Project Report

On

Credit Card Segmentation

By

Swapnil Samudre

INDEX

1 Introduction

Now a day's people are doing more purchases through Credit cards. For this customer needs to have good credit score to avail greater credit limit. To identify which segment of people are potential customer to purchase and maintain good relationship with bank can be offered good offers on purchases and bonus /redeem points to provide better services with profits.

1.1 Problem Statement

This case requires trainees to develop a customer segmentation to define marketing strategy. The sample dataset summarizes the usage behavior of about 9000 active credit card holders during the last 6 months. The file is at a customer level with 18 Behavioral variables.

1.2 Data and DataSet

Understanding of data is the very first and important step in the process of finding solution of any business problem. Here in our case
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.

Size of Dataset provided: - 8950 rows, 18 Columns
Missing Values: Yes
Outliers Presented: Yes

2 Methodology:

2.1 Pre-Processing (EDA):

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

1. Data exploration and Cleaning
2. Missing values Analysis
3. Outlier Analysis
4. Feature Selection
5. Features Scaling
6. Visualization

- **2.2 Modeling:**
  - K-Means Clustering:

Once all the Pre-Processing steps has been done on our data set, we will now further move to our next step which is modeling. Modeling plays an important role to find out the good inferences from the data. Choice of models depends upon the problem statement and data set. As per our problem statement and dataset, we will use K-means Algorithm on our preprocessed data and use a metrics like Elbow method to determine the Number of Clusters we should take in account.

- We have also used Dimensionality Reduction (Principal Component Analysis) to reduce the number of features in our dataset.
  - ✓ Principal Component Analysis:

3 Pre-processing:

3.1 Data exploration and Cleaning (Missing Values and Outliers)

The very first step which comes with any data science project is data exploration and cleaning which includes following points as per this project:

a. As we can see that we have two faeture with missing value i.e¶

1. **MINIMUM_PAYMENTS - 313(3.50%)**
2. **CREDIT_LIMIT - 1 (0.01%)**

b. There are some outlier figures in the DataSet. As there are so many Outliers in are dataset removing those would be loss in information. We'll perform Log transformation to outperform outlier effect.

c. Feature Selection using PCA.

d. Feature Scaling using Normalization to make it a same range and also for PCA analysis.

e. Data visualization For derived New KPI'S and other features.

3. 2 Creating New variables:

Here in our data set we have derived new KPI'S

- monthly average purchase
- cash advance amount
- purchases by type
- limit usage
- payments to minimum payments ratio

3.3 Feature Scaling:

Most of the times, your dataset will contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use Euclidian distance between two data points in their computations, this is a problem.

If left alone, these algorithms only take in the magnitude of features neglecting the units. The results would vary greatly between different units, 5kg and 5000gms. The features with high magnitudes will weigh in a lot more in the distance calculations than features with low magnitude. To supress this effect, we need to bring all features to the same level of magnitudes.

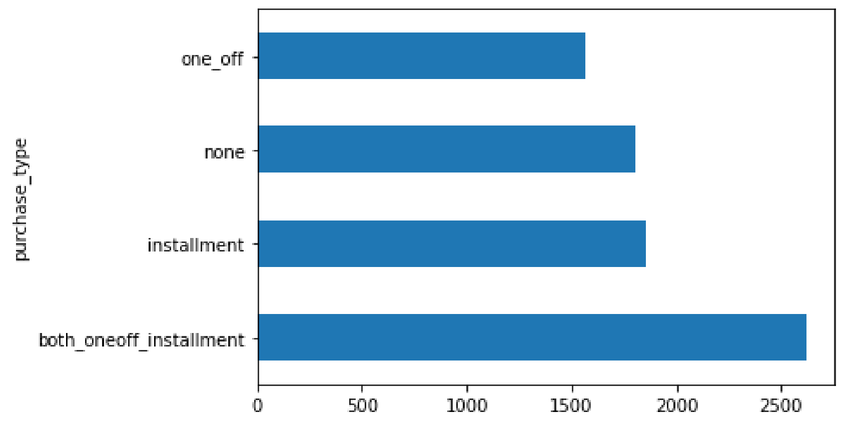
In our Dataset the data are Skewed, **Skewness** is asymmetry in a statistical distribution, in which the curve appears distorted or skewed either to the left or to the right. Skewness can be quantified to define the extent to which a distribution differs from a normal distribution. Here we have used Normalization (MinMaxScaler) on the data that will scale to a fixed range—usually 0 to 1.

3.4 Data Visualization:

BALANCE FREQUENY BY PURCHASE TYPE

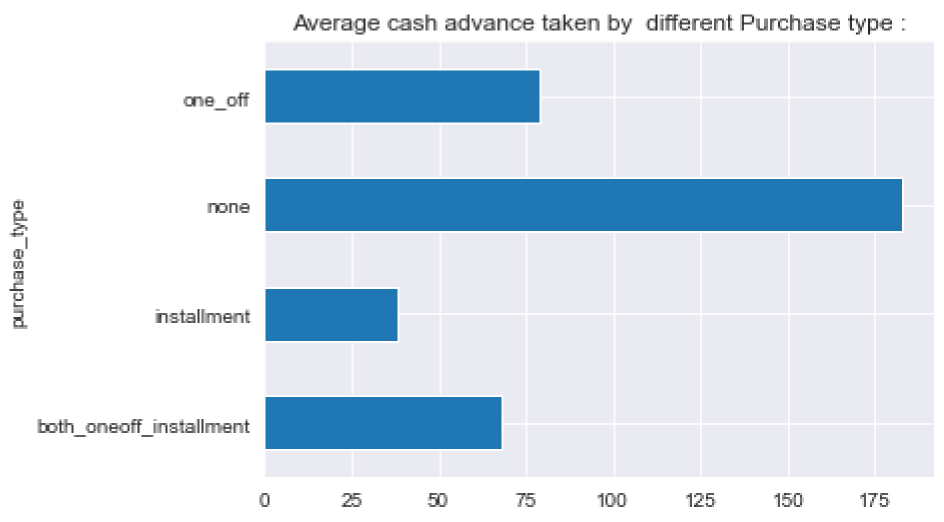
Insights Form New KPI'S ¶

In [37]:
```
cc_data.groupby('purchase_type')['BALANCE_FREQUENCY'].sum().plot.barh()
```
Out[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x20e001bcb08>
```



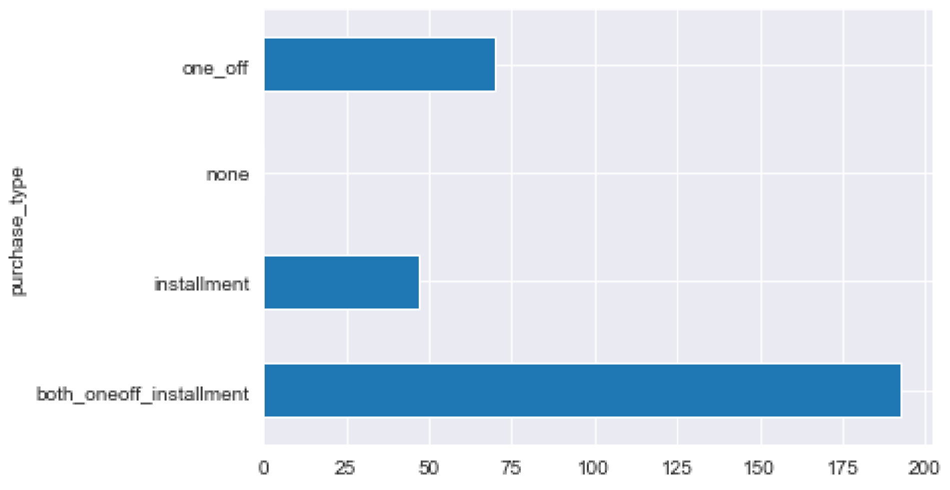
CASH ADVANCE BY PURCHASE TYPE

In [101]:
```
#%matplotlib notebook
cc_data.groupby('purchase_type')['CASH_AVG_AMOUNT'].mean().plot.barh()

plt.title('Average cash advance taken by  different Purchase type :')
plt.show()
```



MONTHLY PURCHASE BY PURCHASE TYPE

In [102]:
```
cc_data.groupby('purchase_type')['MONTHLY_AVG_PURCHASE'].mean().plot.barh()
```
Out[102]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x20e0441f648>
```



4 Modeling:

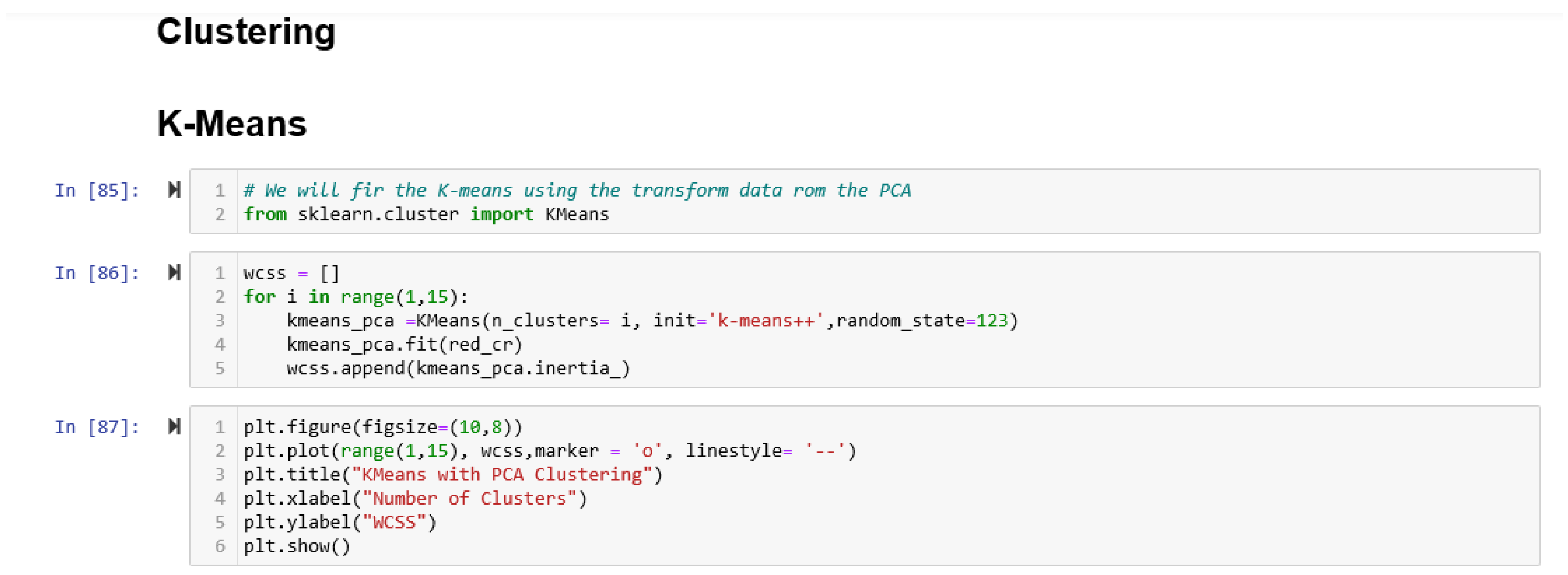
We will use Machine Learning Models on the processed data to make different segments/cluster/groups. We'll use K-Means Clustering Algorithm to make different Clusters of Customer:

- K-Means Clustering:

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

The way k-means algorithm works is as follows:

- Specify number of clusters K.
- Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
- Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.
- Compute the sum of the squared distance between data points and all centroids.
- Assign each data point to the closest cluster (centroid).
- Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.



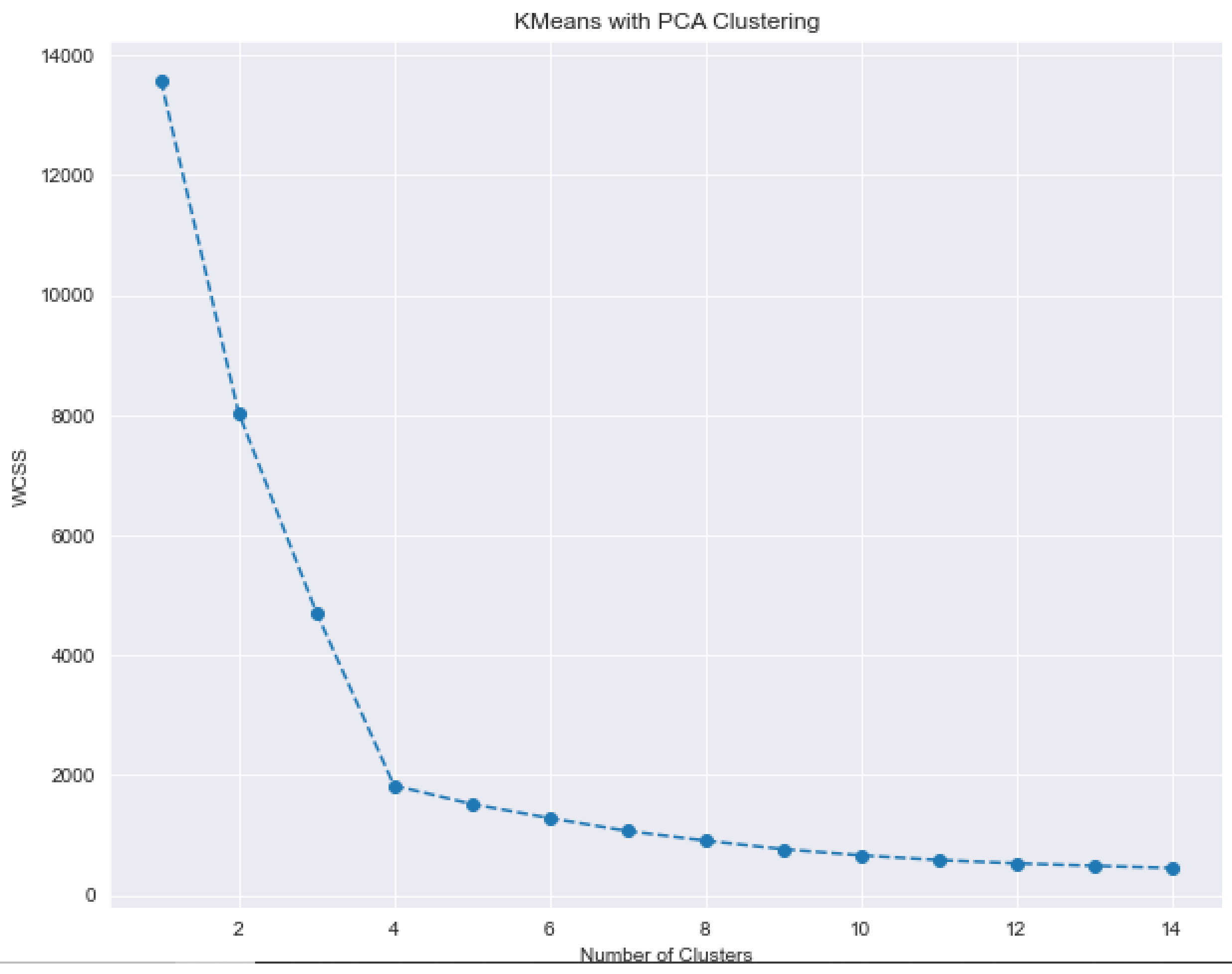
Clustering

K-Means

In [85]:
```
# We will fir the K-means using the transform data rom the PCA
from sklearn.cluster import KMeans
```

In [86]:
```
wcss = []
for i in range(1,15):
    kmeans_pca =KMeans(n_clusters= i, init='k-means++',random_state=123)
    kmeans_pca.fit(red_cr)
    wcss.append(kmeans_pca.inertia_)
```

In [87]:
```
plt.figure(figsize=(10,8))
plt.plot(range(1,15), wcss,marker = 'o', linestyle= '--')
plt.title("KMeans with PCA Clustering")
plt.xlabel("Number of Clusters")
plt.ylabel("WCSS")
plt.show()
```



Elbow Method

we determine the number of clusters we'd like to keep. To that effect, we use the **Elbow-method**. The approach consists of looking for a kink or **elbow** in the **WCSS graph**. Usually, the part of the graph before the elbow would be steeply declining, while the part after it – much smoother. In this instance, the kink comes at the **4 clusters mark**. So, we'll be keeping a **four-cluster solution**.

```
[88]:
#We have Chosen Four Clusters,so we run K-means with number of Cluster equals Four.
kmeans_pca=KMeans(n_clusters=4,init='k-means++',random_state=123)
```

```
[89]:
kmeans_pca.fit(red_cr)
```

```
Out[89]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
                n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
                random_state=123, tol=0.0001, verbose=0)
```

```
[90]:
pd.Series(kmeans_pca.labels_).value_counts()
```

```
Out[90]: 2    2774
         0    2260
         1    2042
         3    1874
         dtype: int64
```

Let's analyze PCA & K-means Results

we'll create a new data frame. It allows us to add in the values of the separate components to our segmentation data set. The components' scores are stored in the 'red_cr' variable. Let's label them Component 1, 2 and 3. In addition, we also append the 'K means P C A' labels to the new data frame.

In [91]:
```python
col_ =['PURCHASES_TRX','MONTHLY_AVG_PURCHASE','CASH_AVG_AMOUNT','LIMIT_USAGE','CASH_ADVANCE_TRX',
      'PAY_MIN_RATIO','both_oneoff_installment','installment','one_off','none','CREDIT_LIMIT']
```

In [92]:
```python
# we'll create a new DF with original features and add the PCA scores and Kmeans clusters
cr_dum_data_pca_kmeans = pd.concat([cc_original[col_].reset_index(drop=True),pd.DataFrame(red_cr)],axis=1)
cr_dum_data_pca_kmeans.columns.values[-4:] = ['Component 1','Component 2', 'Component 3','Component 4']
cr_dum_data_pca_kmeans['Cluster_4'] = kmeans_pca.labels_
```

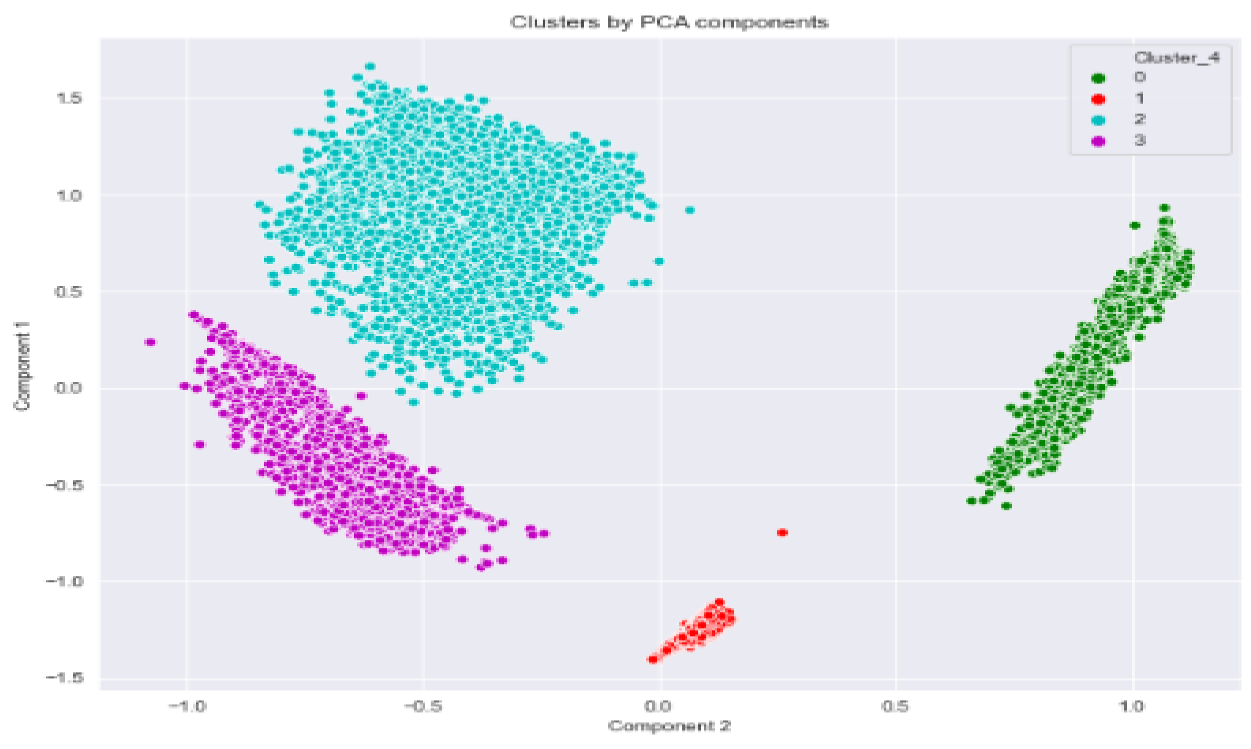
In [93]:
```python
cr_dum_data_pca_kmeans.head()
```

Out[93]:

	PURCHASES_TRX	MONTHLY_AVG_PURCHASE	CASH_AVG_AMOUNT	LIMIT_USAGE	CASH_ADVANCE_TRX	PAY_MIN_RATIO	both_oneoff_installment
0	2	7.950000	0.000000	0.040901	0	1.446508	0
1	0	0.000000	536.912124	0.457495	4	3.826241	0
2	12	64.430833	0.000000	0.332687	0	0.991682	0
3	1	124.916667	17.149001	0.222223	1	0.000000	0
4	1	1.333333	0.000000	0.681429	0	2.771075	0

Now let's **Visualize Clusters by Components**

In [94]:
```python
# Plot data by components. The Y-axis is the 1 component and X-axis is the 2 component.
x_axis = cr_dum_data_pca_kmeans['Component 2']
y_axis = cr_dum_data_pca_kmeans['Component 1']
plt.figure(figsize=(10,8))
sns.scatterplot(x_axis, y_axis, hue=cr_dum_data_pca_kmeans['Cluster_4'],palette=['g', 'r','c','m'])
plt.title('Clusters by PCA components')
plt.show()
```

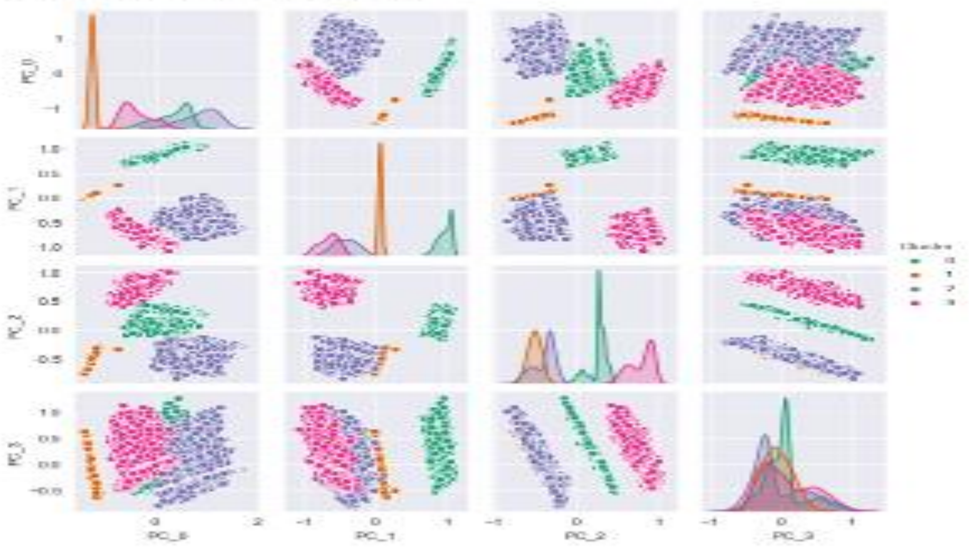


Let's Visualize all components by pair plots

```
In [95]:  df_pair_plot=pd.DataFrame(red_cr,columns=['PC_' +str(i) for i in range(4)])

In [96]:  df_pair_plot['Cluster']=kmeans_pca.labels_

In [97]:  #pairwise relationship of components on the data
          plt.figure(figsize=(15,7))
          sns.pairplot(df_pair_plot,hue='Cluster', palette= 'Dark2', diag_kind='kde',size=1.85)
```

```
E:\Ana_2020\lib\site-packages\seaborn\axisgrid.py:2079: UserWarning: The `size` parameter has been renamed to `height`; ple
ase update your code.
  warnings.warn(msg, UserWarning)
```

```
Out[97]: <seaborn.axisgrid.PairGrid at 0x20e01be8108>

         <Figure size 1080x504 with 0 Axes>
```

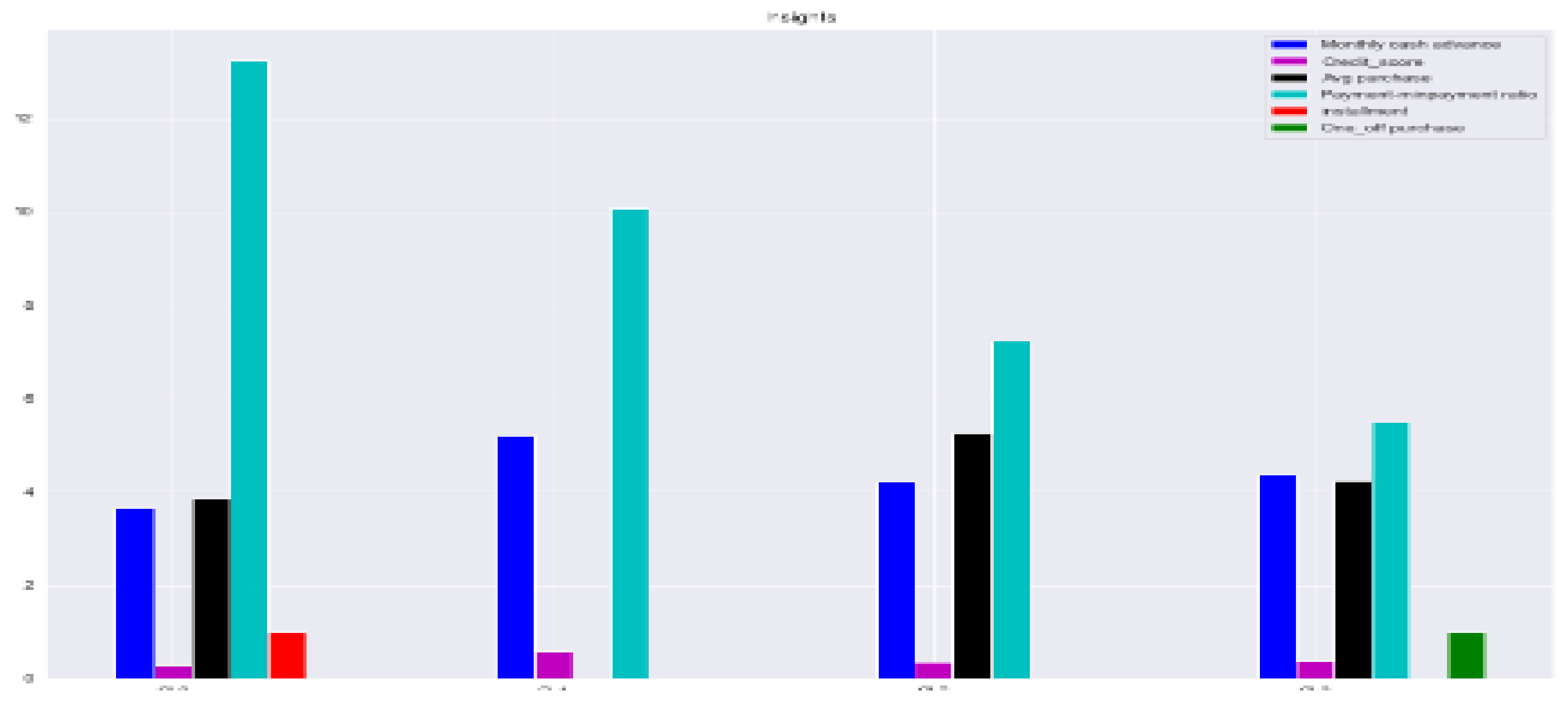


It shows that first two components are able to indentify clusters

```
fig,ax=plt.subplots(figsize=(15,10))
index=np.arange(len(cluster_4.columns))

cash_advance=np.log(cluster_4.loc['CASH_AVG_AMOUNT',:].values)
credit_score=(cluster_4.loc['LIMIT_USAGE',:].values)
purchase= np.log(cluster_4.loc['MONTHLY_AVG_PURCHASE',:].values)
payment=cluster_4.loc['PAY_MIN_RATIO',:].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()
```

```
E:\Ana_2020\lib\site-packages\ipykernel_launcher.py:6: RuntimeWarning: divide by zero encountered in log
```

```
]: <matplotlib.legend.Legend at 0x20e04523808>
```

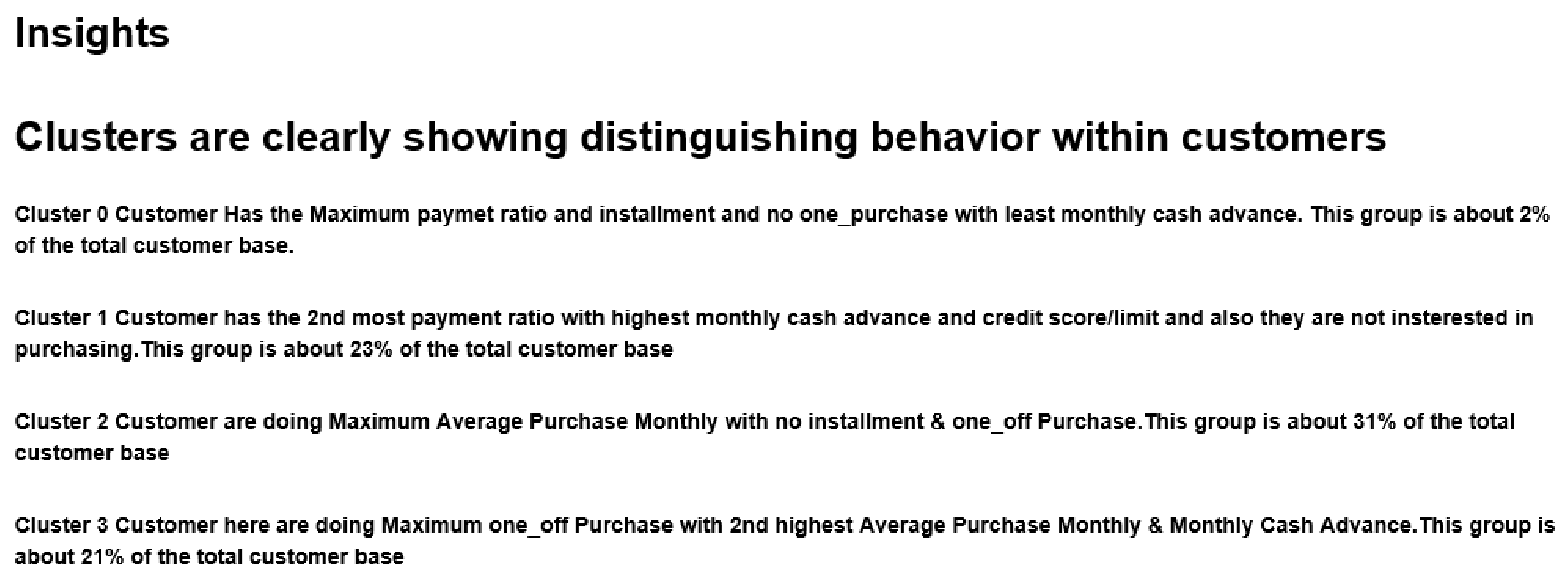


5 Conclusion:

Model Evaluation:

Finally we have used 4 clusters as Elbow method was clearly show casting for 4 clusters.

The Insights from those clusters are as follows:



Insights

Clusters are clearly showing distinguishing behavior within customers

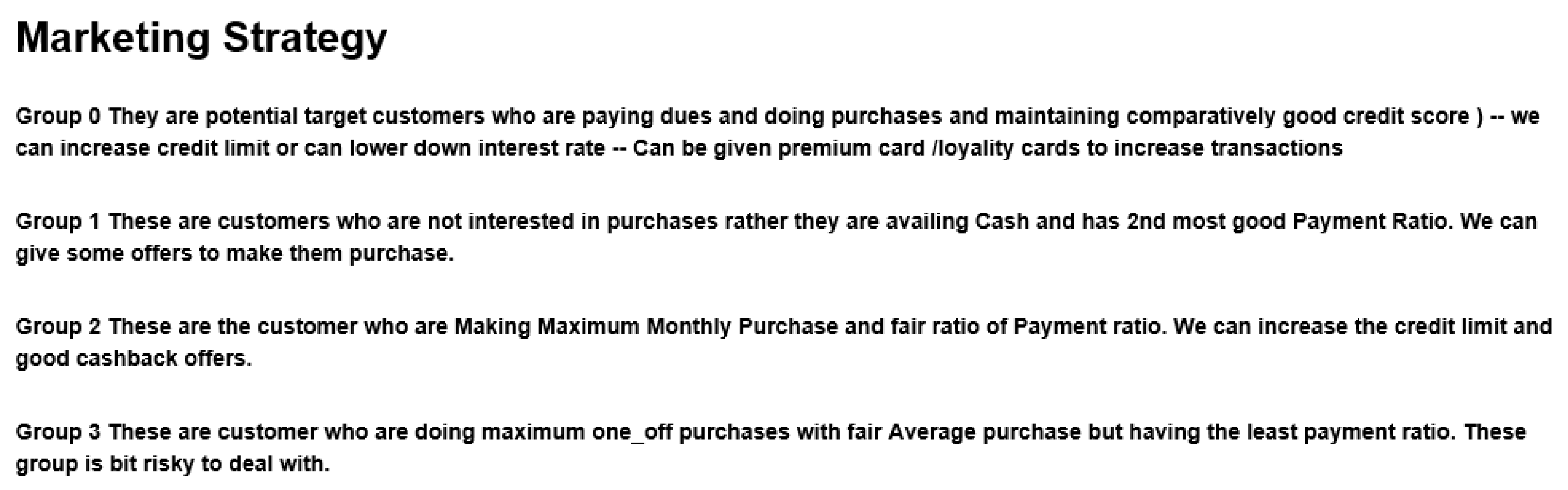
Cluster 0 Customer Has the Maximum paymet ratio and installment and no one_purchase with least monthly cash advance. This group is about 2% of the total customer base.

Cluster 1 Customer has the 2nd most payment ratio with highest monthly cash advance and credit score/limit and also they are not insterested in purchasing.This group is about 23% of the total customer base

Cluster 2 Customer are doing Maximum Average Purchase Monthly with no installment & one_off Purchase.This group is about 31% of the total customer base

Cluster 3 Customer here are doing Maximum one_off Purchase with 2nd highest Average Purchase Monthly & Monthly Cash Advance.This group is about 21% of the total customer base

The Marketing Strategy was suggested is as follows:



Marketing Strategy

Group 0 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 1 These are customers who are not interested in purchases rather they are availing Cash and has 2nd most good Payment Ratio. We can give some offers to make them purchase.

Group 2 These are the customer who are Making Maximum Monthly Purchase and fair ratio of Payment ratio. We can increase the credit limit and good cashback offers.

Group 3 These are customer who are doing maximum one_off purchases with fair Average purchase but having the least payment ratio. These group is bit risky to deal with.

R code:

```
rm(list=ls())

getwd()


setwd("E:/CC R Project")


install.packages(c('tidyverse','DataExplorer','Amelia'))


##############################################Read the da-
ta#############################################
credit = read.csv("credit-card-data.csv")


##############################################Explore the da-
ta#############################################

str(credit)

summary(credit)


#We will take out the CUST_ID since it was a unique variable and we can’t get
further information from it.
credit = credit[-1]

head(credit)

####################################Missing Values Analy-
sis##################################################

cc_func <- function(x){
  nmiss = sum(is.na(x))
  a = x[!is.na(x)]
  n = length(a)
  m = mean(a)
  min = min(a)
  max = max(a)
  s = sd(a)
  p1 = quantile(a, 0.95)
  p2 = quantile(a, 0.99)
  UL = m+3*s
  LL = m-3*s
  return(c(n=n, nmiss=nmiss, Mean=m, Min=min, Max=max, StDev=s, P1=p1, 
P2=p2, 'Upper Limit'=UL, 'Lower Limit'=LL))
}

vars <-
c("BALANCE","BALANCE_FREQUENCY","PURCHASES","ONEOFF_PUR
CHASES","INSTALLMENTS_PURCHASES",

"CASH_ADVANCE","PURCHASES_FREQUENCY","ONEOFF_PURCHASES
_FREQUENCY","PURCHASES_INSTALLMENTS_FREQUENCY",

"CASH_ADVANCE_FREQUENCY","CASH_ADVANCE_TRX","PURCHASE
S_TRX","CREDIT_LIMIT","PAYMENTS",
        "MINIMUM_PAYMENTS","PRC_FULL_PAYMENT","TENURE")

describe_stats <- t(data.frame(apply(credit[vars],2,cc_func)))
describe_stats




#As we can see that we have two faeture with missing value i.e
#1.MINIMUM_PAYMENTS 313(3.50%)
#2.CREDIT_LIMIT    1(0.01%)

library(Amelia)
missmap(credit, main = "Missing value map")


#Mean Method
credit$MINIMUM_PAYMENTS[is.na(credit$MINIMUM_PAYMENTS)] = 
mean(credit$MINIMUM_PAYMENTS, na.rm = T)

credit$CREDIT_LIMIT [is.na(credit$CREDIT_LIMIT )] = 
mean(credit$CREDIT_LIMIT , na.rm = T)

#Check Null values after Imputing by Mean
sum(is.na(credit))


############################################Deriving New 
KPI'S############################################
#1.Monthly_avg_purchase
```

```
credit$Monthly_avg_purchase = credit$PURCHASES/credit$TENURE
#2.Monthly_cash_advance
credit$Monthly_cash_advance = credit$CASH_ADVANCE/credit$TENURE
#3.Limit_Usage
credit$Limit_Usage = credit$BALANCE/credit$CREDIT_LIMIT
#4.payment_minpay
credit$payment_minpay = credit$PAYMENTS/credit$MINIMUM_PAYMENTS

sum(is.na(credit))


library(tidyverse)
library(DataExplorer)

plot_missing(credit)


plot_histogram(credit)

boxplot(credit)

# Standardise
ccdata = scale(credit)


# Use the "elbow method" to find the optimal number of clusters
# Gives us 8 clusters being optimal
set.seed(123)
WCSS <- vector()
for (i in 1:20) WCSS[i] <- sum(kmeans(credit, i)$withins)
plot(1:20, WCSS, type = "b", main = "Elbow Method", xlab = "Clusters", ylab =
"WCSS")

#We are getting 5 cluster


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")


pc <- princomp(credit)
plot(pc)


summary(pc)
loadings(pc)

pc <- princomp(credit)
plot(pc)
plot(pc, type='l')
summary(pc)

# Scale
data2 <- data.frame(scale(credit))
# Verify variance is uniform
plot(sapply(data2, var))

# Get principal component vectors using prcomp instead of princomp
pc <- prcomp(data2)

# First for principal components
comp <- data.frame(pc$x[,1:4])

# Plot
plot(comp, pch=16, col=rgb(0,0,0,0.5))

#Determine number of clusters
wss <- (nrow(data2)-1)*sum(apply(data2,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(data2,
                              centers=i)$withinss)
```

```
plot(1:15, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")


# Apply k-means with k=4
k <- kmeans(comp, 4, nstart=25, iter.max=1000)
library(RColorBrewer)
library(scales)
palette(alpha(brewer.pal(9,'Set1'), 0.5))
plot(comp, col=k$clust, pch=16)
```

End