



# Market Segmentation Based On Clustering Analysis of Credit Card Customers

Submitted by - Lekhansh



# Objective

Marketing helps businesses increase sales target, brand awareness and engagement. One of the methods for the marketing team to understand their customer, is by dividing their customer by their characteristics which is called customer segmentation.

Customer segmentation is the process by which you divide your customers up based on common characteristics – such as demographics or behaviours, so you can market to those customers more effectively.

We have a data set that contains a summary of the usage behavior of about 9000 active credit card holders during the last 6 months. We will perform clustering analysis of our data using non-supervised learning which will help to segmentize the customer by finding a certain pattern. In hope we could find some characteristics between each customer segment.



# Methodology

In order to fully prepare our dataset for non supervised learning technique. We have to explore and analysis our dataset to make it more meaningful and to help understand the data much better.

We have performed following steps on our dataset:

1. Data Understanding and Exploratory Data Analysis
2. Data Pre-processing
3. Data modelling
4. Results and Comparisons




# Data Understanding

We will first try to import the data in the Collab and use `head()` function to just check the data and to observe anything noteworthy.

We see that `cust_id` column is not required for our modelling, hence we will remove it.

	cust_id	balance	balance_frequency	purchases	oneoff_purchases	installments_purchases	cash_advance	purch
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.000000	



After using the `count()` function, we can see that:  
Total number of credit card holders = 8950

Missing values columns = **credit\_limit** and **minimum\_payments**

I will further impute these NaN values during pre-processing.

```
[ ] df.count()

cust_id                8950
balance                8950
balance_frequency      8950
purchases              8950
oneoff_purchases       8950
installments_purchases 8950
cash_advance           8950
purchases_frequency    8950
oneoff_purchases_frequency 8950
purchases_installments_frequency 8950
cash_advance_frequency 8950
cash_advance_trx       8950
purchases_trx          8950
credit_limit           8949
payments              8950
minimum_payments       8637
prc_full_payment       8950
tenure                 8950
dtype: int64
```

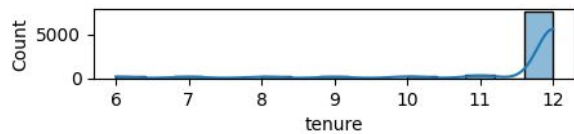
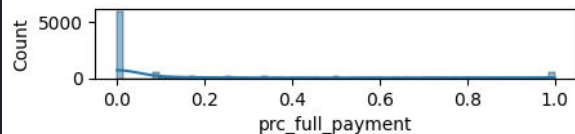
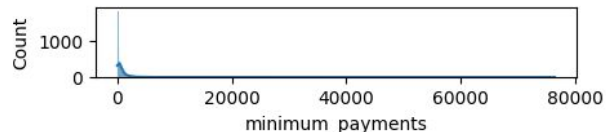
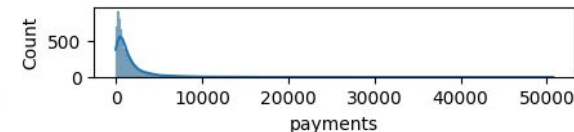
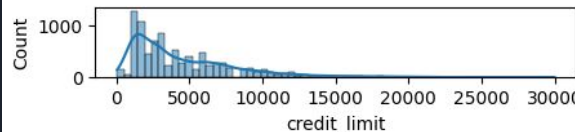
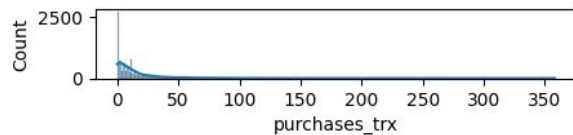
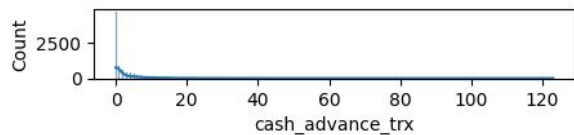
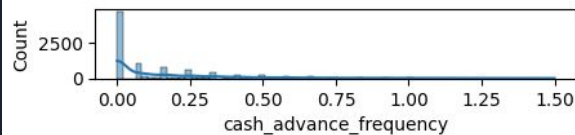
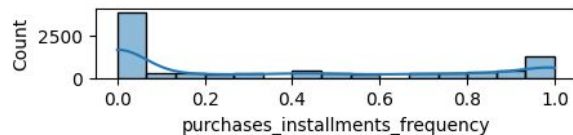
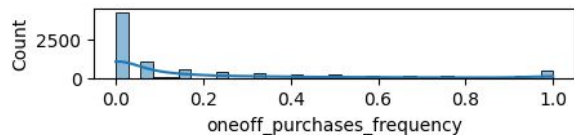
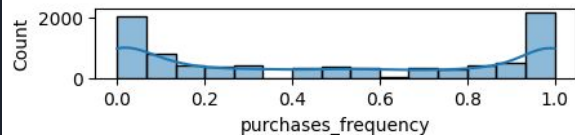
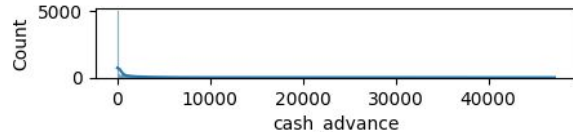
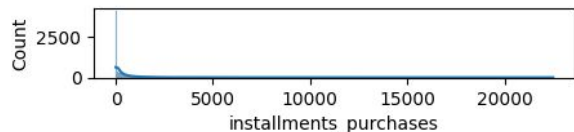
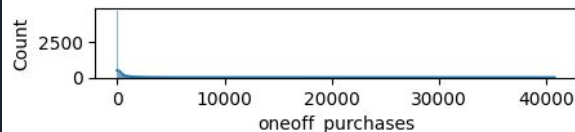
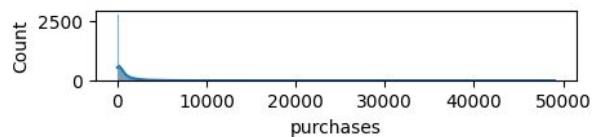
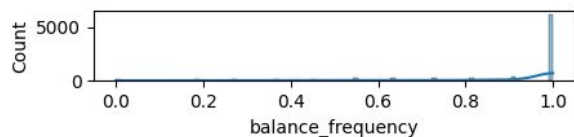
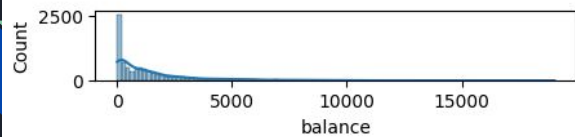


# Univariate Analysis

Univariate analysis is the simplest form of method to analyze data. Here, I will try to analyse the behaviour of all the variables by plotting histogram and using Kernel Distribution Estimation (KDE).

From this, we will try to find the patterns in the data.

The result from histogram analysis and its result are given in the next slides.





# Univariate Analysis Results

1. Features like 'balance', 'purchases', 'oneoff\_purchases', 'installment\_purchases' and 'cash\_advance' show similar trend.
2. For around 6000 customers, the balance frequency are frequently updated.
3. The purchases are being made frequently by around 2000 customers. Also the purchase frequency can be categorised into 2 categories - one who frequently purchases and those who rarely purchases.
4. There were more customers who made purchases using installments rather than one-go purchases.
5. Majority of credit limit for a user falls between 0 and 15000.
6. Around 5% of users have made full payment while around 6000 customers still haven't made any payment/installment.
7. Around 80% of customers have credit card tenure service in 12th month.
8. The balances of 95% of users are between 0 and 5000.

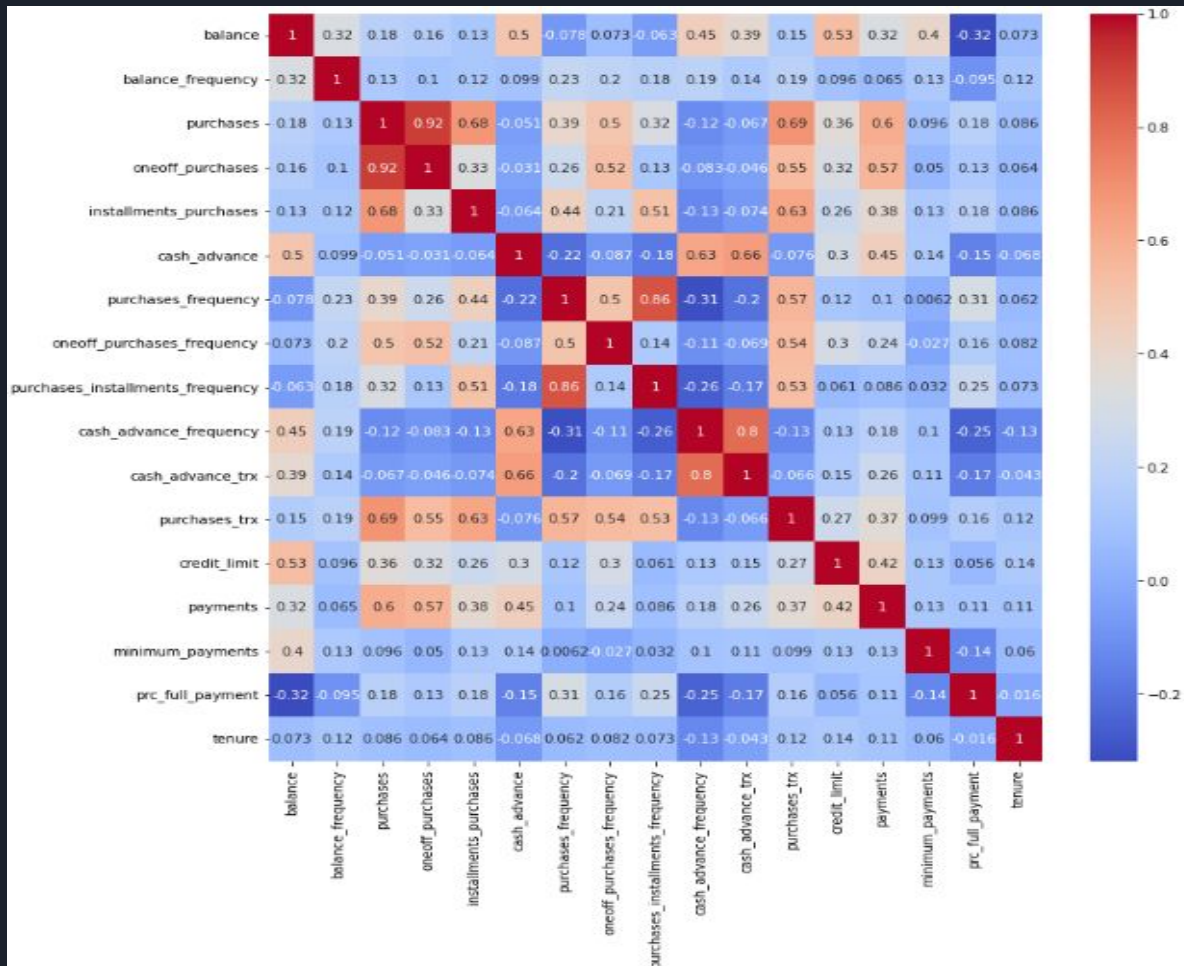




# Bivariate Analysis

Here, I will try to analyse the relationship of one variable with another, for the purpose of determining the empirical relationship between them. The main purpose of bivariate analysis is to determine the degree of association between two variables, if it exists.

For this purpose, I have used heatmaps. In heatmaps, two variables are plotted against one another on both the axes. It is the graphical representation of data where each value of a matrix is represented as a color.





# Bivariate Analysis Results

We can see almost **perfect positive correlation** between following features:

- \* Purchases and One-off purchases
- \* Cash Advance Frequency and Cash advance transactions
- \* Purchase frequency and purchase installments frequency

## **Medium Correlation between:**

- \* Purchases and Installments purchases
- \* Purchases and Purchase Transactions
- \* Purchases and Payments
- \* Cash Advance and Cash Advance Transactions / Cash Advance Frequency




# Data Pre-Processing

Data pre-processing refers to manipulation or dropping of data. The main purpose of data pre-processing is to clean the data and to make it ready before modelling is performed on it.

I have done the following steps in my data pre-processing:

1. Removal of unnecessary columns.
2. Checking for duplicates and removal (if any).
3. Handling missing values.
4. Checking for outliers and handling.


- 
1. Removal of unnecessary columns - Since cust\_id is not necessary for our data, so we have removed it from our dataframe.
  2. Checking for duplicates

```
# Checking for duplicates  
df_new[df_new.duplicated()]
```

```
balance  balance_frequency  purchases  onecoff_purchases  installments_purchases  cash_advance  purchases_frequency  onecoff_purchases_frequency  purchases_ins
```


3. Missing values handling : After for missing data, I found out that 2 columns: credit\_limit and minimum\_payments have NaN values. I choose to input these missing values with the median. Since it is a good option to consider when the data is skewed.

However we can replace missing values with 0, mean, or mode of the variable column

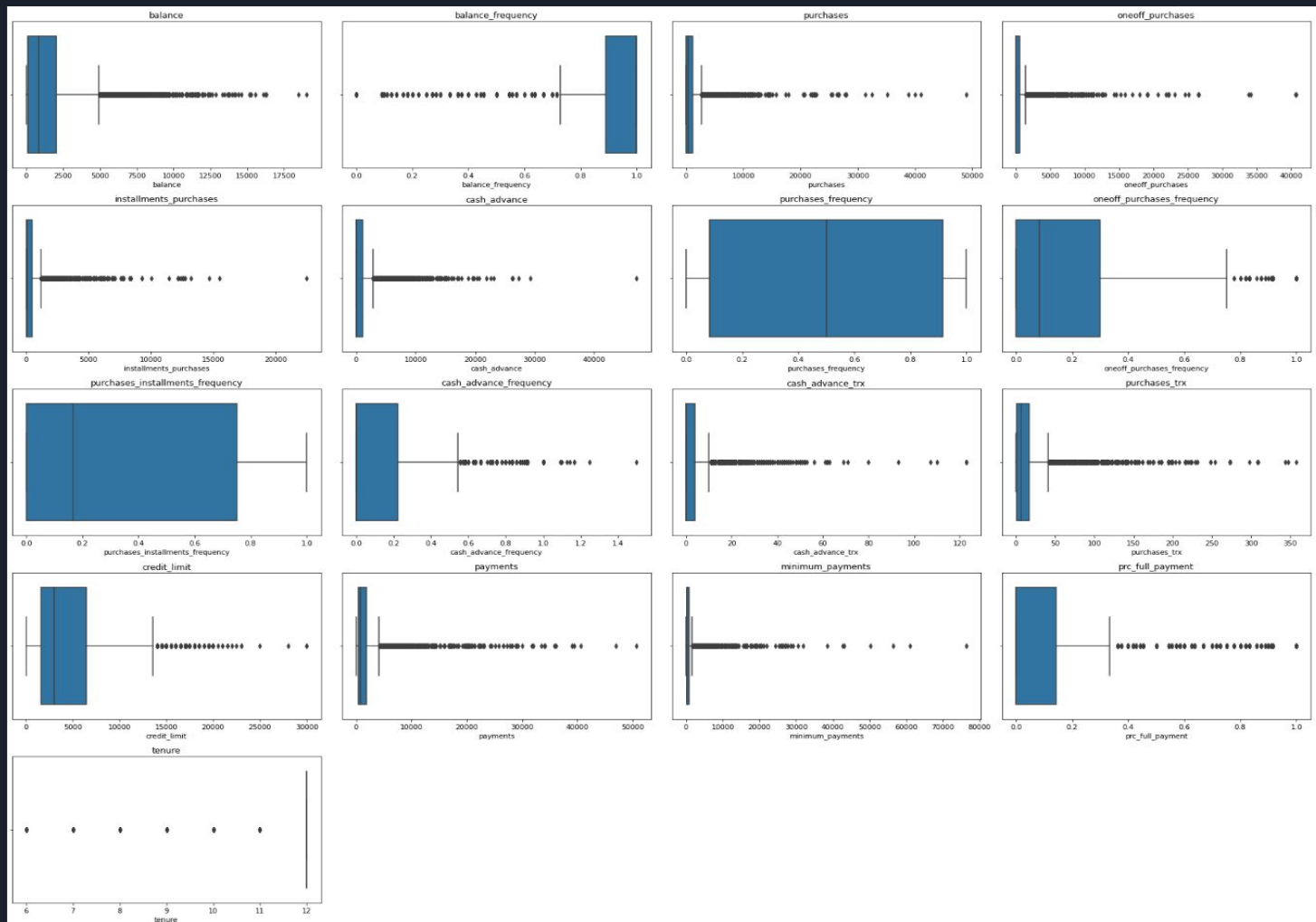


```
balance                0
balance_frequency      0
purchases              0
oneoff_purchases       0
installments_purchases 0
cash_advance           0
purchases_frequency    0
oneoff_purchases_frequency 0
purchases_installments_frequency 0
cash_advance_frequency 0
cash_advance_trx       0
purchases_trx          0
credit_limit           1
payments              0
minimum_payments      313
prc_full_payment       0
tenure                0
dtype: int64
```


```
[ ] # Replacing missing value for median
df_new['credit_limit'].fillna(df_new['credit_limit'].median(), inplace = True)
df_new['minimum_payments'].fillna(df_new['minimum_payments'].median(), inplace = True)
```



**4. Checking for outliers :** I used box plot to find if any variable has outliers or not. Box plot is a method for graphically depicting groups of numerical data through quartiles. It is based on 5 number summary i.e. minimum, first quartile, median, third quartile and maximum. Many statistical parameters like means, standard deviation and correlations are highly sensitive to outliers. Despite this, it is not always recommended to drop an observations just because it is an outlier. They should be removed if they represent measurement errors, data entry or poor sampling. After applying boxplot, since the percentage of outliers is high, so instead of removing the outliers we will try to normalize our data and reduce skewness. We will apply square root transformation to reduce the percentage of outliers in the data.







Applying square root transformation on the dataset to handle outliers and so that skewness and kurtosis is reduced.

```
[7] data_sqrt = np.sqrt(df_new)
```

```
[8] data_sqrt.head()
```

	balance	balance_frequency	purchases	oneoff_purchases	installments_purchases	cash_advance	purchases_frequency	oneoff_purchases_frequency	purchases_instal
0	6.395369	0.904534	9.767292	0.000000	9.767292	0.000000	0.408249	0.000000	
1	56.590347	0.953463	0.000000	0.000000	0.000000	80.267961	0.000000	0.000000	
2	49.951465	1.000000	27.805935	27.805935	0.000000	0.000000	1.000000	1.000000	
3	40.824877	0.797724	38.716921	38.716921	0.000000	14.345313	0.288675	0.288675	
4	28.595705	1.000000	4.000000	4.000000	0.000000	0.000000	0.288675	0.288675	

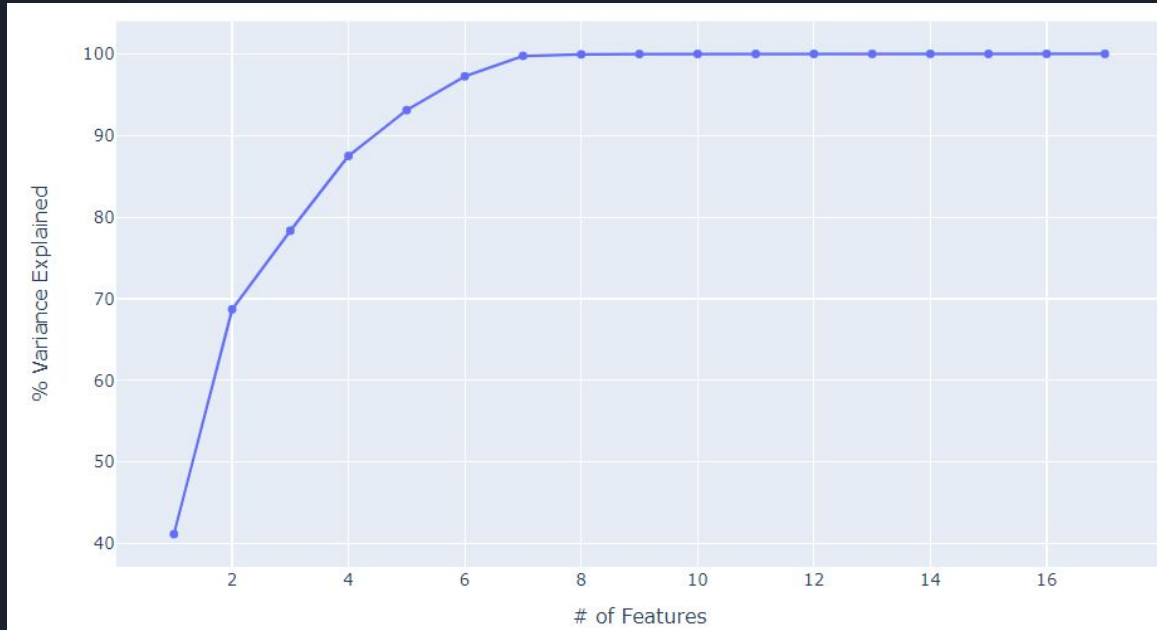


# Principal Component Analysis


Since our dataset has around 15 variables, we will try to apply PCA on our dataset to reduce the dimensionality. This will help us better visualize the clusters.

PCA helps to interpret the data, by simplifying the complexity in high-dimensional data while retaining patterns and trends. This is done by transforming the data into fewer dimensions which act as summaries of features.

For this, I first applied PCA while taking all the features into considerations. I then plotted the graph between the number of components and the percentage of variability explained.



Information extracted from 2 components = Approx 70%  
Now we will apply PCA on our data and reduce it into 2-Dimensions.



After applying PCA on our data and reducing the dataset into 2 variables. I renamed  
The dataset as **X\_principal1** with 2 columns : **C1** and **C2**.

	<b>C1</b>	<b>C2</b>
<b>0</b>	-52.679069	4.660530
<b>1</b>	44.150781	-63.756182
<b>2</b>	13.452254	8.591083
<b>3</b>	7.110526	12.249603
<b>4</b>	-36.904244	-6.436452
...	...	...
<b>8945</b>	-49.215784	12.236793
<b>8946</b>	-47.684274	10.723703
<b>8947</b>	-55.404320	7.087477
<b>8948</b>	-65.019136	-6.198274
<b>8949</b>	-29.860969	16.509145
8950 rows × 2 columns		



# Data Modelling

Next step is to perform data modelling on our dataset. I have used 3 non-supervised clustering techniques for this process.

1. K-Means
2. HDBSCAN
3. Hierarchical (Agglomerative) Clustering

To find the hyper parameters for our model ( optimal number of clusters, eps value ), I have used these techniques:

1. Inertia Plot and Silhouette Visualiser - K-means
2. Davies Bouldin method - Hierarchical
3. Silhouette Score - HDBSCAN



# 1. K-MEANS

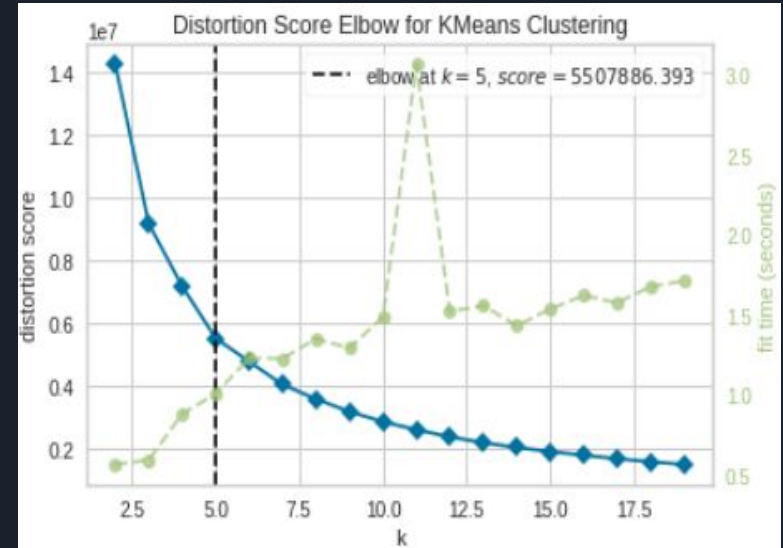
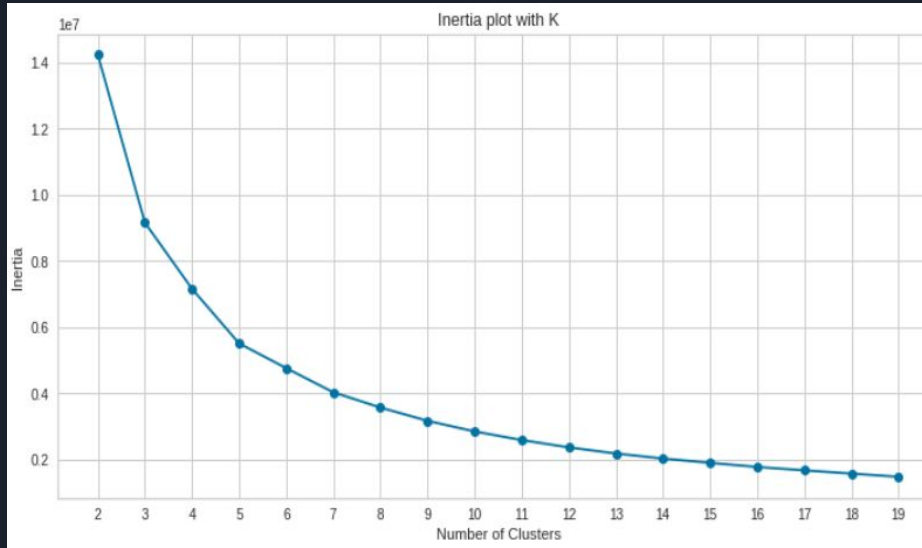
K-Means clustering groups similar items in the form of clusters. The number of groups is represented by K.

It works in 3 steps.

1. Select the k values.
2. Initialize the centroids.
3. Select the group and find the average.

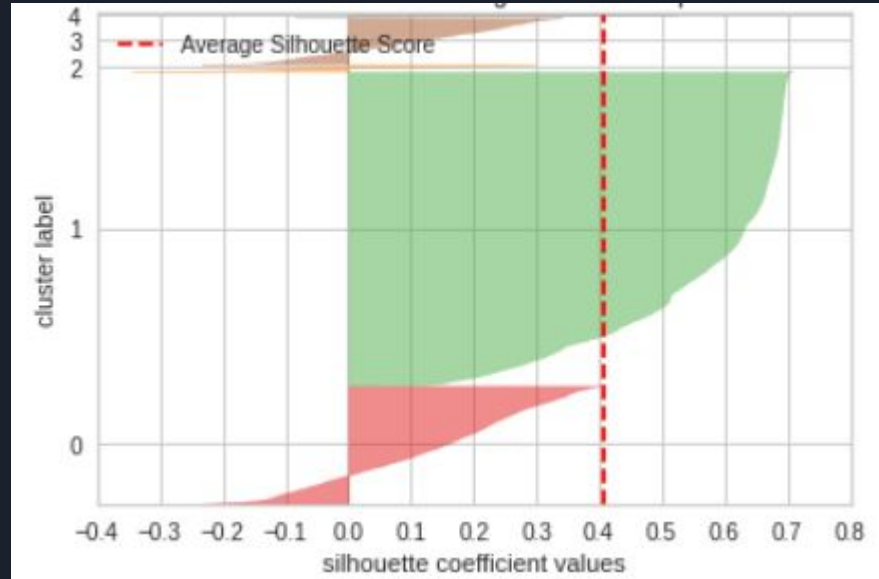
I need to find the optimal number of clusters for our model. So I will train the model for the different values of K( such as in range from 1 to 15). I then used inertia plot , KElbow Visualizer and Silhouette Visualizer to find the value of K.

Inertia Plot : The inertia or *within cluster of sum of squares* value tells us of how coherent the different clusters are. The smaller the Inertia value, the more coherent are the different clusters. For this, Elbow method can be used. For this, the elbow point is determined on the inertia plot.



When adding more clusters, the inertia value decreases, but the information contained in a cluster also decreases further. So based on Elbow method, I chose  $K = 5$ .

After modelling k-means on X\_principal1 dataset, I used silhouette visualizer to visualize the clusters with respect to the data points. The value of range of silhouette is between -1 and 1, where 1 is highly dense clusters and -1 is completely incorrect clusters. Silhouette Visualizer displays the coefficient for each sample on a per-cluster basis.



Based on this, It can be said that clusters are uneven in size and maximum silhouette score is 0.7 for cluster label '1'. I will further analyze clusters from HDBSCAN and agglomerative to compare results.





## 2. Hierarchical Clustering (Agglomerative)

Hierarchical clustering is a general family of clustering algorithms that build nested clusters by merging or splitting them successively. This hierarchy of clusters is represented as a tree (or dendrogram). The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample.

**For linkage criteria : Ward method, Complete method**

**Ward** minimizes the sum of squared differences within all clusters. It is a variance-minimizing approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.

**Complete** : The link between two clusters contains all element pairs, and the distance between clusters equals the distance between those two elements (one in each cluster) that are farthest away from each other.

**For finding the value of K:**

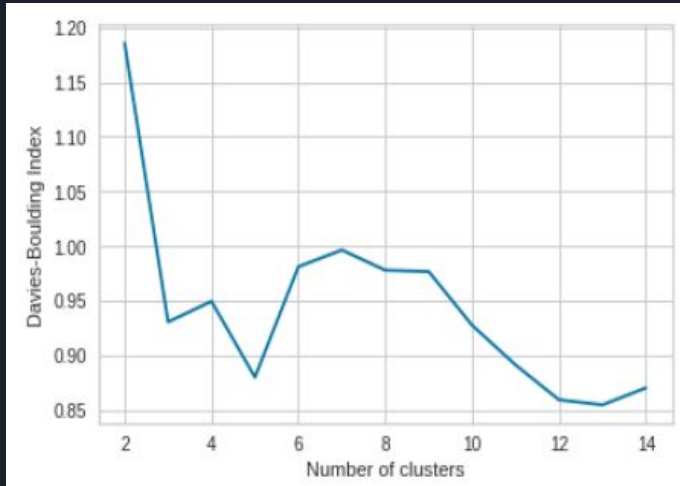
Davies Bouldin Method

## Davies Bouldin Score

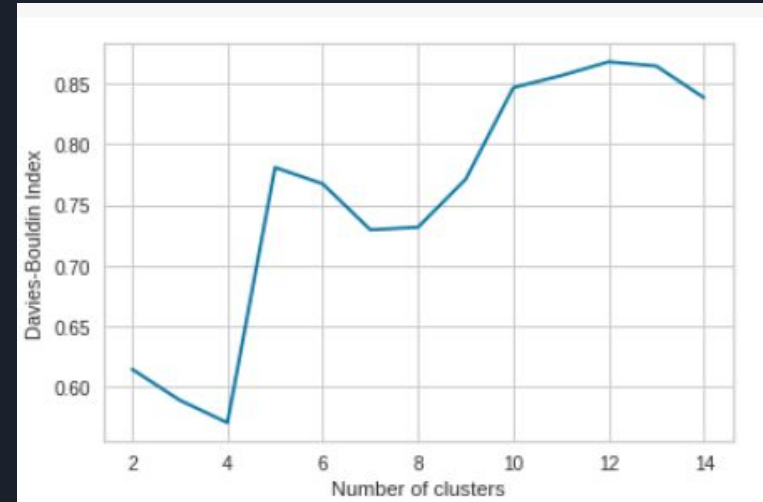
It is most commonly used to evaluate the goodness of split by a K-Means clustering algorithm for a given number of clusters.

DBI) is calculated as the average similarity of each cluster with a cluster most similar to it. The lower the average similarity is, the better the clusters are separated and the better is the result of the clustering performed.


After running the model for a range of K (2,15) and plotting against DB score. Below is the graph obtained



Ward Linkage



Complete Linkage



Linkage method = Ward, Complete

K = 5

Affinity = euclidean

After fitting the model based on above hyper-parameters, we get 5 clusters with cluster labels namely 0, 1, 2, 3, 4. (Ward method)

```
hier
```

```
array([4, 2, 1, ..., 4, 4, 4])
```

Using Linkage method, we get 4 clusters with labels : 0, 1, 2, 3.

```
hier1
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

## Comparing clusters obtained from ward and complete linkage

Since DB score is minimized, when we use complete method. Hence we will prefer it over ward.

$K = 4$



Ward Linkage



Complete Linkage



## 3. HDBSCAN Modelling


HDBSCAN is a high density based clustering technique which works using following steps:

1. Estimate the densities .
2. Pick regions of high density.
3. Combine points in these selected regions.

Selecting hyper parameters

1. `Min_cluster_size`
2. `Min_samples`

We set many values for `min_cluster_size` and `min_samples` and see which combination gives us the best by calculating the silhouette score and arranging them in a table.



After importing the HDBSCAN library in the collab, and reiterating the model using the various values of clusters and samples:

clusters=[15,20,30,40,50,60]

samples=[5,10,15]

I obtained the silhouette score for each of the combination.

	min_cluster_size	min_samples	SILHOUETTE SCORE
6	15	10	0.363515
13	20	15	0.282685
11	60	10	0.274484
3	40	5	0.266352
4	50	5	0.109850
5	60	5	0.109850
16	50	15	0.063609
15	40	15	0.063609
9	40	10	0.038850
8	30	10	0.038850

Based on the high silhouette score, I selected the hyper parameters to fit on our model.

Min\_cluster\_size = 15

Min\_samples = 10

Using the selected hyper parameters and fitting them on our HDBSCAN model, I get 3 clusters with labels : **-1, 0, 1**.

```
array([-1,  0,  1])
```

After visualising the HDBSCAN model against our columns C1 and C2. Below is the graph to check clusters. Please note that the clusters using this method are overlapping and not a correct estimation to group the customers.





# Results and Comparisons

After modelling using 3 clustering techniques, below are the number of clusters obtained.

K - Means = 5

HDBSCAN = 3

Hierarchical using Ward = 5

Hierarchical using Complete Linkage = 4

To evaluate which model works best on our dataset, I calculated and compared the silhouette score for each model.



Calculating the silhouette score for all 4 models and converting the scores into a dataframe.

best_model				
	kmeans	hdbscan	hier_ward	hier_complete
0	0.4	0.36	0.36	0.55

We can see the best model compared using Silhouette score is Hierarchical model using complete linkage with score of **0.55**.

