**PROJECT REPORT**

**ON**

**CUSTOMER CARD SEGMENTATION**

**By**

**TEJAS A C**

**Chapter 1: INTRODUCTION**

* **1.1 Problem Statement**
* **1.2 Data**

**Chapter 2: Data Preprocessing**

* **2.1 Missing Value Analysis**
* **2.2 Distribution of Variables**
* **2.3 Feature Selection**
* **2.4 Feature Scaling**
* **2.5 Dimensionality Reduction(PCA)**
* **2.6 Insights of KPI’s**

**Chapter 3: Modelling**

* **3.1 Elbow Method**
* **3.2 Clustering using Kmeans**
* **3.3 Model Evaluation**

**Appendix**

**Customer Card Segmentation**

**Chapter 1: INTRODUCTION**

**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:**

In this project our task is to apply a data reduction technique factor analysis and a clustering algorithm to reveal the behavioral segments of credit card holders.

The data that we have to predict the behavior of the credit card holders are given below:

* 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
* INSTALLMENT\_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 purchase 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

**Chapter 2: Data Preprocessing**

Data preprocessing is the major step before training the model because the Real-world data we obtained is incomplete and inconsistent there are lack of behaviors. In some cases, we need to extract the new features from the data we have this can be done on a clear understanding of business problem and data.

**2.1 Missing Value Analysis:**

In missing value analysis, we find if there are any missing cells present in the data or not. If there is any data then we need to fill that using various techniques like mean, median, KNNImputation etc.

We find missing values in the data using

* credit\_data.isnull().sum()

|  | Missing\_Values | Columns | Missing\_Percentage |
| --- | --- | --- | --- |
|  |  |  |  |
| 1 | 0 | CUST\_ID | 0.00000000 |
| 2 | 0 | BALANCE | 0.00000000 |
| 3 | 0 | BALANCE\_FREQUENCY | 0.00000000 |
| 4 | 0 | PURCHASES | 0.00000000 |
| 5 | 0 | ONEOFF\_PURCHASES | 0.00000000 |
| 6 | 0 | INSTALLMENTS\_PURCHASES | 0.00000000 |
| 7 | 0 | CASH\_ADVANCE | 0.00000000 |
| 8 | 0 | PURCHASES\_FREQUENCY | 0.00000000 |
| 9 | 0 | ONEOFF\_PURCHASES\_FREQUENCY | 0.00000000 |
| 10 | 0 | PURCHASES\_INSTALLMENTS\_FREQUENCY | 0.00000000 |
| 11 | 0 | CASH\_ADVANCE\_FREQUENCY | 0.00000000 |
| 12 | 0 | CASH\_ADVANCE\_TRX | 0.00000000 |
| 13 | 0 | PURCHASES\_TRX | 0.00000000 |
| 14 | 1 | CREDIT\_LIMIT | 0.01117318 |
| 15 | 0 | PAYMENTS | 0.00000000 |
| 16 | 313 | MINIMUM\_PAYMENTS | 3.49720670 |
| 17 | 0 | PRC\_FULL\_PAYMENT | 0.00000000 |
| 18 | 0 | TENURE | 0.00000000 |

Showing 1 to 15 of 18 entries, 3 total columns

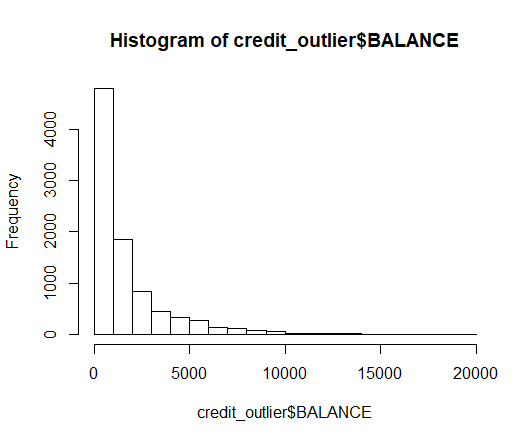
So here we have missing values present in two variables, we will impute those missing values using the median method.

**2.2 Distribution of Variables and Outlier Analysis**

Let us look few of the variable’s distribution

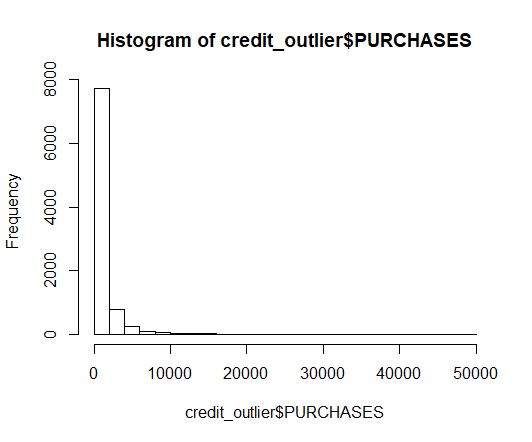
Let us see if we can find some interesting trend in the data distribution, especially on Balance, Purchase, Credit\_Limit and Tenure.

Data Distribution of Balance variable:

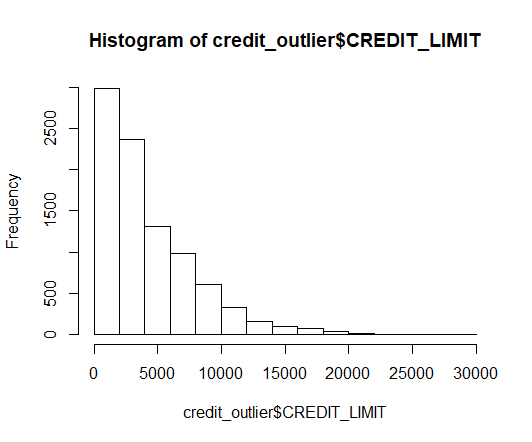


It seems most of the customers who use their Credit Card would maximize the usage of their credit balance, until it reaches 0.

Data Distribution of Purchases variable

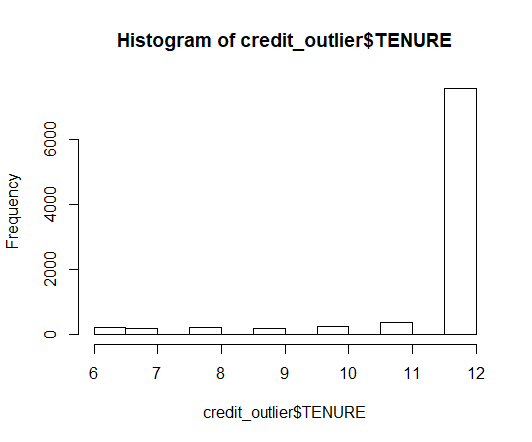


Data Distribution of Credit\_Limit variable:



All the mentioned histograms are Right skewed or has a positive skew distribution.

Data Distribution of Tenure variable:



The Tenure variable is left skewed or has a negative distribution.

Most of the customers have 12 months Tenure, with significant numbers compared to other tenure.

Has we can see the variables are either Right skewed or Left skewed there is a need for a transformation technique.

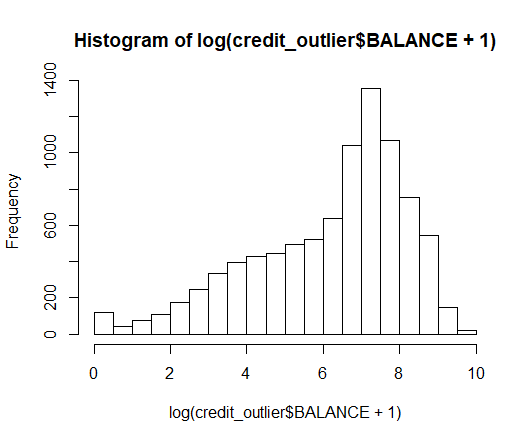
Since there are variables having extreme values, so I am doing log-transformation on the dataset to remove outlier effect.

After performing log-transformation we see.

**Log-Transformation:**

The **log transformation** can be used to **make** highly skewed distributions less skewed. This can be valuable both for making patterns in the data more interpretable and for helping to meet the assumptions of inferential statistics.

Data Distribution for Balance after performing log-transformation:



**2.3 Feature Selection**

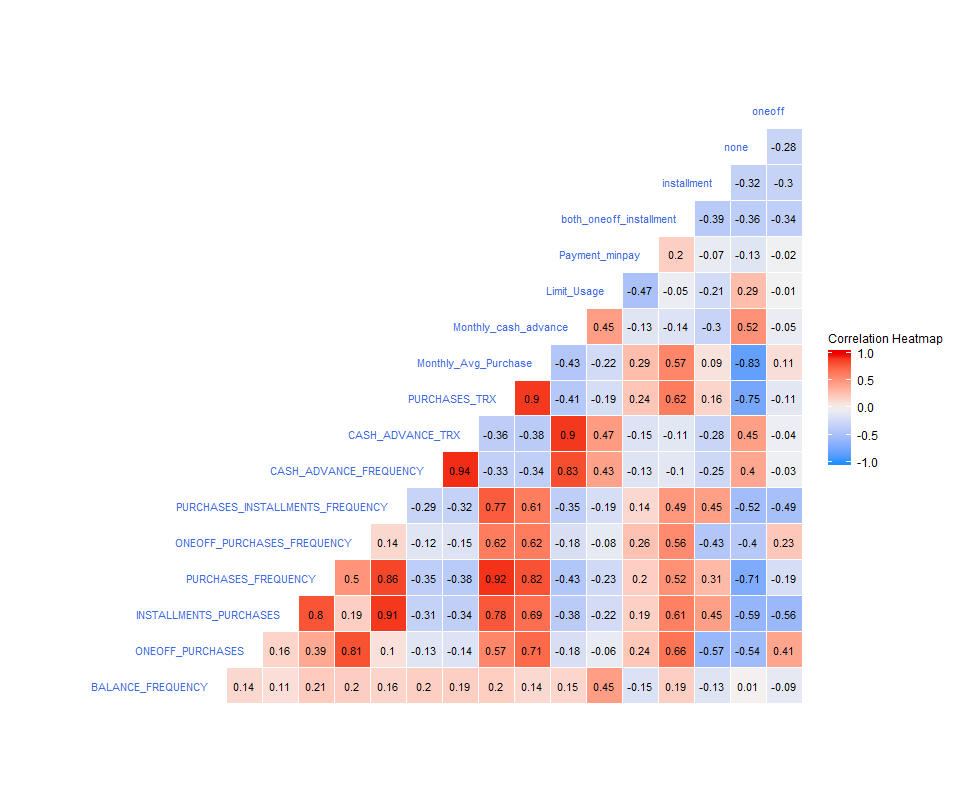
Feature selection is very important for modelling the dataset. Here every dataset has good and unwanted features. The unwanted features would affect on performance of model, so we have to delete those features. We have to select best features by using ANOVA, Chi-Square test and correlation matrix statistical techniques and so on. In this, we are selecting best features by using Correlation matrix.

After deriving ‘intelligent’ KPI’s such as monthly average purchase and cash advance amount, purchases by type (one-off, instalments), limit usage (balance to credit limit ratio), payments to minimum payments ratio etc.

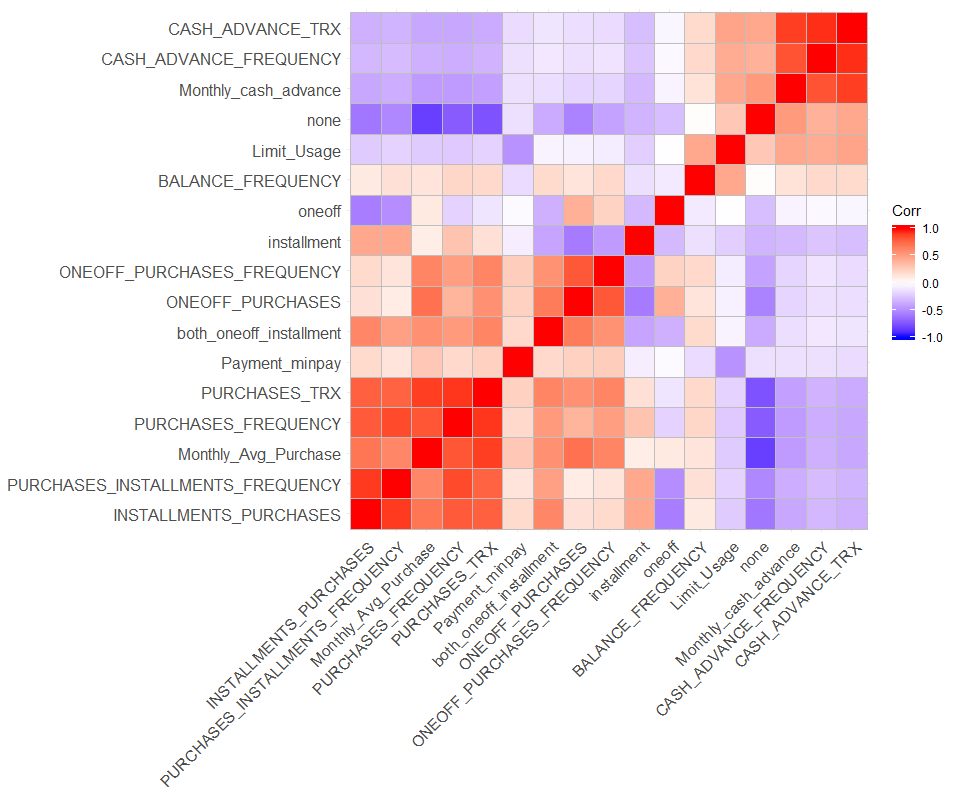
We come up with new useful variables for the purpose of data understanding, so when we derive some useful variables out of already existing variables, we need to drop those variables which is less informative.

We will come to this topic later when I explain about the derived KPI’s, for now we will understand the correlation diagram.

Plotting Correlation Matrix



Correlation Heat Map



* Since all of the variables are numeric, it will easier to see their correlation by using correlation matrix.
* Heat map shows that many features are co-related so applying dimensionality reduction will help negating multi-collinearity in data.
* Some of them are strongly corelated either positive or negative such as CASH\_ADVANCE\_TRX and CASH\_ADVANCE\_FREQUENCY and Monthly\_Avg\_Purchase and none.

**2.4 Feature Scaling**

In the given dataset the data in not scaled that is we have different scales for different variables, this will increase the weightage on particular variables while we develop the clustering model.

So there arises a need to scale the dataset in order to have uniform weightage to all the variables present in the dataset.

To perform scaling, we need to make sure we just have numeric variables, which we have already done, because scaling can only be done on numeric variables.

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.

We scale the data with commands like:

* scale(dataframe.name)

In python we use a particular library to scale the dataset.

* from sklearn.preprocessing import StandardScaler

then using this library, we fit the dataset and transform it so the dataset is scaled.

**2.5 Dimensionality Reduction (PCA)**

We are applying Principal Component Analysis because we have the dataset with only numeric variables.

**Principal Component Analysis:**

The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent. The same is done by transforming the variables to a new set of variables, which are known as the principal components (or simply, the PCs) and are orthogonal, ordered such that the retention of variation present in the original variables decreases as we move down in the order. So, in this way, the 1st principal component retains maximum variation that was present in the original components. The principal components are the eigenvectors of a covariance matrix, and hence they are orthogonal.

Importantly, the dataset on which PCA technique is to be used must be scaled. The results are also sensitive to the relative scaling. As a layman, it is a method of summarizing data.

Intuitively, Principal Component Analysis can supply the user with a lower-dimensional picture, a projection or "shadow" of this object when viewed from its most informative viewpoint.

**PCA Technique:**

Before determine the cluster of Customers, we need to conduct PCA to reduce the dimensionality but maintain information as much as possible. Therefore, we need to determine which PCs that we are going to select based on the **Proportion of Variance.**

Importance of components:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Standard deviation** | **Proportion of Variance** | **Cumulative Proportion** |
| **PC1** | 2.6144 | 0.4021 | 0.4021 |
| **PC2** | 1.7521 | 0.1806 | 0.5826 |
| **PC3** | 1.5824 | 0.1473 | 0.7299 |
| **PC4** | 1.17784 | 0.08161 | 0.81154 |
| **PC5** | 1.05532 | 0.06551 | 0.87706 |
| **PC6** | 0.84089 | 0.04159 | 0.91865 |
| **PC7** | 0.61769 | 0.02244 | 0.94109 |
| **PC8** | 0.59061 | 0.02052 | 0.96161 |
| **PC9** | 0.45852 | 0.01237 | 0.97398 |
| **PC10** | 0.40421 | 0.00961 | 0.98359 |
| **PC11** | 0.32295 | 0.00614 | 0.98972 |
| **PC12** | 0.22697 | 0.00303 | 0.99276 |
| **PC13** | 0.21168 | 0.00264 | 0.99539 |
| **PC14** | 0.20906 | 0.00257 | 0.99796 |
| **PC15** | 0.16871 | 0.00167 | 0.99964 |
| **PC16** | 0.07866 | 0.00036 | 1.00000 |
| **PC17** | 1.956e-14 | 0.000e+00 | 1.000e+00 |

After looking the summary, we will decide to reduce the dimensionality when the Cumulative Proportion reach 90%, which is PC6. The reason why we would like to take the 90% proportion is because that number is considerably able to describe the overall customer information.

So initially we had 17 variables now this is reduced to 6 variables.

With these 6 variables we build eigen vectors for each component.

**2.6 Insights of KPI’s**

**Deriving new KPI’s**

1. Monthly Average Purchase
2. Monthly Cash Advance Amount
3. Purchases by type (one-off, installment)
4. Average amount per purchase
5. Cash advance transaction
6. Limit usage (balance to credit limit ratio)
7. Payments to min payments ratio

* Purchase by type:

purchase\_type variable is a categorical variable which as four types of categories in it. So, in the code section we have created a loop to extract values which belong to so and so categories, such as none, one\_off, both\_oneoff\_installment and installment.

The below table gives the count of values which comes under each category.

|  |  |
| --- | --- |
| both\_oneoff\_installment | 2774 |
| installment | 2260 |
| none | 2042 |
| one\_off | 1874 |
|  | 8950(total records) |

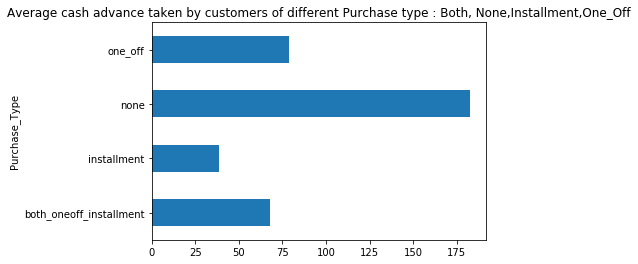
* Average amount per purchase and Cash advance transaction:

We are not going to create these two variables as these two variables gives a lot of missing values when computed.

Insights:

Deriving insights with respect to Purchase Type

* Customers who don't do either one-off or installment purchases take more cash on advance



frequency

* Customers with installment purchases are paying dues



frequency

* Customers with purchase type



**Chapter 3: Modelling**

**Clustering**

Our aim is to determine the clusters of customers, where each customer belongs to a either of a determined clusters.

**KMeans:**

After defining which dimensions that are going to used in Clustering, now we will use K-Means to determine how many Clusters do we need to divide Customers, which may represent their profile and hopefully we can determine what kind of treatment should be given to them.

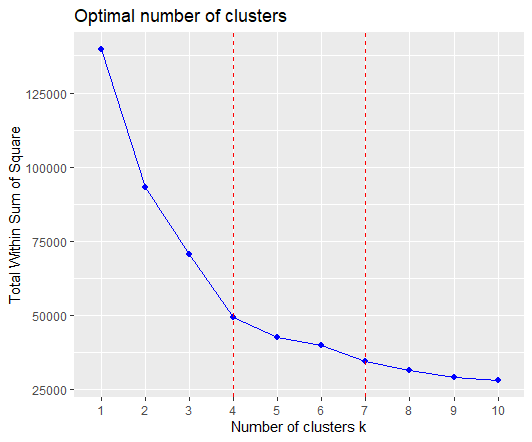
**3.1 Elbow method:**

The **elbow method** is **used** to determine the optimal number of clusters in k-means clustering.

The **elbow method** runs **k**-**means** clustering on the dataset for a range of values for **k** (say from 1-10) and then for each value of **k** computes an average score for all clusters.

In our modelling process we have used the elbow method to determine how many clusters we need to build our model going forward with the kmeans clustering algorithm.

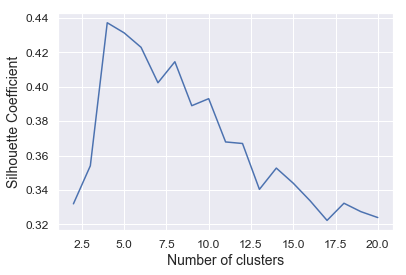
Below given is the elbow graph determining how many clusters we need to build.



* Based on the Elbow Method, we can say that the potential number of cluster (k) that may represent the customer segmentation is lies on 4-7.
* As we can see there is less difference between cluster 6 and 7 even though there is again a dip in the graph.
* We can ignore cluster 7 & continue examining cluster 4, 5 & 6.

**Silhouette Coefficient:**

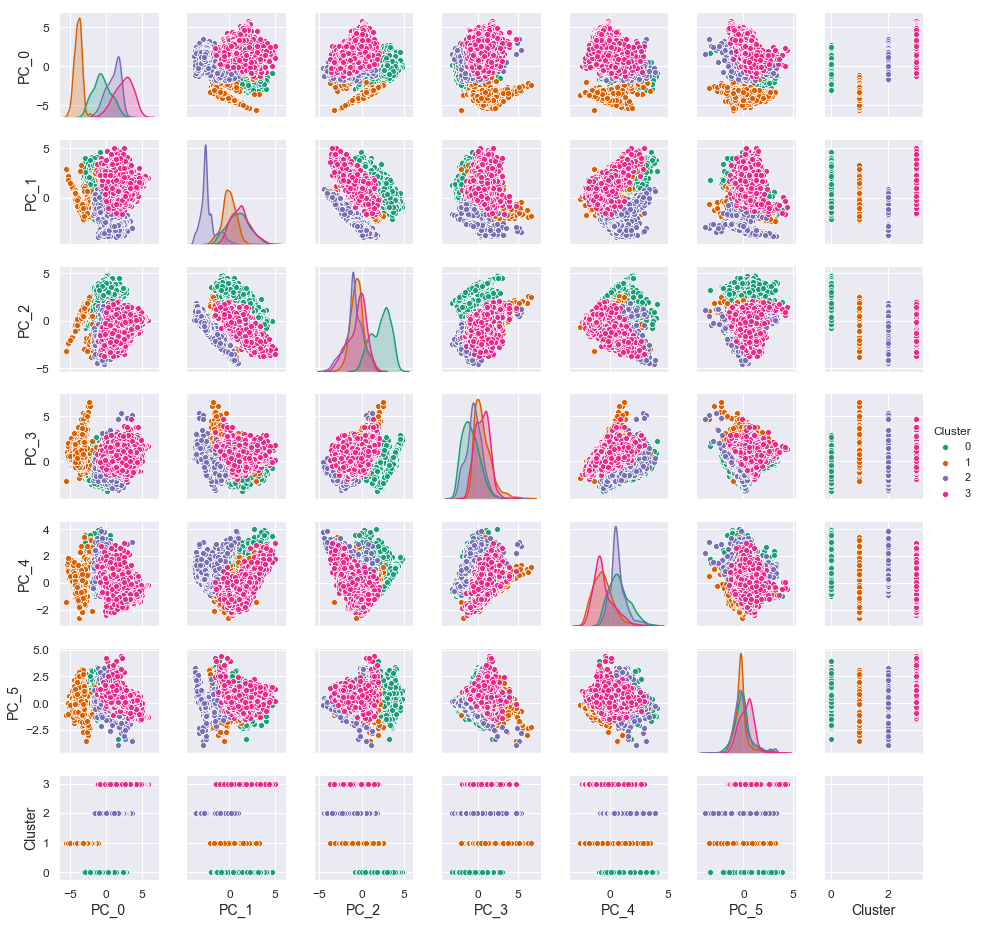
Compute silhouette information according to a given clustering in k clusters.

****

**Pair-plot:**

Pairwise relationship of components on the data

* It shows that first two components are able to identify clusters.



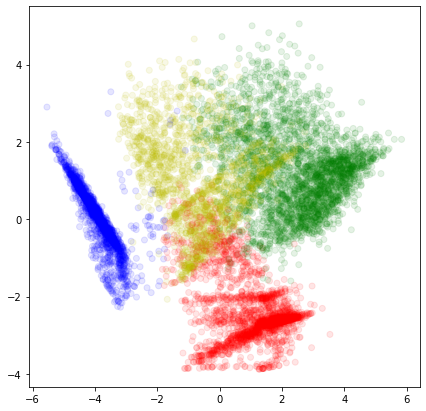
* It is very difficult to draw individual plot for cluster, so we will use pair plot which will provide us all graph in one shot.

**3.2 Clustering using Kmeans**

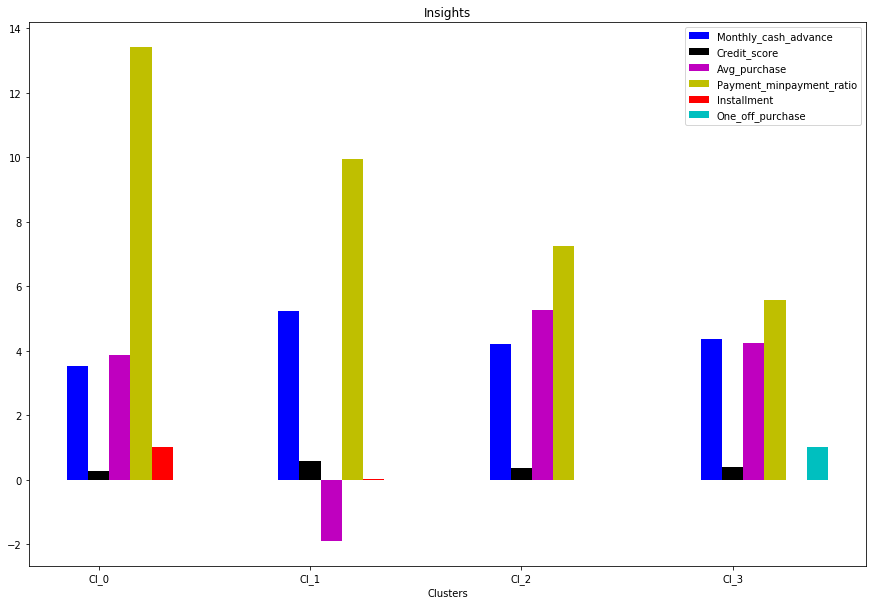
We will start with 4 clusters then we will go to 5 and 6.

**Analyzing behaviors with 4 clusters**

**Scatter plot:**

****

**Bar plot:**



Percentage of each cluster in the total customer base:

|  |  |  |
| --- | --- | --- |
|  | Size | Percentage |
| 1 | 2224 | 24.85 |
| 2 | 2088 | 23.33 |
| 3 | 2769 | 30.94 |
| 4 | 1869 | 20.90 |

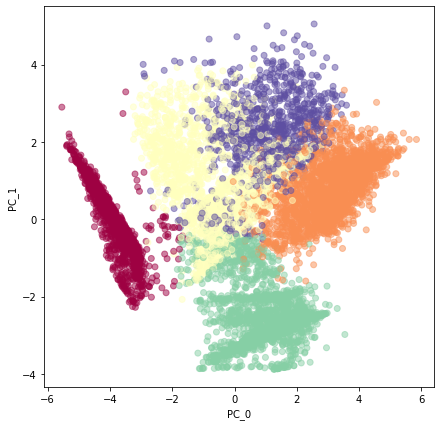
**Insights**

**Clusters are clearly distinguishing behavior with customers**

* Cluster 0 is the group of customers who have highest Payment Ratio & doing installment purchases, have comparatively good credit score. \* This group is about 25% of the total customer base.
* Cluster 1 is taking maximum advance\_cash and is paying comparatively less minimum payment and doing no purchase transaction. \* This group is about 23% of the total customer base\*
* Cluster 2 customers have maximum Purchase transaction and are paying dues and are doing least installment purchases. \* This group is about 31% of the total customer base \*
* Cluster 3 customers are doing maximum One\_Off transactions and least payment ratio. \* This group is about 21% of the total customer base \*

**Analyzing behaviors with 5 Clusters**

**Scatter plot:**

****

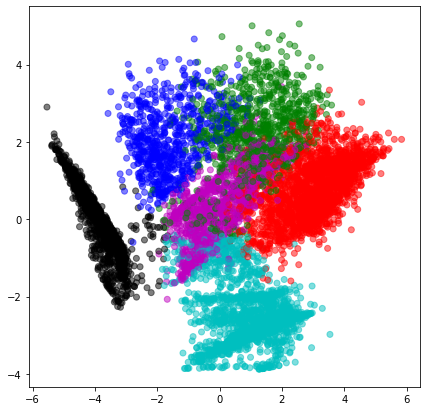
**Insights of 5 Clusters**

* We have a group of customers (cluster 1) having highest average purchases but there is Cluster 4 also having highest cash advance & second highest purchase behavior but their type of purchases is same.
* Cluster 1 and Cluster 4 are behaving similar in terms of Credit\_limit and have cash transactions is on higher side

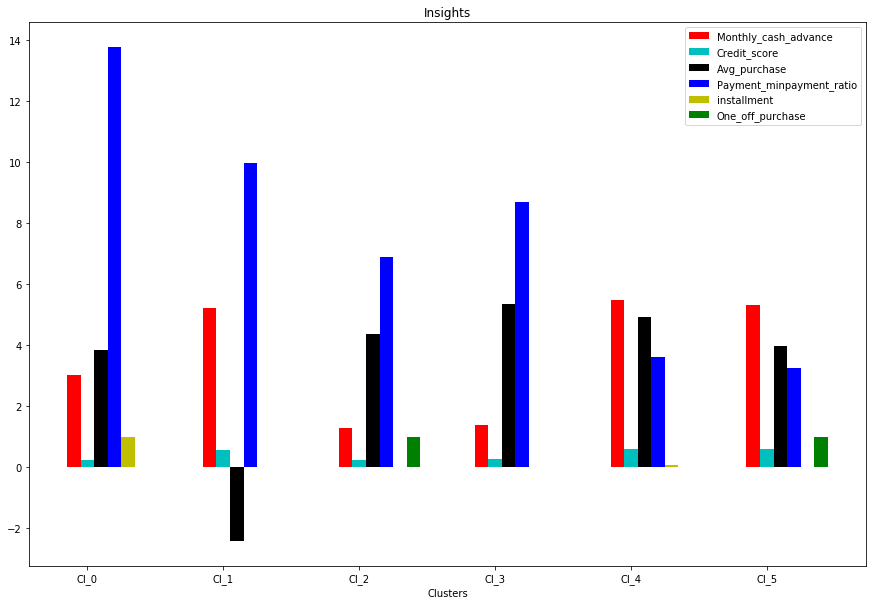
\*So we don't have quite distinguishable characteristics with 5 clusters\*

**Analysing behavior with 6 Clusters**

**Scatter plot:**

****

**Bar plot:**

****

**Insights of 6 Clusters**

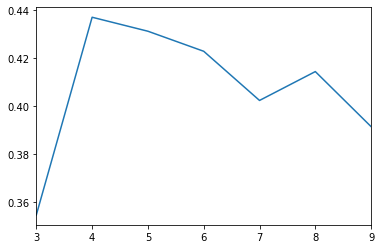
* Here also groups are overlapping.
* Cluster 4 and Cluster 5 are seem to behave same.

**3.3 Model Evaluation**

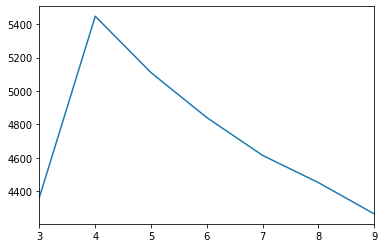
**Checking Performance Metrics for Kmeans**

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

**Silhouette Score**

****

**Calinski Harabasz Score**

****

* Performance metrics also suggest that K-means with 4 cluster is able to show distinguished characteristics of each cluster.

**\*Insights with 4 Clusters\***

* Cluster 0 is the group of customers who have highest Payment Ratio & doing installment purchases, have comparatively good credit score. \* This group is about 25% of the total customer base.
* Cluster 1 is taking maximum advance\_cash and is paying comparatively less minimum payment and doing no purchase transaction. \* This group is about 23% of the total customer base\*
* Cluster 2 customers have maximum Purchase transaction and are paying dues and are doing least installment purchases. \* This group is about 31% of the total customer base \*
* Cluster 3 customers are doing maximum One\_Off transactions and least payment ratio. \* This group is about 21% of the total customer base \*

**Suggested Marketing Strategy**

* Group 0

1. This group is performing best among all as customers are maintaining good credit score and paying dues on time. -- Giving rewards point will make them perform more purchases.

* b. Group 1

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

* c. Group 2

3. 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 /loyalty cards to increase transactions.

* d. Group 3

4. This group is having minimum paying ratio and using card for just one-off transactions (may be for utility bills only). This group seems to be risky group.

**Appendix**

**R\_Code:**

rm(list = ls())

setwd("G:/Tejas/R\_practice/Project1\_prac")

getwd()

credit\_data = read.csv("credit\_card\_data.csv", header = T, na.strings = c(" ", "", "NA"))

View(credit\_data)

str(credit\_data)

sum(is.na(credit\_data$MINIMUM\_PAYMENTS))

sum(is.na(credit\_data$CREDIT\_LIMIT))

names(credit\_data)

#Missing Value Analysis

#Create Dataframe with missing percentage

missing\_val = data.frame(apply(credit\_data, 2, function(x){sum(is.na(x))}))

View(missing\_val)

#Convert row names into column

missing\_val$Columns = row.names(missing\_val)

row.names(missing\_val) = NULL

#Rename the variable name

names(missing\_val)[1] = "Missing\_Values"

#Calculate Percentage of Missing Values

missing\_val$Missing\_Percentage = (missing\_val$Missing\_Values/nrow(credit\_data)) \* 100

#Write output result into disk

write.csv(missing\_val, "Missing\_perc.csv", row.names = F)

x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071", "Information", "MASS", "rpart", "gbm", "ROSE")

lapply(x, require, character.only = TRUE)

#Imputing Missing Values through Median Method

credit\_data[5204, 14]

credit\_data$CREDIT\_LIMIT[is.na(credit\_data$CREDIT\_LIMIT)] = median(credit\_data$CREDIT\_LIMIT, na.rm = T)

credit\_data[5204, 14]

#Same we will do for Minimum Payments

credit\_data[4, 16]

credit\_data$MINIMUM\_PAYMENTS[is.na(credit\_data$MINIMUM\_PAYMENTS)] = median(credit\_data$MINIMUM\_PAYMENTS, na.rm = T)

credit\_data[4, 16]

#So now we have Imputed all the Missing Values

sum(is.na(credit\_data))

#Deriving New KPI

# 1. Monthly average purchase and cash advance amount

credit\_data$Monthly\_Avg\_Purchase = credit\_data$PURCHASES/credit\_data$TENURE

head(credit\_data$Monthly\_Avg\_Purchase)

credit\_data$Monthly\_cash\_advance = credit\_data$CASH\_ADVANCE/credit\_data$TENURE

head(credit\_data$Monthly\_cash\_advance)

sum(credit\_data$ONEOFF\_PURCHASES == 0 & credit\_data$INSTALLMENTS\_PURCHASES == 0)

sum(credit\_data$ONEOFF\_PURCHASES > 0 & credit\_data$INSTALLMENTS\_PURCHASES > 0)

sum(credit\_data$ONEOFF\_PURCHASES > 0 & credit\_data$INSTALLMENTS\_PURCHASES == 0)

sum(credit\_data$ONEOFF\_PURCHASES == 0 & credit\_data$INSTALLMENTS\_PURCHASES > 0)

#As per the above details there are 4 types of purchase behaviour in the dataset. We will derive a categorical variable based on their behavior

library(dplyr)

#Function for creating categorical variable

Purchase = function(credit\_data)

{

sapply(1:nrow(credit\_data), function(i){

if(!credit\_data$ONEOFF\_PURCHASES[i] && !credit\_data$INSTALLMENTS\_PURCHASES[i])

return("none")

if(credit\_data$ONEOFF\_PURCHASES[i] && !credit\_data$INSTALLMENTS\_PURCHASES[i])

return("oneoff")

if(!credit\_data$ONEOFF\_PURCHASES[i] && credit\_data$INSTALLMENTS\_PURCHASES[i])

return("installment")

if(credit\_data$ONEOFF\_PURCHASES[i] & credit\_data$INSTALLMENTS\_PURCHASES[i])

return("both\_oneoff\_installment")

})

}

credit\_data$Purchase\_Type = Purchase(credit\_data)

#We will not be doing the third KPI since it gives a lot of na's

# 4. Usage Limit(balance to credit limit ratio)

#We will not be doing the third KPI since it gives a lot of na's

#Credit Card Utilization

credit\_data$Limit\_Usage = credit\_data$BALANCE/credit\_data$CREDIT\_LIMIT

head(credit\_data$Limit\_Usage)

dim(credit\_data)

# 5.Payments to minimum payments ratio

credit\_data$Paument\_minpay = credit\_data$PAYMENTS/credit\_data$MINIMUM\_PAYMENTS

head(credit\_data$Paument\_minpay)

library(tidyverse)

names(credit\_data)[names(credit\_data) == "Paument\_minpay"] = "Payment\_minpay"

str(credit\_data)

head(credit\_data$Payment\_minpay)

sum(is.na(credit\_data))

#Outlier Analysis

#Since there are variables having extreme values we will remove the outlier values

dim(credit\_data)

credit\_outlier = data.frame(credit\_data)

credit\_outlier = select(credit\_outlier, -c(CUST\_ID, Purchase\_Type))

dim((credit\_outlier))

str(credit\_outlier)

#As we see all the variables of type numeric

#We know that outlier analysis can only be applied on numeric variables and not on categorical

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

hist(credit\_outlier$BALANCE)

hist(credit\_outlier$PURCHASES, breaks = 20)

hist(credit\_outlier$CREDIT\_LIMIT)

#All the three histogram plots where right skewed

hist(credit\_outlier$TENURE)

#This is left skewed

#We will apply log-transformation to make the skewed data to approximately conform to normality

hist(log(credit\_outlier$BALANCE+1))

credit\_outlier\_log = data.frame(credit\_outlier)

#We are doing log-transformation to the entire dataset

credit\_outlier\_log = (log(credit\_outlier\_log+1))

view(credit\_outlier\_log)

library(freqdist)

head(freqdist(credit\_data$ONEOFF\_PURCHASES), 20)

col = c('BALANCE', 'PURCHASES','CASH\_ADVANCE','TENURE','PAYMENTS','MINIMUM\_PAYMENTS','PRC\_FULL\_PAYMENT','CREDIT\_LIMIT')

cre\_pre = data.frame(credit\_outlier\_log)

names(cre\_pre)

cre\_pre = select(credit\_outlier\_log, -c(BALANCE, PURCHASES, CASH\_ADVANCE, TENURE, PAYMENTS, MINIMUM\_PAYMENTS, PRC\_FULL\_PAYMENT, CREDIT\_LIMIT))

names(cre\_pre)

#Insight on the customer profiles

#Average payment\_minpayment ratio for each purchse type.

x = group\_by(credit\_data, Purchase\_Type)

y = summarise(x, mean\_pay\_minpay = mean(Payment\_minpay))

head(y)

#Plotting barplots

barplot(height = y$mean\_pay\_minpay, names.arg = y$Purchase\_Type, main = "Mean payment\_minpayment ratio for each purchse type", horiz = TRUE, col = "blue", border = "red", ylab = "Purchase\_Type", xlab = "Frequency")

#customers with installment purchases are paying dues

z = summarise(x, mean\_mon\_cash\_ad = mean(Monthly\_cash\_advance))

head(z)

barplot(height = z$mean\_mon\_cash\_ad, names.arg = y$Purchase\_Type, col = "blue", border = "red", main = "Average cash advance taken by customers of different Purchase type : Both, None,Installment,One\_Off", horiz = TRUE, ylab = "Purchase\_Type", xlab = "Frequency")

#Customers who don't do either one-off or installment purchases take more cash on advance

w = summarise(x , mean\_limit\_usage = mean(Limit\_Usage))

barplot(height = w$mean\_limit\_usage, names.arg = y$Purchase\_Type, ylab = "Purchase\_Type", xlab = "Frequency", col = "blue", border = "red", horiz = TRUE, main = "Customers with Limit Usage")

str(credit\_data)

unique(credit\_data$Purchase\_Type)

b = data.frame(credit\_data)

#Creating levels for the Purchase\_Type variable

library(dummies)

b$Purchase\_Type = as.factor(b$Purchase\_Type)

str(b)

b.new = dummy.data.frame(b, names = c("Purchase\_Type"), sep = "")

View(b.new)

names(b.new)[21] = "both\_oneoff\_installment"

names(b.new)[22] = "installment"

names(b.new)[23] = "none"

names(b.new)[24] = "oneoff"

View(b.new)

#Preparing Dataset for Machine Learning Algorithm

l = "Purchase\_Type"

dummy1 = select(filter(b.new), c("both\_oneoff\_installment", "installment", "none", "oneoff"))

View(dummy1)

cr\_dummy = data.frame(cre\_pre, dummy1)

is.null(cr\_dummy)

sum(is.null(cr\_dummy))

sum(is.na(cr\_dummy))

str(cr\_dummy)

library(corrplot)

install.packages("ggcorrplot")

ggcorrplot(cor(cr\_dummy))

library(ggcorrplot)

ggcorrplot(cor(cr\_dummy))

ggcorrplot(cor(cr\_dummy), hc.order = TRUE)

cor(cr\_dummy)

#Heat map shows that many features are co-related so applying dimensionality reduction will help negating multi-colinearity in data

install.packages("GGally")

library(ggplot2)

library(GGally)

ggcorr(cr\_dummy, label = T, label\_size = 3, label\_round = 2, hjust = 1, size = 3, color = "royalblue", layout.exp = 5, low = "dodgerblue", mid = "gray95", high = "red2", name = "Correlation Heatmap")

#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.

#Standardizing Data

#Making each data of same scale

cr\_scaled = scale(cr\_dummy)

head(cr\_scaled)

#Applying PCA

#Principal Component Analysis is used to reduce the features(Dimentionality reduction)

cr\_pca = prcomp(cr\_scaled)

cr\_pca

summary(cr\_pca)

#After looking the summary, we will decide to reduce the dimensionality when the Cumulative Proportion reach 90%

#Here we have 6\_components

#The reason why we would like to take the 90% proportion is because that number is considerably able to describe the overall customer information.

library(factoextra)

#Visualization on PCA

fviz\_pca\_biplot(cr\_pca, axes = c(1:2), col.var = "orange", col.ind = "rolayblue",

labelsize = 3) +

theme\_gray() +

labs(title = "Biplot of PC1 and PC2")

#Visualization on PCA

fviz\_pca\_biplot(cr\_pca, axes = c(1:2), col.var = "orange", col.ind = "red", labelsize = 3, select.ind = list(contrib = 5)) +

theme\_grey() +

labs(title = "Outlier of PC1 and PC2")

#As seen on the Outlier Plot, there are 5 Outliers with the following conditions:

credit\_outlier[c("1257", "1605", "3938", "6952", "8316"), ]

#1. Customers 1257, 1605 and 3938 have a very high PURCHASES and ONEOFF\_PUCHASES.

#2. Customer 6952: Have a very high Limit\_Usage.

#3. Customer 8316: Have a very high CASH\_ADVANCE

#variable Data Plot

fviz\_pca\_var(cr\_pca, col.var = "orange") +

theme\_gray()+

labs(title = "Variable Factor Map - PC\_1 and PC\_2")

View(cr\_dummy)

#Some Interpretations:

#1. As we can see in the Variable Data Plot, the Monthly\_Avg\_Purchase and PURCHASES\_TRX variables are close to the circle, indicating that those variables are highly contributed to the PC1 and PC2.

#2. On the other hand, the BALANCE\_FREQUENCY, one\_off and Payment\_minpay are variables that less contributed to PC1 and PC2.

#Dimentionality Reduction

view(cr\_pca$x)

credit\_final = cr\_pca$x[, 1:6]

View(credit\_final)

#Based on interepertations from above, we will decide to take only 6 dimensions and put to new dataset called credit\_final

head(credit\_final)

#K-Means Clustering

#Elbow Method

fviz\_nbclust(credit\_final,

kmeans, method = "wss",

linecolor = "blue") +

geom\_vline(xintercept = c(4, 7), linetype = 2, col = "red") +

theme\_gray()

#As we can see there is less difference between cluster 6 and 7 even though there is again a dip in the graph

#we can ignore cluster 7 and continue examining cluster 4, 5 and 6

#K-Means Clustering

for(i in 4:6){

set.seed(289)

model = kmeans(credit\_final, i)

print(paste("WSS of K", i, "=", model$tot.withinss))

print(paste("BSS Proportion of K", i, "=", model$betweenss/model$totss))

print(paste("Cluster Size of K", i, "="))

print(paste(model$size))

print(fviz\_cluster(model, credit\_outlier, palette = "Set1") +

theme\_gray())

}

credit\_km4 = kmeans(credit\_final, 4)

credit\_outlier$CLUSTER = credit\_km4$cluster

head(credit\_outlier, 10)

view(credit\_outlier)

#Performance metrics also suggest that K-means with 4 cluster is able to show distinguished characteristics of each cluster

#so we stop with the cluster 4

savehistory(file = "Project1\_R.Rhistory")

**Python Code:**

#Import packages

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

#Load the csv dataset

credit\_data = pd.read\_csv("credit\_card\_data.csv")

credit\_data.head()

# Information about dataset

credit\_data.info()

credit\_data.shape

#Initial descriptive analysis of data

credit\_data.describe()

#Find the total number of missing values in the dataframe

print("Missing values: ", credit\_data.isnull().sum().values.sum(), "\n")

#Printing total number of unique values in the dataframe

print("Unique Values:\n", credit\_data.nunique())

# Missing Value Analysis

credit\_data.isnull().sum()

credit\_data['CREDIT\_LIMIT'].count()

credit\_data[credit\_data['CREDIT\_LIMIT'].isnull()]

#Impute missing values with median

credit\_data['CREDIT\_LIMIT'].fillna(credit\_data['CREDIT\_LIMIT'].median(), inplace=True)

credit\_data['CREDIT\_LIMIT'].count()

credit\_data.iloc[5203, 13]

min\_pay = credit\_data[credit\_data['MINIMUM\_PAYMENTS'].isnull()]

min\_pay.head()

credit\_data['MINIMUM\_PAYMENTS'].fillna(credit\_data['MINIMUM\_PAYMENTS'].median(), inplace=True)

credit\_data['MINIMUM\_PAYMENTS'].count()

credit\_data.iloc[3, 15]

credit\_data.iloc[45, 15]

credit\_data.isnull().sum()

# Deriving New KPI

# 1.Monthly average purchase and cash advance amount

#Monthly Average Purchase

credit\_data['Monthly\_Avg\_Purchase'] = credit\_data['PURCHASES']/credit\_data['TENURE']

print(credit\_data['Monthly\_Avg\_Purchase'].head(), '\n', credit\_data['TENURE'].head(), '\n', credit\_data['PURCHASES'].head())

#Monthly Cash Advance Amount

credit\_data['Monthly\_cash\_advance'] = credit\_data['CASH\_ADVANCE']/credit\_data['TENURE']

credit\_data.shape

credit\_data[credit\_data['ONEOFF\_PURCHASES'] == 0]['ONEOFF\_PURCHASES'].count()

# 2.Purchases by type (one-off, instalments)

#Find what type of purchases are made on credit card

credit\_data.iloc[:, [4, 5]]

# Details for both the columns

credit\_data[(credit\_data.iloc[:, 4]==0) & (credit\_data.iloc[:, 5]==0)].shape

credit\_data[(credit\_data.iloc[:, 4]>0) & (credit\_data.iloc[:, 5]>0)].shape

credit\_data[(credit\_data.iloc[:, 4]>0) & (credit\_data.iloc[:, 5]==0)].shape

credit\_data[(credit\_data.iloc[:, 4]==0) & (credit\_data.iloc[:, 5]>0)].shape

#As per the above details there are 4 types of purchase behaviour in the dataset.

#We will derive a categorical variable based on their behavior

def purchase(credit\_data):

if(credit\_data['ONEOFF\_PURCHASES']==0) & (credit\_data['INSTALLMENTS\_PURCHASES']==0):

return 'none'

if(credit\_data['ONEOFF\_PURCHASES']>0) & (credit\_data['INSTALLMENTS\_PURCHASES']>0):

return 'both\_oneoff\_installment'

if(credit\_data['ONEOFF\_PURCHASES']>0) & (credit\_data['INSTALLMENTS\_PURCHASES']==0):

return 'one\_off'

if(credit\_data['ONEOFF\_PURCHASES']==0) & (credit\_data['INSTALLMENTS\_PURCHASES']>0):

return 'installment'

credit\_data['Purchase\_Type'] = credit\_data.apply(purchase, axis=1)

credit\_data['Purchase\_Type'].value\_counts()

credit\_data.shape

# 3. Average amount per purchase and Cash advance transaction

#Average amount per purchase

credit\_data['Average\_amt\_per\_purchase'] = credit\_data.iloc[:, 3]/credit\_data.iloc[:, 12]

credit\_data['Avg\_cash\_advance'] = credit\_data.iloc[:, 6]/credit\_data.iloc[:, 11]

credit\_data.shape

credit\_data.head(10)

#On calculating Average amount per purchase and Cash advance transaction we get a lot of Null values(NaN),

#So we gonna drop both the variables from the dataset

credit\_data.drop(['Average\_amt\_per\_purchase'], axis=1)

credit\_data.drop(['Avg\_cash\_advance'], axis=1)

credit\_data = credit\_data.drop(['Average\_amt\_per\_purchase', 'Avg\_cash\_advance'], axis=1)

credit\_data.shape

credit\_data.isnull().sum()

# 4.Limit Usage (balance to credit limit ratio)

#Credit Card Utilization

credit\_data['Limit\_Usage'] = credit\_data.apply(lambda x: x['BALANCE']/x['CREDIT\_LIMIT'], axis=1)

credit\_data['Limit\_Usage'].head(5)

credit\_data.shape

# 5.Payments to minimum payments ratio

credit\_data['Payment\_minpay'] = credit\_data.apply(lambda x: x['PAYMENTS']/x['MINIMUM\_PAYMENTS'], axis = 1)

credit\_data['Payment\_minpay'].head(10)

credit\_data.shape

credit\_data.isnull().sum()

# Outlier Analysis

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

credit\_outlier = credit\_data.drop(['CUST\_ID', 'Purchase\_Type'], axis=1).applymap(lambda x:np.log(x+1))

credit\_outlier.head(10)

credit\_outlier.shape

credit\_outlier.describe()

col=['BALANCE','PURCHASES','CASH\_ADVANCE','TENURE','PAYMENTS','MINIMUM\_PAYMENTS','PRC\_FULL\_PAYMENT','CREDIT\_LIMIT']

cr\_pre=credit\_outlier[[x for x in credit\_outlier.columns if x not in col ]]

cr\_pre.columns

credit\_outlier.columns

# Insight on the customer profiles

#Average payment\_minpayment ratio for each purchse type.

x=credit\_data.groupby('Purchase\_Type').apply(lambda x: np.mean(x['Payment\_minpay']))

type(x)

x.values

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')

#customers with installment purchases are paying dues

credit\_data.groupby('Purchase\_Type').apply(lambda x: np.mean(x['Monthly\_cash\_advance'])).plot.barh()

plt.title('Average cash advance taken by customers of different Purchase type : Both, None,Installment,One\_Off')

#Customers who don't do either one-off or installment purchases take more cash on advance

credit\_data.groupby('Purchase\_Type').apply(lambda x: np.mean(x['Limit\_Usage'])).plot.barh()

credit\_original = pd.concat([credit\_data, pd.get\_dummies(credit\_data['Purchase\_Type'])], axis=1)

credit\_original.head(10)

# Preparing dataset for Machine Learning Algorithm

#There are a few categorical data which we need to convert with dummy creations

# Creating dummies for categorical variable

cr\_pre['Purchase\_Type'] = credit\_data.loc[:, 'Purchase\_Type']

pd.get\_dummies(cr\_pre['Purchase\_Type'])

#Now merge the created dummy with the original dataframe

cr\_dummy = pd.concat([cr\_pre, pd.get\_dummies(cr\_pre['Purchase\_Type'])], axis=1)

cr\_dummy.isnull().any()

l = ['Purchase\_Type']

#We are removing the Payment\_Type variable

cr\_dummy = cr\_dummy.drop(l, axis=1)

cr\_dummy.isnull().any()

cr\_dummy.shape

cr\_dummy.info()

cr\_dummy.head(10)

#Plot a correlation heatmap

sns.heatmap(cr\_dummy.corr())

#Heat map shows that many features are co-related so applying dimensionality reduction will help negating multi-colinearity in data

#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.

# Standardizing Data

#Making each data of same scale

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

cr\_scaled = sc.fit\_transform(cr\_dummy)

cr\_scaled

# Applying PCA

#Principal Component Analysis is used to reduce the features(Dimentionality reduction)

from sklearn.decomposition import PCA

cr\_dummy.shape

#In PCA the n component is the total no of features

pc = PCA(n\_components=17)

cr\_pca = pc.fit(cr\_scaled)

#Let's check if we take 17 component then how much varience it explain. Ideally it should be 1 i.e 100%

sum(cr\_pca.explained\_variance\_ratio\_)

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

pd.Series(var\_ratio).plot()

Since 6 components are explaining about 90% variance so we select 6 components

pc = PCA(n\_components=6)

p = pc.fit(cr\_scaled)

cr\_scaled.shape

p.explained\_variance\_

np.sum(p.explained\_variance\_)

pd.Series(var\_ratio).plot()

pc\_final = PCA(n\_components=6).fit(cr\_scaled)

reduced\_cr = pc\_final.fit\_transform(cr\_scaled)

dd = pd.DataFrame(reduced\_cr)

dd.head()

#So Initially we had 17 variables now we have 6 so the variables go reduced

dd.shape

col\_list = cr\_dummy.columns

col\_list

pd.DataFrame(pc\_final.components\_.T, columns=['PC\_' +str(i) for i in range(6)], index = col\_list)

#So above data gave us eigen vector for each component, if we had all eigen vector value very small we can remove those variable but in our case its not

#Factor Analysis : variance explained by each component

pd.Series(pc\_final.explained\_variance\_ratio\_, index=['PC\_' +str(i) for i in range(6)])

# 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 sklearn.cluster import KMeans

km\_cls4 = KMeans(n\_clusters=4, random\_state=123)

km\_cls4.fit(reduced\_cr)

km\_cls4.labels\_

pd.Series(km\_cls4.labels\_).value\_counts()

#Hear we donot have known k value so we will find the K. To do that we need to take a cluster range between 1 and 21.

# Identifying Cluster Error

cluster\_range = range(1, 21)

cluster\_errors = []

for num\_clusters in cluster\_range:

clusters = KMeans(num\_clusters)

clusters.fit(reduced\_cr)

cluster\_errors.append(clusters.inertia\_)

#clusters.inertia\_ is actually cluster error here

clusters\_df = pd.DataFrame({"num\_clusters": cluster\_range, "cluster\_errors": cluster\_errors})

clusters\_df[0:21]

#Elbow Method

# allow plots to appear in the notebook

%matplotlib inline

import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))

plt.plot(clusters\_df.num\_clusters, clusters\_df.cluster\_errors, marker = "o")

#From the above elbow method we find the elbow range here as 4, 5, 6

# Shilouette Coeffiecient

from sklearn import metrics

# 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\_))

scores

#Plot the results

plt.plot(k\_range, scores)

plt.xlabel('Number of Clusters')

plt.ylabel('Silhouette Coefficient')

plt.grid(True)

color\_map = {0 : 'r', 1 : 'b', 2 : 'g', 3 : 'y'}

# Here l is Purchase\_Type(Behavior) we had 4 types

label\_color = [color\_map[l] for l in km\_cls4.labels\_]

plt.figure(figsize=(7, 7))

plt.scatter(reduced\_cr[:, 0], reduced\_cr[:, 1], c = label\_color, cmap = 'Spectral', alpha = 0.1)

#It is very difficult to draw individual plot for cluster, so we will use pair plot which will provide us all graph in one shot. To do that we need to take following steps

df\_pair\_plot = pd.DataFrame(reduced\_cr, columns=['PC\_' +str(i) for i in range(6)])

#Add cluster column in the dataframe which specifies which cluster it belongs to

df\_pair\_plot['Cluster'] = km\_cls4.labels\_

df\_pair\_plot.head()

#pairwise relationship of components on the data

sns.pairplot(df\_pair\_plot, hue='Cluster', palette= 'Dark2', diag\_kind='kde', height=1.85)

#\*It shows that first two components are able to indentify clusters\*\*\*

\*\*\*Now we have done here with Principal Component Analysis now we need to bring our original dataframe and we will merge the cluster with them.\*\*\*

\*\*\*To interprate result we need to use our dataframe\*\*\*

# Key performace variable selection . here i am taking varibales which we will use in derving new KPI.

#We can take all 17 variables but it will be difficult to interprate.So are are selecting less no of variables.

col\_kpi = ['PURCHASES\_TRX','Monthly\_Avg\_Purchase','Monthly\_cash\_advance','Limit\_Usage','CASH\_ADVANCE\_TRX', 'Payment\_minpay','both\_oneoff\_installment','installment','one\_off','none','CREDIT\_LIMIT']

cr\_pre.describe()

#Concatenating labels found through KMeans with data

clusters\_df\_4 = pd.concat([credit\_original[col\_kpi], pd.Series(km\_cls4.labels\_, name = 'Cluster\_4')], axis=1)

clusters\_df\_4.head()

# Mean value gives a good indication of the distribution of data. So we find mean value for each variable for each cluster

cluster\_4 = clusters\_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 = 'k', label = 'Credit\_score', width = bar\_width)

b3 = plt.bar(index+2\*bar\_width, purchase, color = 'm', label = 'Avg\_purchase', width = bar\_width)

b4 = plt.bar(index+3\*bar\_width, payment, color = 'y', 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 = 'c', label = 'One\_off\_purchase', width = bar\_width)

plt.xlabel("Clusters")

plt.title("Insights")

plt.xticks(index + bar\_width, ('Cl\_0', 'Cl\_1', 'Cl\_2', 'Cl\_3'))

plt.legend()

# Percentage of each cluster in the total customer base

s = clusters\_df\_4.groupby('Cluster\_4').apply(lambda x: x['Cluster\_4'].value\_counts())

print(s), '\n'

per = pd.Series((s.values.astype('float')/ clusters\_df\_4.shape[0]) \* 100, name = 'Percentage')

print("Cluster -4"), '\n'

print(pd.concat([pd.Series(s.values, name = 'Size'), per], axis = 1))

#Insights

#Clusters are clearly distinguishing behaviour within clustomers

1. Cluster 0 is the group of customers who have highest Payment Ratio and doing installment purchases, have comparatively good credit score. \* This group is about 25% of the total customer base \*
2. Cluster 1 is taking maximum advance\_cash and is paying comparatively less minimum payment and doing no purchase transaction. \* This group is about 23% of the total customer base \*
3. Cluster 2 customers have maximum Purchase transaction and are paying dues and are doing least installment purchases. \* This group is about 31% of the total customer base \*
4. Cluster 3 customers are doing maximum One\_Off transactions and least payment ratio. \* This group is about 21% of the total customer base \*

# Analyzing behaviors with 5 Clusters

km\_cls5 = KMeans(n\_clusters=5, random\_state=123)

km\_cls5 = km\_cls5.fit(reduced\_cr)

km\_cls5.labels\_

pd.Series(km\_cls5.labels\_).value\_counts()

plt.figure(figsize=(7, 7))

plt.scatter(reduced\_cr[:, 0], reduced\_cr[:, 1], c = km\_cls5.labels\_, cmap = 'Spectral', alpha=0.5)

plt.xlabel('PC\_0')

plt.ylabel('PC\_1')

clusters\_df\_5 = pd.concat([credit\_original[col\_kpi], pd.Series(km\_cls5.labels\_, name = 'Cluster\_5')], axis = 1)

# Finding Mean of features for each cluster

clusters\_df\_5.groupby('Cluster\_5')\

.apply(lambda x: x[col\_kpi].mean()).T

#Insights of 5 Clusters

#1. We have a group of customers (cluster 1) having highest average purchases but there is Cluster 4 also having highest cash advance & second highest purchase behavior but their type of purchases are same.

#2. Cluster 1 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

no\_cls5 = clusters\_df\_5.groupby('Cluster\_5').apply(lambda x: x['Cluster\_5'].value\_counts())

print ("Cluster -5"),'\n'

percentage\_5=pd.Series((no\_cls5.values.astype('float')/ clusters\_df\_5.shape[0])\*100, name = 'Percentage')

print (pd.concat([pd.Series(no\_cls5.values, name = 'Size'), percentage\_5], axis=1))

# Analysing behavior with 6 Clusters

km\_cls6 = KMeans(n\_clusters=6).fit(reduced\_cr)

km\_cls6.labels\_

color\_map = {0:'c', 1:'k', 2:'m', 3:'r', 4:'g', 5:'b'}

label\_color = [color\_map[l] for l in km\_cls6.labels\_]

plt.figure(figsize=(7, 7))

plt.scatter(reduced\_cr[:, 0], reduced\_cr[:, 1], c = label\_color, cmap='Spectral', alpha=0.5)

clusters\_df\_6 = pd.concat([credit\_original[col\_kpi], pd.Series(km\_cls6.labels\_, name = 'Cluster\_6')], axis=1)

no\_cls6 = clusters\_df\_6.groupby('Cluster\_6').apply(lambda x: x[col\_kpi].mean()).T

no\_cls6

fig,ax = plt.subplots(figsize = (15, 10))

index = np.arange(len(no\_cls6.columns))

cash\_advance = np.log(no\_cls6.loc['Monthly\_cash\_advance',:].values)

credit\_score = (no\_cls6.loc['Limit\_Usage',:].values)

purchase = np.log(no\_cls6.loc['Monthly\_Avg\_Purchase',:].values)

payment = no\_cls6.loc['Payment\_minpay',:].values

installment = no\_cls6.loc['installment',:].values

one\_off = no\_cls6.loc['one\_off',:].values

bar\_width = .10

b1 = plt.bar(index, cash\_advance, color='r', label = 'Monthly\_cash\_advance', width=bar\_width)

b2 = plt.bar(index+bar\_width, credit\_score, color = 'c', 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 = 'b', label = 'Payment\_minpayment\_ratio', width = bar\_width)

b5 = plt.bar(index+4\*bar\_width, installment, color = 'y', 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('Clusters')

plt.title('Insights')

plt.xticks(index + bar\_width, ('Cl\_0', 'Cl\_1', 'Cl\_2', 'Cl\_3', 'Cl\_4', 'Cl\_5'))

plt.legend()

# Insights of 6 Clusters

#1. Here also groups are overlapping

#2. Cluster 4 and Cluster 5 are seem to behave same

# Checking Performance Metrics for KMeans

#\*\*\*I am validating performance with 2 metrics Calinski Harabaz and Silhouette score\*\*\*

from sklearn.metrics import calinski\_harabasz\_score, silhouette\_score

score = {}

score\_calinski = {}

for n in range(3, 10):

km\_score = KMeans(n\_clusters = n)

km\_score.fit(reduced\_cr)

score\_calinski[n] = calinski\_harabasz\_score(reduced\_cr, km\_score.labels\_)

score[n] = silhouette\_score(reduced\_cr, km\_score.labels\_)

pd.Series(score).plot()

pd.Series(score\_calinski).plot()

# Performance metrics also suggest that K-means with 4 cluster is able to show distinguished characteristics of each cluster.