

Project Name
Credit Card Segmentation

Prepared by:
Kriti Upadhyay

Submission Date: 30.11.2019

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.

Data Set provided:

[credit-card-data.csv](#)

Data set contains below 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

Derived KPIs from the Data Set:

1. PAYMENTS_TO_MINIMUM_PAYMENT_RATIO,
2. AVERAGE_AMOUNT_PER_PURCHASE,
3. MONTHLY_INSTALLMENTS_PURCHASES,
4. MONTHLY_CASH_ADVANCE_TRANSACTION,
5. MONTHLY_CASH_ADVANCE,
6. MONTHLY_ONEOFF_PURCHASES,
7. LIMIT_USAGE,
8. MONTHLY_AVERAGE_PURCHASE

PART: 1: Implementation by R

Procedure:

Step-1

Data Loading

Data Set is taken from the link provided and then loaded into the R environment for performing Data pre-processing techniques which are the necessary steps in the data science to organize data into proper format before feeding it to the model, because every Model accepts the data in a specific format form only.

Following commands used to perform above task:

```
#Removing RAM
```

```
rm(list=ls())
```

```
#Setting working directory
```

```
setwd("C://Users//akhil//Desktop//Kriti Data//EDWISOR_invoice//data science study material//Project")
```

```
#Checking the working directory
```

```
getwd()
```

```
#Loading csv data into Credit_Card_Data object
```

```
Credit_Card_Data=read.csv("credit-card-data.csv",header = TRUE)
```

```
#Opening the loaded data
```

```
Credit_Card_Data
```

Step-2

Missing Value Analysis

After loading the data, the first step performed is missing value analysis where we compute the missing values or information present in the data set. Here the missing value analysis is computed by using the method of KNN Imputation, because it computes the missing values by using Euclidean Distance formula and this method gives accurate results mostly than compared to mean median method.

Following code is being used:

```
#loading libraries for visualization and preprocessing of data
```

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

```
install.packages(x)
```

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

```
rm(x)
```

MISSING VALUE ANALYSIS

#Missing value detection in each variable of the dataset by using apply function that uses Sum(is.na()) function as an argument and will return total count of the missing values present in each variable of the dataset.

```
Variable_with_Missing_VAlues=data.frame(apply(Credit_Card_Data,2, function(x)
{sum(is.na(x))}))
Variable_with_Missing_Values
```

#Only two variables in the dataset contain missing values they are CREDIT_LIMIT and MINIMUM_PAYMENTS

#Computing missing values of the variable

#Calculating the missing value percentage

```
MINIMUM_PAYMENTS_Missing_Per=(sum(is.na(Credit_Card_Data$MINIMUM_PAYMENTS))/nr
ow(Credit_Card_Data))*100
```

MINIMUM_PAYMENTS_Missing_Per

```
CREDIT_LIMIT_Missing_Per=(sum(is.na(Credit_Card_Data$CREDIT_LIMIT))/nrow(Credit_Card_D
ata))*100
```

CREDIT_LIMIT_Missing_Per

#Since the missing value percentage of both the variable is less than 30% so we cannot ignore the missing values.

#Deleting some values manually and finding missing values to find best method for missing value analysis

#Finding missing value using mean:

```
#Credit_Card_Data[3,16]=NA
```

```
#Credit_Card_Data$MINIMUM_PAYMENTS
```

```
#Credit_Card_Data$MINIMUM_PAYMENTS[is.na(Credit_Card_Data$MINIMUM_PAYMENTS)]=m
ean(Credit_Card_Data$MINIMUM_PAYMENTS,na.rm=T)
```

```
#Credit_Card_Data
```

```
# original value= 627.28479
```

```
# missing value computed by mean method= 824.23398
```

#finding missing value using KNN imputation method:

```
#Credit_Card_Data[3,16]=NA
```

```
#Credit_Card_Data$MINIMUM_PAYMENTS
```

```
#Credit_Card_Data=knnImputation(Credit_Card_Data,k=5)
```

```
#Credit_Card_Data$MINIMUM_PAYMENTS
```

```
#Original value= 627.28479
```

```
#missing value computed by KNN imputation= 689.7210
```

```
#On comparing missing values computed by the KNN imputation is more closer to the actual
value so in this case KNN imputation is the best method for computing missing values
#Computing missing values for the variables CREDIT_LIMIT and MINIMUM_PAYMENTS using
KNN imputation method
```

```
Credit_Card_Data=knnImputation(Credit_Card_Data,k=5)
```

```
#ReCheck for missing values
sum(is.na(Credit_Card_Data))
```

Step-3:

Outliers Detection

Once, the missing values are computed, the next step is to detect and remove outliers present in the data set. These outliers are nothing but the extreme values present in the data set.

In this project, missing values are detected using Box-plot method because it gives the graphical representation of the presence of the outliers in the data set and completely distinguish the outliers present in the lower and upper fence of the box plot.

Code used for outlier detection:

```
##### OUTLIER ANALYSIS #####
##### DETECTION OF OUTLIERS METHOD #####
```

```
#Selecting only numeric variables
Numeric_index= sapply(Credit_Card_Data,is.numeric)
Numeric_data=Credit_Card_Data[,Numeric_index]
#Variables names containing numeric data
Cnames=colnames(Numeric_data)
Cnames

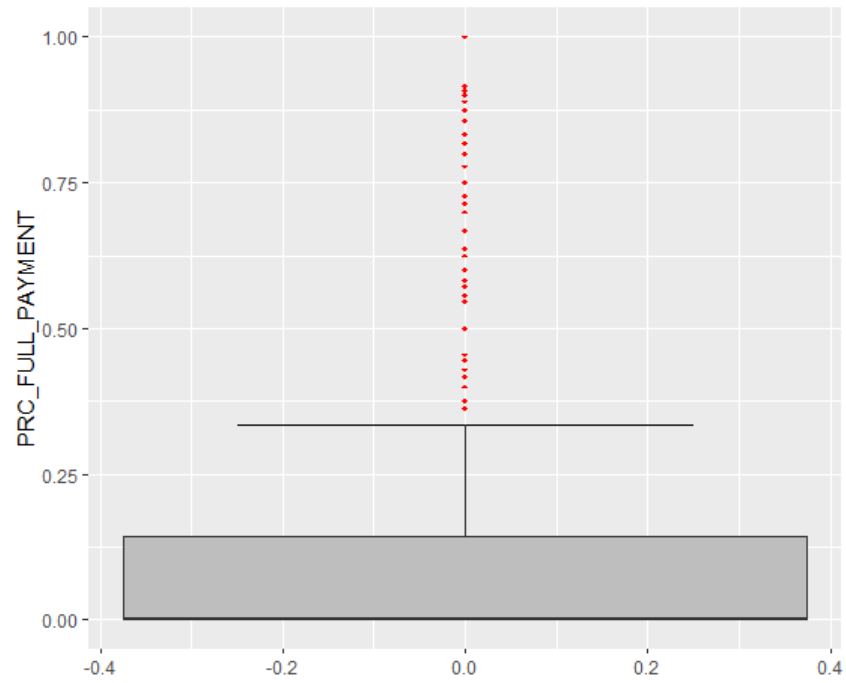
# Outlier check
for(i in 1:length(Cnames))
{ assign(paste0("gn",i),ggplot(aes_string(y=Cnames[i])),data=subset(Credit_Card_Data))
  +stat_boxplot(geom="errorbar",width=0.5)
  +geom_boxplot(outlier.colour = "red",fill="grey",outlier.shape = 18,outlier.size = 1,notch
=FALSE)
  +theme(legend.position = "bottom")+labs(y=Cnames[i])
  +ggtitle(paste("BOX PLOT FOR",Cnames[i]))
}

#Displaying boxplot for outlier check
gridExtra::grid.arrange(gn1)
```

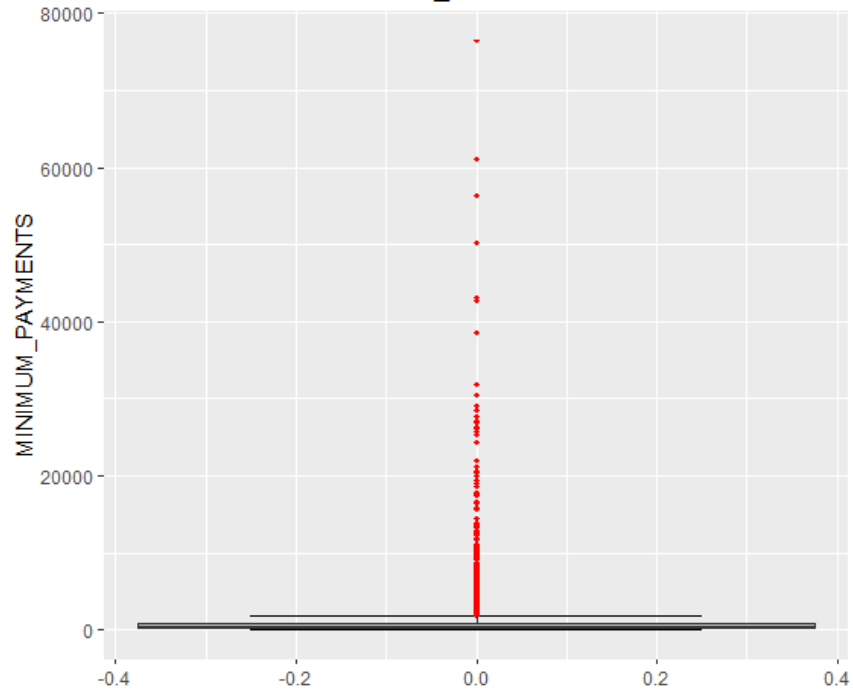
```
gridExtra::grid.arrange(gn2)
gridExtra::grid.arrange(gn3)
gridExtra::grid.arrange(gn4)
gridExtra::grid.arrange(gn5)
gridExtra::grid.arrange(gn6)
gridExtra::grid.arrange(gn7)
gridExtra::grid.arrange(gn8)
gridExtra::grid.arrange(gn9)
gridExtra::grid.arrange(gn10)
gridExtra::grid.arrange(gn11)
gridExtra::grid.arrange(gn12)
gridExtra::grid.arrange(gn13)
gridExtra::grid.arrange(gn14)
gridExtra::grid.arrange(gn15)
gridExtra::grid.arrange(gn16)
gridExtra::grid.arrange(gn17)
```

Detection of the outliers present in the variables of the given data set using Box plots are shown below:

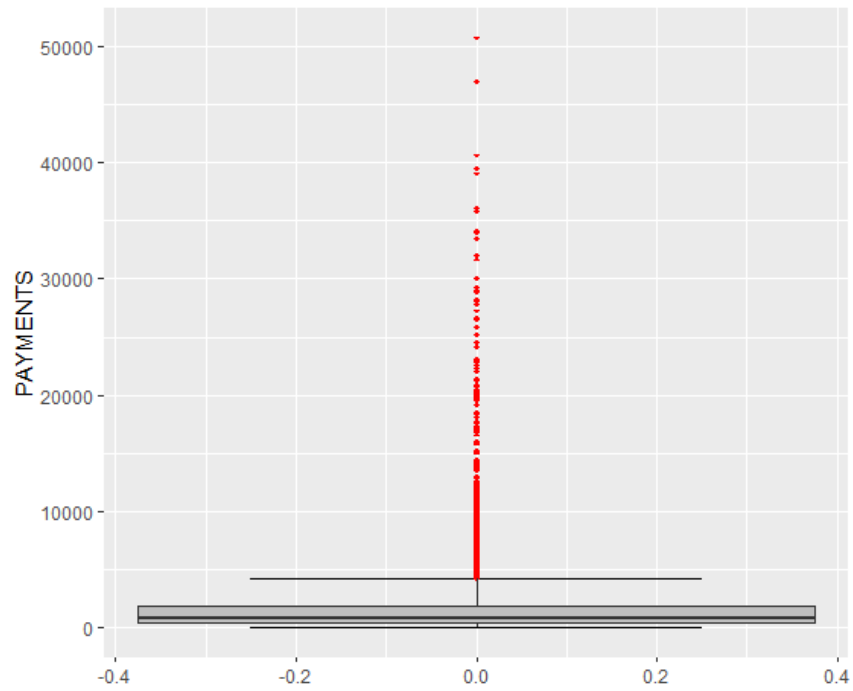
BOX PLOT FOR PRC_FULL_PAYMENT



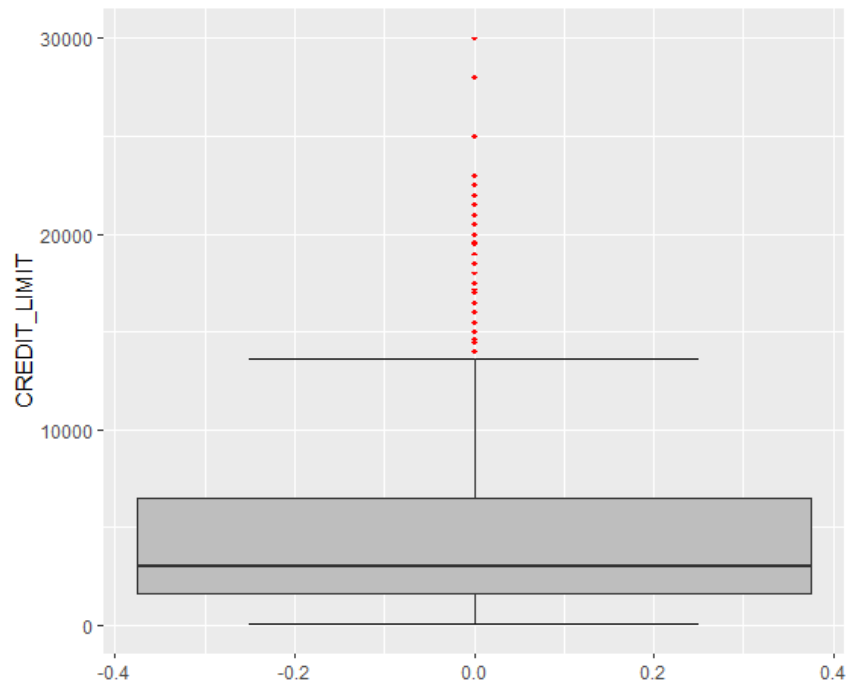
BOX PLOT FOR MINIMUM_PAYMENTS



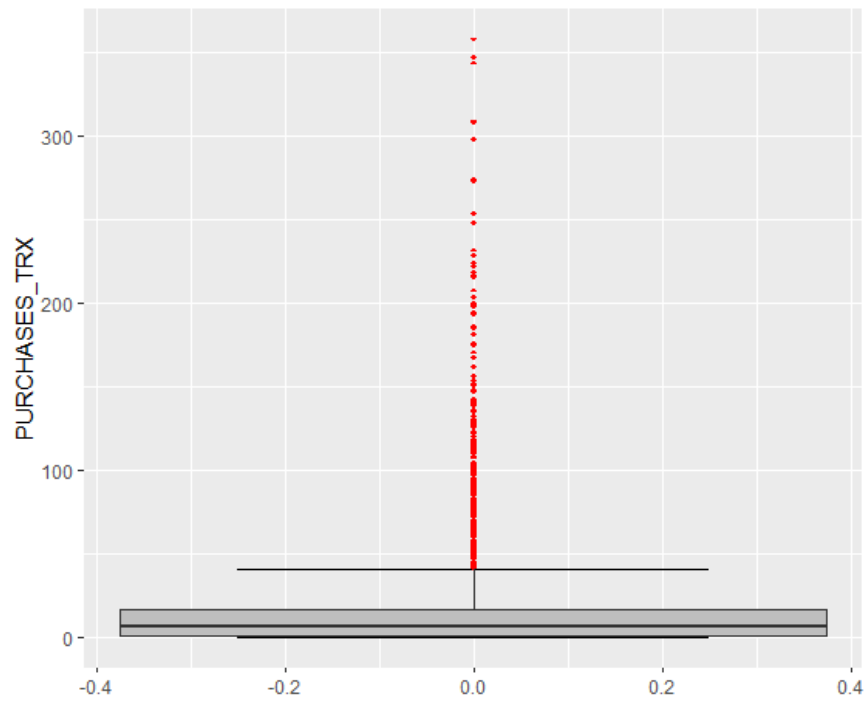
BOX PLOT FOR PAYMENTS



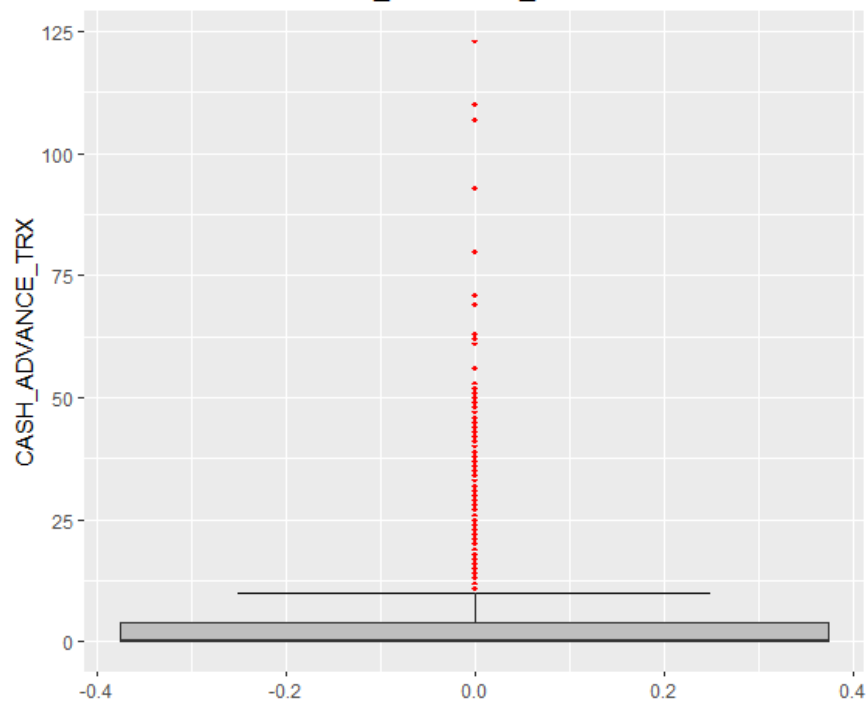
BOX PLOT FOR CREDIT_LIMIT

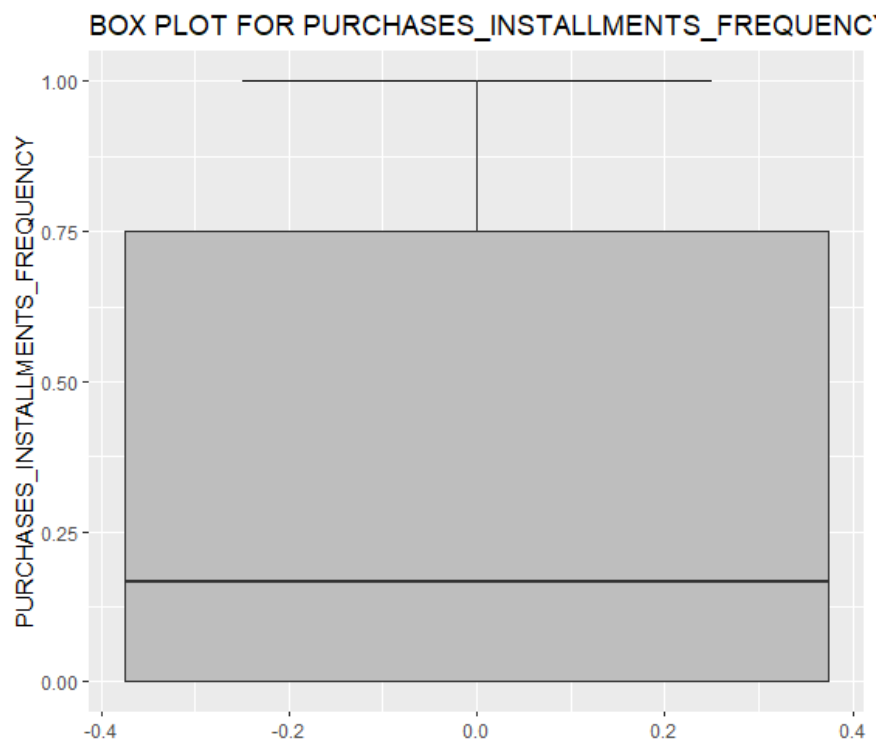
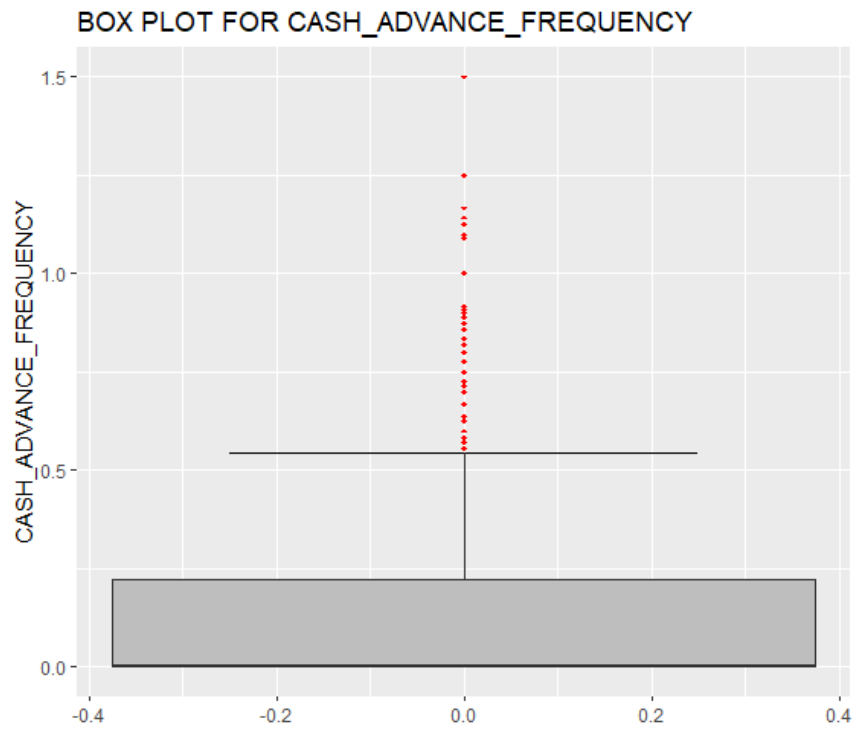


BOX PLOT FOR PURCHASES_TRX

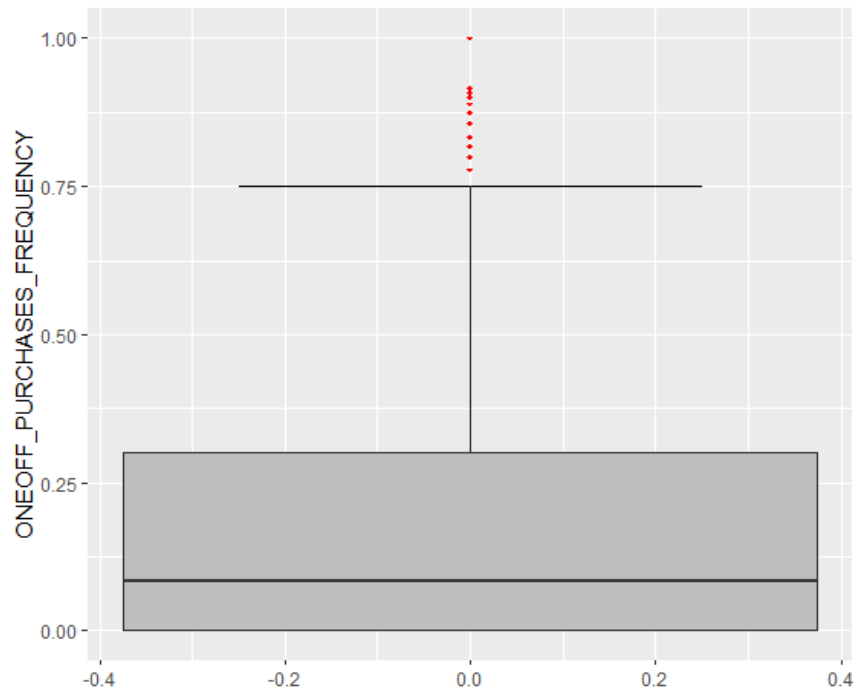


BOX PLOT FOR CASH_ADVANCE_TRX

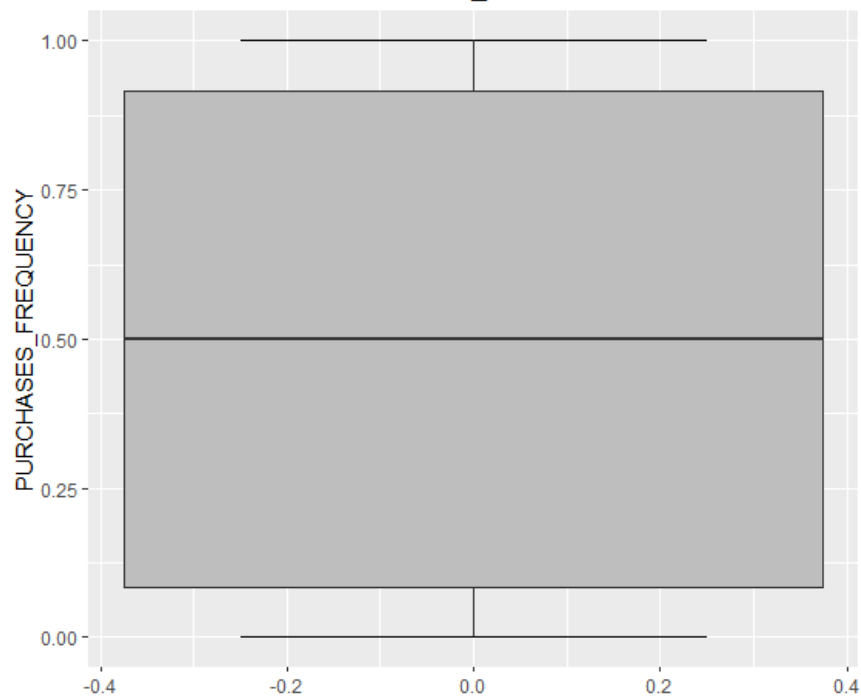




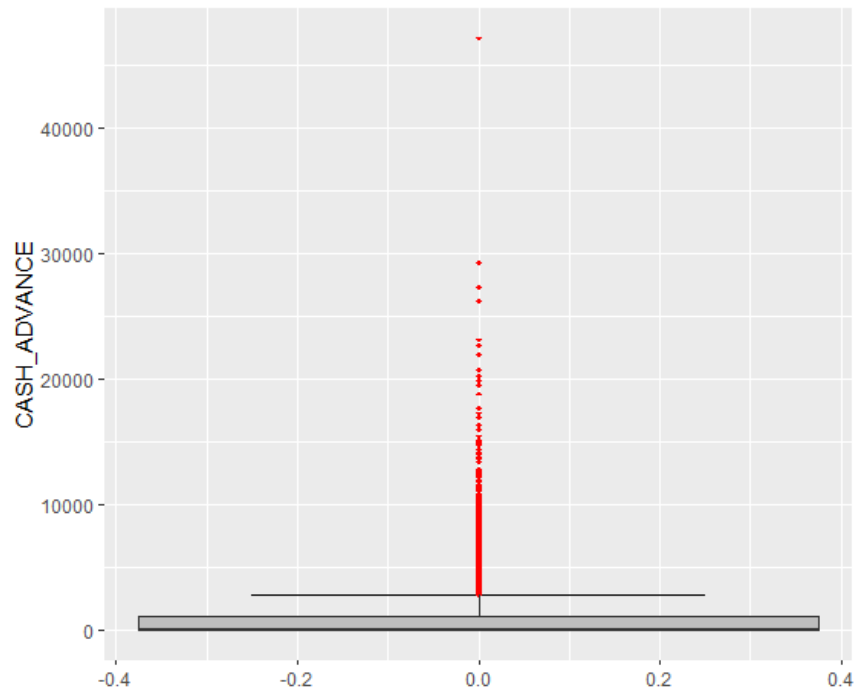
BOX PLOT FOR ONEOFF_PURCHASES_FREQUENCY



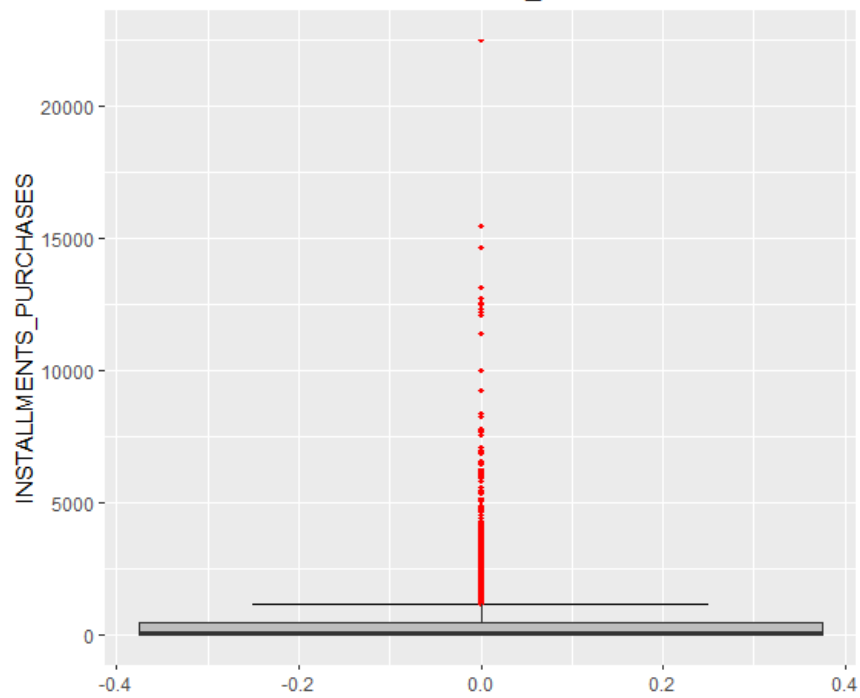
BOX PLOT FOR PURCHASES_FREQUENCY



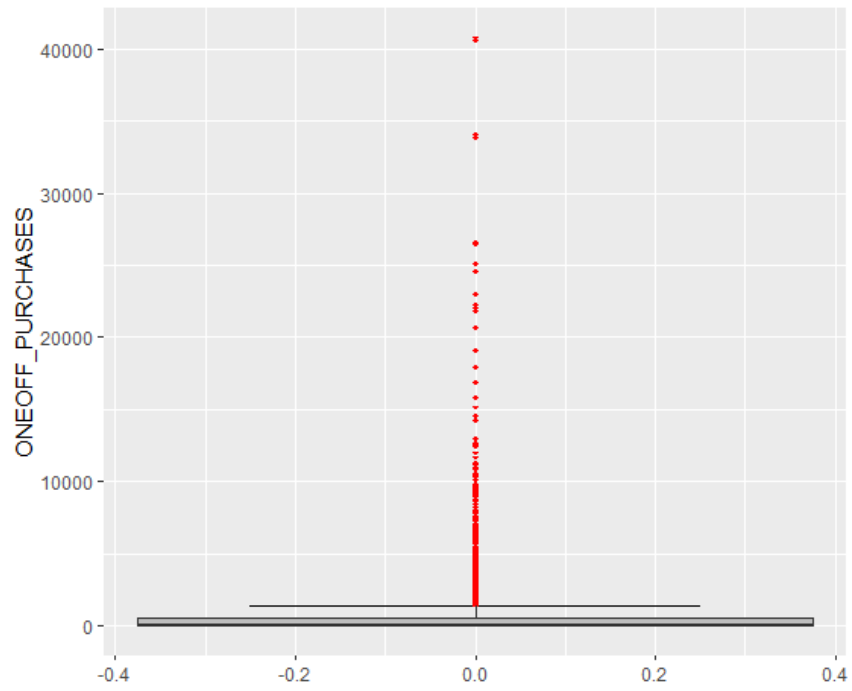
BOX PLOT FOR CASH_ADVANCE



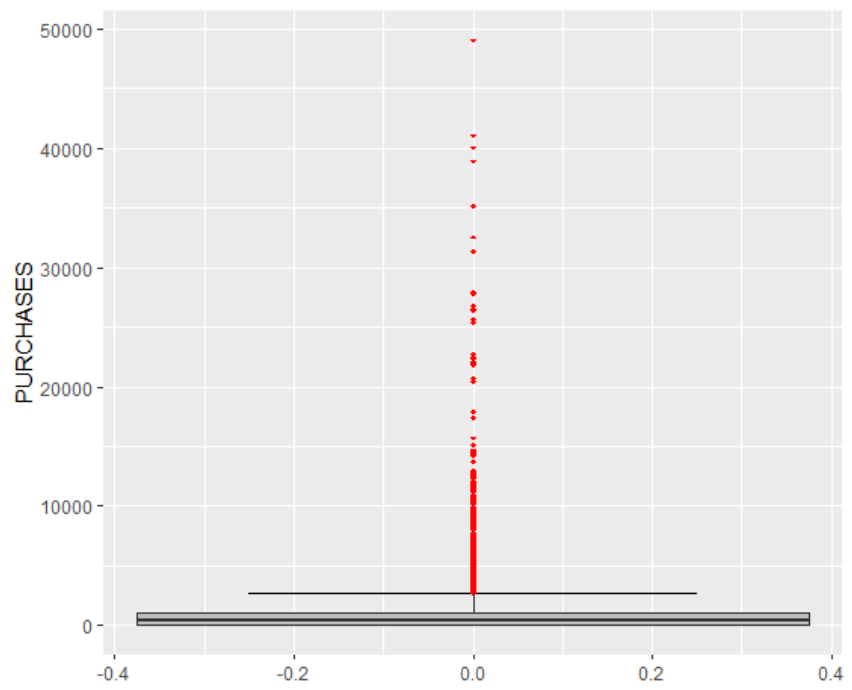
BOX PLOT FOR INSTALLMENTS_PURCHASES



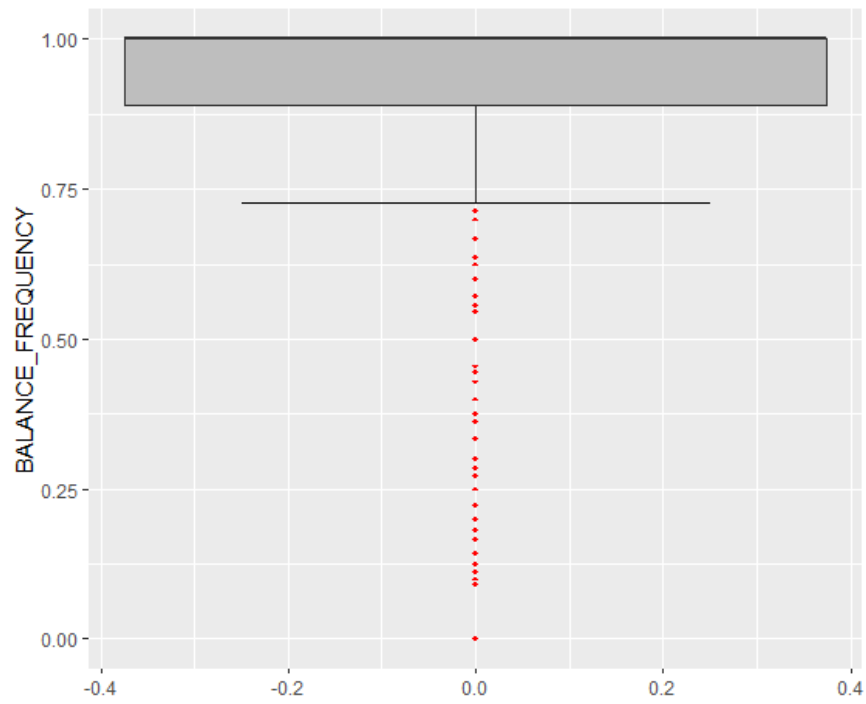
BOX PLOT FOR ONEOFF_PURCHASES



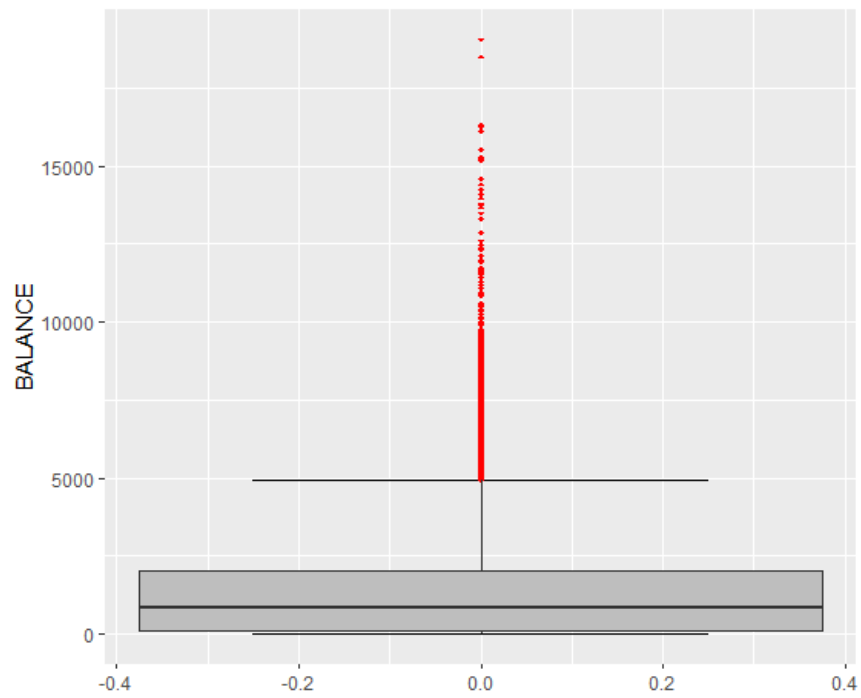
BOX PLOT FOR PURCHASES

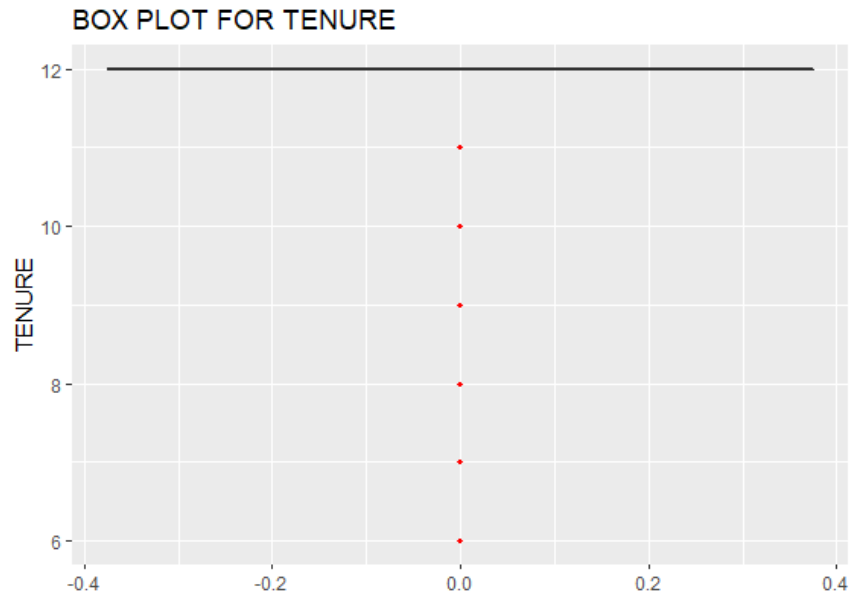


BOX PLOT FOR BALANCE_FREQUENCY



BOX PLOT FOR BALANCE





STEP-4

Outlier Removal:

Outlier replaced by NA method: This method is chosen because we can not afford deletion of the outliers as sometimes outliers contains important information. After replacing outliers with NA, the NA values are further computed by the method KNN imputation.

Code used as below:

#Replacing outliers with NA present in the dataset as this is the best method to deal with the NA as taught in lecture because on deleting outliers we could lose some of the important data.

```
for(i in Cnames)
{
  Credit_Card_Data[,i]=as.data.frame(Credit_Card_Data[,i])
  Val=Credit_Card_Data[,i][Credit_Card_Data[,i] %in% boxplot.stats(as.list(Credit_Card_Data[,i]))$out]

  Credit_Card_Data[,i][Credit_Card_Data[,i] %in% Val]=NA
}

#NA values computation
```

```
Credit_Card_Data=knnImputation(Credit_Card_Data,k=3)
```

```
#ReCheck for missing values
```

```
sum(is.na(Credit_Card_Data))
```

```
#Check for Dataset dimensions
```

```
dim(Credit_Card_Data)
```

STEP 5

Advance Data Preparation

Below Key Performance Indicators are derived from the existing dataset:

1. PAYMENTS_TO_MINIMUM_PAYMENT_RATIO,
2. AVERAGE_AMOUNT_PER_PURCHASE,
3. MONTHLY_INSTALLMENTS_PURCHASES,
4. MONTHLY_CASH_ADVANCE_TRANSACTION,
5. MONTHLY_CASH_ADVANCE,
6. MONTHLY_ONEOFF_PURCHASES,
7. LIMIT_USAGE,
8. MONTHLY_AVERAGE_PURCHASE

Code used as below:

```
##### ADVANCE DATA PREPARATION (CREATING NEW VARIABLES) #####
```

```
Credit_Card_Data$MONTHLY_AVERAGE_PURCHASE =  
Credit_Card_Data$PURCHASES/(Credit_Card_Data$PURCHASES_FREQUENCY*Credit_Card_Data$TENURE)
```

```
Credit_Card_Data$LIMIT_USAGE=Credit_Card_Data$BALANCE/Credit_Card_Data$CREDIT_LIMIT
```

```
Credit_Card_Data$PAYMENTS_TO_MINIMUM_PAYMENT_RATIO =  
Credit_Card_Data$PAYMENTS/Credit_Card_Data$MINIMUM_PAYMENTS
```

```
Credit_Card_Data$AVERAGE_AMOUNT_PER_PURCHASE=Credit_Card_Data$PURCHASES/Credit_Card_Data$PURCHASES_TRX
```

```
Credit_Card_Data$MONTHLY_CASH_ADVANCE =  
Credit_Card_Data$CASH_ADVANCE/(Credit_Card_Data$CASH_ADVANCE_FREQUENCY*Credit_Card_Data$TENURE)
```

```
Credit_Card_Data$MONTHLY_INSTALLMENTS_PURCHASES =  
Credit_Card_Data$INSTALLMENTS_PURCHASES/(Credit_Card_Data$PURCHASES_INSTALLMENTS_FREQ  
UENCY*Credit_Card_Data$TENURE)
```

```
Credit_Card_Data$MONTHLY_ONEOFF_PURCHASES =  
Credit_Card_Data$ONEOFF_PURCHASES/(Credit_Card_Data$ONEOFF_PURCHASES_FREQUENCY*Credit  
_Card_Data$TENURE)
```

```
Credit_Card_Data$MONTHLY_CASH_ADVANCE_TRANSACTION =  
Credit_Card_Data$CASH_ADVANCE/Credit_Card_Data$CASH_ADVANCE_TRX
```

#Checking no of observations and variables present in the object Credit_Card_Data

```
dim(Credit_Card_Data)
```

#writing the data into file after preparing Advance data

```
write.csv(Credit_Card_Data,"Advance_KPI_DataSet.csv",row.names=T)
```

#Copying data and storing KPI in Advance_KPI_Dataset

```
Advance_KPI_Dataset=Credit_Card_Data
```

#Dropping derived KPI from the data set to use actual data for further computation purpose

```
Credit_Card_Data = subset(Credit_Card_Data, select = -  
c(PAYMENTS_TO_MINIMUM_PAYMENT_RATIO,AVERAGE_AMOUNT_PER_PURCHASE,MONTHLY_INSTALL  
MENTS_PURCHASES,MONTHLY_CASH_ADVANCE_TRANSACTION,MONTHLY_CASH_ADVANCE,MONTHL  
Y_ONEOFF_PURCHASES,LIMIT_USAGE,MONTHLY_AVERAGE_PURCHASE))
```

```
dim(Credit_Card_Data)
```

STEP-6

Feature selection

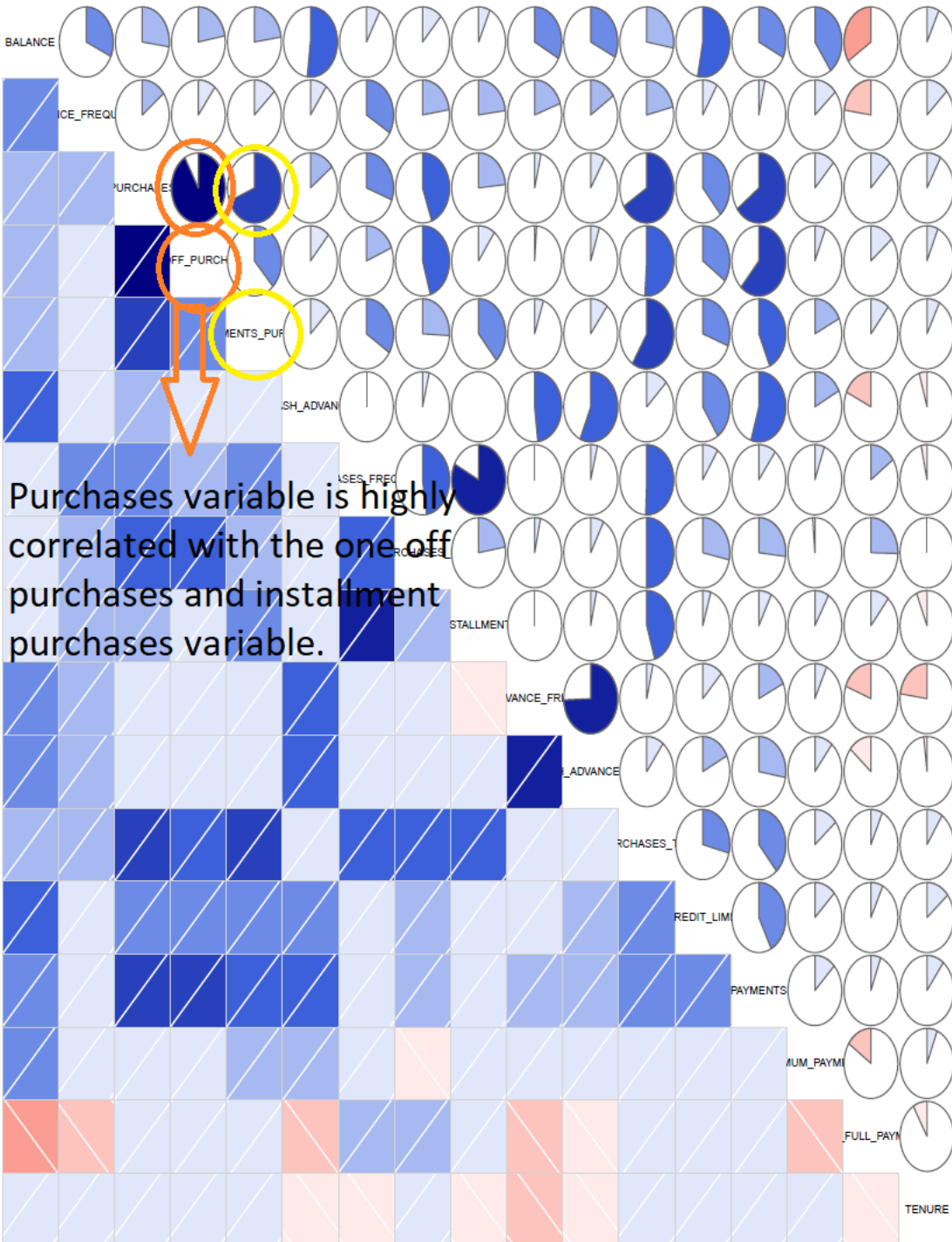
This stage involves the process of reducing variables on the basis of correlation present in the variables of the dataset. Correlation plot is being used to analyze the correlation among the variables and reduce the variables using factor analysis method.

This factor Analysis method is used to identify the latent relational structure among a set of variables and narrow down to smaller number of variables. Also, this method is especially helpful in segmenting the data.

Code used as below:

```
##### FEATURE SELECTION #####  
##### CORRELATION PLOT #####  
  
#Selecting only numeric variables  
  
Numeric_index= supply(Credit_Card_Data,is.numeric)  
  
Numeric_data=Credit_Card_Data[,Numeric_index]  
  
#Variables names containing numeric data  
  
Cnames=colnames(Numeric_data)  
  
Cnames  
  
#Correlation plot  
  
#Variables are reduced by Factor Analysis technique.  
  
#So, not dropping variables that are highly correlated with more than one variables  
  
corrgram(Credit_Card_Data[,Cnames],order=F,upper.panel=panel.pie,text.panel=panel.txt,main="CORR  
ELATION PLOT")  
  
#Checking no of observations and variables present in the object Credit_Card_Data  
  
dim(Credit_Card_Data)
```

CORRELATION PLOT



```
##### FACTOR ANALYSIS #####
```

```
install.packages('psych')
```

```
install.packages('GPArotation')
```

```
library(psych)
```

```
library(GPArotation)
```

#We'll be using "Psych" package's "fa.parallel" function to execute parallel analysis. It will also find acceptable number of factors and generate the "scree plot"

```
Parallel_Analysis=fa.parallel(Credit_Card_Data[-1], fm = 'ml', fa = 'fa')
```

#The blue line shows eigenvalues of actual data and the two red lines (placed on top of each other) show simulated and resampled data.

#Here we look at the large drops in the actual data and spot the point where it levels off to the right. Also we locate the point of inflection - the point where the gap between simulated data and actual data tends to be minimum.

#Looking at scree plot and parallel analysis, anywhere between 3 to 6 factors. Factors would be good choice.

#Parallel analysis suggests that the maximum number of factors = 6

#Create the correlation matrix

```
Credit_Card_Cor = cor(Credit_Card_Data[-1])
```

```
write.csv(Credit_Card_Cor, "Customer_Correlation_Matrix.csv", row.names = T)
```

#display correlation matrix

```
Credit_Card_Cor
```

#In order to perform factor analysis, we'll use `psych` package's `fa()` function. Given below are the arguments we'll supply:

#r- Raw data or correlation or covariance matrix

#nfactors - Number of factors to extract

#rotate - Although there are various types rotations, `Varimax` and `Oblimin` are most popular

#fm - One of the factor extraction techniques like `Minimum Residual (OLS)`, `Maximum Likelihood`, `Principal Axis` etc.

#In this case, we will select oblique rotation (rotate = "oblimin") as we believe that there is correlation in the factors.

```
Factor_Analysis=fa(r=Credit_Card_Data[-1],nfactors =6,rotate = "varimax",fm="ml")  
print(Factor_Analysis)
```

#we need to consider the loadings more than 0.3 and not loading on more than one factor

```
print(Factor_Analysis$loadings,cutoff = 0.5)  
print(Factor_Analysis)
```

#Writing loadings value in the csv file

```
Loadings=data.frame(unclass(Factor_Analysis$loadings))  
Loadings=cbind(rownames(Loadings),Loadings)  
rownames(Loadings)=NULL  
colnames(Loadings)[1]=" Features "  
write.csv(Loadings,"Factor_Analysis_Loadings.csv",row.names = T)
```

#Diagrammatical view of factor analysis

```
fa.diagram(Factor_Analysis)
```

The factor analysis results in six factors when applied on the dataset. The variables contain by each factor are shown below:



Considering only those variables which we got by applying the factor analysis and dropping others variables from the data set for performing clustering algorithm.

#Dropping variables as per Factor Analysis

```
Credit_Card_Data = subset(Credit_Card_Data, select = -  
c(BALANCE_FREQUENCY,PURCHASES_TRX,CREDIT_LIMIT,MINIMUM_PAYMENTS,PRC_FULL_PAYMENT,T  
ENURE))
```

Since our Data Set doesn't contain target variable so it falls under unsupervised machine learning algorithm which can only be evaluated by using different clustering algorithms. In Clustering algorithm, we divide data into different clusters in accordance with the common characteristics present among them.

STEP-7

STANDARDISING DATA

Before feeding pre processed data, we need to scale the data properly in order to compute the efficient results.

Code used:

Standarizing data

```
Credit_Card_New_Data=data.frame(scale(Credit_Card_Data[-1]))
```

#Displaying Standardized data

```
Credit_Card_New_Data
```

STEP-8

EXTRACTION OF CLUSTERS

In this project, we would be using K-means clustering algorithm to compute clustering of the data set. The K-means method of the clustering algorithm require number of clusters as a parameter which need to be calculated prior to the clustering algorithm. NbClust Method is used to find the optimum number of clusters to be build.

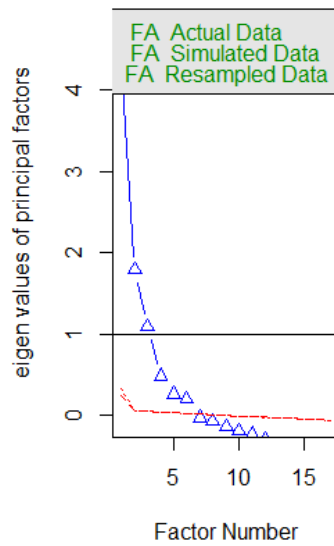
We need to pass the dataset into the parameter of NbClust method however when we are passing the dataset into the parameter of this method, we are getting an error of not enough space in RAM. Therefore, I passed factor loading as an argument in the NbClust Method to compute optimum number of clusters to be build.

Code used:

#Extracting no of clusters to build using Factor loadings because using dataset directly is giving me error of not enough RAM space

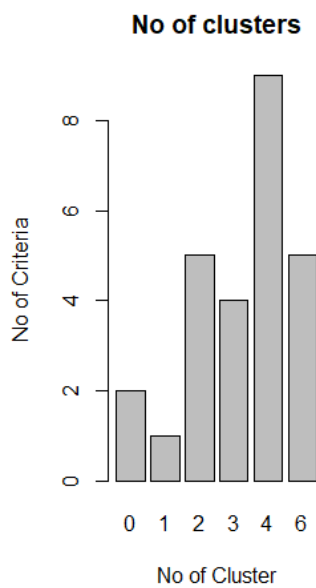
```
NBclust_Creditcard=NbClust(Factor_Analysis$loadings,min.nc=2,max.nc=6,method="kmeans")
```

Parallel Analysis Scree Plot



#Barplot to analyse the optimum clusters

```
barplot(table(NBclust_Creditcard$Best.n[1,]),xlab="No of Cluster",ylab="No of Criteria",main="No of clusters")
```



STEP-9

PERFORMING CLUSTERING ALGORITHM

After extracting the optimum numbers of clusters, performing clustering algorithm using K-means clustering.

Code used:

```
##### CLUSTERING ALGORITHM #####  
  
install.packages("NbClust")  
  
library(NbClust)  
  
#Dropping variables as per Factor Analysis  
  
Credit_Card_Data = subset(Credit_Card_Data, select = -  
c(BALANCE_FREQUENCY,PURCHASES_TRX,CREDIT_LIMIT,MINIMUM_PAYMENTS,PRC_FULL_PAYMENT,T  
ENURE))  
  
# Standarizing data  
  
Credit_Card_New_Data=data.frame(scale(Credit_Card_Data[-1]))  
  
#Displaying Standardized data  
  
Credit_Card_New_Data  
  
#Extracting no of clusters to build using Factor loadings because using dataset directly is giving me error  
of not enough RAM space  
  
NBclust_Creditcard=NbClust(Factor_Analysis$loadings,min.nc=2,max.nc=6,method="kmeans")  
  
#Barplot to analyse the optimum clusters  
  
barplot(table(NBclust_Creditcard$Best.n[1,]),xlab="No of Cluster",ylab="No of Criteria",main="No of  
clusters")
```

```
#K-means clustering
```

```
#According to the majority rule, the best number of clusters suggested by NbClust method = 4
```

```
Kmeans_model=kmeans( Credit_Card_New_Data,4, nstart=25)
```

```
#Writing cluster_data into csv file which contains clustering information of the data
```

```
Cluster_data =data.frame( Credit_Card_New_Data,Kmeans_model$cluster)
```

```
Cluster_data
```

```
Cluster_data=cbind(rownames(Cluster_data),Cluster_data)
```

```
rownames(Cluster_data)=NULL
```

```
colnames(Cluster_data)[1]=" Features "
```

```
write.csv(Cluster_data,"Cluster_Data_Of_Creditcard_Holders.csv",row.names = T)
```

```
#summarizing model
```

```
Kmeans_model
```

```
table(Kmeans_model$cluster)
```

```
dim(Credit_Card_Data)
```

STEP-10

VISULIZATION OF CLUSTERS

After K-means algorithm, all the observations from the dataset are placed in the different clusters according to the common characteristics which is then visualized using fviz cluster method:

Code Used:

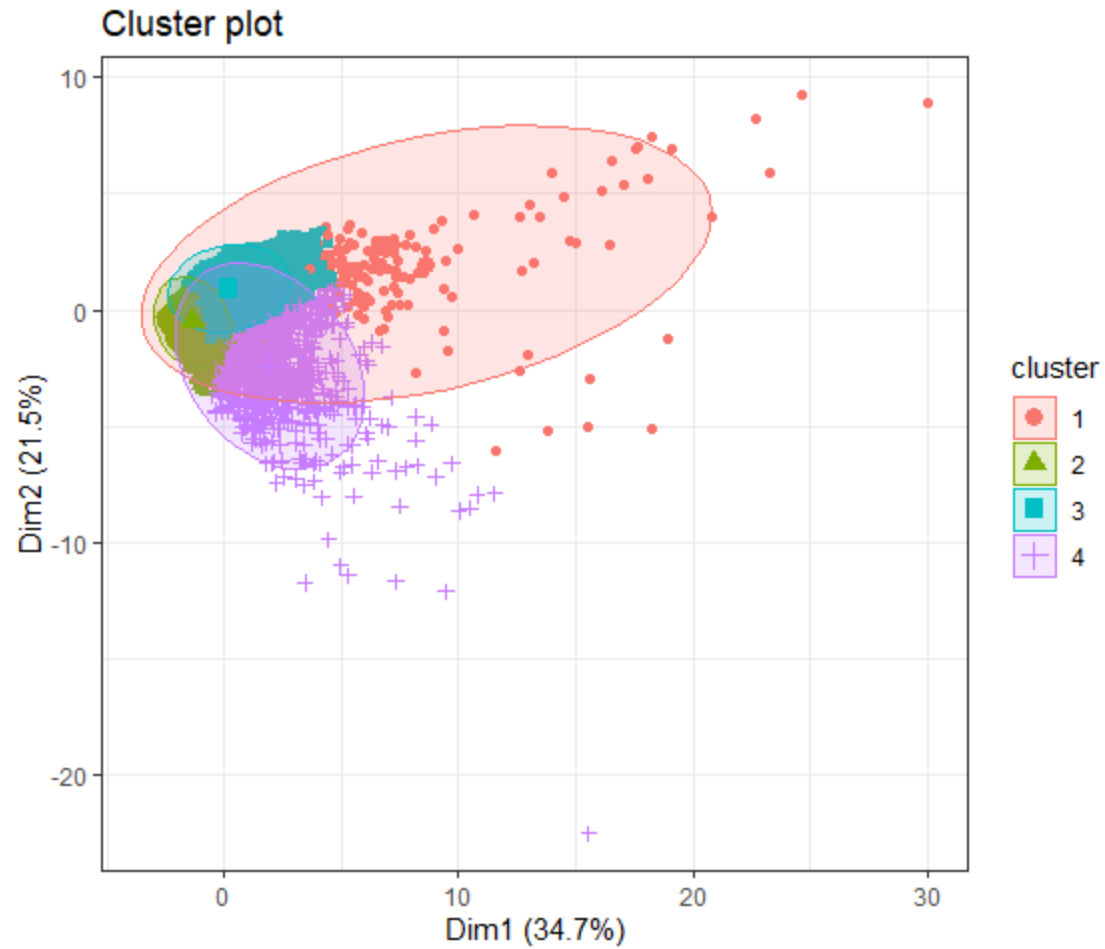
```
#Cluster Visualization
```

```
install.packages("factoextra")
```

```
library(factoextra)
```

```
library(cluster)
```

```
fviz_cluster(Kmeans_model, data = Credit_Card_New_Data, geom = "point",stand = FALSE, frame.type  
= "norm") + theme_bw()
```



STEP-11

RESULT BY R

By implementing through R and after performing all the necessary steps, we got 4 number of clusters where we need to place our dataset. The number of observations of the dataset present in each cluster is then shown below:

Clusters	1	2	3	4	Total
Observations	142	3232	4433	1143	8950
Percentage	1.6%	36.1%	49.5%	12.7%	
Insights:	Premium Customer	Average_1	Average_2	Poor_customers	

Total: 8950

Category 1 Customers: Cluster 1: 1.6%

Insights-

These Customers have the maximum Transaction, Highest Purchases (both types), Highest Cash advance transactions highest Payments and their payment to minimum payments is also the highest. Also, they don't exhaust their accounts.

Strategy-

We can provide them a lower interest rate for the cash advance, offers on their credit card purchases to upsell, and we should also increase their credit limit so that they make relatively more no of TRX.

We can also introduce a scheme of loyalty points for these customers and provide better customer support so that they continue being a loyal customer and not churn in future as they provide the company maximum profits

Category 4: Cluster 4: 12.7%

Insights-

These customers make more purchases compared to category 1 customers; the balance is relatively low to majority in their accounts.

Credit limit of these customers are little above the average, they do not perform any cash transactions may be due to extra interest charges.

These are the customers which considerably perform highest no purchases and frequency transaction are more than others.

Strategy –

we should provide these customers a lower interest rate on the cash advance to that they start using the feature.

Credit limit should be increased as their limit usage is lowest and they won't be risky.

We should target these customers as they are likely to perform better.

Category 3: Cluster 3: 49.5%

Insights-

These customers perform more cash transactions rather than card usage at POS. Purchase (of both types are very low), but cash advance being high tends to make their payments and min payments very high. Balance also remains highest throughout.

Strategy-

We can provide them better offers on the cards. With offers like cashback, discounts on purchase, lower fee charges for the cards

To boost installments charges we can keep interest rate low so they tend to buy more

Since these are the heavy cash advance users, we can put a higher interest rate as their min to payment ratio is low and they are likely to default more which would add profit
These customers should be targeted and we can perform upsell on them

Category 2 Customers: Cluster 2: 36.1%

Insights-

These customers perform very low usage, balance also low maybe due to their income. They perform low on almost everything.

Strategy-

We can provide them with credit cards and see if they perform better with a very low interest rate and low card charges and upsell them the products

PART 2: IMPLEMENTATION by PYTHON

Procedure:

Step-1

Data Loading

Data Set is taken from the link provided and then loaded into the R environment for performing Data pre-processing techniques which are the necessary steps in the data science to organize data into proper format before feeding it to the model, because every Model accepts the data in a specific format form only.

Following commands used to perform above task:

#Importing required libraries

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from factor_analyzer import FactorAnalyzer
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
```

#Setting working directory

```
os.chdir("C:\\Users\\akhil\\Desktop\\Kriti Data\\EDWISOR_invoice\\data science study material\\Project")
```

#Check for working directory

```
os.getcwd()
#Loading csv file data into Credit_Card_Data object
Credit_Card_Data=pd.read_csv("credit-card-data.csv",sep=";")
```

#Check for loaded data

```
Credit_Card_Data
```

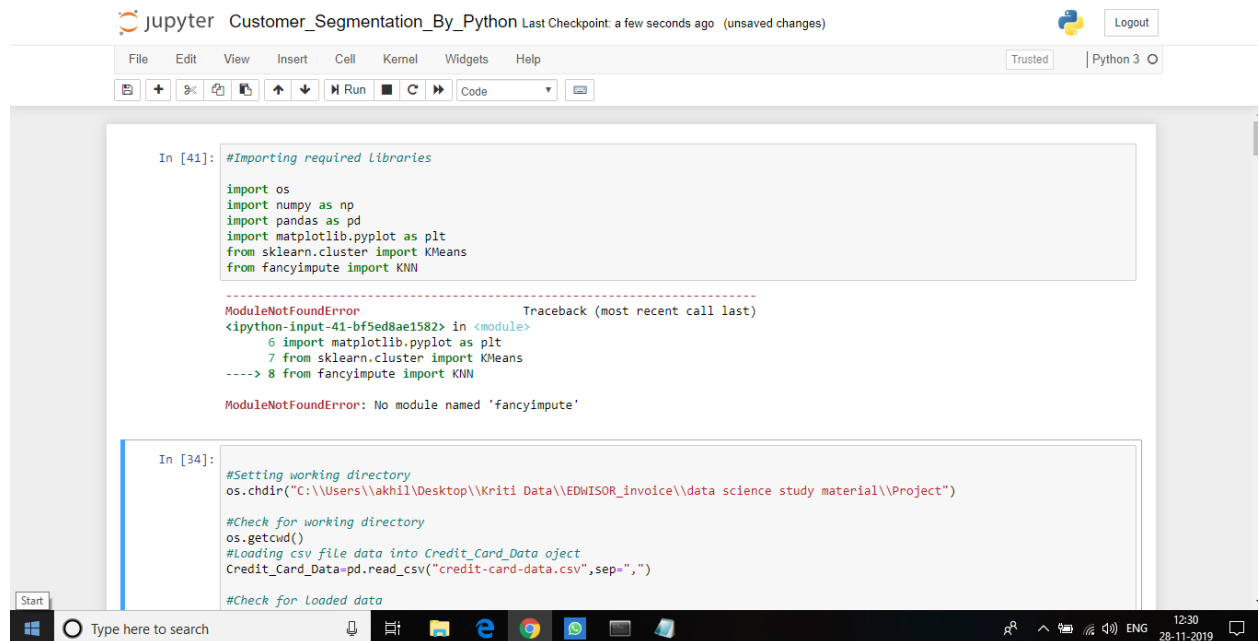
Step-2

Missing Value Analysis

After loading the data, the first step performed is missing value analysis where we compute the missing values or information present in the data set. Here the missing value analysis is computed by using the mean method because fancyimpute package could not be installed in my system.

Also tried using sys.path function but could not able to install. Hence computed missing values using mean method.

Screenshot of Error:



The screenshot shows a Jupyter Notebook titled "Customer_Segmentation_By_Python". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and code execution. The notebook has two code cells. The first cell, labeled "In [41]:", contains the following code:

```
#Importing required Libraries
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from fancyimpute import KNN
```

Below the code, a red error message is displayed:

```
ModuleNotFoundError: No module named 'fancyimpute'
```

The second cell, labeled "In [34]:", contains the following code:

```
#Setting working directory
os.chdir("C:\\Users\\akhil\\Desktop\\Kriti Data\\EDWISOR_invoice\\data science study material\\Project")

#Check for working directory
os.getcwd()

#Loading csv file data into Credit_Card_Data object
Credit_Card_Data=pd.read_csv("credit-card-data.csv",sep=",")

#Check for Loaded data
```

The Windows taskbar at the bottom shows the Start button, a search bar, and several application icons. The system clock indicates the time is 12:30 on 28-11-2019.

Following code is being used:

MISSING VALUE ANALYSIS

#Finding number of missing values in the variables

```
Miss_value=pd.DataFrame(Credit_Card_Data.isnull().sum())
```

Miss_value

#Only variable Credit_limit and Minimum_payments contains missing value so calculating missing percentage to compute missing values

#Missing value percentage of variable CREDIT_LIMIT

```
Miss_value_CREDIT_LIMIT = pd.DataFrame(Credit_Card_Data, columns=["CREDIT_LIMIT"])
```

```
Miss_value_CREDIT_LIMIT_Sum=pd.DataFrame(Miss_value_CREDIT_LIMIT.isnull().sum())
```

```
Miss_value_CREDIT_LIMIT_per =
```

```
(Miss_value_CREDIT_LIMIT_Sum/Credit_Card_Data.shape[0])*100
```

```
Miss_value_CREDIT_LIMIT_per
```

#Missing value percentage of variable MINIMUM_PAYMENTS

```

Miss_value_MINIMUM_PAYMENTS = pd.DataFrame(Credit_Card_Data, columns
=["MINIMUM_PAYMENTS"])
Miss_value_MINIMUM_PAYMENTS_Sum=pd.DataFrame(Miss_value_MINIMUM_PAYMENTS.isn
ull().sum())
Miss_value_MINIMUM_PAYMENTS_per =
(Miss_value_MINIMUM_PAYMENTS_Sum/Credit_Card_Data.shape[0])*100
Miss_value_MINIMUM_PAYMENTS_per

```

#Since missing value percentage is below 30% we need to compute it.Using mean method for computing missing value since.
 #I tried many times to compute it by KNN imputation but fancyimpute package is not supported by my version of python

```

Credit_Card_Data["CREDIT_LIMIT"]=Credit_Card_Data["CREDIT_LIMIT"].fillna(Credit_Card_Data
["CREDIT_LIMIT"].mean())
Credit_Card_Data["MINIMUM_PAYMENTS"]=Credit_Card_Data["MINIMUM_PAYMENTS"].fillna(
Credit_Card_Data["MINIMUM_PAYMENTS"].mean())

```

```

#Checking the number of missing values in the variables after computing missing value.
Miss_value=pd.DataFrame(Credit_Card_Data.isnull().sum())
Miss_value

```

Step-3: **Outliers Detection**

Once, the missing values are computed, the next step is to detect and remove outliers present in the data set. These outliers are nothing but the extreme values present in the data set.

In this project, missing values are detected using Box-plot method because it gives the graphical representation of the presence of the outliers in the data set and completely distinguish the outliers present in the lower and upper fence of the box plot.

```

##### OUTLIER ANALYSIS #####

```

```

#Plot boxplot to visualize outliers
%matplotlib inline

```

```

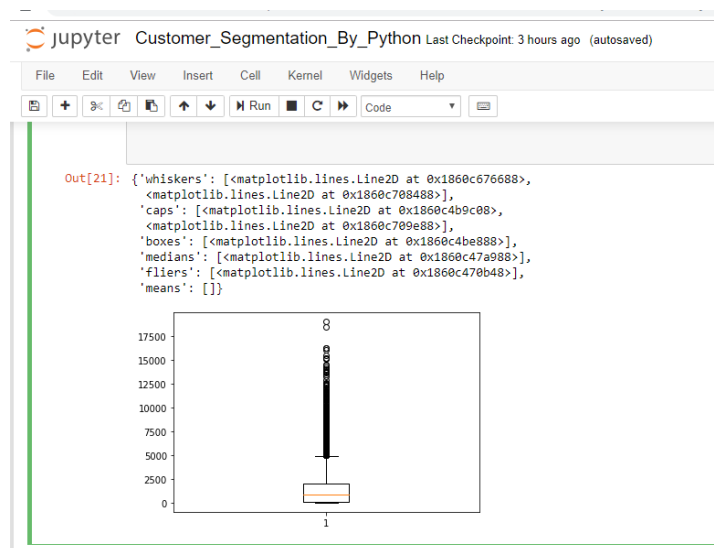
plt.boxplot(Credit_Card_Data["BALANCE"])
plt.boxplot(Credit_Card_Data["BALANCE_FREQUENCY"])
plt.boxplot(Credit_Card_Data["PURCHASES"])
plt.boxplot(Credit_Card_Data["ONEOFF_PURCHASES"])
plt.boxplot(Credit_Card_Data["INSTALLMENTS_PURCHASES"])
plt.boxplot(Credit_Card_Data["CREDIT_LIMIT"])
plt.boxplot(Credit_Card_Data["CASH_ADVANCE"])

```

```

plt.boxplot(Credit_Card_Data["PURCHASES_FREQUENCY"])
plt.boxplot(Credit_Card_Data["ONEOFF_PURCHASES_FREQUENCY"])
plt.boxplot(Credit_Card_Data["PURCHASES_INSTALLMENTS_FREQUENCY"])
plt.boxplot(Credit_Card_Data["CASH_ADVANCE_FREQUENCY"])
plt.boxplot(Credit_Card_Data["CASH_ADVANCE_TRX"])
plt.boxplot(Credit_Card_Data["PURCHASES_TRX"])
plt.boxplot(Credit_Card_Data["PAYMENTS"])
plt.boxplot(Credit_Card_Data["MINIMUM_PAYMENTS"])
plt.boxplot(Credit_Card_Data["PRC_FULL_PAYMENT"])
plt.boxplot(Credit_Card_Data["TENURE"])

```



STEP-4

Outlier Deletion:

Outlier replace by NA method is the best method for removing outliers but when computing it using the mean method, it is generating repetitive duplicate values so to avoid duplication by mean method deleting outliers by calculating inter quartile range. And also the reason KNN imputation method is not working in my system

Code used as below:

#Saving Numeric Column names

```

Cnames=["BALANCE","BALANCE_FREQUENCY","TENURE","PRC_FULL_PAYMENT","MINIMUM_PAYMENT",
S","PAYMENTS","PURCHASES_TRX","CASH_ADVANCE_FREQUENCY","PURCHASES_INSTALLMENTS_FREQ",
UENCY","PURCHASES","ONEOFF_PURCHASES","INSTALLMENTS_PURCHASES","CREDIT_LIMIT","CASH_A",
DVANCE","PURCHASES_FREQUENCY","ONEOFF_PURCHASES_FREQUENCY"]

```

Detect and delete outliers from data because outliers present in the dataset in large amount so imputing it by mean or median is creating duplicate values.

for i in Cnames:

```
q75,q25=np.percentile(Credit_Card_Data.loc[:,i],[75,25])
```

```
iqr=q75-q25
```

```
min=q25-(iqr*1.5)
```

```
max=q75+(iqr*1.5)
```

```
Credit_Card_Data=Credit_Card_Data.drop(Credit_Card_Data[Credit_Card_Data.loc[:,i]<min].index )
```

```
Credit_Card_Data=Credit_Card_Data.drop(Credit_Card_Data[Credit_Card_Data.loc[:,i]>max].index )
```

#Check dataset

```
Credit_Card_Data.shape
```

STEP 5

Advance Data Preparation

Below Key Performance Indicators are derived from the existing dataset:

1. PAYMENTS_TO_MINIMUM_PAYMENT_RATIO,
2. AVERAGE_AMOUNT_PER_PURCHASE,
3. MONTHLY_INSTALLMENTS_PURCHASES,
4. MONTHLY_CASH_ADVANCE_TRANSACTION,
5. MONTHLY_CASH_ADVANCE,
6. MONTHLY_ONEOFF_PURCHASES,
7. LIMIT_USAGE,
8. MONTHLY_AVERAGE_PURCHASE

Code used as below:

```
##### ADVANCE DATA PREPARATION (CREATING NEW VARIABLES) #####
```

```
Credit_Card_Data["MONTHLY_AVERAGE_PURCHASE"] =  
Credit_Card_Data["PURCHASES"]/(Credit_Card_Data["PURCHASES_FREQUENCY"]*Credit_Card_Data["T  
ENURE"])
```

```
Credit_Card_Data["LIMIT_USAGE"]=Credit_Card_Data["BALANCE"]/Credit_Card_Data["CREDIT_LIMIT"]
```

```
Credit_Card_Data["PAYMENTS_TO_MINIMUM_PAYMENT_RATIO"] =  
Credit_Card_Data["PAYMENTS"]/Credit_Card_Data["MINIMUM_PAYMENTS"]
```

```
Credit_Card_Data["AVERAGE_AMOUNT_PER_PURCHASE"]=Credit_Card_Data["PURCHASES"]/Credit_Card_Data["PURCHASES_TRX"]
```

```
Credit_Card_Data["MONTHLY_CASH_ADVANCE"] =  
Credit_Card_Data["CASH_ADVANCE"]/(Credit_Card_Data["CASH_ADVANCE_FREQUENCY"]*Credit_Card_Data["TENURE"])
```

```
Credit_Card_Data["MONTHLY_INSTALLMENTS_PURCHASES"] =  
Credit_Card_Data["INSTALLMENTS_PURCHASES"]/(Credit_Card_Data["PURCHASES_INSTALLMENTS_FREQUENCY"]*Credit_Card_Data["TENURE"])
```

```
Credit_Card_Data["MONTHLY_ONEOFF_PURCHASES"] =  
Credit_Card_Data["ONEOFF_PURCHASES"]/(Credit_Card_Data["ONEOFF_PURCHASES_FREQUENCY"]*Credit_Card_Data["TENURE"])
```

```
Credit_Card_Data["MONTHLY_CASH_ADVANCE_TRANSACTION"] =  
Credit_Card_Data["CASH_ADVANCE"]/Credit_Card_Data["CASH_ADVANCE_TRX"]
```

```
#Checking dataset
```

```
Credit_Card_Data
```

```
#Checking the number of missing values in the variables after computing missing value.
```

```
Miss_value=pd.DataFrame(Credit_Card_Data.isnull().sum())
```

```
Miss_value
```

```
Credit_Card_Data["MONTHLY_AVERAGE_PURCHASE"]=Credit_Card_Data["MONTHLY_AVERAGE_PURCHASE"].fillna(Credit_Card_Data["MONTHLY_AVERAGE_PURCHASE"].mean())
```

```
Credit_Card_Data["AVERAGE_AMOUNT_PER_PURCHASE"]=Credit_Card_Data["AVERAGE_AMOUNT_PER_PURCHASE"].fillna(Credit_Card_Data["AVERAGE_AMOUNT_PER_PURCHASE"].mean())
```

```
Credit_Card_Data["MONTHLY_CASH_ADVANCE"]=Credit_Card_Data["MONTHLY_CASH_ADVANCE"].fillna(Credit_Card_Data["MONTHLY_CASH_ADVANCE"].mean())
```

```
Credit_Card_Data["MONTHLY_INSTALLMENTS_PURCHASES"]=Credit_Card_Data["MONTHLY_INSTALLMENTS_PURCHASES"].fillna(Credit_Card_Data["MONTHLY_INSTALLMENTS_PURCHASES"].mean())
```

```
Credit_Card_Data["MONTHLY_ONEOFF_PURCHASES"]=Credit_Card_Data["MONTHLY_ONEOFF_PURCHASES"].fillna(Credit_Card_Data["MONTHLY_ONEOFF_PURCHASES"].mean())
```

```
Credit_Card_Data["MONTHLY_CASH_ADVANCE_TRANSACTION"]=Credit_Card_Data["MONTHLY_CASH_ADVANCE_TRANSACTION"].fillna(Credit_Card_Data["MONTHLY_CASH_ADVANCE_TRANSACTION"].mean())
```

```
#Checking the number of missing values in the variables after computing missing value.
```

```
Miss_value=pd.DataFrame(Credit_Card_Data.isnull().sum())
```

```
Miss_value
```

```
#Writing the KPI dataset into the csv file
```

```
#Credit_Card_Data.to_csv("Derived_KPI_Python.csv",index= False)
```

STEP-6

Feature selection

This stage involves the process of reducing variables on the basis of correlation present in the variables of the dataset. Correlation plot is being used to analyze the correlation among the variables and reduce the variables using factor analysis method.

This factor Analysis method is used to identify the latent relational structure among a set of variables and narrow down to smaller number of variables. Also, this method is especially helpful in segmenting the data.

Code used as below:

```
##### FEATURE SELECTION #####
```

```
##### CORRELATION PLOT #####
```

```
#Plotting correlation graph for the variables to look the correlation among variables.
```

```
#For reduction of the data I am using Factor Analysis that is why not dropping columns by looking into the correlation plot.
```

```
#Set the width and height of the plot
```

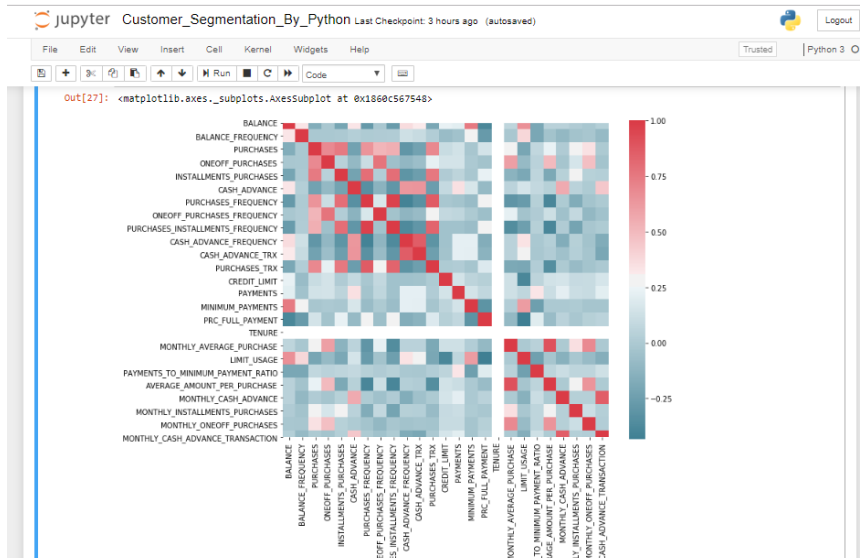
```
f,ax=plt.subplots(figsize=(10,8))
```

```
#Generating correlation matrix
```

```
corr=Credit_Card_Data.corr()
```

```
#Plot using seaborn library
```

```
sns.heatmap(corr,mask=np.zeros_like(corr,dtype=np.bool),cmap=sns.diverging_palette(220,10,as_cmap=True),square=True,ax=ax)
```



```
##### FACTOR ANALYSIS #####
```

```
#Loading csv file data into Credit_Card_Data object
```

```
Credit_Card_Data1=pd.read_csv("credit-card-data.csv",sep=",")
```

```
#Dropping categorical variable for computing loadings
```

```
Credit_Card_Data1=Credit_Card_Data1.drop(["CUST_ID"],axis=1)
```

```
#Create factor analysis object and perform factor analysis using random no of factors
```

```
fa = FactorAnalyzer(n_factors=6,rotation="varimax")
```

```
fa.fit(Credit_Card_Data1)
```

```
#Extracting loadings
```

```
loading=fa.loadings_
```

```
#Check for loadings
```


Loading

```
array([[ 0.09806281, -0.06107016,  0.28432516,  0.87615293,  0.07541759,
        0.36747411],
       [ 0.04841717,  0.20082362,  0.1449082 ,  0.33739635,  0.21793668,
       -0.06243859],
       [ 0.97587809,  0.21070448, -0.07802712,  0.08347717,  0.13541129,
        0.08458256],
       [ 0.82523208,  0.00533117, -0.05489563,  0.01779491,  0.29009276,
        0.12125943],
       [ 0.54123724,  0.48793844, -0.08035176,  0.13479604, -0.10150008,
        0.08895366],
       [-0.01784736, -0.09774277,  0.70620787,  0.1545424 , -0.08317565,
        0.45244868],
       [ 0.1441214 ,  0.86221511, -0.16351159, -0.05018351,  0.35417646,
        0.04014554],
       [ 0.35305544,  0.16231121, -0.07314447, -0.03095046,  0.83091274,
        0.12811212],
       [ 0.1170871 ,  0.94531582, -0.12453273,  0.01879683, -0.04015627,
       -0.00899201],
       [-0.03279619, -0.16055903,  0.89152176,  0.23353102,  0.01339853,
       -0.06418237],
       [-0.0067468 , -0.06593682,  0.84007598,  0.14327894, -0.00167299,
        0.06944575],
       [ 0.57255595,  0.48952239, -0.07135854,  0.13373529,  0.25477682,
        0.05982019],
       [ 0.26468542,  0.05109707,  0.09446975,  0.27808129,  0.16243334,
        0.51453507],
       [ 0.60265928,  0.05568064,  0.27903175,  0.01839585, -0.02706882,
        0.51374691],
       [ 0.072758 ,  0.03860842,  0.04958926,  0.39413072, -0.07150255,
        0.07985388],
       [ 0.13202822,  0.25051707, -0.13865081, -0.37456383,  0.04183147,
        0.14243871],
       [ 0.06038356,  0.05404423, -0.12383022,  0.12601082,  0.04761865,
        0.10407081]])
```

#Check Eigenvalues to know no of factors to be extract

#The output of the eigenvalues states 5 factors to be built because only 5 values are greater than one

```
ev, v = fa.get_eigenvalues()
```

```
ev
```

```
array([4.64060413e+00, 3.45334253e+00, 1.49824089e+00, 1.27151915e+00,  
       1.05820397e+00, 9.75809596e-01, 8.30182344e-01, 7.30864552e-01,  
       6.45703978e-01, 5.23600386e-01, 4.03282677e-01, 3.01472838e-01,  
       2.42735664e-01, 2.06876326e-01, 1.72153395e-01, 4.53959227e-02,  
       1.16488555e-05])
```

Create scree plot using matplotlib to extract no of factors to be built

```
plt.scatter(range(1,df.shape[1]+1),ev)
```

```
plt.plot(range(1,df.shape[1]+1),ev)
```

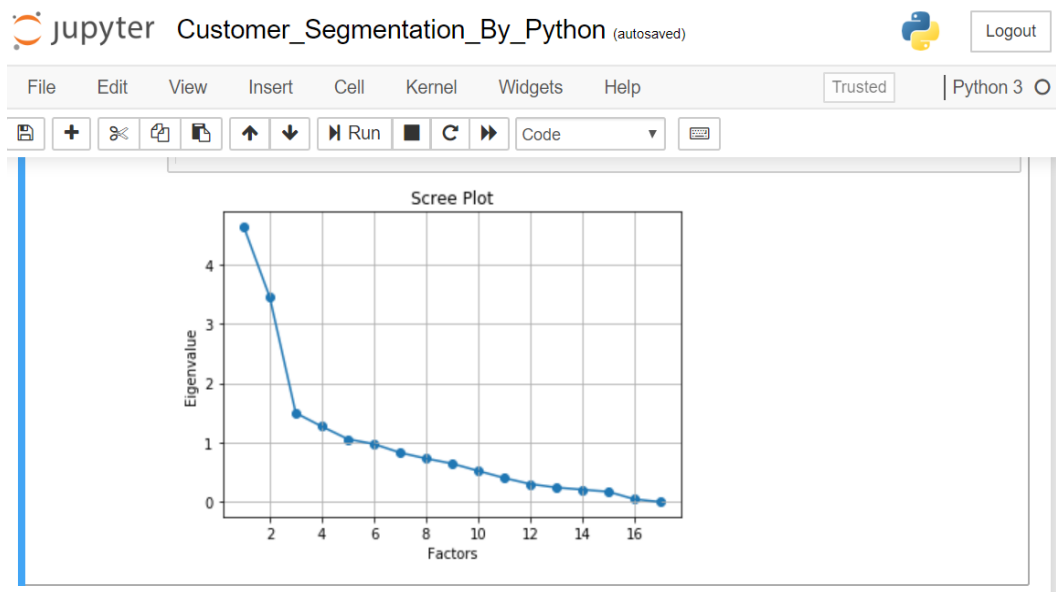
```
plt.title('Scree Plot')
```

```
plt.xlabel('Factors')
```

```
plt.ylabel('Eigenvalue')
```

```
plt.grid()
```

```
plt.show()
```



The factor analysis results in five factors when applied on the dataset.

Create factor analysis object and perform factor analysis using 5 factors

```
fa = FactorAnalyzer(n_factors=5,rotation="varimax")
```

```
fa.fit(df)
```

```
loading=fa.loadings_
```

#Getting loadings

Loading

The figure below is made from the loading data considering 5 factors extracted by scree plot



```
# Get variance of each factors
```

```
fa.get_factor_variance()
```

```
#Total 64% cumulative Variance explained by the 5 factors.
```

```
(array([2.8862607 , 2.34214435, 2.29315251, 1.41678247, 1.09170505,
        0.97022058]),
 array([0.16978004, 0.1377732 , 0.13489132, 0.08334015, 0.06421794,
        0.0570718 ]),
 array([0.16978004, 0.30755324, 0.44244456, 0.52578471, 0.59000265,
        0.64707445]))
```

Considering only those variables which we got by applying the factor analysis and dropping others variables from the data set for performing clustering algorithm.

```
#Dropping columns from the dataset as per Factor Analysis to compute cluster.
```

```
Credit_Card_Data1=Credit_Card_Data1.drop(["BALANCE_FREQUENCY","PURCHASES_TRX","CREDIT_LIMIT","MINIMUM_PAYMENTS","PRC_FULL_PAYMENT","TENURE"],axis=1)
```

Since our Data Set doesn't contain target variable so it falls under unsupervised machine learning algorithm which can only be evaluated by using different clustering algorithms. In Clustering algorithm, we divide data into different clusters in accordance with the common characteristics present among them.

STEP-7

EXTRACTION OF CLUSTERS

In this project, we would be using KMeans clustering algorithm to compute clustering of the data set. The KMeans method of the clustering algorithm require number of clusters as a parameter which need to be calculated prior to the clustering algorithm. KMeans Method is used to find the optimum number of clusters to be build.

Code used:

```
##### K-MEANS CLUSTER ANALYSIS #####
```

```
#Feeding dataset after removing missing values and the outliers from it
```

```
#Estimating optimum no of clusters
```

```
cluster_range=range(1,20)
```

```
cluster_errors=[]
```

```

for num_clusters in cluster_range:

    clusters=KMeans(num_clusters).fit(Credit_Card_Data1.iloc[:,0:11])

    cluster_errors.append(clusters.inertia_)

#Creating Dataframe

Cluster_Credit_Card_Data=pd.DataFrame({"num_clusters":cluster_range,"cluster_errors":cluster_errors
})

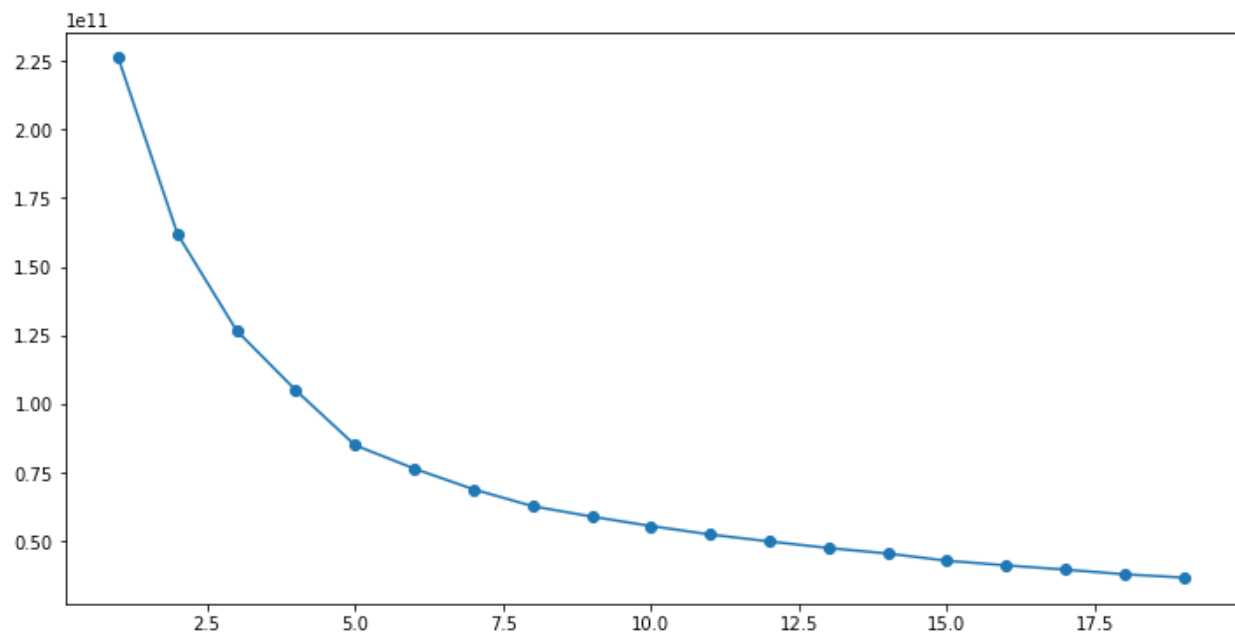
#Plot line to visualize no of clusters

%matplotlib inline

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

plt.plot(Cluster_Credit_Card_Data.num_clusters,Cluster_Credit_Card_Data.cluster_errors,marker="o")

```



From the graph, we can see that the number of clusters should be 4, because after 4.0 in horizontal axis, the value is decreasing uniformly.

STEP-8

PERFORMING CLUSTERING ALGORITHM

After extracting the optimum numbers of clusters, performing clustering algorithm using KMeans clustering.

Code used:

#Implementation of Kmeans

#According to the Elbow graph we determine the clusters number as 4

#Applying k-means algorithm to the Credit_Card_Data1 dataset.

Kmeans_model=KMeans(n_clusters=4).fit(Credit_Card_Data1)

#Summarizing output

pd.crosstab(Credit_Card_Data1["PURCHASES"],Kmeans_model.labels_)

#pd.crosstab(Credit_Card_Data1["ONEOFF_PURCHASES"],Kmeans_model.labels_)

#pd.crosstab(Credit_Card_Data1["INSTALLMENTS_PURCHASES"],Kmeans_model.labels_)

#pd.crosstab(Credit_Card_Data1["PURCHASES_FREQUENCY"],Kmeans_model.labels_)

#pd.crosstab(Credit_Card_Data1["ONEOFF_PURCHASES_FREQUENCY"],Kmeans_model.labels_)

#pd.crosstab(Credit_Card_Data1["PURCHASES_INSTALLMENTS_FREQUENCY"],Kmeans_model.labels_)



#pd.crosstab(Credit_Card_Data1["CASH_ADVANCE_FREQUENCY"],Kmeans_model.labels_)


#pd.crosstab(Credit_Card_Data1["CASH_ADVANCE"],Kmeans_model.labels_)

#pd.crosstab(Credit_Card_Data1["CASH_ADVANCE_TRX"],Kmeans_model.labels_)

#pd.crosstab(Credit_Card_Data1["BALANCE"],Kmeans_model.labels_)

#pd.crosstab(Credit_Card_Data1["PAYMENTS"],Kmeans_model.labels_)

localhost:8888/notebooks/Desktop/Kriti%20Data/EDWISOR_invoice/data%20science%20study%20material/Project/Custom...  jupyter Customer_Segmentation_By_Python Last Checkpoint: 43 minutes ago (unsaved changes)  Logout


File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 

```
#pd.crosstab(Credit_Card_Data["CASH_ADVANCE_TRX"],Kmeans_model.Labels_)
#pd.crosstab(Credit_Card_Data["BALANCE"],Kmeans_model.Labels_)
pd.crosstab(Credit_Card_Data["PAYMENTS"],Kmeans_model.Labels_)
```

Out[30]:

col_0	0	1	2	3
PAYMENTS				
0.000000	1	10	0	229
0.049513	0	0	0	1
0.056466	0	0	0	1
2.389583	0	0	0	1
3.500505	0	0	0	1
...
39048.597620	0	1	0	0
39461.965800	0	0	1	0
40627.595240	1	0	0	0
46930.598240	0	0	1	0
50721.483360	0	0	1	0

8711 rows x 4 columns

jupyter Customer_Segmentation_By_Python Last Checkpoint: 2 minutes ago (unsaved changes)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python

```
#pd.crosstab(Credit_Card_Data["PAYMENTS"],Kmeans_model.Labels_)
```

Out[65]:

col_0	0	1	2	3
ONEOFF_PURCHASES				
0.00	32	3743	527	0
0.01	0	5	2	0
0.02	0	1	1	0
0.05	0	1	0	0
0.24	0	1	0	0
...
26547.43	0	0	0	1
33803.84	0	0	0	1
34087.73	0	0	0	1
40624.06	0	0	0	1
40761.25	0	0	0	1

4014 rows x 4 columns

In []:

In []:

Jupyter Customer_Segmentation_By_Python Last Checkpoint: 4 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Out[66]:

	col_0	0	1	2	3
INSTALLMENTS_PURCHASES					
	0.00	84	3237	592	3
	1.95	0	1	0	0
	4.44	0	0	1	0
	4.80	0	1	0	0
	6.33	0	1	0	0

	12738.47	1	0	0	0
	13184.43	0	0	0	1
	14686.10	1	0	0	0
	15497.19	0	0	0	1
	22500.00	1	0	0	0

4452 rows x 4 columns

In []:

In []:

PROBLEM FACED:

1. Installing fancyimpute package for Missing value computation.
2. Performing Factor analysis .
3. Plotting cluster plot in python.
4. Segmentation of the data as variance between cluster is not high so containing overlapping values.