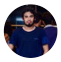
# MALL CUSTOMER SEGMENTATION USING CLUSTERING ALGORITHM

**Conference Paper** · March 2021



Asith Ishantha
Future University Hakodate

# MALL CUSTOMER SEGMENTATION USING CLUSTERING ALGORITHM

M.A.Ishantha

USE/2017/OCT/0045

Lanka Nippon BizTech Institute

March 2021

# Declaration

I declare that this report is my own work and has not been submitted in any form for another degree or diploma at any university or other institution of higher education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

| | |
|---|---|
| Name of Student | Signature of Student |
| Asith Ishantha | |
| Date: 02/03/2021 | |
| Name of supervisor: | Signature of Supervisor |
| Janith Perera | Date: |

# Acknowledgement

*I would like to express my special thanks of gratitude to our lecturer "Janith Perera" who gave us the golden opportunity to do this wonderful project on the "MALL CUSTOMER SEGMENTATION USING CLUSTERING ALGORITHM", which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to him. Secondly, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.*

# Abstract

Customer segmentation is a separation of a market into multiple distinct groups of consumers who share the similar characteristics. Segmentation of market is an effective way to define and meet customer needs. Unsupervised Machine Learning Techniques, K-Means Clustering Algorithm, Minibatch K-Means and Hierarchical Clustering are used to perform Market Basket Analysis. Market Basket Analysis is carried out to predict the target customers who can be easily converged, among all the customers. In order to allow the marketing team to plan the strategy to market the new products to the target customers which are similar to their interests.

Key words: Target Customers, Clusters, Unsupervised Learning, K-Means, Minibatch K-Means, Hierarchical Clustering Segmentation, Market Basket Analysis

# Table of Contents

# Introduction

Management and maintain of customer relationship have always played a vital role to provide business intelligence to organizations to build, manage and develop valuable long term customer relationships. The importance of treating customers as an organizations main asset is increasing in value in present day and era. Organizations have an interest to invest in the development of customer acquisition, maintenance and development strategies. The business intelligence has a vital role to play in allowing companies to use technical expertise to gain better customer knowledge and Programs for outreach. By using clustering techniques like k-means, customers with similar means are clustered together. Customer segmentation helps the marketing team to recognize and expose different customer segments that think differently and follow different purchasing strategies. Customer segmentation helps in figuring out the customers who vary in terms of preferences, expectations, desires and attributes. The main purpose of performing customer segmentation is to group people, who have similar interest so that the marketing team can converge in an effective marketing plan. Clustering is an iterative process of knowledge discovery from vast amounts of raw and unorganized data. Clustering is a type of exploratory data mining that is used in many applications, such as machine learning, classification and pattern recognition.

# Literature Review

## Customer Segmentation

Over the years, as there is very strong competition in the business world, the organizations have to enhance their profits and business by satisfying the demands of their customers and attract new customers according to their needs. The identification of customers and satisfying the demands of each customer is a very complex task. This is because customers may be different according to their demands, desires, preferences and so on. Instead of "one-size-fits-all" approach, customer segmentation clusters the customers into groups sharing the same properties or behavioural characteristics. According to, [1] customer segmentation is a strategy of dividing the market into homogenous groups. The data used in customer segmentation technique that divides the customers into groups depends on various factors like, demographical conditions, data geographical conditions and economic conditions as well as behavioural patterns. The customer segmentation technique allows the business to make better use of their marketing budgets, gain a competitive edge over their rival companies, demonstrating the better knowledge of the needs of the customer. It also helps an organization in, increasing their marketing efficiency, plan the marketing budget, determining new market opportunities, making better brand strategy, identifying customers retention.

According to [1], Decision makers use many variables to segment customers. Demographic variables such as age, gender, family, education level and income are the easiest and common variables for segmentation. Socio- cultural, geographic, psychographic and behavioural variables are the other major variables that are used for segmentation.

[2], presented various clustering algorithms taking into account the characteristics of Big Data such as size, noise, dimensionality, algorithm calculations, cluster shape and presented a brief overview of the various clustering algorithms grouped under partitioning, hierarchical, density, grid-based and model-based algorithms.

[4] explored the necessity of segmentation of the customers using clustering algorithms as the core functionality of CRM. The mostly used K-Means and Hierarchical Clustering were studied and the advantages and disadvantages of these techniques were highlighted. At last, the idea of creating a hybrid approach is addressed by integrating the above two strategies with the potential to surpass the individual designs.

[5], Merged clustering of fuzzy c-means and genetic algorithms to cluster, steel industry customers, by using the LRFM variables (length, recency, frequency, monetary value) system, customers were divided into two clusters

### Clustering and K-Means Algorithm

Clustering algorithms generates clusters such that within the clusters are similar based on some characteristics. Similarity is defined in terms of how close the objects are in space.

According to [1], K-means algorithm in one of the most popular centroid based algorithms. Suppose data set, D, contains n objects in space. Partitioning methods distribute the objects in D into k clusters, C1,……Ck , that is, Ci ⊂ D and Ci ∩Cj = ∅ for (1 ≤ i, j ≤ k). A centroid-based partitioning technique uses the centroid of a cluster, Ci, to represent that cluster. Conceptually, the centroid of a cluster is its centre point. The difference between an object p ∈ Ci and ci, the representative of the cluster, is measured by dist(p,ci), where dist(x,y) is the Euclidean distance between two points x and y.

**Algorithm:** The k-means algorithm for partitioning, where each cluster's centre is represented by the mean value of the objects in the cluster. Input: k: the number of clusters, D: a data set containing n objects. Output: A set of k clusters. Method: (1) arbitrarily choose k objects from D as the initial cluster centres; (2) repeat (3) (re)assigns each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster; (4) update the cluster means, that is, calculate the mean value of the objects for each cluster; (5) until no change.

## Proposed System

We are going to aim to cluster a data set that is about behaviour of the customers having credit card using many unsupervised algorithms.

**Our research question is "How many clusters can we distinguish the customers according to their transactions or behaviours?"**

### General View of Data

The data set has 8950 transactions or information about account that belong to customers.

### Features

CUSTID: Identification of Credit Card holder (Categorical)

BALANCE: Balance amount left in their account to make purchases

BALANCE FREQUENCY: How frequently the Balance is updated, score between 0 and 1 (1 = frequently updated, 0 = not frequently updated)

PURCHASES: Number of purchases made from account

ONE OFF PURCHASES: Maximum purchase amount done in one-go

INSTALLMENTS PURCHASES: Amount of purchase done in installment

CASH ADVANCE: Cash in advance given by the user

PURCHASES FREQUENCY: How frequently the Purchases are being made, score between 0 and 1 (1 = frequently purchased, 0 = not frequently purchased)

ONE OFF PURCHASES FREQUENCY: How frequently Purchases are happening in one-go (1 = frequently purchased, 0 = not frequently purchased)

PURCHASES INSTALLMENTS FREQUENCY: How frequently purchases in installments are being done (1 = frequently done, 0 = not frequently done)

CASH ADVANCE FREQUENCY: How frequently the cash in advance being paid

CASH ADVANCE TRX: Number of Transactions made with "Cash in Advanced"

PURCHASES TRX: Number of purchase transactions made

CREDIT LIMIT: Limit of Credit Card for user

PAYMENTS: Amount of Payment done by user

MINIMUM PAYMENTS: Minimum number of payments made by user

PRC FULL PAYMENT: Percent of full payment paid by user

TENURE: Tenure of credit card service for user

## Methodology

### Clustering

Clustering is one of the most common methods used in exploring data to obtain a clear understanding of the data structure. It can be characterized as the task of finding the subtitles and subgroups in the complete dataset. Similar data is clustered in many subgroups. A cluster refers to a collection of aggregated data points due to some similarities. Clustering is used in Market basket analysis used to segment the customers based on their behaviours and transactions.

### K Means Clustering Algorithm

K Means Clustering is the most common and simplest Machine learning algorithm and it follows an iterative approach which attempts to partition the dataset into different “k” number of predefined and non-overlapping subgroups where each data point belongs to only one subgroup according to their similar qualities.

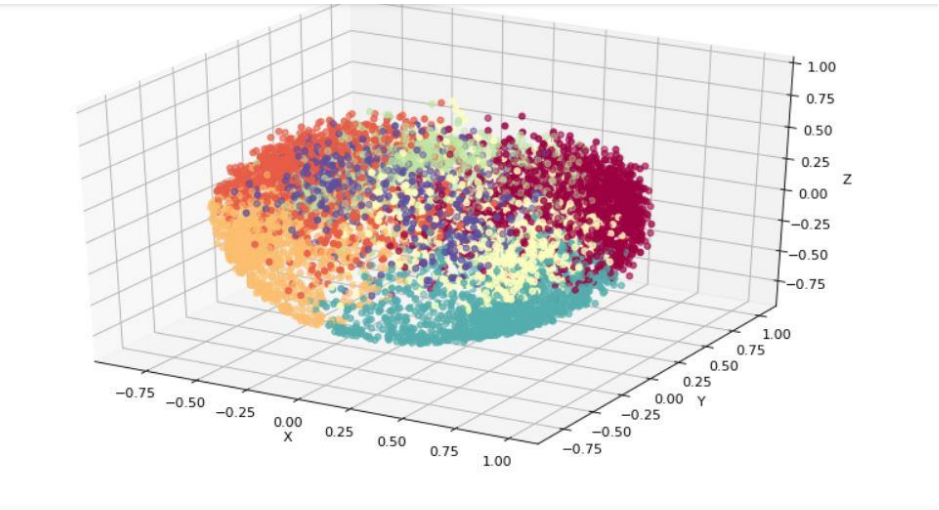


*Figure 1:K Means Clustering Algorithm*

### Minibatch K-Means Clustering Algorithm

Mini Batch K-means algorithm ‘s [6] main idea is to use small random batches of data of a fixed size, so they can be stored in memory. Each iteration a new random sample from the dataset is obtained and used to update the clusters and this is repeated until convergence. Each mini batch updates the clusters using a convex combination of the values of the prototypes and the data, applying a learning rate that decreases with the number of iterations. This learning rate is the inverse of the number of data assigned to a cluster during the process. As the number of iterations increases, the effect of new data is reduced, so convergence can be detected when no changes in the clusters occur in several consecutive iterations.

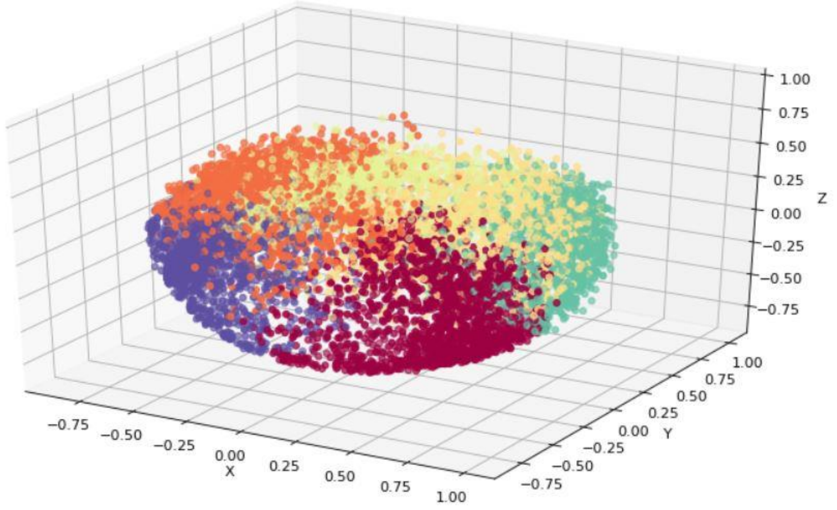


*Figure 2:Minibatch K-Means Clustering Algorithm*

## Hierarchical Clustering Segmentation

[7]The output of this model is a set of visualized clusters, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other in features.

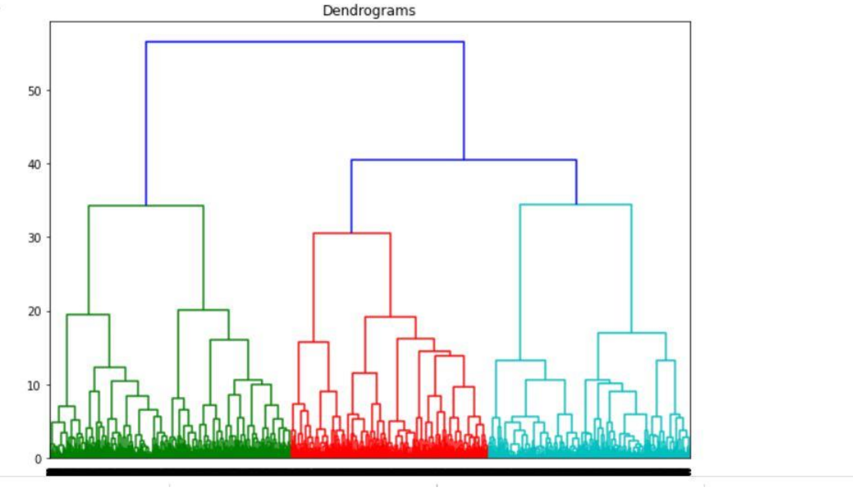


*Figure 3:Hierarchical Clustering Segmentation*

## Elbow Method

Elbow method is a tool used for analysing the clusters formed from our dataset and helps to interpret the appropriate number of optimal clusters in dataset. From this method the optimal number of clusters for our dataset is found to be seven.

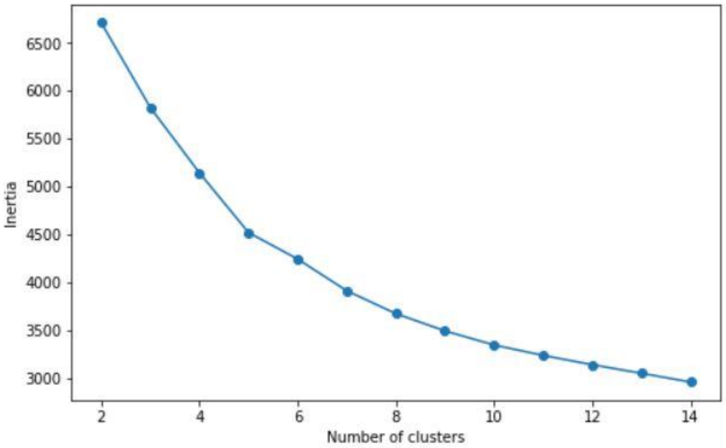


*Figure 4:Elbow Method*

## General View of Data

The data set has 8950 transactions or information about account that belong to customers.

## Features

CUSTID: Identification of Credit Card holder (Categorical)

BALANCE: Balance amount left in their account to make purchases

BALANCE FREQUENCY: How frequently the Balance is updated, score between 0 and 1 (1 = frequently updated, 0 = not frequently updated)

PURCHASES: Number of purchases made from account

ONE OFF PURCHASES: Maximum purchase amount done in one-go

INSTALLMENTS PURCHASES: Amount of purchase done in installment

CASH ADVANCE: Cash in advance given by the user

PURCHASES FREQUENCY: How frequently the Purchases are being made, score between 0 and 1 (1 = frequently purchased, 0 = not frequently purchased)

ONE OFF PURCHASES FREQUENCY: How frequently Purchases are happening in one-go (1 = frequently purchased, 0 = not frequently purchased)

PURCHASES INSTALLMENTS FREQUENCY: How frequently purchases in installments are being done (1 = frequently done, 0 = not frequently done)

CASH ADVANCE FREQUENCY: How frequently the cash in advance being paid

CASH ADVANCE TRX: Number of Transactions made with "Cash in Advanced"

PURCHASES TRX: Number of purchase transactions made

CREDIT LIMIT: Limit of Credit Card for user

PAYMENTS: Amount of Payment done by user

MINIMUM PAYMENTS: Minimum number of payments made by user

PRC FULL PAYMENT: Percent of full payment paid by user

TENURE: Tenure of credit card service for user

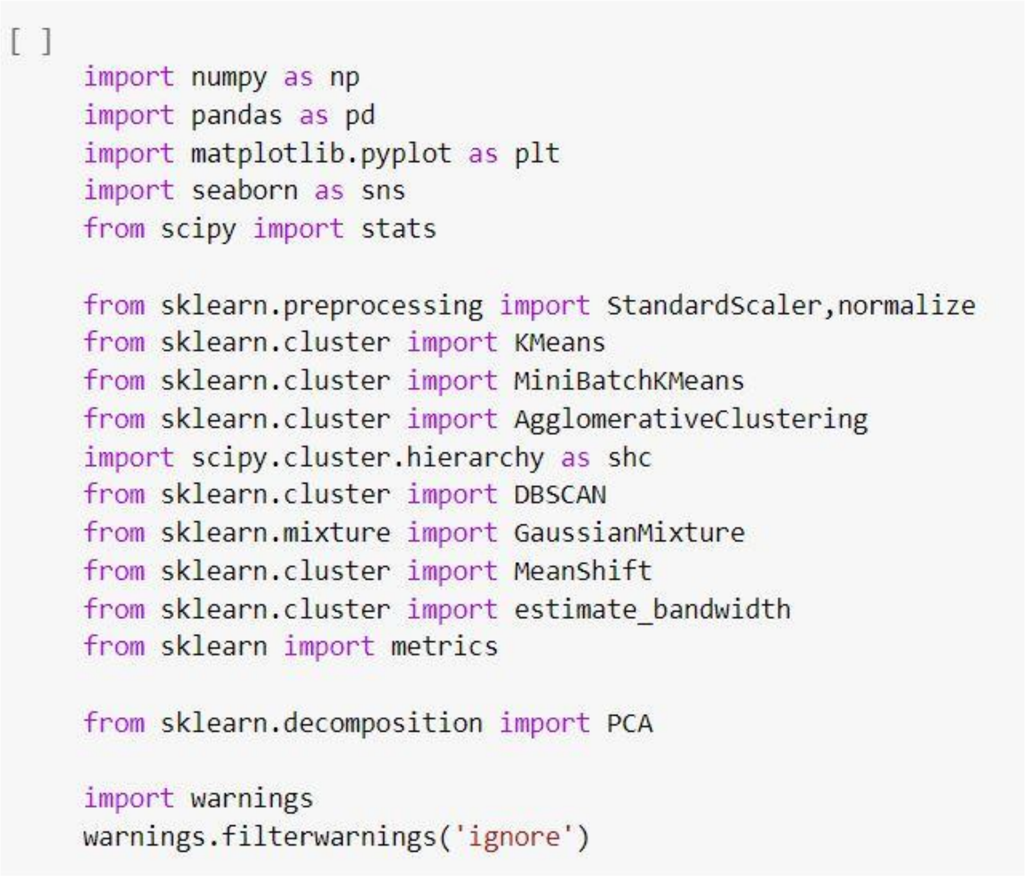
**Reason to use Unsupervised Learning Algorithms**

Unlike Supervised Learning, Unsupervised Learning has only independent variables and no corresponding target variable. The data is unlabelled. The aim of unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

We are going to examine a dataset that is about credit card users for segmentation. There is no any feature about label of customers. That is to say, we don’t have information about customer’s characteristics. We are going to try clustering clients through identifying similarities with machine learning algorithms. Segmentation of customers has a pretty significant position for companies in new marketing disciplines. Firms must reach to the right target audiences with right approaches because of costs.

we have started with data pre-processing such as filling the missing values, standardization etc.

First of all, we have to import all necessary libraries

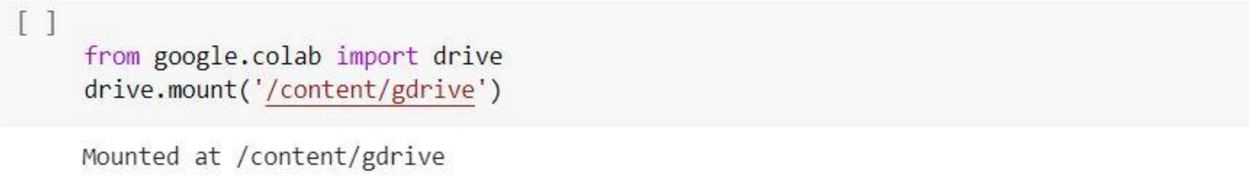


```
[ ]
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from scipy import stats

    from sklearn.preprocessing import StandardScaler,normalize
    from sklearn.cluster import KMeans
    from sklearn.cluster import MiniBatchKMeans
    from sklearn.cluster import AgglomerativeClustering
    import scipy.cluster.hierarchy as shc
    from sklearn.cluster import DBSCAN
    from sklearn.mixture import GaussianMixture
    from sklearn.cluster import MeanShift
    from sklearn.cluster import estimate_bandwidth
    from sklearn import metrics

    from sklearn.decomposition import PCA

    import warnings
    warnings.filterwarnings('ignore')
```

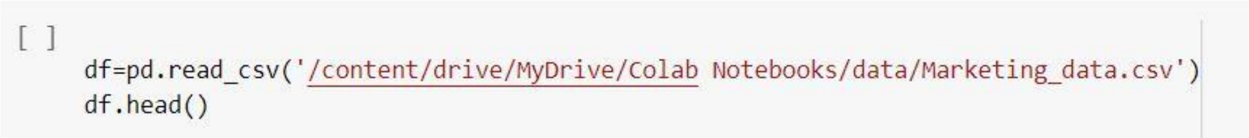
*Figure 5: importing libraries*

After that, we have to mount Jupiter notebook in to drive with destination folder and save the dataset inside folder.

We can use following code to read data from .csv file.



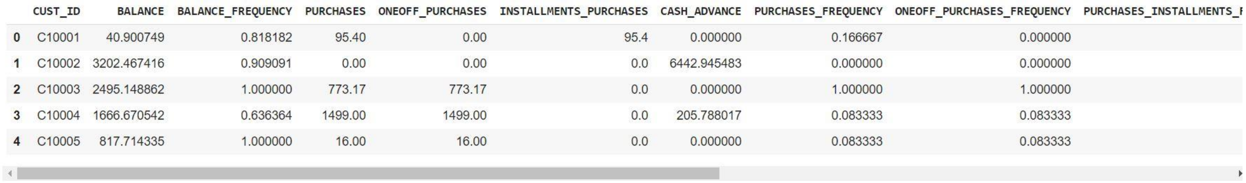
```
[ ]
    from google.colab import drive
    drive.mount('/content/gdrive')

    Mounted at /content/gdrive
```



```
[ ]
    df=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/data/Marketing_data.csv')
    df.head()
```

*Figure 6: read csv file*

We can use df.head() to see first 5 data records from dataset.



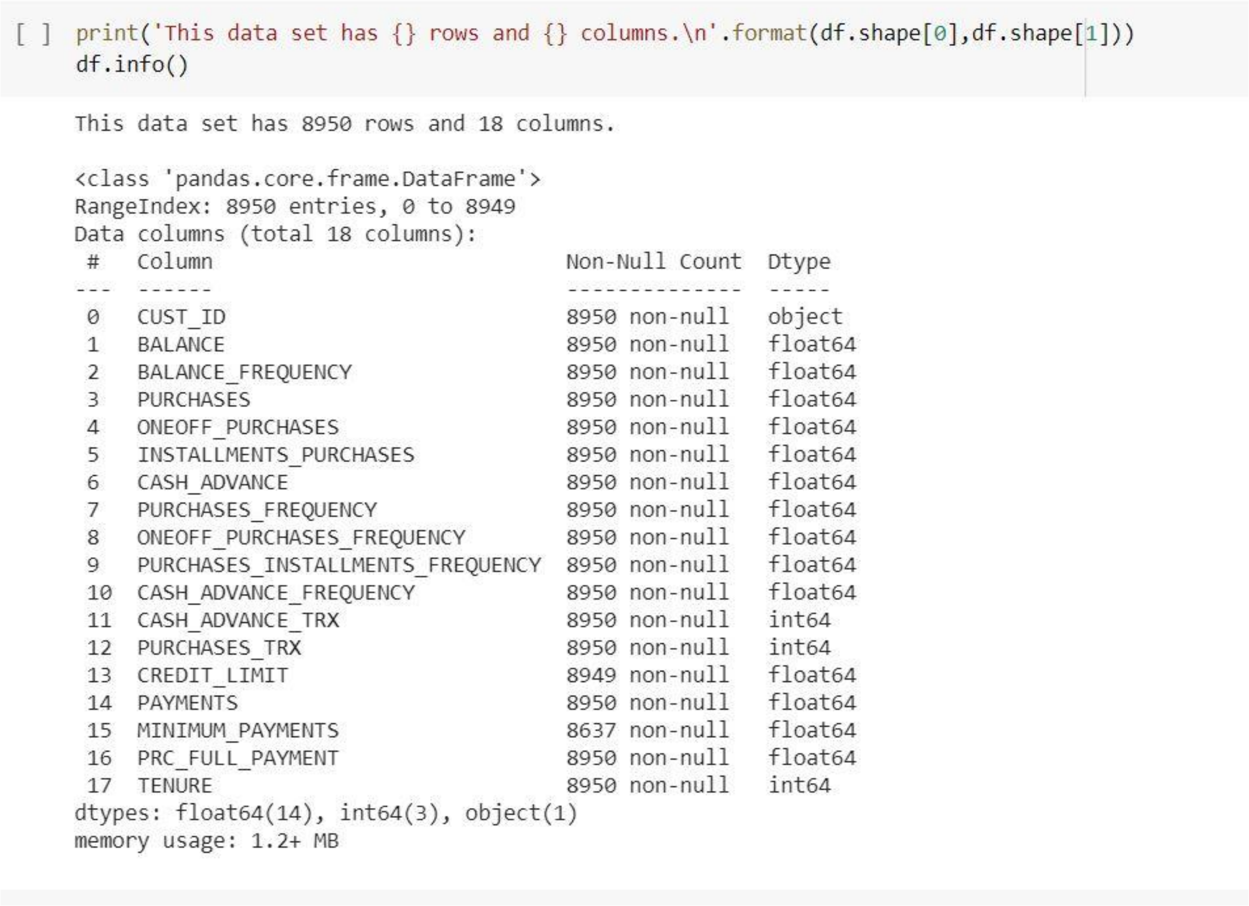
| | CUST_ID | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMENTS_PURCHASES | CASH_ADVANCE | PURCHASES_FREQUENCY | ONEOFF_PURCHASES_FREQUENCY | PURCHASES_INSTALLMENTS_F |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | C10001 | 40.900749 | 0.818182 | 95.40 | 0.00 | 95.4 | 0.000000 | 0.166667 | 0.000000 | |
| 1 | C10002 | 3202.467416 | 0.909091 | 0.00 | 0.00 | 0.0 | 6442.945483 | 0.000000 | 0.000000 | |
| 2 | C10003 | 2495.148862 | 1.000000 | 773.17 | 773.17 | 0.0 | 0.000000 | 1.000000 | 1.000000 | |
| 3 | C10004 | 1666.670542 | 0.636364 | 1499.00 | 1499.00 | 0.0 | 205.788017 | 0.083333 | 0.083333 | |
| 4 | C10005 | 817.714335 | 1.000000 | 16.00 | 16.00 | 0.0 | 0.000000 | 0.083333 | 0.083333 | |

*Figure 7:df.head()*

We can use following code to get information regarding dataset.

df.shape[0] gives number of rows and  df.shape[1] gives number of columns regarding dataset.

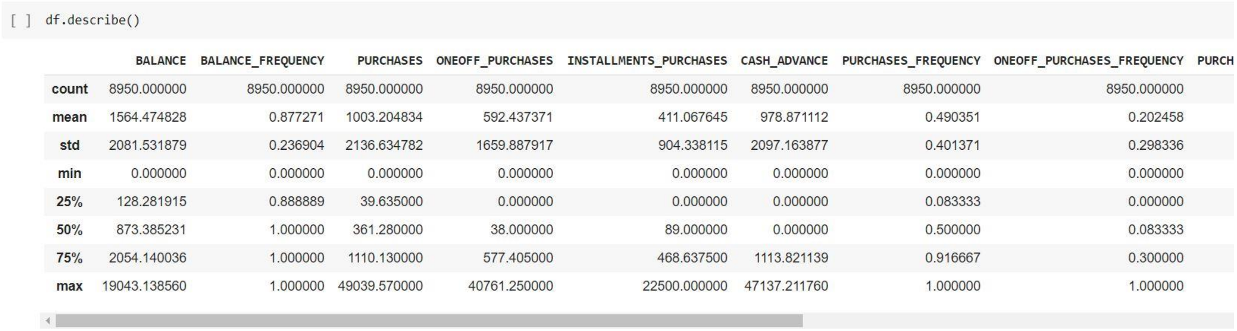
To get information regarding each column, we can use df.info () as follows.



```
[ ]  print('This data set has {} rows and {} columns.\n'.format(df.shape[0],df.shape[1]))
     df.info()
```

```
This data set has 8950 rows and 18 columns.

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   CUST_ID                           8950 non-null   object
 1   BALANCE                           8950 non-null   float64
 2   BALANCE_FREQUENCY                 8950 non-null   float64
 3   PURCHASES                         8950 non-null   float64
 4   ONEOFF_PURCHASES                  8950 non-null   float64
 5   INSTALLMENTS_PURCHASES            8950 non-null   float64
 6   CASH_ADVANCE                      8950 non-null   float64
 7   PURCHASES_FREQUENCY               8950 non-null   float64
 8   ONEOFF_PURCHASES_FREQUENCY        8950 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY  8950 non-null   float64
 10  CASH_ADVANCE_FREQUENCY            8950 non-null   float64
 11  CASH_ADVANCE_TRX                  8950 non-null   int64
 12  PURCHASES_TRX                     8950 non-null   int64
 13  CREDIT_LIMIT                      8949 non-null   float64
 14  PAYMENTS                          8950 non-null   float64
 15  MINIMUM_PAYMENTS                  8637 non-null   float64
 16  PRC_FULL_PAYMENT                  8950 non-null   float64
 17  TENURE                            8950 non-null   int64
dtypes: float64(14), int64(3), object(1)
memory usage: 1.2+ MB
```

*Figure 8:df.info()*



```
[ ] df.describe()
```

| | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMENTS_PURCHASES | CASH_ADVANCE | PURCHASES_FREQUENCY | ONEOFF_PURCHASES_FREQUENCY | PURCH |
|---|---|---|---|---|---|---|---|---|---|
| count | 8950.000000 | 8950.000000 | 8950.000000 | 8950.000000 | 8950.000000 | 8950.000000 | 8950.000000 | 8950.000000 | |
| mean | 1564.474828 | 0.877271 | 1003.204834 | 592.437371 | 411.067645 | 978.871112 | 0.490351 | 0.202458 | |
| std | 2081.531879 | 0.236904 | 2136.634782 | 1659.887917 | 904.338115 | 2097.163877 | 0.401371 | 0.298336 | |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | |
| 25% | 128.281915 | 0.888889 | 39.635000 | 0.000000 | 0.000000 | 0.000000 | 0.083333 | 0.000000 | |
| 50% | 873.385231 | 1.000000 | 361.280000 | 38.000000 | 89.000000 | 0.000000 | 0.500000 | 0.083333 | |
| 75% | 2054.140036 | 1.000000 | 1110.130000 | 577.405000 | 468.637500 | 1113.821139 | 0.916667 | 0.300000 | |
| max | 19043.138560 | 1.000000 | 49039.570000 | 40761.250000 | 22500.000000 | 47137.211760 | 1.000000 | 1.000000 | |

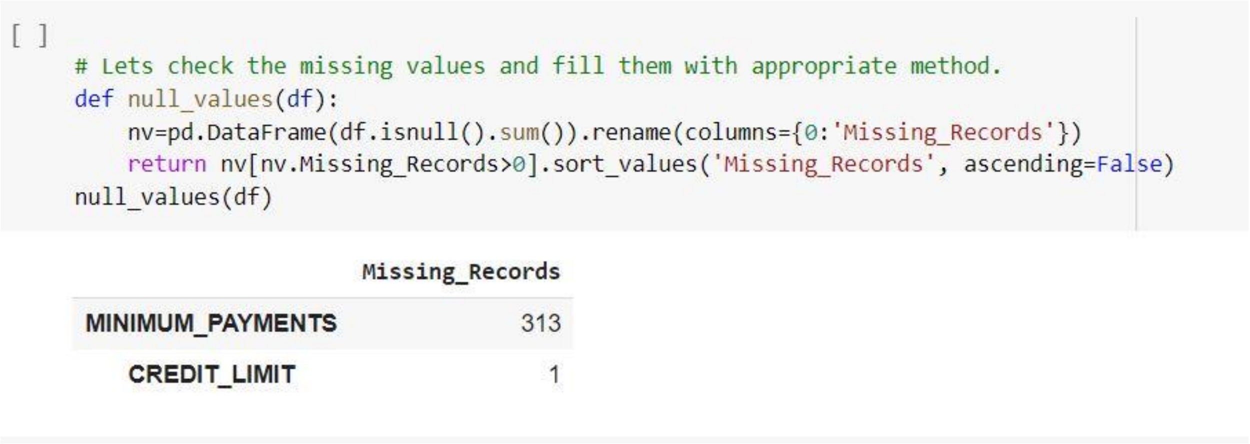
*Figure 9: df.describe()*

df.describe() use to get mathematical  information about each columns' data.

Ex: count, mean, std, min,25%,50%,75% of each column.

From now on, we have to prepare dataset for clustering. Before enter dataset as input to the clustering model, we have to clean the dataset. It means that we are fixing if there is any null values or errors.

Following code describes, if there is missing values or not in dataset,

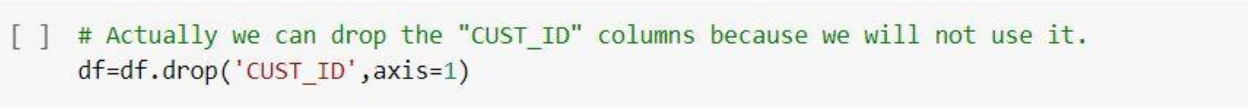


```
[ ]
    # Lets check the missing values and fill them with appropriate method.
    def null_values(df):
        nv=pd.DataFrame(df.isnull().sum()).rename(columns={0:'Missing_Records'})
        return nv[nv.Missing_Records>0].sort_values('Missing_Records', ascending=False)
    null_values(df)
```

| | Missing_Records |
|---|---|
| MINIMUM_PAYMENTS | 313 |
| CREDIT_LIMIT | 1 |

*Figure 10: remove null values*

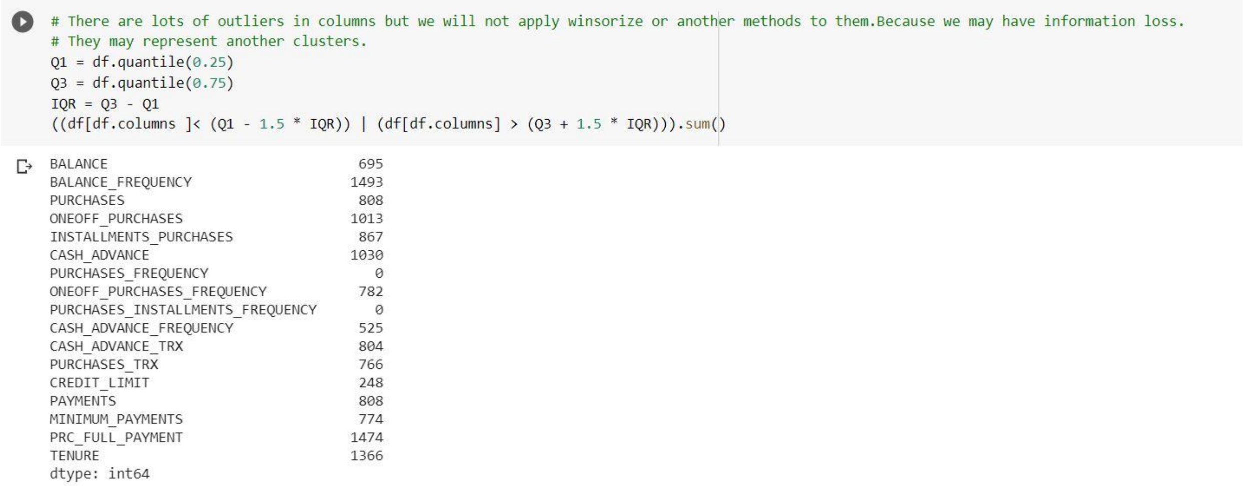
if we want to remove unnecessary columns from dataset, following code can used. Before use data set for clustering, we have to remove customer_Id column.



```
[ ]  # Actually we can drop the "CUST_ID" columns because we will not use it.
     df=df.drop('CUST_ID',axis=1)
```

*Figure 11: remove unnecessary columns*

There are lots of outliers in columns but we will not apply winsorize or other methods to them.

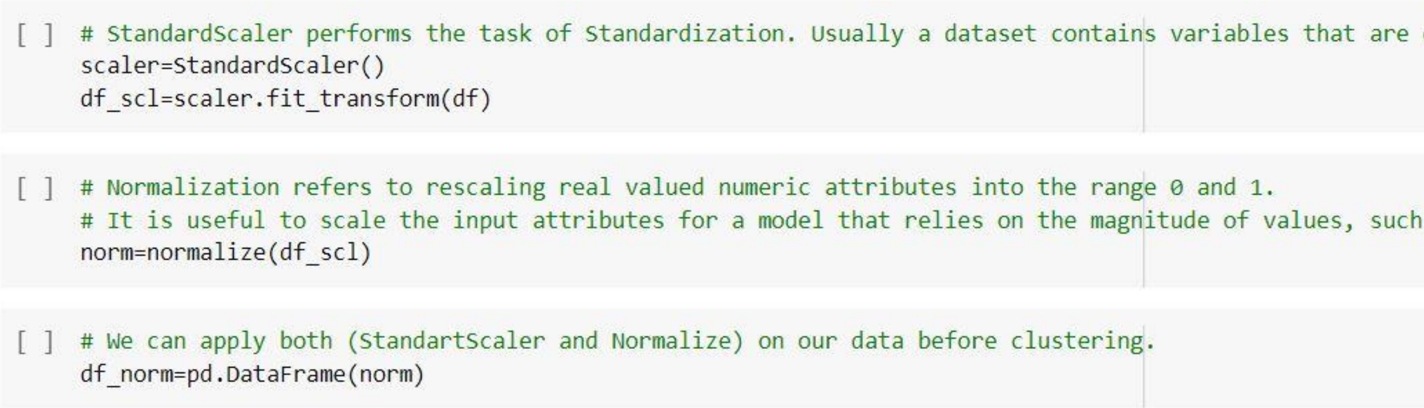


```
# There are lots of outliers in columns but we will not apply winsorize or another methods to them.Because we may have information loss.
# They may represent another clusters.
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
((df[df.columns ]< (Q1 - 1.5 * IQR)) | (df[df.columns] > (Q3 + 1.5 * IQR))).sum()
```

```
BALANCE                              695
BALANCE_FREQUENCY                   1493
PURCHASES                            808
ONEOFF_PURCHASES                    1013
INSTALLMENTS_PURCHASES               867
CASH_ADVANCE                        1030
PURCHASES_FREQUENCY                    0
ONEOFF_PURCHASES_FREQUENCY           782
PURCHASES_INSTALLMENTS_FREQUENCY       0
CASH_ADVANCE_FREQUENCY               525
CASH_ADVANCE_TRX                     804
PURCHASES_TRX                        766
CREDIT_LIMIT                         248
PAYMENTS                             808
MINIMUM_PAYMENTS                     774
PRC_FULL_PAYMENT                    1474
TENURE                              1366
dtype: int64
```

*Figure 12: outliers*

Because we may have information loss. They may represent another clusters.



```
[ ]  # StandardScaler performs the task of Standardization. Usually a dataset contains variables that are 
     scaler=StandardScaler()
     df_scl=scaler.fit_transform(df)
```

```
[ ]  # Normalization refers to rescaling real valued numeric attributes into the range 0 and 1.
     # It is useful to scale the input attributes for a model that relies on the magnitude of values, such
     norm=normalize(df_scl)
```

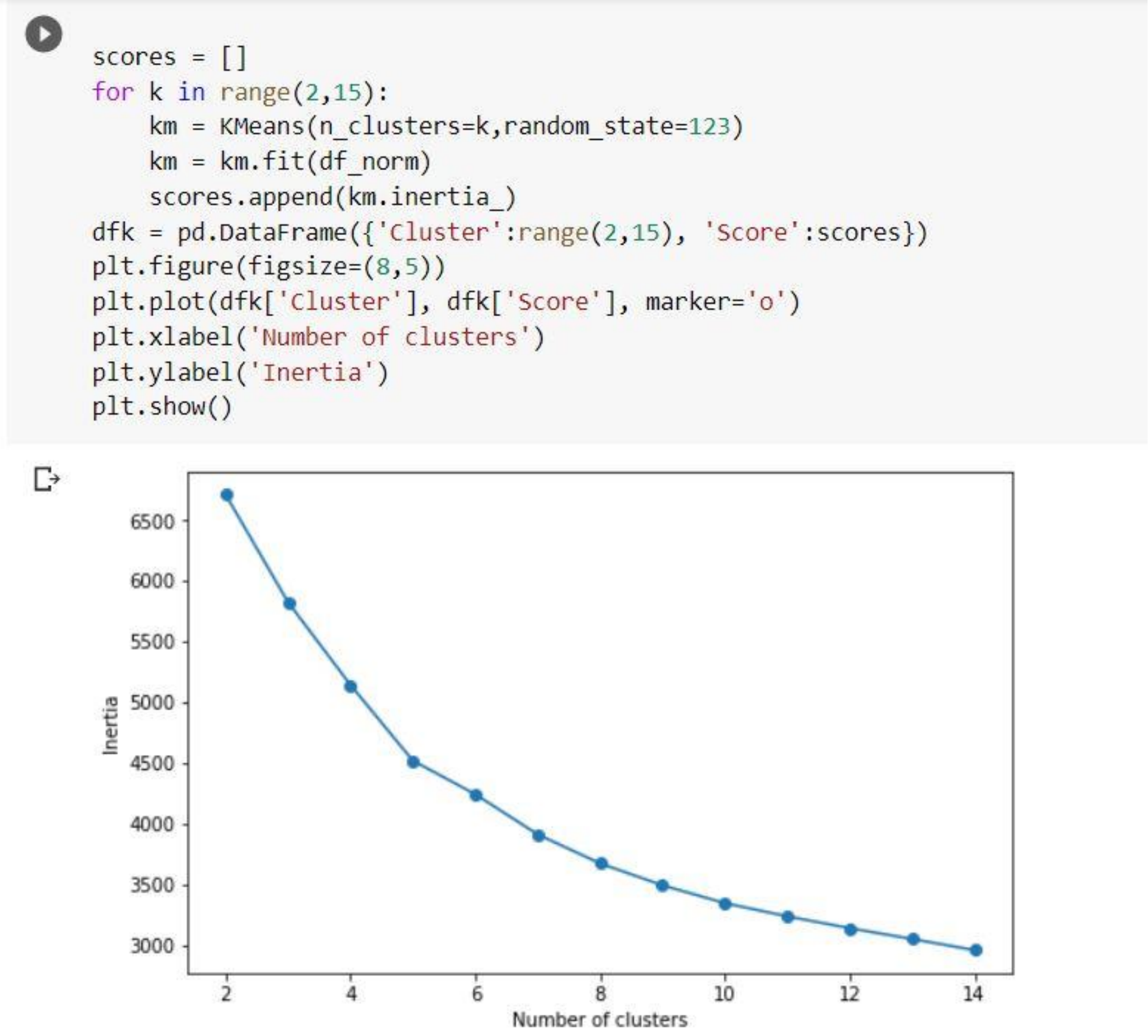
```
[ ]  # We can apply both (StandartScaler and Normalize) on our data before clustering.
     df_norm=pd.DataFrame(norm)
```

*Figure 13: clear dataset last*

Now our dataset is ready to use clustreing algorithm model.

### K-Means

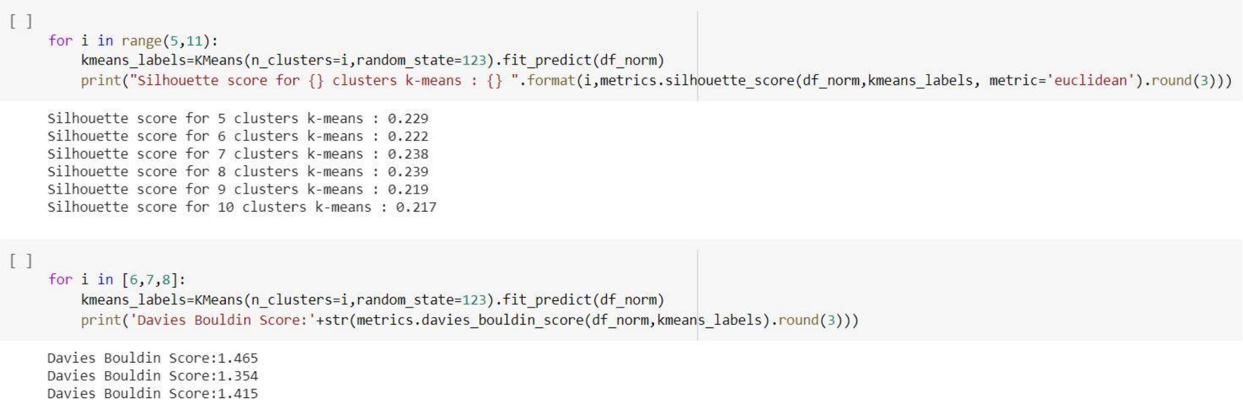
K Means Clustering is the most common and simplest Machine learning algorithm and it follows an iterative approach which attempts to partition the dataset into different “k” number of predefined and non-overlapping subgroups where each data point belongs to only one subgroup according to their similar qualities.



```
scores = []
for k in range(2,15):
    km = KMeans(n_clusters=k,random_state=123)
    km = km.fit(df_norm)
    scores.append(km.inertia_)
dfk = pd.DataFrame({'Cluster':range(2,15), 'Score':scores})
plt.figure(figsize=(8,5))
plt.plot(dfk['Cluster'], dfk['Score'], marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()
```

*Figure 14:kmeans elbow method*

The silhouette value measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation).



```
[ ]
    for i in range(5,11):
        kmeans_labels=KMeans(n_clusters=i,random_state=123).fit_predict(df_norm)
        print("Silhouette score for {} clusters k-means : {} ".format(i,metrics.silhouette_score(df_norm,kmeans_labels, metric='euclidean').round(3)))

    Silhouette score for 5 clusters k-means : 0.229
    Silhouette score for 6 clusters k-means : 0.222
    Silhouette score for 7 clusters k-means : 0.238
    Silhouette score for 8 clusters k-means : 0.239
    Silhouette score for 9 clusters k-means : 0.219
    Silhouette score for 10 clusters k-means : 0.217
```

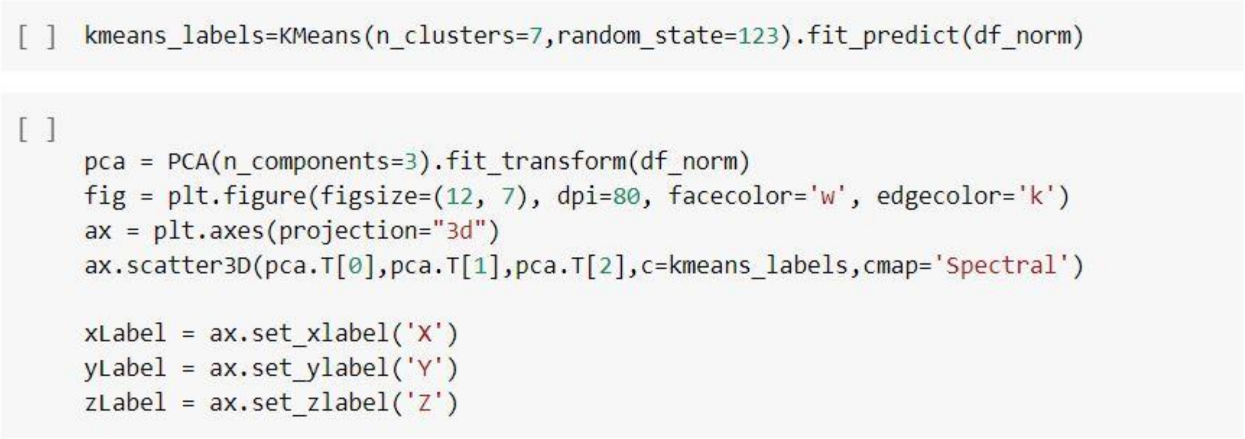
```
[ ]
    for i in [6,7,8]:
        kmeans_labels=KMeans(n_clusters=i,random_state=123).fit_predict(df_norm)
        print('Davies Bouldin Score:'+str(metrics.davies_bouldin_score(df_norm,kmeans_labels).round(3)))

    Davies Bouldin Score:1.465
    Davies Bouldin Score:1.354
    Davies Bouldin Score:1.415
```

*Figure 15: kmeans Davies Bouldin*

The values of silhouette score are close to each other in range 6 to 8. In the circumstances, Let's look at another metric. The metric is Davies Bouldin that is defined as the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances. The minimum score is zero, with lower values indicating better clustering.

Unlike Davies Bouldin, we want to be high of Silhouette score. Hence, when we evaluate both Elbow technique and Silhouette score, optimal cluster numbers are 7 according to K-Means Algorithm. So, I have determined 7 as the k values of the K-means model.

In [0]:



```
[ ] kmeans_labels=KMeans(n_clusters=7,random_state=123).fit_predict(df_norm)
```

```
[ ]
    pca = PCA(n_components=3).fit_transform(df_norm)
    fig = plt.figure(figsize=(12, 7), dpi=80, facecolor='w', edgecolor='k')
    ax = plt.axes(projection="3d")
    ax.scatter3D(pca.T[0],pca.T[1],pca.T[2],c=kmeans_labels,cmap='Spectral')

    xLabel = ax.set_xlabel('X')
    yLabel = ax.set_ylabel('Y')
    zLabel = ax.set_zlabel('Z')
```

*Figure 16:kmeans model clustering*

Now, Let's visualize "CC GENERAL" dataset in three-dimensional space. Hence, we should apply PCA before.

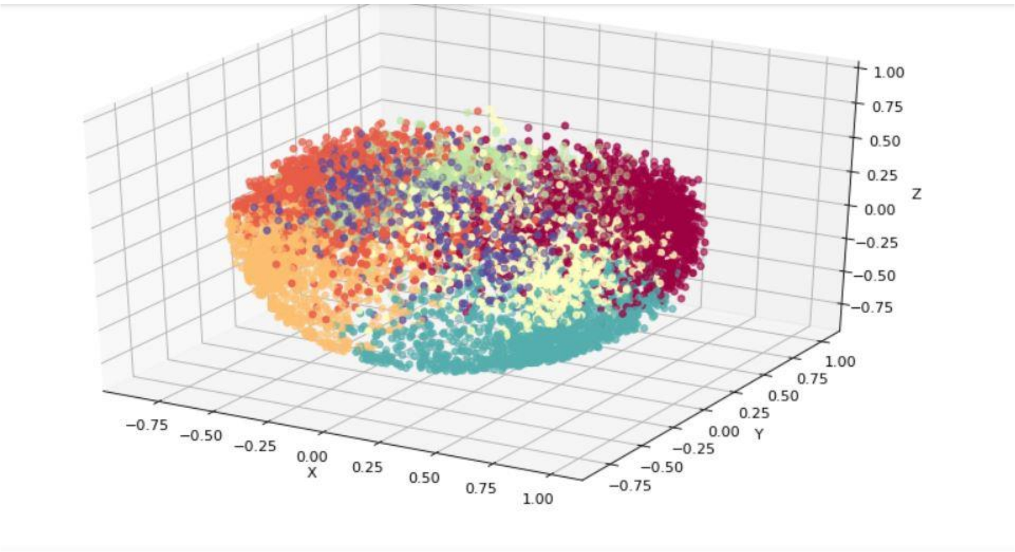
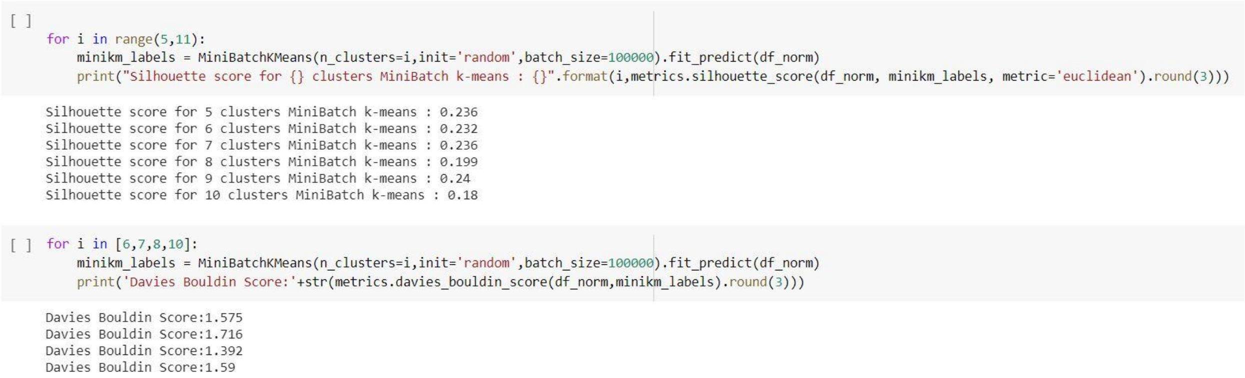


*Figure 17:kmeans diagram*

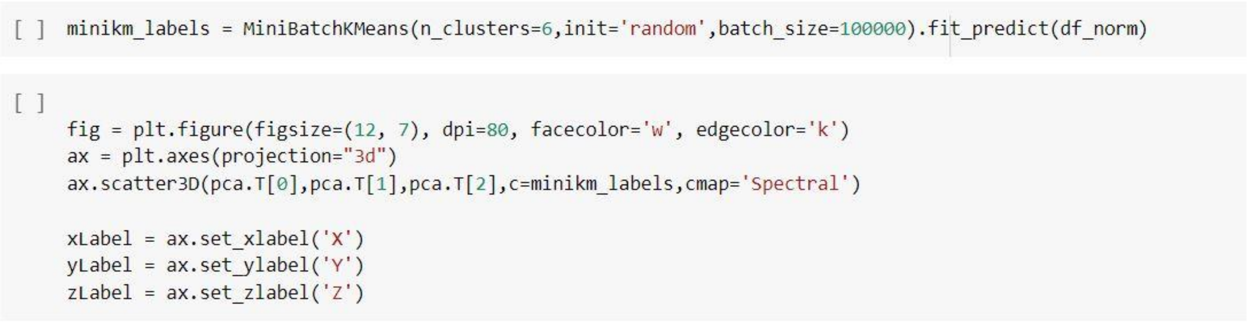
## MiniBatch K-Means



```
[ ]
    for i in range(5,11):
        minikm_labels = MiniBatchKMeans(n_clusters=i,init='random',batch_size=100000).fit_predict(df_norm)
        print("Silhouette score for {} clusters MiniBatch k-means : {}".format(i,metrics.silhouette_score(df_norm, minikm_labels, metric='euclidean').round(3)))

    Silhouette score for 5 clusters MiniBatch k-means : 0.236
    Silhouette score for 6 clusters MiniBatch k-means : 0.232
    Silhouette score for 7 clusters MiniBatch k-means : 0.236
    Silhouette score for 8 clusters MiniBatch k-means : 0.199
    Silhouette score for 9 clusters MiniBatch k-means : 0.24
    Silhouette score for 10 clusters MiniBatch k-means : 0.18
```

```
[ ] for i in [6,7,8,10]:
        minikm_labels = MiniBatchKMeans(n_clusters=i,init='random',batch_size=100000).fit_predict(df_norm)
        print('Davies Bouldin Score:'+str(metrics.davies_bouldin_score(df_norm,minikm_labels).round(3)))

    Davies Bouldin Score:1.575
    Davies Bouldin Score:1.716
    Davies Bouldin Score:1.392
    Davies Bouldin Score:1.59
```

*Figure 18: MiniBatch K-Means*



```
[ ]  minikm_labels = MiniBatchKMeans(n_clusters=6,init='random',batch_size=100000).fit_predict(df_norm)
```

```
[ ]
     fig = plt.figure(figsize=(12, 7), dpi=80, facecolor='w', edgecolor='k')
     ax = plt.axes(projection="3d")
     ax.scatter3D(pca.T[0],pca.T[1],pca.T[2],c=minikm_labels,cmap='Spectral')

     xLabel = ax.set_xlabel('X')
     yLabel = ax.set_ylabel('Y')
     zLabel = ax.set_zlabel('Z')
```

*Figure 19: MiniBatch K-Means diagram code*

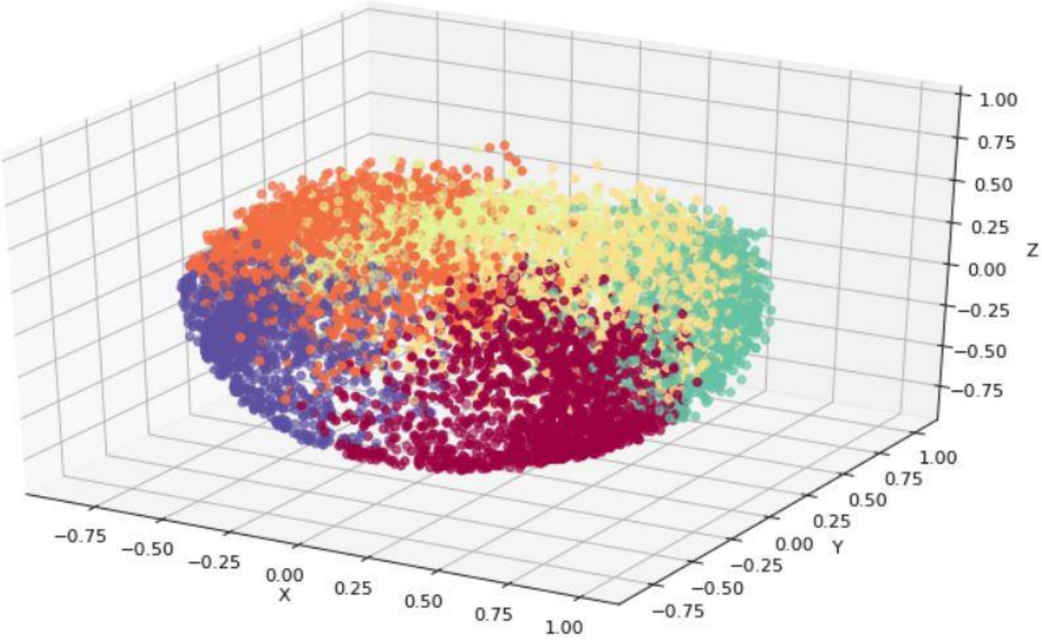
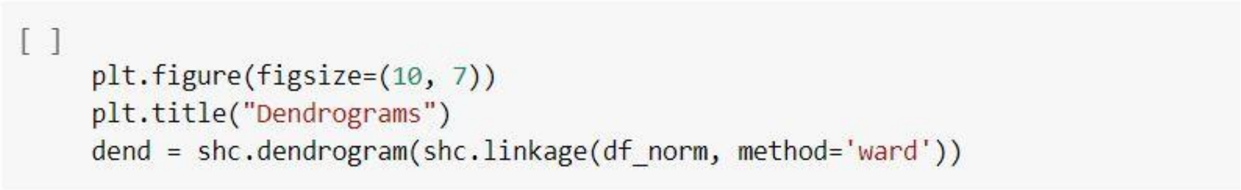


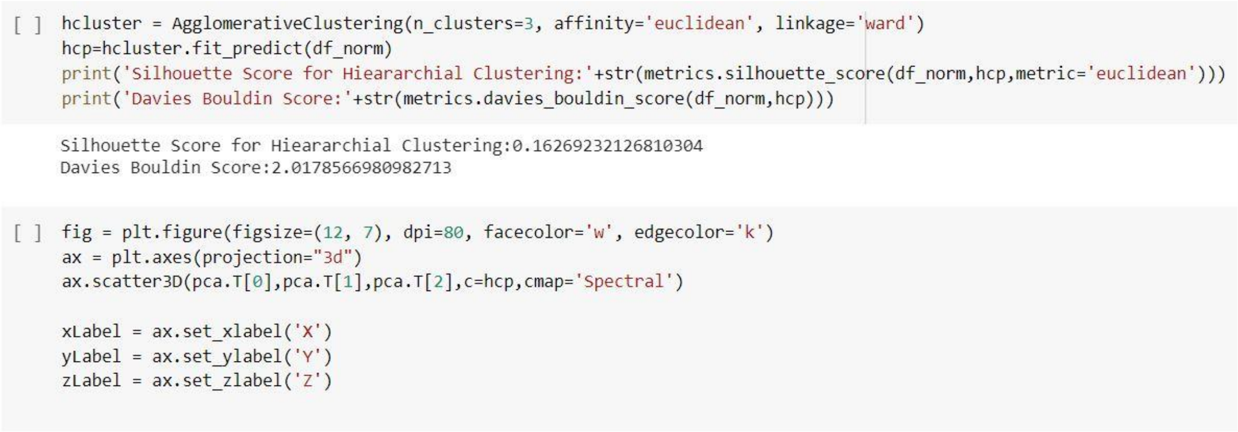
*Figure 20: MiniBatch K-Means*

## Hierarchical Clustering Segmentation



```
[ ]
    plt.figure(figsize=(10, 7))
    plt.title("Dendrograms")
    dend = shc.dendrogram(shc.linkage(df_norm, method='ward'))
```

*Figure 21:Hierarchical Clustering Segmentation1*



```
[ ] hcluster = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='ward')
    hcp=hcluster.fit_predict(df_norm)
    print('Silhouette Score for Hieararchial Clustering:'+str(metrics.silhouette_score(df_norm,hcp,metric='euclidean')))
    print('Davies Bouldin Score:'+str(metrics.davies_bouldin_score(df_norm,hcp)))
```

```
    Silhouette Score for Hieararchial Clustering:0.16269232126810304
    Davies Bouldin Score:2.0178566980982713
```

```
[ ] fig = plt.figure(figsize=(12, 7), dpi=80, facecolor='w', edgecolor='k')
    ax = plt.axes(projection="3d")
    ax.scatter3D(pca.T[0],pca.T[1],pca.T[2],c=hcp,cmap='Spectral')

    xLabel = ax.set_xlabel('X')
    yLabel = ax.set_ylabel('Y')
    zLabel = ax.set_zlabel('Z')
```

*Figure 22:Hierarchical Clustering Segmentation2*

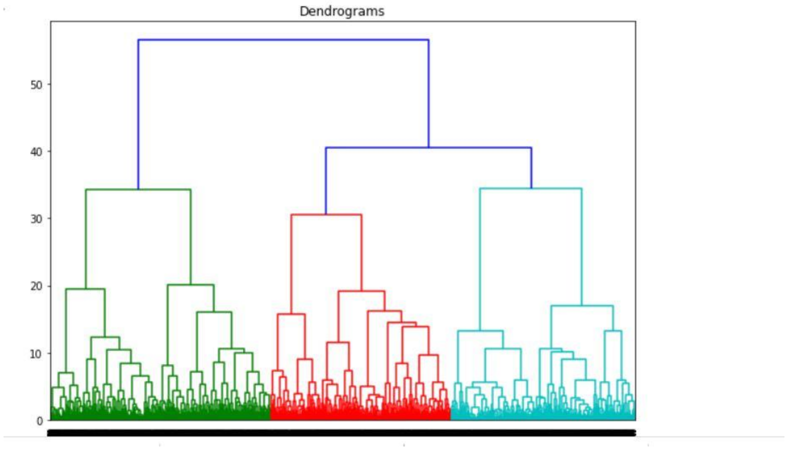
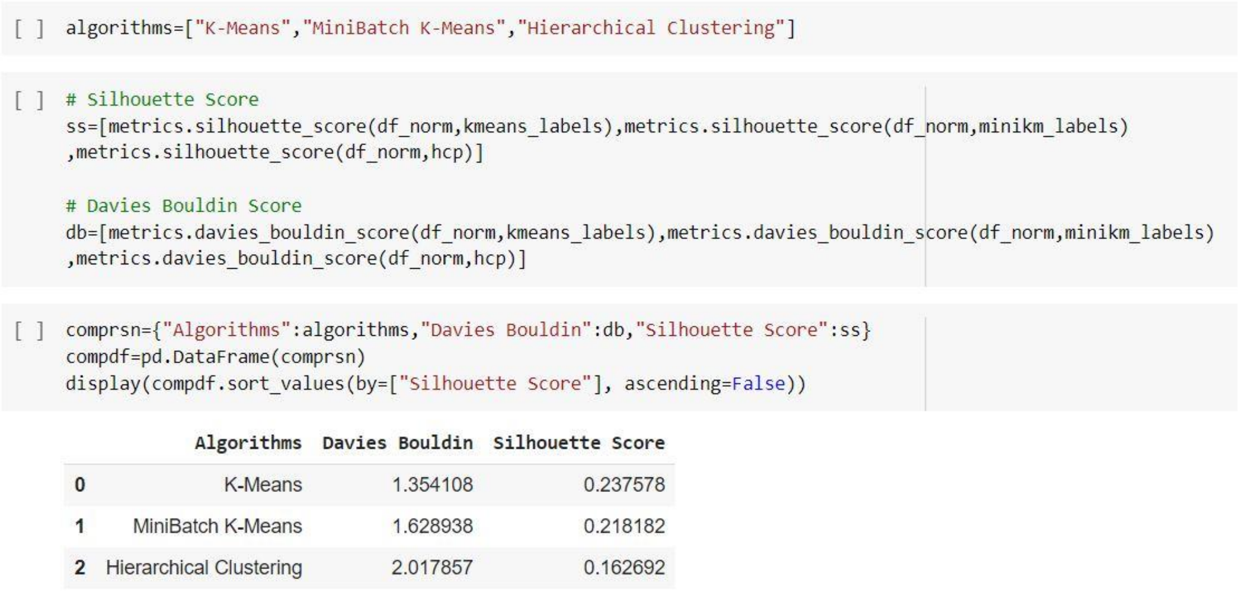


*Figure 23:Hierarchical Clustering Segmentation diagram*

## Comparison of Results



```
[ ]  algorithms=["K-Means","MiniBatch K-Means","Hierarchical Clustering"]
```

```
[ ]  # Silhouette Score
     ss=[metrics.silhouette_score(df_norm,kmeans_labels),metrics.silhouette_score(df_norm,minikm_labels)
     ,metrics.silhouette_score(df_norm,hcp)]

     # Davies Bouldin Score
     db=[metrics.davies_bouldin_score(df_norm,kmeans_labels),metrics.davies_bouldin_score(df_norm,minikm_labels)
     ,metrics.davies_bouldin_score(df_norm,hcp)]
```

```
[ ]  comprsn={"Algorithms":algorithms,"Davies Bouldin":db,"Silhouette Score":ss}
     compdf=pd.DataFrame(comprsn)
     display(compdf.sort_values(by=["Silhouette Score"], ascending=False))
```

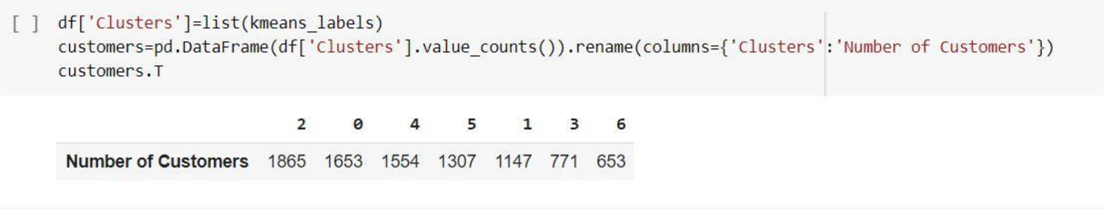
| | Algorithms | Davies Bouldin | Silhouette Score |
|---|---|---|---|
| **0** | K-Means | 1.354108 | 0.237578 |
| **1** | MiniBatch K-Means | 1.628938 | 0.218182 |
| **2** | Hierarchical Clustering | 2.017857 | 0.162692 |

*Figure 24:comparison of results*

Finally, we have tried three algorithms. K-Means has the best Silhouette and Hierarchical has best Davies Bouldin score.

However, we are going to 3 algorithms for make conclusion separately.

## K-Means model

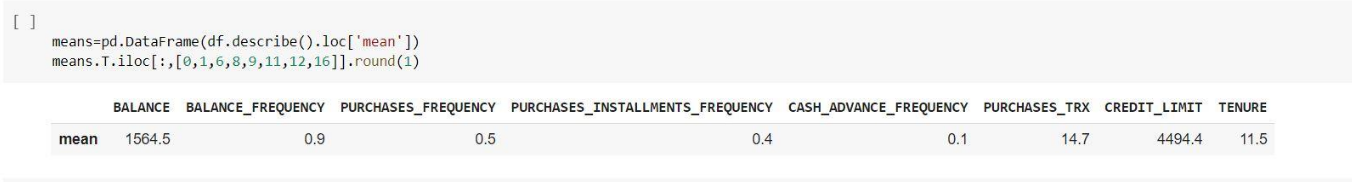


```
[ ] df['Clusters']=list(kmeans_labels)
    customers=pd.DataFrame(df['Clusters'].value_counts()).rename(columns={'Clusters':'Number of Customers'})
    customers.T
```

| | 2 | 0 | 4 | 5 | 1 | 3 | 6 |
|---|---|---|---|---|---|---|---|
| **Number of Customers** | 1865 | 1653 | 1554 | 1307 | 1147 | 771 | 653 |

*Figure 25:kmeans number of clusters*

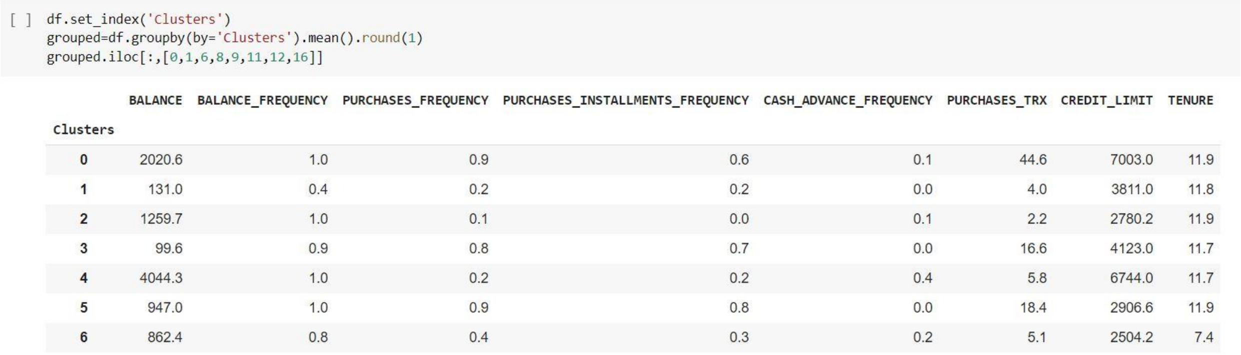
we have 7 customer types. Let’s try to understand behaviours or labels of customers.



```
[ ]
    means=pd.DataFrame(df.describe().loc['mean'])
    means.T.iloc[:,[0,1,6,8,9,11,12,16]].round(1)
```

| | BALANCE | BALANCE_FREQUENCY | PURCHASES_FREQUENCY | PURCHASES_INSTALLMENTS_FREQUENCY | CASH_ADVANCE_FREQUENCY | PURCHASES_TRX | CREDIT_LIMIT | TENURE |
|---|---|---|---|---|---|---|---|---|
| **mean** | 1564.5 | 0.9 | 0.5 | 0.4 | 0.1 | 14.7 | 4494.4 | 11.5 |

*Figure 26:understand behaviours or labels of customers.*



```
[ ] df.set_index('Clusters')
    grouped=df.groupby(by='Clusters').mean().round(1)
    grouped.iloc[:,[0,1,6,8,9,11,12,16]]
```

| | BALANCE | BALANCE_FREQUENCY | PURCHASES_FREQUENCY | PURCHASES_INSTALLMENTS_FREQUENCY | CASH_ADVANCE_FREQUENCY | PURCHASES_TRX | CREDIT_LIMIT | TENURE |
|---|---|---|---|---|---|---|---|---|
| **Clusters** | | | | | | | | |
| **0** | 2020.6 | 1.0 | 0.9 | 0.6 | 0.1 | 44.6 | 7003.0 | 11.9 |
| **1** | 131.0 | 0.4 | 0.2 | 0.2 | 0.0 | 4.0 | 3811.0 | 11.8 |
| **2** | 1259.7 | 1.0 | 0.1 | 0.0 | 0.1 | 2.2 | 2780.2 | 11.9 |
| **3** | 99.6 | 0.9 | 0.8 | 0.7 | 0.0 | 16.6 | 4123.0 | 11.7 |
| **4** | 4044.3 | 1.0 | 0.2 | 0.2 | 0.4 | 5.8 | 6744.0 | 11.7 |
| **5** | 947.0 | 1.0 | 0.9 | 0.8 | 0.0 | 18.4 | 2906.6 | 11.9 |
| **6** | 862.4 | 0.8 | 0.4 | 0.3 | 0.2 | 5.1 | 2504.2 | 7.4 |

*Figure 27:kmeans values fo clusters*

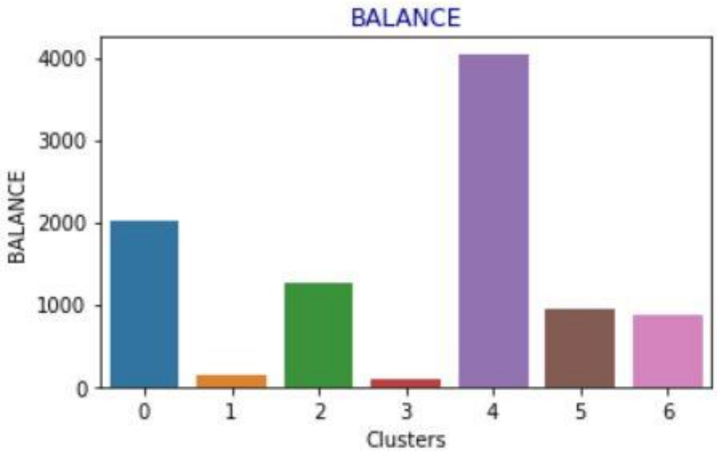


*Figure 28:kmeans balance*

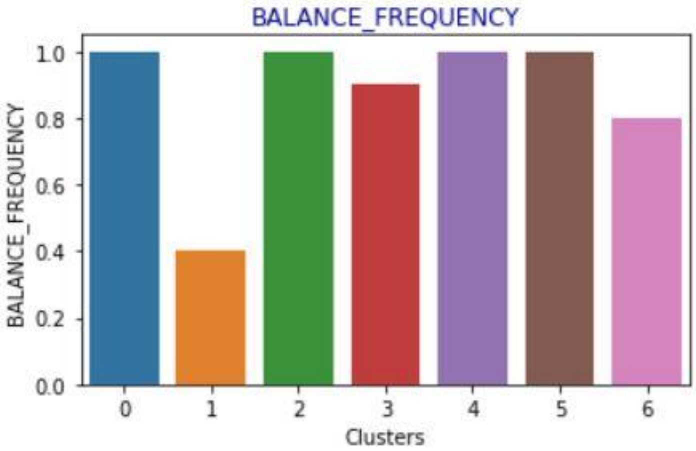


*Figure 29:kmeans  balance frequency*

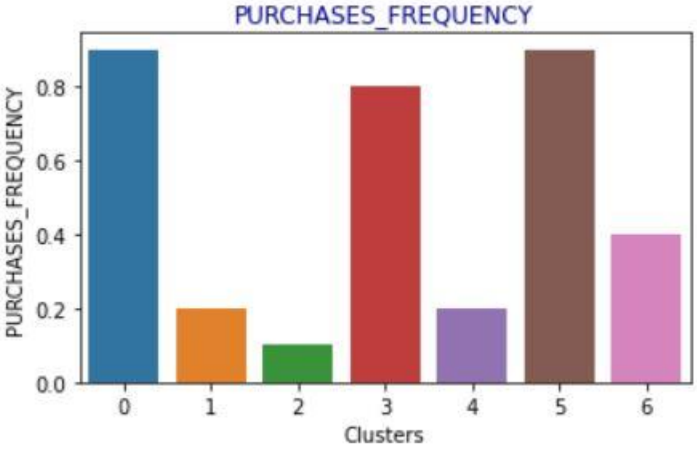


*Figure 30:kmeans  purchases frequency*

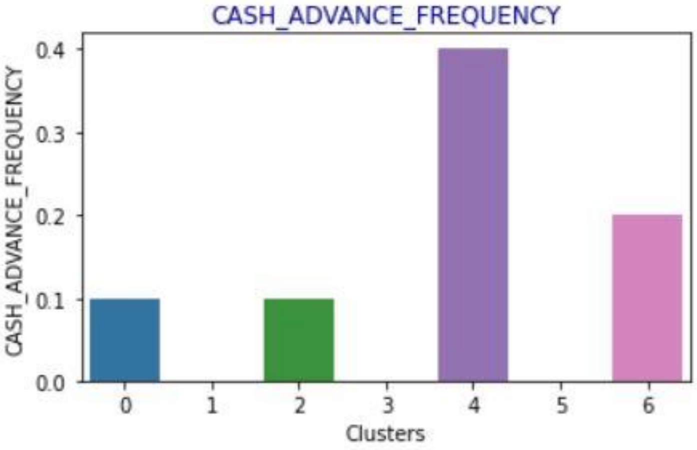


*Figure 31: kmeans cash advance frequency*

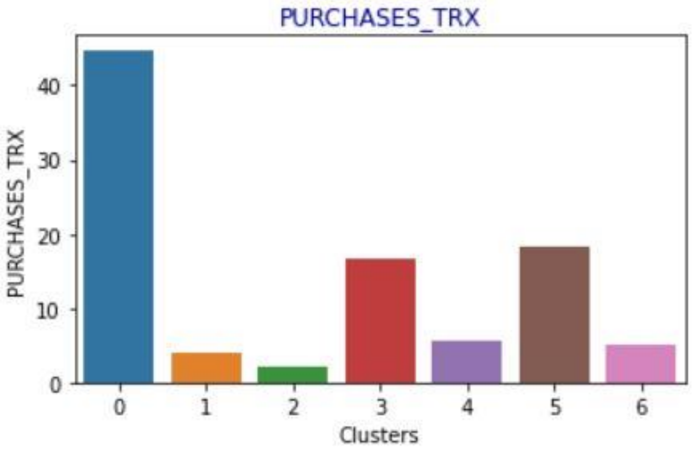


*Figure 32:kmeans purchases_trx*

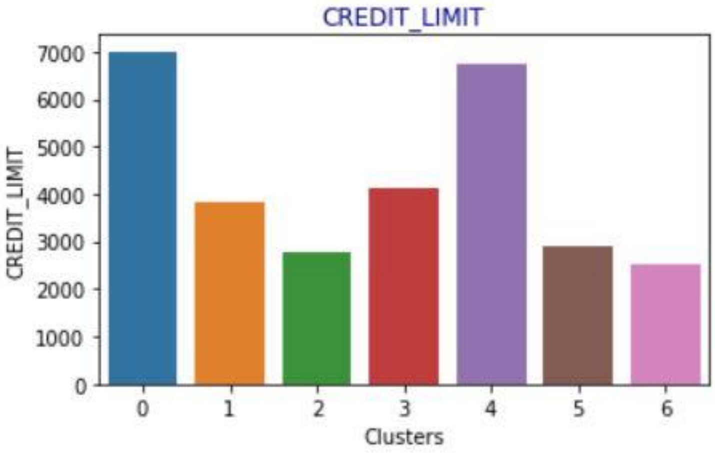


*Figure 33:kmeans credit limit*

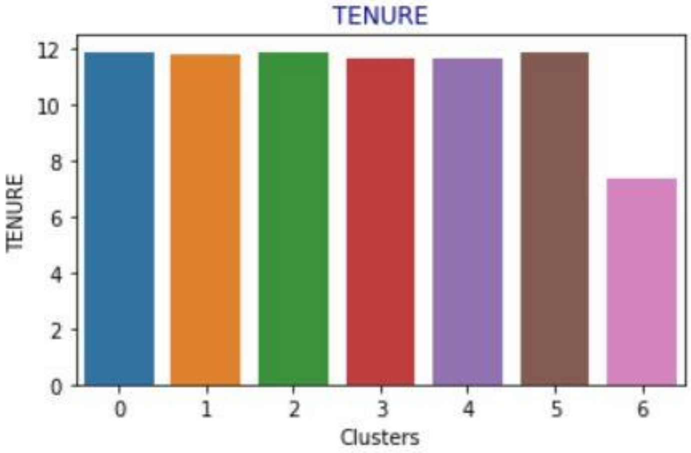


*Figure 34:kmeans tenure*

We have chosen some columns that are significant to identify the clusters.

### K-Means model conclusion

Cluster 0: The highest purchase frequency which tend to pay in instalment, that is higher credit limit and long duration customers.

Cluster 1: Pretty low balance and purchase frequency. They rarely use credit card and also, they have lower credit limit.

Cluster 2: This group is having the highest number of customers and lowest usage of cards. Inactive customers, also long duration customers.

Cluster 3: High tendency of payment instalment, higher purchase frequency and their tenure time is above average.

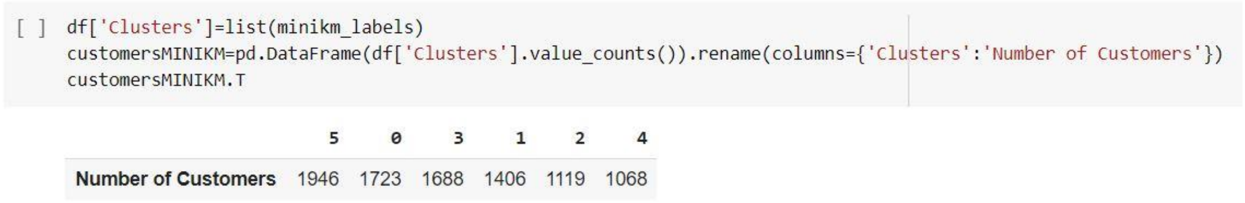
Cluster 4: The highest balance amount but purchase frequency is not that good. Tend to cash in advance, higher credit limit than others. They don't like spending money.

Cluster 5: Second highest purchase frequency and also higher tendency payment in instalment. They are long duration customers.

Cluster 6: The least quantity of customer is in this group which are below average of purchase frequency and a shortly duration customers.

Firstly, we have started with data pre-processing. Then, we applied clustering algorithms. After comparing these clustering models than, we decided to use K-Means as the first model. Then, we divided the data into seven clusters, because seven clusters can be easily used to determine the behaviours of customers. However, each of the clusters have their own characteristics.

### MiniBatch K-Means model

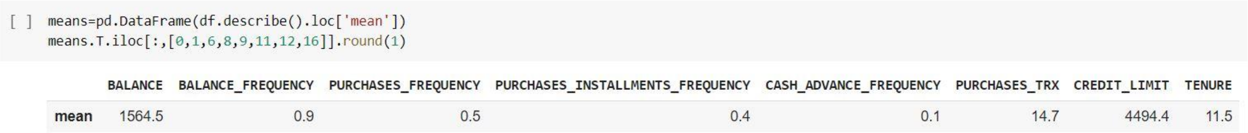


```
[ ]  df['Clusters']=list(minikm_labels)
     customersMINIKM=pd.DataFrame(df['Clusters'].value_counts()).rename(columns={'Clusters':'Number of Customers'})
     customersMINIKM.T
```

| | 5 | 0 | 3 | 1 | 2 | 4 |
|---|---|---|---|---|---|---|
| **Number of Customers** | 1946 | 1723 | 1688 | 1406 | 1119 | 1068 |

*Figure 35:MiniBatch K-Means  number of clusters*

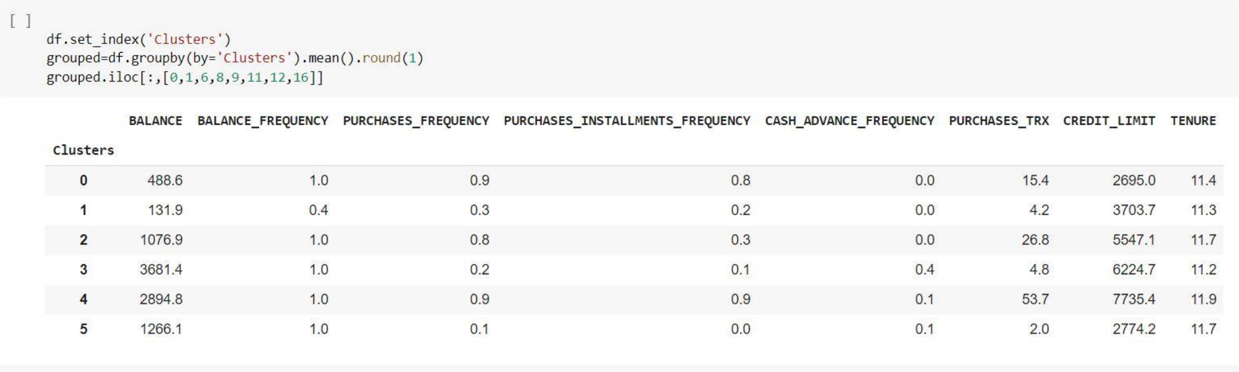
we have 6 customer types. Let's try to understand behaviours or labels of customers.



```
[ ] means=pd.DataFrame(df.describe().loc['mean'])
    means.T.iloc[:,[0,1,6,8,9,11,12,16]].round(1)
```

| | BALANCE | BALANCE_FREQUENCY | PURCHASES_FREQUENCY | PURCHASES_INSTALLMENTS_FREQUENCY | CASH_ADVANCE_FREQUENCY | PURCHASES_TRX | CREDIT_LIMIT | TENURE |
|---|---|---|---|---|---|---|---|---|
| **mean** | 1564.5 | 0.9 | 0.5 | 0.4 | 0.1 | 14.7 | 4494.4 | 11.5 |

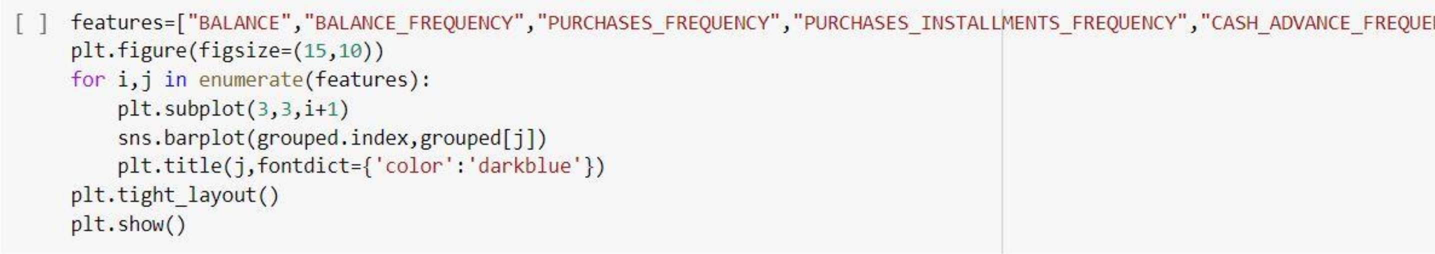
*Figure 36:MiniBatch K-Means cluster mean*



```
[ ]
    df.set_index('Clusters')
    grouped=df.groupby(by='Clusters').mean().round(1)
    grouped.iloc[:,[0,1,6,8,9,11,12,16]]
```

| | BALANCE | BALANCE_FREQUENCY | PURCHASES_FREQUENCY | PURCHASES_INSTALLMENTS_FREQUENCY | CASH_ADVANCE_FREQUENCY | PURCHASES_TRX | CREDIT_LIMIT | TENURE |
|---|---|---|---|---|---|---|---|---|
| **Clusters** | | | | | | | | |
| **0** | 488.6 | 1.0 | 0.9 | 0.8 | 0.0 | 15.4 | 2695.0 | 11.4 |
| **1** | 131.9 | 0.4 | 0.3 | 0.2 | 0.0 | 4.2 | 3703.7 | 11.3 |
| **2** | 1076.9 | 1.0 | 0.8 | 0.3 | 0.0 | 26.8 | 5547.1 | 11.7 |
| **3** | 3681.4 | 1.0 | 0.2 | 0.1 | 0.4 | 4.8 | 6224.7 | 11.2 |
| **4** | 2894.8 | 1.0 | 0.9 | 0.9 | 0.1 | 53.7 | 7735.4 | 11.9 |
| **5** | 1266.1 | 1.0 | 0.1 | 0.0 | 0.1 | 2.0 | 2774.2 | 11.7 |

*Figure 37:MiniBatch K-Means cluster characteristics*



```
[ ] features=["BALANCE","BALANCE_FREQUENCY","PURCHASES_FREQUENCY","PURCHASES_INSTALLMENTS_FREQUENCY","CASH_ADVANCE_FREQUE
    plt.figure(figsize=(15,10))
    for i,j in enumerate(features):
        plt.subplot(3,3,i+1)
        sns.barplot(grouped.index,grouped[j])
        plt.title(j,fontdict={'color':'darkblue'})
    plt.tight_layout()
    plt.show()
```

*Figure 38: label generating code*

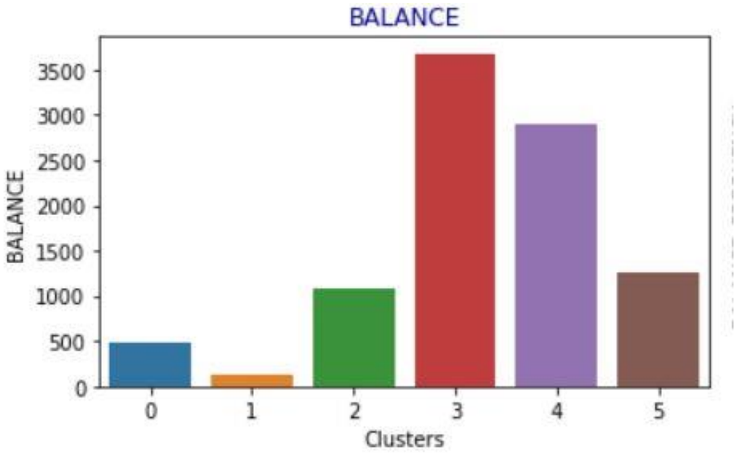


*Figure 39:MiniBatch K-Means balance*

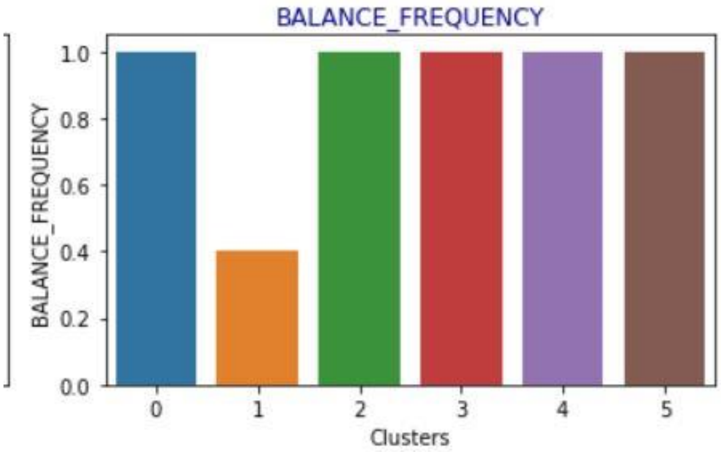


*Figure 40:MiniBatch K-Means balance frequency*

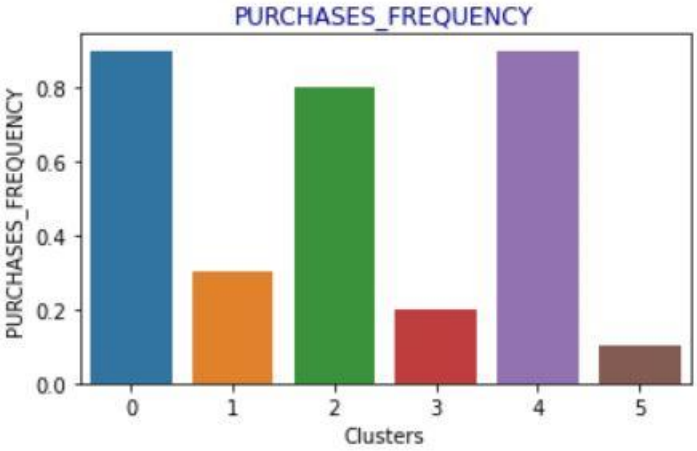


*Figure 41:MiniBatch K-Means purchases frequency*

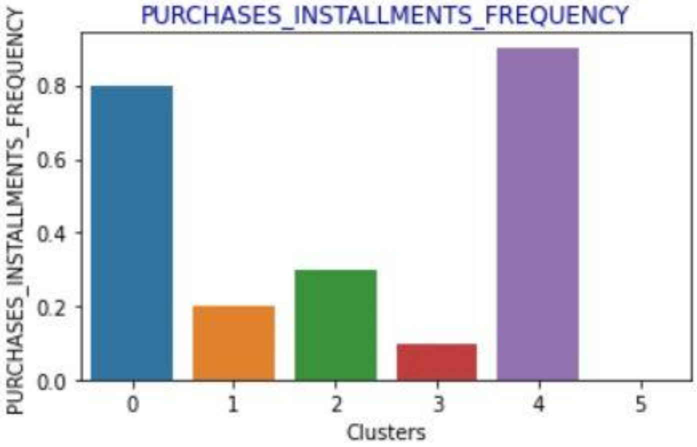


*Figure 42:MiniBatch K-Means purchase installment frequency*

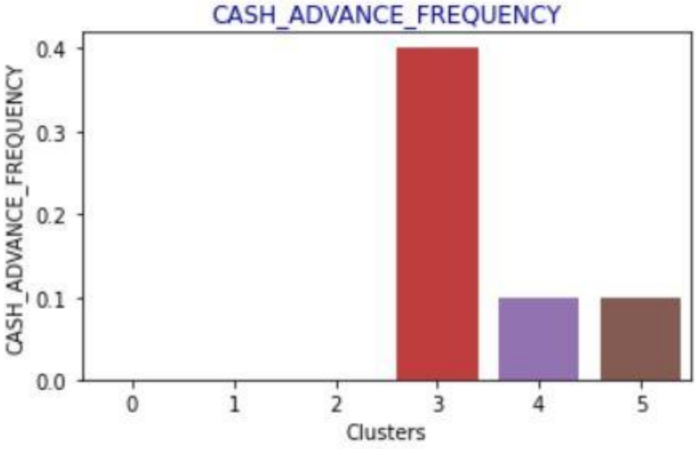


*Figure 43:MiniBatch K-Means cash advance frequency*

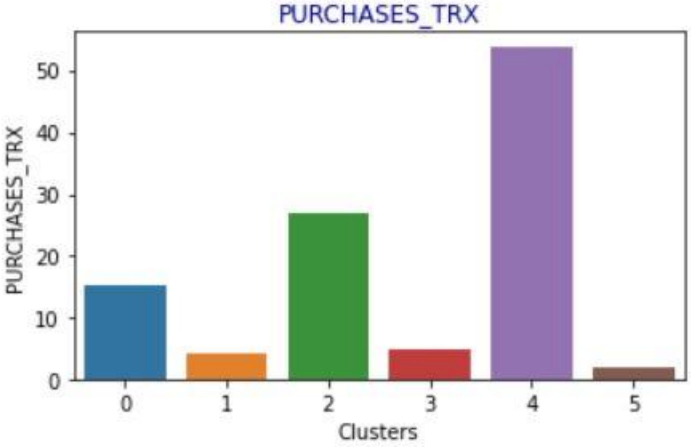


*Figure 44:MiniBatch K-Means purchases frequency*

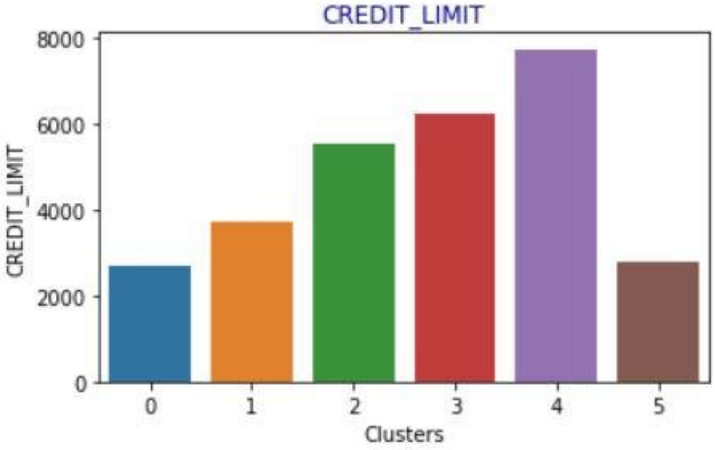


*Figure 45:MiniBatch K-Means credit limit*

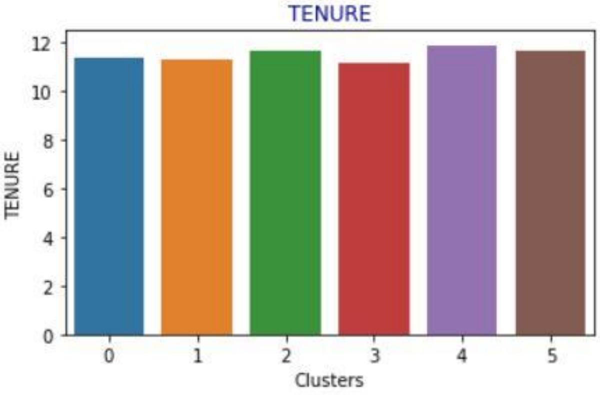


*Figure 46:MiniBatch K-Means tenure*

We have chosen some columns that are significant to identify the clusters.

**MiniBatch K-Means model conclusion**

Cluster 0: Second highest purchase frequency which tend to pay in instalment, that is lower credit limit and long duration customers.

Cluster 1: Pretty low balance and purchase frequency. They rarely use credit card and also, they have lower credit limit.

Cluster 2: This group is having the highest number of customers and lowest usage of cards. Inactive customers, also long duration customers.

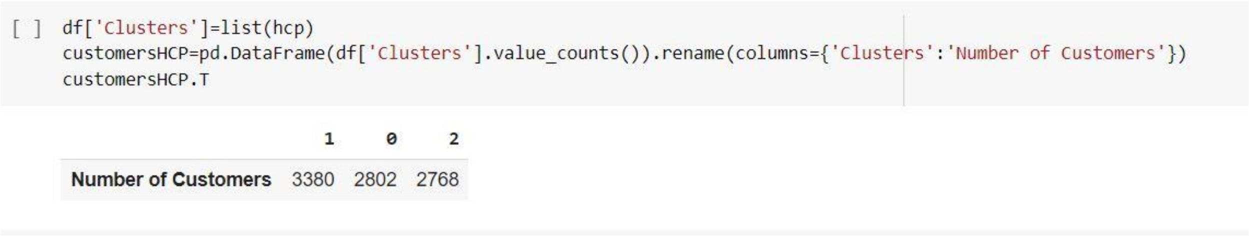
Cluster 3: High tendency of payment instalment, higher purchase frequency and their tenure time is above average.

Cluster 4: The highest balance amount but purchase frequency is not that good. Tend to cash in advance, higher credit limit than others. They don’t like spending money.

Cluster 5: Second highest purchase frequency and also higher tendency payment in instalment. They are long duration customers.

Firstly, we have started with data pre-processing. Then, we applied clustering algorithms. After comparing these clustering models than, we decided to use MiniBatch K-Means as the second model. Then, we divided the data into six clusters, because six clusters can be easily used to determine the behaviours of customers. However, each of the clusters have their own characteristics.

**Hierarchical Clustering Segmentation model**

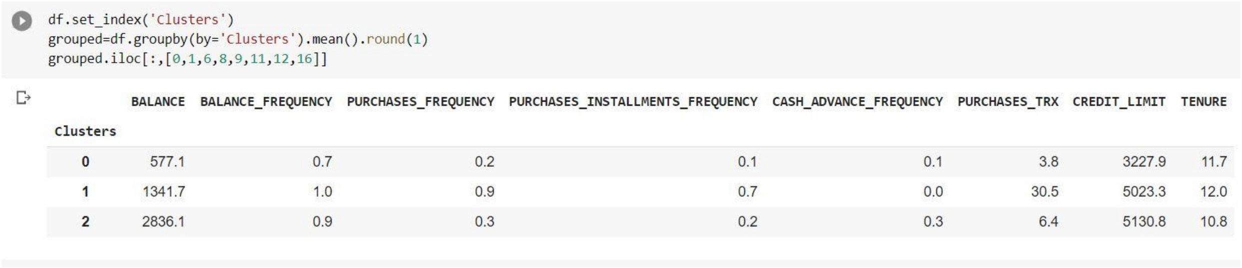


```
df['Clusters']=list(hcp)
customersHCP=pd.DataFrame(df['Clusters'].value_counts()).rename(columns={'Clusters':'Number of Customers'})
customersHCP.T
```

| | 1 | 0 | 2 |
|---|---|---|---|
| **Number of Customers** | 3380 | 2802 | 2768 |

*Figure 47:Hierarchical Clustering Segmentation model clustering*

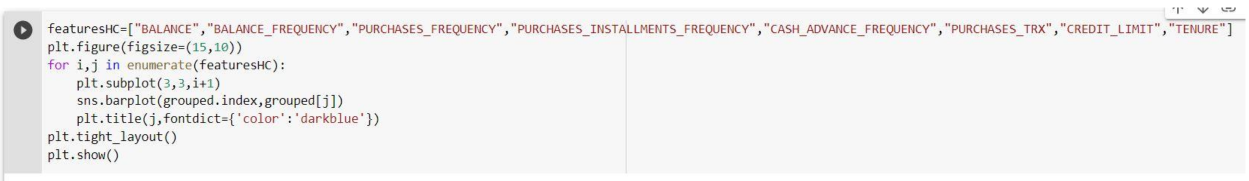
we have 3 customer types. Let’s try to understand behaviours or labels of customers.



```
df.set_index('Clusters')
grouped=df.groupby(by='Clusters').mean().round(1)
grouped.iloc[:,[0,1,6,8,9,11,12,16]]
```

| | BALANCE | BALANCE_FREQUENCY | PURCHASES_FREQUENCY | PURCHASES_INSTALLMENTS_FREQUENCY | CASH_ADVANCE_FREQUENCY | PURCHASES_TRX | CREDIT_LIMIT | TENURE |
|---|---|---|---|---|---|---|---|---|
| **Clusters** | | | | | | | | |
| **0** | 577.1 | 0.7 | 0.2 | 0.1 | 0.1 | 3.8 | 3227.9 | 11.7 |
| **1** | 1341.7 | 1.0 | 0.9 | 0.7 | 0.0 | 30.5 | 5023.3 | 12.0 |
| **2** | 2836.1 | 0.9 | 0.3 | 0.2 | 0.3 | 6.4 | 5130.8 | 10.8 |

*Figure 48:Hierarchical Clustering Segmentation model characteristics*



```
featuresHC=["BALANCE","BALANCE_FREQUENCY","PURCHASES_FREQUENCY","PURCHASES_INSTALLMENTS_FREQUENCY","CASH_ADVANCE_FREQUENCY","PURCHASES_TRX","CREDIT_LIMIT","TENURE"]
plt.figure(figsize=(15,10))
for i,j in enumerate(featuresHC):
    plt.subplot(3,3,i+1)
    sns.barplot(grouped.index,grouped[j])
    plt.title(j,fontdict={'color':'darkblue'})
plt.tight_layout()
plt.show()
```

*Figure 49:Hierarchical Clustering Segmentation model label diagram*

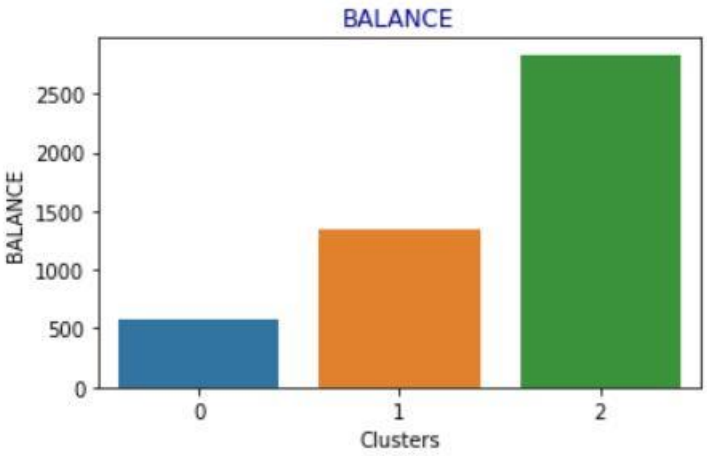


*Figure 50:Hierarchical Clustering Segmentation model balance*

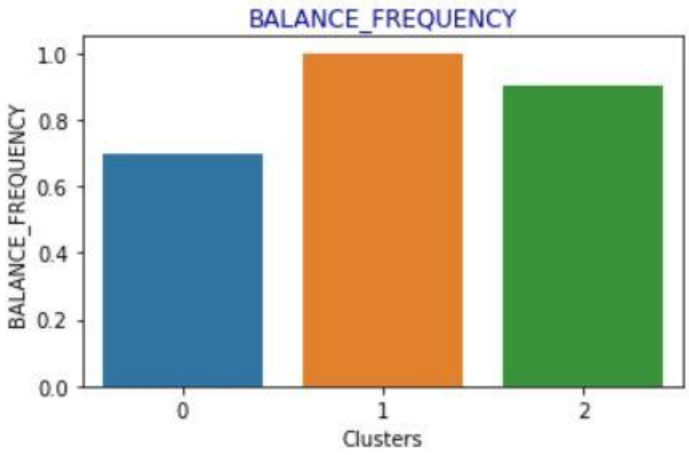


*Figure 51:Hierarchical Clustering Segmentation model balance frequency*

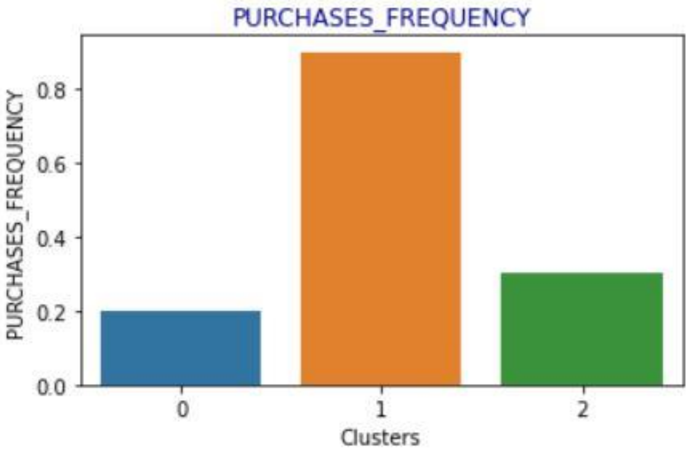


*Figure 52:Hierarchical Clustering Segmentation model purchases frequency*

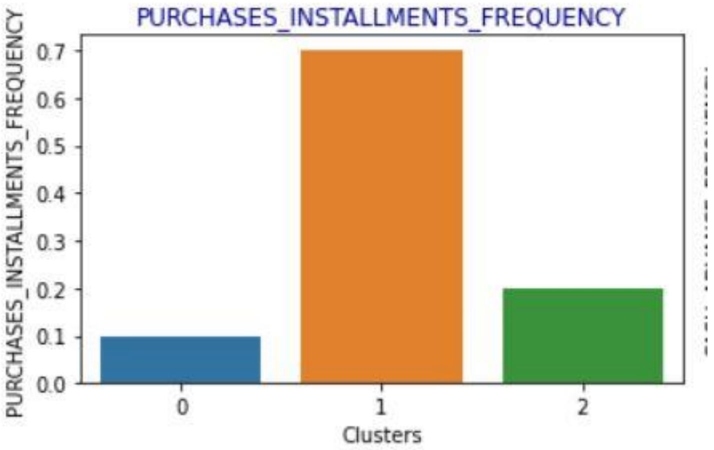


*Figure 53:Hierarchical Clustering Segmentation model purchases instalment frequency*

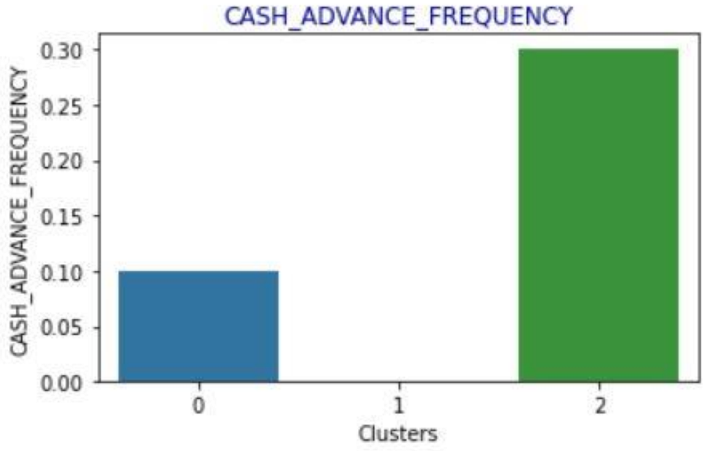


*Figure 54:Hierarchical Clustering Segmentation model cash advance frequency*

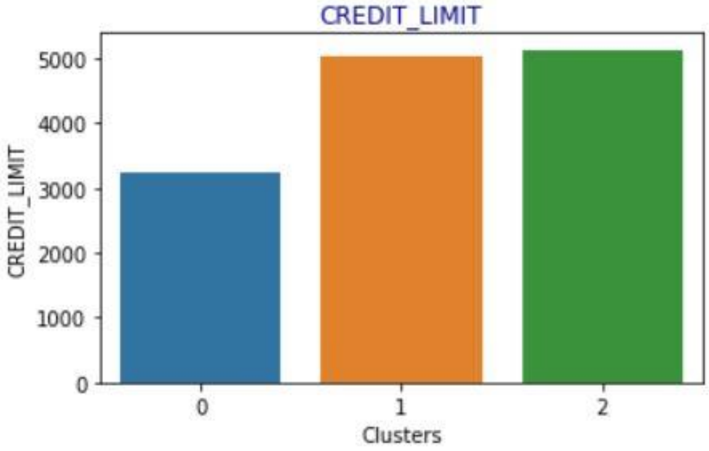


*Figure 55:Hierarchical Clustering Segmentation model credit limit*

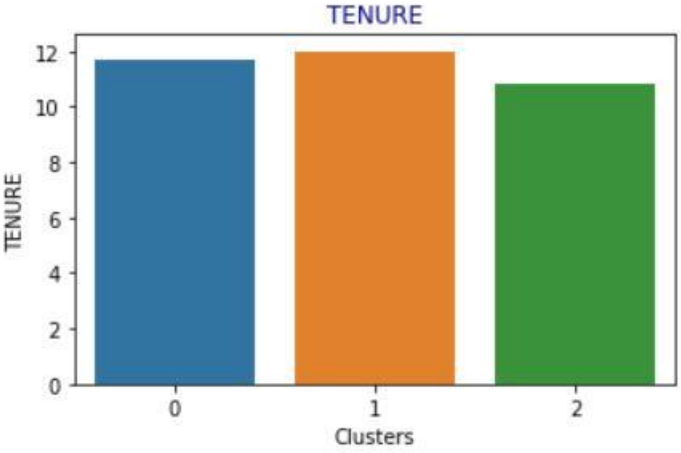


*Figure 56:Hierarchical Clustering Segmentation model tenure*

We have chosen some columns that are significant to identify the clusters.

**Hierarchical Clustering Segmentation model conclusion**
Cluster 0: The lowest purchase frequency which tend to pay in instalment, that is one of lower credit limit and long duration customers.

Cluster 1: The highest purchase frequency which tend to pay in instalment mostly, that is one of higher credit limit and long duration customers.

Cluster 2: This group is having the highest balance and second  highest purchase frequency .

## References

[1] D. P. Yash Kushwaha, “Customer Segmentation using K-Means Algorithm,” 8th Semester Student of B.tech in Computer Science and Engineering.

[2] E. A. Onur DOĞAN1, “CUSTOMER SEGMENTATION BY USING RFM”.

[3] C. M. S. R. a. K. V. N. T. Sajana, “A Survey on,” in *Indian Journal of Science and Technology, Volume 9, Issue 3,,* Jan 2016..

[4] A. B. P. E. Shreya Tripathi, “Approaches to,” in *International Journal of Engineering and Technology, Volume 7,* 2018.

[5] A. R. Azarnoush Ansari, ““Customer Clustering Using a,” in *International Journal of Business and Management, Volume 11, Issue 7,,* 2016.

[6] “mini-batch-k-means-clustering-algorithm,” [Online]. Available: https://www.geeksforgeeks.org/ml-mini-batch-k-means-clustering-algorithm/.

[7] “/hierarchical-clustering-customer-segmentation,” [Online]. Available: https://www.coursera.org/projects/hierarchical-clustering-customer-segmentation.