Customer Segmentation

(Clustering Project)

By Srikant Ganguly

Our Task

We are required to develop a customer segmentation to define marketing strategy. We have the data of 9000 active credit card holders for last 6 months with 18 behavioral variables.

We will use unsupervised machine learning algorithms to group (cluster) customers with alike characteristics.

Outline

- Loading the data, required python libraries, etc.
- Learning about the data like data type, missing values, statistical parameters.
- Data Processing (Data Analysis).
- Principal Component Analysis (PCA)
- K-Means Clustering
- Agglomerative Clustering
- Density Based Clustering
- Insights and Conclusion

Descriptive Statistics

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

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURC
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	

Data Types and count.

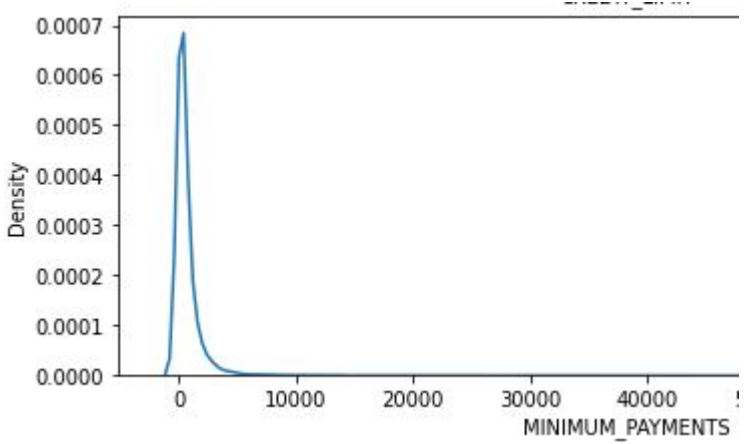
```
[5]  df.info()

<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
```

Imputing Missing Values

- Minimum Payments has missing values and since it is a bit skewed, we will follow the standard method of imputing the values by Median.

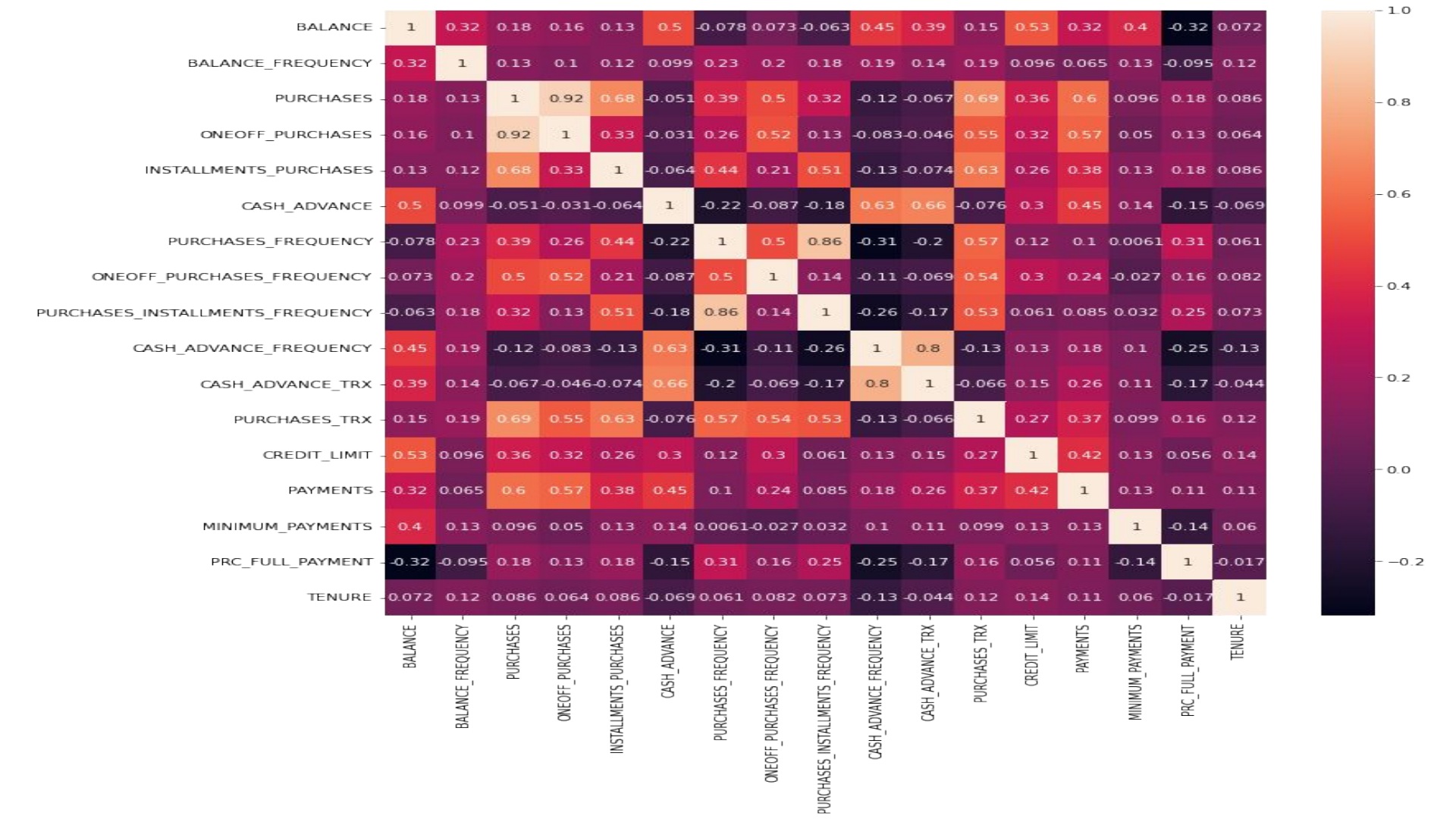
- Credit Limit has one missing value so we just drop it rather than taking the pain of imputing it for negligible benefits.



Correlation Analysis

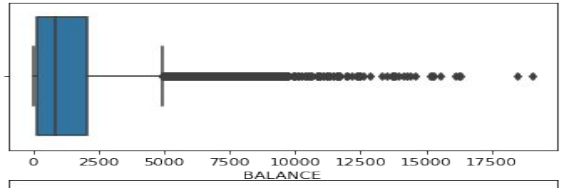
- We do a correlation analysis to check the correlation between different variables.
- In the next slide, we have the heatmap for correlation analysis. Following observations can be made from that.

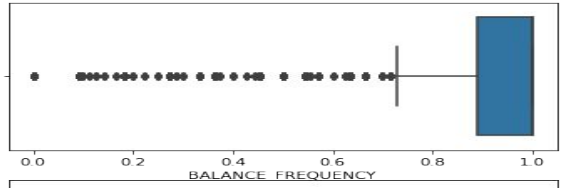
1. Purchases and one-off purchases are highly correlated which means majority of purchases are one-off purchases.
2. Purchase frequency has a negative correlation with cash-advance frequency thus indicating frequent buyers don't pay beforehand.
3. Full Payment has negative correlation with Balance which suggests people still don't make payments even when balance is high.

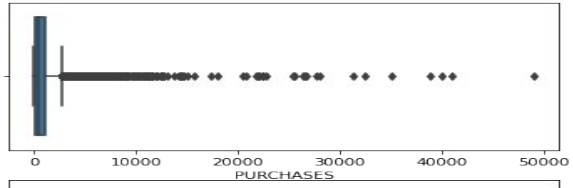

BALANCE 1 0.32 0.18 0.16 0.13 0.5 -0.078 0.073 -0.063 0.45 0.39 0.15 0.53 0.32 0.4 -0.32 0.072
BALANCE_FREQUENCY 0.32 1 0.13 0.1 0.12 0.099 0.23 0.2 0.18 0.19 0.14 0.19 0.096 0.065 0.13 -0.095 0.12
PURCHASES 0.18 0.13 1 0.92 0.68 -0.051 0.39 0.5 0.32 -0.12 -0.067 0.69 0.36 0.6 0.096 0.18 0.086
ONEOFF_PURCHASES 0.16 0.1 0.92 1 0.33 -0.031 0.26 0.52 0.13 -0.083 -0.046 0.55 0.32 0.57 0.05 0.13 0.064
INSTALLMENTS_PURCHASES 0.13 0.12 0.68 0.33 1 -0.064 0.44 0.21 0.51 -0.13 -0.074 0.63 0.26 0.38 0.13 0.18 0.086
CASH_ADVANCE 0.5 0.099 -0.051 -0.031 -0.064 1 -0.22 -0.087 -0.18 0.63 0.66 -0.076 0.3 0.45 0.14 -0.15 -0.069
PURCHASES_FREQUENCY -0.078 0.23 0.39 0.26 0.44 -0.22 1 0.5 0.86 -0.31 -0.2 0.57 0.12 0.1 0.0061 0.31 0.061
ONEOFF_PURCHASES_FREQUENCY 0.073 0.2 0.5 0.52 0.21 -0.087 0.5 1 0.14 -0.11 -0.069 0.54 0.3 0.24 -0.027 0.16 0.082
PURCHASES_INSTALLMENTS_FREQUENCY -0.063 0.18 0.32 0.13 0.51 -0.18 0.86 0.14 1 -0.26 -0.17 0.53 0.061 0.085 0.032 0.25 0.073
CASH_ADVANCE_FREQUENCY 0.45 0.19 -0.12 -0.083 -0.13 0.63 -0.31 -0.11 -0.26 1 0.8 -0.13 0.13 0.18 0.1 -0.25 -0.13
CASH_ADVANCE_TRX 0.39 0.14 -0.067 -0.046 -0.074 0.66 -0.2 -0.069 -0.17 0.8 1 -0.066 0.15 0.26 0.11 -0.17 -0.044
PURCHASES_TRX 0.15 0.19 0.69 0.55 0.63 -0.076 0.57 0.54 0.53 -0.13 -0.066 1 0.27 0.37 0.099 0.16 0.12
CREDIT_LIMIT 0.53 0.096 0.36 0.32 0.26 0.3 0.12 0.3 0.061 0.13 0.15 0.27 1 0.42 0.13 0.056 0.14
PAYMENTS 0.32 0.065 0.6 0.57 0.38 0.45 0.1 0.24 0.085 0.18 0.26 0.37 0.42 1 0.13 0.11 0.11
MINIMUM_PAYMENTS 0.4 0.13 0.096 0.05 0.13 0.14 0.0061 -0.027 0.032 0.1 0.11 0.099 0.13 0.13 1 -0.14 0.06
PRC_FULL_PAYMENT -0.32 -0.095 0.18 0.13 0.18 -0.15 0.31 0.16 0.25 -0.25 -0.17 0.16 0.056 0.11 -0.14 1 -0.017
TENURE 0.072 0.12 0.086 0.064 0.086 -0.069 0.061 0.082 0.073 -0.13 -0.044 0.12 0.14 0.11 0.06 -0.017 1
BALANCE
BALANCE_FREQUENCY
PURCHASES
ONEOFF_PURCHASES
INSTALLMENTS_PURCHASES
CASH_ADVANCE
PURCHASES_FREQUENCY
ONEOFF_PURCHASES_FREQUENCY
PURCHASES_INSTALLMENTS_FREQUENCY
CASH_ADVANCE_FREQUENCY
CASH_ADVANCE_TRX
PURCHASES_TRX
CREDIT_LIMIT
PAYMENTS
MINIMUM_PAYMENTS
PRC_FULL_PAYMENT
TENURE
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

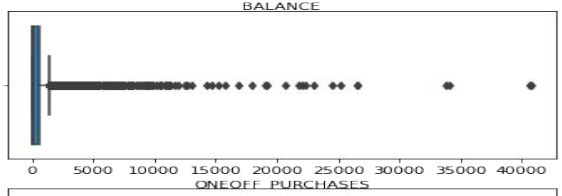
Outlier Detection

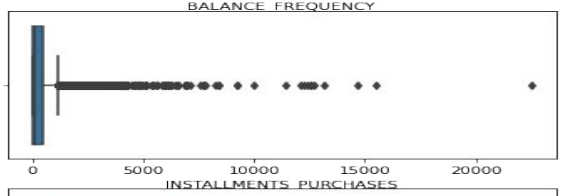
- In the following slide I show the boxplot diagrams for each feature to check for the outliers.
- We observe that the outliers make a significant portion of the data and thus cannot be eliminated completely as that would reduce the information (data) considerably.
- To get around the issue of outliers, we normalize the data subsequently.

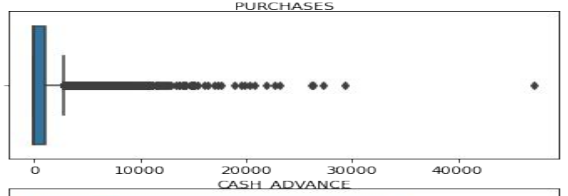

0
2500
5000
7500
10000
12500
15000
17500
BALANCE

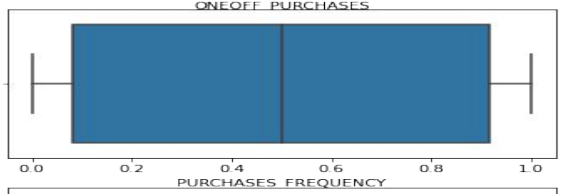

0.0
0.2
0.4
0.6
0.8
1.0
BALANCE_FREQUENCY


0
10000
20000
30000
40000
50000
PURCHASES

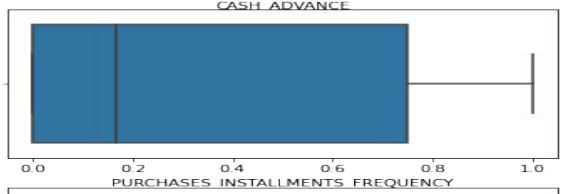

0
5000
10000
15000
20000
25000
30000
35000
40000
ONEOFF_PURCHASES

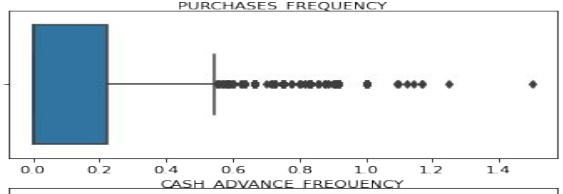

0
5000
10000
15000
20000
INSTALLMENTS_PURCHASES

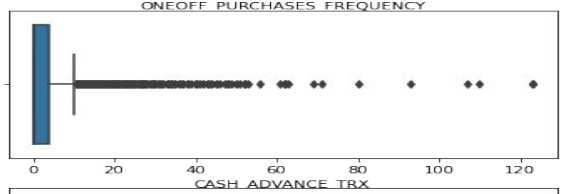

0
10000
20000
30000
40000
CASH_ADVANCE


0.0
0.2
0.4
0.6
0.8
1.0
PURCHASES_FREQUENCY

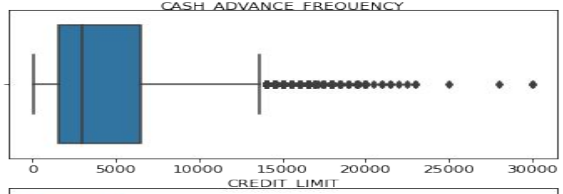

0.0
0.2
0.4
0.6
0.8
1.0
ONEOFF_PURCHASES_FREQUENCY

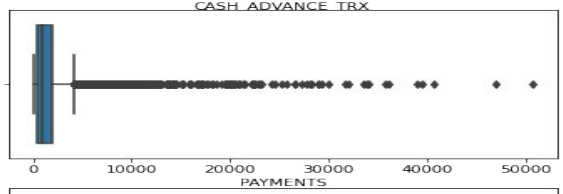

0.0
0.2
0.4
0.6
0.8
1.0
PURCHASES_INSTALLMENTS_FREQUENCY

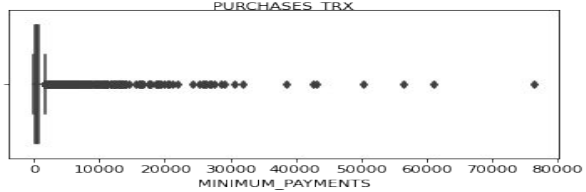

0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
CASH_ADVANCE_FREQUENCY

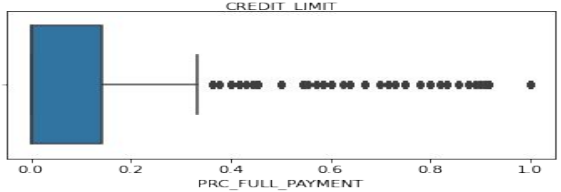

0
20
40
60
80
100
120
CASH_ADVANCE_TRX

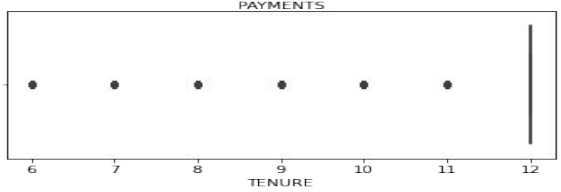

0
50
100
150
200
250
300
350
PURCHASES_TRX


0
5000
10000
15000
20000
25000
30000
CREDIT_LIMIT


0
10000
20000
30000
40000
50000
PAYMENTS


0
10000
20000
30000
40000
50000
60000
70000
80000
MINIMUM_PAYMENTS


0.0
0.2
0.4
0.6
0.8
1.0
PRC_FULL_PAYMENT


6
7
8
9
10
11
12
TENURE

Normalizing the data

- Min-max normalization preserves the relationship between original data values.
- With normalization, we end up with smaller standard deviations and thus suppress the effect of outliers.
- Also called *feature scaling*.

```
# Normalizing the data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df_normal = scaler.fit_transform(df)
df_normal.shape
```

Clustering

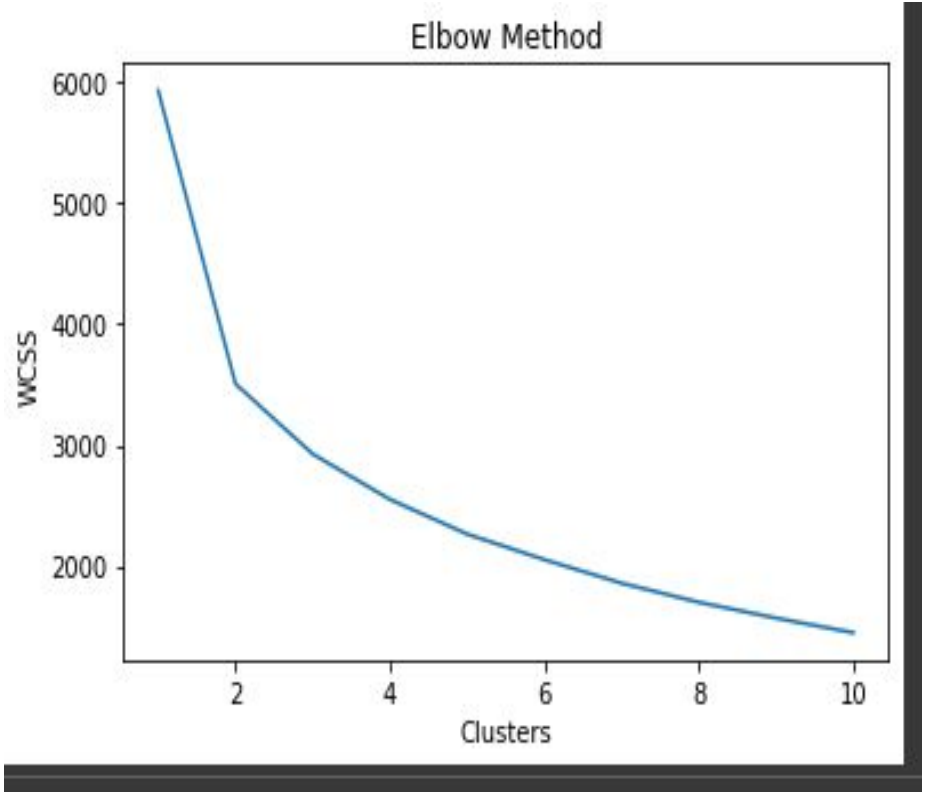
- Now we are ready to do what we set out to do that is clustering.
- We will start with K-means, then we will do the agglomerative clustering and finally we will delve into Density-based clustering.
- We will also do PCA for dimension reduction and for visualizing the clusters.
- Finally we make some inferences about our methods and results.

K-Means

- K-means algorithm works on the principle of assigning every data point to a nearest cluster with a centroid keeping this as low as possible.
- The “means” in K-means refer to the averaging of the data we do to find the centroid.
- Number of clusters can be found using “Elbow method”, determining the silhouette coefficient, DB coefficient, etc.
- We will use Silhouette coefficient and Elbow method to determine the best possible cluster number.

Elbow Method

- We can see an elbow at K=2 and also at K=3.
- But in this case, this method is not that conclusive as it suggests that cluster number be 2 which is not a very good idea for a data of this size.
- So, we will go ahead and get a more concrete interpretation from comparing the Silhouette coefficients.
- WCSS stands for ‘Within Cluster sum of Squares’.

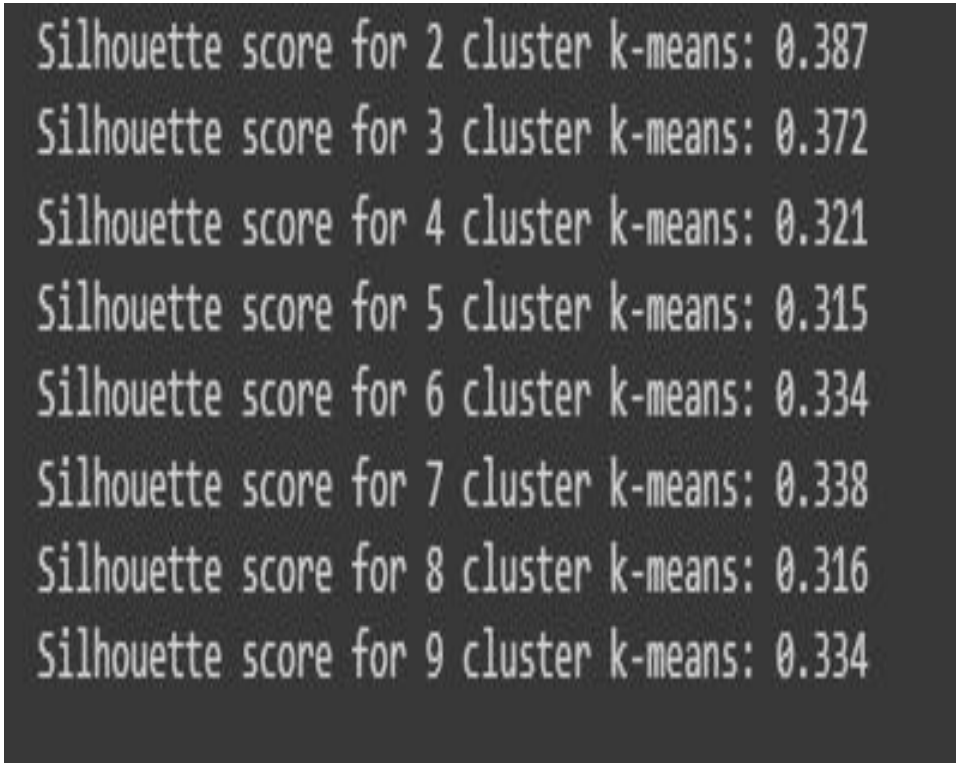


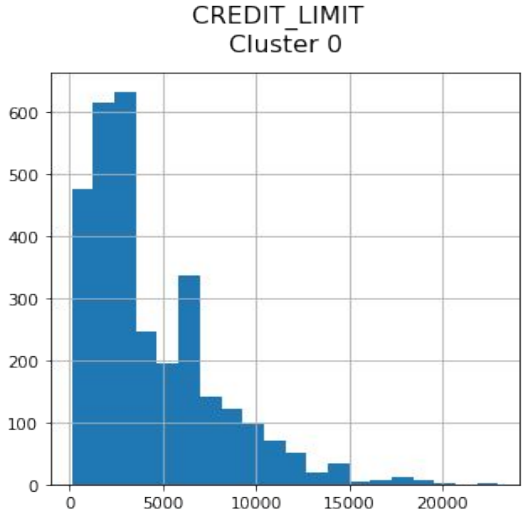
Silhouette Scores

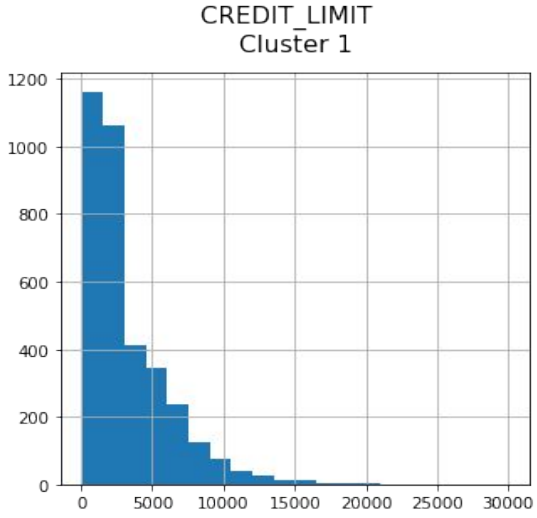
As we can see that the biggest decrease in silhouette coefficient is from K=3 to K=4,
I would like to proceed with K=3 clusters.

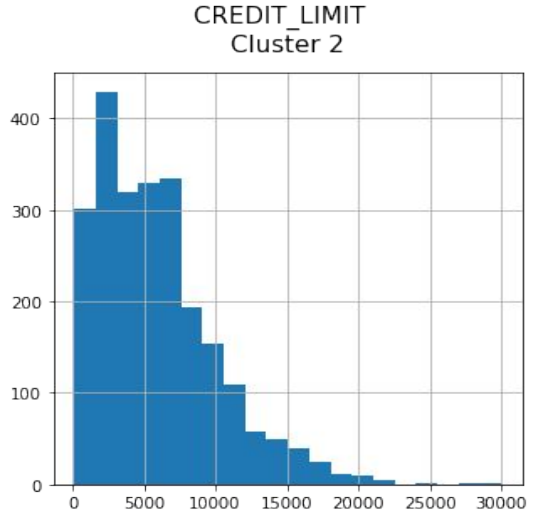
Also, there is not a great change in the value from K=2 to K=3. Thus there is not much harm in proceeding with K=3.

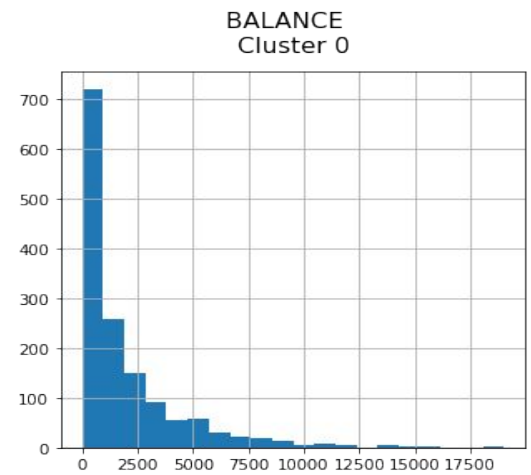
It should be noted that K=2 is equally good estimation as far as Silhouette score is concerned.

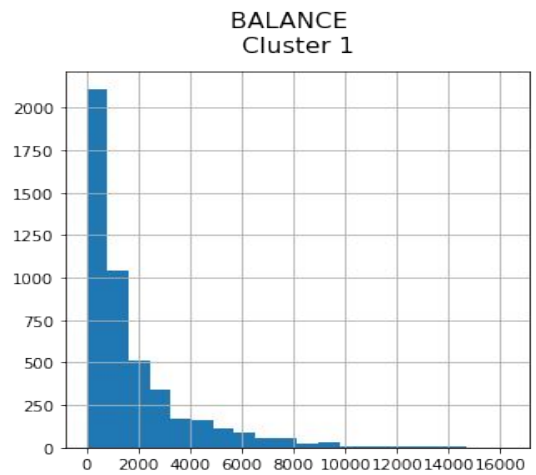


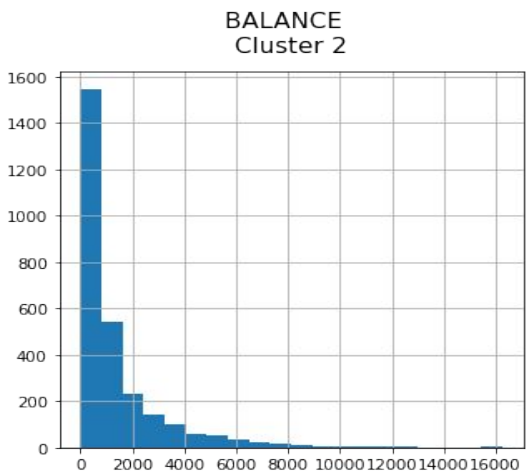

CREDIT_LIMIT
Cluster 0
0
100
200
300
400
500
600
0
5000
10000
15000
20000

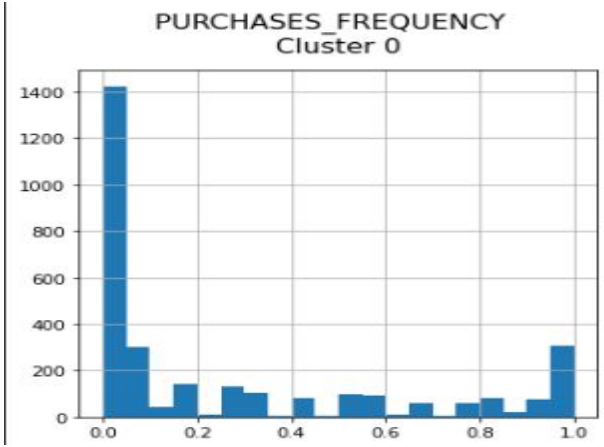

CREDIT_LIMIT
Cluster 1
0
200
400
600
800
1000
1200
0
5000
10000
15000
20000
25000
30000

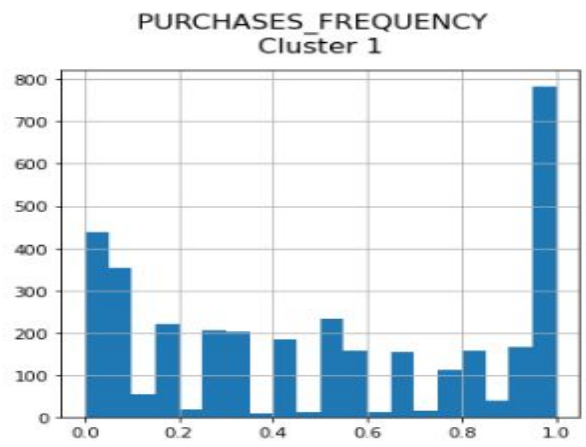

CREDIT_LIMIT
Cluster 2
0
100
200
300
400
0
5000
10000
15000
20000
25000
30000

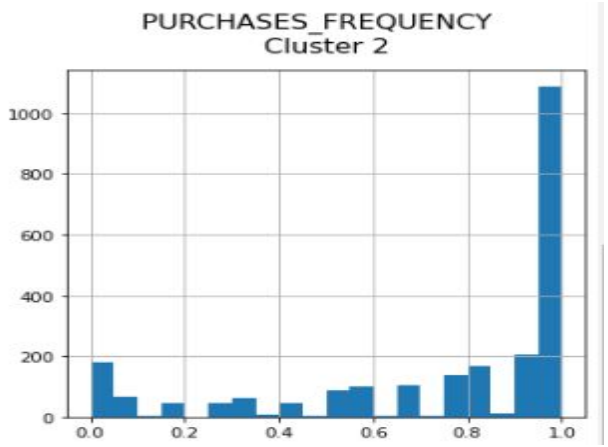

BALANCE
Cluster 0
0
100
200
300
400
500
600
700
0
2500
5000
7500
10000
12500
15000
17500


BALANCE
Cluster 1
0
250
500
750
1000
1250
1500
1750
2000
0
2000
4000
6000
8000
10000
12000
14000
16000


BALANCE
Cluster 2
0
200
400
600
800
1000
1200
1400
1600
0
2000
4000
6000
8000
10000
12000
14000
16000







INFERENCES

1. Cluster0 - People who don't make a purchase often, has medium spending power and average balance.
2. Cluster 1- People who make a purchase regularly but do not have much spending power.
3. Cluster 2- People who have very good spending power and do make purchases regularly.

Principal Component Analysis

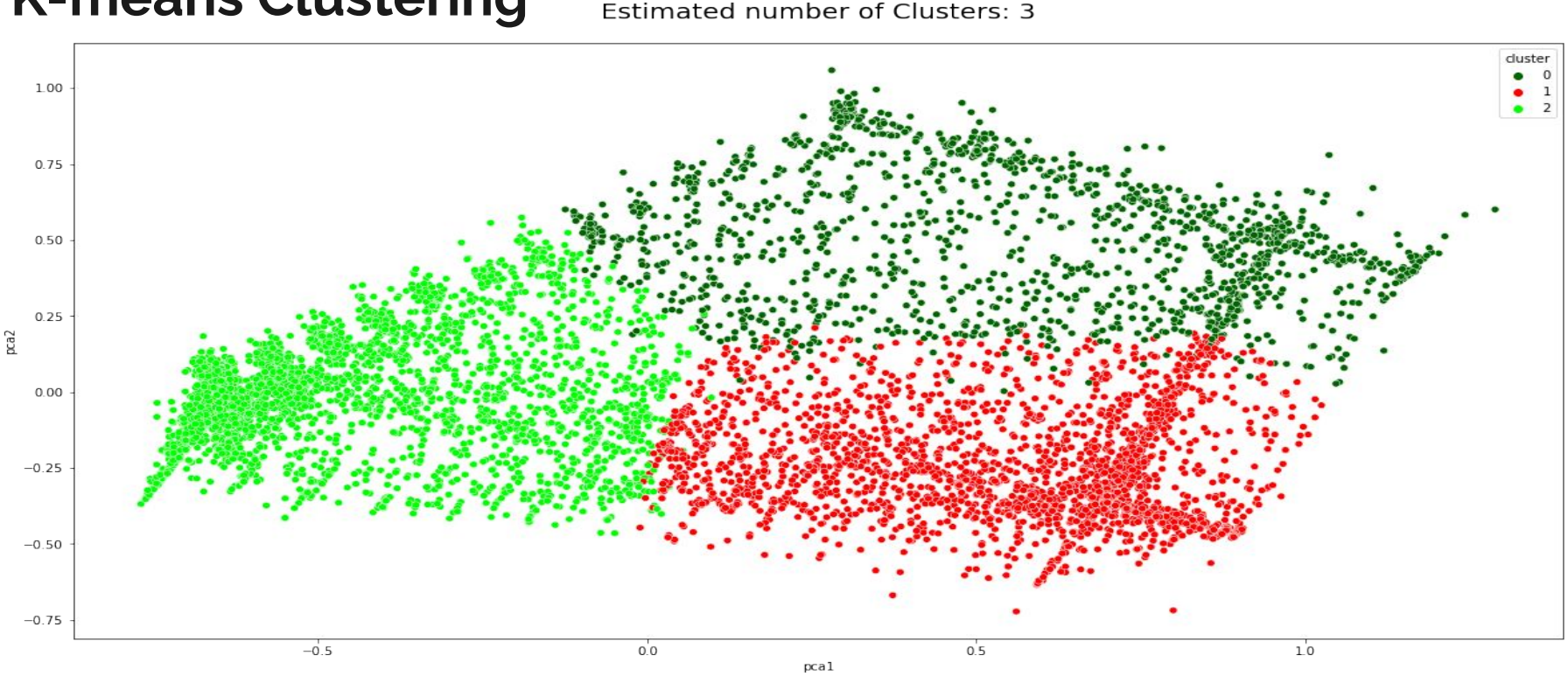
- Now since our data has more features than that can be plotted for visualization, it is not feasible to visualize them as it is with all the information intact.
- So we take use of a mathematical technique in Linear Algebra known as Principal Component Analysis (PCA).
- Not going into much mathematical detail, we can just say that it helps reduce the dimension of the data keeping most of the information intact.
- It is often used in machine learning for better visualization of the data.

- A new dataframe is created with two components named ‘pca1’ and ‘pca2’
- We will visualize the clustering plot with respect to these two components
- This is just to visualize the data and the cluster still represents the information they represented earlier.

```
pca_df = pd.DataFrame(data = principal_comp, columns = ['pca1', 'pca2'])
pca_df.head()
```

	pca1	pca2
0	-1.199279	-3.053819
1	-3.002632	1.935542
2	1.264356	0.827281
3	-0.685641	-0.244761
4	-1.428334	-1.780801

K-means Clustering



Agglomerative Clustering

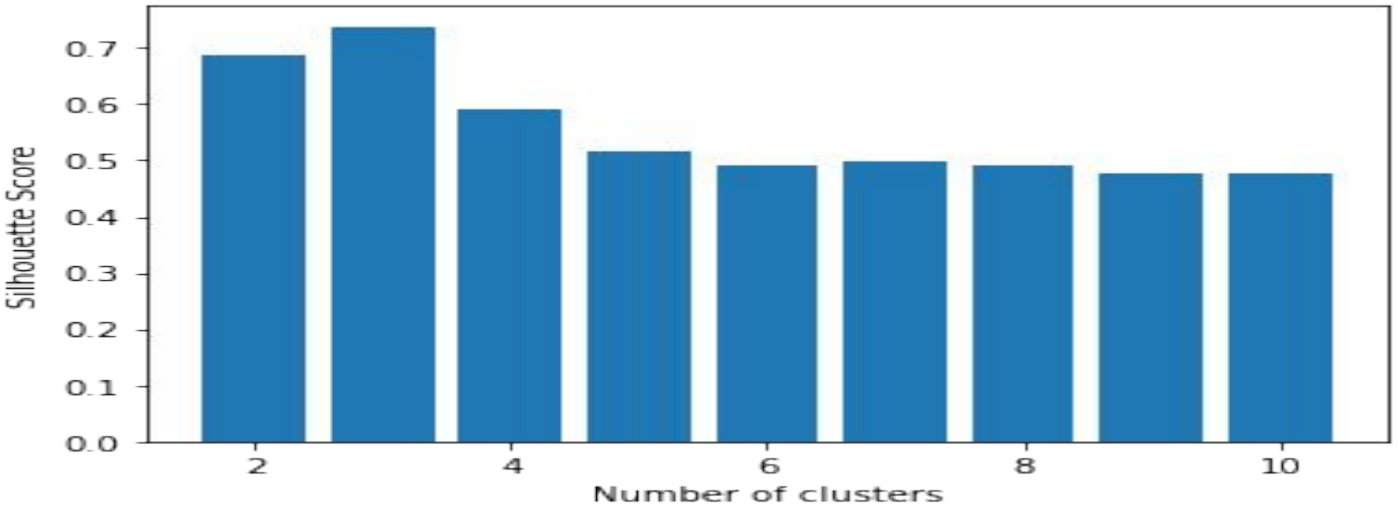
- One of the most common hierarchical clustering techniques.

- An Unsupervised learning algorithm that links data points based on distance to form cluster and subsequently linking those to form a structure of cluster with subclusters.

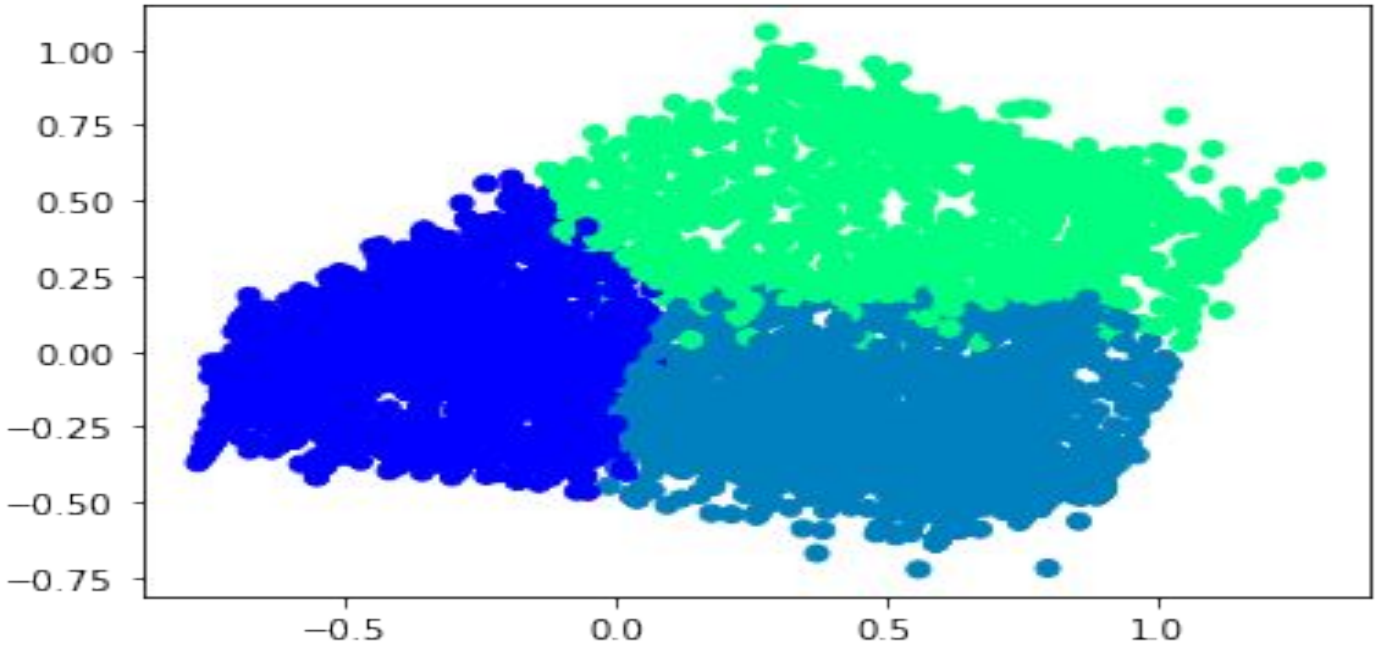
- We will use the ‘Silhouette Score’ to find the optimal number of clusters in this case.

Agglomerative Clustering

Using Silhouette score to get optimal number of clusters.



- As we can see that Silhouette score is maximum for number of cluster = 3, we will do the clustering with 3 clusters.
- Below is the cluster diagram.



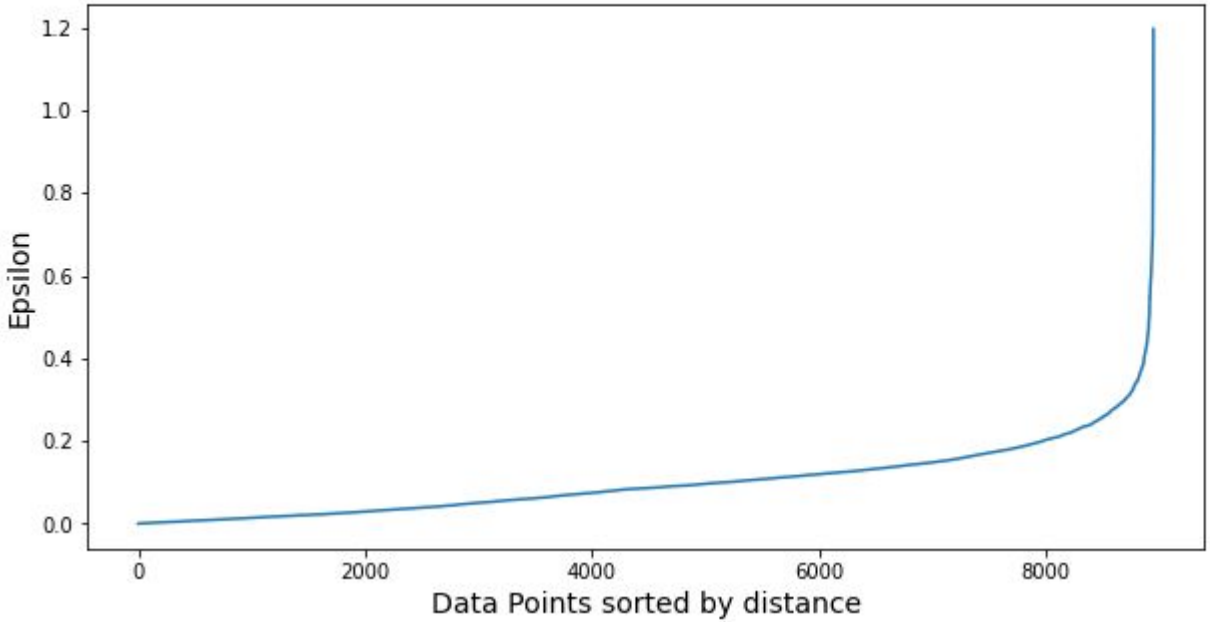
Density Based Clustering

- We will now perform a third and for the part final type of clustering known as “Density Based Spatial Clustering of Applications with Noise (DB-SCAN)”.

- It is a density-based clustering algorithm that forms clusters of dense regions ignoring low-dense regions.

- We will apply standard techniques to determine the ‘epsilon’ and ‘minimum points’ hyperparameters for this clustering.

The optimum value of epsilon is at the point of maximum curvature in the K-Distance Graph, which is 0.3 in this case.



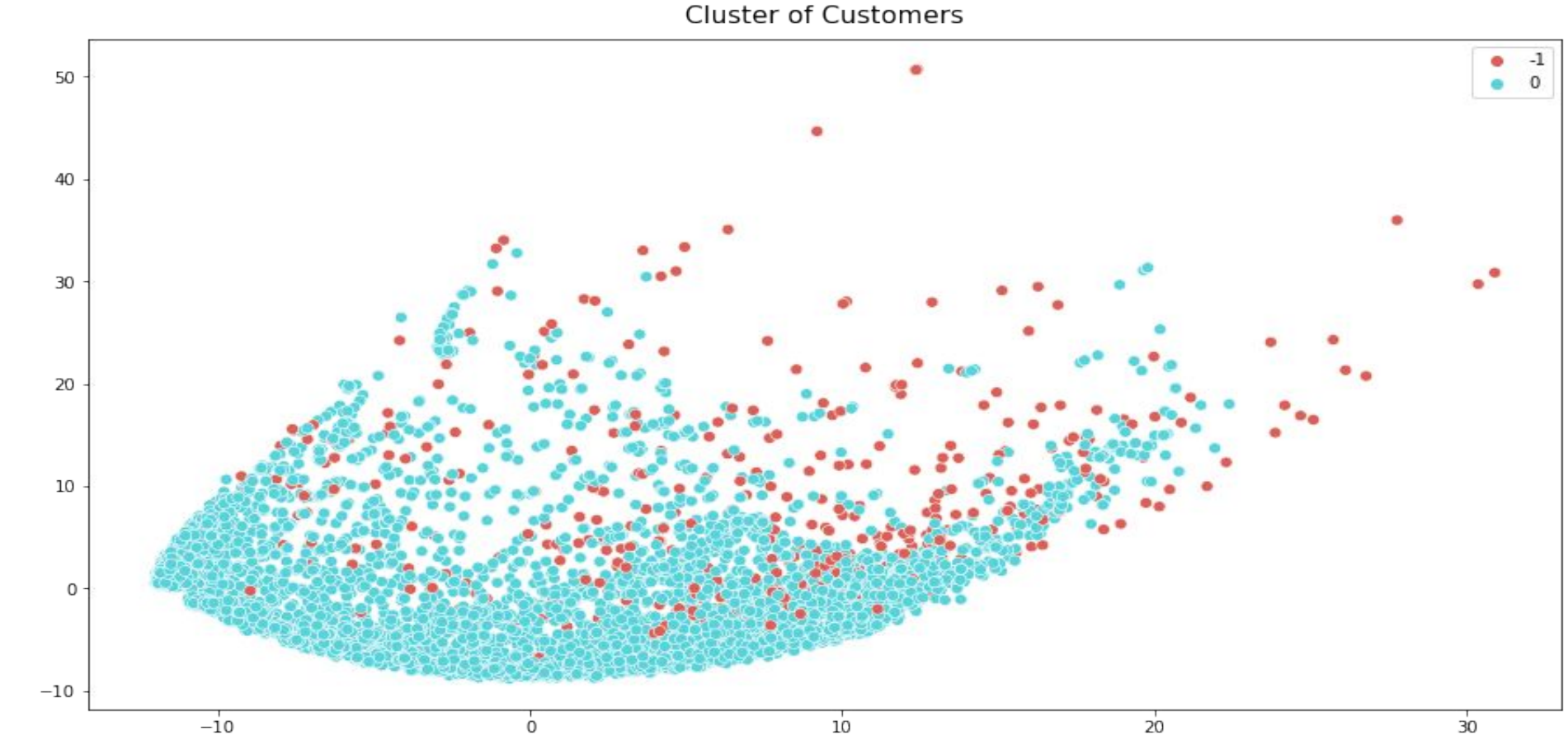
We see that the maximum curvature is around “epsilon” = 0.3. So we take epsilon = 0.3 for our further calculations.

Next to get the other required hyper-parameter, the minimum neighboring points (minpts), I calculate the 'Silhouette Score' for different values of 'MinPts' and select the one which give the maximum value.

As we can see that the maximum Silhouette score we get is for MinPts=6.

(I ran the simulation for MinPts=2 to MinPts=20).

	MinPts	SILHOUETTE SCORE
4	6	0.211219
5	7	0.207138
6	8	0.205659
7	9	0.204329
8	10	0.201015
9	11	0.200776
10	12	0.200215
11	13	0.197570
12	14	0.196241
16	18	0.187466


Cluster of Customers
-1
0
50
40
30
20
10
0
−10
−10
0
10
20
30

Inference for DBSCAN

- As we can see that DB-SCAN did not perform well in this case as it put majority of the sample into one cluster and probably considered the remaining points as noise.

- This could be due to the effects of inherent characteristics of the data itself (such as clusters with different densities) or the hyper-parameters/data processing chosen by us.

- This again illustrates the fact that not all clustering algorithms work all the time in every case.

Summary

For K-means : Number of Clusters = 3, Silhouette score =0.372

For Agglomerative Clustering: Number of Clusters = 3, Silhouette score = 0.72

For DBSCAN : Number of clusters =2, Silhouette score = 0.21

Conclusion

- In the case we studied, we see that Agglomerative clustering produced the best clustering as it produced the maximum Silhouette score out of all three which is considered as a good benchmark for clustering.
- K-means also fared decently well and gave us important insights into the data.
- DB-SCAN did not perform well at all and gave the poorest result. This could be due to some inherent characteristics of the data or our methodology of data processing/hyper-parameter selection.