**CREDIT CARD**

**SEGMENTATION**

**By**

**Jerin Joseph**

**Project Name – Credit Card Segmentation**

**Deadline - 15 Days**

**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.

**Expectations from the student:**

1. Advanced data preparation. Build an ‘enriched’ customer profile by deriving ‘intelligent’ KPI’s 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 Set:**

credit-card-data.csv

**Number of attributes:**

* CUST\_ID: Credit card holder ID
* BALANCE: Monthly average balance (based on daily balance averages)
* BALANCE\_FREQUENCY: Ratio of last 12 months with balance
* PURCHASES: Total purchase amount spent during last 12 months
* ONEOFF\_PURCHASES: Total amount of one-off purchases
* INSTALLMENTS\_PURCHASES: Total amount of installment purchases
* 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

**Overview**

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

Let’s understand this with an example. Suppose, we are the head of a rental store and wish to understand preferences of our customers to scale up our business. Is it possible for us to look at details of each costumer and devise a unique business strategy for each one of them? Definitely not. But what we can do is to cluster all of our customers into say 10 groups based on their purchasing habits and use a separate strategy for costumers in each of these 10 groups. And this is what we call clustering.

**Types of clustering algorithms:**

Since the task of clustering is subjective, the means that can be used for achieving this goal are plenty. Every methodology follows a different set of rules for defining the ‘similarity’ among data points. In fact, there are more than 100 clustering algorithms known. But few of the algorithms are used popularly, let’s look at them in detail:

**Connectivity models:** As the name suggests, these models are based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away. These models can follow two approaches. In the first approach, they start with classifying all data points into separate clusters & then aggregating them as the distance decreases. In the second approach, all data points are classified as a single cluster and then partitioned as the distance increases. Also, the choice of distance function is subjective. These models are very easy to interpret but lacks scalability for handling big datasets. Examples of these models are hierarchical clustering algorithm and its variants.

**Centroid models:** These are iterative clustering algorithms in which the notion of similarity is derived by the closeness of a data point to the centroid of the clusters. K-Means clustering algorithm is a popular algorithm that falls into this category. In these models, the no. of clusters required at the end have to be mentioned beforehand, which makes it important to have prior knowledge of the dataset. These models run iteratively to find the local optima.

**Distribution models:** These clustering models are based on the notion of how probable is it that all data points in the cluster belong to the same distribution (For example: Normal, Gaussian). These models often suffer from overfitting. A popular example of these models is Expectation-maximization algorithm which uses multivariate normal distributions.

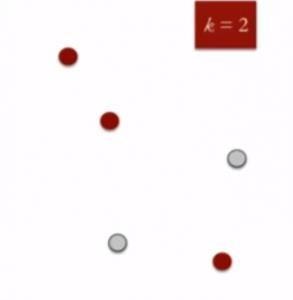
**Density Models:** These models search the data space for areas of varied density of data points in the data space. It isolates various different density regions and assign the data points within these regions in the same cluster. Popular examples of density models are DBSCAN and OPTICS.

Now I will be taking you through the most popular clustering algorithms in detail – K Means clustering

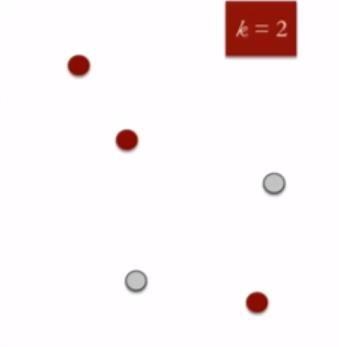
**K Means Clustering**

K means is an iterative clustering algorithm that aims to find local maxima in each iteration. This algorithm works in these 5 steps:

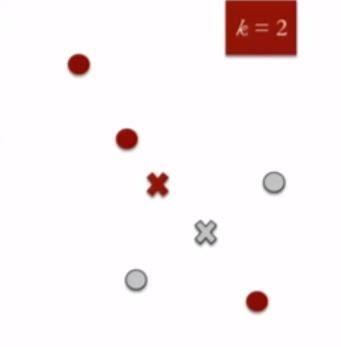
1. Specify the desired number of clusters K: Let us choose k=2 for these 5 data points in 2-D space.



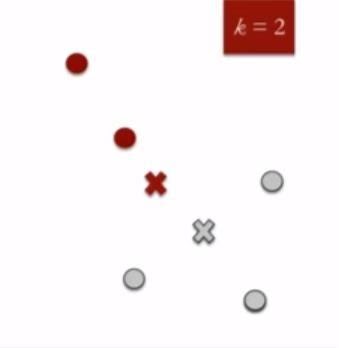
1. Randomly assign each data point to a cluster: Let’s assign three points in cluster 1 shown using red colour and two points in cluster 2 shown using grey colour.



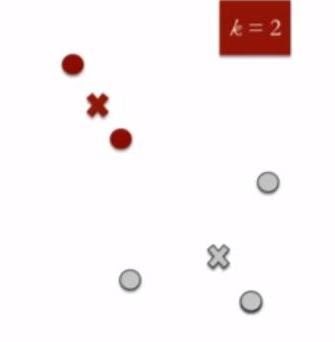
1. Compute cluster centroids: The centroid of data points in the red cluster is shown using red cross and those in grey cluster using grey cross.



1. Re-assign each point to the closest cluster centroid: Note that only the data point at the bottom is assigned to the red cluster even though its closer to the centroid of grey cluster. Thus, we assign that data point into grey cluster



1. Re-compute cluster centroids: Now, re-computing the centroids for both the clusters.



1. Repeat steps 4 and 5 until no improvements are possible: Similarly, we’ll repeat the 4th and 5th steps until we’ll reach global optima. When there will be no further switching of data points between two clusters for two successive repeats. It will mark the termination of the algorithm if not explicitly mentioned.

**PREVIEW OF OUR PROJECT:**

From the problem statement and the input attributes we can clearly understand that this belongs to unsupervised machine learning model in which there will be no target label, every attribute has to be considered as input feature, we have to find the hidden patterns among these features and establish the hidden patterns.

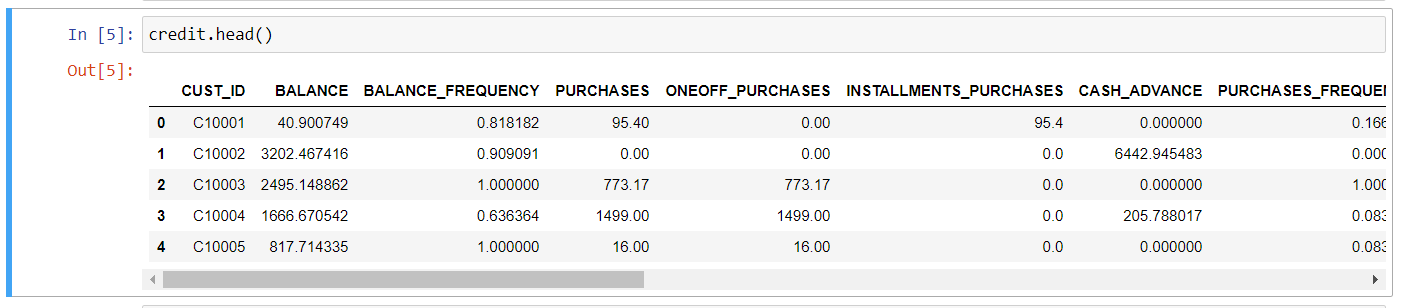
We intend to segment the customer who are using credit cards, by using K Mean model as it a clustering project and comes under unsupervised learning. We will analyse the customer insights and derive the KPI’s which would enable the organization to focus on the key areas. To start with, we will be using Python and later on R.

**Business Problem: Credit Card Segmentation**

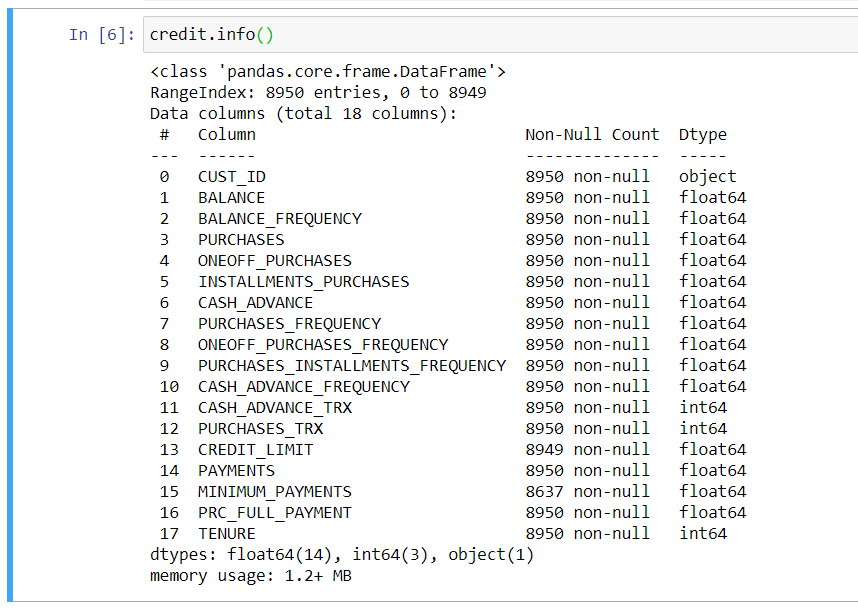
**LOAD THE DATA**

|  |
| --- |
| import os  import pandas as pd  import seaborn as sns  import matplotlib.pyplot as plt  import numpy as np |

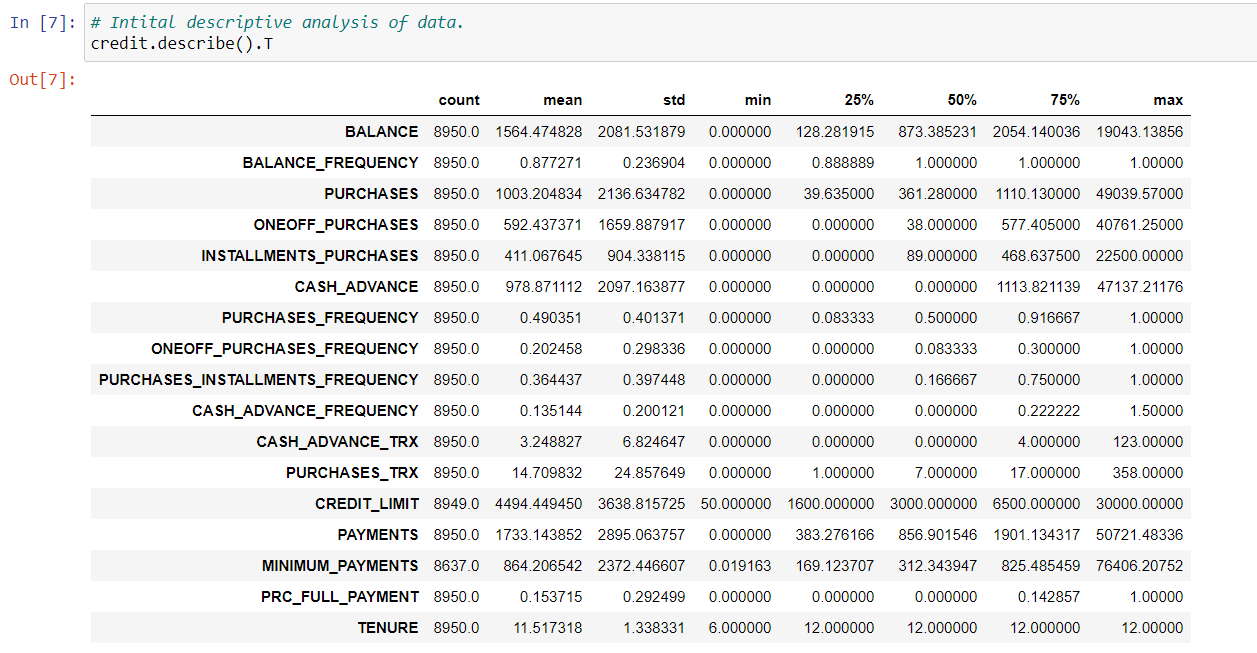
|  |
| --- |
| #set working directory  path = "C:/Users/jerin/Desktop/PYTHON WORK/PYTHON PROJECT/EDWISOR PROJECTS/CREDIT CARD SEGMENTATION"  os.chdir(path)  os.getcwd()  credit = pd.read\_csv("CC GENERAL.csv") |



|  |
| --- |
| credit.info() |

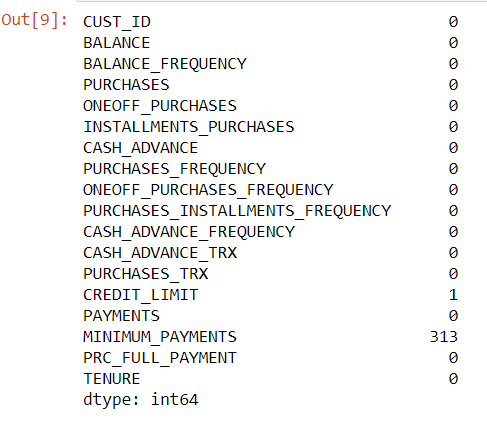


|  |
| --- |
| # Initial descriptive analysis of data.  credit.describe().T |



**MISSING VALUE ANALYSIS**

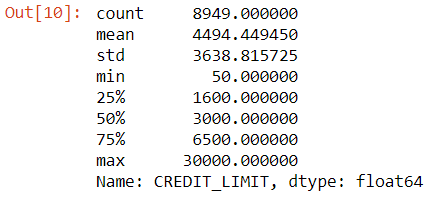
|  |
| --- |
| # finding missing values  credit.isnull().sum() |

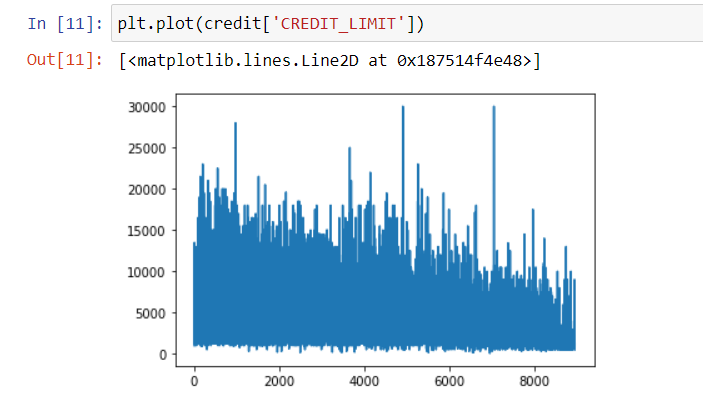


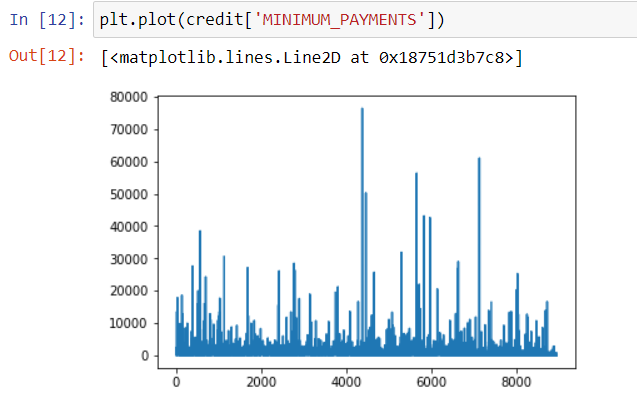
**Observation**

* there are missing values in the data so we will have to treat them accordingly

|  |
| --- |
| credit['CREDIT\_LIMIT'].describe() |



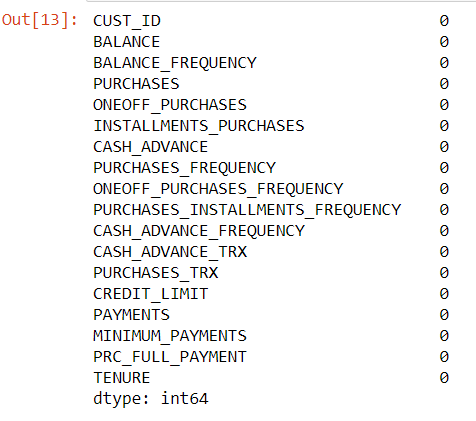




**Observation**

* From the graph we can see that there are some outlier data in the distribution of columns "CREDIT\_LIMIT" and "MINIMUM\_PAYMENTS" and also, we don't want any data to be lost in this dataset and hence we will fill the null values with median imputation rather than mean imputation.
* This is because mean can't give the measure of central tendency if there is any outlier data available in the data distribution.

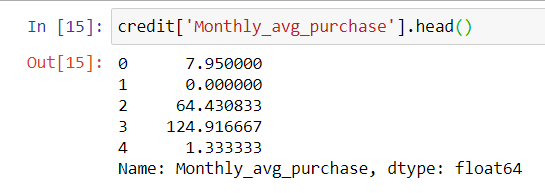
|  |
| --- |
| # imputing missing values with median  credit['CREDIT\_LIMIT'].fillna(credit['CREDIT\_LIMIT'].median(),inplace=True)  credit['MINIMUM\_PAYMENTS'].fillna(credit['MINIMUM\_PAYMENTS'].median(),inplace=True)  credit.isnull().sum() |

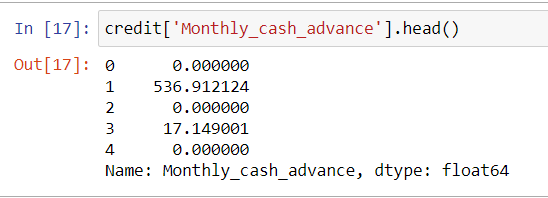


**Deriving Key Performance Indicators (KPI)**

1. **Monthly average purchase and cash advance amount**

|  |
| --- |
| credit['Monthly\_avg\_purchase']=credit['PURCHASES']/credit['TENURE']  credit['Monthly\_cash\_advance']=credit['CASH\_ADVANCE']/credit['TENURE'] |



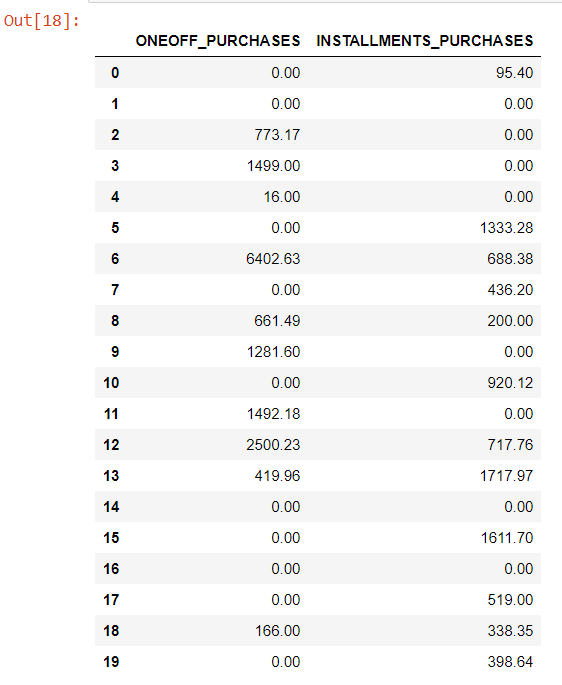


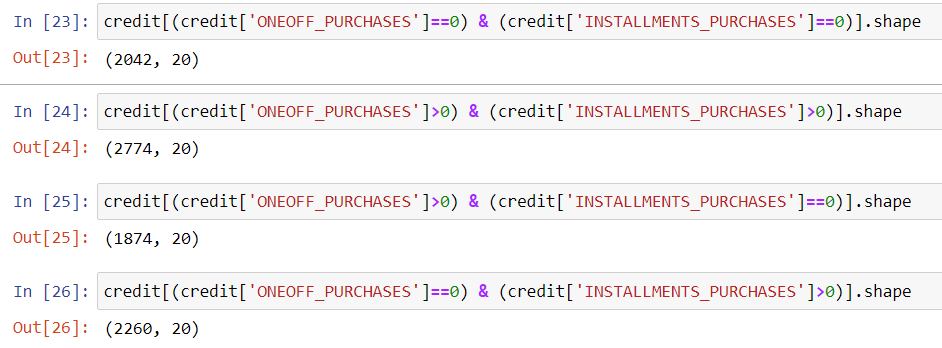
|  |
| --- |
| credit['ONEOFF\_PURCHASES'][credit['ONEOFF\_PURCHASES']==0].count()  Out: 4302 |

1. **Purchase\_type**

* To find what type of purchases customers are making on credit card, let’s explore the data.

|  |
| --- |
| credit.loc[:,['ONEOFF\_PURCHASES','INSTALLMENTS\_PURCHASES']].head(20) |





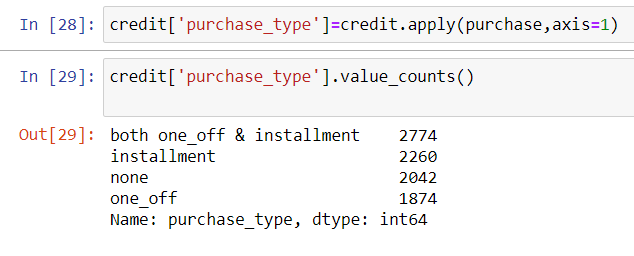
**Observations:**

We can see that there are four types of customers in the entire dataset, they are

* Customers who do only one-off purchase transactions
* Customers who do only installment purchase transaction
* Customers who do both one-off purchase and installment purchase transactions
* Customers who neither do one-off purchase transactions nor installment purchase transactions.

So, deriving a categorical variable based on the behavior.

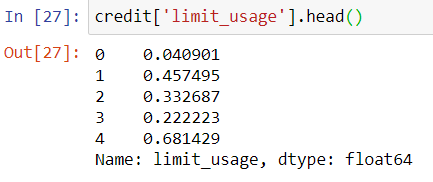
|  |
| --- |
| def purchase(credit):  if (credit['ONEOFF\_PURCHASES']==0) & (credit['INSTALLMENTS\_PURCHASES']==0):  return 'none'  if (credit['ONEOFF\_PURCHASES']>0) & (credit['INSTALLMENTS\_PURCHASES']>0):  return 'both one\_off & installment'  if (credit['ONEOFF\_PURCHASES']>0) & (credit['INSTALLMENTS\_PURCHASES']==0):  return 'one\_off'  if (credit['ONEOFF\_PURCHASES']==0) & (credit['INSTALLMENTS\_PURCHASES']>0):  return 'installment' |



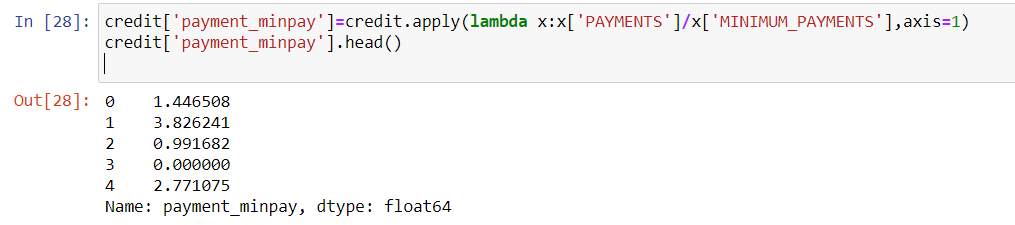
**3. Limit\_Usage (balance to credit limit ratio)**

* Lower value implies customers are maintaining their balance properly. Lower value means good credit score

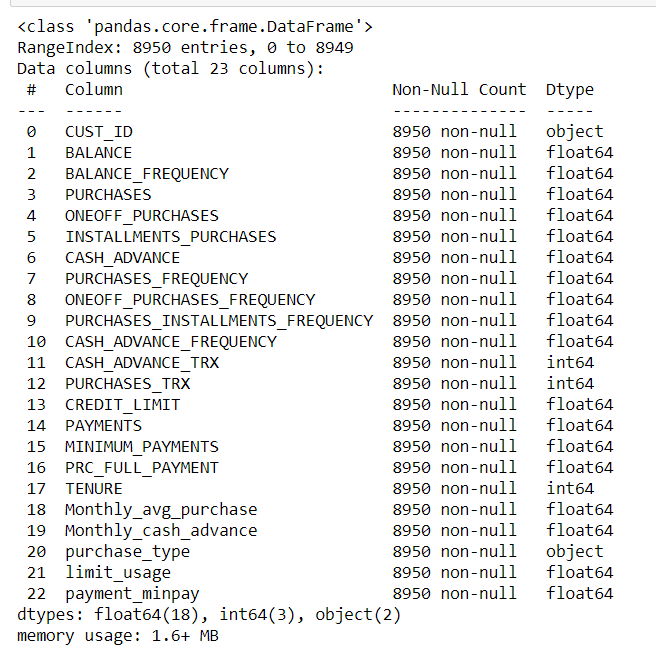
|  |
| --- |
| credit['limit\_usage']=credit.apply(lambda x: x['BALANCE']/x['CREDIT\_LIMIT'], axis=1)  credit['limit\_usage'].head() |



1. **Payment to minimum payments Ratio**

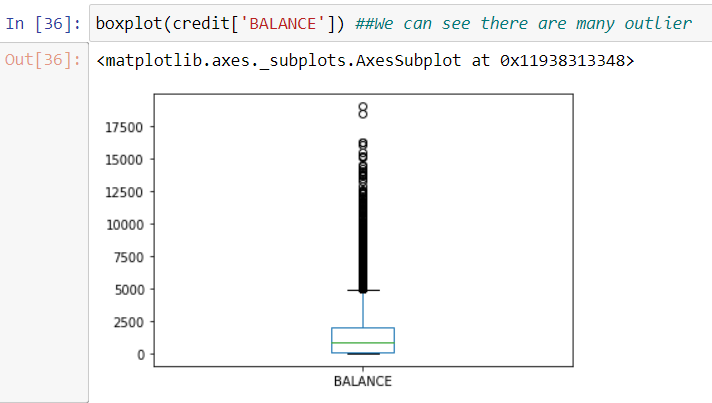


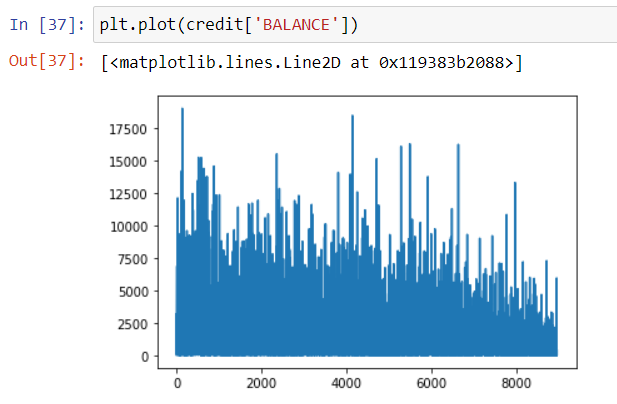
|  |
| --- |
| credit.info() |

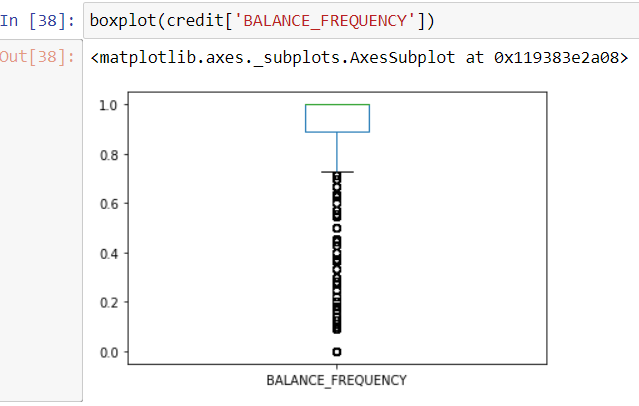


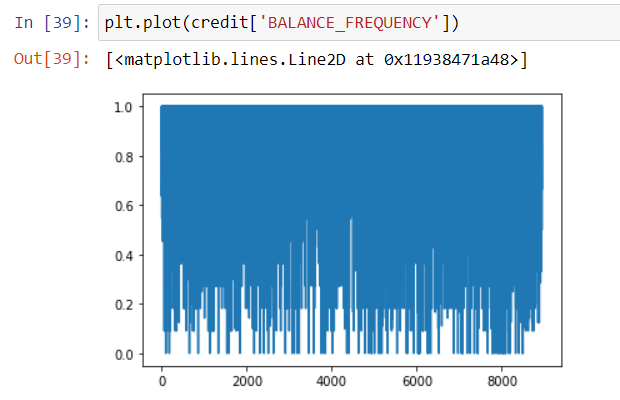
**OUTLIER ANALYSIS**

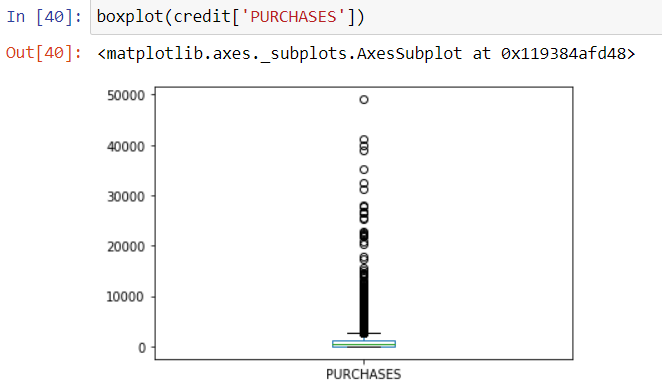
|  |
| --- |
| ##make the function to check the outlier  def boxplot(value):  return value.plot.box() |

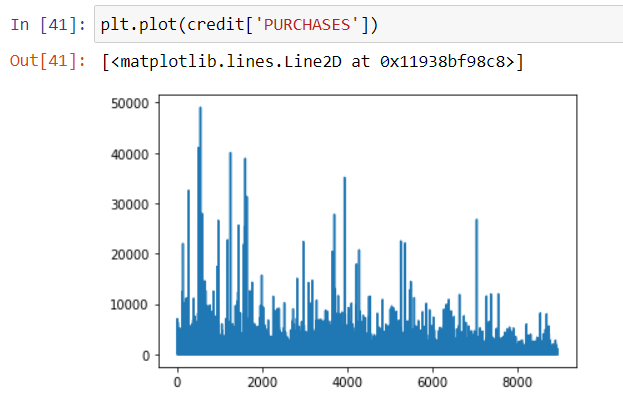








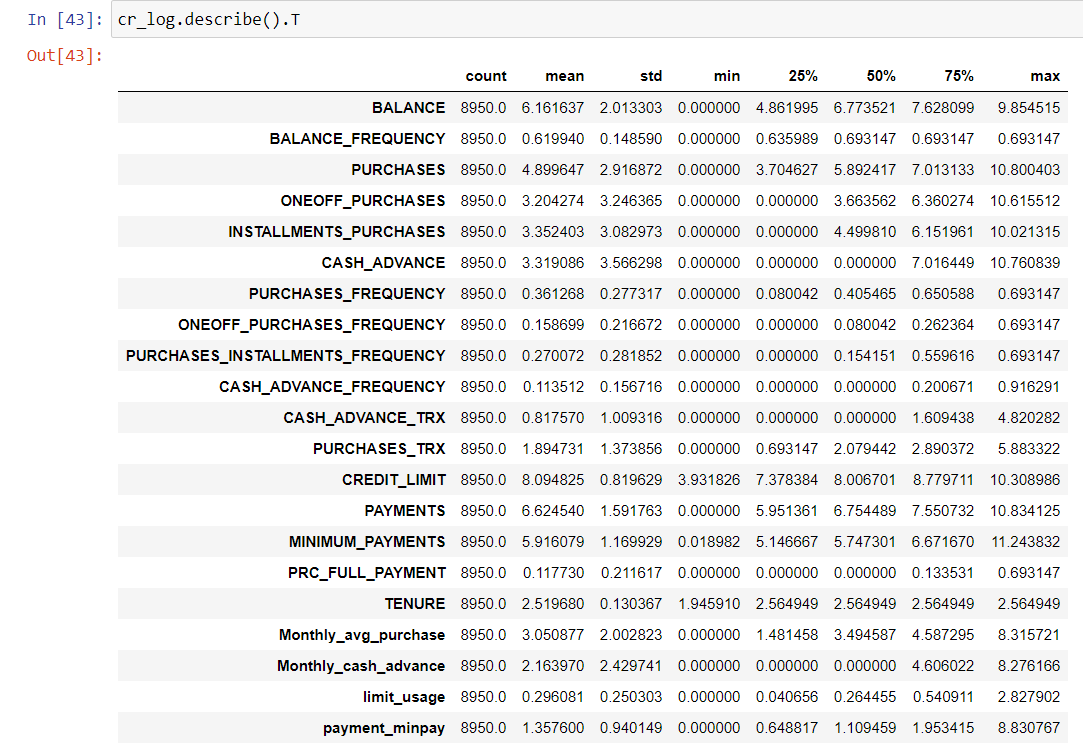




**Observations:**

* From the above description of some variables, we can see that there is high variance among the values and this leads to the skewness in the data.
* Hence to avoid this we will be applying log transformation on all the variables present in the dataset, this solves the problem of skewness.

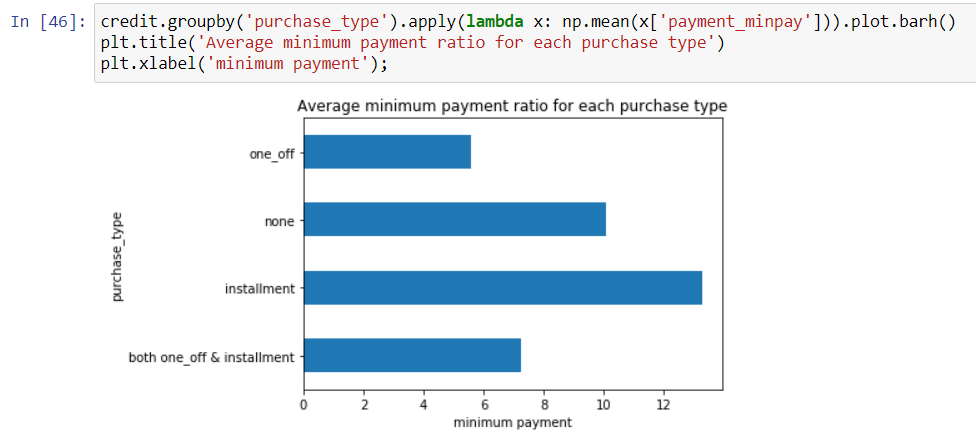
|  |
| --- |
| # log transformation  cr\_log=credit.drop(['CUST\_ID','purchase\_type'],axis=1).applymap(lambda x: np.log(x+1)) |



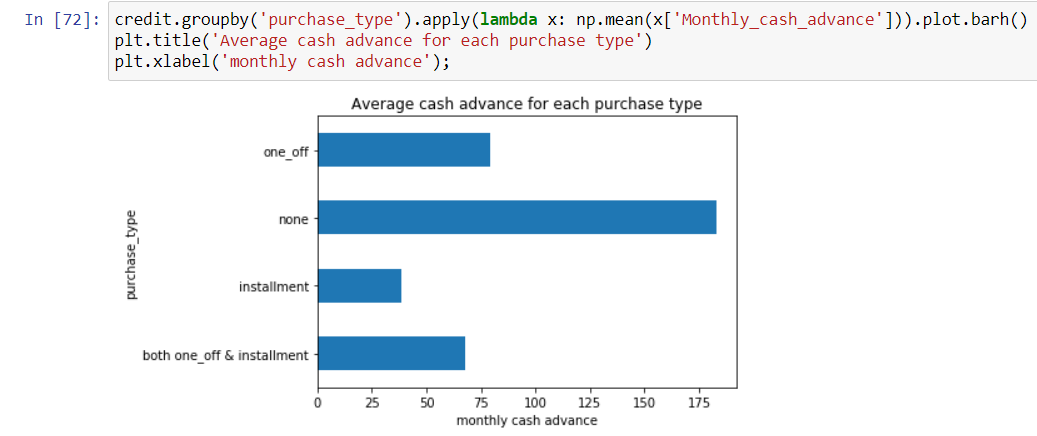
|  |
| --- |
| col=['BALANCE','PURCHASES','CASH\_ADVANCE','TENURE','PAYMENTS','MINIMUM\_PAYMENTS','PRC\_FULL\_PAYMENT','CREDIT\_LIMIT']  cr\_pre=cr\_log[[x for x in cr\_log.columns if x not in col ]] |

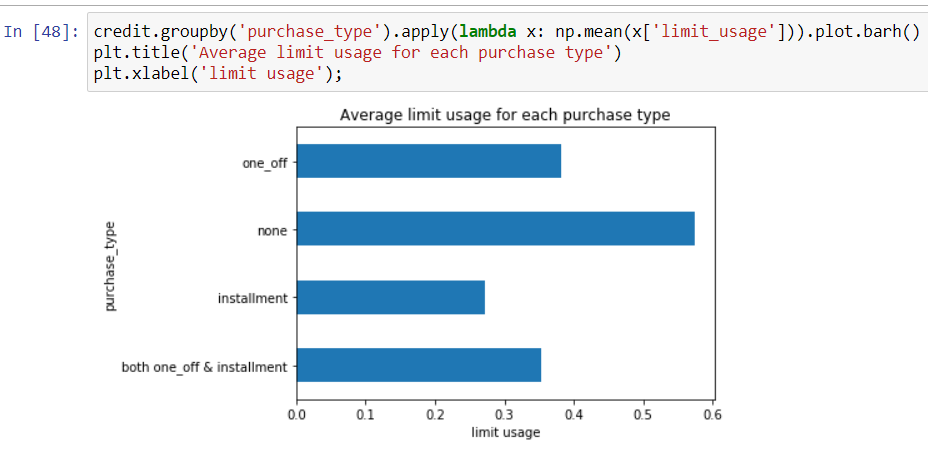
**Finding insights from the data**

|  |
| --- |
| # Average payment\_minpayment ratio for each purchase type.  x=credit.groupby('purchase\_type').apply(lambda x: np.mean(x['payment\_minpay']))  type(x)  x.values  Out : array([ 7.23698216, 13.2590037 , 10.08745106, 5.57108156]) |



**Insight 1:** Customers who make transactions in installments are paying the amount regularly

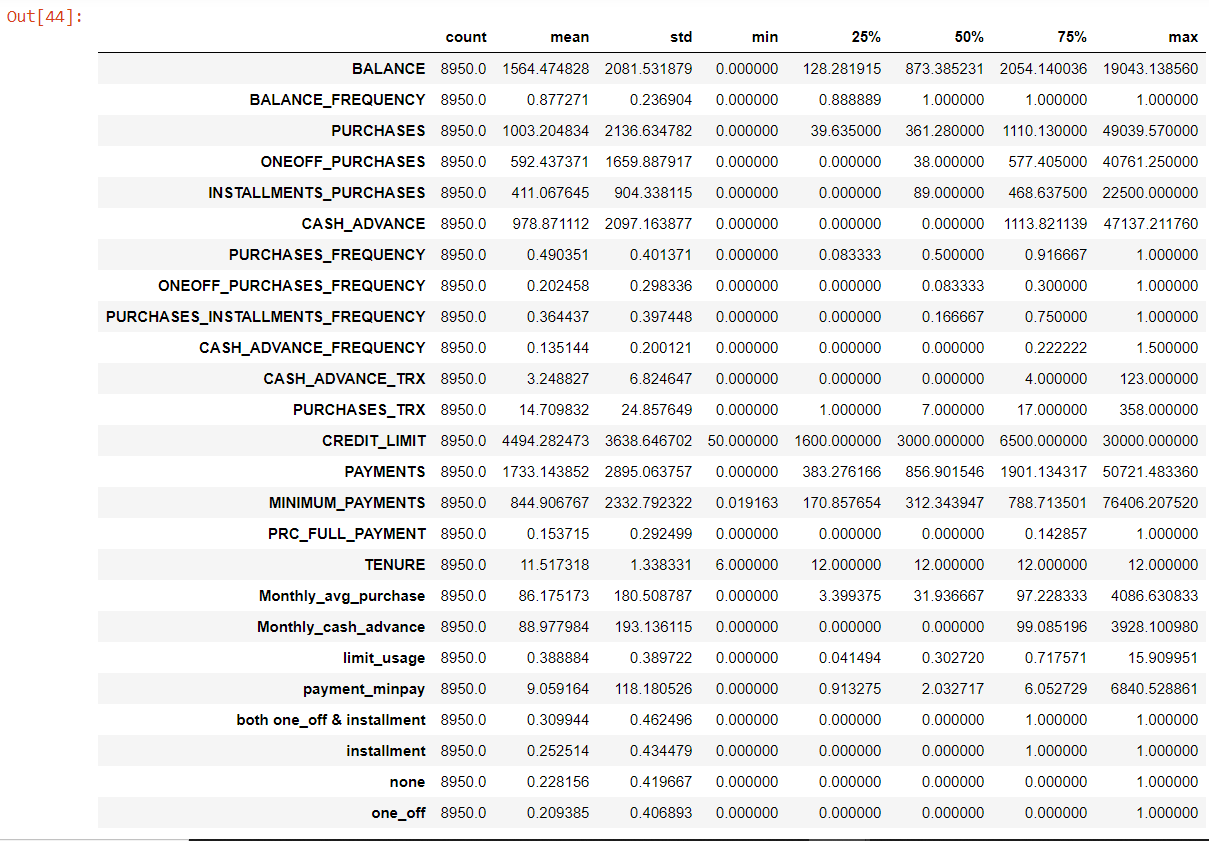


**Insight 2:** Customers who neither make a transaction in one-off payments nor installments are having high monthly cash advances

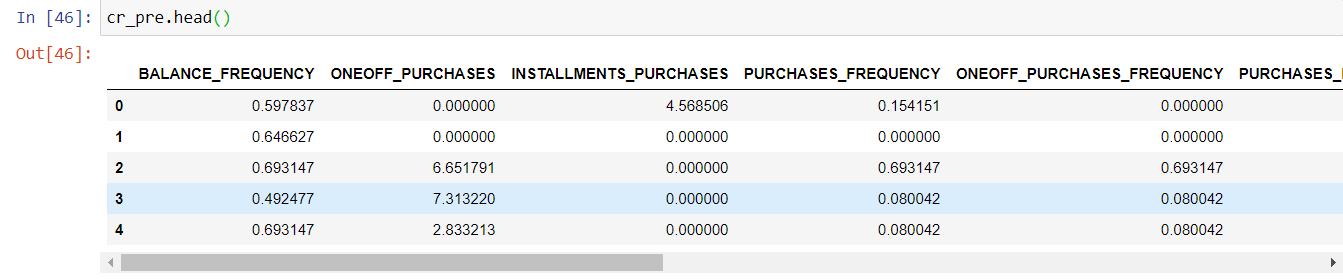
**Insight 3:** Less limit usage gives high credit score and the good score is with the customers who make transactions in installments

**Dataset Preparations for model selection**

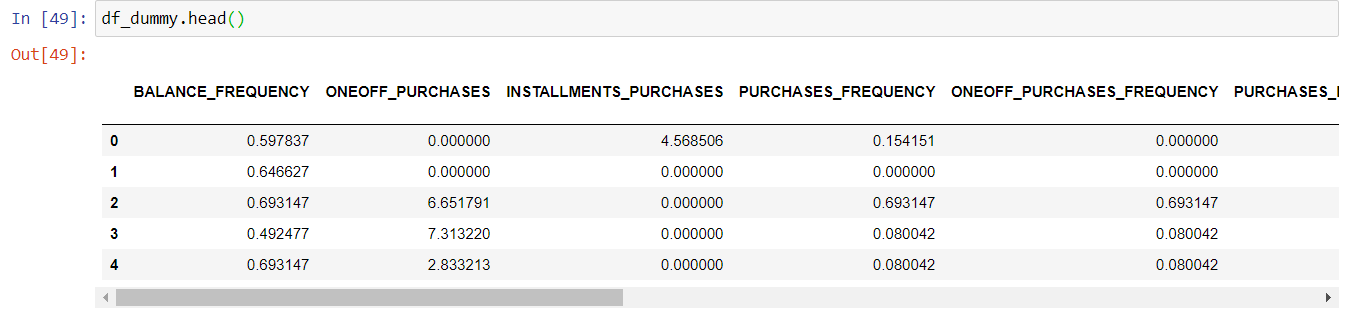
|  |
| --- |
| # Original dataset with categorical column converted to number type.  cre\_original=pd.concat([credit,pd.get\_dummies(credit['purchase\_type'])],axis=1)  cre\_original.describe().T |

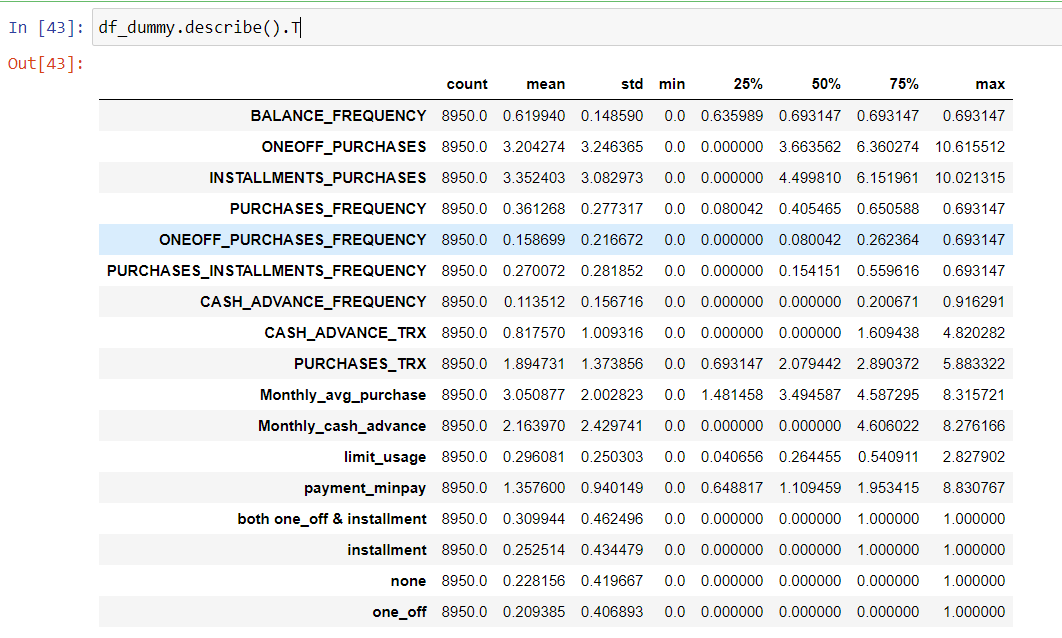


|  |
| --- |
| cr\_pre['purchase\_type']=credit.loc[:,'purchase\_type']  cr\_pre.head() |



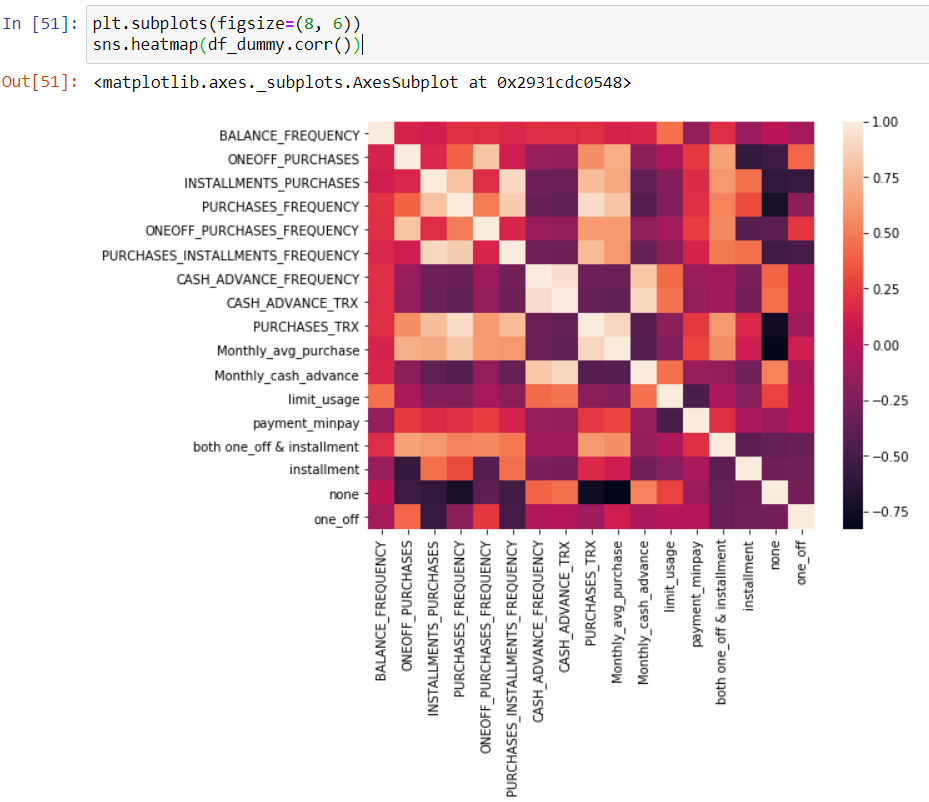
|  |
| --- |
| df\_dummy=pd.concat([cr\_pre,pd.get\_dummies(cr\_pre['purchase\_type'])],axis=1)  df\_dummy=df\_dummy.drop(['purchase\_type'],axis=1)  df\_dummy.head() |





* Finding the correlation among the variables in dataset

|  |
| --- |
| plt.subplots(figsize=(8, 6))  sns.heatmap(df\_dummy.corr()) |



**Observation**

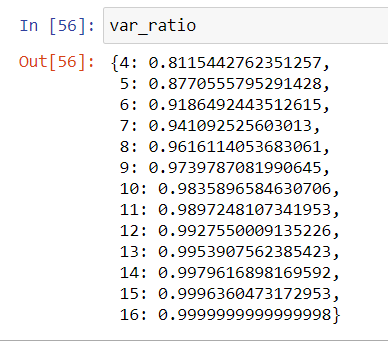
* The variables available for the model selection are very high in this dataset and this leads to dimensionality curse. In order to reduce the high dimensionality, curse we will use Principal Component Analysis technique.
* Before applying PCA we will standardize data to avoid effect of scale on our result. Centering and Scaling will make all features with equal weight. So we use standard scaler technique if there are any weightage issues among the variables of the dataset.

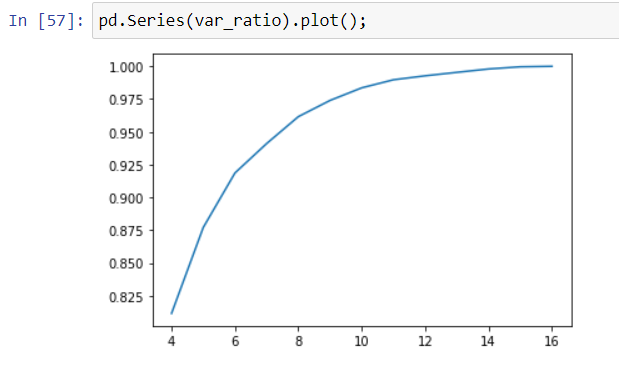
**Standardizing data**

* To put data on the same scale

|  |
| --- |
| from sklearn.preprocessing import StandardScaler  sc=StandardScaler()  df\_scaled=sc.fit\_transform(df\_dummy) |

|  |
| --- |
| from sklearn.decomposition import PCA  var\_ratio={}  for n in range(4,17):  pc=PCA(n\_components=n,svd\_solver='full')  df\_pca=pc.fit(df\_scaled)  var\_ratio[n]=sum(df\_pca.explained\_variance\_ratio\_)  var\_ratio |



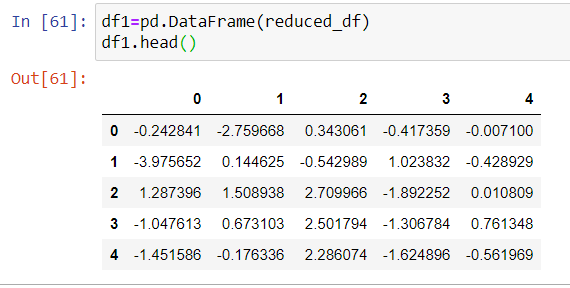


**Observation**

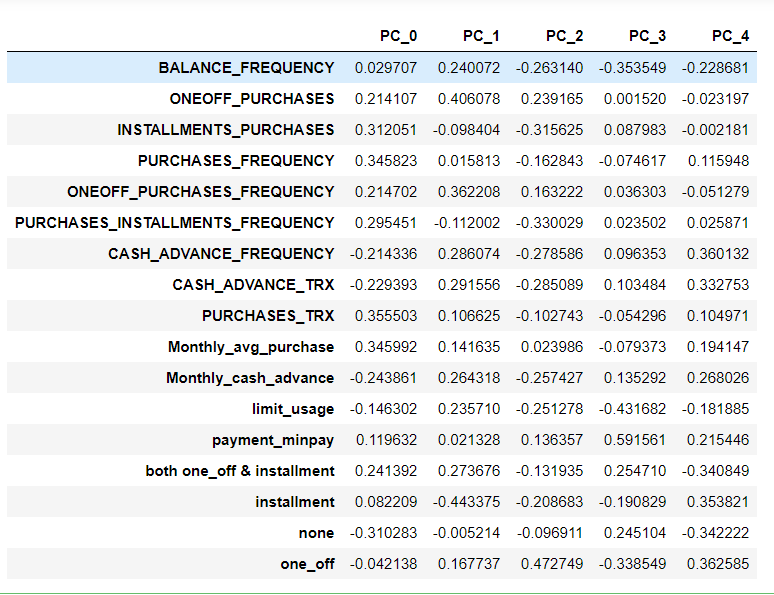
* From the above variance ratio, we can see that the maximum variance of about 87% is explained when the number of components is 5. Hence, we choose n\_components as 5 to reduce the dimensionality in the dataset.

|  |
| --- |
| df\_scaled.shape  Out : (8950,17) |

|  |
| --- |
| pc\_final=PCA(n\_components=5,svd\_solver='full').fit(df\_scaled)  reduced\_df=pc\_final.fit\_transform(df\_scaled)  df1=pd.DataFrame(reduced\_df)  df1.head() |



|  |
| --- |
| df1.shape  Out:(8950, 5)  col\_list=df\_dummy.columns  col\_list  Out: Index(['BALANCE\_FREQUENCY', 'ONEOFF\_PURCHASES', 'INSTALLMENTS\_PURCHASES',  'PURCHASES\_FREQUENCY', 'ONEOFF\_PURCHASES\_FREQUENCY',  'PURCHASES\_INSTALLMENTS\_FREQUENCY', 'CASH\_ADVANCE\_FREQUENCY',  'CASH\_ADVANCE\_TRX', 'PURCHASES\_TRX', 'Monthly\_avg\_purchase',  'Monthly\_cash\_advance', 'limit\_usage', 'payment\_minpay',  'both one\_off & installment', 'installment', 'none', 'one\_off'],  dtype='object')  pd.DataFrame(pc\_final.components\_.T, columns=['PC\_' +str(i) for i in range(5)],index=col\_list) |



|  |
| --- |
| # Factor Analysis: variance explained by each component-  pd.Series(pc\_final.explained\_variance\_ratio\_,index=['PC\_'+ str(i) for i in range(5)])  Out: PC\_0 0.402058  PC\_1 0.180586  PC\_2 0.147294  PC\_3 0.081606  PC\_4 0.065511  dtype: float64 |

**Model Selection**

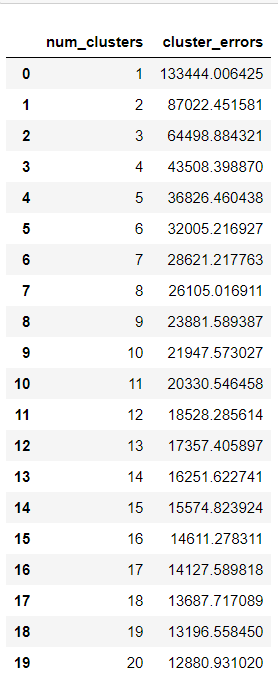
**Clustering**

Based on our 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.**

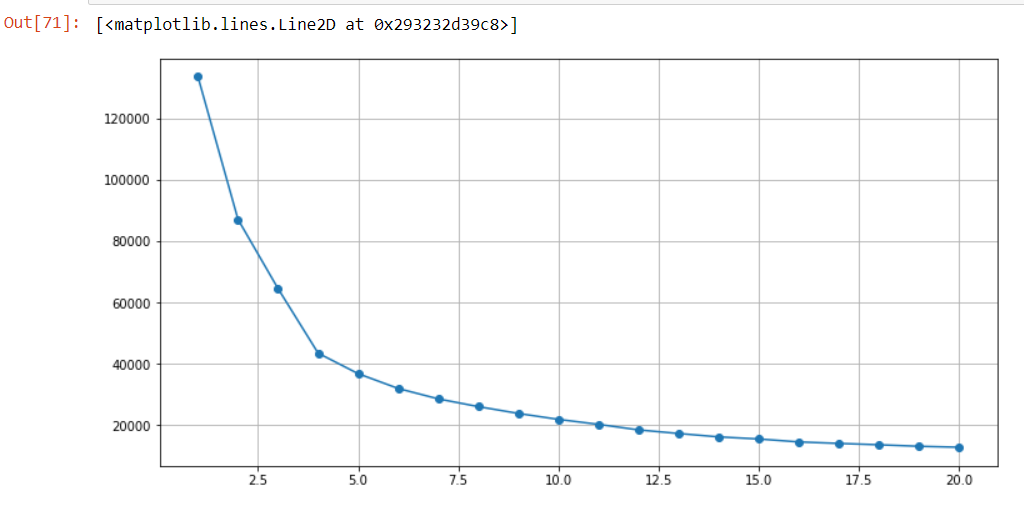
|  |
| --- |
| from sklearn.cluster import KMeans  km\_4=KMeans(n\_clusters=4,random\_state=42)  km\_4.fit(reduced\_df)  km\_4.labels\_  Out: array([0, 1, 3, ..., 0, 1, 3])  pd.Series(km\_4.labels\_).value\_counts()  Out:  2 2758  0 2228  1 2090  3 1874  dtype: int64 |

* Here we do not have known k value so we will find the K. To do that we need to take a cluster range between 1 and 21.

|  |
| --- |
| # Identify cluster errors  cluster\_range = range( 1, 21 )  cluster\_errors = []  for num\_clusters in cluster\_range:  clusters = KMeans( num\_clusters )  clusters.fit( reduced\_df )  cluster\_errors.append( clusters.inertia\_ ) # clusters.inertia\_ is basically cluster error here  clusters\_df = pd.DataFrame( { "num\_clusters":cluster\_range, "cluster\_errors": cluster\_errors } )  clusters\_df[0:21] |

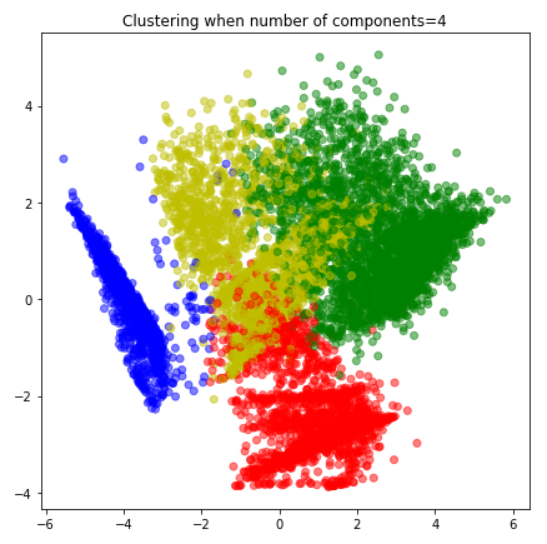


|  |
| --- |
| # checking k value from elbow plot  import matplotlib.pyplot as plt  plt.figure(figsize=(12,6))  plt.grid()  plt.plot( clusters\_df.num\_clusters, clusters\_df.cluster\_errors, marker = "o" ) |

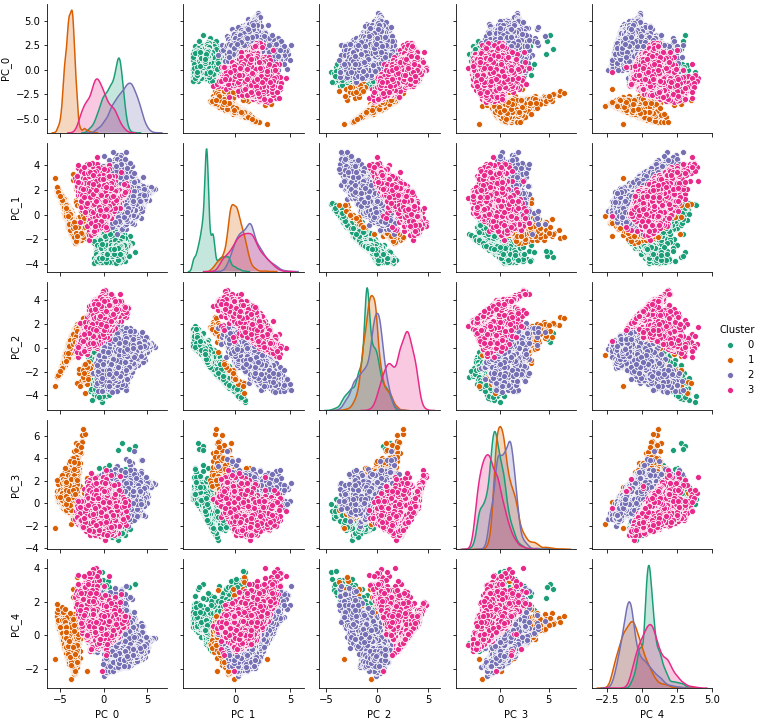


* From above graph, it is can take we can take k as 4,5 or 6

|  |
| --- |
| 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\_df[:,0],reduced\_df[:,1],c=label\_color,cmap='Spectral',alpha=0.5)  plt.title('Clustering when number of components=4'); |



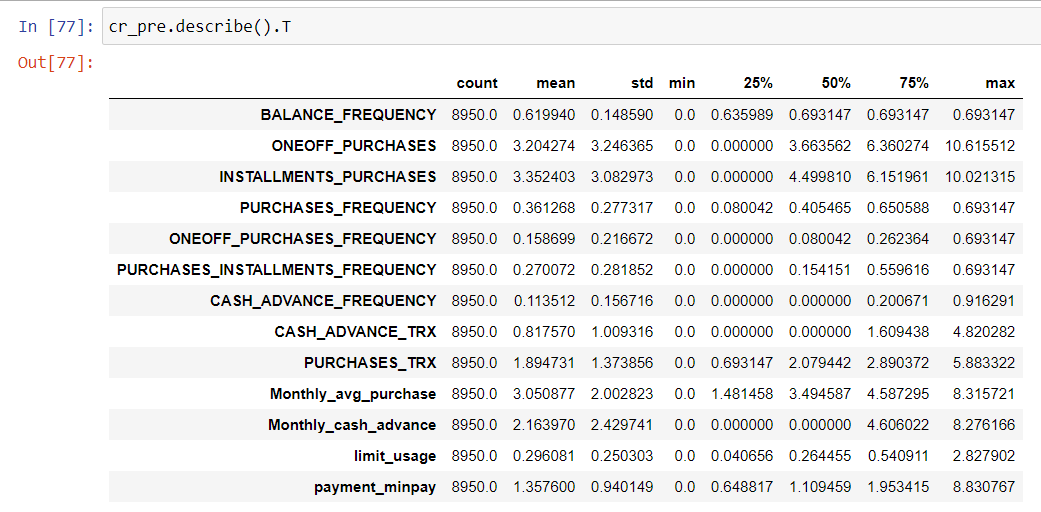
|  |
| --- |
| df\_pair\_plot=pd.DataFrame(reduced\_df,columns=['PC\_' +str(i) for i in range(5)])  df\_pair\_plot['Cluster']=km\_4.labels\_  #pairwise relationship of components on the data  sns.pairplot(df\_pair\_plot,hue='Cluster', palette= 'Dark2', diag\_kind='kde',height=2)  plt.savefig("pairplot") |



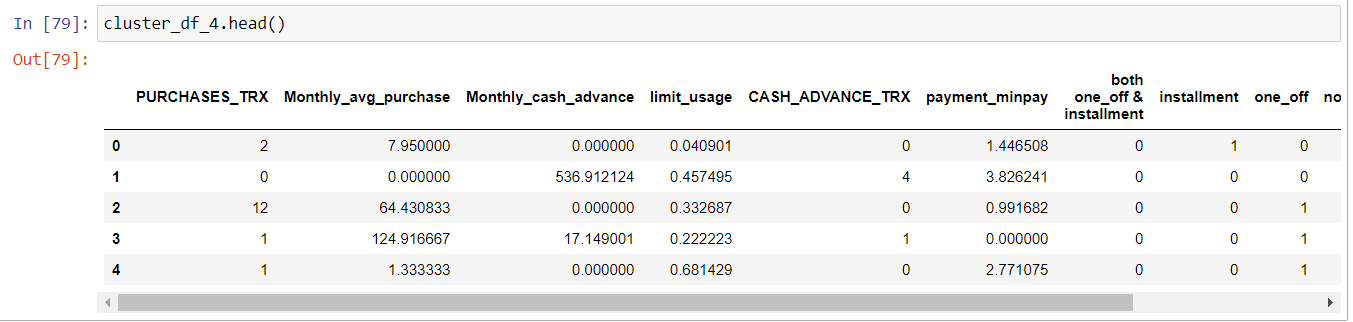
**Observations:**

* From the above graphs we can conclude that the only PC\_0 and PC\_1 are identifiable clusters and hence we go with further analysis by increasing the number of clusters value to identify more number of insights about the customers present in the dataset.

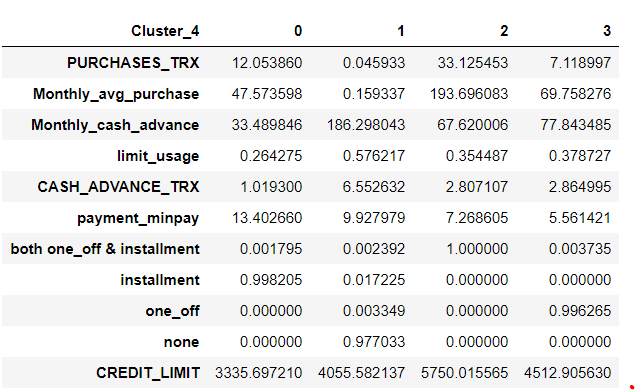
|  |
| --- |
| # Key performance variable selection . here I am dropping variables which are used in deriving new KPI  col\_kpi=['PURCHASES\_TRX','Monthly\_avg\_purchase','Monthly\_cash\_advance','limit\_usage','CASH\_ADVANCE\_TRX',  'payment\_minpay','both one\_off & installment','installment','one\_off','none','CREDIT\_LIMIT']  cr\_pre.describe().T |



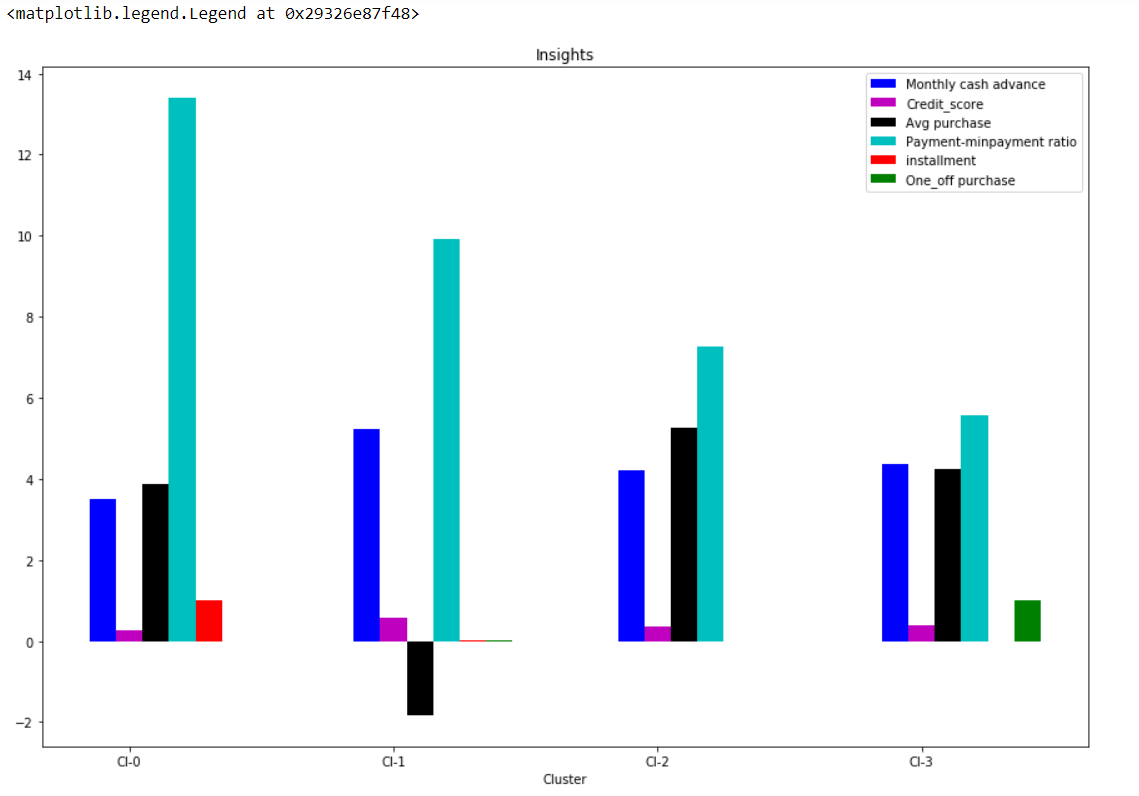
|  |
| --- |
| # Concatenating labels found through Kmeans with data  cluster\_df\_4=pd.concat([cre\_original[col\_kpi],pd.Series(km\_4.labels\_,name='Cluster\_4')],axis=1) |



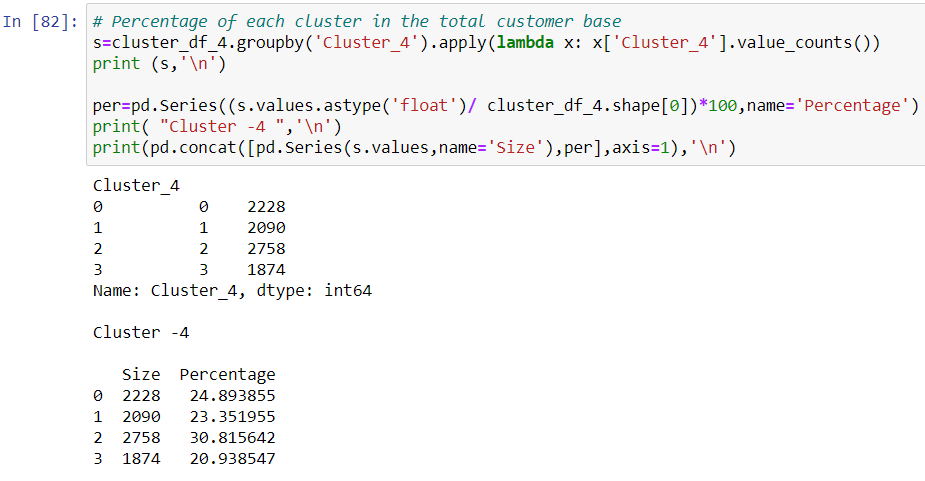
|  |
| --- |
| # Mean value gives a good indication of the distribution of data. So we are finding mean value for each variable for each cluster  cluster\_4=cluster\_df\_4.groupby('Cluster\_4')\  .apply(lambda x: x[col\_kpi].mean()).T  cluster\_4 |



|  |
| --- |
| fig,ax=plt.subplots(figsize=(15,10))  index=np.arange(len(cluster\_4.columns))  cash\_advance=np.log(cluster\_4.loc['Monthly\_cash\_advance',:].values)  credit\_score=(cluster\_4.loc['limit\_usage',:].values)  purchase= np.log(cluster\_4.loc['Monthly\_avg\_purchase',:].values)  payment=cluster\_4.loc['payment\_minpay',:].values  installment=cluster\_4.loc['installment',:].values  one\_off=cluster\_4.loc['one\_off',:].values  bar\_width=.10  b1=plt.bar(index,cash\_advance,color='b',label='Monthly cash advance',width=bar\_width)  b2=plt.bar(index+bar\_width,credit\_score,color='m',label='Credit\_score',width=bar\_width)  b3=plt.bar(index+2\*bar\_width,purchase,color='k',label='Avg purchase',width=bar\_width)  b4=plt.bar(index+3\*bar\_width,payment,color='c',label='Payment-minpayment ratio',width=bar\_width)  b5=plt.bar(index+4\*bar\_width,installment,color='r',label='installment',width=bar\_width)  b6=plt.bar(index+5\*bar\_width,one\_off,color='g',label='One\_off purchase',width=bar\_width)  plt.xlabel("Cluster")  plt.title("Insights")  plt.xticks(index + bar\_width, ('Cl-0', 'Cl-1', 'Cl-2', 'Cl-3'))  plt.legend() |

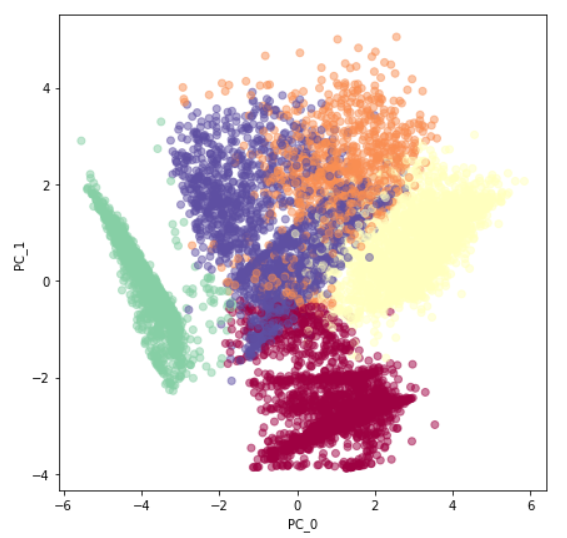
   
**Observation**

* From the above graph we can see that the four clusters have been categorised perfectly so that the difference in each cluster can be understood

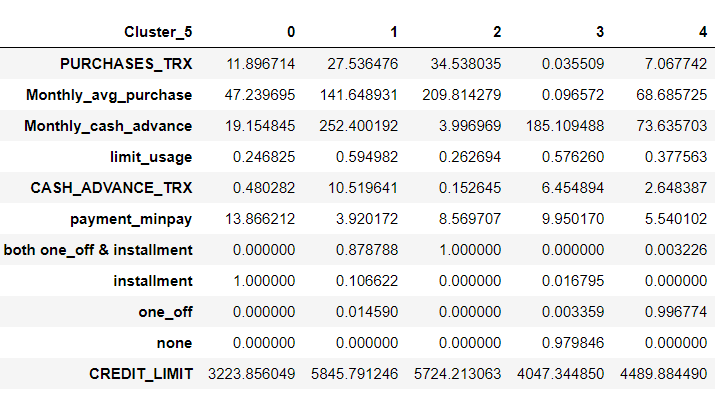


**Exploring the insights if the number of clusters=5**

|  |
| --- |
| #kmeans with 5 clusters  km\_5=KMeans(n\_clusters=5,random\_state=42)  km\_5=km\_5.fit(reduced\_df)  km\_5.labels\_  Out:  array([0, 3, 4, ..., 0, 3, 4])  pd.Series(km\_5.labels\_).value\_counts()  Out:  0 2130  3 2084  2 1985  4 1860  1 891  dtype: int64  plt.figure(figsize=(7,7))  plt.scatter(reduced\_df[:,0],reduced\_df[:,1],c=km\_5.labels\_,cmap='Spectral',alpha=0.5)  plt.xlabel('PC\_0')  plt.ylabel('PC\_1') |

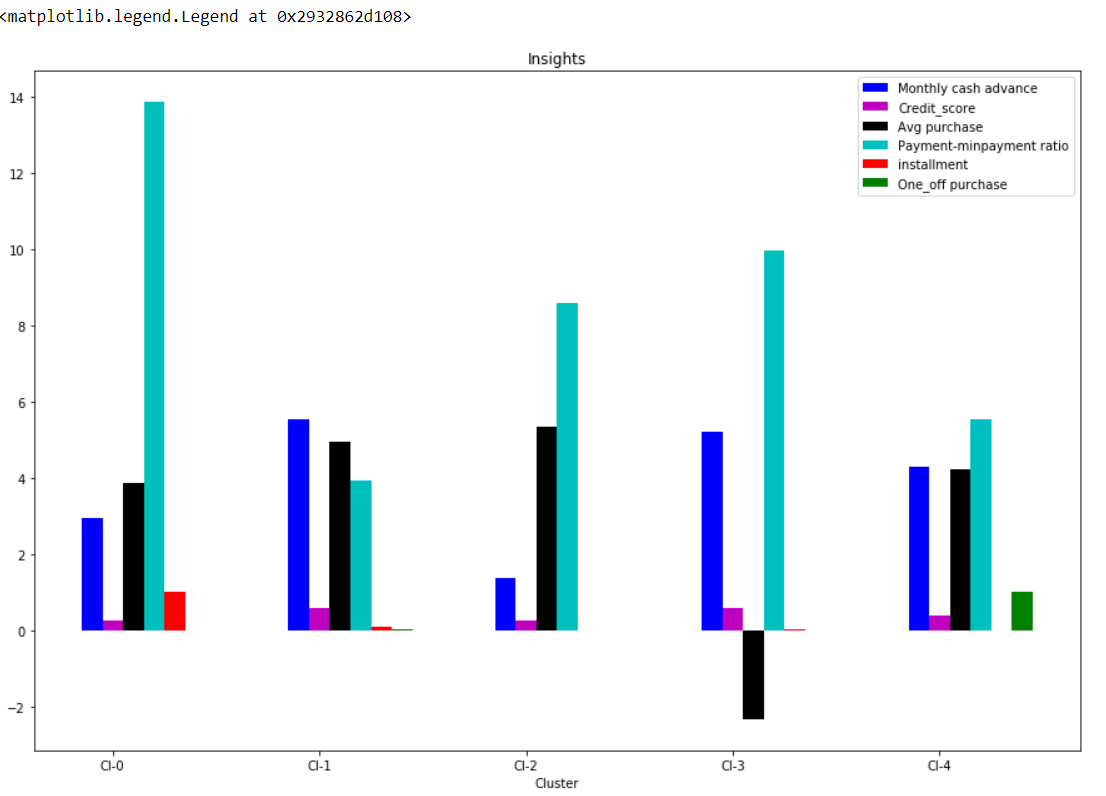


|  |
| --- |
| cluster\_df\_5=pd.concat([cre\_original[col\_kpi],pd.Series(km\_5.labels\_,name='Cluster\_5')],axis=1)  # Finding Mean of features for each cluster  five\_cluster=cluster\_df\_5.groupby('Cluster\_5')\  .apply(lambda x: x[col\_kpi].mean()).T  five\_cluster |



|  |
| --- |
| s1=cluster\_df\_5.groupby('Cluster\_5').apply(lambda x: x['Cluster\_5'].value\_counts())  print(s1)  Out:  Cluster\_5  0 0 2130  1 1 891  2 2 1985  3 3 2084  4 4 1860  Name: Cluster\_5, dtype: int64 |

|  |
| --- |
| fig,ax=plt.subplots(figsize=(15,10))  index=np.arange(len(five\_cluster.columns))  cash\_advance=np.log(five\_cluster.loc['Monthly\_cash\_advance',:].values)  credit\_score=(five\_cluster.loc['limit\_usage',:].values)  purchase= np.log(five\_cluster.loc['Monthly\_avg\_purchase',:].values)  payment=five\_cluster.loc['payment\_minpay',:].values  installment=five\_cluster.loc['installment',:].values  one\_off=five\_cluster.loc['one\_off',:].values  bar\_width=.10  b1=plt.bar(index,cash\_advance,color='b',label='Monthly cash advance',width=bar\_width)  b2=plt.bar(index+bar\_width,credit\_score,color='m',label='Credit\_score',width=bar\_width)  b3=plt.bar(index+2\*bar\_width,purchase,color='k',label='Avg purchase',width=bar\_width)  b4=plt.bar(index+3\*bar\_width,payment,color='c',label='Payment-minpayment ratio',width=bar\_width)  b5=plt.bar(index+4\*bar\_width,installment,color='r',label='installment',width=bar\_width)  b6=plt.bar(index+5\*bar\_width,one\_off,color='g',label='One\_off purchase',width=bar\_width)  plt.xlabel("Cluster")  plt.title("Insights")  plt.xticks(index + bar\_width, ('Cl-0', 'Cl-1', 'Cl-2', 'Cl-3','Cl-4'))  plt.legend() |



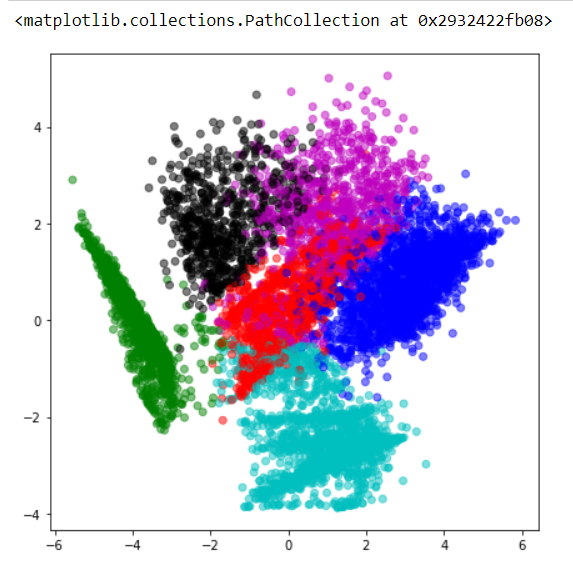
**Observation**

* From the above graph, we can't come to a particular conclusion regarding the behavior of customer groups, because cluster 2 is having highest average purchases in the transactions, but at the same time cluster1 has highest cash advance and second highest purchases.

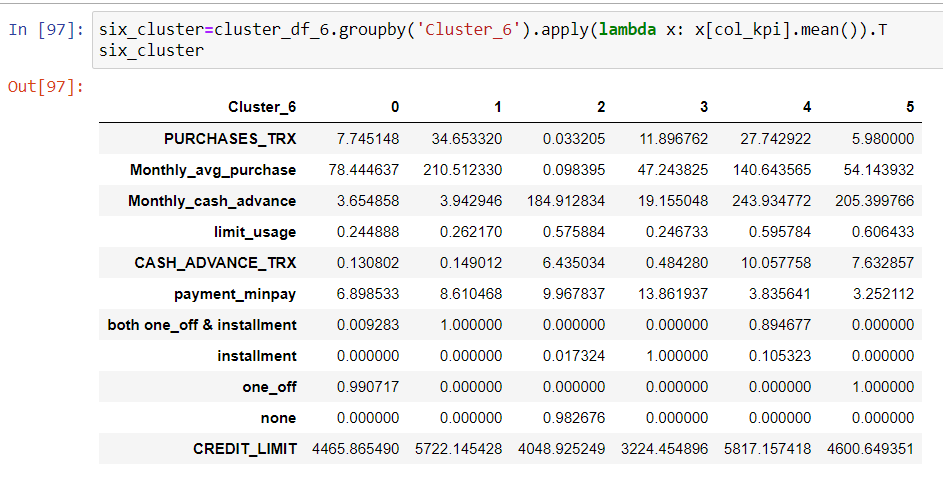
|  |
| --- |
| # percentage of each cluster  print("Cluster-5")  per\_5=pd.Series((s1.values.astype('float')/ cluster\_df\_5.shape[0])\*100,name='Percentage')  print(pd.concat([pd.Series(s1.values,name='Size'),per\_5],axis=1)) |

**Exploring the insights if the number of cluster=6**

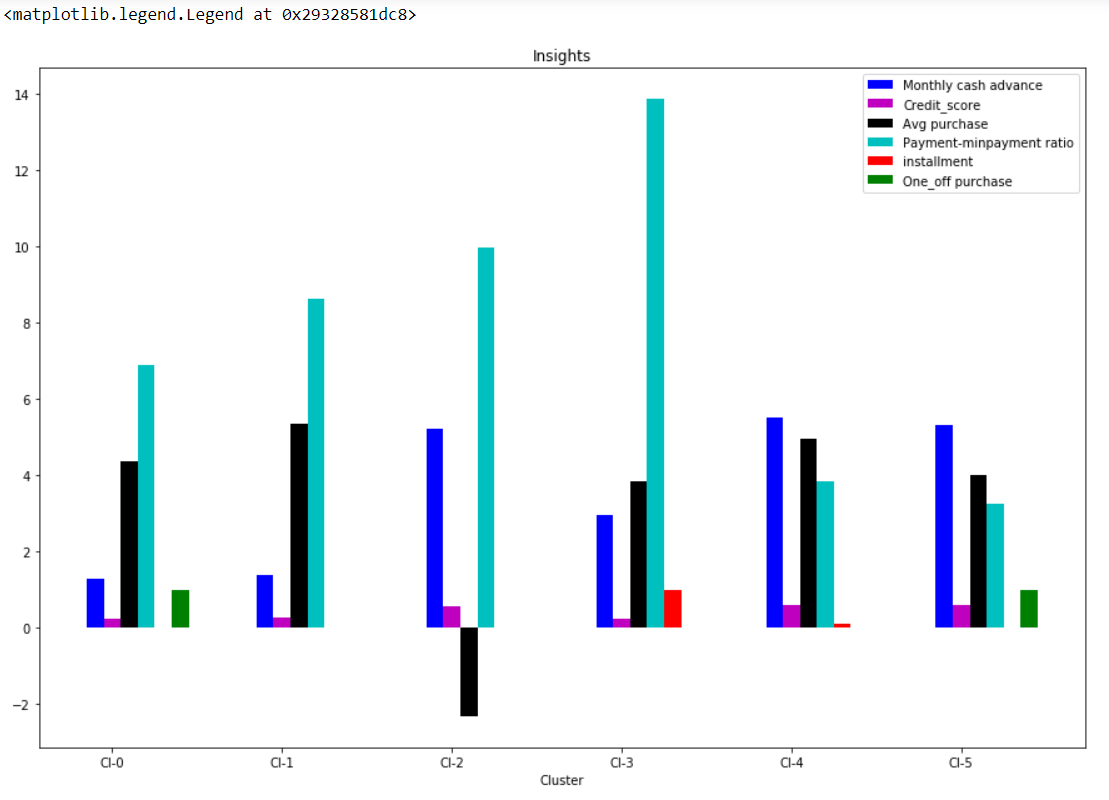
|  |
| --- |
| km\_6=KMeans(n\_clusters=6).fit(reduced\_df)  km\_6.labels\_  Out:  array([3, 2, 0, ..., 3, 2, 5])  color\_map={0:'r',1:'b',2:'g',3:'c',4:'m',5:'k'}  label\_color=[color\_map[l] for l in km\_6.labels\_]  plt.figure(figsize=(7,7))  plt.scatter(reduced\_df[:,0],reduced\_df[:,1],c=label\_color,cmap='Spectral',alpha=0.5) |



|  |
| --- |
| cluster\_df\_6=pd.concat([cre\_original[col\_kpi],pd.Series(km\_6.labels\_,name='Cluster\_6')],axis=1)  six\_cluster=cluster\_df\_6.groupby('Cluster\_6').apply(lambda x: x[col\_kpi].mean()).T  six\_cluster |



|  |
| --- |
| fig,ax=plt.subplots(figsize=(15,10))  index=np.arange(len(six\_cluster.columns))  cash\_advance=np.log(six\_cluster.loc['Monthly\_cash\_advance',:].values)  credit\_score=(six\_cluster.loc['limit\_usage',:].values)  purchase= np.log(six\_cluster.loc['Monthly\_avg\_purchase',:].values)  payment=six\_cluster.loc['payment\_minpay',:].values  installment=six\_cluster.loc['installment',:].values  one\_off=six\_cluster.loc['one\_off',:].values  bar\_width=.10  b1=plt.bar(index,cash\_advance,color='b',label='Monthly cash advance',width=bar\_width)  b2=plt.bar(index+bar\_width,credit\_score,color='m',label='Credit\_score',width=bar\_width)  b3=plt.bar(index+2\*bar\_width,purchase,color='k',label='Avg purchase',width=bar\_width)  b4=plt.bar(index+3\*bar\_width,payment,color='c',label='Payment-minpayment ratio',width=bar\_width)  b5=plt.bar(index+4\*bar\_width,installment,color='r',label='installment',width=bar\_width)  b6=plt.bar(index+5\*bar\_width,one\_off,color='g',label='One\_off purchase',width=bar\_width)  plt.xlabel("Cluster")  plt.title("Insights")  plt.xticks(index + bar\_width, ('Cl-0', 'Cl-1', 'Cl-2', 'Cl-3','Cl-4','Cl-5'))  plt.legend() |



**Observation:**

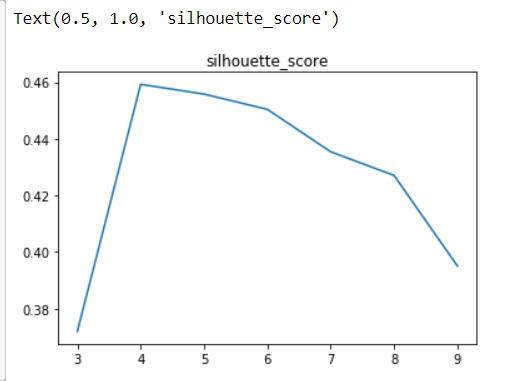
* From the above graph we can see that cluster 2 and cluster 4 have similar behavior regarding the parameters, hence distinguishing between the clusters is hard when we have the number of clusters as 6

|  |
| --- |
| cash\_advance=np.log(six\_cluster.iloc[2,:].values)  credit\_score=list(six\_cluster.iloc[3,:].values)  print(cash\_advance)  print(credit\_score)  Out:  [1.29605733 1.37192804 5.21988454 2.95256629 5.49690086 5.32495816]  [0.24488793326165034, 0.26216962861657617, 0.5758841059122126, 0.24673287577047076, 0.5957844119450174, 0.6064330330654714] |

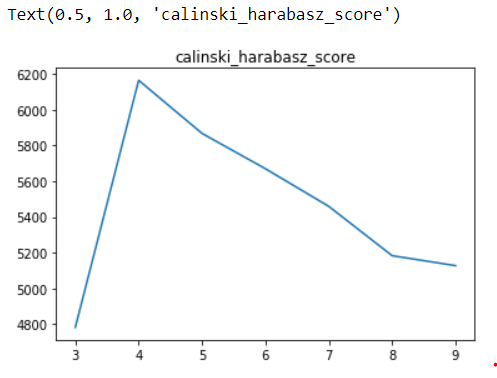
**Checking performance metrics for K means**

* I am validating performance with 2 metrics Calinski harabaz and Silhouette score

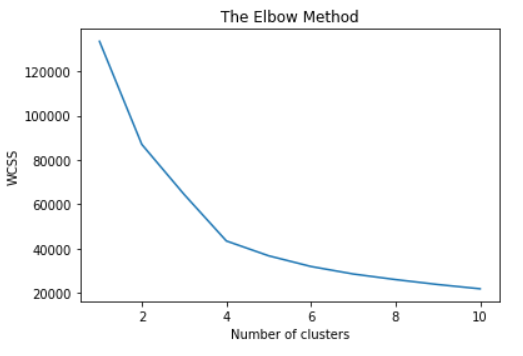
|  |
| --- |
| from sklearn.metrics import calinski\_harabasz\_score,silhouette\_score  score={}  score\_c={}  for n in range(3,10):  km\_score=KMeans(n\_clusters=n)  km\_score.fit(reduced\_df)  score\_c[n]=calinski\_harabasz\_score(reduced\_df,km\_score.labels\_)  score[n]=silhouette\_score(reduced\_df,km\_score.labels\_)  print(score)  Out:  {3: 0.37199332646474775, 4: 0.45925855175999947, 5: 0.4557969467383015, 6: 0.45040500121395316, 7: 0.4354310442029417, 8: 0.42706833598976296, 9: 0.39512255230583815}  pd.Series(score).plot()  plt.title('silhouette\_score') |



|  |
| --- |
| print(score\_c)  Out:  {3: 4781.934521021165, 4: 6164.054484808374, 5: 5867.003840603487, 6: 5669.5040396521545, 7: 5458.826824079906, 8: 5182.940231634099, 9: 5127.351583136153}  pd.Series(score\_c).plot()  plt.title('calinski\_harabasz\_score') |



|  |
| --- |
| from sklearn.cluster import KMeans  wcss = []  for i in range(1, 11):  kmeans = KMeans(n\_clusters = i, init = 'k-means++', random\_state = 42)  kmeans.fit(reduced\_df)  wcss.append(kmeans.inertia\_)  plt.plot(range(1, 11), wcss)  plt.title('The Elbow Method')  plt.xlabel('Number of clusters')  plt.ylabel('WCSS')  plt.show() |

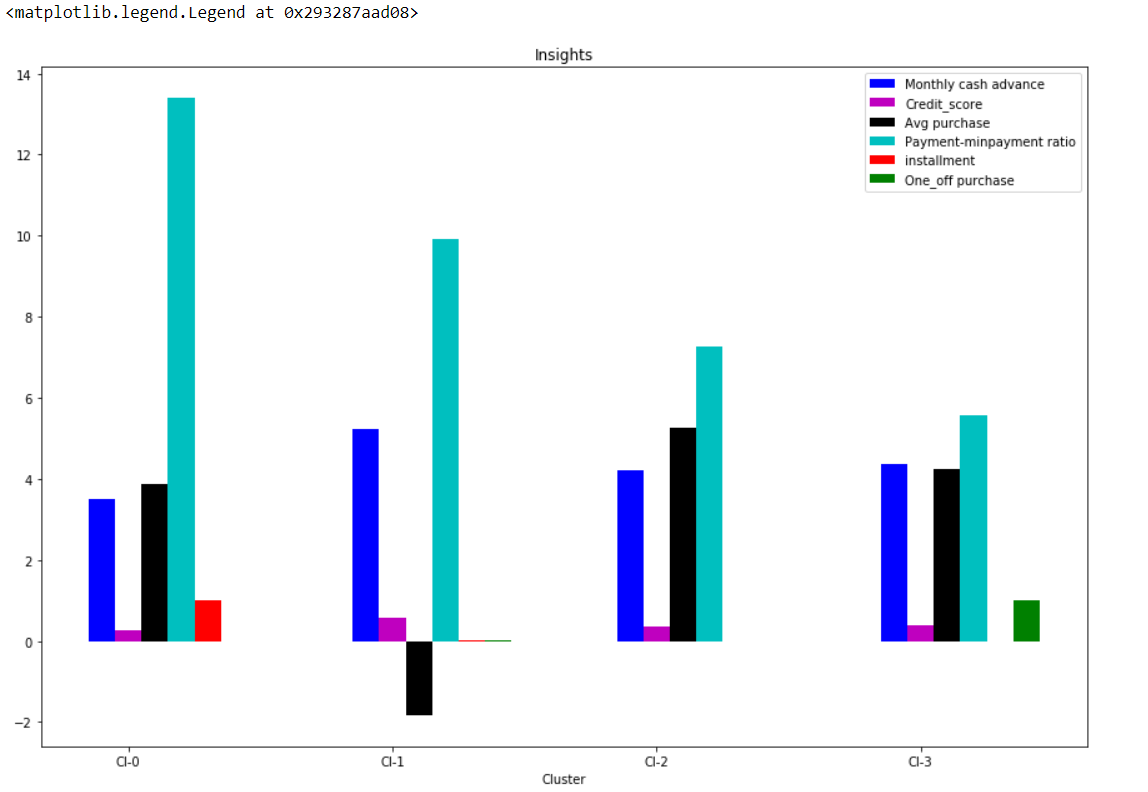


**Observation:**

* From all the above graphs we can conclude the performance of the KMeans Model regarding the explanation of data distribution and measure of spread is highest when we consider the number of clusters as four.

**Final KMeans Model**

|  |
| --- |
| fig,ax=plt.subplots(figsize=(15,10))  index=np.arange(len(cluster\_4.columns))  cash\_advance=np.log(cluster\_4.loc['Monthly\_cash\_advance',:].values)  credit\_score=(cluster\_4.loc['limit\_usage',:].values)  purchase= np.log(cluster\_4.loc['Monthly\_avg\_purchase',:].values)  payment=cluster\_4.loc['payment\_minpay',:].values  installment=cluster\_4.loc['installment',:].values  one\_off=cluster\_4.loc['one\_off',:].values  bar\_width=.10  b1=plt.bar(index,cash\_advance,color='b',label='Monthly cash advance',width=bar\_width)  b2=plt.bar(index+bar\_width,credit\_score,color='m',label='Credit\_score',width=bar\_width)  b3=plt.bar(index+2\*bar\_width,purchase,color='k',label='Avg purchase',width=bar\_width)  b4=plt.bar(index+3\*bar\_width,payment,color='c',label='Payment-minpayment ratio',width=bar\_width)  b5=plt.bar(index+4\*bar\_width,installment,color='r',label='installment',width=bar\_width)  b6=plt.bar(index+5\*bar\_width,one\_off,color='g',label='One\_off purchase',width=bar\_width)  plt.xlabel("Cluster")  plt.title("Insights")  plt.xticks(index + bar\_width, ('Cl-0', 'Cl-1', 'Cl-2', 'Cl-3'))  plt.legend() |



**Marketing Strategies**

**Cluster 0:**

Customers who fall under this category of cluster are having the best credit card and also paying the dues on time without defaults. Hence these group of customers must be rewarded with reward points and thus make them do more transactions in future.

**Cluster1:**

Customers belong to this category of cluster having the highest cash advance and poor credit score yet these customers pay the due amounts of the installments on time. Hence these customers may be given with the loan amounts at less interest charges, thus help the banks providing continuous services to these group of customers in future

**Cluster2:**

Customers belong to this cluster must be the primary focus regarding the marketing strategy because the customers under this cluster are making frequent purchases and also paying the dues on time thus maintaining good credit score. Customers in this cluster must be given with good reward points and provided with increased credit limit or the premium credit cards with some exciting offers make them do more transactions in the future.

**Cluster3:**

Customers belong to this cluster has the least minimum payment ratio and always does the one off payment transactions, hence no bank offers can excite these kind of customers. The marketing to this group of customers is hard and when the usage is minimum, this group can be ignored from the marketing strategy. Further the customers falling under this category can be rejected from issuing the credit cards in future.

**SAME THINGS WE DO IN R**

**R CODE**

|  |
| --- |
| library(dplyr)  #load the data  setwd("C:/Users/jerin/Desktop/R work/EDWISOR PROJECT")  seg <-read.csv("CC GENERAL.csv")  View(seg)  sum(is.na(seg$CUST\_ID))  sum(is.na(seg$BALANCE))  sum(is.na(seg$BALANCE\_FREQUENCY))  sum(is.na(seg$PURCHASES))  sum(is.na(seg$ONEOFF\_PURCHASES))  sum(is.na(seg$INSTALLMENTS\_PURCHASES))  sum(is.na(seg$CASH\_ADVANCE))  sum(is.na(seg$PURCHASES\_FREQUENCY))  sum(is.na(seg$ONEOFF\_PURCHASES\_FREQUENCY))  sum(is.na(seg$PURCHASES\_INSTALLMENTS\_FREQUENCY))  sum(is.na(seg$CASH\_ADVANCE\_FREQUENCY))  sum(is.na(seg$CASH\_ADVANCE\_TRX))  sum(is.na(seg$PURCHASES\_TRX))  sum(is.na(seg$CREDIT\_LIMIT))##1  sum(is.na(seg$PAYMENTS))  sum(is.na(seg$MINIMUM\_PAYMENTS))##313  sum(is.na(seg$PRC\_FULL\_PAYMENT))  sum(is.na(seg$TENURE))  # Identifying Outliers  mystats = function(x) {  nmiss=sum(is.na(x))  a = x[!is.na(x)]  m = mean(a)  n = length(a)  s = sd(a)  min = min(a)  p1=quantile(a,0.01)  p5=quantile(a,0.05)  p10=quantile(a,0.10)  q1=quantile(a,0.25)  q2=quantile(a,0.5)  q3=quantile(a,0.75)  p90=quantile(a,0.90)  p95=quantile(a,0.95)  p99=quantile(a,0.99)  max = max(a)  UC = m+2\*s  LC = m-2\*s  outlier\_flag= max>UC | min<LC  return(c(n=n, nmiss=nmiss, outlier\_flag=outlier\_flag, mean=m, stdev=s,min = min, p1=p1,p5=p5,p10=p10,q1=q1,q2=q2,q3=q3,p90=p90,p95=p95,p99=p99,max=max, UC=UC, LC=LC ))  }  #New Variables creation#  seg$Monthly\_Avg\_PURCHASES = seg$PURCHASES/(seg$PURCHASES\_FREQUENCY\*seg$TENURE)  seg$Monthly\_CASH\_ADVANCE = seg$CASH\_ADVANCE/(seg$CASH\_ADVANCE\_FREQUENCY\*seg$TENURE)  seg$LIMIT\_USAGE = seg$BALANCE/seg$CREDIT\_LIMIT  seg$MIN\_PAYMENTS\_RATIO = seg$PAYMENTS/seg$MINIMUM\_PAYMENTS  -+  +  -Num\_Vars = c(  "BALANCE",  "BALANCE\_FREQUENCY",  "PURCHASES",  "Monthly\_Avg\_PURCHASES",  "ONEOFF\_PURCHASES",  "INSTALLMENTS\_PURCHASES",  "CASH\_ADVANCE",  "Monthly\_CASH\_ADVANCE",  "PURCHASES\_FREQUENCY",  "ONEOFF\_PURCHASES\_FREQUENCY",  "PURCHASES\_INSTALLMENTS\_FREQUENCY",  "CASH\_ADVANCE\_FREQUENCY",  "CASH\_ADVANCE\_TRX",  "PURCHASES\_TRX",  "CREDIT\_LIMIT",  "LIMIT\_USAGE",  "PAYMENTS",  "MINIMUM\_PAYMENTS",  "MIN\_PAYMENTS\_RATIO",  "PRC\_FULL\_PAYMENT",  "TENURE")  Outliers=t(data.frame(apply(seg[Num\_Vars], 2, mystats)))  View(Outliers)  write.csv(Outliers,"Outliers.csv")  # Outlier Treatment  seg$BALANCE[seg$BALANCE>5727.53]=5727.53  seg$BALANCE\_FREQUENCY[seg$BALANCE\_FREQUENCY>1.3510787]=1.3510787  seg$PURCHASES[seg$PURCHASES>5276.46]=5276.46  seg$Monthly\_Avg\_PURCHASES[seg$Monthly\_Avg\_PURCHASES>800.03] = 800.03  seg$ONEOFF\_PURCHASES[seg$ONEOFF\_PURCHASES>3912.2173709]=3912.2173709  seg$INSTALLMENTS\_PURCHASES[seg$INSTALLMENTS\_PURCHASES>2219.7438751]=2219.7438751  seg$CASH\_ADVANCE[seg$CASH\_ADVANCE>5173.1911125]=5173.1911125  seg$Monthly\_CASH\_ADVANCE[seg$Monthly\_CASH\_ADVANCE>2558.53] = 2558.53  seg$PURCHASES\_FREQUENCY[seg$PURCHASES\_FREQUENCY>1.2930919]=1.2930919  seg$ONEOFF\_PURCHASES\_FREQUENCY[seg$ONEOFF\_PURCHASES\_FREQUENCY>0.7991299]=0.7991299  seg$PURCHASES\_INSTALLMENTS\_FREQUENCY[seg$PURCHASES\_INSTALLMENTS\_FREQUENCY>1.1593329]=1.1593329  seg$CASH\_ADVANCE\_FREQUENCY[seg$CASH\_ADVANCE\_FREQUENCY>0.535387]=0.535387  seg$CASH\_ADVANCE\_TRX[seg$CASH\_ADVANCE\_TRX>16.8981202]=16.8981202  seg$PURCHASES\_TRX[seg$PURCHASES\_TRX>64.4251306]=64.4251306  seg$CREDIT\_LIMIT[seg$CREDIT\_LIMIT>11772.09]=11772.09  seg$LIMIT\_USAGE[seg$LIMIT\_USAGE>1.1683] = 1.1683  seg$PAYMENTS[seg$PAYMENTS>7523.26]=7523.26  seg$MINIMUM\_PAYMENTS[seg$MINIMUM\_PAYMENTS>5609.1065423]=5609.1065423  seg$MIN\_PAYMENTS\_RATIO[seg$MIN\_PAYMENTS\_RATIO>249.9239] = 249.9239  seg$PRC\_FULL\_PAYMENT[seg$PRC\_FULL\_PAYMENT>0.738713]=0.738713  seg$TENURE[seg$TENURE>14.19398]=14.19398  # Missing Value Imputation with mean  seg$MINIMUM\_PAYMENTS[which(is.na(seg$MINIMUM\_PAYMENTS))] = 721.9256368  seg$CREDIT\_LIMIT[which(is.na(seg$CREDIT\_LIMIT))] = 4343.62  seg$Monthly\_Avg\_PURCHASES[which(is.na(seg$Monthly\_Avg\_PURCHASES))] =184.8991609  seg$Monthly\_CASH\_ADVANCE[which(is.na(seg$Monthly\_CASH\_ADVANCE))] = 717.7235629  seg$LIMIT\_USAGE[which(is.na(seg$LIMIT\_USAGE))] =0.3889264  seg$MIN\_PAYMENTS\_RATIO[which(is.na(seg$MIN\_PAYMENTS\_RATIO))] = 9.3500701  # Checking Missing Value  check\_Missing\_Values=t(data.frame(apply(seg[Num\_Vars], 2, mystats)))  View(check\_Missing\_Values)  write.csv(seg,"Missing\_value\_treatment.csv")  # Variable Reduction (Factor Analysis)  Step\_nums = seg[Num\_Vars]  corrm= cor(Step\_nums)  View(corrm)  write.csv(corrm, "Correlation\_matrix.csv")  scree(corrm,factors=T,pc=T,main="scree plot", hline=NULL, add=FALSE)### SCREE PLOT  eigen(corrm)$values  eigen\_values = mutate(data.frame(eigen(corrm)$values)  ,cum\_sum\_eigen=cumsum(eigen.corrm..values)  , pct\_var=eigen.corrm..values/sum(eigen.corrm..values)  , cum\_pct\_var=cum\_sum\_eigen/sum(eigen.corrm..values))  write.csv(eigen\_values, "EigenValues2.csv")  # standardizing the data  segment\_prepared =seg[Num\_Vars]  segment\_prepared = scale(segment\_prepared)  write.csv(segment\_prepared, "standardized data.csv")  #building clusters using k-means clustering  cluster\_three = kmeans(segment\_prepared,3)  cluster\_four = kmeans(segment\_prepared,4)  cluster\_five = kmeans(segment\_prepared,5)  cluster\_six = kmeans(segment\_prepared,6)  seg\_new=cbind(seg,km\_clust\_3=cluster\_three$cluster,km\_clust\_4=cluster\_four$cluster,km\_clust\_5=cluster\_five$cluster ,km\_clust\_6=cluster\_six$cluster )  View(seg\_new)  # Profiling  Num\_Vars2 = c(  "Monthly\_Avg\_PURCHASES",  "Monthly\_CASH\_ADVANCE",  "CASH\_ADVANCE",  "CASH\_ADVANCE\_TRX",  "CASH\_ADVANCE\_FREQUENCY",  "ONEOFF\_PURCHASES",  "ONEOFF\_PURCHASES\_FREQUENCY",  "PAYMENTS",  "CREDIT\_LIMIT",  "LIMIT\_USAGE",  "PURCHASES\_INSTALLMENTS\_FREQUENCY",  "PURCHASES\_FREQUENCY",  "INSTALLMENTS\_PURCHASES",  "PURCHASES\_TRX",  "MINIMUM\_PAYMENTS",  "MIN\_PAYMENTS\_RATIO",  "BALANCE",  "TENURE"  )  library(tables)  tt =cbind(tabular(1+factor(km\_clust\_3)+factor(km\_clust\_4)+factor(km\_clust\_5)+  factor(km\_clust\_6)~Heading()\*length\*All(seg[1]),  data=seg\_new),tabular(1+factor(km\_clust\_3)+factor(km\_clust\_4)+factor(km\_clust\_5)+  factor(km\_clust\_6)~Heading()\*mean\*All(seg[Num\_Vars2]),  data=seg\_new))  tt2 = as.data.frame.matrix(tt)  View(tt2)  rownames(tt2)=c(  "ALL",  "KM3\_1",  "KM3\_2",  "KM3\_3",  "KM4\_1",  "KM4\_2",  "KM4\_3",  "KM4\_4",  "KM5\_1",  "KM5\_2",  "KM5\_3",  "KM5\_4",  "KM5\_5",  "KM6\_1",  "KM6\_2",  "KM6\_3",  "KM6\_4",  "KM6\_5",  "KM6\_6")  colnames(tt2)=c(  "SEGMENT\_SIZE",  "Monthly\_Avg\_PURCHASES",  "Monthly\_CASH\_ADVANCE",  "CASH\_ADVANCE",  "CASH\_ADVANCE\_TRX",  "CASH\_ADVANCE\_FREQUENCY",  "ONEOFF\_PURCHASES",  "ONEOFF\_PURCHASES\_FREQUENCY",  "PAYMENTS",  "CREDIT\_LIMIT",  "LIMIT\_USAGE",  "PURCHASES\_INSTALLMENTS\_FREQUENCY",  "PURCHASES\_FREQUENCY",  "INSTALLMENTS\_PURCHASES",  "PURCHASES\_TRX",  "MINIMUM\_PAYMENTS",  "MIN\_PAYMENTS\_RATIO",  "BALANCE",  "TENURE"  )  tt2  cluster\_profiling2 = t(tt2) |

**References**

<https://www.kaggle.com>

<http://rprogramming.net/>

<https://medium.com/>

<https://stackoverflow.com/>

<https://www.rdocumentation.org>

<https://www.analyticsvidhya.com/blog>