Credit Card Segmentation

10.12.2020

Akshay Rahate

Contents

Problem Statement

This case requires trainees to develop a customer segmentation to define marketing strategy. The sample dataset summarizes the usage behaviour of about 9000 active credit card holders during the last 6 months. The file is at a customer level with 18 behavioural variables.

Data

The given data attributes are

credit-card-data.csv

CUST_ID Credit card holder ID

BALANCE Monthly average balance (based on daily balance averages)

BALANCE_FREQUENCY Ratio of last 12 months with balance

PURCHASES Total purchase amount spent during last 12 months

ONEOFF_PURCHASES Total amount of one-off purchases

INSTALLMENTS_PURCHASES Total amount of installment purchases

CASH_ADVANCE Total cash-advance amount

PURCHASES_ FREQUENCY - Frequency of purchases (percentage of months with at least on purchase)

ONEOFF_PURCHASES_FREQUENCY Frequency of one-off-purchases

PURCHASES_INSTALLMENTS_FREQUENCY Frequency of installment purchases

CASH_ADVANCE_ FREQUENCY Cash-Advance frequency

AVERAGE_PURCHASE_TRX Average amount per purchase transaction

CASH_ADVANCE_TRX Average amount per cash-advance transaction

PURCHASES_TRX Average amount per purchase transaction

CREDIT_LIMIT Credit limit

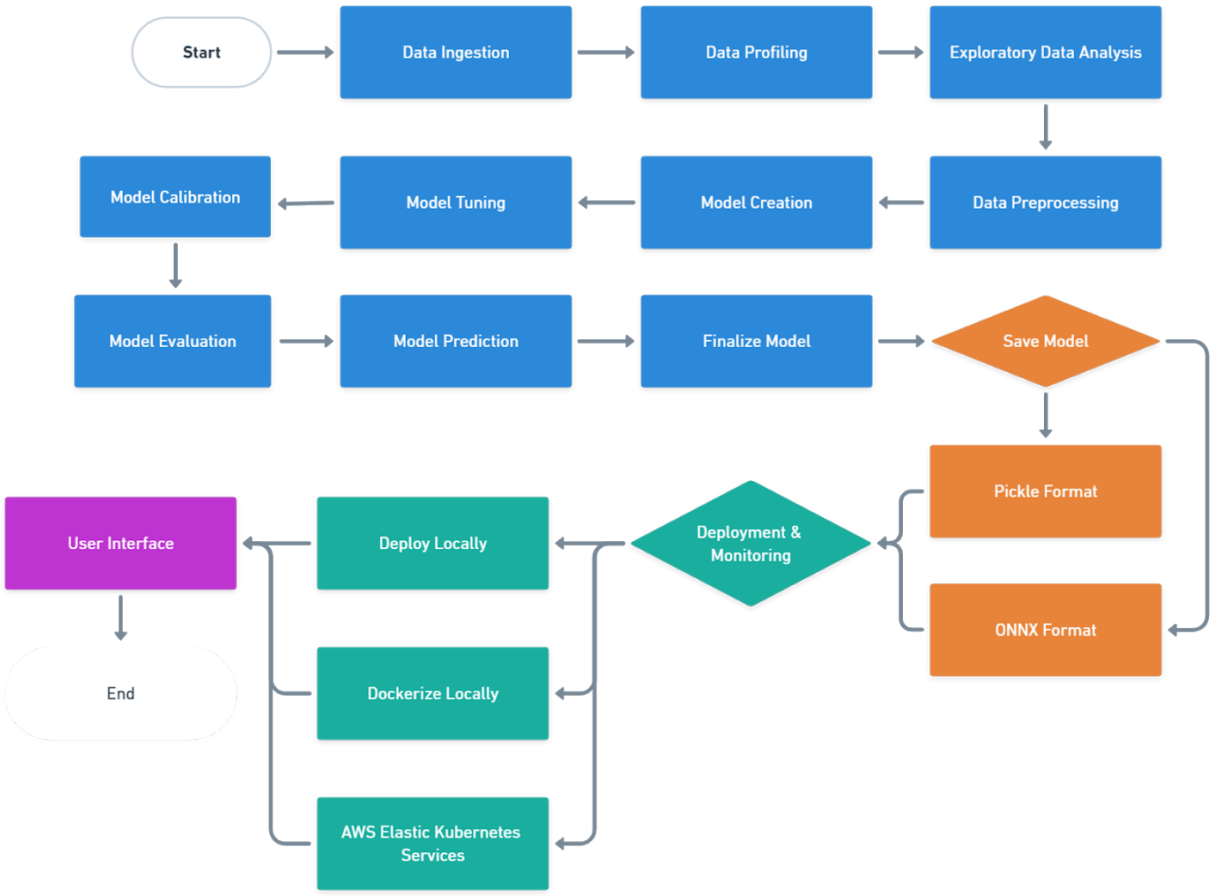
PAYMENTS - Total payments (due amount paid by the customer to decrease their statement balance) in the period

MINIMUM_PAYMENTS Total minimum payments due in the period.

PRC_FULL_PAYMENT - Percentage of months with full payment of the due statement balance

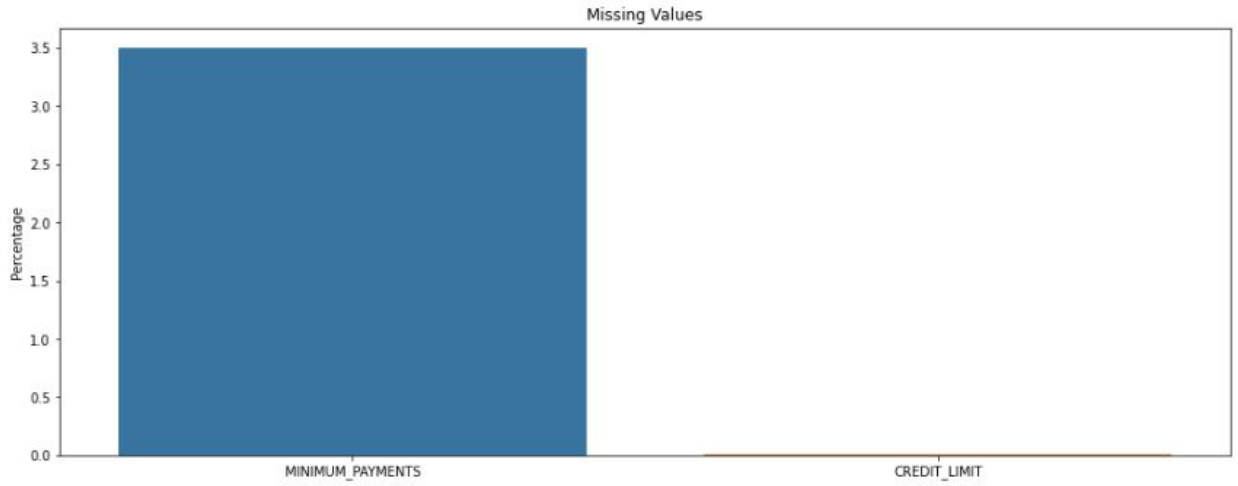
TENURE Number of months as a customer

High Level Architecture



Missing Value Analysis

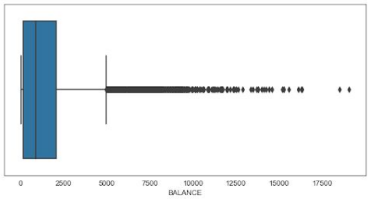
BALANCE	0
BALANCE_FREQUENCY	0
PURCHASES	0
ONEOFF_PURCHASES	0
INSTALLMENTS_PURCHASES	0
CASH_ADVANCE	0
PURCHASES_FREQUENCY	0
ONEOFF_PURCHASES_FREQUENCY	0
PURCHASES_INSTALLMENTS_FREQUENCY	0
CASH_ADVANCE_FREQUENCY	0
CASH_ADVANCE_TRX	0
PURCHASES_TRX	0
CREDIT_LIMIT	1
PAYMENTS	0
MINIMUM_PAYMENTS	313
PRC_FULL_PAYMENT	0
TENURE	0

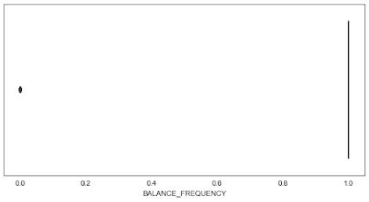


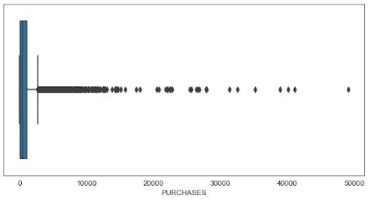
Imputing NaN values with KNN Algorithm

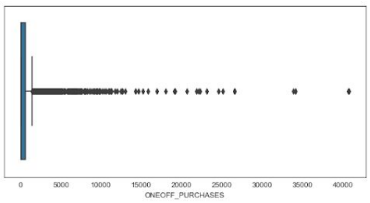
Outlier Analysis

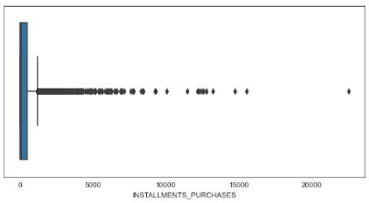
Boxplot

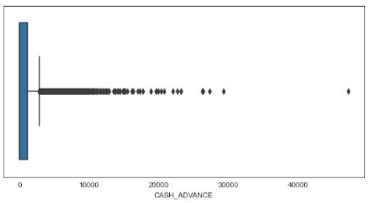


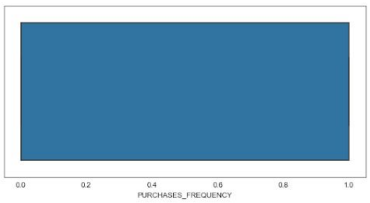


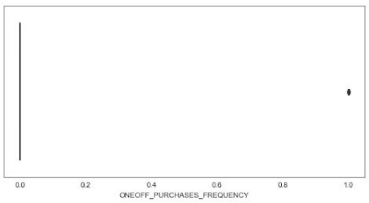


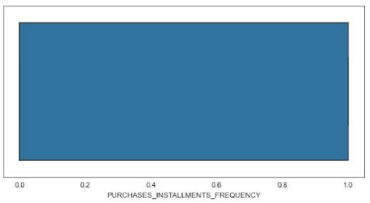


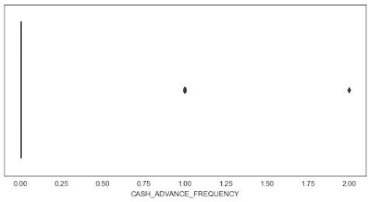


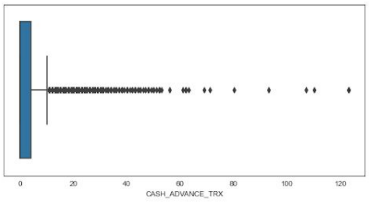


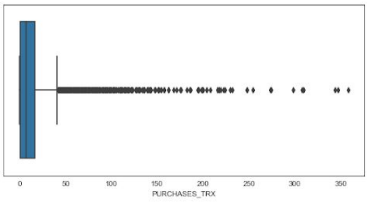


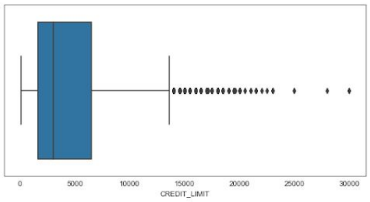


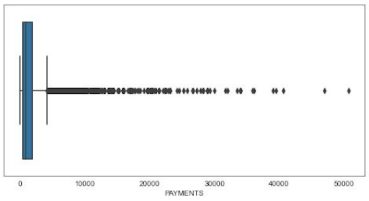


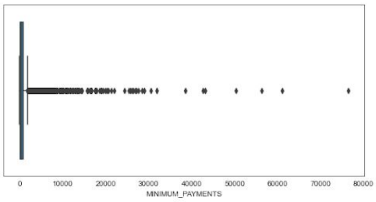


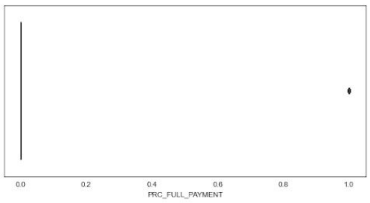


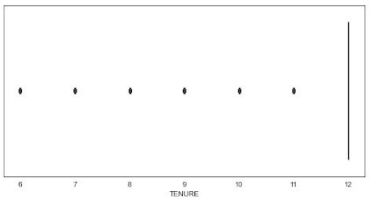




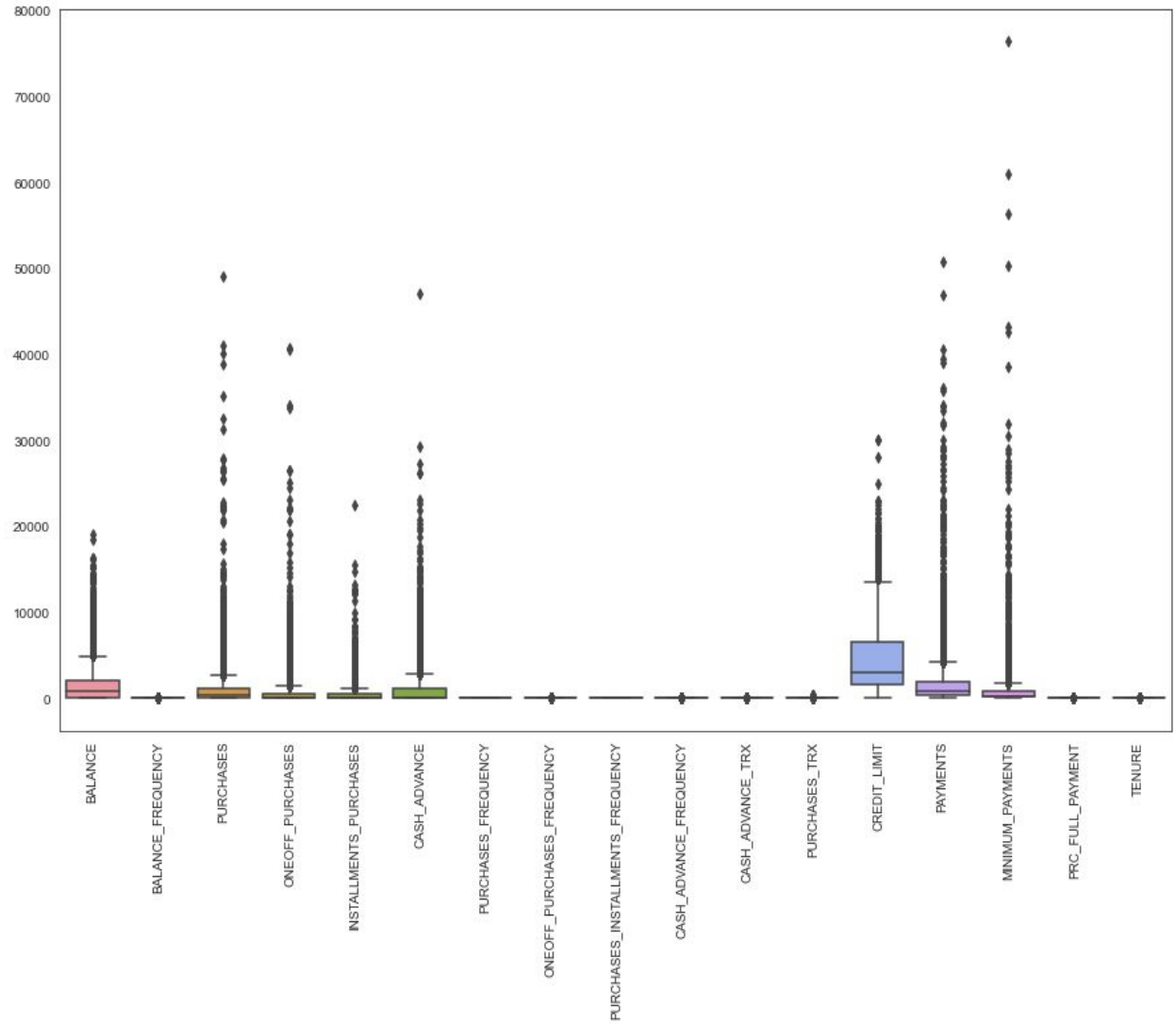




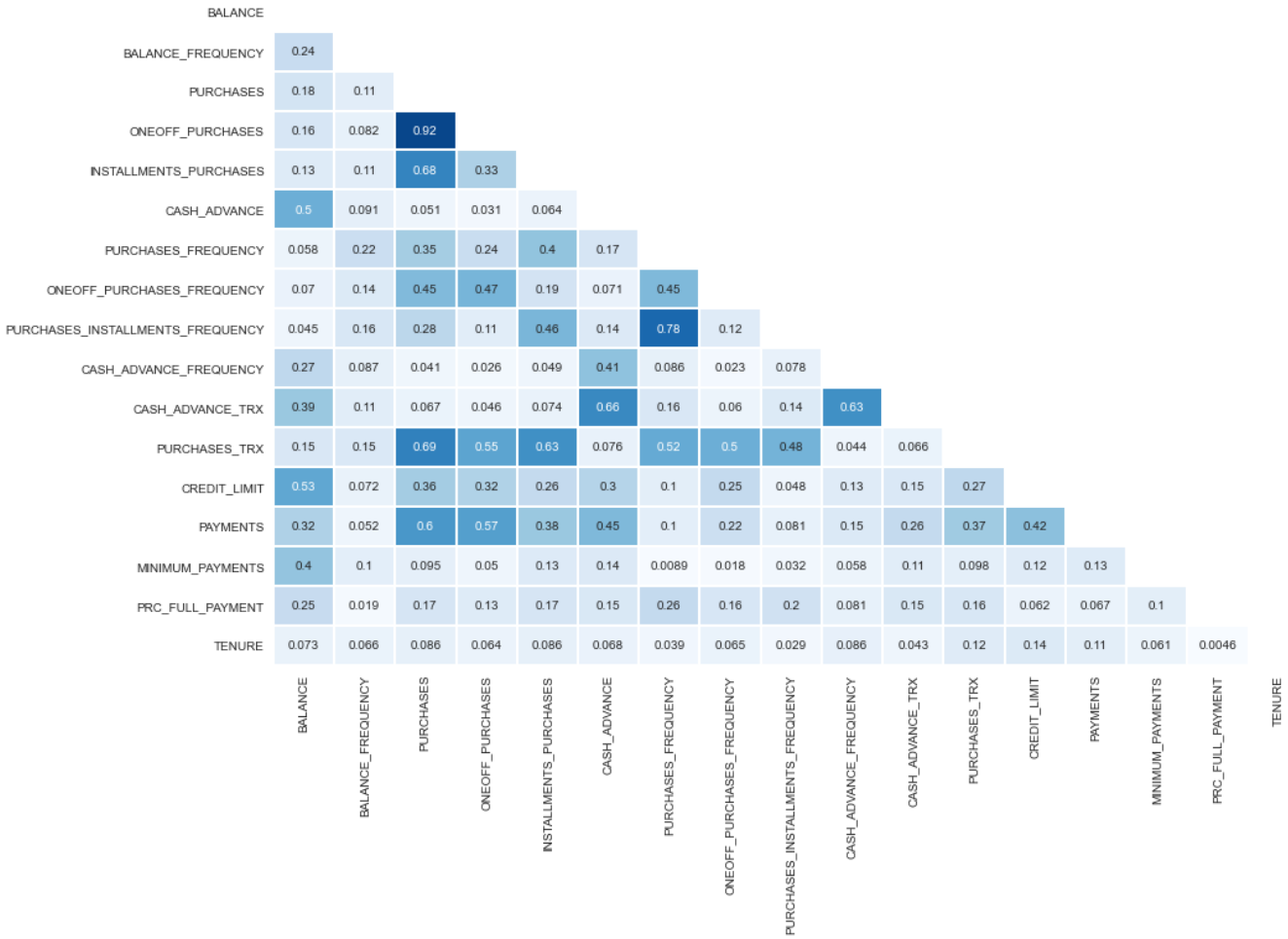




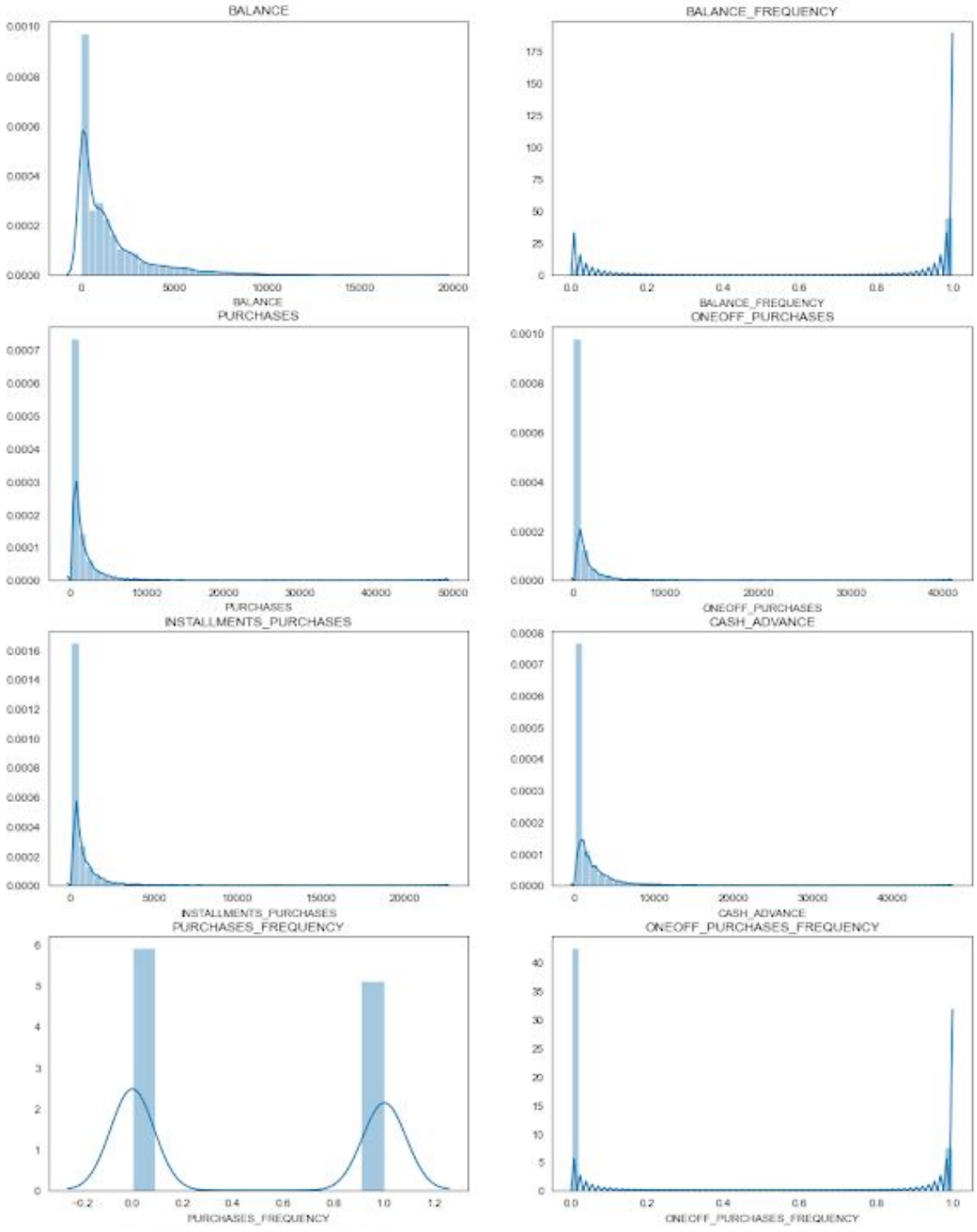
```
-----------------------------------
BALANCE_FREQUENCY
# values outlier:  0
0.00% of the total data
-----------------------------------
PURCHASES
# values outlier:  80
0.89% of the total data
-----------------------------------
ONEOFF_PURCHASES
# values outlier:  74
0.83% of the total data
-----------------------------------
INSTALLMENTS_PURCHASES
# values outlier:  79
0.88% of the total data
-----------------------------------
CASH_ADVANCE
# values outlier:  98
1.09% of the total data
-----------------------------------
PURCHASES_FREQUENCY
# values outlier:  0
0.00% of the total data
-----------------------------------
ONEOFF_PURCHASES_FREQUENCY
# values outlier:  0
0.00% of the total data
-----------------------------------
PURCHASES_INSTALLMENTS_FREQUENCY
# values outlier:  0
0.00% of the total data
-----------------------------------
CASH_ADVANCE_FREQUENCY
# values outlier:  1
0.01% of the total data
-----------------------------------
CASH_ADVANCE_TRX
# values outlier:  80
0.89% of the total data
-----------------------------------
PURCHASES_TRX
# values outlier:  95
1.06% of the total data
-----------------------------------
CREDIT_LIMIT
# values outlier:  29
0.32% of the total data
-----------------------------------
PAYMENTS
# values outlier:  92
1.03% of the total data
-----------------------------------
MINIMUM_PAYMENTS
# values outlier:  78
0.87% of the total data
-----------------------------------
PRC_FULL_PAYMENT
# values outlier:  0
0.00% of the total data
-----------------------------------
TENURE
# values outlier:  204
2.28% of the total data
```

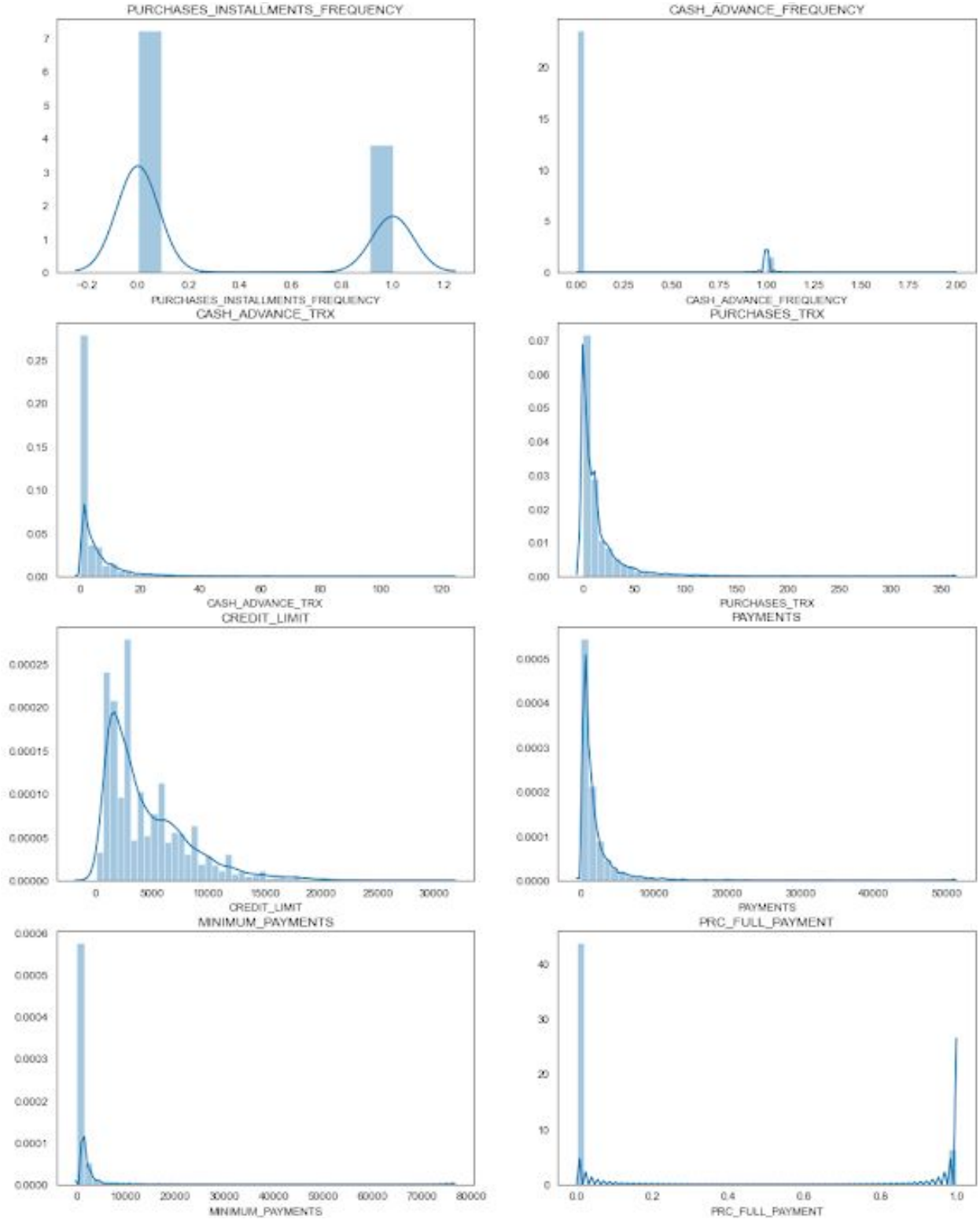

80000
70000
60000
50000
40000
30000
20000
10000
0
BALANCE
BALANCE_FREQUENCY
PURCHASES
ONEOFF_PURCHASES
INSTALLMENTS_PURCHASES
CASH_ADVANCE
PURCHASES_FREQUENCY
ONEOFF_PURCHASES_FREQUENCY
PURCHASES_INSTALLMENTS_FREQUENCY
CASH_ADVANCE_FREQUENCY
CASH_ADVANCE_TRX
PURCHASES_TRX
CREDIT_LIMIT
PAYMENTS
MINIMUM_PAYMENTS
PRC_FULL_PAYMENT
TENURE

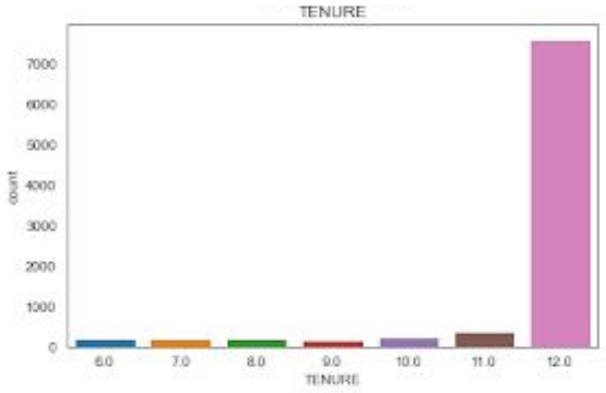
Correlation Plot



Univariate Analysis



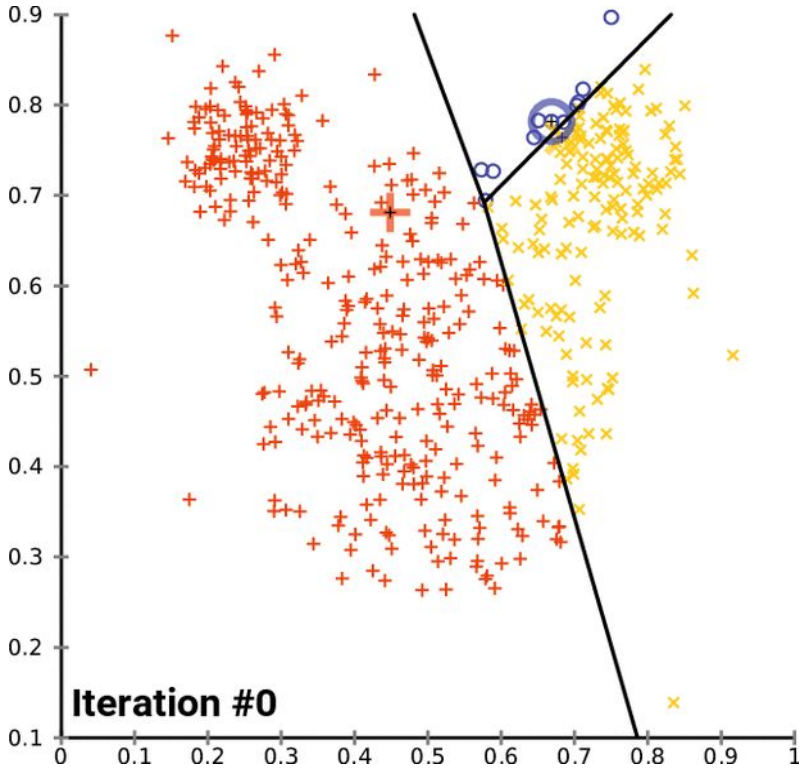

PURCHASES_INSTALLMENTS_FREQUENCY
CASH_ADVANCE_FREQUENCY
CASH_ADVANCE_TRX
PURCHASES_TRX
CREDIT_LIMIT
PAYMENTS
MINIMUM_PAYMENTS
PRC_FULL_PAYMENT



KMeans Clustering

K-means clustering is a clustering method that subdivides a single cluster or a collection of data points into K different clusters or groups.

The algorithm analyzes the data to find organically similar data points and assigns each point to a cluster that consists of points with similar characteristics. Each cluster can then be used to label the data into different classes based on the characteristics of the data.K-Means clustering works by constantly trying to find a centroid with closely held data points. This means that each cluster will have a centroid and the data points in each cluster will be closer to its centroid compared to the other centroids.



K-Means Algorithm

1. Selecting an appropriate value for K which is the number of clusters or centroids
2. Selecting random centroids for each cluster
3. Assigning each data point to its closest centroid
4. Adjusting the centroid for the newly formed cluster in step 4
5. Repeating step 4 and 5 till all the data points are perfectly organised within a cluster space.

Choosing The Right Number Of Clusters

The number of clusters that we choose for a given dataset cannot be random. Each cluster is formed by calculating and comparing the distances of data points within a cluster to its centroid. An ideal way to figure out the right number of clusters would be to calculate the Within-Cluster-Sum-of-Squares (WCSS).

WCSS is the sum of squares of the distances of each data point in all clusters to their respective centroids.

$$\text{WCSS} = \sum_{C_k}^{C_n} \left(\sum_{d_i in C_i}^{d_m} distance(d_i, C_k)^2 \right)$$

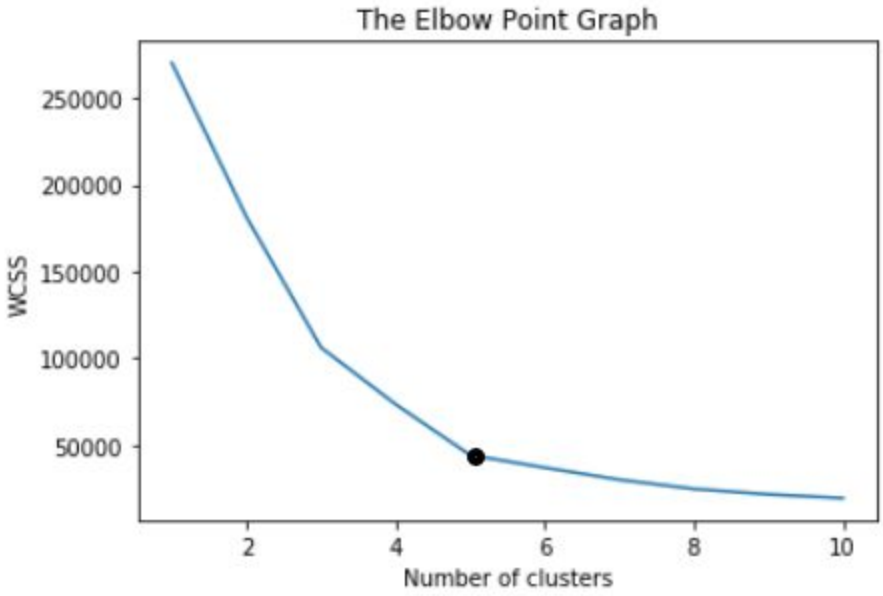
Where,
C is the cluster centroids and d is the data point in each Cluster.

The idea is to minimise the sum. Suppose there are n observation in a given dataset and we specify n number of clusters (k = n) then WCSS will become zero since data points themselves will act as centroids and the distance will be zero and ideally this forms a perfect cluster, however this doesn't make any sense as we have as many clusters as the observations. Thus there exists a threshold value for K which we can find using the Elbow point graph.

Elbow method

We can find the optimum value for K using an Elbow point graph. We randomly initialise the K-Means algorithm for a range of K values and will plot it against the WCSS for each K value.

The resulting graph would look something like what's shown below:



For the above-given graph, the optimum value for K would be 5. As we can see that with an increase in the number of clusters the WCSS value decreases. We select the value for K on the basis of the rate of decrease in WCSS. For example, from cluster 1 to 2 to 3 in the above graph we see a sudden and huge drop in WCSS. After 5 the drop is minimal and hence we chose 5 to be the optimal value for K.

Silhouette Score

Silhouette score is used to evaluate the quality of clusters created using clustering algorithms such as K-Means in terms of how well samples are clustered with other samples that are similar to each other. The Silhouette score is calculated for each sample of different clusters. To calculate the Silhouette score for each observation/data point, the following distances need to be found out for each observations belonging to all the clusters:

- Mean distance between the observation and all other data points in the same cluster. This distance can also be called a mean intra-cluster distance. The mean distance is denoted by ‘a’
- Mean distance between the observation and all other data points of the next nearest cluster. This distance can also be called a mean nearest-cluster distance. The mean distance is denoted by ‘b’

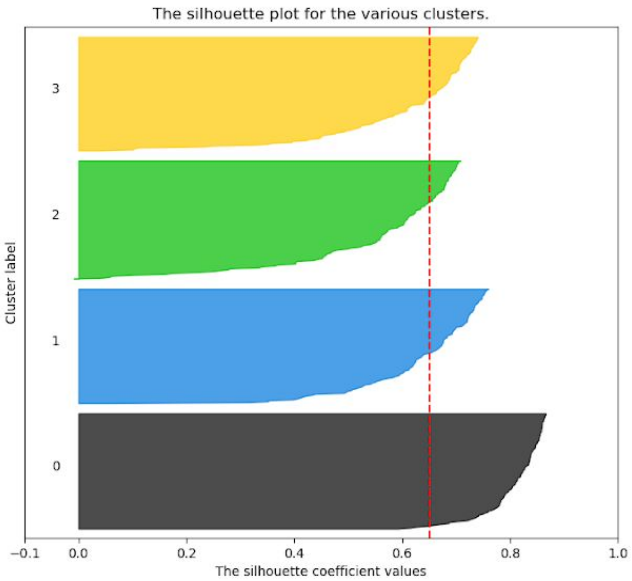
Silhouette score, S, for each sample is calculated using the following formula:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

The value of the Silhouette score varies from -1 to 1. If the score is 1, the cluster is dense and well-separated than other clusters. A value near 0 represents overlapping clusters with samples very close to the decision boundary of the neighboring clusters. A negative score [-1, 0] indicates that the samples might have got assigned to the wrong clusters.

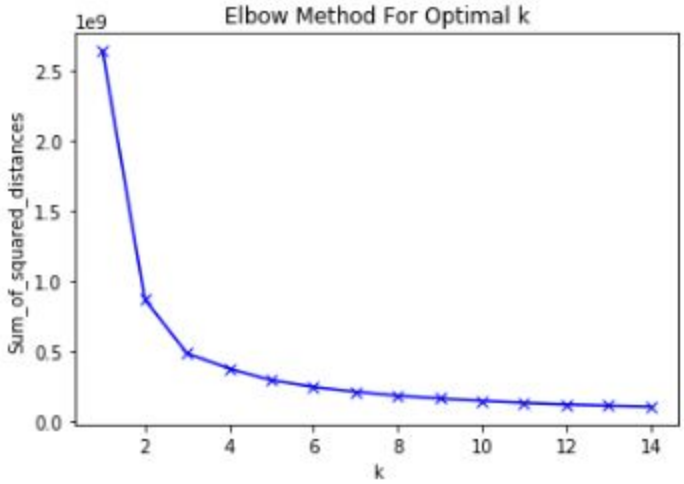
Here is the summary of what you learned in this post in relation to silhouette score concepts:

- Silhouette score for a set of sample data points is used to measure how dense and well-separated the clusters are.
- Silhouette score takes into consideration the intra-cluster distance between the sample and other data points within the same cluster (a) and inter-cluster distance between the sample and the next nearest cluster (b).
- The silhouette score falls within the range [-1, 1].
- The silhouette score of 1 means that the clusters are very dense and nicely separated. The score of 0 means that clusters are overlapping. The score of less than 0 means that data belonging to clusters may be wrong/incorrect.
- The silhouette plots can be used to select the most optimal value of the K (no. of cluster) in K-means clustering.
- The aspects to look out for in Silhouette plots are cluster scores below the average silhouette score, wide fluctuations in the size of the clusters, and also the thickness of the silhouette plot.



Inertia

It is defined as the mean squared distance between each instance and its closest centroid. Logically, as per the definition lower the inertia better the model. In general, the KMeans package of sklearn runs the algorithm for 'n_init' number of times and chooses the one having the lowest inertia. This approach has a limitation, as the number of clusters increases, closest will be the clusters from the centroids and lower will be the inertia. Therefore, in this case, it is a good idea to plot the graph of inertia as the number of clusters increase. Elbow rule is used in order to find the optimal number of clusters. As depicted in the following diagram, the curve looks like a hand and the number of clusters to be chosen over there should be equal to 3 as after that curve reaches a plateau.



BIRCH Clustering

Clustering algorithms like K-means clustering do not perform clustering very efficiently and it is difficult to process large datasets with a limited amount of resources (like memory or a slower CPU). So, regular clustering algorithms do not scale well in terms of running time and quality as the size of the dataset increases. This is where BIRCH clustering comes in.

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is a clustering algorithm that can cluster large datasets by first generating a small and compact summary of the large dataset that retains as much information as possible. This smaller summary is then clustered instead of clustering the larger dataset.

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)

- It is a scalable clustering method.
- Designed for very large data sets

- Only one scan of data is necessary
- It is based on the notation of CF (Clustering Feature) a CF Tree.
- CF tree is a height balanced tree that stores the clustering features for a hierarchical clustering.
- Cluster of data points is represented by a triple of numbers (N,LS,SS) Where

  N= Number of items in the sub cluster

  LS=Linear sum of the points

  SS=sum of the squared of the points

A CF Tree structure is given as below:

Each non-leaf node has at most B entries.

Each leaf node has at most L CF entries which satisfy threshold T, a maximum diameter of radius

P(page size in bytes) is the maximum size of a node

Compact: each leaf node is a subcluster, not a data point

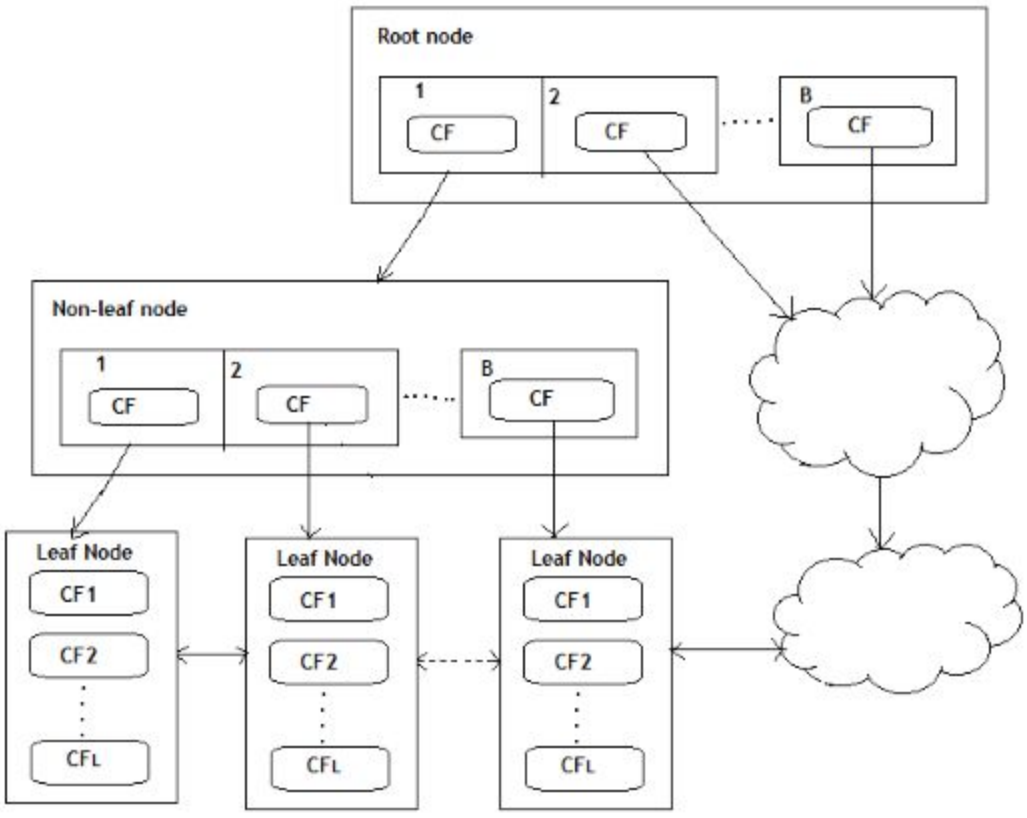


Fig: A CF tree structure

Basic Algorithm:

- **Phase 1**: Load data into memory

  Scan DB and load data into memory by building a CF tree. If memory is exhausted, rebuild the tree from the leaf node.

- **Phase 2**: Condense data

  Resize the data set by building a smaller CF tree

  Remove more outliers

  Condensing is optional

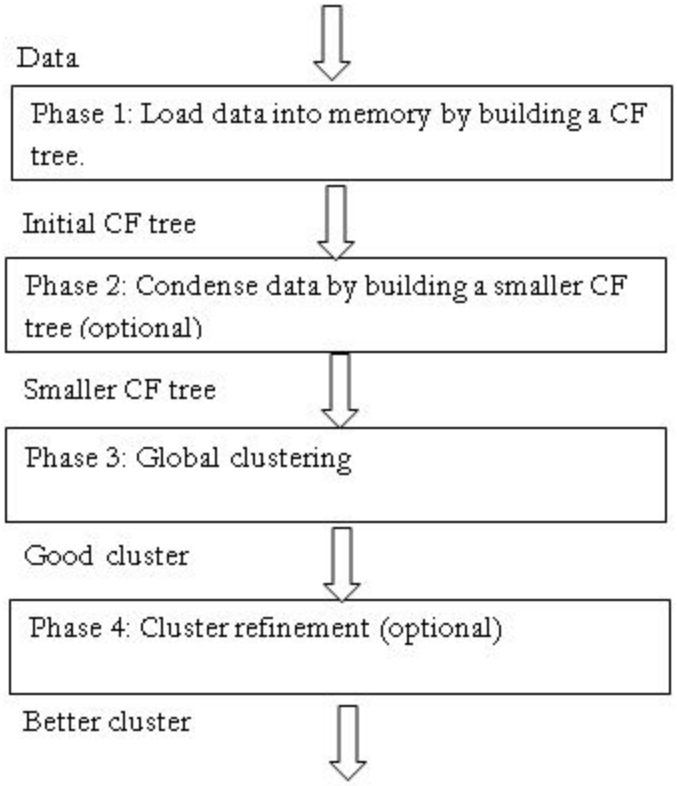
- **Phase 3**: Global clustering

  Use existing clustering algorithm (e.g. KMeans, HC) on CF entries

- **Phase 4**: Cluster refining

  Refining is optional

  Fixes the problem with CF trees where same valued data points may be assigned to different leaf entries.

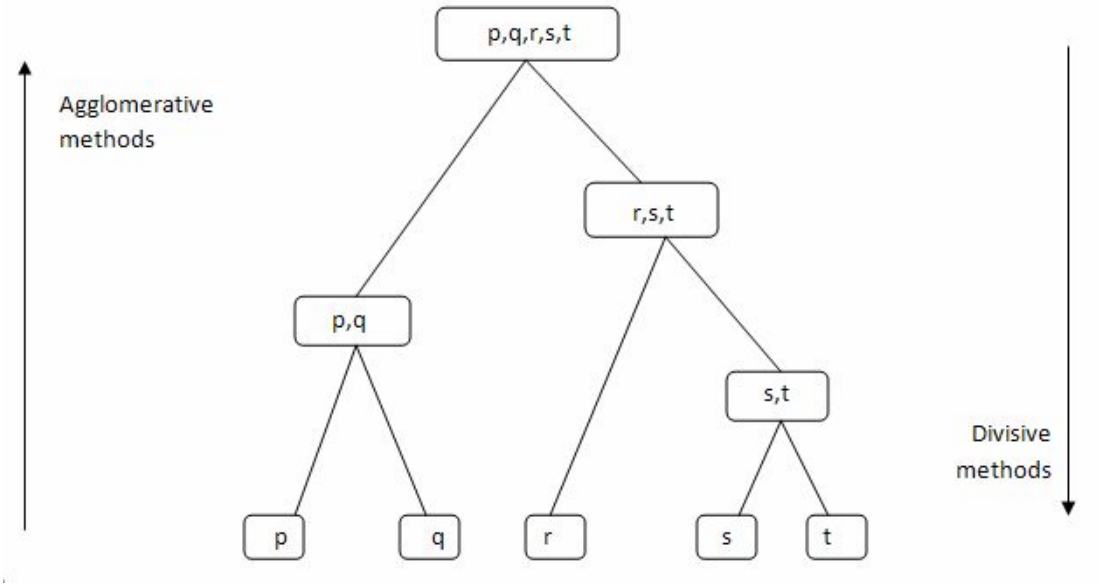


Agglomerative Clustering

Hierarchical clustering algorithms group similar objects into groups called clusters. There are two types of hierarchical clustering algorithms:

Agglomerative — Bottom up approach. Start with many small clusters and merge them together to create bigger clusters.

Divisive — Top down approach. Start with a single cluster than break it up into smaller clusters.



Recursively merges the pair of clusters that minimally increases a given linkage distance.

The agglomerative clustering is the most common type of hierarchical clustering used to group objects in clusters based on their similarity. It's also known as AGNES (Agglomerative Nesting). The algorithm starts by treating each object as a singleton cluster. Next, pairs of clusters are successively merged until all clusters have been merged into one big cluster containing all objects. The result is a tree-based representation of the objects, named dendrogram.

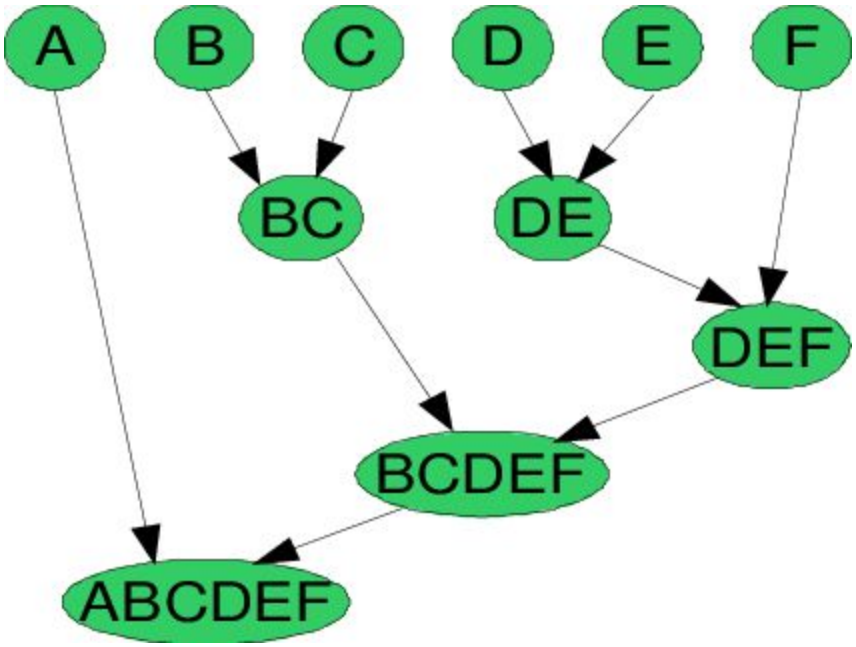
The basic algorithm of Agglomerative is straight forward.

- Compute the proximity matrix
- Let each data point be a cluster
- Repeat: Merge the two closest clusters and update the proximity matrix
- Until only a single cluster remains

Key operation is the computation of the proximity of two clusters

To understand better let's see a pictorial representation of the Agglomerative Hierarchical clustering Technique. Let's say we have six data points {A,B,C,D,E,F}.

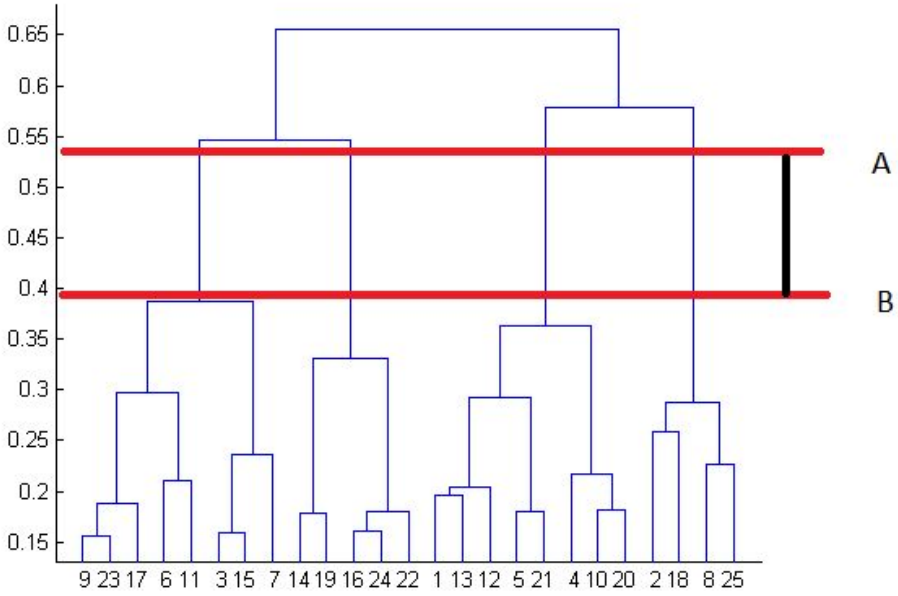
- **Step- 1**: In the initial step, we calculate the proximity of individual points and consider all the six data points as individual clusters as shown in the image below.



- **Step- 2**: In step two, similar clusters are merged together and formed as a single cluster. Let's consider B,C, and D,E are similar clusters that are merged in step two. Now, we're left with four clusters which are A, BC, DE, F.
- **Step- 3**: We again calculate the proximity of new clusters and merge the similar clusters to form new clusters A, BC, DEF.
- **Step- 4**: Calculate the proximity of the new clusters. The clusters DEF and BC are similar and merged together to form a new cluster. We're now left with two clusters A, BCDEF.
- **Step- 5**: Finally, all the clusters are merged together and form a single cluster.

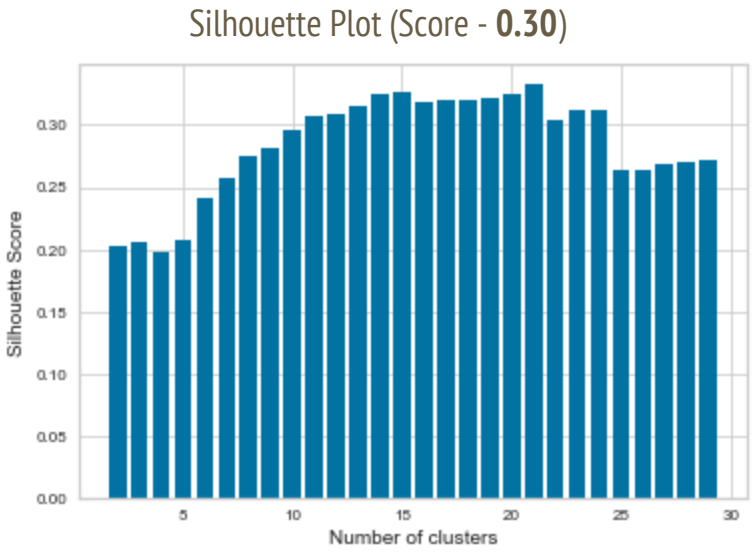
The Hierarchical clustering Technique can be visualized using a **Dendrogram**. A **Dendrogram** is a tree-like diagram that records the sequences of merges or splits.

- Determine the largest vertical distance that doesn't intersect any of the other clusters
- Draw a horizontal line at both extremities
- The optimal number of clusters is equal to the number of vertical lines going through the horizontal line



Observation

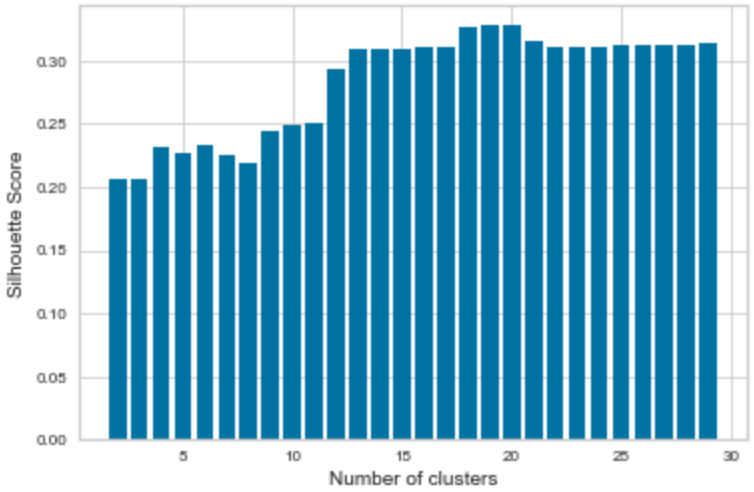
Agglomerative Clustering



We choose 10 as the optimal number of clusters as it gives the highest silhouette score. The cluster errors are almost similar and also, the slope of the line is almost constant as well.

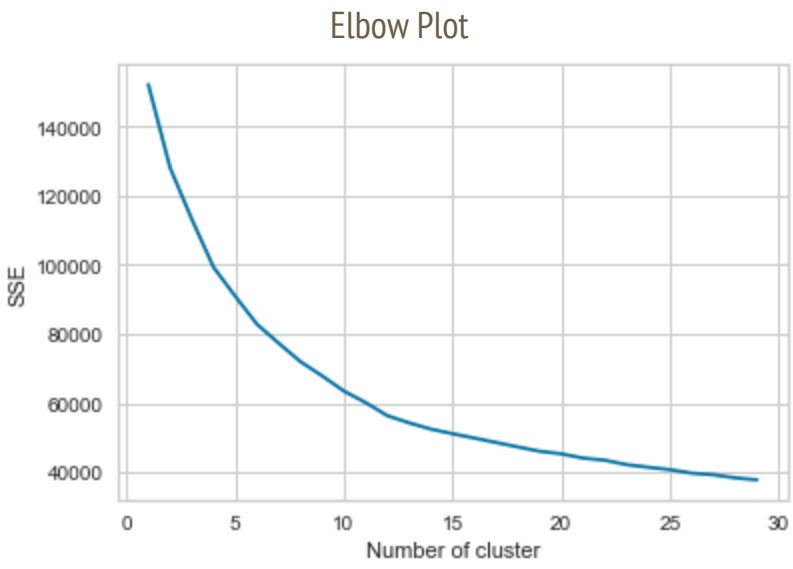
Birch Clustering

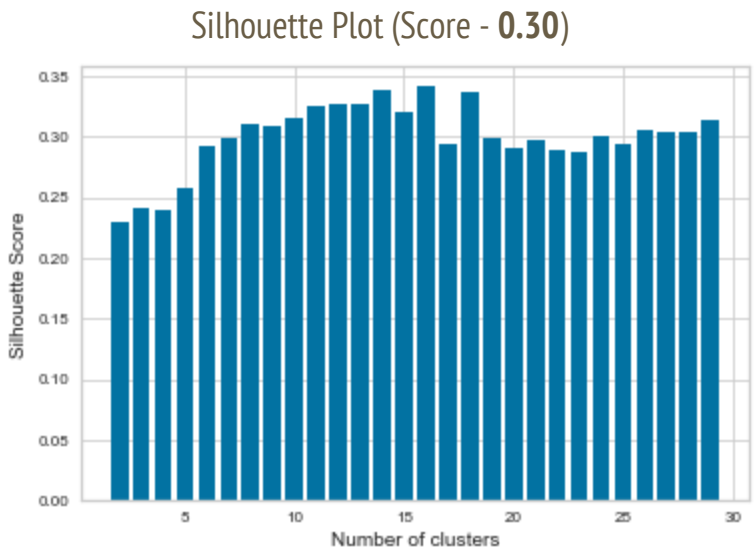
Silhouette Plot (Score - **0.31**)

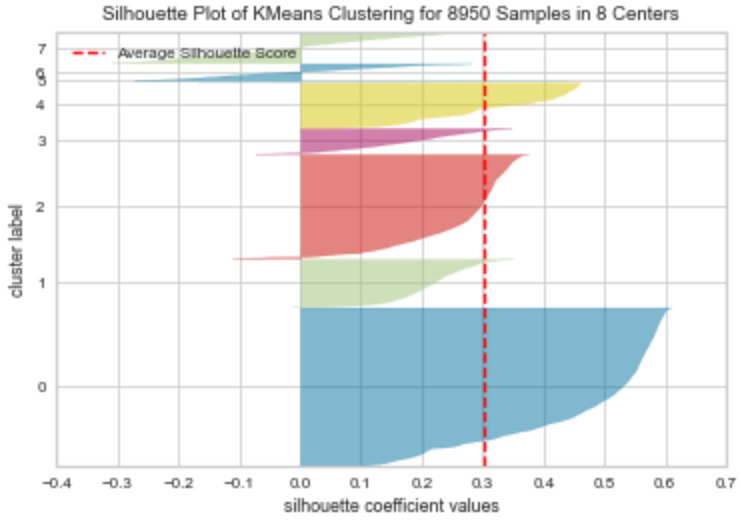


We choose 13 as the optimal number of clusters as it gives the highest silhouette score. The cluster errors are almost similar and also, the slope of the line is almost constant as well.

KMeans Clustering







We choose 8 as the optimal number of clusters as it gives the highest silhouette score and also comparable inertia value. The cluster errors are almost similar and also, the slope of the line is almost constant as well.

Key Performance Indicator

BALANCE monthly average purchase

CASH_ADVANCE cash advance amount

ONEOFF_PURCHASES Total amount of one-off purchases

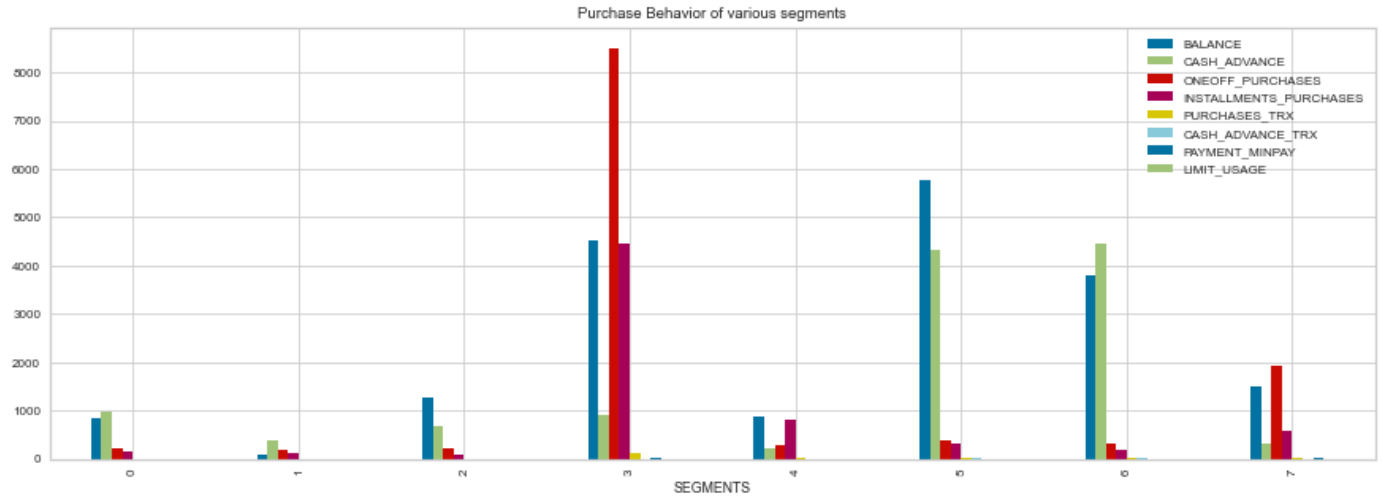
INSTALLMENTS_PURCHASES Total amount of installment purchases

PURCHASES_TRX average amount per purchase

CASH_ADVANCE_TRX cash advance transaction

BALANCE / CREDIT_LIMIT limit usage (balance to credit limit ratio)

PAYMENTS / MINIMUM_PAYMENTS payments to minimum payments ratio



Conclusion

Cluster 3

This cluster shows significantly high ONEOFF_PURCHASES and INSTALLMENTS_PURCHASES. Also has high BALANCE. Customers in this cluster have very high spending capacity. Targeted marketing campaigns could be term useful.

Cluster 5 & 6

This cluster shows significantly high BALANCE & CASH_ADVANCE. Potential Customers with the ability to spend more.

Cluster 7

This cluster shows average spending capability. Marketing campaigns may be fruitful.

Cluster 0, 1, 2 and 4

This cluster shows below average spending capability. Any marketing campaign will be a futile attempt.