Credit Card Segmentation

Abstraction:

In this project we need to find different types of customers based on their card usage, with the help of attributes like customer id, monthly cash advance, balance frequency, online/offline purchases,

Instalment purchases, etc. We need to suggest marketing strategy for the bank/organisation.

Here we used R studio and Jupyter Notebook as platforms to work on the project. At first, we need to clean data using Exploratory Data Analysis like check for Missing Values, Outliners, then Feature Extraction, Feature Selection, Feature Scaling. Achieving the goal was quite challenging. Using Machine learning concepts like Unsupervised learning, we do some clustering using K-means cluster, in order to categorize the customers based on their usage of card, which helps us to suggest a marketing strategy for the bank/organisation/firm.

Project Index

1. INTRODUCTION:

Credit card segmentation is a process of filtering the customers based on their purchases and try to develop some strategies in order to run the bank/organisation.

In this project, our objective is to categorize the customers based on their interest. The data contains 8950 observations and 19 variables. i.e Customer id, Balance, Balance Frequency, Purchases, Online/offline purchases, Instalment Purchases, Cash advance, purchase frequency, online/offline purchase frequency, Purchase instalments frequency, cash advance frequency, cash advance transaction, purchase transaction, credit limit, payments, minimum payments, PRC full payments, Tenure.

Our aim is to categorize the customers based on their interest of their credit card transactions and purchases, we need to suggest the marketing strategy to the bank/organisation.

2. 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.

Number of attributes:

- Customer id
- Balance
- Balance Frequency
- Purchases
- Online/offline purchases
- Instalment Purchases
- Cash advance
- purchase frequency
- online/offline purchase frequency
- Purchase instalments frequency
- cash advance frequency
- cash advance transaction
- purchase transaction
- credit limit
- minimum payments
- PRC full payments
- Tenure

3. Loading Data into Python:

After installing important packages.

Here we are using python Jupyter notebook and we are loading data into python environment. Using following command:

Orginal_data = pd.read_csv("C:/Users/Hp/Desktop/BLESSED 2/credit-card-data.csv")

```
In [6]: # loading the required libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import scipy.stats as stats
        from fancyimpute import KNN
        from sklearn import metrics

In [2]: # reading data into dataframe
        Orginal_data = pd.read_csv("C:/Users/Hp/Desktop/BLESSED 2/credit-card-data.csv")

In [3]: h = Orginal_data.copy()

In [4]: h.info()
        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 8950 entries, 0 to 8949
        Data columns (total 18 columns):
        CUST_ID                             8950 non-null object
        BALANCE                             8950 non-null float64
        BALANCE_FREQUENCY                   8950 non-null float64
        PURCHASES                           8950 non-null float64
        ONEOFF_PURCHASES                    8950 non-null float64
        INSTALLMENTS_PURCHASES              8950 non-null float64
        CASH_ADVANCE                        8950 non-null float64
        PURCHASES_FREQUENCY                 8950 non-null float64
        ONEOFF_PURCHASES_FREQUENCY          8950 non-null float64
        PURCHASES_INSTALLMENTS_FREQUENCY    8950 non-null float64
        CASH_ADVANCE_FREQUENCY              8950 non-null float64
        CASH_ADVANCE_TRX                    8950 non-null int64
        PURCHASES_TRX                       8950 non-null int64
        CREDIT_LIMIT                        8949 non-null float64
        PAYMENTS                            8950 non-null float64
        MINIMUM_PAYMENTS                    8637 non-null float64
        PRC_FULL_PAYMENT                    8950 non-null float64
        TENURE                              8950 non-null int64
        dtypes: float64(14), int64(3), object(1)
        memory usage: 1.2+ MB
```

4.Missing value Analysis:

Here we are check for missing values in the dataset like empty rows which was filled with Na. we found some missing values in our dataset. Now missing values can be found by using different techniques like mean, median and Knn imputation.

First, we are selecting 1000 rows randomly and performing mean, median, Knn imputation, so that we can choose any one by comparing actual value and predicted value.

There are 3 types of methods to predict the missing values.

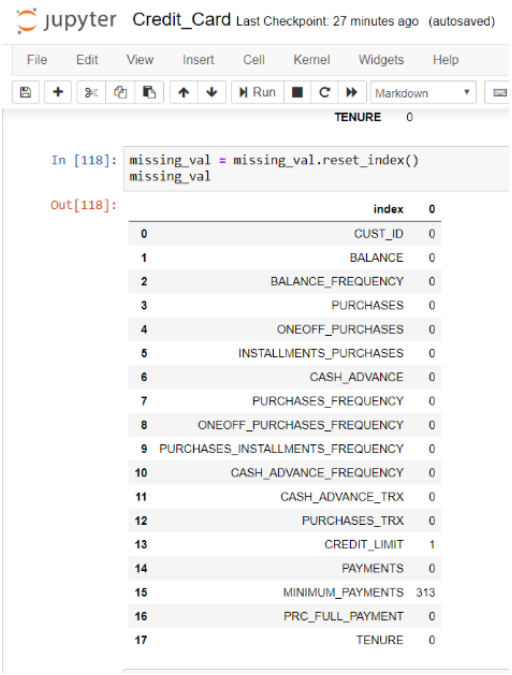
1. MEAN method

2.MEDIAN method

3.KNN

We need to decide which method is suitable for our dataset, fo r that we are randomly taking actual value of observation of cr edit limit, minimum payment variables and check the predicte d values of above 3 methods with actual value and decide whic h method is suitable.

Here we can use knn for finding missing values but knn works only on numerical variables but we have a variable with datat ype value factor, we can convert them into numeric but as we are doing clustering, we are not supposed to do that.so, we are going for median.

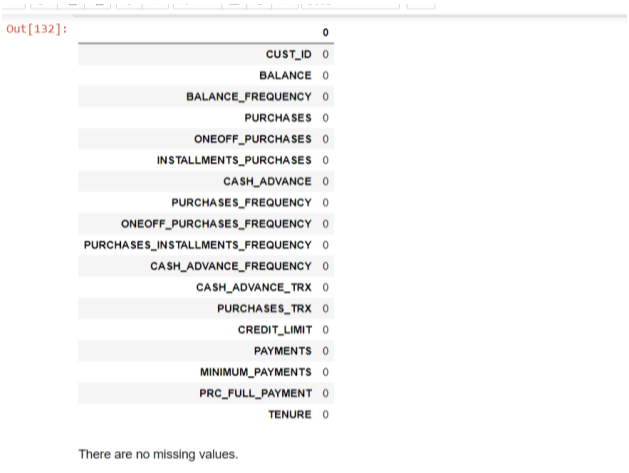


Here the variables credit limit and minimum payments are ha ving some missing values, so using, median method. We are im puting the missing values. Using following command:

h['CREDIT_LIMIT'].fillna(h['CREDIT_LIMIT'].median(),inplace =True)

h['MINIMUM_PAYMENTS'].fillna(h['MINIMUM_PAYMENTS']. median(),inplace=True)

After imputation, checking for missing values. We found that t here are no missing values after imputing with median.



5.Feature Extraction (Deriving New KPI):

I. Monthly average purchase

We are creating Monthly average purchase from Purchases and Tenure, using following command:

```
h['Monthly_avg_purchase']=h['PURCHASES']/h['TENURE']
```

II. Monthly cash advance

We are creating Monthly cash advance from Purchases and Tenure, using following command:

```
h['Monthly_cash_advance']=h['CASH_ADVANCE']/h['TENURE']
```

III. Purchase Type

Here we are trying to extract different types of customers based on their purchases. We have two variables
1.ONEOFF_PURCHASES 2. INSTALMENTS PURCHASES
using these variables we extract customer behaviour, using following command:

```
h[(h['ONEOFF_PURCHASES']==0)&(h['INSTALLMENTS_PURCHASES']==0)].shape

h[(h['ONEOFF_PURCHASES']==0)&(h['INSTALLMENTS_PURCHASES']>0)].shape

h[(h['ONEOFF_PURCHASES']>0)&(h['INSTALLMENTS_PURCHASES']==0)].shape

h[(h['ONEOFF_PURCHASES']>0)&(h['INSTALLMENTS_PURCHASES']>0)].shape
```

We can observe, they are 4 types of customer behaviours.

1). Customers who do only online/offline Purchases.
2). Customers who purchases only through Installments.
3). Customers who purchases both through online/offline and installments.
4). Customers who don't do any type of purchases.

Now we are creating new variable called Purchase Type which will be of categorical values. Using following command line code.

```
def purchase(h):
   if (h['ONEOFF_PURCHASES']==0) &
(h['INSTALLMENTS_PURCHASES']==0):
       return 'none'

 if (h['ONEOFF_PURCHASES']>0) &
(h['INSTALLMENTS_PURCHASES']>0):
        return 'both_online/offline_installment'

   if (h['ONEOFF_PURCHASES']>0) &
(h['INSTALLMENTS_PURCHASES']==0):
       return 'online/offline_Purchases'

   if (h['ONEOFF_PURCHASES']==0) &
(h['INSTALLMENTS_PURCHASES']>0):
       return 'installment'
```

IV. Limit Usage:

Here we are creating new variable called limit usage from two variables Balance and credit limit. Using following command.

```
h['limit_usage']=h.apply(lambdax:x['BALANCE']/x['CREDIT_LIMIT'], axis=1)
```

V. Minimum payments ratio:

Here we are creating new variable called Minimum payments ratio from payments and minimum payments ratio. Using following command.
h['payment_minpay']=h.apply(lambdax:x['PAYMENTS']/x['MINIMUM_PAYMENTS'],axis=1)

Note: There is no need of extraction of Average amount per purchase and cash advance transaction, because they are already present in the dataset.

6.Outlier Analysis

Here in order to save time and lot of fuss, we used log transformation technique in order to remove the outlier effect from our dataset.

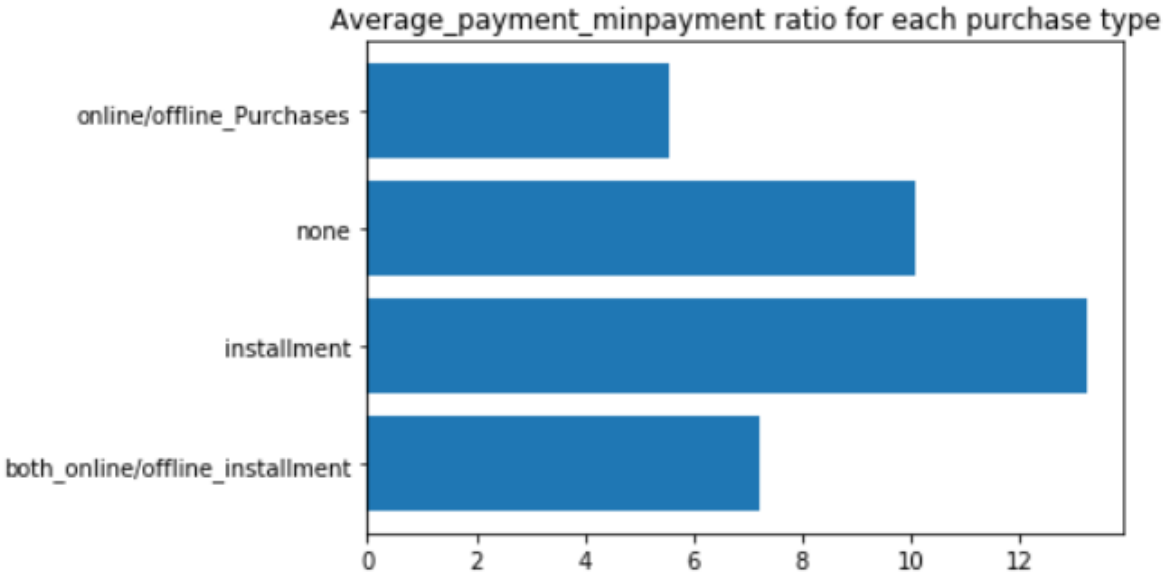
Log transformation: It is used to transform skew data distribution to normal data distribution. Coming to our case, here it removed outliers as log transformation does not allow outliers.

Here we eliminate the outliers using following command:

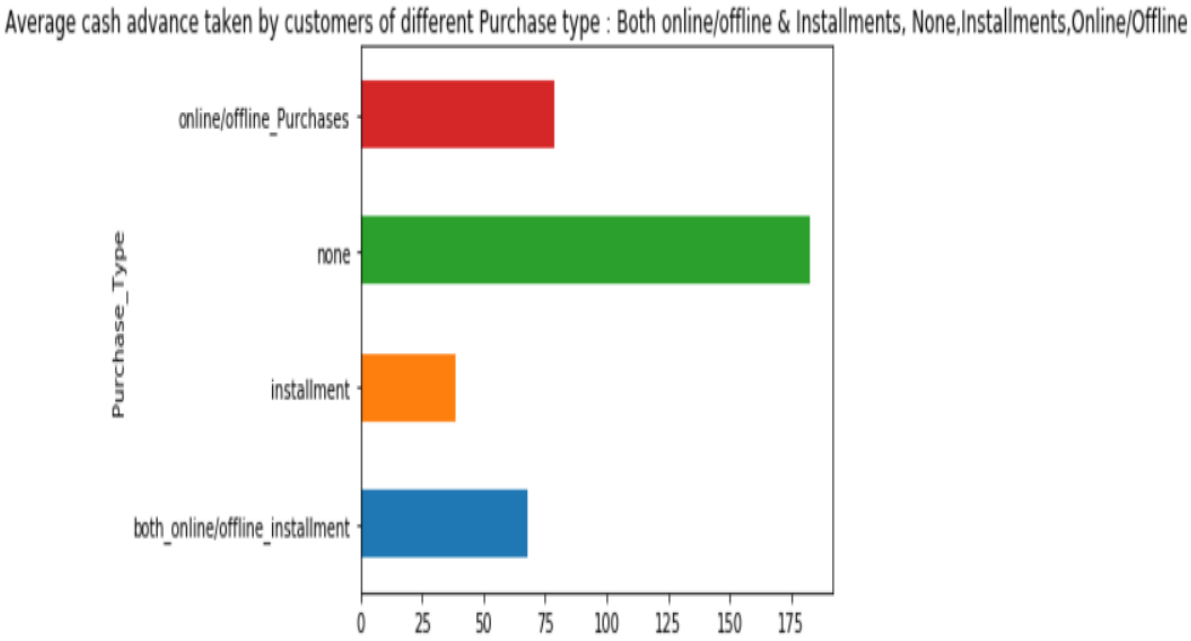
data=h.drop(['CUST_ID','Purchase_Type'],axis=1).applymap(lambda x: np.log(x+1))

Information drawn from KPI:

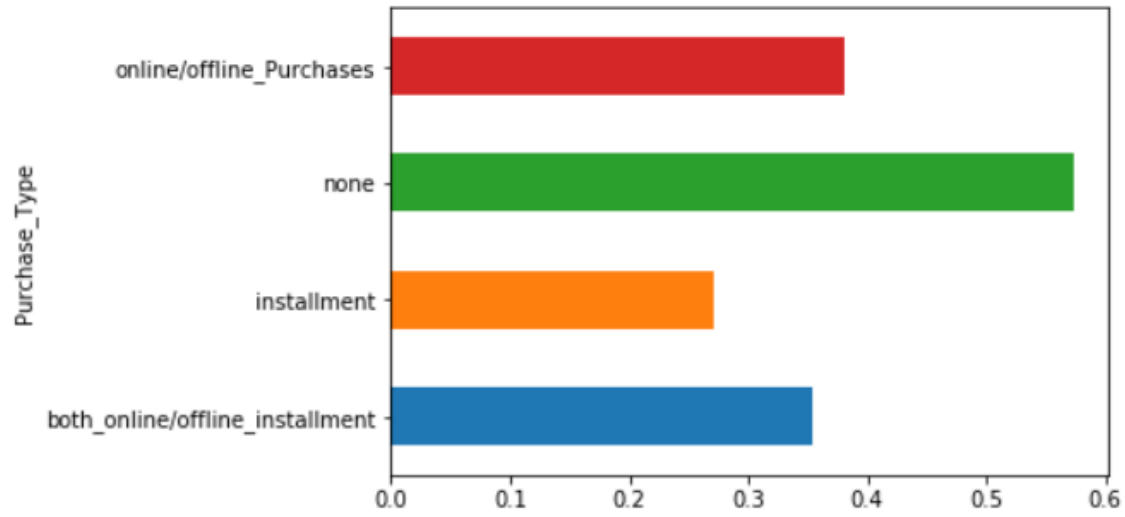
1. Customers who purchases through instalments uses credit card to pay their dues.



2. Customers who does not do any kind of transactions, takes more cash in advance.



3. Customers with instalment purchases has a good credit score.



The type of customer with low limit-usage value has good credit score

Model Tunning

Here we are creating dummies from the purchase type variable, using the following command:

```
data_pre['Purchase_Type']=h.loc[:,'Purchase_Type']

data_original=pd.concat([h,pd.get_dummies(h['Purchase_Type'])],axis=1)

data_dummy=pd.concat([data_pre,pd.get_dummies(data_pre['Purchase_Type'])],axis=1)
```

```
In [172]: data_dummy.info()
```

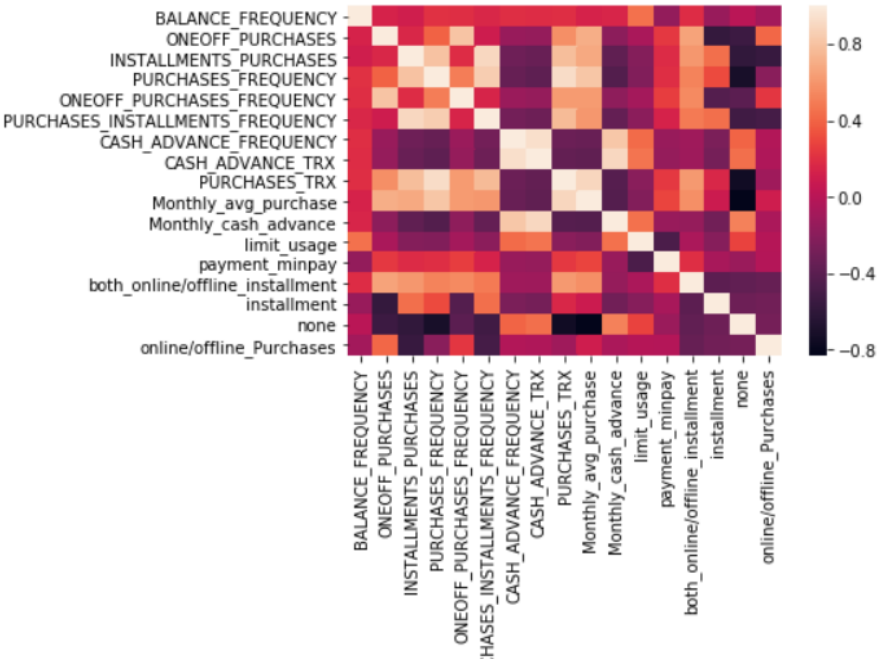
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
BALANCE_FREQUENCY                   8950 non-null float64
ONEOFF_PURCHASES                    8950 non-null float64
INSTALLMENTS_PURCHASES              8950 non-null float64
PURCHASES_FREQUENCY                 8950 non-null float64
ONEOFF_PURCHASES_FREQUENCY          8950 non-null float64
PURCHASES_INSTALLMENTS_FREQUENCY    8950 non-null float64
CASH_ADVANCE_FREQUENCY              8950 non-null float64
CASH_ADVANCE_TRX                    8950 non-null float64
PURCHASES_TRX                       8950 non-null float64
Monthly_avg_purchase                8950 non-null float64
Monthly_cash_advance                8950 non-null float64
limit_usage                         8950 non-null float64
payment_minpay                      8950 non-null float64
Purchase_Type                       8950 non-null object
both_online/offline_installment     8950 non-null uint8
installment                         8950 non-null uint8
none                                8950 non-null uint8
online/offline_Purchases            8950 non-null uint8
dtypes: float64(13), object(1), uint8(4)
memory usage: 1013.9+ KB
```

7. Feature selection

Here as we are dealing with numerical variables, we are finding corelation between variables using a heat map technique.Following command is used to find the corelation between variables.

sns.heatmap(data_ dummy.corr())

```
Out[177]: <matplotlib.axes._subplots.AxesSubplot at 0x135a0e5a780>
```



Heat map shows that there is a corelation between many features, so in order to remove multicollinearity, we are using dimensionality reduction technique. But, before that we need to get all datapoints into a common scale, which is done through Feature scaling.

After Feature scaling, by using the dimensionality reduction technique, we can remove the variables which are corelated to each other.

8.Feature Scaling

Here we are using standardization method for getting all datapoints into a common scale. Using the following command:

data_scaled= StandardScaler().fit_transform(data_dummy)

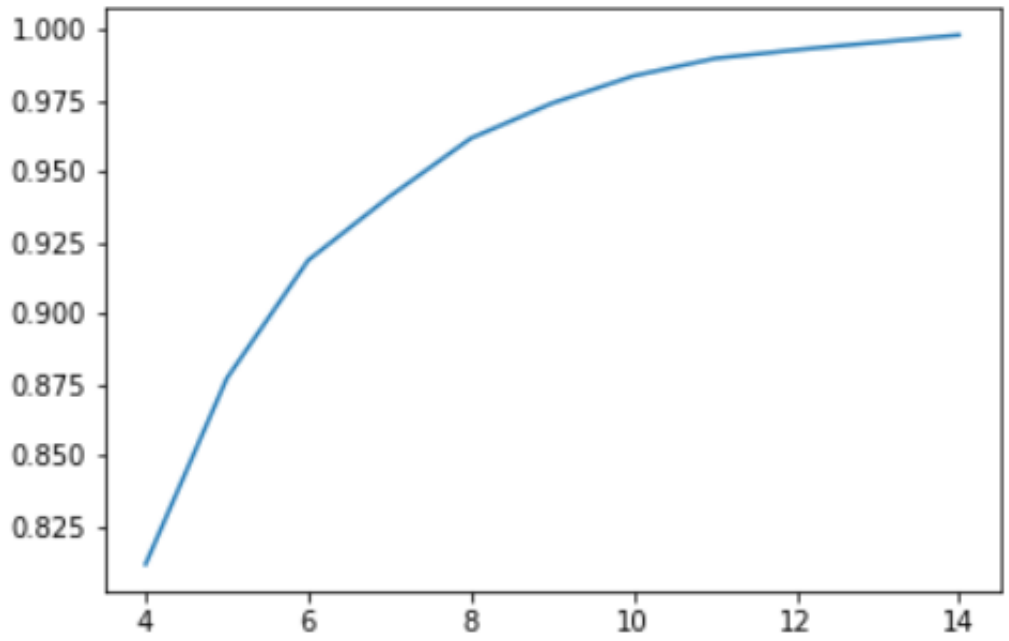
Applying PCA: (Feature selection)

Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation which converts a set of correlated variables to a set of uncorrelated variables. PCA is a most widely used tool in exploratory data analysis and in machine learning for predictive models. Moreover, PCA is an unsupervised statistical technique used to examine the interrelations among a set of variables. It is also known as a general factor analysis where regression determines a line of best fit.

Using following command to apply PCA technique:

```
var_ratio={}
for n in range(4,15):
   pc=PCA(n_components=n
   data_pca=pc.fit(data_scaled)
   var_ratio[n]=sum(data_pca.explained_variance_ratio_)
pc=PCA(n_components=5)
p=pc.fit(data_scaled)
```

here we are selecting 5 components because there is high variance in it.



pd.DataFrame(pc_final.components_.T, columns=['PC_' +str(i) for i in range(5)],index=columns_list)

	PC_0	PC_1	PC_2	PC_3	PC_4
BALANCE_FREQUENCY	0.029707	0.240072	-0.263140	-0.353549	-0.228681
ONEOFF_PURCHASES	0.214107	0.406078	0.239165	0.001520	-0.023197
INSTALLMENTS_PURCHASES	0.312051	-0.098404	-0.315625	0.087983	-0.002181
PURCHASES_FREQUENCY	0.345823	0.015813	-0.162843	-0.074617	0.115948
ONEOFF_PURCHASES_FREQUENCY	0.214702	0.362208	0.163222	0.036303	-0.051279
PURCHASES_INSTALLMENTS_FREQUENCY	0.295451	-0.112002	-0.330029	0.023502	0.025871
CASH_ADVANCE_FREQUENCY	-0.214336	0.286074	-0.278586	0.096353	0.360132
CASH_ADVANCE_TRX	-0.229393	0.291556	-0.285089	0.103484	0.332753
PURCHASES_TRX	0.355503	0.106625	-0.102743	-0.054296	0.104971
Monthly_avg_purchase	0.345992	0.141635	0.023986	-0.079373	0.194147
Monthly_cash_advance	-0.243861	0.264318	-0.257427	0.135292	0.268026
limit_usage	-0.146302	0.235710	-0.251278	-0.431682	-0.181885
payment_minpay	0.119632	0.021328	0.136357	0.591561	0.215446
both_online/offline_installment	0.241392	0.273676	-0.131935	0.254710	-0.340849
installment	0.082209	-0.443375	-0.208683	-0.190829	0.353821
none	-0.310283	-0.005214	-0.096911	0.245104	-0.342222
online/offline_Purchases	-0.042138	0.167737	0.472749	-0.338549	0.362585

9. K means Clustering & Error Metrices

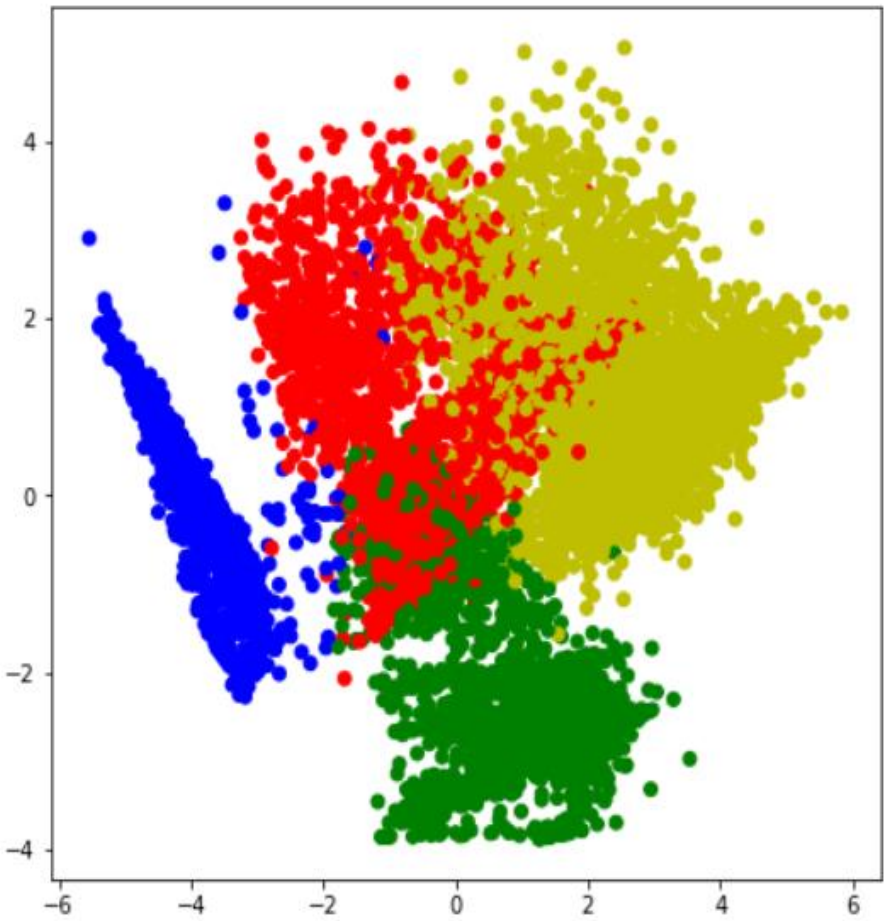
Trying with 4 clusters

We are using 4 clusters, using following command:

```
km_4=KMeans(n_clusters=4,random_state=112)
km_4.fit(reduced_cr)
```

Graphically representation of 4 clusters

```
color_map={0:'r',1:'b',2:'g',3:'y'}
label_color=[color_map[l] for l in km_4.labels_]
plt.figure(figsize=(7,7))
plt.scatter(reduced_cr[:,0],reduced_cr[:,1],c=label_color)
```



Checking error metrice for 4 clusters using Calinski Harabaz score and Silhouette Score

Calinski Harabaz score:

There is no "acceptable" cut-off value. You simply compare CH values. The higher the value, the "better" is the solution. If on the line-plot of CH values there appears that one solution give a peak or at least an abrupt elbow, choose it. If, on the contrary, the line is smooth - horizontal or ascending or descending - then there is no reason to prefer one solution to others.

Silhouette Score:

Silhouette values lies in the range of [-1, 1]. A value of +1 indicates that the sample is far away from its neighboring cluster and very close to the cluster its assigned. Similarly, value of -1 indicates that the point is close to its neighboring cluster than to the cluster its assigned. And, a value of 0 means its at the boundary of the distance between the two cluster. Value of +1 is idea and -1 is least preferred. Hence, higher the value better is the cluster configuration.

The command functions, to check the Calinski Harabaz score and Silhouette Score are:

calinski_harabaz_score(reduced_cr,km_4.labels_)

silhouette_score(reduced_cr,km_4.labels_)

Calinski Harabaz score for 4 clusters are 6164.054484808305

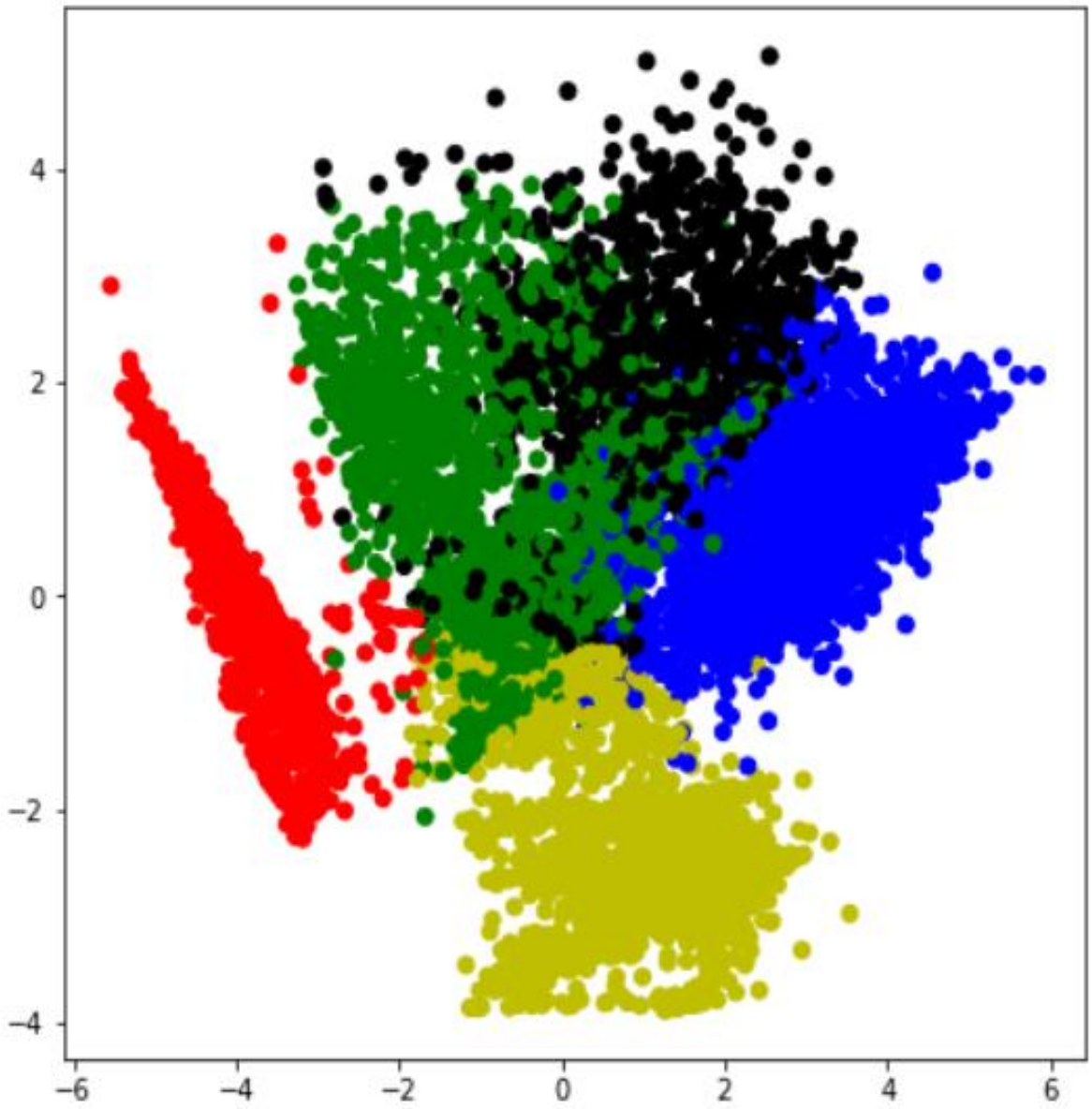
Silhouette Score for 4 clusters are 0.4592855176006003

Trying with 5 clusters:

```
km_5=KMeans(n_clusters=5,random_state=112)
km_5.fit(reduced_cr)
```

Graphically representation of 5 clusters

```
color_map={0:'r',1:'b',2:'g',3:'y',4:'k'}
label_color=[color_map[l] for l in km_5.labels_]
plt.figure(figsize=(7,7))
plt.scatter(reduced_cr[:,0],reduced_cr[:,1],c=label_color)
```



Checking error metrice for 5 clusters using Calinski Harabaz score and Silhouette Score

The command functions, to check the Calinski Harabaz score and Silhouette Score are:

calinski_harabaz_score(reduced_cr,km_4.labels_)

silhouette_score(reduced_cr,km_4.labels_)

Calinski Harabaz score for 5 clusters are 5867.020860915632

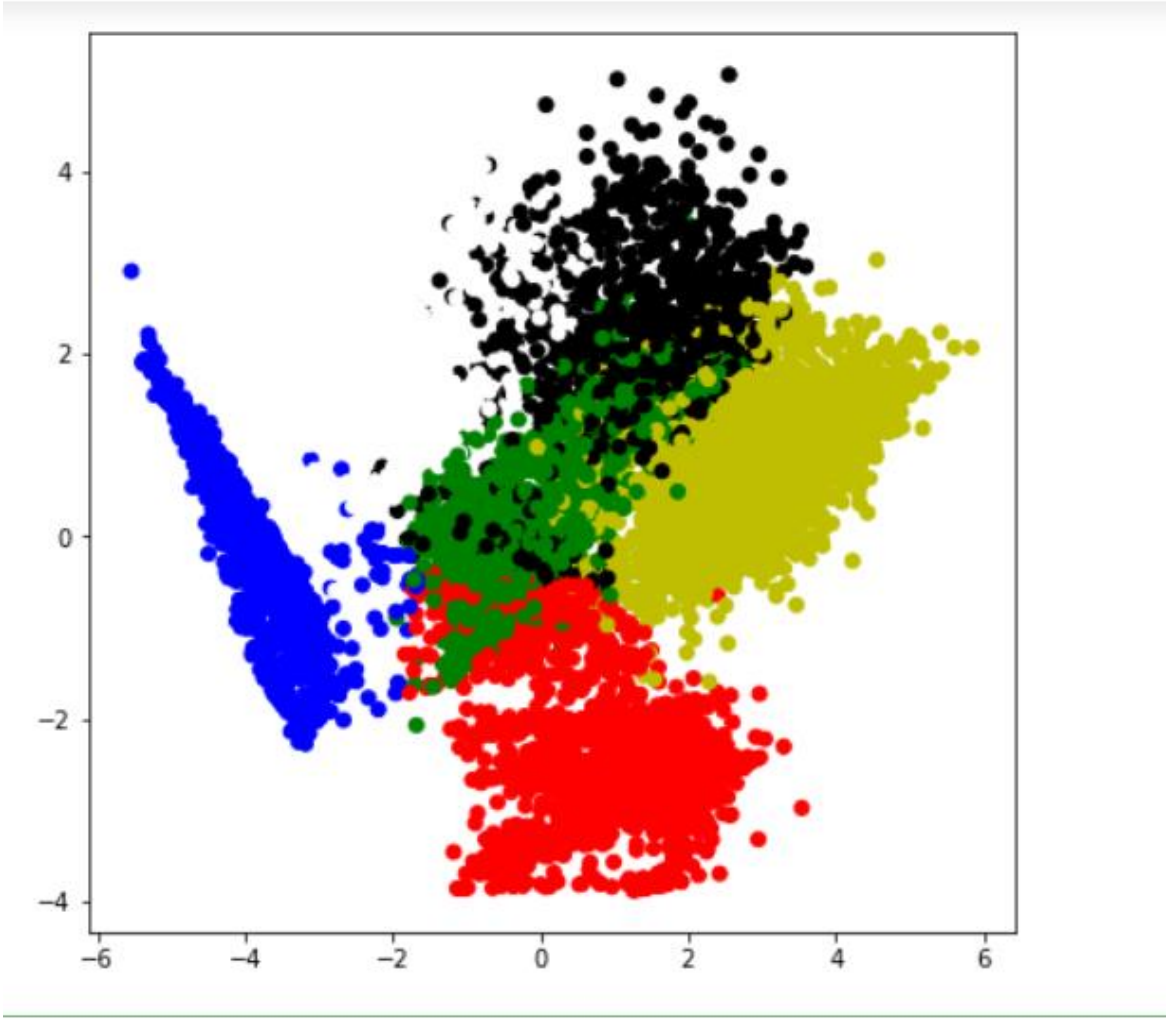
Silhouette Score for5 clusters are 0.4557517692881165

Trying with 6 clusters:

```
km_6=KMeans(n_clusters=6,random_state=112)
km_6.fit(reduced_cr)
```

Graphically representation of 6 clusters

```
color_map={0:'r',1:'b',2:'g',3:'y',4:'w',5:'k'}
label_color=[color_map[l] for l in km_6.labels_]
plt.figure(figsize=(7,7))
plt.scatter(reduced_cr[:,0],reduced_cr[:,1],c=label_color)
```



Checking error metrice for 6 clusters using Calinski Harabaz score and Silhouette Score

The command functions, to check the Calinski Harabaz score and Silhouette Score are:

calinski_harabaz_score(reduced_cr,km_6.labels_)

silhouette_score(reduced_cr,km_6.labels_)

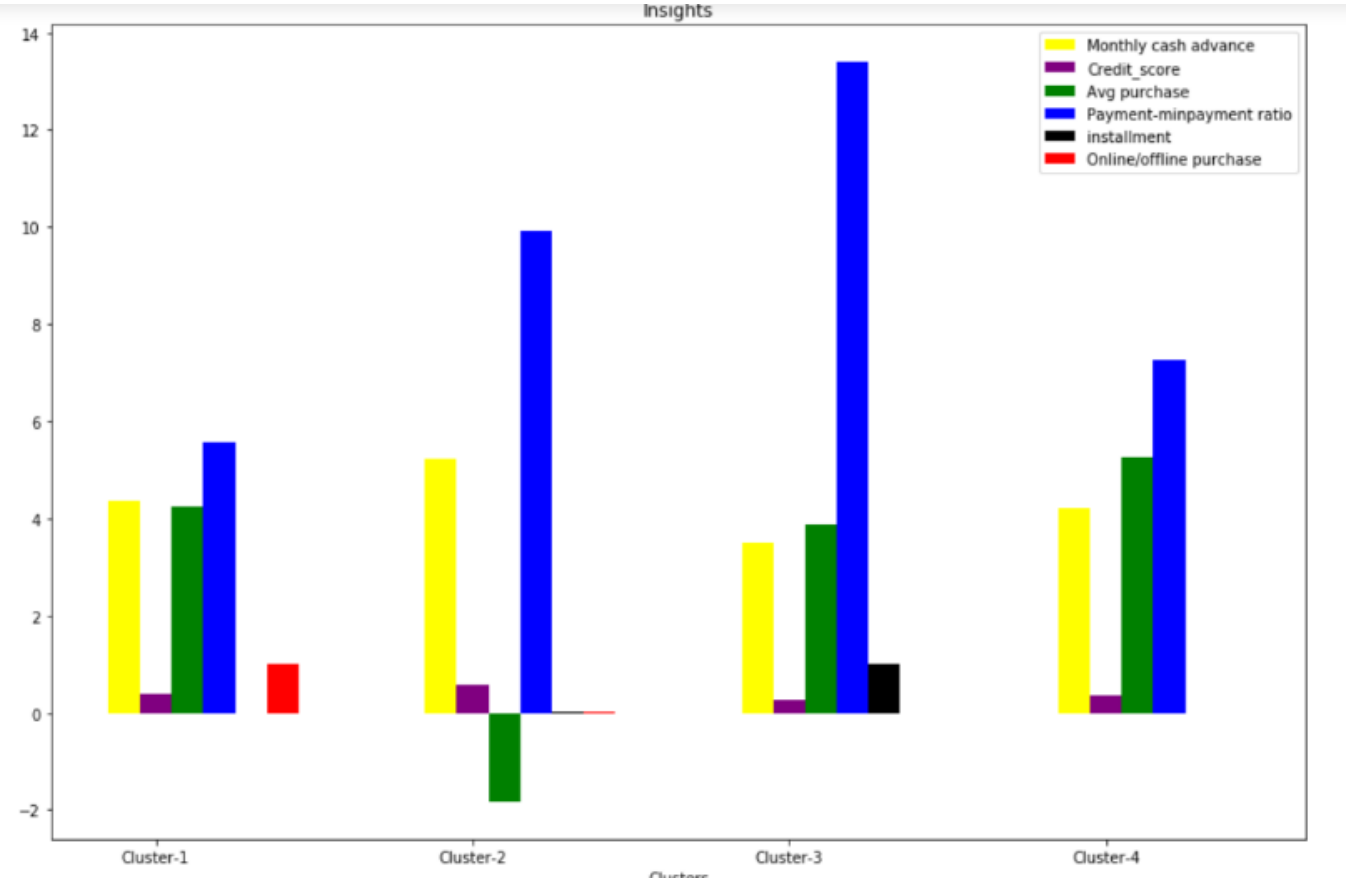
Calinski Harabaz score for 6 clusters are 5669.505574153349

Silhouette Score for 6 clusters are 0.450412553407320 84

Error Metrices	For 4 clusters	For 5 clusters	For 6 clusters
Calinski Harabaz score	6164.054484808305	5867.020860915632	5669.505574153349
Silhouette Score	0.45925855176006003	0.4557517692881165	0.45041255340732084

As Calinski Haraba score and Silhousette score is better for 4 clusters while compared to other clusters, we are choosing for 4 clusters.

The graphicall representation on categorizing cutomers using 4 clusters:



10. Conclusion form the theory of 4 clusters:

1. The cluster-1 has highest 'online/offline purchases, least 'Payment-ratio', while compared to other clusters.

2. The cluster-2 has highest 'Monthly cash advance' and least 'Average purchases'. It also has 'online/offline' and 'installment' purchases. It also maintained good credit score, compared to other clusters.

3. The cluster-3 has highest 'Payment ratio' and 'installment' purchases, compared to other clusters. It also has good 'Average Purchase'. But it does not have any 'installment' and 'online/offline' purchases. It also has least credit-score and Monthly cash advance.

4. The cluster-4 has highest 'Average Purchase'. It has good 'Monthly cash advance'. It also does not have any online/offline and instalment purchases.

11. Suggestions on Marketing

1. Category 1: This group have maximum online/offline purchases but has least payment-ratio. So, we can conclude that they are not purchasing much through online/offline, may be there purchases are loan cuttings, bill payments, etc.

2. Category 2: This group takes much cash in advance and do not spend more money, though they make some online/offline and instalment purchases. We also need to remember that these are the group that maintained, good credit score, which is more important. We can increase their interest for purchasing, by giving reward points, offers on every transaction.

3. Category 3: This group has least monthly cash advance, so we need to lower the interest rate, and give cash backs on their every purchase.

4. Category 4: This group are main useful to gain profit as they take cash in advance, make purchases' and also does online/offline and installment purchases. We can give them loyalty cards.