Table of Contents

1. Data Overview

1.1 Purpose of project

Dataset is a data file of transaction information of credit card accounts, consisting of 8,950 lines and 18 columns. This data stores information about the relevant transaction history of customers such as: account balance, purchase amount in the month, advance amount in the month, installment or non-installment purchase. This essay uses k-Means to categorise into four groups of customers based on consumer behavior, thereby making recommendations based on each of these customer groups.

1.2 Data Understanding

The meaning of variables in data.

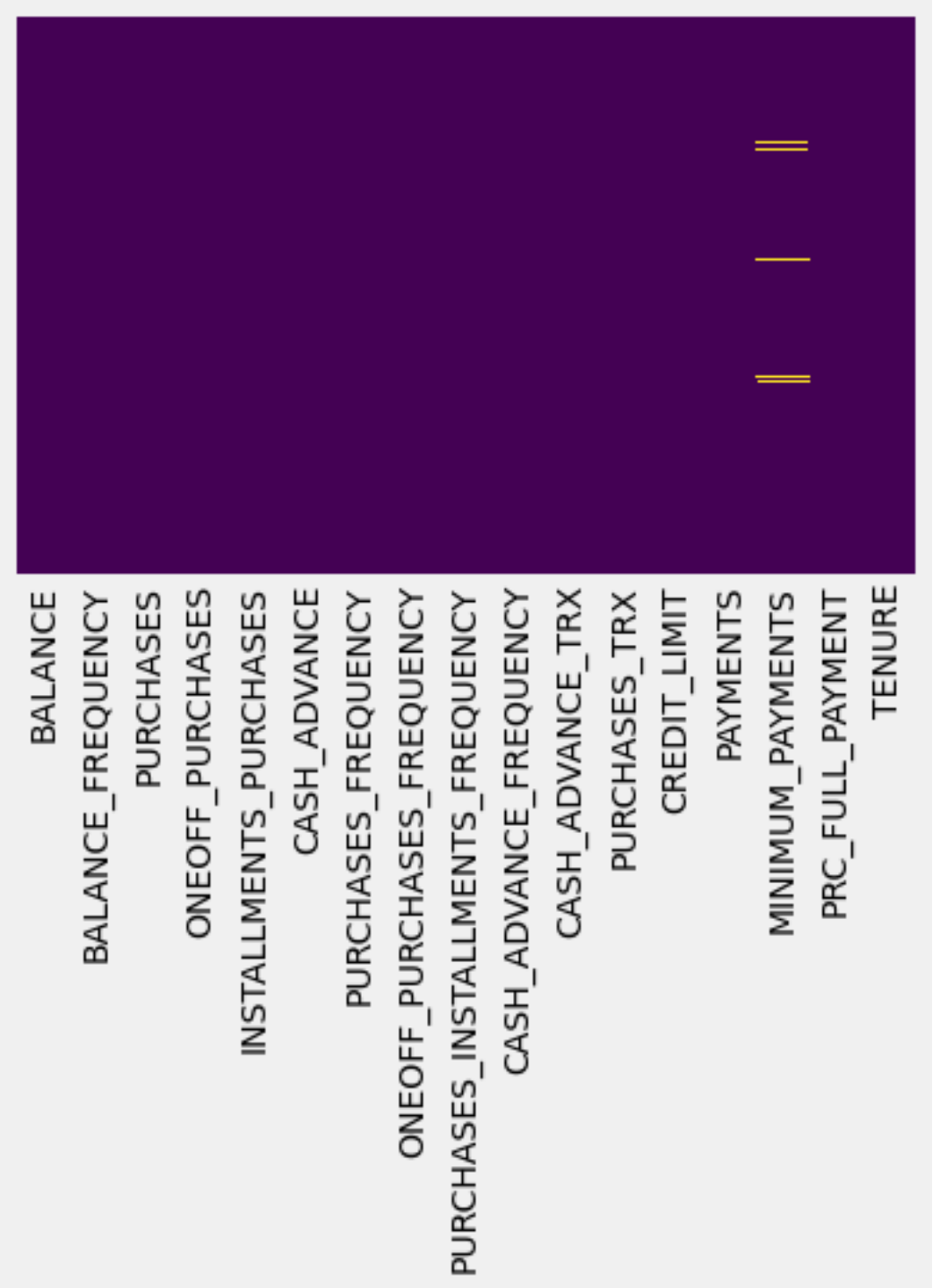
- CUST_ID: number ID of customer.
- BALANCE: account balance of credit card.
- BALANCE_FREQUENCY: account balance ratio.
- PURCHASES: the amount of money the customer has purchased.
- ONEOFF_PURCHASES: one-time payment transactions.
- INSTALLMENT_PURCHASES: installment payment transaction.
- CASH_ADVANCE: cash withdrawals/applications.
- PURCHASES_FREQUENCY: purchase rate.
- ONEOFF_PURCHASES_FREQUENCY: one-time payment rate.
- PURCHASES_INSTALLMENTS_FREQUENCY: installment payment rate.
- CASH_ADVANCE_FREQUENCY: cash withdrawal/application rate.
- CASH_ADVANCE_TRX: number of cash withdrawals/applications.
- PURCHASES_TRX: number of purchases made.
- CREDIT_LIMIT: limit of credit card.
- PAYMENTS: total amount payed.
- MINIMUM_PAYMENTS: minimum payment amount.
- PRC_FULL_PAYMENT: full payout ratio to total outstanding balance. Customers do not leave outstanding debts.

- TENURE: term (tenor).

2. Methodology

2.1 Data Pre-processing

Because data is a data set consisting of 8,950 lines and 18 columns, including a total of 314 null values in 2 fields. CREDIT_LIMIT has 1 null value and MINIMUM_PAYMENTS has 313 null values.



```
CREDIT_LIMIT
False    8949
True        1
Name: CREDIT_LIMIT, dtype: int64

MINIMUM_PAYMENTS
False    8637
True      313
Name: MINIMUM_PAYMENTS, dtype: int64
```

2.2 Data Cleaning and Transformation

- Handle null values: since all lines have both account balances and transactions, null values in CREDIT_LIMIT and MINIMUM_PAYMENTS cannot be replaced by 0. In order not to create peripheral values that are desirable and suitable for data, the method of replacing null values in this article is median.

- Handle the value gap in variables: because the values in this data are different and have very large gaps. If these values are taken for analysis, they will generate peripheral values and misunderstand variables. Therefore, this article brings the values of logarithmic values to avoid the disadvantages just stated.

2.3 Creating New Variables for Analysis

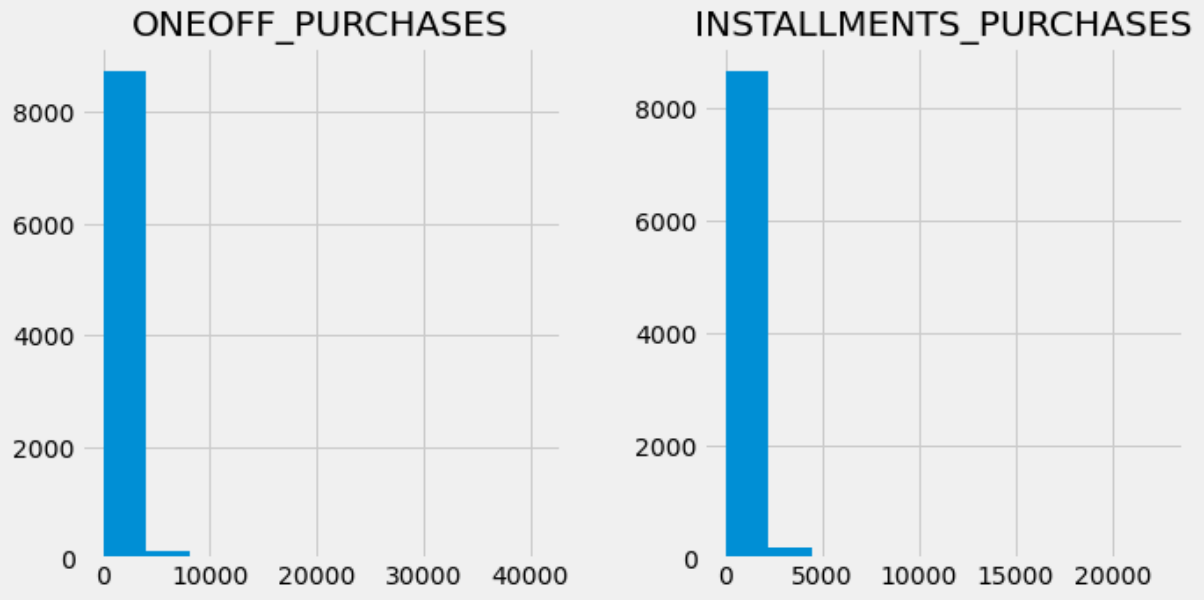
From this data, we can create more fields to better understand the behavior of customers using tags. Specifically, as follows:

- Monthly purchase value based on term. This value is calculated by: Monthly_purchase = PURCHASES / TENURE.

- The amount of money customers makes cash withdrawals/advances. This value is calculated by: Monthly_cash_advance = CASH_ADVANCE / TENURE.

- The remaining limit rate is used. This value is calculated by: limit_usage = BALANCE / CREDIT_LIMIT.

- Payment rate compared to minimum payment. This value is calculated by: pay_minpay = PAYMENTS / MINIMUM_PAYMENTS.



In this data, attention should be paid to ONEOFF_PURCHASES and INSTALLMENT_PURCHASES fields, which show the transactions purchased by customers are paid once and transactions purchased by customers through installments. From here we can classify customers into 4 groups as follows:

- The group of customers who make non-purchase payments through installments or one-time payments. That is, customers can only make cash withdrawals/ cash advances.
- The group of customers has just made an installment purchase and paid once.
- The customer group only make installment purchases.
- Customer group only make one-time payment purchase.

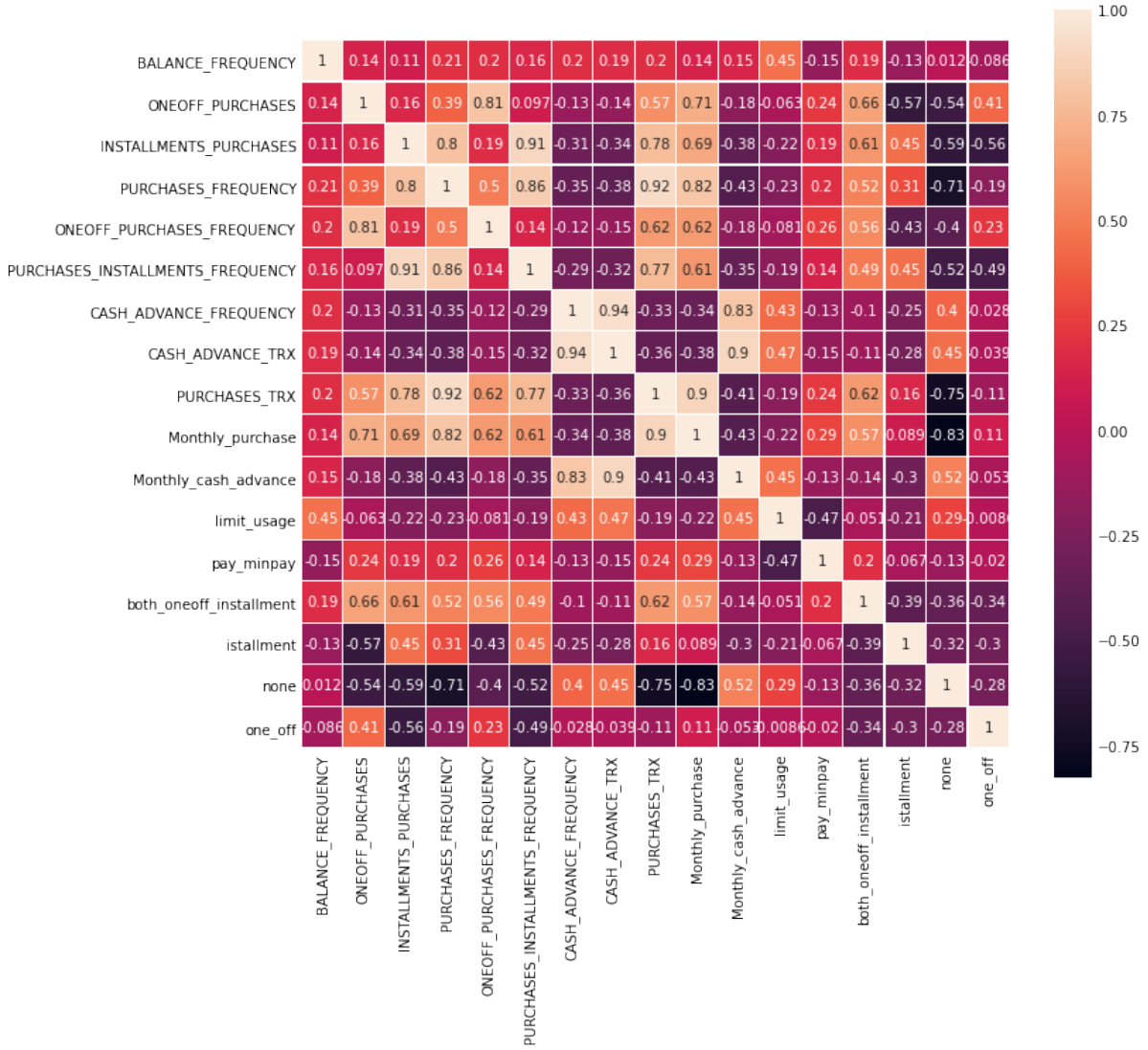
2.4 Normalize Data

From the above 4 groups of customers, for the convenience of analysis, it is necessary to turn these 4 groups of customers into 4 different fields, which are discrete and have a value of 0 or 1.

Monthly_purchase	Monthly_cash_advance	limit_usage	pay_minpay	purchase_kind	both_oneoff_installment	istallment	none	one_off
2.191654	0.000000	0.040086	0.894662	istallment	0	1	0	0
0.000000	6.287695	0.376719	1.574068	none	0	0	1	0
4.180994	0.000000	0.287197	0.688979	one_off	0	0	0	1
4.835620	2.898616	0.200671	0.000000	one_off	0	0	0	1
0.847298	0.000000	0.519644	1.327360	one_off	0	0	0	1

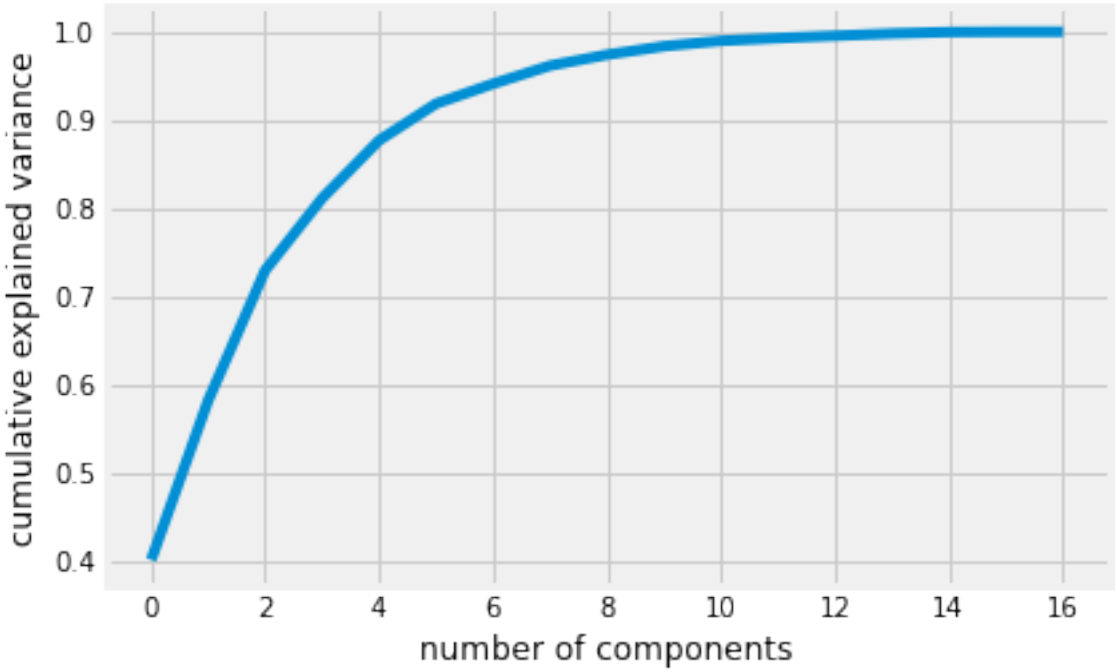
3. Analytical Method

3.1 Correlation Matrix

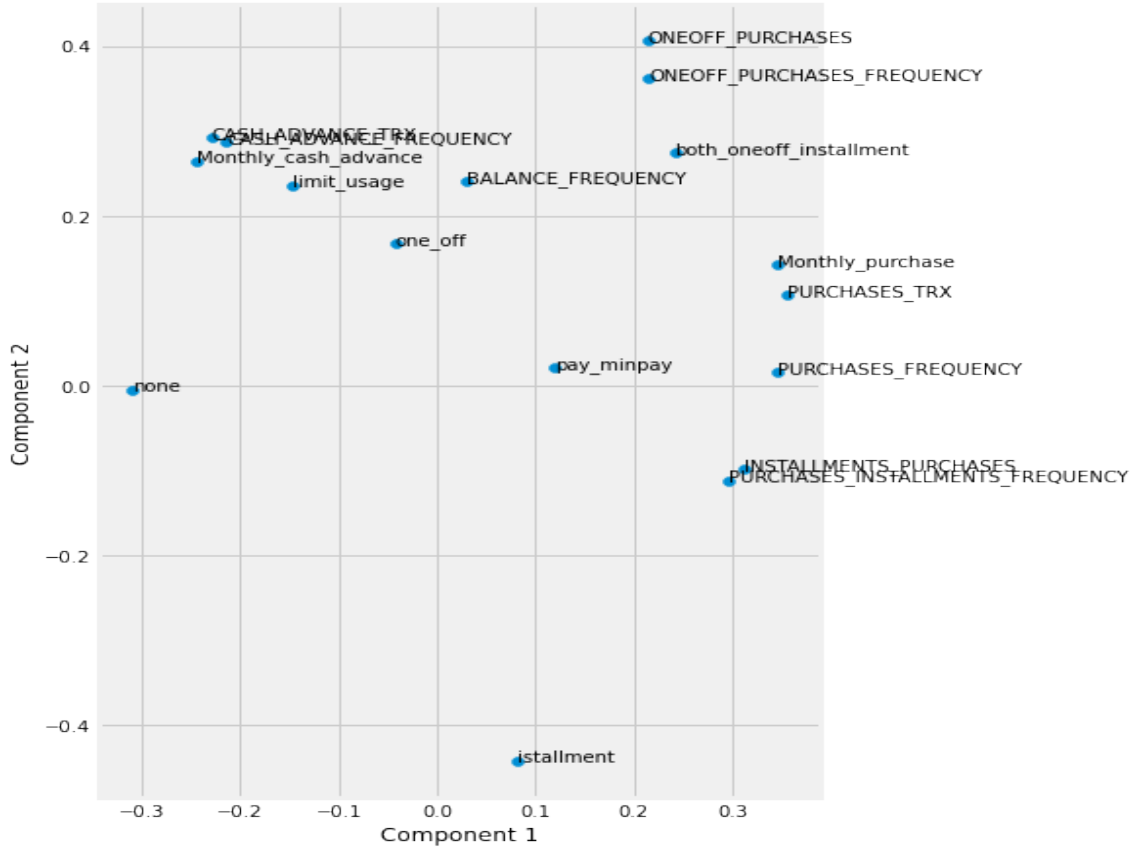


Since all variables in data are continuous values, we can only check the correlation between variables. Variables with an absolute value greater than 0.7 are strongly related. The chart above shows that there are many variables that are strongly related to each other, so in this article I use the PCA analysis method to simplify the data set without losing the original information from the data.

3.2 Finding number of components in PCA

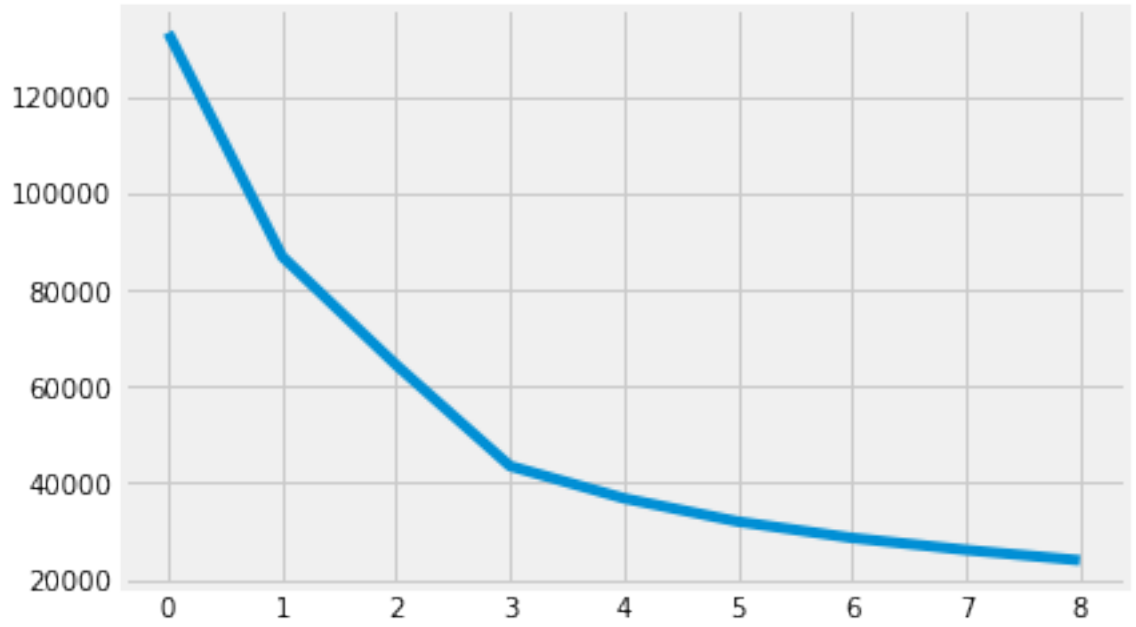


From the chart above, we can see in the component value of 5 explaining 92% of the variance. We choose the value 5 because the slope after the value 5 does not change much. Fields in the data can be seen in the following chart:

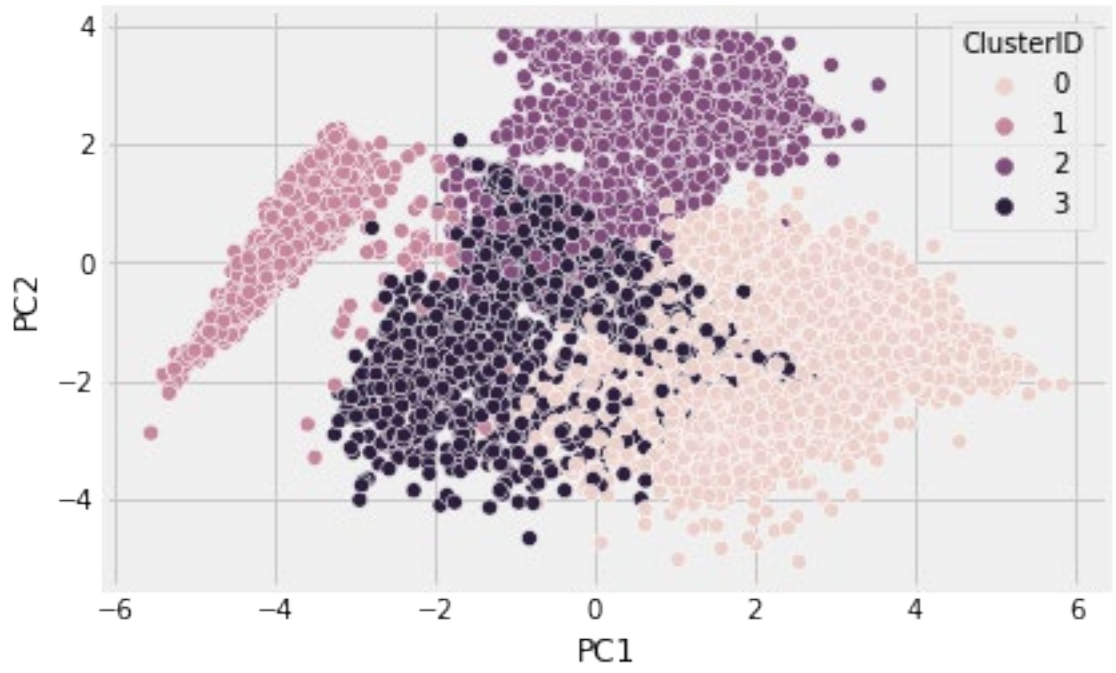


3.3 k-Means Cluster Model

First, use the StandardScaler method in the sklearn library to standardize data.



In this chart, we can see that the value k=4 (in the order of 0.1,2,3) is the peak of the slope down. After that k=4 value, the slope is less even and also corresponds to the number of groups of customers to classify in this article. The following scatter plot shows the clustering of customers into 4 groups.



4. Conclusion and Recommendation

ClusterID	0	1	2	3
BALANCE	1811.24	2177.57	798.27	1429.02
PURCHASES	2280.61	1.86	543.58	787.35
PURCHASES_FREQUENCY	0.80	0.00	0.70	0.32
PURCHASES_TRX	33.14	0.05	12.05	7.12
CASH_ADVANCE_TRX	2.81	6.55	1.02	2.86
CREDIT_LIMIT	5748.84	4055.58	3338.24	4512.91
Monthly_purchase	193.76	0.16	47.56	69.76
Monthly_cash_advance	67.64	186.30	33.47	77.84
limit_usage	0.35	0.58	0.26	0.38
pay_minpay	7.26	9.93	13.40	5.56
both_oneoff_installment	1.00	0.00	0.00	0.00
istallment	0.00	0.02	1.00	0.00
none	0.00	0.98	0.00	0.00
one_off	0.00	0.00	0.00	1.00

With customer clustering group 0, this is the group of customers who simultaneously make installment purchases and one-time purchases, and also the group with the highest frequency of product purchases among 4 customers. That proves the total purchase, the monthly purchasing power of this group is also the highest. However, this group of customers only uses 35% of the limit. That shows that customers use cash more than advance, in fact in the post shows that the monthly purchase amount of the customer is 193.76 while the customer's advance/withdrawal transaction is only 67.64. Therefore, from a banking perspective, it is necessary to encourage this group of customers to use the service of credit cards more because the banking benefits can bring more incentives, through which the bank can charge fees from affiliates more while this group of customers still has benefits through accumulating gift points.

With customer clustering group 1, this is the group with the least frequency of shopping but chooses the most advance option and has the highest usage limit in the group of 4 customers. This means that this group of customers does not need to shop but when shopping prefers to use the advance method. Therefore, this solution to stimulate this group of customers is to understand their shopping needs to be able to provide the right products and services to increase the frequency of shopping more.

With customer clustering group 2, this is a group that only does installment shopping and is also the group with the best payment rate in 4 groups. This is therefore a group of potential customers who need to increase their credit limit or provide better card services.

Customer clustering group 3 is the group with the best trading activity, the lowest risk of the 4 groups of customers.

APPENDIX

- Library settings and graphing styles.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.decomposition import IncrementalPCA
from sklearn.metrics import silhouette_score
plt.style.use('fivethirtyeight')
```

- Reading data.

```
data = pd.read_csv('Credit Card.csv')
df= data.drop('CUST_ID', axis=1)
print(data.shape)
data.head(10)
```

- Checking null value.

```
missing_data = df.isnull()
sns.heatmap(missing_data, cbar=False, yticklabels=False, cmap='viridis')
```

- Counting number of value.

```
missing_data = df.isnull()
for column in missing_data.columns.values.tolist():
    print(column)
    print (missing_data[column].value_counts())
    print("")
```

- Replace null values by median method.

```
df['CREDIT_LIMIT'].fillna(df['CREDIT_LIMIT'].median(),inplace=True)
df['MINIMUM_PAYMENTS'].fillna(df['MINIMUM_PAYMENTS'].median(),inplace=T
rue)
df.isnull().sum()
```

- Creating new variables for analysis. (Mục 2.3)

```
df['Monthly_purchase']=df['PURCHASES']/df['TENURE']
df['Monthly_cash_advance']=df['CASH_ADVANCE']/df['TENURE']
```

```
df['limit_usage']=df.apply(lambda x: x['BALANCE']/x['CREDIT_LIMIT'], axis=1)
df['pay_minpay']=df.apply(lambda
x:x['PAYMENTS']/x['MINIMUM_PAYMENTS'],axis=1)
```

- Understanding data for group customers.

```
df[['ONEOFF_PURCHASES','INSTALLMENTS_PURCHASES']].hist(bins = 10,
figsize=(10,5))
plt.show()
df.loc[:,['ONEOFF_PURCHASES','INSTALLMENTS_PURCHASES']].tail(10)
```

- Customers grouping.

```
def purchase(df):
  if (df['ONEOFF_PURCHASES']==0) & (df['INSTALLMENTS_PURCHASES']==0):
    return 'none'
  if (df['ONEOFF_PURCHASES']>0) & (df['INSTALLMENTS_PURCHASES']>0):
     return 'both_oneoff_installment'
  if (df['ONEOFF_PURCHASES']>0) & (df['INSTALLMENTS_PURCHASES']==0):
    return 'one_off'
  if (df['ONEOFF_PURCHASES']==0) & (df['INSTALLMENTS_PURCHASES']>0):
    return 'istallment'
df['purchase_kind']=df.apply(purchase,axis=1)
df.head()
```

- Convert value to logarithmic value.

```
df_log=df.drop(['purchase_kind'],axis=1).applymap(lambda x: np.log(x+1))
df_log.describe()
col=['BALANCE','PURCHASES','CASH_ADVANCE','TENURE','PAYMENTS','MINI
MUM_PAYMENTS','PRC_FULL_PAYMENT','CREDIT_LIMIT']
df_log=df_log[[x for x in df_log.columns if x not in col ]]
df_log.head()
```

- Convert values in the purchase_kind column to discrete values.

```
df_log['purchase_kind']=df.loc[:,'purchase_kind']
df_dummy=pd.concat([df_log,pd.get_dummies(df_log['purchase_kind'])],axis=1)
df_dummy.head()
```

- Draw correlation matrix charts.

```
%matplotlib inline
fig, ax = plt.subplots(figsize=(10,10))
```

```
ax = sns.heatmap(df_dummy.corr(), linewidths=0.1, square=True, annot=True)
```

- Scale data according to Standard Scaler method.

```
for i in num_cols:
    # fit on training data column
    scale = Scaler().fit(X[[i]])
    X[i] = scale.transform(X[[i]])
  X.head()
```

- Using PCA model.

```
pca = PCA(svd_solver='randomized', random_state=42)
pca.fit(X)
pca.components_
```

- Find the number of components by Line chart.

```
%matplotlib inline
fig = plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.show()
```

- Save components values to dataframes.

```
pca_df = pd.DataFrame({ 'Variable':num_cols,'PC1':pca.components_[0],
                        'PC2':pca.components_[1],'PC3':pca.components_[2],
                        'PC4':pca.components_[3],'PC5':pca.components_[4]})
pca_df
```

- Show component values for each variable.

```
%matplotlib inline
fig = plt.figure(figsize = (8,8))
plt.scatter(pca_df.PC1, pca_df.PC2)
plt.xlabel('Component 1')
plt.ylabel('Component 2')
for i, txt in enumerate(pca_df.Variable):
    plt.annotate(txt, (pca_df.PC1[i],pca_df.PC2[i]))
plt.tight_layout()
plt.show()
```

- Using PCA with component number 5.

```
pca_model = IncrementalPCA(n_components=5)
X_pca = pca_model.fit_transform(X)
```

- Pivot table.

```
X_pca = np.transpose(X_pca)
pca_df1 = pd.DataFrame({'PC1':X_pca[0],'PC2':X_pca[1],'PC3':X_pca[2],'PC4':X_pca[3],'PC5':X_pca[4]})
print(pca_df1.shape)
pca_df1.head()
```

- Find clustered numbers in the k-Means model by line chart.

```
ssd = []
for num_clusters in list(range(1,10)):
    model_clus = KMeans(n_clusters = num_clusters, max_iter=50)
    model_clus.fit(pca_df1)
    ssd.append(model_clus.inertia_)


plt.plot(ssd)
```

- Using a clustered number of 4 in k-Means.

```
df_cluster = KMeans(n_clusters = 4, max_iter=50,random_state = 50)
df_cluster.fit(pca_df1)
df1 = pca_df1
df1.index = pd.RangeIndex(len(df1.index))
df1_k_means = pd.concat([df1, pd.Series(df_cluster.labels_)], axis=1)
df1_k_means.columns = ['PC1', 'PC2','PC3','PC4','PC5','ClusterID']
df1_k_means
```

- Clustered performance using Scatter plot

```
sns.scatterplot(x='PC1',y='PC2',hue='ClusterID',legend='full',data=df1_k_means)
```

- Merge values in data 2 PCA models and cluster. Add customer ID column values and group classification values.

```
data_full_variable = pd.concat([df,pd.get_dummies(df['purchase_kind'])],axis=1)
data_full_variable['CUST_ID'] = data['CUST_ID']
```

```
big_data=pd.merge(data_full_variable, df2, on='CUST_ID', how='inner')
big_data=big_data.drop(['PC1','PC2','PC3','PC4','PC5'],axis=1)


cols = ['BALANCE', 'PURCHASES', 'PURCHASES_FREQUENCY', 'PURCHASES_TRX',
      'CASH_ADVANCE_TRX', 'CREDIT_LIMIT', 'Monthly_purchase', 'Monthly_cash_advance',
      'limit_usage', 'pay_minpay', 'both_oneoff_installment', 'istallment', 'none', 'one_off',]
df_final=[]
for i in cols:
   cluster = big_data.groupby(["ClusterID"])[i].mean()
   df_final.append(cluster)
df_final = pd.concat(df_final, axis=1)
df_final
```

- Save customer clustering dataframes.

```
df_final.to_csv('kmeans.csv',index=True)
```