Credit Card Segmentation

June 02, 2021

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.

Goals

1. Advanced data preparation. Build an ‘enriched’ customer profile by deriving ‘intelligent’ KPIs such as monthly average purchase and cash advance amount, purchases by type (one-off, instalments), average amount per purchase and cash advance transaction, limit usage (balance to credit limit ratio), payments to minimum payments ratio etc.

2. Advanced reporting. Use the derived KPI’s to gain insight on the customer profiles.

3. Clustering. Apply a data reduction technique factor analysis for variable reduction technique and a clustering algorithm to reveal the behavioural segments of credit card holders

Data Dictionary

- **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

Methodology

Pre-Processing

When we require to build a predictive model , we require to look and manipulate the data before we start modelling which includes multiple preprocessing steps such as exploring the data, cleaning the data as well as visualizing the data through graph and plots, all these steps are combined under one shed EDA(Exploratory Data Analysis), which includes following steps:

- Data Exploration and Cleaning
- Missing values treatment
- Outlier analysis
- Feature selection and Feature scaling
- Visualization

Modelling

Once all the Pre-Processing Steps has been done on our data set we will move towards modelling.Modelling plays an important role to find out the good interferences from the data .As per our problem statement and dataset we will try some models on our pre-processed data and post comparing the output result . As per our data set following models need to be tested:

- Data Normalization
- Dimension Reduction using PCA
- Clustering
- Using K-mean

Pre-Processing

This step includes importing needed packages and dataset, checking data summary, handling missing values, checking data types, and selecting the features

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUE
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.16
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.00
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.00
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.08
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.08

The data consists of 8950 rows and 18 columns. Here's the summary of the data..

There are many outliers (look at the max value), but I didn't drop them because they may contain important information, so I treated the outliers as extreme values.

Checking missing values :

```
CUST_ID                              0
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
```

CREDIT_LIMIT and MINIMUM_PAYMENT are having some missing values, we handle them by replacing these missing values by means

```
CUST_ID                             0
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                        0
PAYMENTS                            0
MINIMUM_PAYMENTS                    0
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

There are no null or missing values ,now we check the data types

```
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 23 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                      8950 non-null   float64
 14  PAYMENTS                          8950 non-null   float64
 15  MINIMUM_PAYMENTS                  8950 non-null   float64
 16  PRC_FULL_PAYMENT                  8950 non-null   float64
 17  TENURE                            8950 non-null   int64  
 18  Monthly_avg_purchase              8950 non-null   float64
 19  Monthly_cash_advance              8950 non-null   float64
 20  purchase_type                     8950 non-null   object 
 21  limit_usage                       8950 non-null   float64
 22  payment_minpay                    8950 non-null   float64
dtypes: float64(18), int64(3), object(2)
memory usage: 1.6+ MB
```

Data Normalization

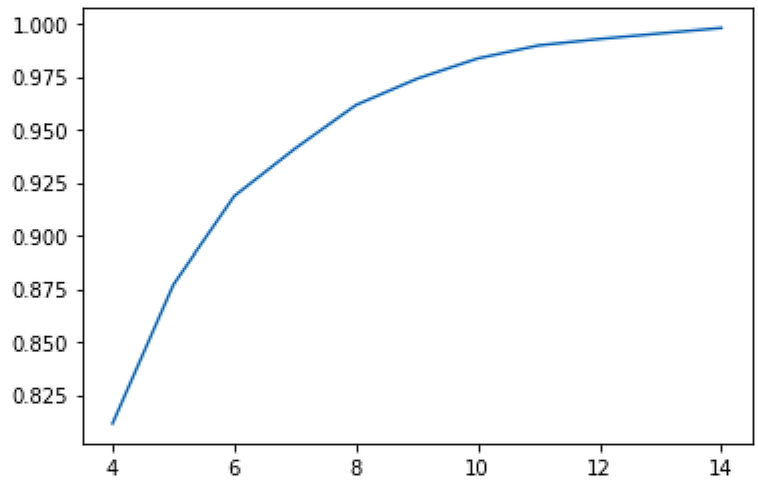
Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges.

Dimension Reduction Using PCA

Here we apply Principal Component Analysis (PCA) to transform data into 2 dimensions for visualization because we won't be able to visualize the data in 17 dimensions. PCA transforms a large set of variables into a smaller one that still contains most of the information in the large set. Reducing the number of variables of data.

```
[ ]  pd.Series(var_ratio).plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f412faf5590>
```




```
[ ]  pc_final=PCA(n_components=5).fit(credit_scaled)
     reduced_credit=pc_final.fit_transform(credit_scaled)
```

```
[ ]  pd.Series(pc_final.explained_variance_ratio_,index=['PC_'+ str(i) for i in range(5)])
```

```
PC_0    0.402058
PC_1    0.180586
PC_2    0.147294
PC_3    0.081606
PC_4    0.065511
dtype: float64
```

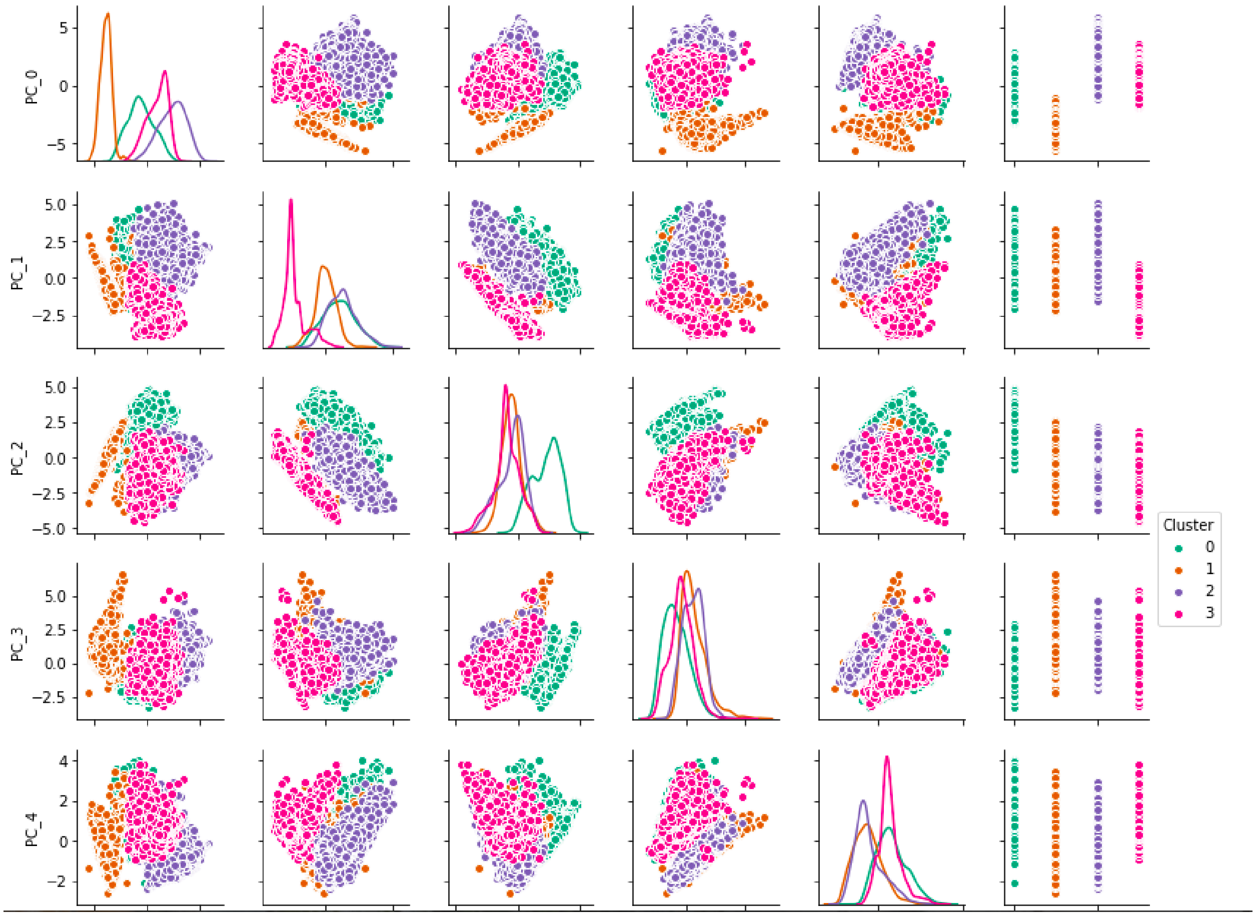
```
[ ]  type(credit_pca)
```

```
sklearn.decomposition._pca.PCA
```

Clustering

Clustering is one of the most common exploratory data analysis techniques used to get an intuition about the structure of the data. It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different.

Here I used the K-means algorithm. K-means algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group.



It shows that first two components are able to identify the clusters

Custers are clearly distinguishing behaviour within customers

The Results are as follows

Cluster_5	0	1	2	3	4
PURCHASES_TRX	34.538035	0.035509	7.067742	27.536476	11.896714
Monthly_avg_purchase	209.814279	0.096572	68.685725	141.648931	47.239695
Monthly_cash_advance	3.996969	185.109488	73.635703	252.400192	19.154845
limit_usage	0.262694	0.576260	0.377563	0.594982	0.246825
CASH_ADVANCE_TRX	0.152645	6.454894	2.648387	10.519641	0.480282
payment_minpay	8.569707	9.950170	5.540102	3.920172	13.866212
both_oneoff_installment	1.000000	0.000000	0.003226	0.878788	0.000000
istallment	0.000000	0.016795	0.000000	0.106622	1.000000
one_off	0.000000	0.003359	0.996774	0.014590	0.000000
none	0.000000	0.979846	0.000000	0.000000	0.000000
CREDIT_LIMIT	5724.213063	4047.344850	4489.884490	5845.791246	3223.856049

- We have a group of customers (cluster 2) having the highest average purchases but there is Cluster 4 also having the highest cash advance & second highest purchase behaviour but their type of purchases are the same.
- Cluster 0 and Cluster 4 are behaving similar in terms of Credit_limit and have cash transactions is on higher side

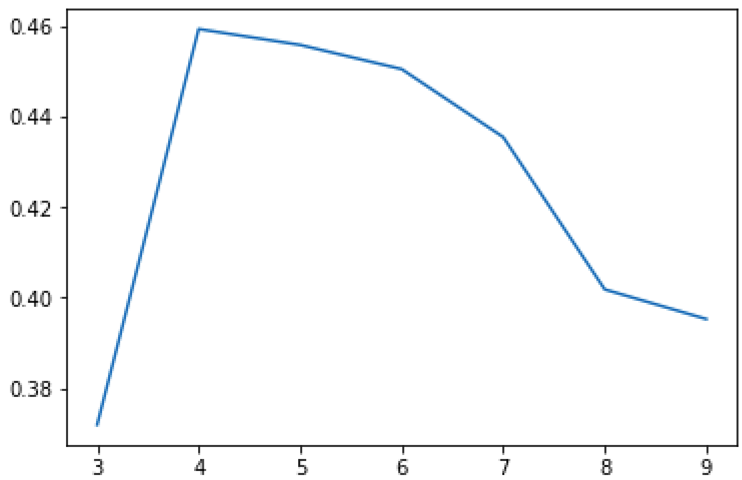
Checking performance metrics for Kmeans

```
from sklearn.metrics import calinski_harabaz_score,silhouette_score
```

```
score={}
score_c={}
for n in range(3,10):
    km_score=KMeans(n_clusters=n)
    km_score.fit(reduced_cr)
    score_c[n]=calinski_harabaz_score(reduced_cr,km_score.labels_)
    score[n]=silhouette_score(reduced_cr,km_score.labels_)
```

```
pd.Series(score).plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x16dbd5f8>
```



Performance metrics also suggest that K-means with 4 cluster is able to show distinguished characteristics of each cluster.

Marketing Strategy Suggested:

a. Group 2

- They are potential target customers who are paying dues and doing purchases and maintaining comparatively good credit score) -- we can increase credit limit or can lower down interest rate -- Can be given premium card /loyality cards to increase transactions

b. Group 1

- They have poor credit scores and take only cash in advance. We can target them by providing less interest rate on purchase transaction

c. Group 0

- This group has a minimum paying ratio and uses cards for just one off transactions (may be for utility bills only). This group seems to be a risky group.

d. Group 3

- This group is performing best among all as customers are maintaining a good credit score and paying dues on time. -- Giving rewards point will make them perform more purchases.