

Customer segmentation

By- Nishtha Rathod

Introduction

- ❑ Suppose we have a company that is selling a product, and we want to know the selling performance of the product.

If we have the data of the consumers, we can segment the customers based on their buying behavior on the market.

- ❑ Keeping in mind that the data is huge and cannot be analyzed with our bare eye, thus here by using the K-means clustering algorithm in Python, customers are clustered into segments based on their buying behavior.

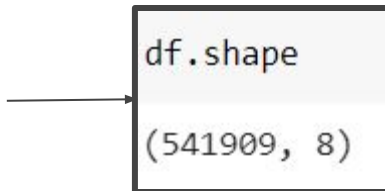
Plan Of Action

- *Gather the data*
 - *Create a Recency Frequency Monetary (RFM) table*
 - *Manage skewness and scale each variable*
 - *Explore the data*
 - *Cluster the data*
 - *Interpret the result*
-

Data Collection

The dataset here has been taken from a UK based Online Retail. This contains transactional data from December 1st, 2010 to December 9th, 2011. Each row represents the transaction that occurred, this includes the product name, quantity, price, and other columns that represent ID.

Size of the
dataset



```
df.shape  
(541909, 8)
```

Here we are not gonna use the whole dataset, but instead, we will sample 10,000 rows from the dataset, and we assume that as the whole transactions the customers do.

(Code for the sampling of Data)

```
#Importing dataset
df = pd.read_excel('/content/Online Retail.xlsx')

#Sampling dataset
df_fix = df.sample(10000, random_state = 42)
df_fix.shape
```

Dataset after sampling

| | InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID | Country |
|--------|-----------|-----------|---------------------------------|----------|---------------------|-----------|------------|----------------|
| 47912 | 540456 | 48185 | DOORMAT FAIRY CAKE | 2 | 2011-01-07 12:14:00 | 7.95 | 13534.0 | United Kingdom |
| 342630 | 566891 | 23013 | GLASS APOTHECARY BOTTLE TONIC | 4 | 2011-09-15 13:51:00 | 3.95 | 14894.0 | United Kingdom |
| 288183 | C562139 | 21313 | GLASS HEART T-LIGHT HOLDER | -4 | 2011-08-03 10:10:00 | 0.85 | 12921.0 | United Kingdom |
| 325368 | 565438 | 22382 | LUNCH BAG SPACEBOY DESIGN | 4 | 2011-09-04 13:56:00 | 1.65 | 17229.0 | United Kingdom |
| 331450 | 566016 | 21212 | PACK OF 72 RETROSPOT CAKE CASES | 24 | 2011-09-08 12:20:00 | 0.55 | 15144.0 | United Kingdom |

Create the RFM table

RFM (Recency, Frequency, Monetary value) Analysis measures how recently, how often, and how much money a customer has given to your brand.

Recency:- Subtract the snapshot date with the date the last transaction occurred.

Frequency:- Count how many transactions by each customer.

Monetary Value:- Summation of all the transactions that occurred.

This will help us find :

- Who are the best customers?
- Which customers can contribute to the churn rate?
- Who has the potential to become valuable customers?

(Code for the RFM table)

Thus we get the following:

| | Recency | Frequency | MonetaryValue |
|------------|---------|-----------|---------------|
| CustomerID | | | |
| 12347.0 | 40 | 5 | 133.20 |
| 12348.0 | 249 | 2 | 120.88 |
| 12349.0 | 19 | 2 | 312.75 |
| 12352.0 | 73 | 5 | 80.85 |
| 12354.0 | 233 | 2 | 33.30 |

```
# Convert to show date only
from datetime import datetime
df_fix["InvoiceDate"] = df_fix["InvoiceDate"].dt.date

# Create TotalSum columnn
df_fix["TotalSum"] = df_fix["Quantity"] * df_fix["UnitPrice"]

# Create date variable that records recency
import datetime
snapshot_date = max(df_fix.InvoiceDate) + datetime.timedelta(days=1)

# Aggregate data by each customer
customers = df_fix.groupby(['CustomerID']).agg({
    'InvoiceDate': lambda x: (snapshot_date - x.max()).days,
    'InvoiceNo': 'count',
    'TotalSum': 'sum'})

# Rename columns
customers.rename(columns = {'InvoiceDate': 'Recency',
                             'InvoiceNo': 'Frequency',
                             'TotalSum': 'MonetaryValue'}, inplace=True)
```

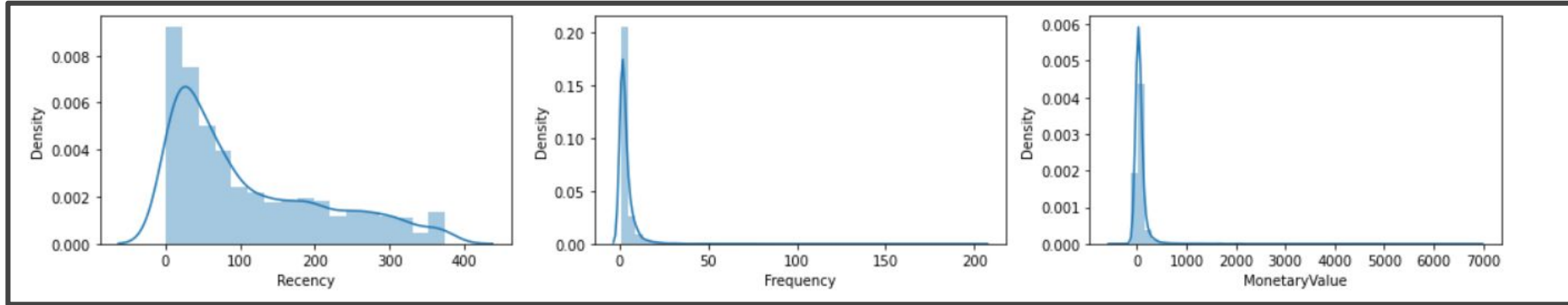
Manage Skewness and Scaling

The data should meet the assumptions where the variations are not skewed and have the same mean and variance.

Because of that, we have to manage the skewness of the variables.

We need to transform the data, so it has a more symmetrical form.

Visualizations of each variable(RFM) is given below:



The methods used to manage the skewness are:

- Log transformation
- Square-root transformation
- Box-cox transformation

When we use the above methods to manage the skewness, the plot in which the variable's value is closest to 0, will tend to have a symmetrical form.

Also, these transformations can only be applied to positive values, thus we need to deal with Monetary Values differently.

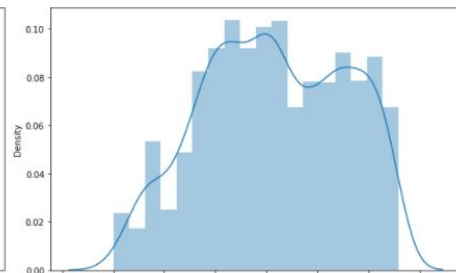
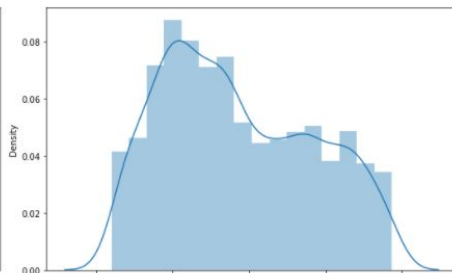
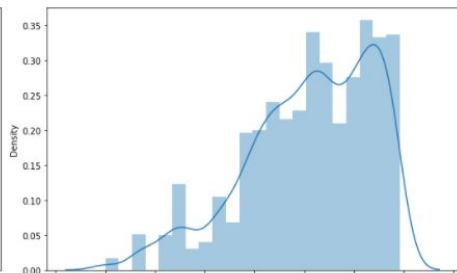
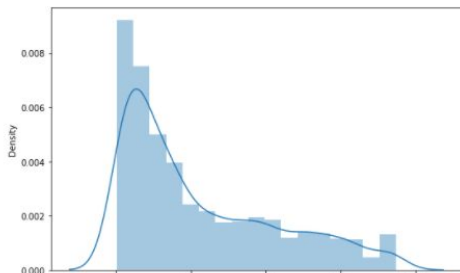
Below are the visualizations of each variable, with and without transformation:

Log
Transformation

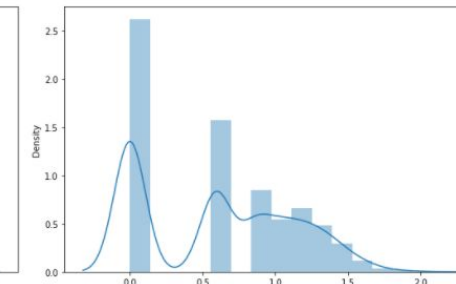
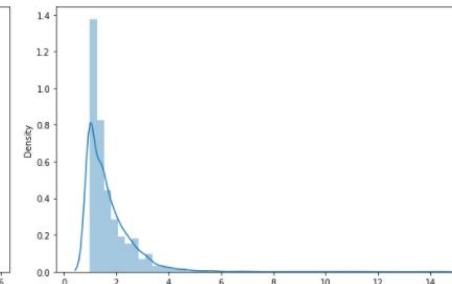
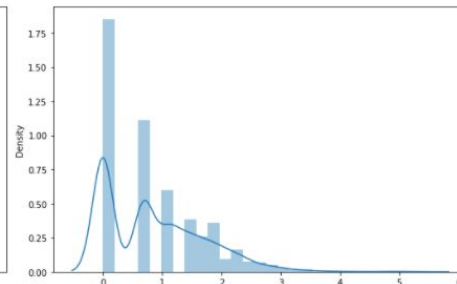
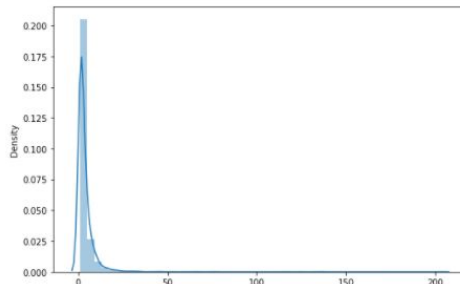
Square-root
Transformation

Box-cox
Transformation

Recency:



Frequency:

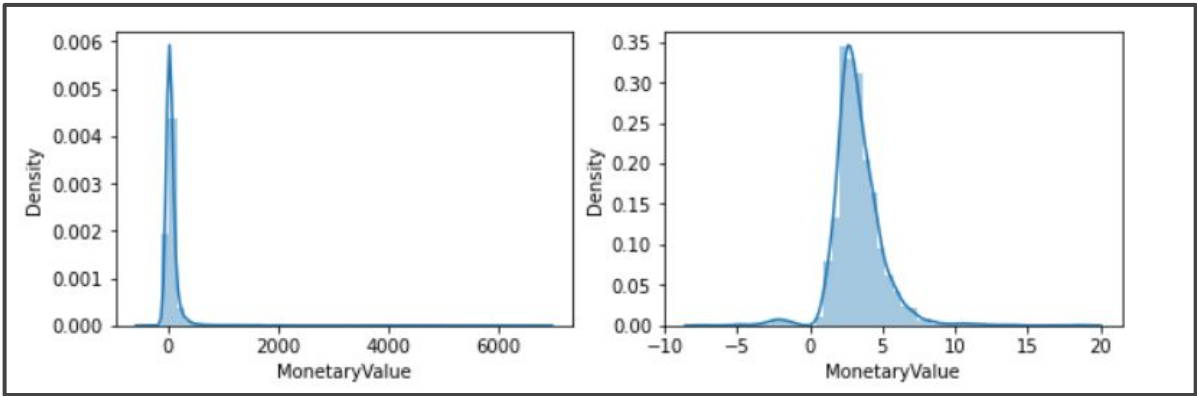


- Based on that visualization, it shows that the variables with box-cox transformation shows a more symmetrical form rather than the other transformations.

To make sure, we calculate each variable using the skew function. The result looks like this,

```
variable, log, sqrt, box-cox
Recency -0.72 0.32 -0.1
Frequency 0.85 3.67 0.16
```

- Based on that calculation, we will utilize variables that use box-cox transformations. Except for the MonetaryValue variable because the variable includes negative values. To handle this variable, we can use cubic root transformation to the data, so the comparison looks like this,



- By using the transformation, we will have data that less skewed. The skewness value declines from 16.63 to 1.16. Therefore, we can transform the RFM table with this code,

```
# Set the Numbers
customers_fix = pd.DataFrame()
customers_fix["Recency"] = stats.boxcox(customers["Recency"])[0]
customers_fix["Frequency"] = stats.boxcox(customers["Frequency"])[0]
customers_fix["MonetaryValue"] = pd.Series(np.cbrt(customers["MonetaryValue"])).values
customers_fix.tail()
```

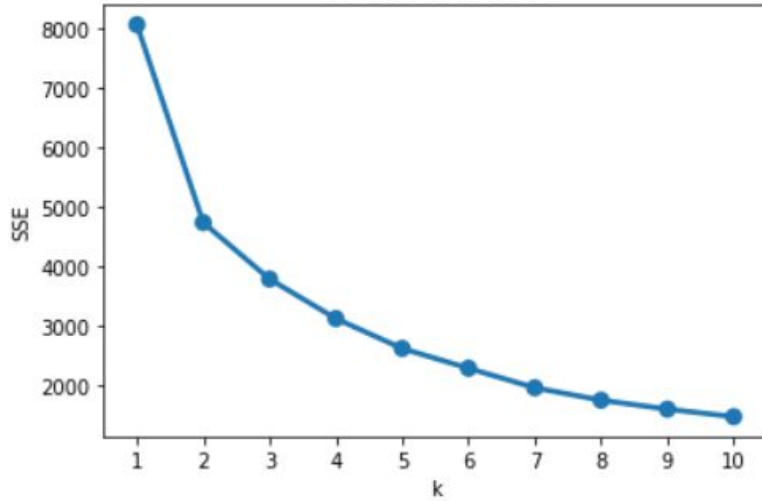
| | Recency | Frequency | MonetaryValue |
|------|----------|-----------|---------------|
| 2685 | 7.832068 | 0.591193 | 3.408514 |
| 2686 | 1.269495 | 1.435599 | 5.907565 |
| 2687 | 4.288385 | 0.591193 | -1.669108 |
| 2688 | 1.665555 | 1.615329 | 4.273206 |
| 2689 | 6.340700 | 1.017445 | 4.087250 |

- If we look at the plot once more, each variable doesn't have the same mean and variance. We have to normalize it. To normalize, we can use the StandardScaler object from scikit-learn library to do it. Then we can finally do clustering of the data.

Modeling

- ❑ Right after we preprocess the data, now we can focus on modeling. To make segmentation from the data, we can use the K-Means algorithm to do this.
- ❑ K-Means algorithm is an unsupervised learning algorithm that uses the geometrical principle to determine which cluster belongs to the data. By determine each centroid, we calculate the distance to each centroid. Each data belongs to a centroid if it has the smallest distance from the other. It repeats until the next total of the distance doesn't have significant changes than before.
- ❑ To make our clustering reach its maximum performance, we have to determine which hyperparameter fits the data. To determine which hyperparameter is the best for our model and data, we can use the "[Elbow method](#)" to decide.

The Elbow Method

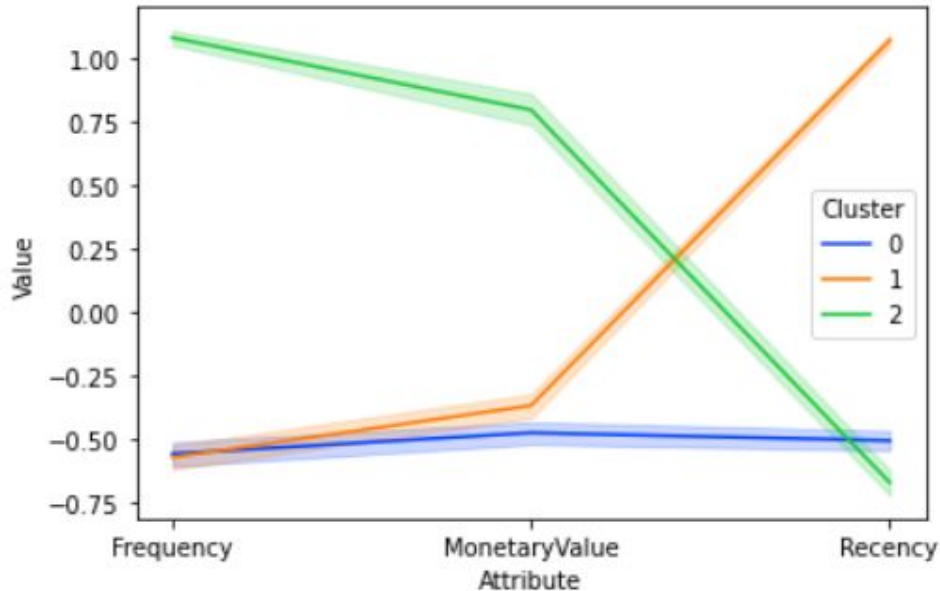


NOTE: SSE is the sum of the squared differences between each observation and its group's mean. It can be used as a measure of variation within a cluster. If all cases within a cluster are identical the SSE would then be equal to 0.

- ❑ The x-axis is the value of the k , and the y-axis is the SSE value of the data. We will take the best parameter by looking at where the k -value will have a linear trend on the next consecutive k .
- ❑ Based on our observation, the k -value of 3 is the best hyperparameter for our model because the next k -value tend to have a linear trend. Therefore, our best model for the data is K-Means with the number of clusters is 3.

Interpret the result

We can analyze the segments using **snake plot**. It requires the normalized dataset and also the cluster labels. By using this plot, we can have a good visualization from the data on how the cluster differs from each other.



- ❑ We infer that cluster 2 is frequent, spend more, and they buy the product recently. Therefore, it could be the cluster of a **loyal customer**.
- ❑ Then, cluster 1 is less frequent, less to spend, but they buy the product recently. Therefore, it could be the cluster of **new customer**.
- ❑ Finally, cluster 0 is less frequent, less to spend, and they buy the product at the old time. Therefore, it could be the cluster of **churned customers**.

In conclusion, customer segmentation is really necessary for knowing what characteristics exist for each customer.

References

-The code used to develop this solution can be found here:

[Github repo](#)

-The notebook may be downloaded and executed on local machines, but the dependencies may need to be installed first.

- All Python code and LaTeX rendering in the notebook and in this pdf is my own.