Final Assignment - Unsupervised Learning - IBM Machine Learning Professional Certificate

Main Objective

A **bank** has asked us to help them to create a **segmentation** of their **credit card customers**. This will help the company to create specific policies for each kind of customer in order to improve the service for each one. The better the service is the greater customer satisfaction is and so the probability to retain the customer is larger. Retaining the customers as much as possible is the base for a stable a sustainable growth that would lead to a higher profit.

Thus, our **main goal** will be to find the best **clustering model** to perform the best possible segmentation of the customers.

Brief Description of the Data

The bank has provided us with a dataset describing the usage behaviour of their customers. The data set consists of **8950 rows** and **18 columns**. Cust_ID column is related to the customer Id, we can see that there are 8950 unique values for this feature so each row represent a different customer.

The columns are:

- **CUST_ID** : 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** : Amount of purchases made from account
- **ONEOFF_PURCHASES** : Maximum purchase amount done in one-go
- **INSTALLMENTS_PURCHASES** : Amount of purchase done in instalment
- **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)
- **ONEOFF_PURCHASES_FREQUENCY** : How frequently Purchases are happening in one-go (1 = frequently purchased, 0 = not frequently purchased)
- **PURCHASES_INSTALLMENTS_FREQUENCY** : How frequently purchases in instalments 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 amount of payments made by user
- **PRC_FULL_PAYMENT** : Percent of full payment paid by user

- **TENURE** : Tenure of credit card service for user

All features are numeric except for the CUST_ID feature which is an Object. We can also see that there are **almost no missing values** for any feature except for MINIMUM_PAYMENTS (see **figure 1**).

Also, after having a look on the statistical summary of the data set, we can observe that some of the different **features have different scales** (see **figure 2**). Furthermore, an initial look on the pair plot shows that many of the **features are skewed** (see **figure 3**), so maybe they will need some kind of transformation, and we can observe that some of these features could have some kind of correlation.

Acknowledgment: The data set used here was posted by Arjun Bhasin on Kaggle (https://www.kaggle.com/arjunbhasin2013/ccdata)

#	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

Figure 1. Table showing feature types and non-null values

	count	mean	std	min	25%	50%	75%	max
BALANCE	8950.0	1564.474828	2081.531879	0.000000	128.281915	873.385231	2054.140036	19043.13856
BALANCE_FREQUENCY	8950.0	0.877271	0.236904	0.000000	0.888889	1.000000	1.000000	1.00000
PURCHASES	8950.0	1003.204834	2136.634782	0.000000	39.635000	361.280000	1110.130000	49039.57000
ONEOFF_PURCHASES	8950.0	592.437371	1659.887917	0.000000	0.000000	38.000000	577.405000	40761.25000
INSTALLMENTS_PURCHASES	8950.0	411.067645	904.338115	0.000000	0.000000	89.000000	468.637500	22500.00000
CASH_ADVANCE	8950.0	978.871112	2097.163877	0.000000	0.000000	0.000000	1113.821139	47137.21176
PURCHASES_FREQUENCY	8950.0	0.490351	0.401371	0.000000	0.083333	0.500000	0.916667	1.00000
ONEOFF_PURCHASES_FREQUENCY	8950.0	0.202458	0.298336	0.000000	0.000000	0.083333	0.300000	1.00000
PURCHASES_INSTALLMENTS_FREQUENCY	8950.0	0.364437	0.397448	0.000000	0.000000	0.166667	0.750000	1.00000
CASH_ADVANCE_FREQUENCY	8950.0	0.135144	0.200121	0.000000	0.000000	0.000000	0.222222	1.50000
CASH_ADVANCE_TRX	8950.0	3.248827	6.824647	0.000000	0.000000	0.000000	4.000000	123.00000
PURCHASES_TRX	8950.0	14.709832	24.857649	0.000000	1.000000	7.000000	17.000000	358.00000
CREDIT_LIMIT	8949.0	4494.449450	3638.815725	50.000000	1600.000000	3000.000000	6500.000000	30000.00000
PAYMENTS	8950.0	1733.143852	2895.063757	0.000000	383.276166	856.901546	1901.134317	50721.48336
MINIMUM_PAYMENTS	8637.0	864.206542	2372.446607	0.019163	169.123707	312.343947	825.485459	76406.20752
PRC_FULL_PAYMENT	8950.0	0.153715	0.292499	0.000000	0.000000	0.000000	0.142857	1.00000
TENURE	8950.0	11.517318	1.338331	6.000000	12.000000	12.000000	12.000000	12.00000

Figure 2. Data stats summary

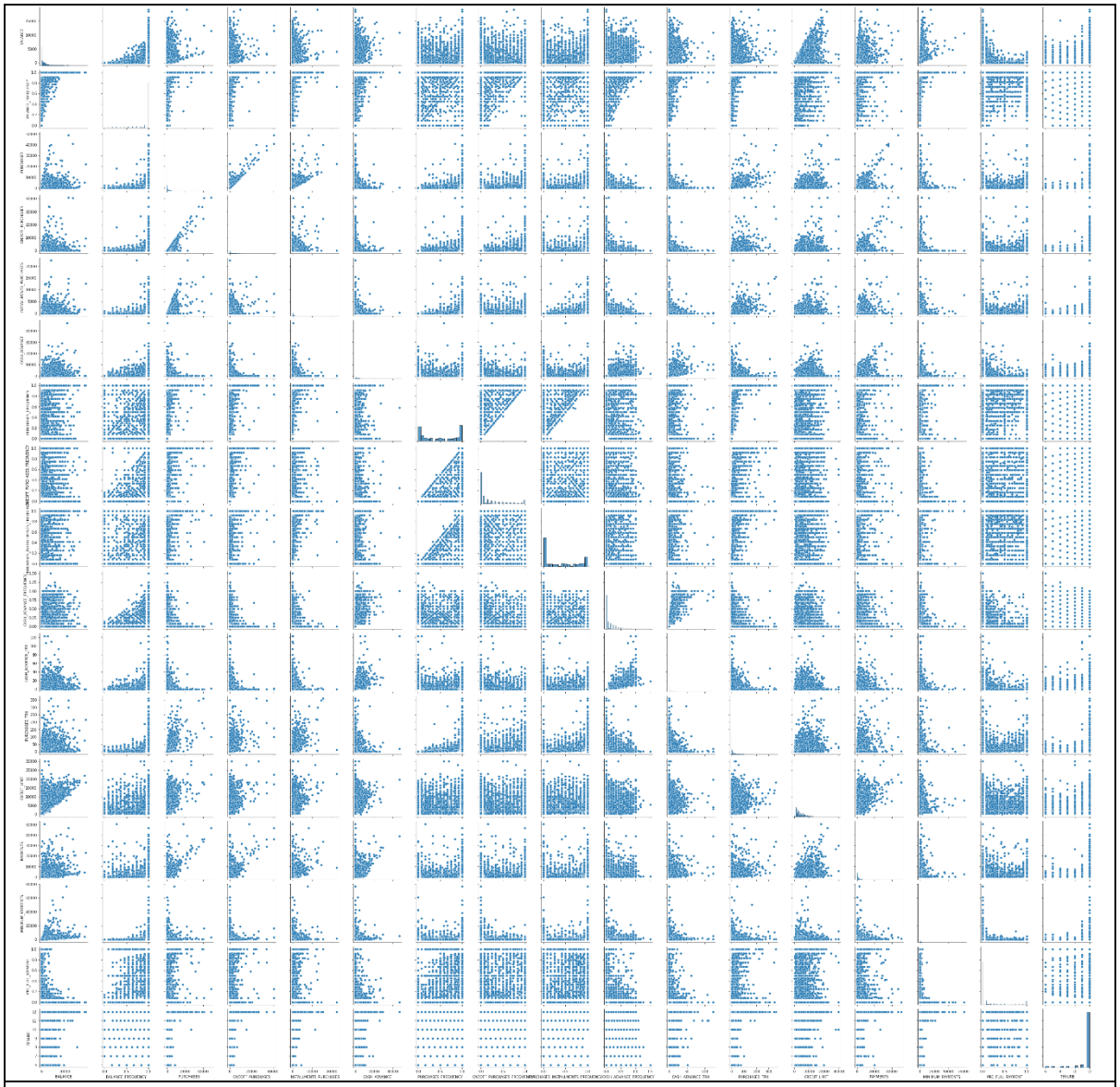

Figure 3. Pair plot showing feature distributions and correlations.

Summary of EDA. Cleaning and feature engineering

First action to take will be to **drop** the **CUST_ID** feature since there is no significant information for this study. So, we just **keep numeric features and no feature encoding** will be needed.

As we already know from the data description, we will need to study which of these **features are highly skewed**, so **need a logarithmic transformation**. Also, from the data description we know that there are non-null on feature MINIMUM_PAYMENT so we need to **deal with missing data**.

Furthermore, we need to **study the correlation** between the different features in order **to delete any duplicity** that could lead to wrong models.

Also, we need to **check** if there are any **outliers**, since some of the models doesn't deal well with them. It could be interesting to see how the outliers affect to the clustering.

Finally, we will **scale** our **data** since the majority of clustering algorithms work based on measuring distance. Thus, we will avoid wrong models.

1) The most of the features are highly skewed except from PURCHASES_FREQUENCY and PURCHASES_INSTALLMENTS_FREQUENCY. On the skewed ones we perform a logarithmic transformation to minimize the skew except for TENURE, CASH_ADVANCE_FREQUENCY, BALANCE_FREQUENCY and ONEOFF_PURCHASES_FREQUENCY for which the logarithmic transformation doesn't improve the distribution (see **figure 4**).

2) Some columns (CASH_ADVANCE_TRX, CREDIT_LIMIT, PAYMENTS) presents outliers. So, we clean them by dropping the outliers. We also drop the null rows from the MINIMUM_PAYMENT.

3) After studying the correlation, we drop the highly correlated features PURCHASES_INSTALLMENTS_FREQUENCY, CASH_ADVANCE_FREQUENCY, PURCHASES_TRX and PURCHASES_FREQUENCY to avoid duplicity and problems on our models. Besides that, a lower dimensionality will improve the model performance (see **figure 5**).

4) Finally, we standardize our data to have all features in the same scale.

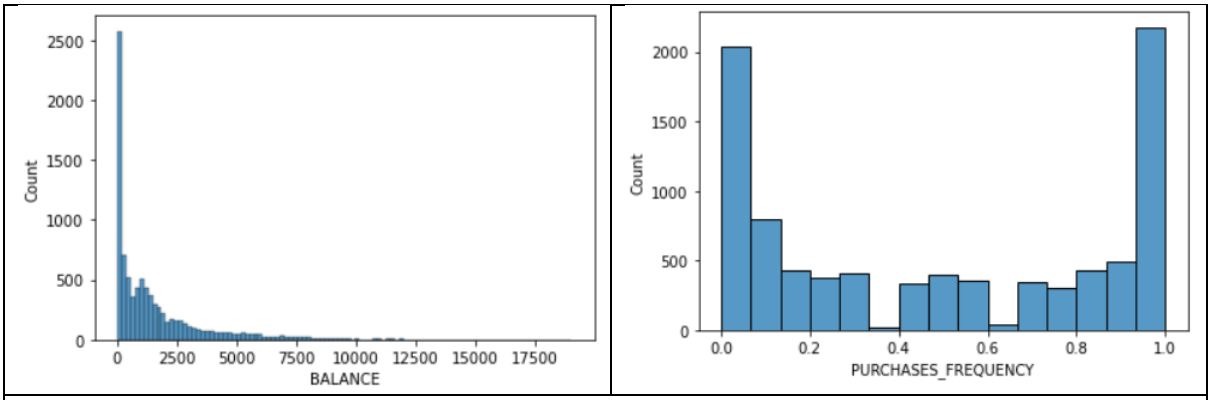

Figure 4. Example of feature distribution before transformation. The left one is an example of skewed feature and the right one no skewed.

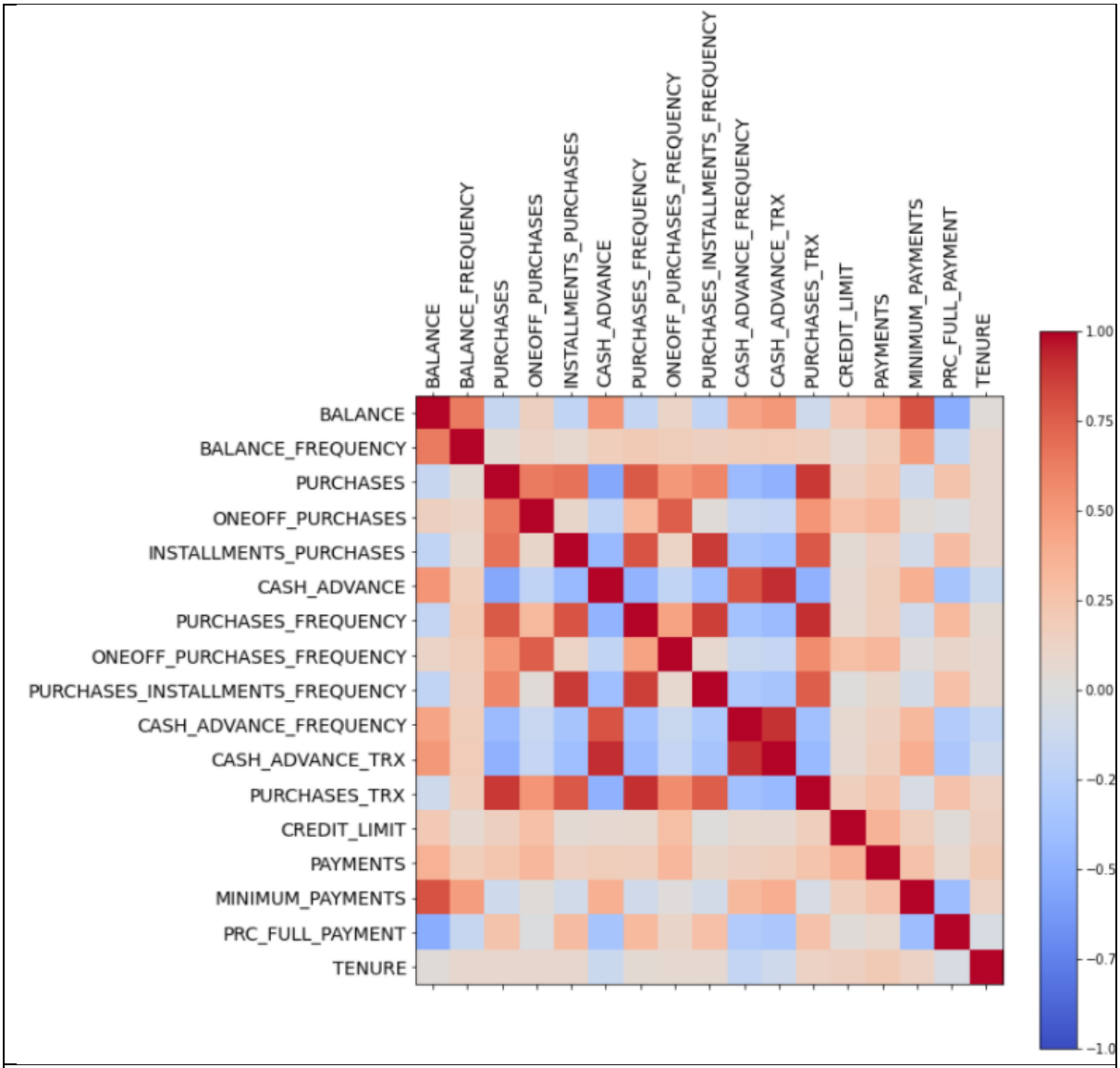

Figure 5. Heatmap with correlations between the features.

Summary of training at least three different clustering models

Our main goal is to perform the best segmentation as possible. There are many algorithms for clustering and each one has their pros and cons. We will train different models and try to figure out which one can fit better to our purpose. The model used in this study are **K-Means, Mean shift and DBSCAN**.

K-Means

First, we try one of the **most common** algorithms for clustering. The K-means. We need to **specify a priori the number of clusters** but since we don't have any information from the business with the expected clusters, we **should try to find first the right K value**. For this purpose, we train different kmean models with different k values and keep the inertia. The

inertia will drop continuously when increasing the k value, but an abrupt change in the slope of k versus inertia can give us a hint (see **figure 6**).

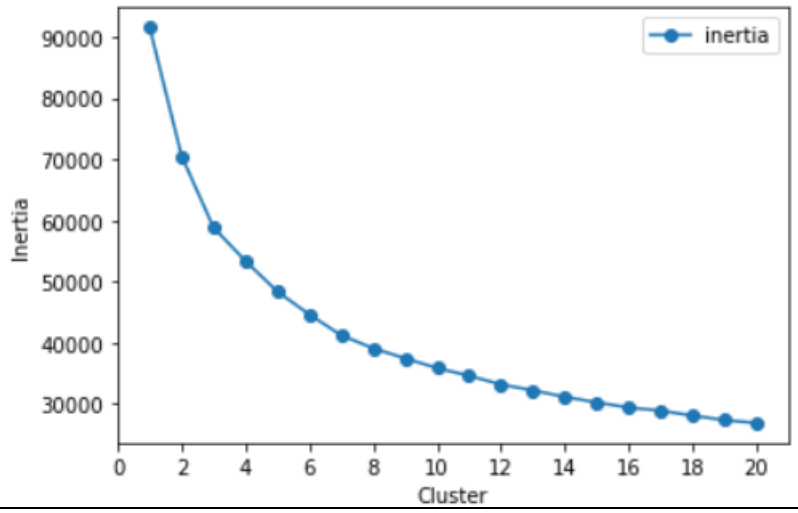

Figure 6. K vs Inertia

Unfortunately, figure 6 **doesn't help enough to decide the proper value**. We could try k=3 or 4. But we cannot by sure which is the proper k value.

Mean Shift

We don't have any specification about the number of different groups that the bank will need to cluster the customers and the study of k versus inertia for k-means didn't help. So, we decide to use a model that **don't need to specify it a priori**. This is the case of Mean Shift. The only hyperparameter that we need to pass to this model is the bandwidth, but we can estimate it. **The result for this model is not satisfactory, since the number of clusters is equal to 1, which is not useful at all for customer segmentation** (see **figure 7**).

```
# The following bandwidth can be automatically detected using
bandwidth = estimate_bandwidth(data_with_out_2)

ms = MeanShift(bandwidth=bandwidth, bin_seeding=True)
ms.fit(data_without_outliers_2)
labels = ms.labels_
cluster_centers = ms.cluster_centers_

labels_unique = np.unique(labels)
n_clusters_ = len(labels_unique)

print("number of estimated clusters : %d" % n_clusters_)
```

```
number of estimated clusters : 1
```

Figure 7. Code showing the Main shift model.

DBSCAN

Finally, we try with a second method that **do not need to specify the number of clusters**, the DBSCAN. In this case we have two hyperparameters, **epsilon and number of data points, which basically defined the density**. It's a little difficult to find these values, but in this case, we can **use the silhouette score to figure out which values are best**.

After many tries, we found that **epsilon** equals to **1.7** and number of **minimum samples** equals to **5**, and the **minimum silhouette score of all the tries (-0.043)** leads to a model with **three different clusters and 370 data points were catalogued as out of any clusters**.

It is **clear** that we have to **discard Mean shift** model since reduce all the sample to a single cluster which is not useful at all for our purpose of finding different customer clusters to apply different policies on them. Then, we also see that it **wasn't clear which number of clusters should be taken into account after training the k-mean models** so will make difficult our analysis. If we make the assumption that all clusters should have, more less, the same density we can **definitely recommend the DBSCAN to cluster the customers. The advantage of this model is that was able to find the number of clusters. Also, we know that DBSCAN deal well with complex shapes**.

Key Findings

As we saw on the previous section, the DBSCAN model group the customers in **three different clusters**. By reducing the dimensionality of our data set using **Principal Component Analysis we can visualize these three clusters** (**see figure 8**)

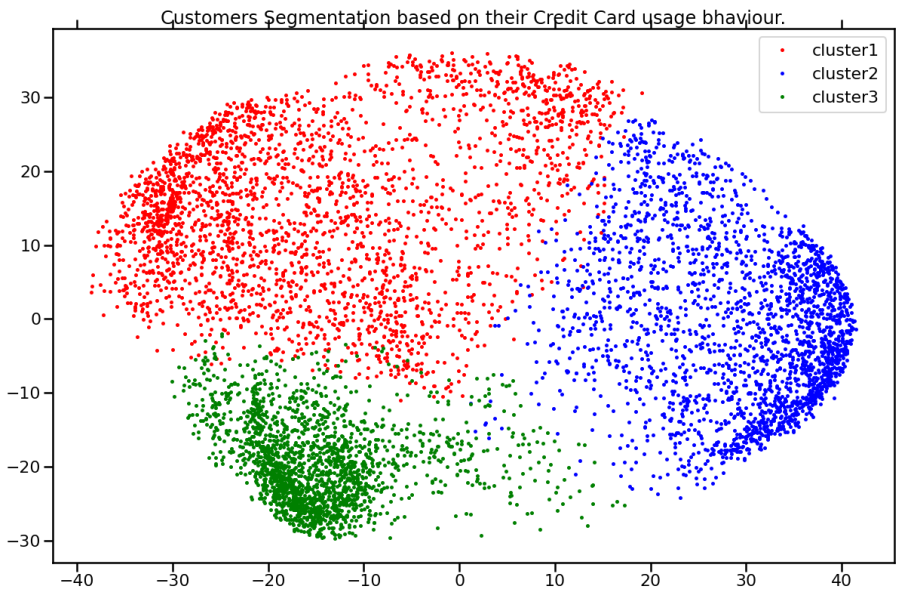

Figure 8. Clusters visualization for PCA principal component vs secondary component

Having a look on how are the **different features distributed for each cluster** we can have some insight how these clusters were grouped (see **figure 9**).

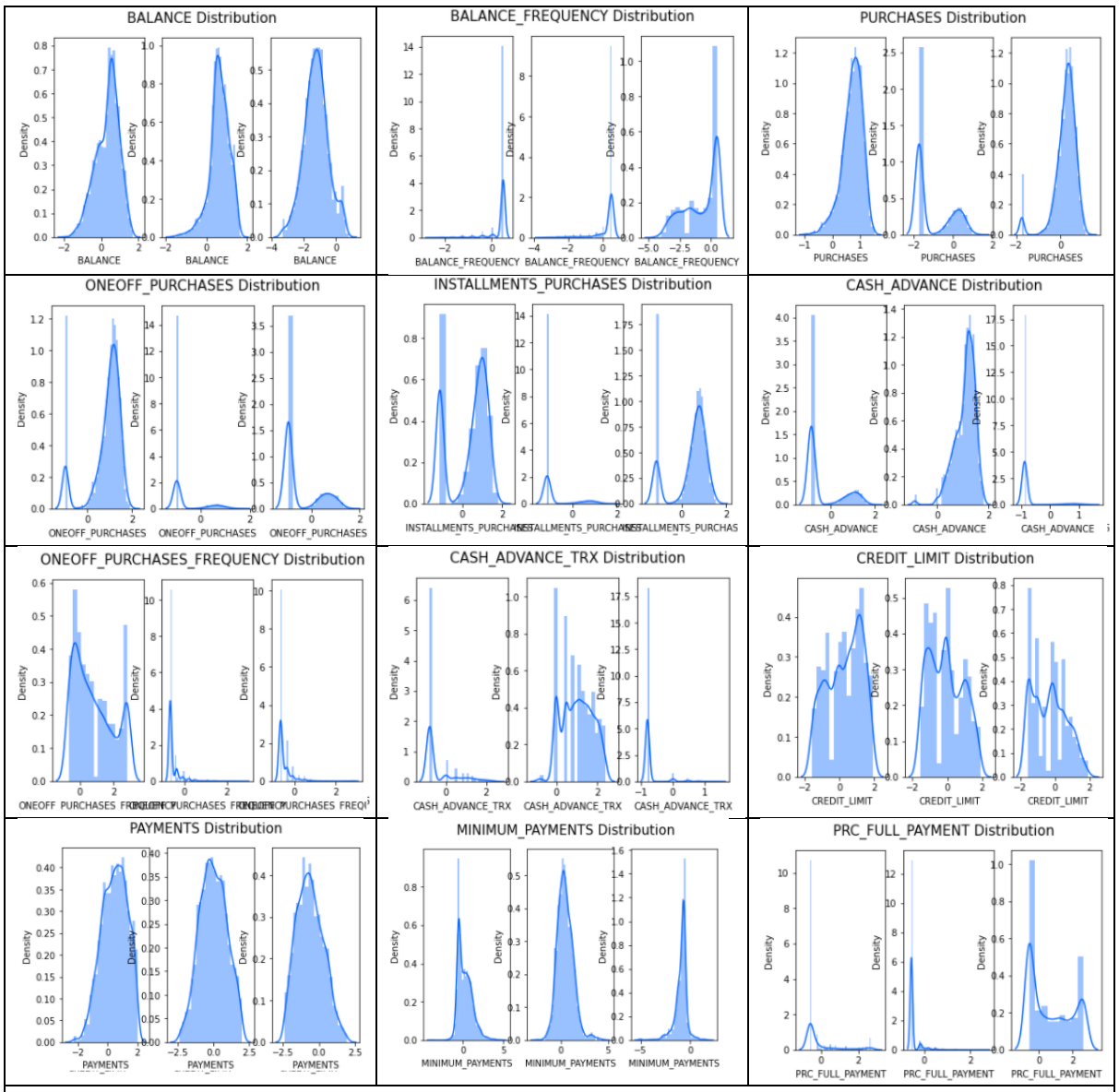

Figure 9. feature distributions by clusters. In each image from left to right, cluster 0, cluster 1 and cluster 2.

From figure 9, we can understand that

1) **Cluster 0**, a positive large balance, they have their balance more frequently updated, they have a larger purchase amount with the largest number of purchases in one go, they pay larger amount, the give a low amount in advance, they also have the largest credit limit and the largest amount paid.
2) **Cluster 1**, they have the highest balance, most of they have their balance frequently updated, a medium purchase amount, and a medium oneoff purchase, they have the largest amount in advance, medium credit limit and medium minimum payments.
3) **Cluster 2**, they have the lowest balance, the balance is the less frequently updated, with the lowest purchases, the lowest oneoff purchases, the largest instalment purchases, they are the customer which less give the cash in advance, the lowest credit limit and lowest minimum payment.

So, we can see that the **clusters are related with the volume of use and amount of money** more less, we could say that **cluster 0 are the customer which uses more the credit card**, **cluster 1 are medium customers** and **cluster 2 consists on customers with a low use** of the credit card.

Next Steps

One of the **flaws** of our model if that **maybe the assumption of similar density for each cluster is wrong**. So, after t**he k value is defined by the DBSCAN model we can use it to train a K-means model** that **does not have the density constrain**. **Also**, we could be to **reduce the dimensionality further by decreasing the threshold that we used previously to drop highly correlated columns**. Thus, we **reduce** the **complexity** and we can understand better the clusters in order to take more specific policies over each customer cluster and try again DBSCAN.