel("Month and Year",size=16)

monthly1.set\_ylabel("Money",size=16)

monthly1.set\_title("Monthly Sales for UK",size=18)



*Figure 4.12: Monthly sales for UK*

Code to get quantities of top brought products during peak months.

df[(df['Country'] == 'United Kingdom') & (df['month'] == 11) & (df['year'] == 2011)].groupby(['Description','monthname'])['Quantity'].sum().sort\_values(ascending = False).head()

|  |  |  |
| --- | --- | --- |
| **Description** | **Month** | **Quantity** |
| ASSTD DESIGN 3D PAPER STICKERS | November | 12551 |
| POPCORN HOLDER | November | 8036 |
| RABBIT NIGHT LIGHT | November | 6179 |
| PAPER CHAIN KIT 50'S CHRISTMAS | November | 5644 |
| JUMBO BAG RED RETROSPOT | November | 5341 |

*Table 4.2: Quantities of product bought during peak month*

Similarly, the analysis could be done for multiple over or underperforming countries to get the statistics. These findings could be useful to draw some useful conclusions towards the end of the report.

5. Apriori Algorithm

Apriori algorithm is used for mining frequent itemset and relevant association rules. It is devised to operate on a database containing a lot of transactions, for instance, items bought by customers in an online store.

5.1 Step by Step evaluation of the Program

The programming language used is python and Jupyter IDE (Integrated development environment) is used to check the real-time output.

Importing Libraries required for apriori.

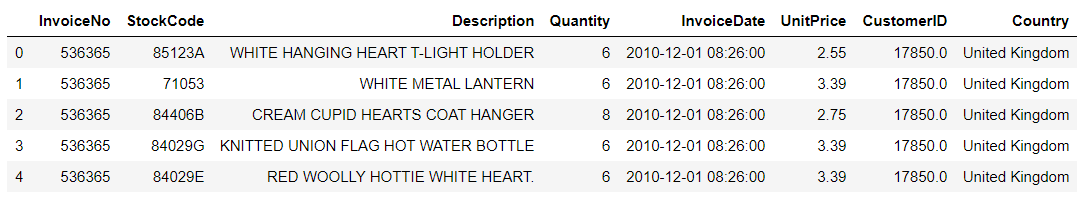
import pandas as pd

from mlxtend.frequent\_patterns import apriori

from mlxtend.frequent\_patterns import association\_rules

from matplotlib import pyplot as plt

df\_real = pd.read\_excel('C:\\OnlineRetail.xlsx',encoding = 'ISO-8859-1')



*Figure 5.1: Data set before apriori*

In the above code, we import the appropriate libraries such as pandas, mlxtend and matplotlib for the analysis of the data set.

Code for cleaning and formatting data

df = df\_real[['StockCode','Description','Quantity','InvoiceDate','UnitPrice','CustomerID','Country','InvoiceNo']]

# finding empty values in rows(axis defines rows

df[df.isnull().any(axis=1)].head()

# here we drop null values from df

df = df.dropna()

# customer id has floating values; convert them to string

df['CustomerID'] = df['CustomerID'].astype('int64')

df = df[df.Quantity > 0]

#Removing empty invoice number rows

df['Description'] = df['Description'].str.strip()

df.dropna(axis=0, subset=['InvoiceNo'], inplace=True)

#Removing credit transactions(invoice number starts with C)

df['InvoiceNo'] = df['InvoiceNo'].astype('str')

df = df[~df['InvoiceNo'].str.contains('C')]

# Adding year feature to the dataset

df['year'] = df['InvoiceDate'].dt.year

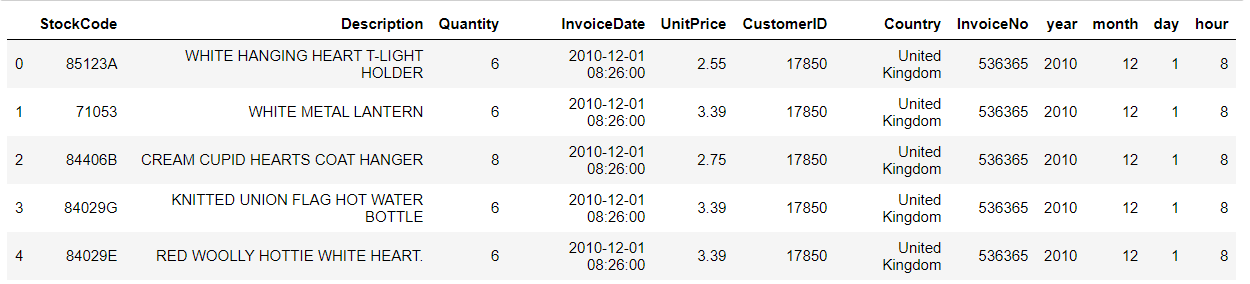
#Adding month and its name

df['month'] = df['InvoiceDate'].dt.month

df['day'] = df['InvoiceDate'].dt.day

df['hour'] = df['InvoiceDate'].dt.hour

df.head()



*Figure 5.2: Data set after cleansing*

Generating Baskets of transactions

basket = (df

.groupby(['InvoiceNo', 'Description'])['Quantity']

.sum().unstack().reset\_index().fillna(0)

.set\_index('InvoiceNo'))

After appropriate cleansing and formatting of data we now generate a basket of transaction. In the above line of code, we are using groupby function with invoice number and description based on quantity summed.

**Formatting inputs as per algorithm requirement (0's and 1's)**

def encode\_units(x):

if x <= 0:

return 0

if x >= 1:

return 1

basket\_sets = basket.applymap(encode\_units)

basket\_sets.drop('POSTAGE', inplace=True, axis=1)

A function is defined to fill in the null values with either 0 or 1 in the postage column, and it is appended to the basket.

**Generating a frequent Itemset for given basket of transactions.**

frequent\_itemsets = apriori(basket\_sets, min\_support=0.02, use\_colnames=True)

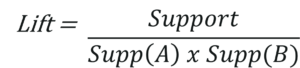
rules = association\_rules(frequent\_itemsets ,metric="lift", min\_threshold=1)

Here we apply Apriori on the basket\_sets generated with a minimum support of two percent.



*Figure 5.3: Frequent itemset*

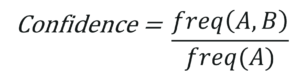
**Lift:** Lift indicates the strength of a rule over the random occurrence of A and B. It basically tells us the strength of any rule.



**Support:** It gives the fraction of transactions which contains item A and B. Basically Support tells us about the frequently bought items or the combination of items bought frequently.



**Confidence:** It tells us how often the items A and B occur together, given the number times A occurs.



**Modifying the association rules such as support, confidence or lift.**

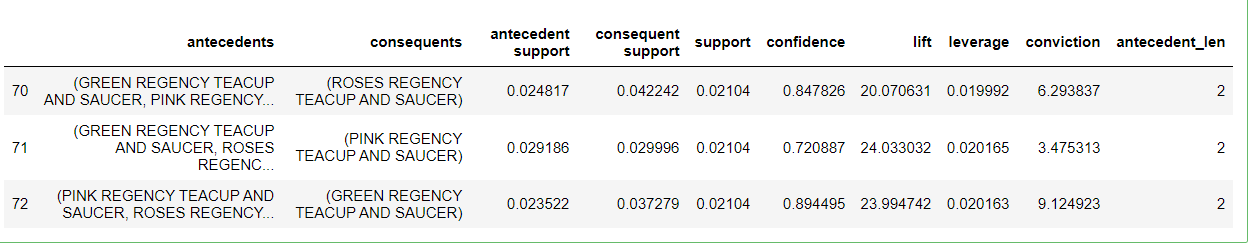
#Taking the variable antecedents length

rules["antecedent\_len"] = rules["antecedents"].apply(lambda x: len(x))

#APPLYING LIFT TO CHECK STRENGTH OF THE RULE BY SPECIFYING VALUES

rules[(rules['antecedent\_len'] >= 2) & (rules['lift'] >= 1) &

(rules['confidence'] >= .5)]



*Figure 5.4: Frequent itemsets after rule modification*

**Generating a frequent Itemset basket for a specific country**

basket\_Ireland = (df[df['Country'] == 'EIRE']

.groupby(['InvoiceNo', 'Description'])['Quantity']

.sum().unstack().reset\_index().fillna(0)

.set\_index('InvoiceNo'))

def encode\_units(x):

if x <= 0:

return 0

if x >= 1:

return 1

basket\_Ireland = basket\_Ireland.applymap(encode\_units)

Now we are generating a frequent itemset for each country with groupby function which groups Invoice no and description based on quantity and then we use the encode unit function to replace the null values with either 0 or 1. Now this process can be repeated by changing the name of the country at df[df['Country'] == 'France'] to generate different baskets for each country.

**6. RFM model and Clustering using k-means**

Customer classification is the division of the customers into separate groups based on similar features. It will help the business understand the customer needs better. Certain deals and discounts will be tailored according to their needs, all this to appeal to the unsatisfied customers and to reward the high performing customers. The dataset chosen for this case study is online retailers we segment customers based on their purchase pattern and spending habits. Segmentation has its benefits such as:

1. Decide the optimal price of a product
2. Design tailored discounts
3. New product development ideas
4. Build a strategy to target high spending customers

The RFM model is used to build the profile of the customers and classify them into different groups using k-means algorithm. Each group of customers has certain importance to the business. The customer is segmented based on three factors:

1. Recency – Number of days since customer made last purchase
2. Frequency – Number of transactions
3. Monetary – Total amount of revenue contribution

k-means clustering algorithm is the most popular unsupervised machine learning algorithm. The idea behind the implementation of k-means is to group similar data points and discover hidden patterns. The group of patterns is called a cluster and the process to form them is mentioned below:

1. The first step is to initialize k points, called means, randomly.
2. Categorize each item in the dataset to its closest mean and then update the mean’s coordinates, which are the averages of the items categorized in that mean so far
3. The second step is repeated for a given number of iterations until there is no change to the centroids, which means the allocation of data points to various clusters does not change

The code and the implementation part will be documented hereafter.

6.1 Libraries, file read, data frame

Import required libraries for the data frame, visualization, and clustering.

Import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import datetime as dt

import sklearn

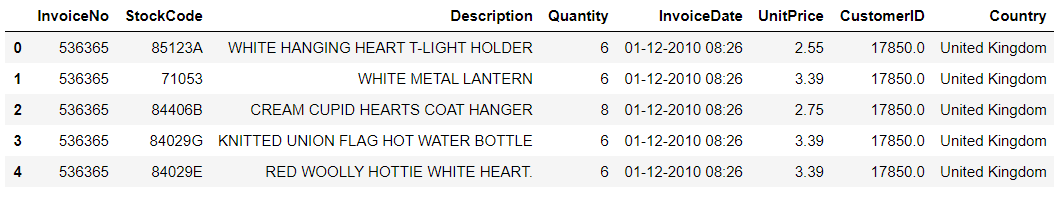
from sklearn.preprocessing import StandardScaler

from sklearn.cluster import Kmeans

Code to read the file and display the initial few rows in the data frame.

df = pd.read\_csv('C://OnlineRetailSecond.csv', sep=",", encoding="ISO-8859-1", header=0)

df.head()



*Figure 6.1: Initial stage of Data frame snapshot*

6.2 Dataset Cleaning

Implementation of RFM model needs a different method of data pre-processing. Below are the steps followed:

Code below is to check the percentage of null in each entity.

round(100\*(df.isnull().sum())/len(df), 2)

|  |  |
| --- | --- |
| **Variable** | **Percentage** |
| InvoiceNo | 0 |
| StockCode | 0 |
| Description | 0.27 |
| Quantity | 0 |
| InvoiceDate | 0 |
| UnitPrice | 0 |
| CustomerID | 24.93 |
| Country | 0 |

Table 6.1: Null percentages in each column

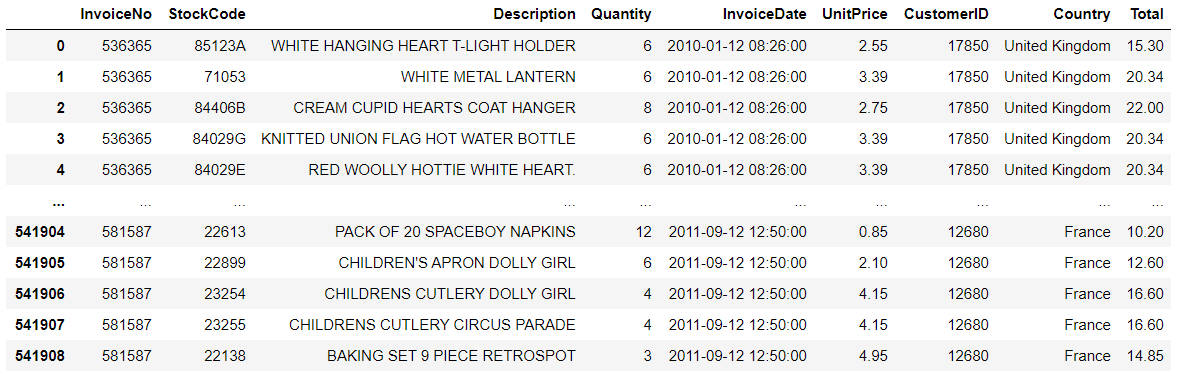
Code to drop missing values.

df = df.dropna()

Code to convert the data type of CustomerID from float to int.

df['CustomerID'] = df['CustomerID'].astype('int64')

The data frame after the cleaning process.



*Figure 6.2: Data frame after cleansing*

6.3 Data Preparation

Customer analysis is done based on three different factors.

1. **(R)Recency**: How recently a customer has made a purchase
2. **(F)Frequency**: How often a customer makes a purchase
3. **(M)Monetary** : How much money a customer spends on purchases

Code to add the ‘Total’ column to the data frame

df['Total'] = df['Quantity']\*df['UnitPrice']

Code the data type of ‘InvoiceDate’ field from string to datetime

df['InvoiceDate'] = pd.to\_datetime(df['InvoiceDate'])

Code to convert the data type of ‘InvoiceDate’ to int

rfm['InvoiceDate'] = rfm['InvoiceDate'].astype(int)

Compute the maximum date to know the last transaction date

max\_date = max(df['InvoiceDate'])

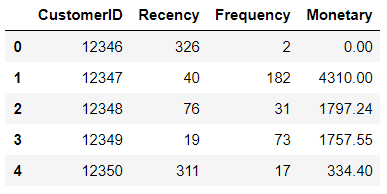
Code to calculate the RFM Modelling scores for each customer

rfm = df.groupby('CustomerID').agg({'InvoiceDate': lambda x: (max\_date - x.max()).days, 'InvoiceNo': lambda x: len(x), 'Total': lambda x: x.sum()})

Code to rename the columns into Recency, Frequency and Monetary

rfm.rename(columns={'InvoiceDate': 'Recency', 'InvoiceNo': 'Frequency', 'Total': 'Monetary'}, inplace=True)

rfm.reset\_index().head()



*Figure 6.3: RFM reference model*

Code to split RFM values into four segments using quantiles.

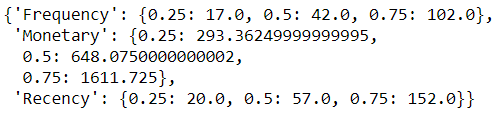
The quantiles are necessary because we need to segregate the discrete values into a specific number of groups, here, three quantiles are defined, which gives four groups.

quantiles = rfm.quantile(q=[0.25,0.5,0.75])

Convert data frame to a matrix like dictionary.

quantiles = quantiles.to\_dict()

quantiles



*Figure 6.4: RFM values split into four segments*

Code to rank the RFM attributes using quantiles.

In this part of the code, we give a score to each group of values.

* Anything with **1 is considered best** and **4 is considered poor**.
* **Recency** should be **low** for a customer to get the score **1**
* **Frequency** should be **high** for a customer to get **1**
* **Monetary** should be **high** for a customer to get **1**

def Recency(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 Freq\_Mon(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

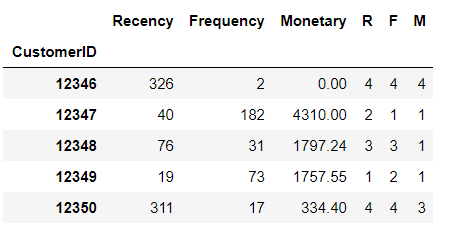
This part of the code calls the previous function which gives RFM scores.

rfm['R'] = rfm['Recency'].apply(Recency, args=('Recency',quantiles,))

rfm['F'] = rfm['Frequency'].apply(Freq\_Mon, args=('Frequency',quantiles,))

rfm['M'] = rfm['Monetary'].apply(Freq\_Mon, args=('Monetary',quantiles,))

rfm.head()



*Figure 6.5: RFM ranking split into four values*

Using the generated R F and M scores, Total rank of a customer is given. This is very important.

* + Here a customer’s rank is said to be best when it is 3

When, R, F, M is 1,1,1

Total Rank = R+F+M = 3.

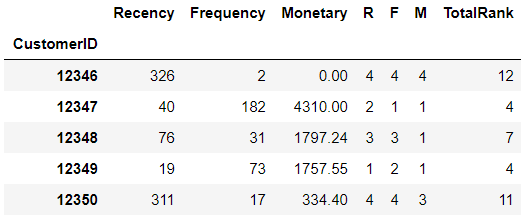
* + A customer’s rank is poor when it is 12

When, R, F, M is 4,4,4

Total Rank = R+F+M = 12.

rfm['TotalRank'] = rfm[['R', 'F', 'M']].sum(axis = 1)

rfm.head()



*Figure 6.6: ‘TotalRank’ column id the addition of R, F, M*

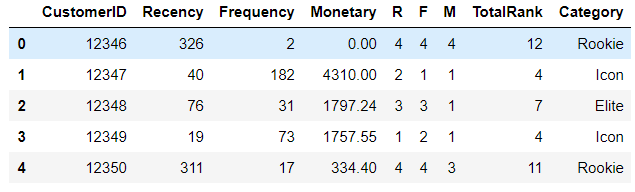
Now, in order to simplify, we give labels to the categories of customers. This is made possible using *pd.qcut* function. It takes the rank, groups number and the labels as input.

Rank\_Label = ['Icon', 'Elite', 'Select', 'Rookie']

Score = pd.qcut(rfm.TotalRank, q = 4, labels = Rank\_Label)

rfm['Category'] = Score.values

rfm.reset\_index().head()



*Figure 6.7: Added Loyalty category column*

6.4 k-means implementation

Code to handle negative and zero values similar to handling infinite numbers during log transformation

def NumHandler(num):

if num <= 0:

return 1

else:

return num

Code to apply handle\_neg\_n\_zero function to Recency and Monetary columns.

rfm['Recency'] = [NumHandler(x) for x in rfm.Recency]

rfm['Monetary'] = [NumHandler(x) for x in rfm.Monetary]

***The data in Recency, Frequency and Monetary is of different units. It is necessary to normalize to have a balanced distribution of points across all three factors. Hence, we perform Normalization on Recency, Frequency, Monetary.***

**Normalization** is a systematic approach of reducing tables to eliminate data repetition and unwanted characteristics like Insertion and Update on data.

Code to perform Log transformation to bring data into normal or near normal distribution.

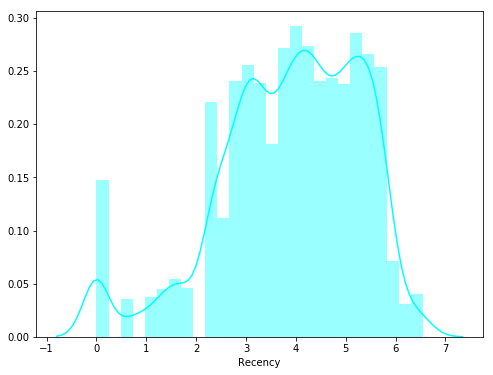
Normalize = rfm[['Recency', 'Frequency', 'Monetary']].apply(np.log, axis = 1).round(3)

Data distribution after data normalization for ‘Recency’.

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

Recency\_Plot = Normalize['Recency']

ax = sns.distplot(Recency\_Plot, color = 'cyan')



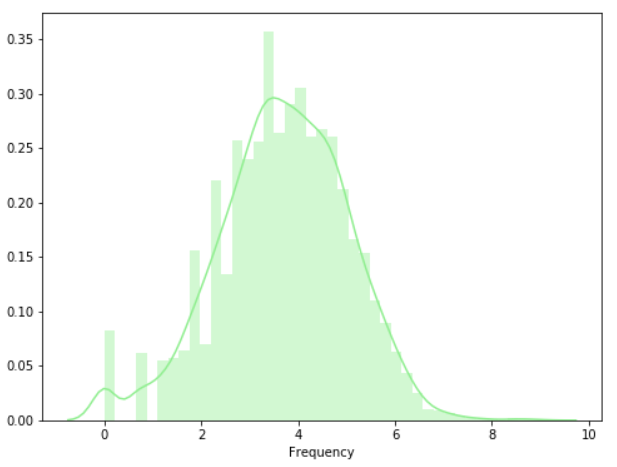
*Figure 6.8: Normalization for recency*

Data distribution after data normalization for Frequency

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

Frequency\_Plot = Normalize['Frequency']

ax = sns.distplot(Frequency\_Plot, color = 'lightgreen')



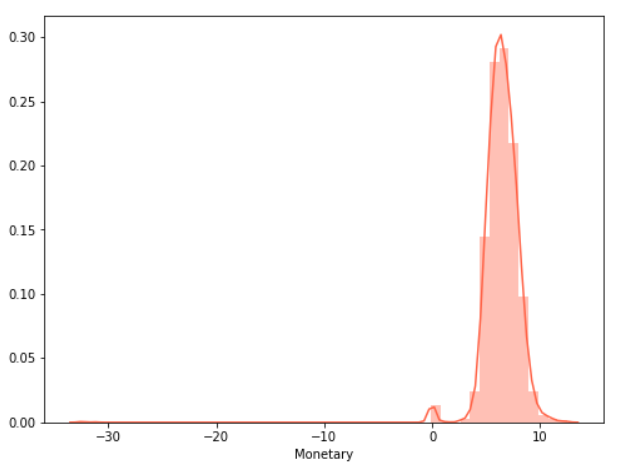
*Figure 6.9: Normalization for Frequency*

Data distribution after data normalization for Monetary

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

Monetary\_Plot = Normalize['Monetary']

ax = sns.distplot(Monetary\_Plot, color = 'tomato')



*Figure 6.10: Normalization for Monetary*

6.5 Rescaling the attributes

It is extremely important to rescale the variables so that they have a comparable scale. There are two common ways of rescaling:

1. Min-Max scaling
2. Standardisation (mean-0, sigma-1)

Here, Standardisation Scaling is used.

***Standardization is to reduce the distance between the features which will alter the clusters. Standardizing will attain fair distancing between points before clustering.***

Initiate scaler function is the function used to maintain a standardized distance between the points

scaler = StandardScaler()

Code to fit the transform

rfm\_scaled = scaler.fit\_transform(Normalize)

Code to convert it back to a data frame.

rfm\_scaled = pd.DataFrame(rfm\_scaled, index = rfm.index, columns = Normalize.columns)

6.6 Building the K -Means Clustering Model

**K-means clustering** is one of the simplest and popular unsupervised machine learning algorithms.

**Cluster**; Its is a group of similar things that are occurring closely together.

The algorithm works as follows:

* First, we initialize k points, called means, randomly.
* We categorize each item to its closest mean, and we update the mean’s coordinates, which are the averages of the items categorized in that mean so far.
* We repeat the process for a given number of iterations and in the end, clusters are formed

**Elbow-curve to get the right number of clusters**.

A fundamental step for any unsupervised algorithm is to determine the optimal number of clusters into which the data may be clustered. The **Elbow Method** is one of the most popular methods to determine this optimal value of k.

Below is the code to find K (Number of clusters)

ssd = {} #Initialize an empty dictionary for distance values

for k in range(1,20):

KM = KMeans(n\_clusters= k, init= 'k-means++', max\_iter= 100)

KM = KM.fit(rfm\_scaled)

ssd[k] = KM.inertia\_

#Plot the graph for the sum of square distance values and Number of Clusters

plt.figure(figsize=(10,8))

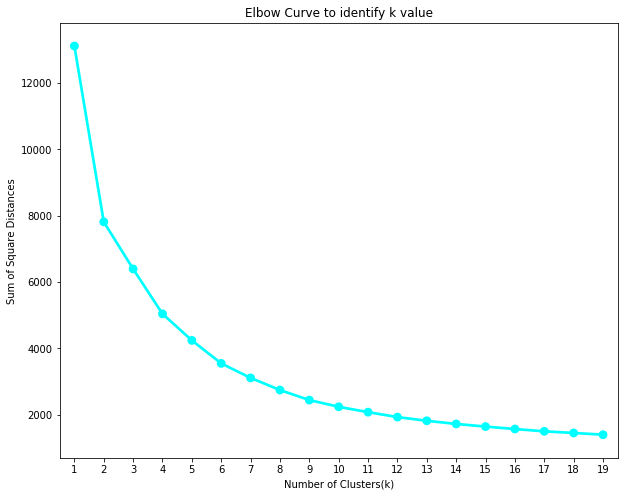
sns.pointplot(x = list(ssd.keys()), y = list(ssd.values()), color='cyan')

plt.xlabel('Number of Clusters(k)')

plt.ylabel('Sum of Square Distances')

plt.title('Elbow Curve to identify k value')

plt.show()



*Figure 6.11: Elbow Curve*

**In the above plot, since we see a consistent decrease only after 3, we select that point as K.**

Code to perform K-Mean Clustering or build the K-Means clustering model

Cluster = KMeans(n\_clusters= 3, init= 'k-means++', max\_iter= 1000)

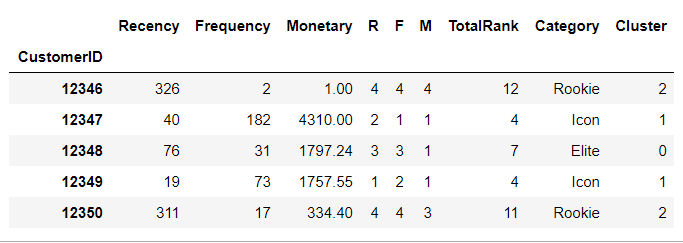
Cluster.fit(rfm\_scaled)

#Find the clusters for the observation given in the dataset

rfm['Cluster'] = Cluster.labels\_

rfm.head()

Here, each category is assigned a cluster after running the above code (Iterated 1000 times). Also, the first two categories, (Rookie and Select) are clubbed into same cluster. i.e. Cluster 2.



*Figure 6.12: Cluster for observation*

**Finally, the clustered plot of the above data is given below .**

from matplotlib import pyplot as plt

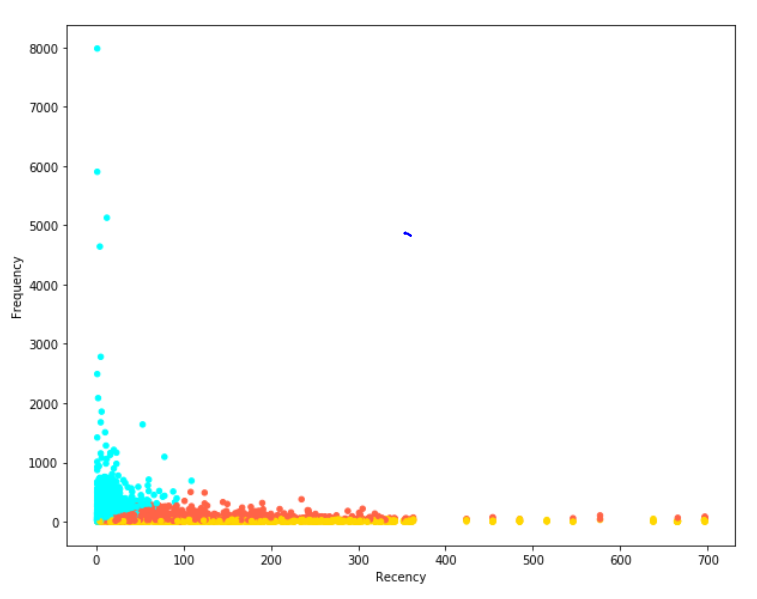
plt.figure(figsize=(7,7))

##Scatter Plot Frequency Vs Recency

Colors = ["tomato", "cyan", "gold"]

rfm['Color'] = rfm['Cluster'].map(lambda f: Colors[f])

ax = rfm.plot(kind="scatter", x="Recency", y="Frequency", figsize=(10,8), c = rfm['Color'])



*Figure 6.13: Frequency vs Recency*

Code to merge the rfm model with the main data frame.

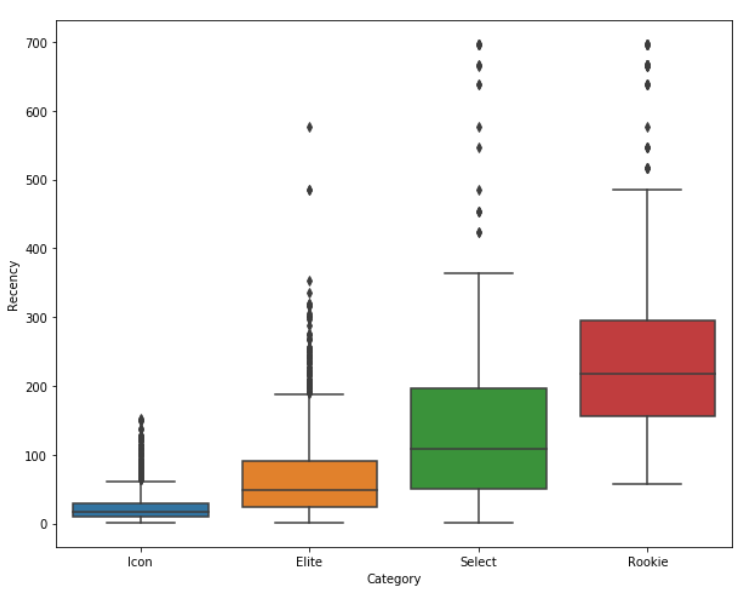
df\_NEW = pd.merge(df, rfm, on='CustomerID', how='inner')

df\_NEW.head()

Displaying a Box plot to visualize a graph between Cluster vs Recency

plt.figure(figsize=(10,8))

sns.boxplot(x='Category', y='Recency', data=rfm

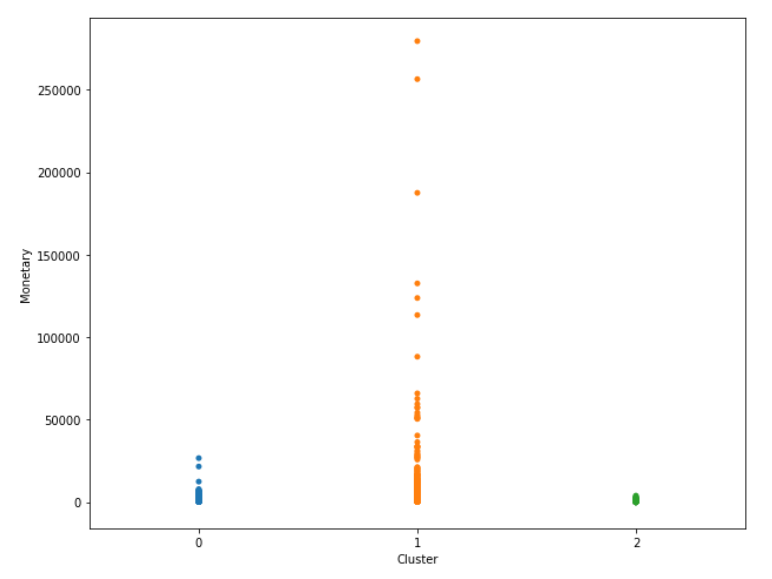


*Figure 6.14: Cluster vs Recency*

Displaying a strip plot to visualize a graph between Cluster vs Monetary

plt.figure(figsize=(10,8))

sns.stripplot(x='Cluster', y='Monetary', data=rfm)

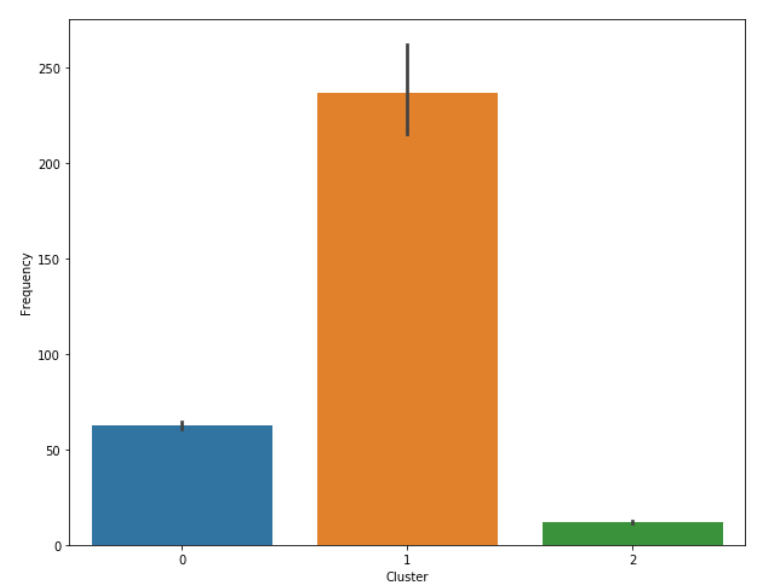


*Figure 6.15: Cluster vs Monetary*

Displaying a bar plot to visualize a graph between Cluster vs Frequency

plt.figure(figsize=(10,8))

sns.barplot(x='Cluster', y='Frequency', data=rfm)



*Figure 6.16: Cluster vs Frequency*

**7. Results, Discussions and Future work**

Descriptive questions:

**Data analysis**:

Which are the over-performing and under-performing countries based on Quantity?

**Over-performing**

**Country** **Quantity**

United Kingdom 4269472

Netherlands 200937

EIRE 140525

Germany 119263

France 111472

**Under-performing**

**Country** **Quantity**

Lebanon 386

Brazil 356

RSA 352

Bahrain 260

Saudi Arabia 80

What are the peak and least hours of sales?

The **peak** hours of sale are observed between **12 and 2 in the noon** and **drops** to lowest **after 6 till 8 in the evening**.

Which day in the week contributes to the highest and lowest sales?

**Thursday** has shown the **highest** amount of sales and **Sunday** notably has the **least** sales among all days.

What months of the year reflect the highest sales and lowest sales?

**Peak Months**

**Month Year Money Spent**

November 2011 1161817.380

October 2011 1039318.790

September 2011 952838.382

The months with the lowest sale are February and April.

What are the distinct characteristics of them?

Which product is getting sold the most?

Here is a list of top 25 most sold products with quantities.

**Description Quantity**

PAPER CRAFT , LITTLE BIRDIE 80995

MEDIUM CERAMIC TOP STORAGE JAR 77916

WORLD WAR 2 GLIDERS ASSTD DESIGNS 54415

JUMBO BAG RED RETROSPOT 46181

WHITE HANGING HEART T-LIGHT HOLDER 36725

ASSORTED COLOUR BIRD ORNAMENT 35362

PACK OF 72 RETROSPOT CAKE CASES 33693

POPCORN HOLDER 30931

RABBIT NIGHT LIGHT 27202

MINI PAINT SET VINTAGE 26076

PACK OF 12 LONDON TISSUES 25345

PACK OF 60 PINK PAISLEY CAKE CASES 24264

BROCADE RING PURSE 22963

VICTORIAN GLASS HANGING T-LIGHT 22433

ASSORTED COLOURS SILK FAN 21876

RED HARMONICA IN BOX 20975

JUMBO BAG PINK POLKADOT 20165

SMALL POPCORN HOLDER 18252

LUNCH BAG RED RETROSPOT 17697

60 TEATIME FAIRY CAKE CASES 17689

JUMBO BAG STRAWBERRY 16807

HEART OF WICKER SMALL 16775

GROW A FLYTRAP OR SUNFLOWER IN TIN 15940

PAPER CHAIN KIT 50'S CHRISTMAS 15617

COLOUR GLASS T-LIGHT HOLDER HANGING 15611

Which countries have spent the most money on orders?

**Country Amount Spent**

United Kingdom 7.308392e+06

Netherlands 2.854463e+05

EIRE 2.655459e+05

Germany 2.288671e+05

France 2.090240e+05

Which countries have ordered the most and the least?

**Country** **Orders**

United Kingdom 354345

Germany 9042

France 8342

EIRE 7238

Spain 2485

**Apriori**:

Which products/items have customers purchased together often?

Below is the list of **top 15** products mostly brought together.

| **antecedents** | **consequents** |
| --- | --- |
| **0** | (ALARM CLOCK BAKELIKE GREEN) | (ALARM CLOCK BAKELIKE RED) |
| **1** | (ALARM CLOCK BAKELIKE RED) | (ALARM CLOCK BAKELIKE GREEN) |
| **2** | (ALARM CLOCK BAKELIKE PINK) | (ALARM CLOCK BAKELIKE RED) |
| **3** | (ALARM CLOCK BAKELIKE RED) | (ALARM CLOCK BAKELIKE PINK) |
| **4** | (SPACEBOY LUNCH BOX) | (DOLLY GIRL LUNCH BOX) |
| **5** | (DOLLY GIRL LUNCH BOX) | (SPACEBOY LUNCH BOX) |
| **6** | (GARDENERS KNEELING PAD KEEP CALM) | (GARDENERS KNEELING PAD CUP OF TEA) |
| **7** | (GARDENERS KNEELING PAD CUP OF TEA) | (GARDENERS KNEELING PAD KEEP CALM) |
| **8** | (GREEN REGENCY TEACUP AND SAUCER) | (PINK REGENCY TEACUP AND SAUCER) |
| **9** | (PINK REGENCY TEACUP AND SAUCER) | (GREEN REGENCY TEACUP AND SAUCER) |
| **10** | (GREEN REGENCY TEACUP AND SAUCER) | (REGENCY CAKESTAND 3 TIER) |
| **11** | (REGENCY CAKESTAND 3 TIER) | (GREEN REGENCY TEACUP AND SAUCER) |
| **12** | (GREEN REGENCY TEACUP AND SAUCER) | (ROSES REGENCY TEACUP AND SAUCER) |
| **13** | (ROSES REGENCY TEACUP AND SAUCER) | (GREEN REGENCY TEACUP AND SAUCER) |
| **14** | (HEART OF WICKER LARGE) | (HEART OF WICKER SMALL) |

**RFM model** and clustering with **K-means** algorithm:

Can the customers be segmented based on their importance to the business?

Yes. Using the dataset, we were able to understand how particular customer has shopped over a year. Based on various factors we can conclude how important a customer is to the business.

Can the customers be segmented based on their recency, frequency, and monetary factors?

Who are the most and least valuable customers to the business?

Yes.

| **CustomerID** | **Recency** | **Frequency** | **Monetary** |
| --- | --- | --- | --- |
| **0** | 12346 | 326 | 2 | 0.00 |
| **1** | 12347 | 40 | 182 | 4310.00 |
| **2** | 12348 | 76 | 31 | 1797.24 |
| **3** | 12349 | 19 | 73 | 1757.55 |
| **4** | 12350 | 311 | 17 | 334.40 |

**Predictive question:**

Which types of customers are more likely to respond to a certain marketing strategy?

Using k-means algorithm model, the top valuable customers have been segmented. Also, apriori algorithm is implemented to find the frequent item sets and the sequence in which the products have been bought. Therefore, finding the frequent item sets exclusively bought by the icon customers would help the business to identify the primary and secondary products which interests their most vauable customers. This basically creates an opportunity for the business to keep their icon customers happy by provding them certain discount, or by sending promotional emails on those particular products and also ensuring that those products never go out of stock. This way the business could ensure that certain section of the customers respond to their marketing strategy.

Future work:

Which products are well performers common to all the countries?

This information helps the business to keep ordering the well performing products from vendors in advance. The benefit for the business is that it has the purchasing power to get items at the required discounts and not worry about the product sales in a country. This is because the product is well performing across all the countries where the online retailer still has its ventures, so the retailer could easily ship out the products to a economically feasible country where the product is still getting sold.

**8. Conclusion**

The approaches used in this project has enabled us to understand the data thoroughly and we’ve been able to extract valuable insights from the dataset. The process involved using multiple libraries in the code which are necessary for the algorithm. Some of them are apriori, mlxtend, sklearn, saeborn, matplotlib, numpy and pandas. The preprocessing part involved scrupulous filtering of all null, NAN, negative and invalid entries once they were identified. Organizing the data frame, dissemination of a few variables for convenient usage was also a part of processing the data. Conversion of datatypes from floating-point to integer and integer to string type was essential for certain values. Later, the approaches; Apriori and K-means Clustering was methodically implemented on the cleansed and formatted dataset. Each approach had its own outcome wherein, Apriori talks about the items: Clearly, item sets which are mostly bought together, with certain support, confidence and lift. We were able to generate the item sets for several instances like peak months, peak days, countries, group of customers and many more. Apriori completely speaks about the items and the pattern connect them which is drawn by user buying patterns. Using this, retailers can understand how to manage their stocks, organize things and plan marketing strategies to enhance their trade. Further, in the second approach, RFM segmentation was used to understand the customer groups we have. This method considers Recency, Frequency and Monetary as the deciding criteria for the customers to be segregated. Using these values, each customer was given a rank and was labelled accordingly for better representation. The ranked groups were finally used in K-means to generate meaningful clusters. It involved finding K value using Elbow curve and then iterating over the functions while finding the new means each time until proper clusters are formed. This enables the retailer to target their customers with varied business strategies for different group types. To club these two approaches, we extracted the frequent itemsets for the best customer group (Icon group), with which the retailer can understand what products can be clubbed with discounts and to which group of users he can advertise them. On the whole, the approaches traverse both over the items analysis and user data analysis to draw out beneficial cognizance.