CREDIT-CARD SEGMENTATION

APPLIED DATA SCIENCE CAPSTONE PROJECT

author: Zoltan Farkas

TABLE OF CONTENTS

INTRODUCTION

Segmentation in marketing is a technique used to divide customers or other entities into groups based on attributes such as behaviour or demographics. It is useful to identify segments of customers who may respond in a similar way to specific marketing techniques such as email subject lines or display advertisements.

As it gives businesses the ability to tailor marketing messages and timing to generate better response rates and provide improved consumer experiences. In this project, the definition of a marketing strategy using a machine learning technique requires the development of a customer segment.

The goal of this analysis report is to discover the Customer Segmentation of a bank, by looking through their behavior/profile while using Credit Card.

Hopefully, I can get a clear segmentation of the customer, so I can deploy effective marketing campaign or sales promotion to the targeted costumer.

DATA ACQUISITION AND CLEANING

The sample dataset summarizes the usage behavior of about 8000+ active credit card holders during 6 months. The file is at a customer level with 18 behavioral variables.

DATA DICTIONARY:

- CUST_ID: Identification of Credit Card holder (Categorical)
- BALANCE: Balance amount left in their account to make purchases
- BALANCE_FREQUENCY: Ratio of last 12 months with balance
- PURCHASES: Amount of purchases made from account
- ONEOFF_PURCHASES: Total amount of one-off purchases
- INSTALLMENTS_PURCHASES: Total amount of installment purchases
- CASH_ADVANCE: Cash in advance given by the user
- PURCHASES_ FREQUENCY: Frequency of purchases (Percent of months with at least one 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_PAYMEN: Percentage of months with full payment of the due statement balance
- TENURE: Number of months as a customer

DATA CLEANING

MISSING VALUES

Missing data is an everyday problem that a data professional need to deal with. Though there are many articles, blogs, videos already available, I found it is difficult to find a concise consolidated information in a single place. That's why I am putting my effort here, hoping it will be useful to any data practitioner or enthusiast.

```
credit.isnull().any()

CUST_ID                             False
BALANCE                             False
BALANCE_FREQUENCY                   False
PURCHASES                           False
ONEOFF_PURCHASES                    False
INSTALLMENTS_PURCHASES              False
CASH_ADVANCE                        False
PURCHASES_FREQUENCY                 False
ONEOFF_PURCHASES_FREQUENCY          False
PURCHASES_INSTALLMENTS_FREQUENCY    False
CASH_ADVANCE_FREQUENCY              False
CASH_ADVANCE_TRX                    False
PURCHASES_TRX                       False
CREDIT_LIMIT                         True
PAYMENTS                            False
MINIMUM_PAYMENTS                     True
PRC_FULL_PAYMENT                    False
TENURE                              False
dtype: bool
```

Solution: Missing values we need to remove with median.

```
### Missing values we need to remove with median.

# CREDIT_LIMIT
credit['CREDIT_LIMIT'].fillna(credit['CREDIT_LIMIT'].median(),inplace=True)
credit['CREDIT_LIMIT'].count()

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

# Now again check the missing values.
credit.isnull().any()

CUST_ID                             False
BALANCE                             False
BALANCE_FREQUENCY                   False
PURCHASES                           False
ONEOFF_PURCHASES                    False
INSTALLMENTS_PURCHASES              False
CASH_ADVANCE                        False
PURCHASES_FREQUENCY                 False
ONEOFF_PURCHASES_FREQUENCY          False
PURCHASES_INSTALLMENTS_FREQUENCY    False
CASH_ADVANCE_FREQUENCY              False
CASH_ADVANCE_TRX                    False
PURCHASES_TRX                       False
CREDIT_LIMIT                        False
PAYMENTS                            False
MINIMUM_PAYMENTS                    False
PRC_FULL_PAYMENT                    False
TENURE                              False
dtype: bool
```

DUPLICATION

"Duplication" just means that you have repeated data in your dataset. This could be due to things like data entry errors or data collection methods. For example, if you're using a web scraper you may happen to scrape the same webpage more than once, or the same information from two different pages. Whatever the reason, deduplication can lead you to make incorrect conclusions by leading you to believe that some observations are more common than they really are.

Solution: Remove duplicates

```
# Remove duplicates
credit = credit.drop_duplicates()
print ("\nNumber of Unique values : \n",credit.nunique())

Number of Unique values :
 CUST_ID                             8950
BALANCE                             8871
BALANCE_FREQUENCY                     43
PURCHASES                           6203
ONEOFF_PURCHASES                    4014
INSTALLMENTS_PURCHASES              4452
CASH_ADVANCE                        4323
PURCHASES_FREQUENCY                   47
ONEOFF_PURCHASES_FREQUENCY            47
PURCHASES_INSTALLMENTS_FREQUENCY      47
CASH_ADVANCE_FREQUENCY                54
CASH_ADVANCE_TRX                      65
PURCHASES_TRX                        173
CREDIT_LIMIT                         205
PAYMENTS                            8711
MINIMUM_PAYMENTS                    8636
PRC_FULL_PAYMENT                      47
TENURE                                 7
dtype: int64
```

EXPLORATORY DATA ANALYSIS

In statistics, exploratory data analysis is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.

CUSTOMER ANALYSE

Customer analysis is a critical component of any business plan in all stages of growth. When you analyze your customers, you define who your target market is, and decide how you'll reach them

To better serve their customer base and more effectively acquire new customers, organizations need to delve into the details of individual interactions to understand the relationship between each customer touch point and the value it delivers to customers.

It's a tide that lifts all boats: marketing can create campaigns with better wording, sales can come up with better pitches, product development will know what features to prioritize, etc.

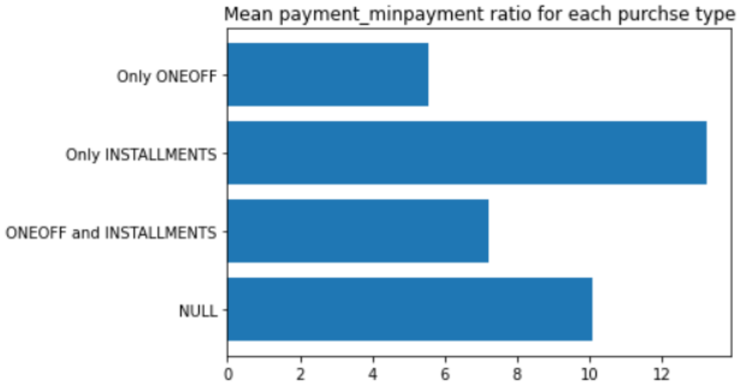
AVERAGE PAYMENT_MINPAYMENT RATIO FOR EACH PURCHSE TYPE

```
x=credit.groupby('purchase_type').apply(lambda x: np.mean(x['payment_minpay']))
type(x)
x.values
```

```
array([10.08745106,  7.23698216, 13.2590037 ,  5.57108156])
```

```
ax.barh?
fig,ax=plt.subplots()
ax.barh(y=range(len(x)), width=x.values,align='center')
ax.set(yticks= np.arange(len(x)),yticklabels = x.index);
plt.title('Mean payment_minpayment ratio for each purchse type')
```

```
Object `ax.barh` not found.
Text(0.5, 1.0, 'Mean payment_minpayment ratio for each purchse type')
```

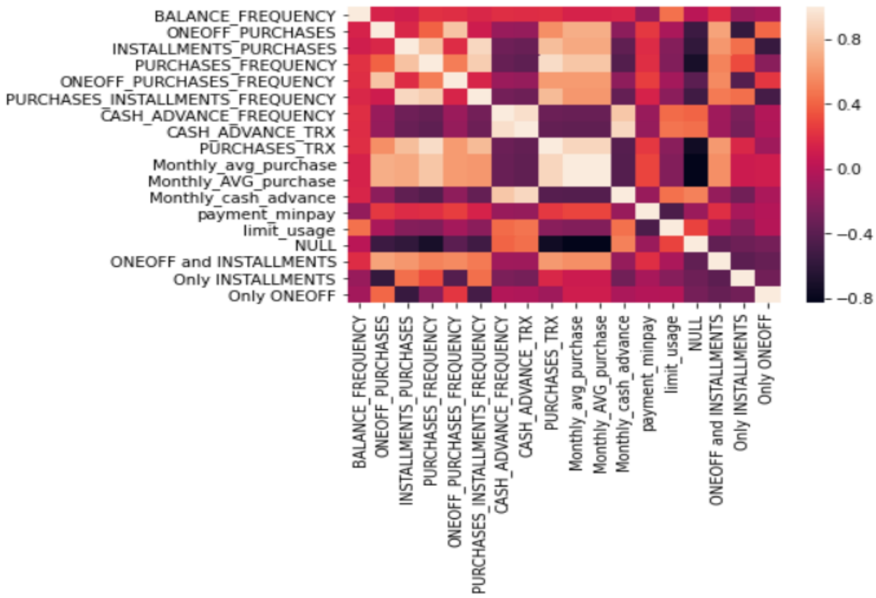


REMOVE OUTLIER EFFECT

Since there are variables having extreme values so I am doing log-transformation on the dataset to remove outlier effect

```
sns.heatmap(cr_dummy.corr())
```

```
<AxesSubplot:>
```



EXPLAINING COMPONETS VARIANCE

There are quite a few explanations of the principal component analysis (PCA) on the internet, some of them quite insightful. However, one issue that is usually skipped over is the variance explained by principal components, as in “the first 5 PCs explain 86% of variance”. So this is my attempt to explain the explained variance.

PCA, being derived from noisy data, is itself noisy. For instance, it would be a mistake to conclude that there exists a parameter that explains 88% of the variability in the actual quantities we have measured. The true fraction of total variance that can be captured by a single variable in this case is only around 60%, and we would get closer to it if we increased our sample size.

```
var_ratio={}
for n in range(2,18):
    pc=PCA(n_components=n)
    cr_pca=pc.fit(cr_scaled)
    var_ratio[n]=sum(cr_pca.explained_variance_ratio_)
```

```
var_ratio
```

```
{2: 0.6000853470314143,
 3: 0.7392613086744784,
 4: 0.816826812351074,
 5: 0.8807344239563172,
 6: 0.920025934206578,
 7: 0.9417262399160146,
 8: 0.9611062650098873,
 9: 0.9728589902487423,
 10: 0.982365285390013,
 11: 0.9901299690140261,
 12: 0.9930752029485337,
 13: 0.9955801684308823,
 14: 0.998028469236758,
 15: 0.999619188003833,
 16: 1.0,
 17: 1.0}
```

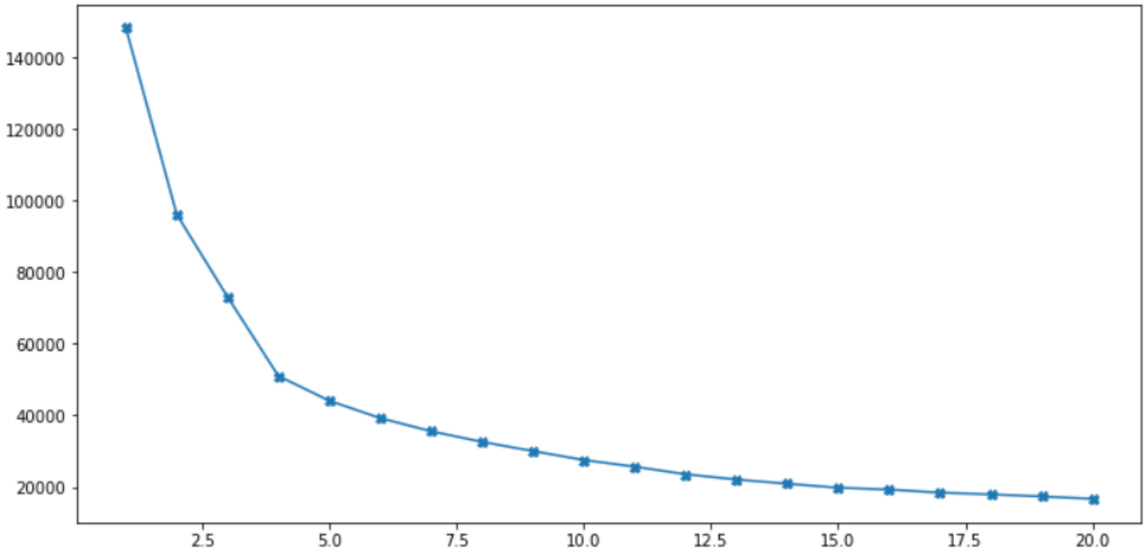
Since 5 components are explaining about 87% variance so we select 5 components

```
pc_final=PCA(n_components=6).fit(cr_scaled)
reduced_cr=pc_final.fit_transform(cr_scaled)

dd=pd.DataFrame(reduced_cr)
dd.head()
```

	0	1	2	3	4	5
0	-0.456062	-2.756620	0.379383	-0.444900	0.008705	0.021145
1	-4.262327	0.244241	-0.535652	1.028881	-0.323219	-0.566935
2	1.465947	1.503989	2.686314	-1.830205	-0.231250	-0.612388
3	-0.635043	0.906002	2.524957	-1.435780	0.841235	1.417289
4	-1.730629	-0.164991	2.283085	-1.535454	-0.772215	-0.682158

CLUSTERING

Based on the intuition on type of purchases made by customers and their distinctive behavior exhibited based on the purchase_type (as visualized above in Insights from KPI) I am starting with 4 clusters.



From above graph we will find elbow range. here it is 4,5,6

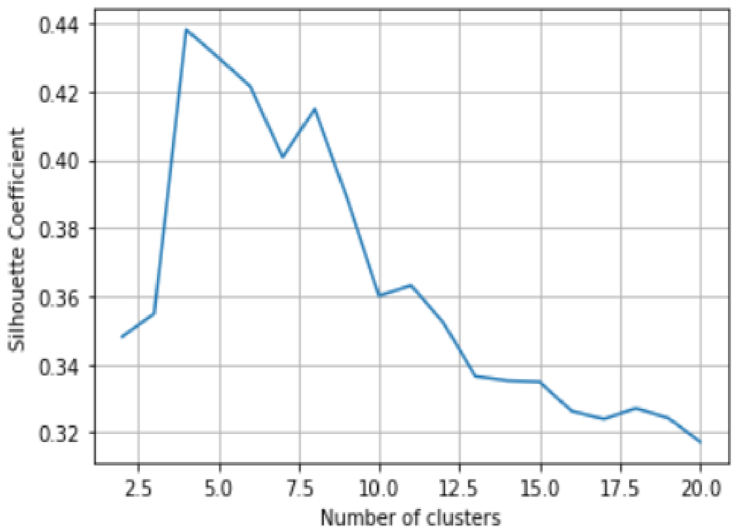
CLUSTER VALIDATION

Silhouette refers to a method of interpretation and validation of consistency within clusters of data. The technique provides a succinct graphical representation of how well each object has been classified. The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).

The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. If most objects have a high value, then the clustering configuration is appropriate. If many points have a low or negative value, then the clustering configuration may have too many or too few clusters.

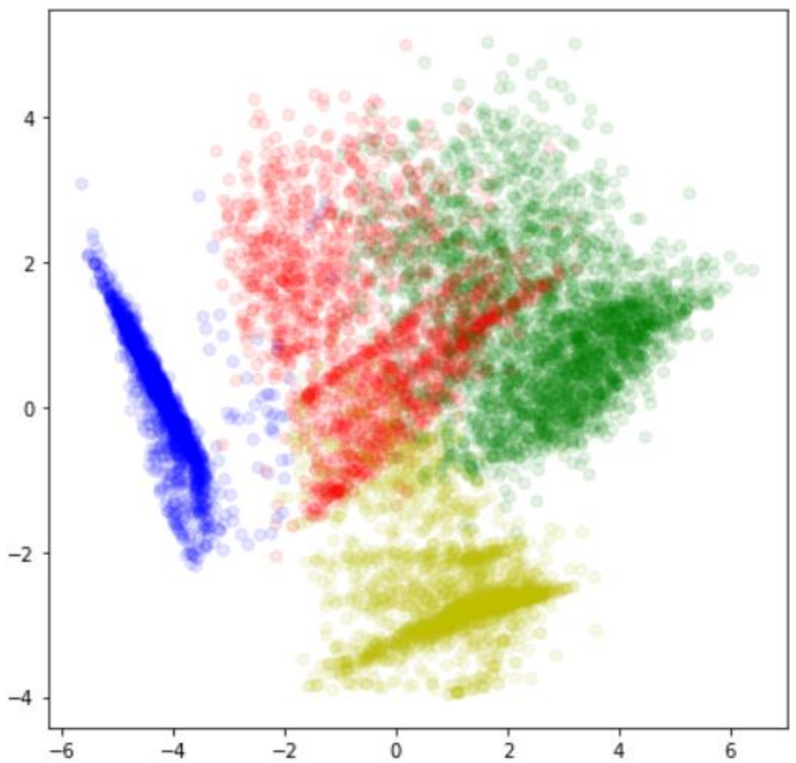
```
# calculate SC for K=3 through K=12
k_range = range(2, 21)
scores = []
for k in k_range:
    km = KMeans(n_clusters=k, random_state=1)
    km.fit(reduced_cr)
    scores.append(metrics.silhouette_score(reduced_cr, km.labels_))
```

```
# plot the results
plt.plot(k_range, scores)
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Coefficient')
plt.grid(True)
```

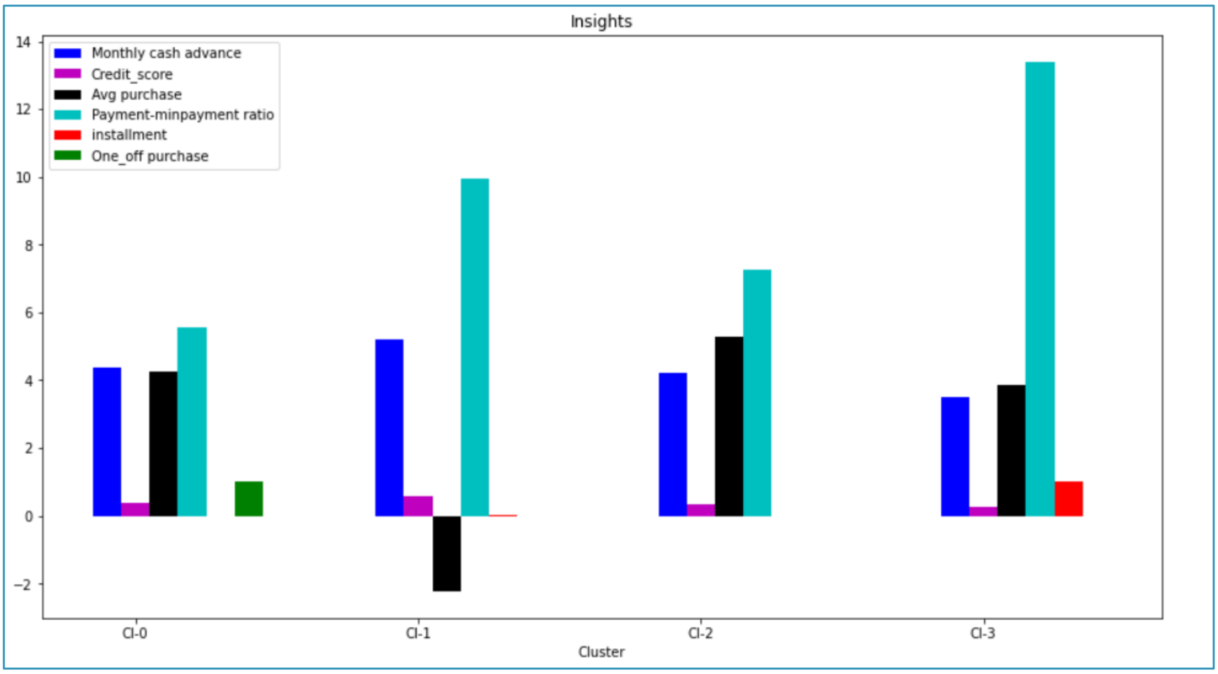


```python
color_map={0:'r',1:'b',2:'g',3:'y'}
label_color=[color_map[l] for l in km_4.labels_]
plt.figure(figsize=(7,7))
plt.scatter(reduced_cr[:,0],reduced_cr[:,1],c=label_color,cmap='Spectral',alpha=0.1)
```

```
<matplotlib.collections.PathCollection at 0x7fc7d2d3b470>
```



INTERPRATE RESULT



RESULT ANALYSE

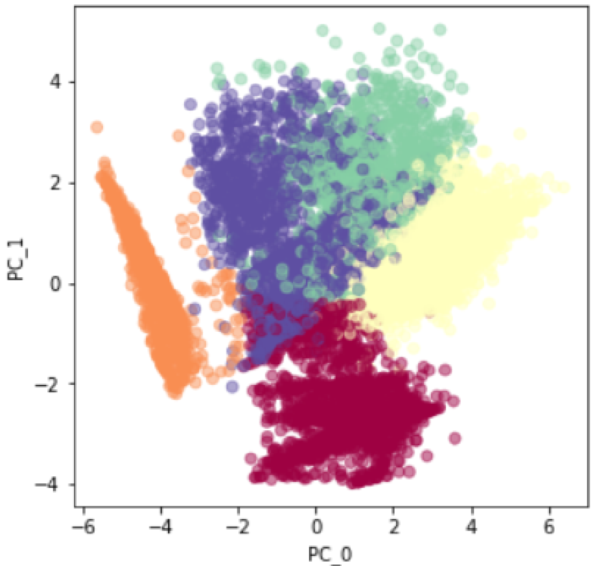
- Cluster 2 is the group of customers who have highest Monthly_avg purchases and doing both installment as well as one_off purchases, have comparatively good credit score. This group is about 31% of the total customer base
- Cluster 1 is taking maximum advance_cash and is paying comparatively less minimum payment and poor credit_score & doing no purchase transaction. This group is about 23% of the total customer base
- Cluster 0 customers are doing maximum One_Off transactions and least payment ratio. This group is about 21% of the total customer base
- Cluster 3 customers have maximum credit score and are paying dues and are doing maximum installment purchases. This group is about 25% of the total customer base

FINDING BEHAVIOUR WITH 5 CLUSTERS

```
km_5=KMeans(n_clusters=5,random_state=123)
km_5=km_5.fit(reduced_cr)
```

```
plt.figure(figsize=(5,5))
plt.scatter(reduced_cr[:,0],reduced_cr[:,1],c=km_5.labels_,cmap='Spectral',alpha=0.5)
plt.xlabel('PC_0')
plt.ylabel('PC_1')
```

```
Text(0, 0.5, 'PC_1')
```



Conclusion With 5 clusters :

```
# Finding Mean of features for each cluster

cluster_df_5=pd.concat([cre_original[col_kpi],pd.Series(km_5.labels_,name='Cluster_5')],axis=1)

cluster_df_5.groupby('Cluster_5')\
.apply(lambda x: x[col_kpi].mean()).T
```

Cluster_5	0	1	2	3	4
PURCHASES_TRX	11.918881	0.032614	34.706599	27.516892	7.095596
Monthly_avg_purchase	47.364663	0.070211	211.315519	140.739289	68.880850
Monthly_cash_advance	20.561824	184.567083	4.019410	249.852297	73.919654
limit_usage	0.248925	0.576341	0.258395	0.601237	0.376991
CASH_ADVANCE_TRX	0.550583	6.421103	0.151777	10.349099	2.695489
payment_minpay	13.806995	9.947252	8.698685	3.637777	5.562159
ONEOFF and INSTALLMENTS	0.000000	0.000000	1.000000	0.897523	0.003759
Only INSTALLMENTS	1.000000	0.016787	0.000000	0.090090	0.000000
Only ONEOFF	0.000000	0.003837	0.000000	0.012387	0.996241
NULL	0.000000	0.979376	0.000000	0.000000	0.000000
CREDIT_LIMIT	3230.169410	4037.969624	5734.270522	5868.750000	4494.084399

We have a group of customers (cluster 2) having highest avergae purchases but there is Cluster 4 also having highest cash advance & secong highest purchase behaviour but their type of purchases are same.

Cluster 0 and Cluster 4 are behaving similar in terms of Credit_limit and have cash transactions is on higher side. So we don't have quite distinguishable characteristics with 5 clusters

```
# percentage of each cluster

s1=cluster_df_5.groupby('Cluster_5').apply(lambda x: x['Cluster_5'].value_counts())
print (s1)

print ("Cluster-5"),'\n'
per_5=pd.Series((s1.values.astype('float')/ cluster_df_5.shape[0])*100,name='Percentage')
print (pd.concat([pd.Series(s1.values,name='Size'),per_5],axis=1))
```

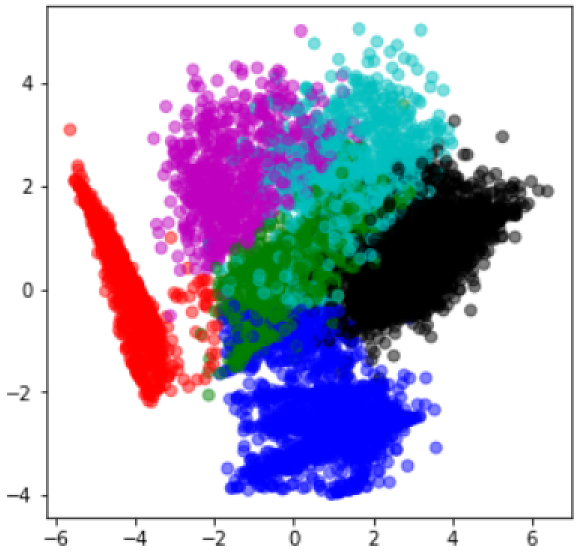
```
Cluster_5
0          0    2145
1          1    2085
2          2    1970
3          3     888
4          4    1862
Name: Cluster_5, dtype: int64
Cluster-5
   Size  Percentage
0  2145   23.966480
1  2085   23.296089
2  1970   22.011173
3   888    9.921788
4  1862   20.804469
```

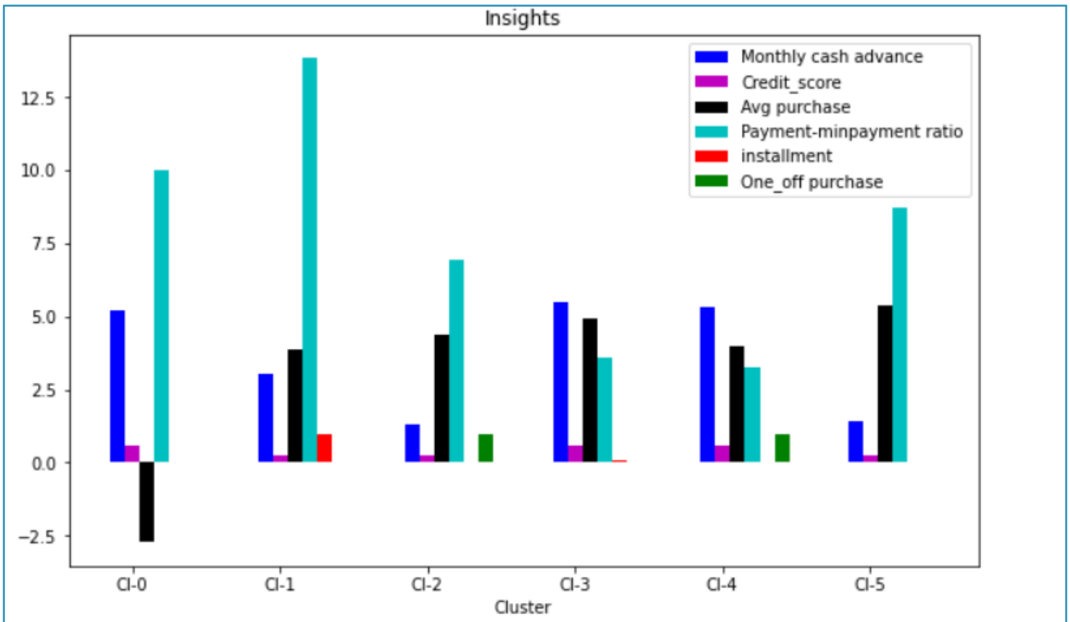
FINDING BEHAVIOR WITH 6 CLUSTERS

```
km_6=KMeans(n_clusters=6).fit(reduced_cr)
```

```
color_map={0:'r',1:'b',2:'g',3:'c',4:'m',5:'k'}
label_color=[color_map[l] for l in km_6.labels_]
plt.figure(figsize=(5,5))
plt.scatter(reduced_cr[:,0],reduced_cr[:,1],c=label_color,cmap='Spectral',alpha=0.5)
```

```
<matplotlib.collections.PathCollection at 0x7fc7d21fb2b0>
```





Conclusion with 6 clusters:

- Here also groups are overlapping. (Cl-0 and Cl-2 behaving same)

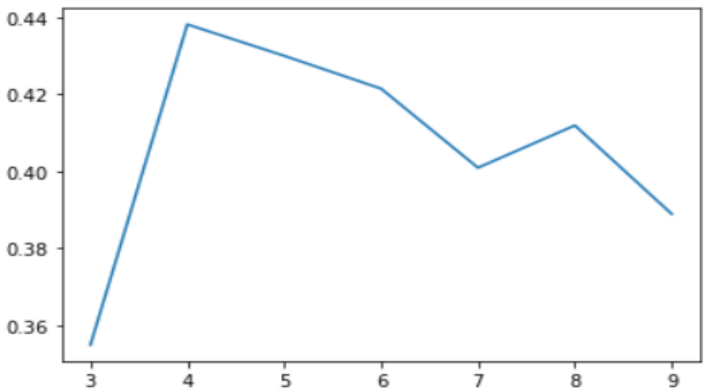
CHECKING PERFORMANCE METRICS FOR KMEANS

Kmeans algorithm is an iterative algorithm that tries to partition the dataset into Kpre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum.

The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster. I am validating performance with 2 metrics Calinski harabaz and Silhouette score

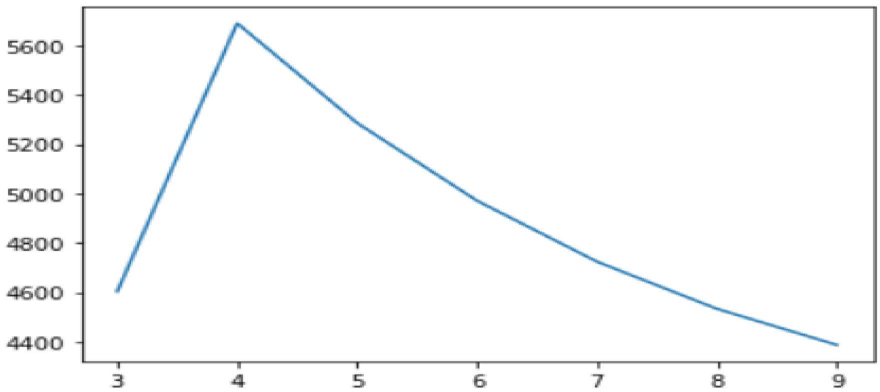
```
score={}
score_c={}
for n in range(3,10):
    km_score=KMeans(n_clusters=n)
    km_score.fit(reduced_cr)
    score_c[n]=calinski_harabaz_score(reduced_cr,km_score.labels_)
    score[n]=silhouette_score(reduced_cr,km_score.labels_)

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

```
<AxesSubplot:>
```




```
pd.Series(score_c).plot()
```

```
<AxesSubplot:>
```



RESULTS

- **Cluster 0** customers are doing maximum One_Off transactions and least payment ratio and credit_score on lower side ***This group is about 21% of the total customer base

- **Cluster 1** is taking maximum advance_cash and is paying comparatively less minimum payment and poor credit_score & doing no purchase transaction. This group is about 23% of the total customer base***

- **Cluster 2** is the group of customers who have highest Monthly_avg purchases and doing both installment as well as one_off purchases, have comparatively good credit score. This group is about 31% of the total customer base

- **Cluster 3** customers have maximum credit score and are paying dues and are doing maximum installment purchases.*** This group is about 25% of the total customer base

SUGGESTED MARKETING STRATEGY

- **Group 0**

This group is has minimum paying ratio and using card for just oneoff transactions (may be for utility bills only). This group seems to be risky group.

- **Group 1**

They have poor credit score and taking only cash on advance. We can target them by providing less interest rate on purchase transaction

- **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

- **Group 3**

This group is performing best among all as cutomers are maintaining good credit score and paying dues on time.

  - Giving rewards point will make them perform more purchases.