CREDIT CARD SEGMENTATION

DATA SCIENCE PROJECT 2

JAYRAM S

06-04-2020

Contents

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 behaviour of about 9000 active credit card holders during the last 6 months. The file is at a customer level with 18 behavioural variables.

1.2 Data

Variables Information:

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

1.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a method to analysing data sets to summarize their main features. In the given data set, there are 18 variables and data types of all variables are either float64 or int64. There are 8950 observations and 18 columns in our data set.

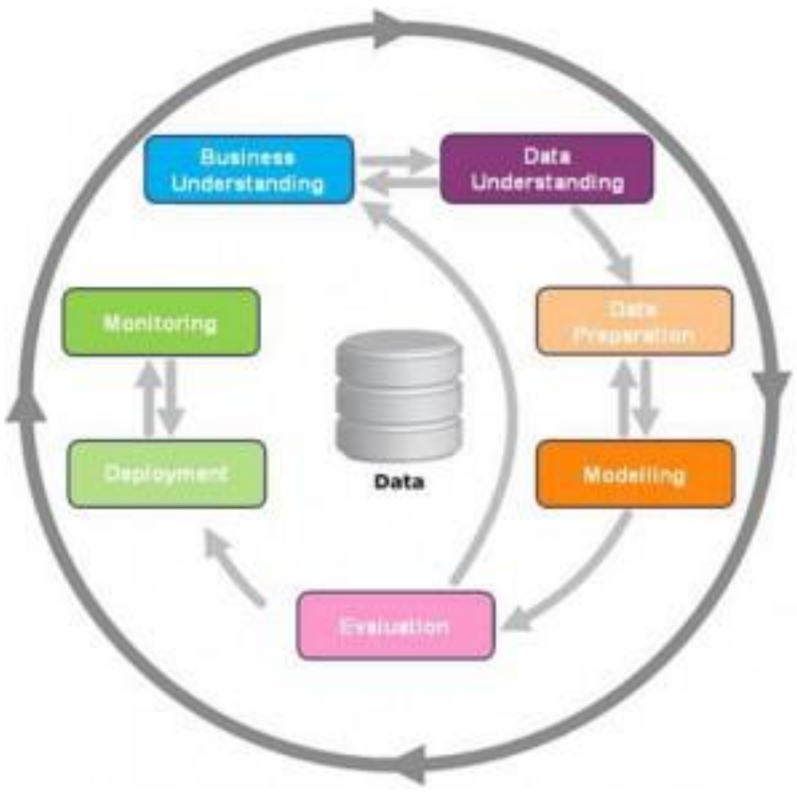
List of columns and their number of unique values -

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

Chapter 2

Methodology

Before feeding the data to the model we need to clean the data and convert it to a proper format. It is the most crucial part of data science project we spend almost 80% of time in it. We are following CRISP DM Methodology



Deriving KPI's and Extracting Insights from them.

As asked in the problem statement, we have to extract different KPI's(Key Performing Issues)

i. Monthly Average Purchases: We can obtain that by dividing Purchases with Tenure.

ii. Monthly Cash Advanced Amount: We can obtain that by dividing cash Advance with Tenure.

iii. Purchase Type: By observing the data, we can infer that there are two types of purchases. ONE_OFF_PURCHASES and INSTALLMENT_PURCHASES.

By Exploring more deeply, we can gain 4 meaningful insights,
There are FOUR types of Purchase behaviour in the data.
1. People who does not make any purchases.

2. People who make both types of purchases.
3. People who make only OneOff_Purchases
4. People who make only Installment_Purchases

We can also extract how many customers belong to each type of Purchase Behaviour.

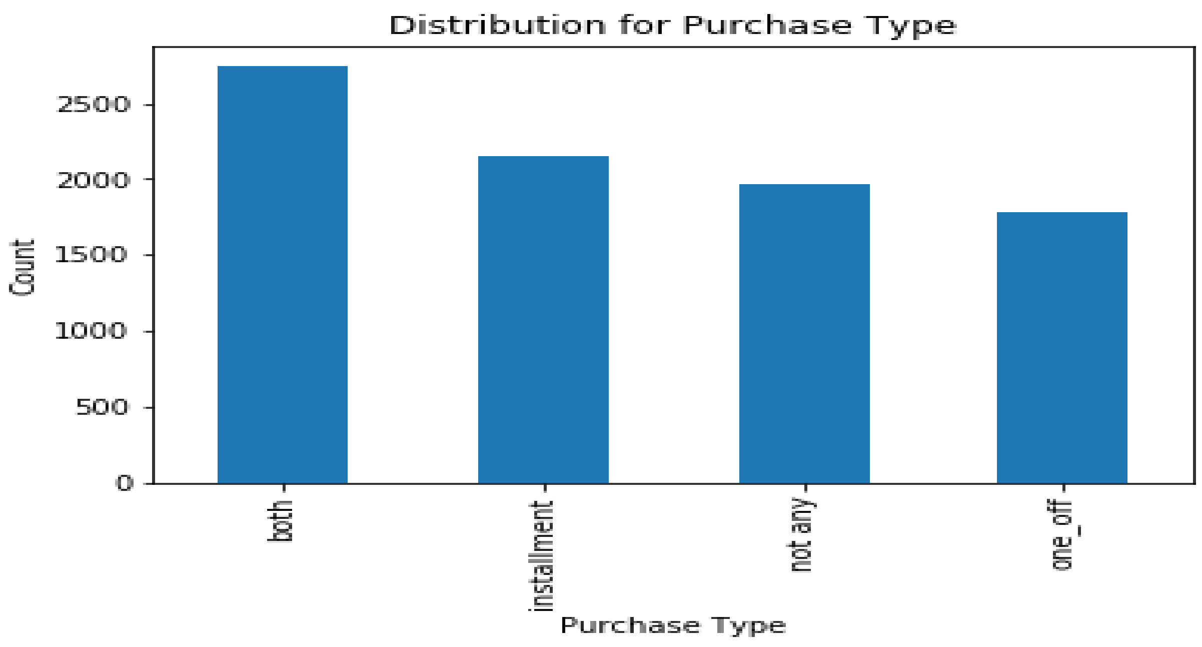


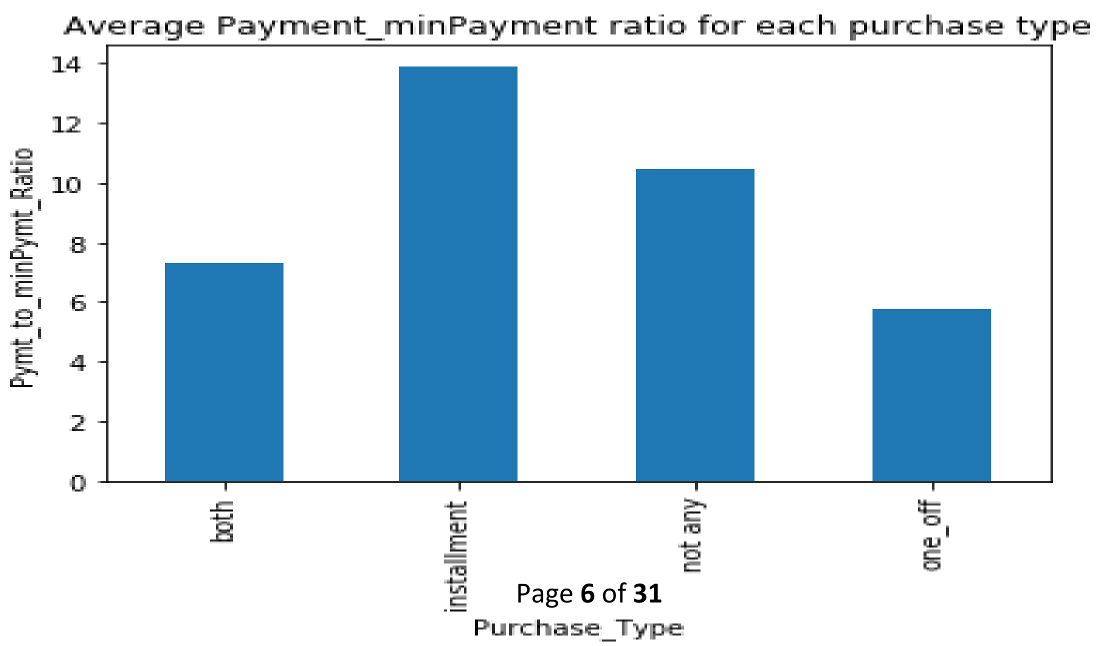
Fig: Distributions for Purchase Type

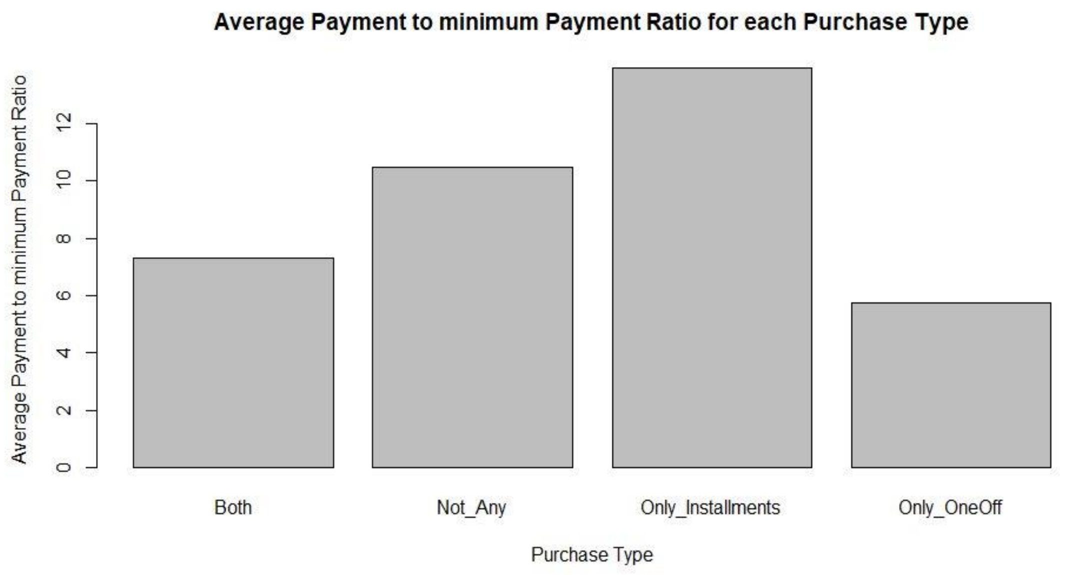
iv. Limit Usage: We can obtain it by dividing Balance with Credit limit.

v. Payments to Minimum payments Ratio: It is obtained by dividing Payments with minimum Payments.

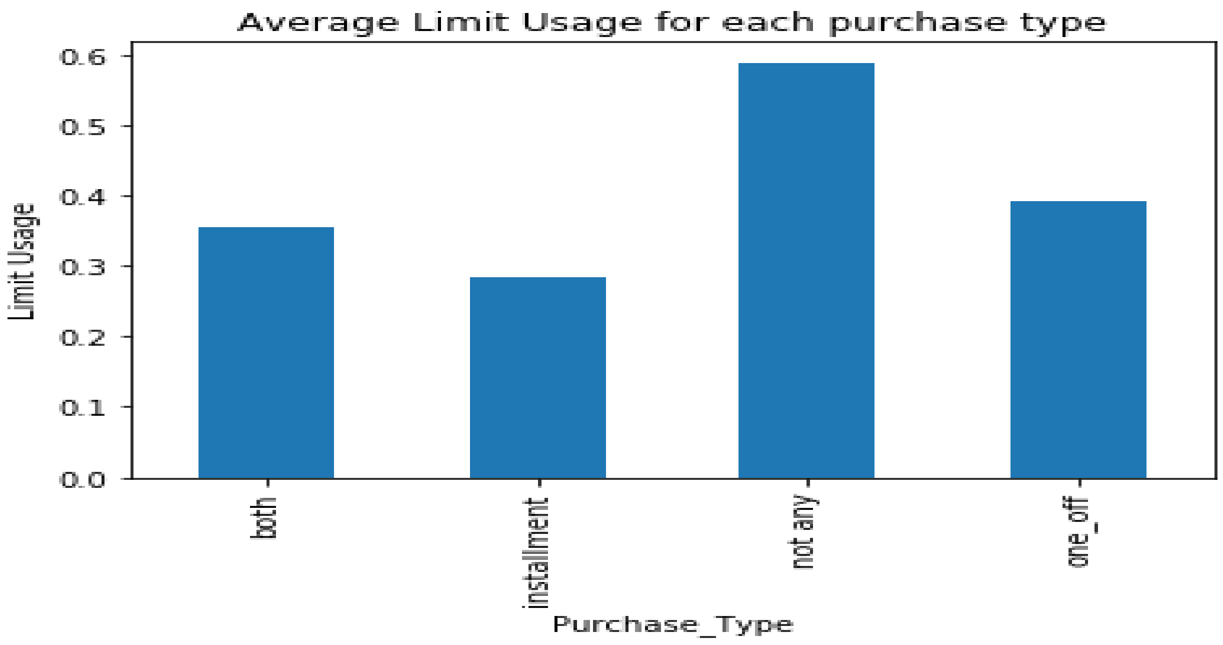
Insights from derived KPI's

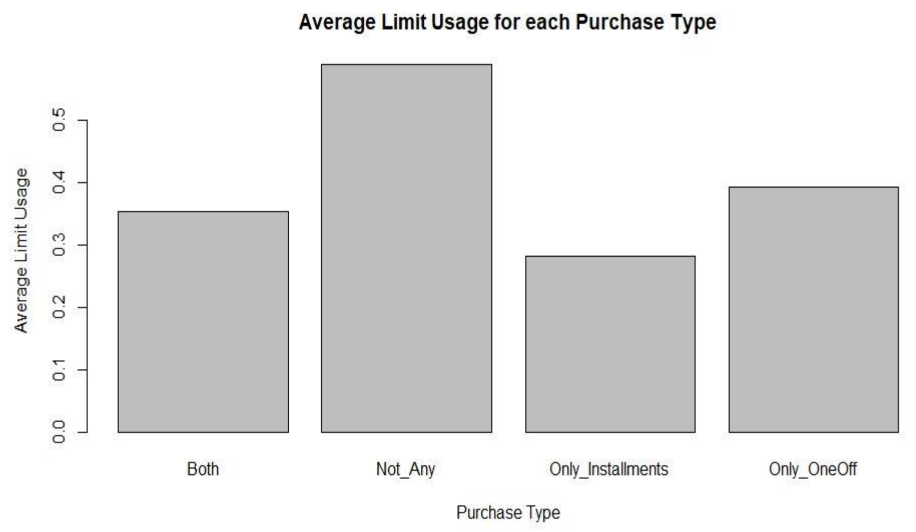
Plotting Bar Graph for Average Payment to minimum Payment Ratio VS Purchase Type





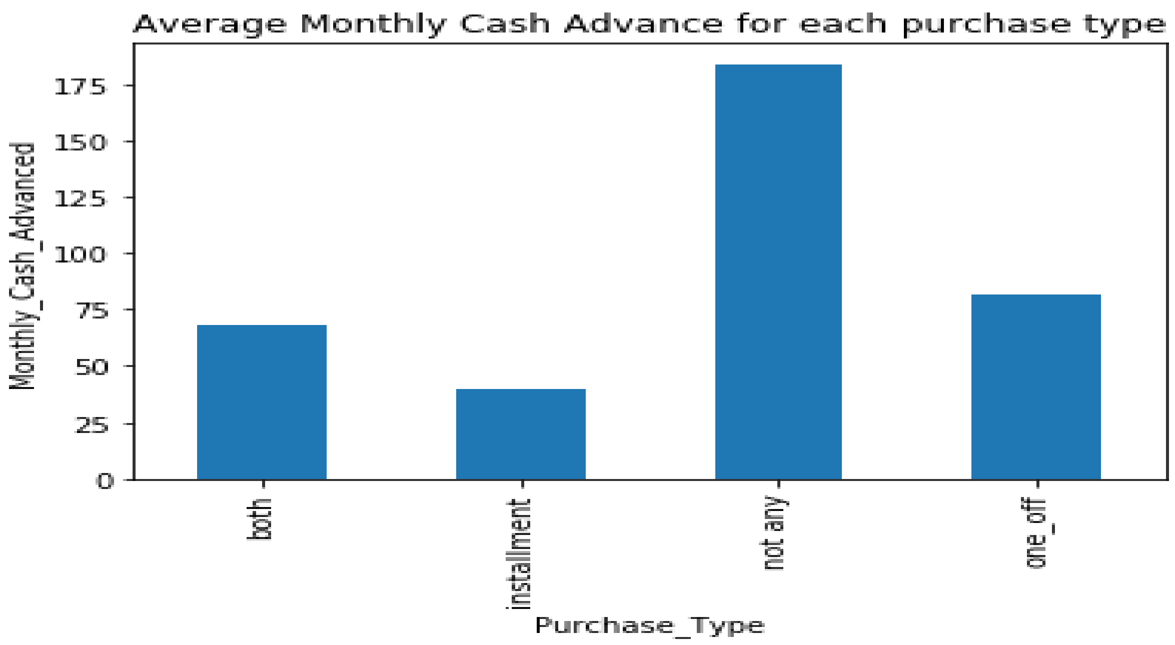
Plotting Bar Graph for Average Limit Usage VS Purchase Type

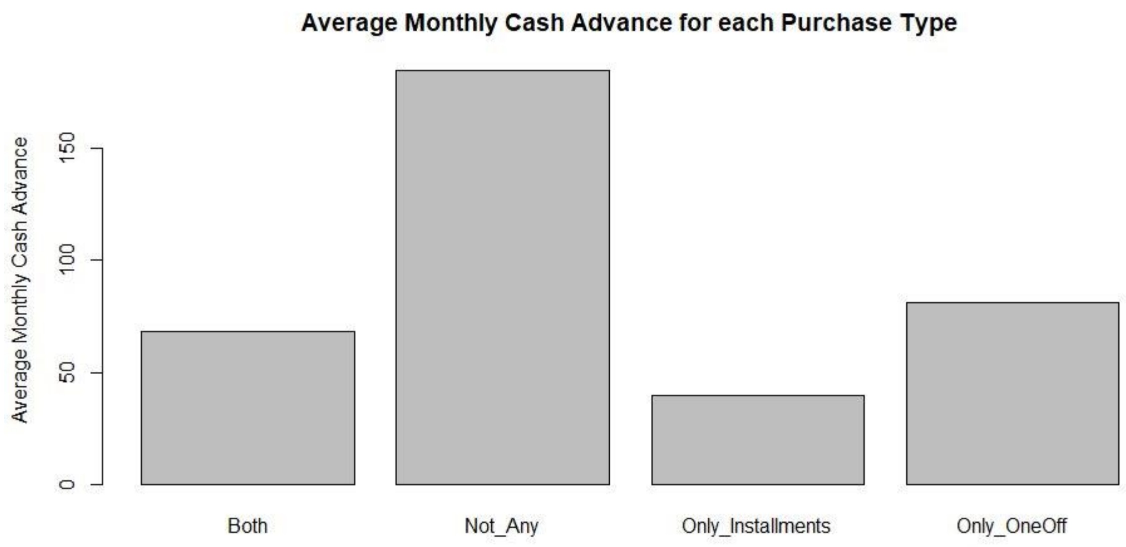




Insight 2: Clients with Purchase type Instalment have good Credit Score

Plotting Bar Graph for Average Monthly Cash Advance VS Purchase Type





Insight 3: Clients who don't do any type of Purchases (One-off, Instalment), Take more Cash on Advance.

2.1 Pre Processing

Any predictive modelling requires that we look at the data before we start modelling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is frequently called as Exploratory Data Analysis. To flinch this process, we will first try and look at all the probability distributions of the variables.

2.2.1 Missing Value Analysis

In statistics, *missing data*, or *missing values*, occur when no *data value* is stored for the variable in an observation. *Missing data* are a common occurrence and can have a significant effect on the conclusions that can be drawn from the *data*. If a column has more than 30% of data as missing value either we ignore the entire column or we ignore those observations

There are different ways to deal with missing values in the data.

- **Imputation**

We can impute the missing values with its mean, median or mode.
There is also another method called KNN imputation, which is available in both R and Python.

- **Partial deletion**

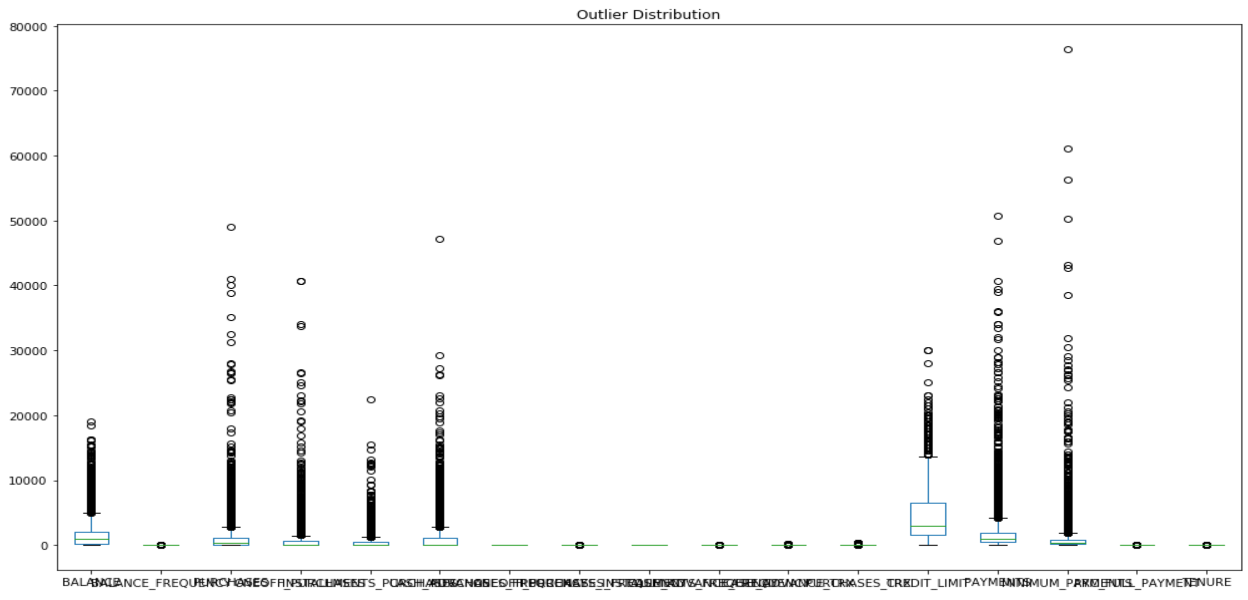
When the missing values in the data are less, we can delete them since they do not affect the model accuracy as they are in less number.

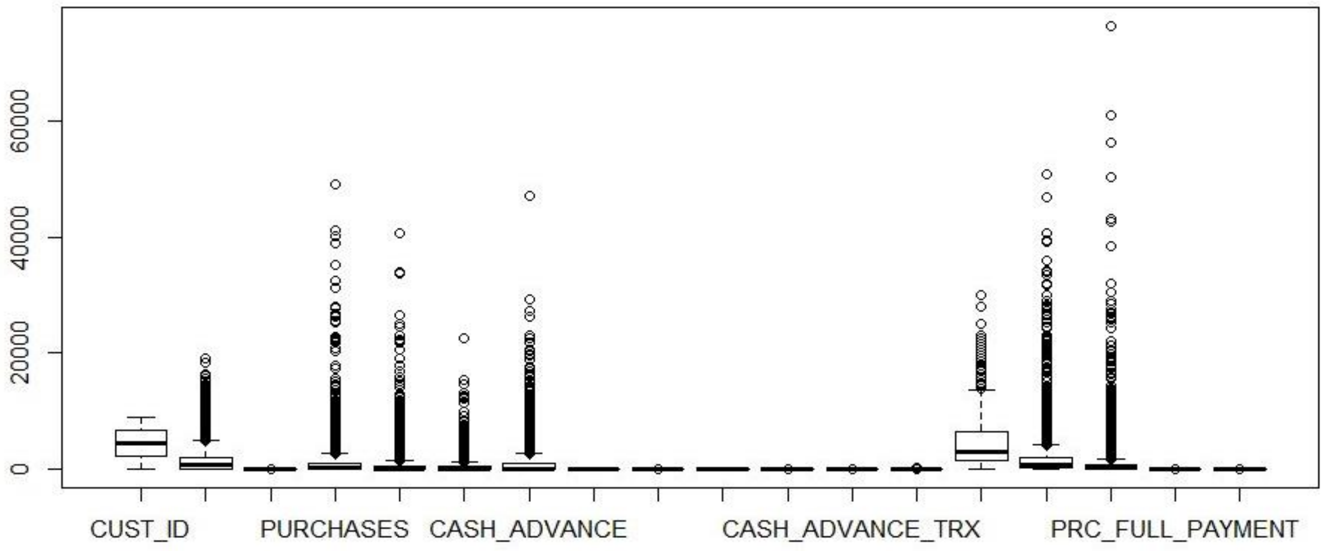
In our case, we are doing partial deletion.

2.1.2 Outlier Analysis

We can clearly observe from these probability distributions that most of the variables are skewed. The skew in these distributions can be most likely explained by the presence of outliers and extreme values in the data. One of the other steps of pre-processing apart from checking for normality is the presence of outliers. In this case we use a classic approach of removing outliers. We visualize the outliers using boxplots.

A simple way to way to detect outlier is to use boxplot.
For our data, the below images show the box plot, which are drawn in both R and Python.





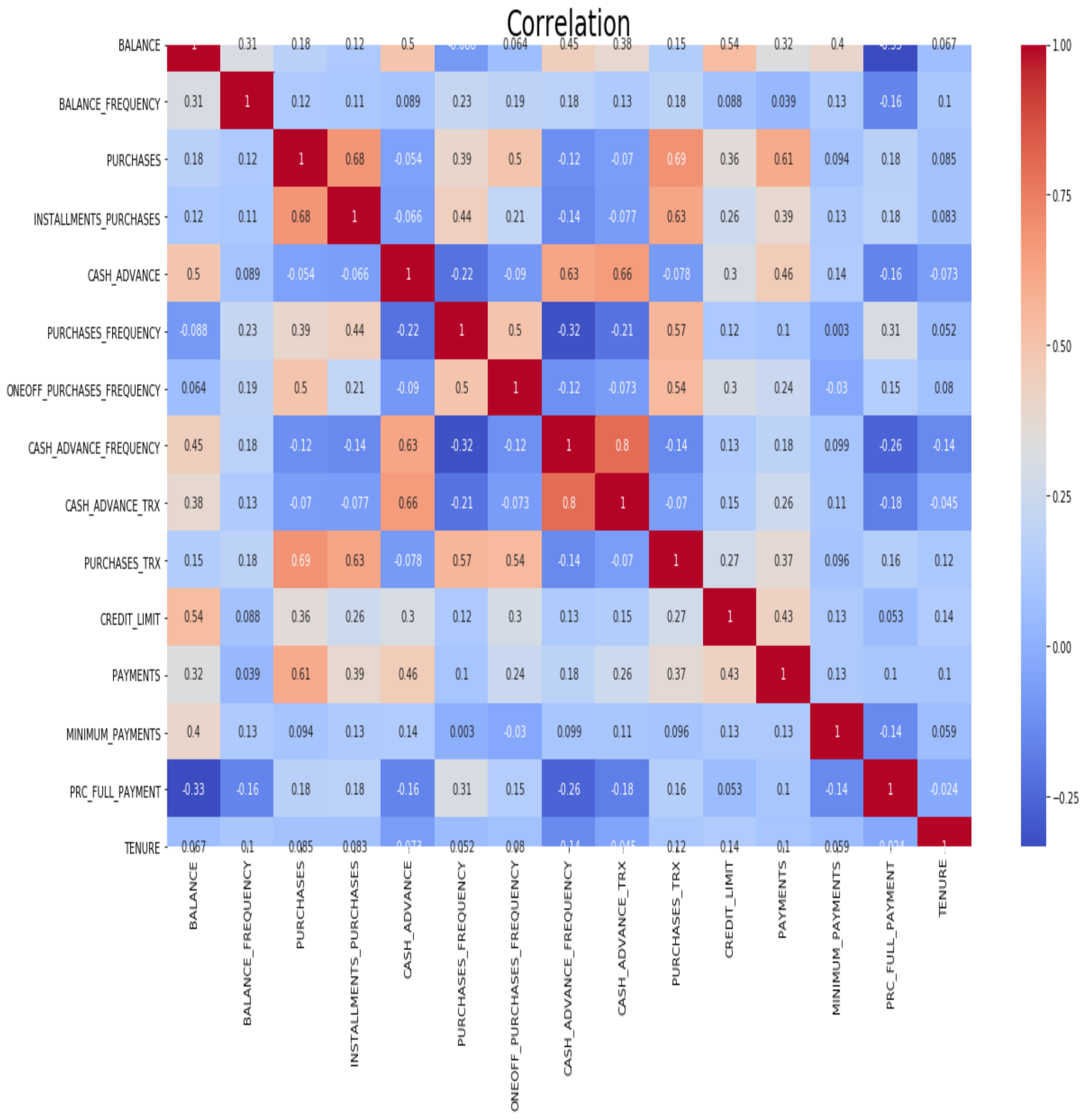
There is too much of data in outlier, deleting them is not an efficient way, because we will lose lot of data.

2.1.3 <u>Feature Selection</u>

For modelling, we have to select the columns or independent variables that will highly contribute to the dependent variable.

If there is no dependent variable, we have to find the collinearity between all the variables and if there is high collinearity between two variables, we can remove any one variable and keep the other. For Example, in our case, PURCHASES_FREQUENCY and PURCHASES_INSTALLMENTS_FREQUENCY have high correlation. Hence, we remove PURCHASES_INSTALLMENTS_FREQUENCY.

Correlation plot:



2.2.4 <u>Feature Scaling</u>

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data pre-processing step. Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization. For example, the majority of classifiers calculate the distance between two points by the Euclidean distance. If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. Since our data is not uniformly distributed as we have seen in (**2.1 Pre-Processing**) we have used **Normalization** as Feature Scaling Method.

2.2 Modeling

1. Clustering or Segmentation or Model Creation

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups.

There are different clustering algorithms. Here we use **K-Means** clustering.

K-means algorithm is an iterative algorithm that tries to partition the dataset into **K** pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to **only one group.** It tries to make the inter-cluster data points as similar as possible while also keeping the clusters as different as possible.

The way K-means algorithm works is as follows:
1. Specify number of clusters *K*.
2. Initialize centroids by first shuffling the dataset and then randomly selecting *K* data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.
4. Compute the sum of the squared distance between data points and all centroids.
5. Assign each data point to the closest cluster (centroid).
6. Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

Chapter 3

Conclusion

3.1 Model Evaluation

We use **Elbow method** to Evaluate the number of clusters(K) to used.
Elbow method gives us an idea on what a good K number of clusters would be based on the sum of squared errors (SSE) between data points and their assigned clusters centroids.

The below images show the elbow graph for our data in Python.

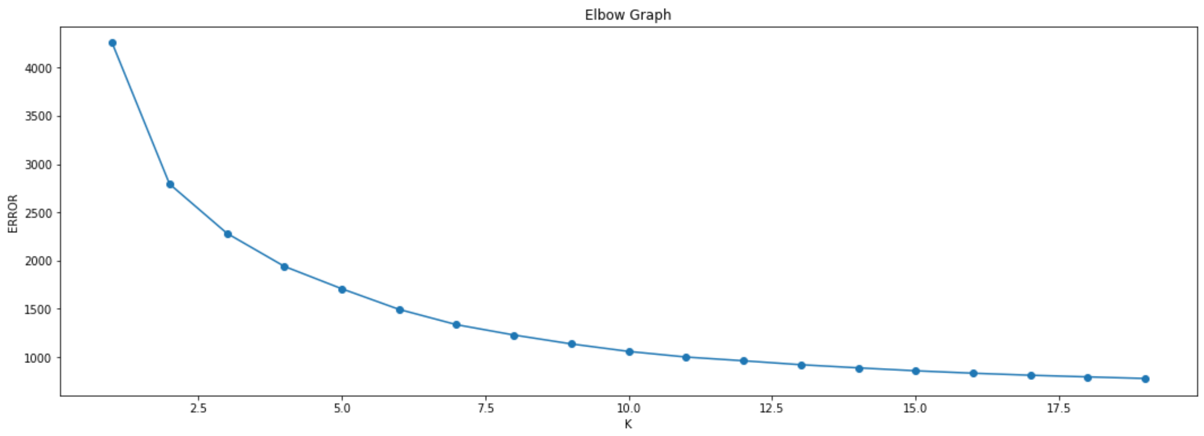


Fig: Elbow Graph for number of Clusters

From the graph, we can select 4 as the optimum number of clusters because, either 4 or 20 clusters, the error difference is very less. But when compared to 2 and 20 clusters, the error difference is more.

Finally, for the cluster of 4, our data is split as below.

Cluster Number	Number of Clients/Customers
0	3900
1	974
2	2487
3	1275

Table: Cluster VS Number of Clients

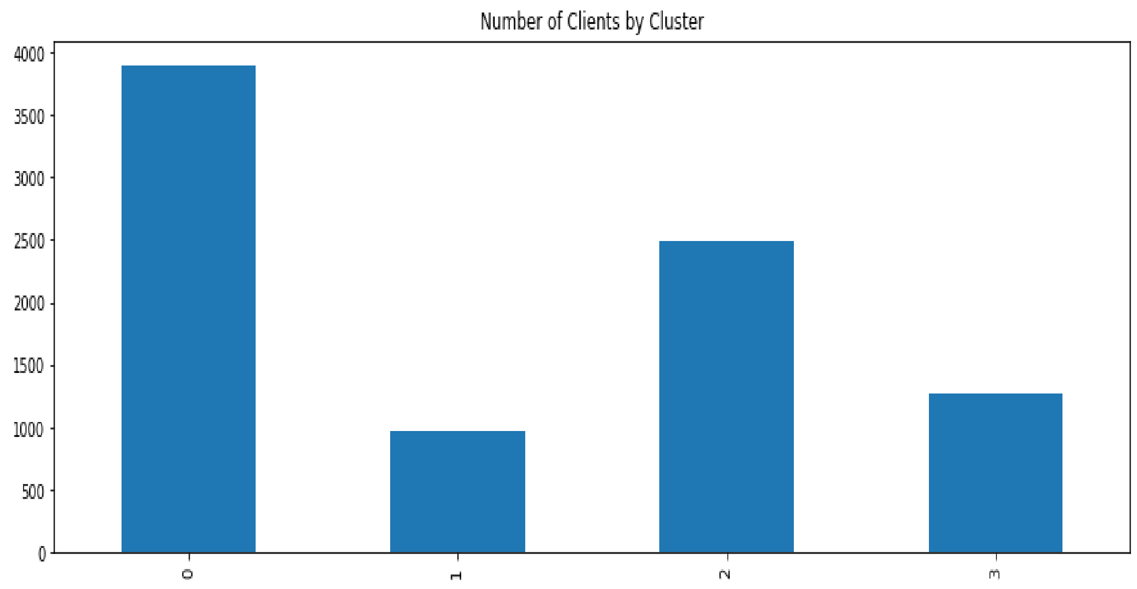


Fig: Clusters VS Number of Clients

In machine learning algorithms, the output will be different each time when we run the code and same cluster won't have same number of count each time. Finally, from the clusters, we can gain insights and derive suggested marketing strategies

Insights and Suggested Marketing Strategies from Clusters

Cluster 0: (Count: 3900)

Insight: These customers have maximum Average Purchase and good Monthly cash advance but this cluster doesn't do frequent instalment or one-off purchases.

Marketing Strategy: They are potential target customers who are paying dues and doing purchases. We can increase credit limit or can lower down interest rate, can be given premium card or loyalty cards to increase transactions

Cluster 1: (Count: 974)

Insight: They do very less One-off purchases and less cash advance and maintain very less Balance.

Marketing Strategy: We can target them by giving them more offers.

Cluster 2: (Count: 2487)

Insight: This group of customers who have highest Monthly cash advance and doing both instalment as well as one-off purchases, have comparatively good credit score but have poor average purchase score.

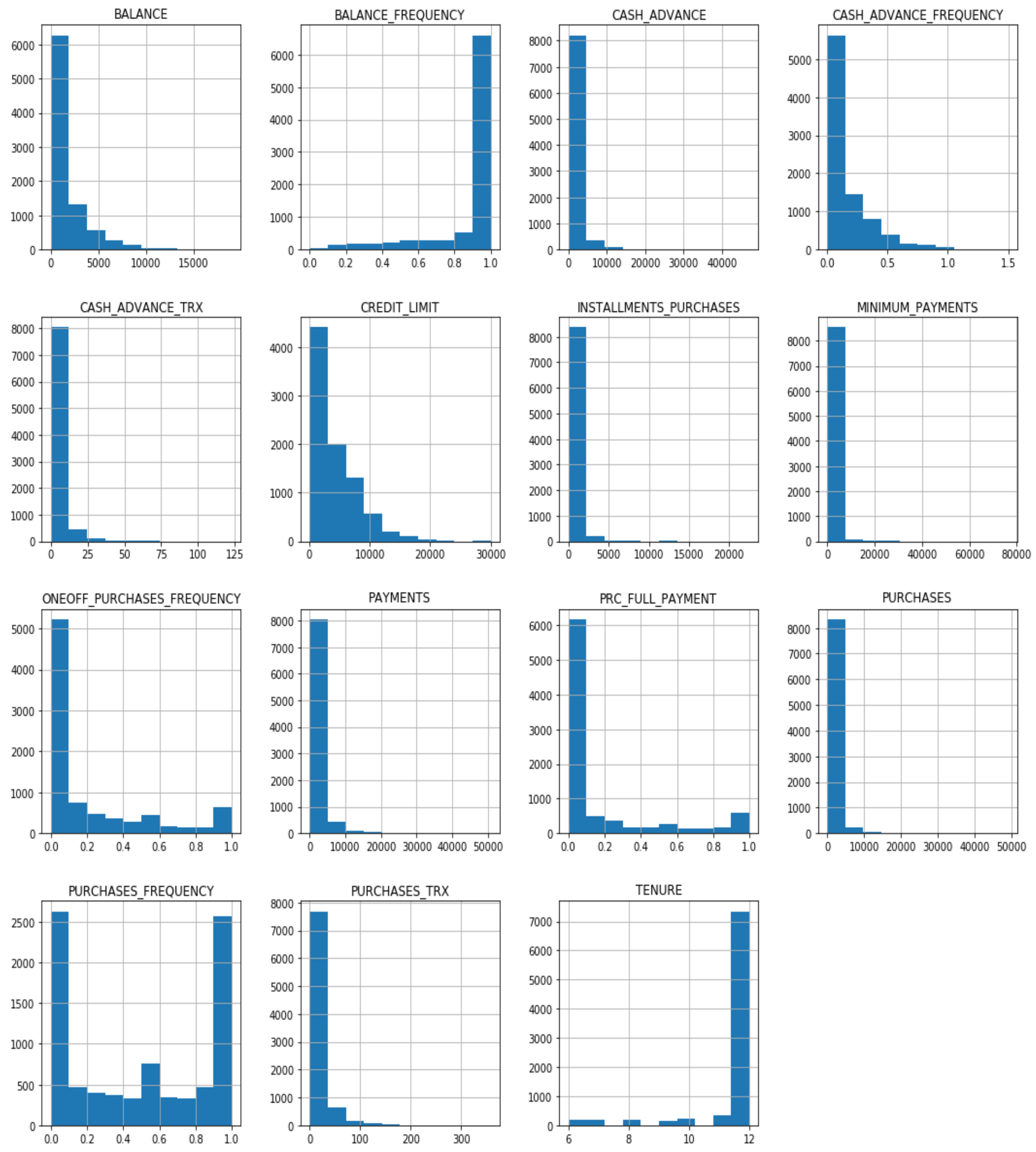
Marketing Strategy: They take only cash on advance. We can target them by providing less interest rate on purchase transaction.

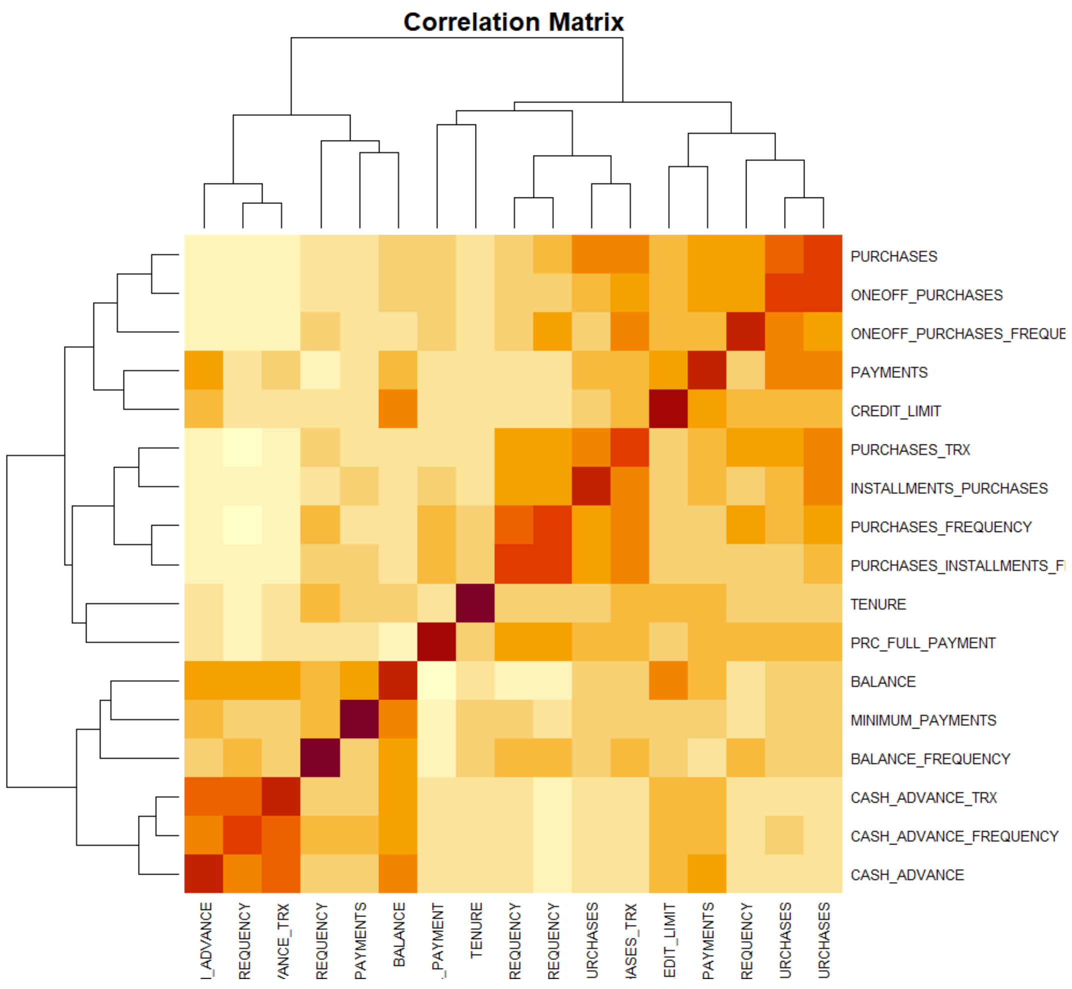
Cluster 3: (Count: 1275)

Insight: Customers are doing maximum One-off transactions and has least payment ratio amongst all the cluster.

Marketing Strategy: 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

Extra Figures :-





CORELATION MATRIX IN R

INSTRUCTIONS TO RUN THE CODE :

STEPS TO RUN PYTHON CODE :-

Step - 1 : Install Anaconda Framework, with Python 3

Step - 2 : Open anaconda prompt, and install the required packages using the command 'pip install <required_package>'

Step - 3 : Using the anaconda prompt, Change to the Drive in which your python file is present.

Step - 4 : In the same prompt, enter 'jupyter notebook' to open the Python Jupyter Notebook.

Step - 5 : Once jupyter is open, migrate to the folder in which your Python code is present and open it.

Step - 6 : Press [ctrl + Enter] or [shift + Enter] to execute each cell. [Again depends on system/application settings]

STEPS TO RUN R CODE :-

Step - 1 : Install RStudio.

Step - 2 : Install the required packages. Code:
install.packages(<required_package>)

Step - 3 : Open the file "Credit_Card_Segmentation_R.R" in RStudio.

Step - 4 : Press [ctrl + Enter] (in windows) to execute each line of the code.
[Depends on the system/application settings]

PYTHON CODE :-

CREDIT CARD SEGMENTATION PROJECT NO - 2

IMPORT THE LIBRARIES

Import the libraries you usually use for data analysis.

```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Get the Data

```
# Change the  working directory
os.chdir("C:/Users/Bustec/Desktop/project2/New folder")
```

Read in the College_Data file using read_csv. Figure out how to set the first column as the index.

```
data = pd.read_csv('creditcard.csv')
```

Check the head of the data

```
data.head()


data.shape


(8950, 18)

data.describe()
data.dtypes
```

Missing Value Analysis

```
def CalMissingValues(data):
    return data.isna().sum()


CalMissingValues(data)
# Calculate the Percentage of  the Missing Values in the dataset.

(CalMissingValues(data).sum()/data.shape[0])*100

#Which means 3.5% rows of the data contains the null values. We can i
mpute them.
#else we can ignore them by not considering for analysis..
```

droping the null values

```
data = data.dropna(axis=0)
```

```
data.shape
```

Deriving the KPI's

Monthly Average Purchase & Cash Advanced Amount

```
data_KPI = pd.DataFrame()
data_KPI['CUST_ID'] = data['CUST_ID']

data_KPI['Monthly_Average_Purchase'] = data['PURCHASES']/data['TENURE']
data_KPI['Monthly_Cash_Advance'] = data['CASH_ADVANCE']/data['TENURE']
```

Purchases by the Type

we can see that there are two types of purchases in the Data. ONE_OFF_PURCHASE and INSTALLMENT_PURCHASE

```
a = data[(data['ONEOFF_PURCHASES'] == 0) & (data['INSTALLMENTS_PURCHASES'
] == 0)]
aa = a.shape

print('There are ' + str(aa[0]) + ' clients who do not make any type of p
urchases.')
There are 1965 clients who do not make any type of purchases.

b = data[(data['ONEOFF_PURCHASES'] > 0) & (data['INSTALLMENTS_PURCHASES']
> 0)]
bb = b.shape

print("There are " + str(bb[0]) + " clients who make use of both the purc
hases according to the data.")
There are 2741 clients who make use of both the purchases according to th
e data.

c = data[(data['ONEOFF_PURCHASES'] > 0) & (data['INSTALLMENTS_PURCHASES']
== 0)]
cc = c.shape
print(str(cc[0]) + " clients make use of only ONEOFF_PURCHASES")
1782 clients make use of only ONEOFF_PURCHASES

d = data[(data['ONEOFF_PURCHASES'] == 0) & (data['INSTALLMENTS_PURCHASES'
] > 0)]
dd = d.shape
print(str(dd[0]) + " clients make use of INSTALLMENTS_PURCHASES")
2148 clients make use of INSTALLMENTS_PURCHASES
```

We found that there are FOUR types of Purchase behaviour in the customers according to Data.

1. People who does not make any purchases.
2. People who make the both types of purchases.
3. People who make only the OneOff_Purchases
4. People who make only the Installment_Purchases

```
#User Defined Function to catogorise the purchase behaviour of the
customers

def purchaseType(data):
    if (data['ONEOFF_PURCHASES'] == 0) & (data['INSTALLMENTS_PURCHASES']
> 0) :
        return 'installment'
    if (data['ONEOFF_PURCHASES'] > 0) & (data['INSTALLMENTS_PURCHASES']
== 0) :
        return 'one_off'
    if (data['ONEOFF_PURCHASES'] > 0) & (data['INSTALLMENTS_PURCHASES'] >
0) :
        return 'both'
    if (data['ONEOFF_PURCHASES'] == 0) & (data['INSTALLMENTS_PURCHASES']
== 0) :
        return 'not any'
data_KPI['Purchase_Type'] = data.apply(purchaseType, axis = 1)
data_KPI
```

```
Purchase_Types = data_KPI['Purchase_Type'].value_counts()
Purchase_Types
```

```
Purchase_Types.plot.bar()
```

The Average Amount per Purchase and the Average Cash Advanced Transaction

```
AvgAmtPerPurchase = data[['CUST_ID', 'PURCHASES_TRX']]
print(AvgAmtPerPurchase.describe())
```

```
AvgAmtPerPurchase[AvgAmtPerPurchase['PURCHASES_TRX'] > AvgAmtPerPurch
ase['PURCHASES_TRX'].quantile(0.75)]
```

```
AvgCashAdvance = data[['CUST_ID', 'CASH_ADVANCE_TRX']]
print(AvgCashAdvance.describe())
```

```
AvgCashAdvance[AvgCashAdvance['CASH_ADVANCE_TRX'] > AvgCashAdvance['C
ASH_ADVANCE_TRX'].quantile(0.75)]
```

```
#LIMIT USAGE
data_KPI['Limit_Usage'] = data['BALANCE']/data['CREDIT_LIMIT']
data_KPI['Limit_Usage']
```

```
#Payments to minimum payments Ratio
data_KPI['Pymt_to_minPymt_Ratio'] = data['PAYMENTS']/data['MINIMUM_PA
YMENTS']
data_KPI['Pymt_to_minPymt_Ratio']
```

VIEW THE INSIGHTS FROM THE NEW KPIS

```
data_KPI.head()
```

```
# Average Payment_minPayment ratio for each purchase type
x = data_KPI.groupby('Purchase_Type').apply(lambda x : np.mean(x['Pym
t_to_minPymt_Ratio']))
x
```

```
x.plot.bar()
plt.title('Average Payment_minPayment ratio for  each  of the purchas
e type')
```

Clients with the Purchase type Installment are the ones with more Dues.

```
# Average Limit Usage for the each purchase type
x1 = data_KPI.groupby('Purchase_Type').apply(lambda x : np.mean(x['Limit_
Usage']))
x1
```

```
x1.plot.bar()
plt.title('Average Limit Usage for each purchase type')
```

Clients with the Purchase type Installment have the good Credit Score

```
x2 = data_KPI.groupby('Purchase_Type').apply(lambda x : np.mean(x['Monthl
y_Cash_Advance']))
x2
```

```
x2.plot.bar()
plt.title('Average Monthly Cash Advance for each purchase type')
```

Clients who don't do the any type of purchases(OneOff, Installments), Take more Cash On the Advance

Performing the Outlier Analysis

```
data.columns
```

```
data[['BALANCE', 'BALANCE_FREQUENCY', 'PURCHASES','ONEOFF_PURCHASES',
'INSTALLMENTS_PURCHASES', 'CASH_ADVANCE',
       'PURCHASES_FREQUENCY', 'ONEOFF_PURCHASES_FREQUENCY',
       'PURCHASES_INSTALLMENTS_FREQUENCY', 'CASH_ADVANCE_FREQUENCY',
       'CASH_ADVANCE_TRX', 'PURCHASES_TRX', 'CREDIT_LIMIT', 'PAYMENTS
',
      'MINIMUM_PAYMENTS', 'PRC_FULL_PAYMENT', 'TENURE']].plot.box(fig
size=(17,10),title='Outlier Distribution')
```

```
data[['BALANCE', 'BALANCE_FREQUENCY', 'PURCHASES',
       'ONEOFF_PURCHASES', 'INSTALLMENTS_PURCHASES',
      'CASH_ADVANCE']].plot.box(figsize=(17,10),legend=True, title='O
utlier Distribution')
```

```
data[['PURCHASES_FREQUENCY', 'ONEOFF_PURCHASES_FREQUENCY',
       'PURCHASES_INSTALLMENTS_FREQUENCY', 'CASH_ADVANCE_FREQUENCY',
       'CASH_ADVANCE_TRX']].plot.box(figsize=(17,10),legend=True, tit
le='Outlier Distribution')
```

```
data[['PURCHASES_TRX', 'CREDIT_LIMIT', 'PAYMENTS',
       'MINIMUM_PAYMENTS', 'PRC_FULL_PAYMENT', 'TENURE']].plot.box(fi
gsize=(17,10),legend=True, title='Outlier Distribution')
```

Feature Selection

```
correlation_matrix = data.corr()
correlation_matrix = pd.DataFrame(correlation_matrix)

plt.figure(figsize=(20,10))
sns.heatmap(correlation_matrix, cmap='coolwarm', annot=True)
plt.title('Correlation', size = 25)
```

```
correlation_matrix.abs()
correlation_matrix[correlation_matrix == 1] = 0
correlation_columns = correlation_matrix.max().sort_values(ascending=
False)
print(correlation_columns[correlation_columns > 0.8])
```

```
copyOfData = data.copy()
#data = copyOfData.copy()
data.copy()
```

```
# Removing the columns ONEOFF_PURCHASES and PURCHASES_INSTALLMENTS_FREQUE
NCY
data = data.drop(columns=["CUST_ID","ONEOFF_PURCHASES","PURCHASES_INSTALL
MENTS_FREQUENCY"])
data
```

Feature Scaling

```
data.hist(figsize=(18,18))
```

The data is not normally or uniformly distributed. We have to normalize the dataset.

```
cnames = data.columns
cnames
```

```
# Normalize the data  in column wise.
for i in cnames:
    data[i] = (data[i] - data[i].min())/(data[i].max() - data[i].min(
))

data
```

Performing the Cluster Analysis

```
from sklearn.cluster import KMeans

# Estimate Optimum number of clusters
cluster_range = range(1, 20)
cluster_errors = []

for noOfClusters in cluster_range:
    clusters = KMeans(noOfClusters).fit(data)
    cluster_errors.append(clusters.inertia_)

cluster_DF = pd.DataFrame( {"noOfClusters": cluster_range, "cluster_error
s": cluster_errors})
cluster_DF
```

PLOT THE ELBOW GRAPH

```
plt.figure(figsize=(18, 6))
plt.plot(cluster_DF.noOfClusters, cluster_DF.cluster_errors, marker = 'o'
)
plt.xlabel('K')
plt.ylabel('ERROR')
plt.title('Elbow Graph')
```

Create the KMeans Model

```
# Size of the Cluster
cluster_size = 4

# Creating the Model
kmeans_model = KMeans(n_clusters = cluster_size).fit(data)

kmeans_model.labels_

pd.crosstab(copyOfData['CUST_ID'], kmeans_model.labels_)

# Merge the Actual data and Derived KPI's into one Table

#columns_to_merge = data_KPI.columns.difference(copyOfData.columns)
cluster_data = pd.merge(copyOfData, data_KPI, on = 'CUST_ID')
cluster_data['cluster'] = kmeans_model.labels_
cluster_data

cluster_data = cluster_data.sort_values(by = 'cluster')
cluster_data
```

```
# Value Counts
print(cluster_data['cluster'].value_counts().sort_index())

# Bar Graph showing Number of clients by Clusters
cluster_data['cluster'].value_counts().sort_index().plot.bar(figsize = (1
5, 5), title='Number of Clients by Cluster')

# Seperating Clients belonging to Different Clusters
for i in range(0, cluster_size):
    globals()['cluster_%s' % i] = pd.DataFrame(cluster_data[cluster_data[
'cluster'] == i])

cluster_0.describe()

cluster_1.describe()

cluster_2.describe()

cluster_3.describe()
```

R CODE :-

```
# Remove all variables
rm(list=ls())

# Load libraries
library(tidyverse)
library(dplyr)

# Set Working Directory
setwd("C:/Users/Bustec/Desktop/project2/New folder")

# Load Data
credit_data = read.csv('credit-card-data.csv')

# Dimensions of the data
dim(credit_data)

# Structure of credit_data
str(credit_data)

# Summary
```

```r
summary(credit_data)

#Missing Value Analysis #

calMissingValues <- function(data){
  missing_values <<- data.frame(apply(data, 2, function(x)
sum(is.na(x))))
  missing_values$Columns <<- row.names(missing_values)
  row.names(missing_values) <<- NULL
  names(missing_values)[1] <<- 'Num_Of_Missing_Values'
  missing_values <<- missing_values[order(-
missing_values$Num_Of_Missing_Values),]
  missing_values <<- missing_values[, c(2, 1)]
}

# See sum of Missing values in each column
View(calMissingValues(credit_data), title = 'credit_data_Missing_Values')

# Calculate missing value percentage in minimum_payments column
(sum(is.na(credit_data$MINIMUM_PAYMENTS))/nrow(credit_data)) * 100
# Since only 3.4% ie., 313 rows out of 8950 rows contain missing values.
# we can impute them or delete the respective rows. For now lets delete
credit_data = na.omit(credit_data)

# Check the missing values
View(calMissingValues(credit_data), title = 'credit_data_Missing_Values')

#Deriving KPI's #

data_KPI = data.frame(credit_data['CUST_ID'])

# Monthly Average Purchase
data_KPI['Monthly_Average_Purchases'] =
credit_data$PURCHASES/credit_data$TENURE

# Monthly Cash Advanced Amount
data_KPI['Monthly_Cash_Advanced_Amount'] =
credit_data$CASH_ADVANCE/credit_data$TENURE

# Purchase Type
# Seeing the data, we can infer that there are two types of Purchases.
ONE_OFF_PURCHASE and INSTALLMENT_PURCHASE
# Let us Explore little more
# %>% can be found in tidyverse package.
aa = credit_data %>% filter(credit_data$ONEOFF_PURCHASES == 0,
credit_data$INSTALLMENTS_PURCHASES == 0)
dim(aa)
cat('Therefore', dim(aa)[1], 'Clients use either OneOff_Purchases or
Installment_Purchases.')

bb = credit_data %>% filter(credit_data$ONEOFF_PURCHASES > 0,
credit_data$INSTALLMENTS_PURCHASES == 0)
dim(bb)
cat('There are', dim(bb)[1], 'Clients use only OneOff_Purchases.')

cc = credit_data %>% filter(credit_data$ONEOFF_PURCHASES == 0,
credit_data$INSTALLMENTS_PURCHASES > 0)
dim(cc)
cat('Therefore', dim(cc)[1], 'Clients use only Installment_Purchases.')

dd = credit_data %>% filter(credit_data$ONEOFF_PURCHASES > 0,
credit_data$INSTALLMENTS_PURCHASES > 0)
dim(dd)
```

```
cat('Therefore', dim(dd)[1], 'Clients use BOTH OneOff_Purchases or
Installment_Purchases.')

# Result: We found that there are FOUR types of Purchase behaviour in the
data.
# 1. People who does not make any purchases.
# 2. People who make both types of purchases.
# 3. People who make only OneOff_Purchases
# 4. People who make only Installment_Purchases

View(calMissingValues(data_KPI), title = 'data_KPI_Missing_Values')

# User Defined function to catogorise the Purchase behaviour of the data
purchaseType <- function(credit_data){
  if ((credit_data['ONEOFF_PURCHASES'] == 0) &&
(credit_data['INSTALLMENTS_PURCHASES'] == 0)){
    return <- 'Not_Any'
  }
  else if ((credit_data['ONEOFF_PURCHASES'] > 0) &&
(credit_data['INSTALLMENTS_PURCHASES'] == 0)){
    return <- 'Only_OneOff'
  }
  else if((credit_data['ONEOFF_PURCHASES'] == 0) &&
(credit_data['INSTALLMENTS_PURCHASES'] > 0)){
    return <- 'Only_Installments'
  }
  else if((credit_data['ONEOFF_PURCHASES'] > 0) &&
(credit_data['INSTALLMENTS_PURCHASES'] > 0)){
    return <- 'Both'
  }
}

for (row in 1:nrow(credit_data)){
  data_KPI$Purchase_Type[row] = purchaseType(credit_data[row,])
}

View(data_KPI, title = 'data_KPI')

View(table(data_KPI$Purchase_Type), title = 'GroupBy_PurchaseType')

# Plotting a bar graph for Purchase Type vs Count
counts <- table(data_KPI$Purchase_Type)
barplot(counts, main = 'Distribution for Purchase Type', xlab='Purchase
Type', ylab='Count')

# Limit Usage [BALANCE to CREDIT_LIMIT Ratio]
data_KPI$Limit_Usage = credit_data$BALANCE/credit_data$CREDIT_LIMIT

# Payments to Minimum_Payments Ratoi
data_KPI$Payments_to_Minimum_payments_Ratio =
credit_data$PAYMENTS/credit_data$MINIMUM_PAYMENTS

# Calculate the missing values
View(calMissingValues(data_KPI))

# If there are any NULL/NA/NaN values in data_KPI. Run the below TWO
lines of code, and check the missing values.
# Filling NaN values with Zero(0)
data_KPI$Monthly_Average_Purchases[is.na(data_KPI$Monthly_Average_Purchas
es)] <- 0
data_KPI$Monthly_Cash_Advanced_Amount[is.na(data_KPI$Monthly_Cash_Advance
d_Amount)] <- 0

# Checking for missing values
```

```
View(calMissingValues(data_KPI), title = 'data_KPI_Missing_Values')

# Final Values of data_KPI
View(data_KPI, title = 'data_KPI')
```

#Insights From New KPI's#

```
# Average Payment to minimum Payment Ratio for each Purchase Type
a1 <- data.frame(data_KPI %>% group_by(.$Purchase_Type) %>%
summarise(Avg_Pymt_minPymt_Ratio =
mean(Payments_to_Minimum_payments_Ratio)))
# Ploting Bar Graph for Average Payment to minimum Payment Ratio VS
Purchase Type
barplot(unlist(a1[2]), names.arg = unlist(a1[1]), xlab='Purchase Type',
ylab='Average Payment to minimum Payment Ratio', main = 'Average Payment
to minimum Payment Ratio for each Purchase Type')
# Insight 1: Clients with Purchase Type Installment are with more dues.

# Average Limit Usage for each Purchase Type
a2 = data.frame(data_KPI %>% group_by(.$Purchase_Type) %>%
summarise(Avg_Limit_Usage = mean(Limit_Usage)))
# Plotting Bar Graph for Average Limit Usage VS Purchase Type
barplot(unlist(a2[2]), names.arg = unlist(a2[1]), xlab = 'Purchase Type',
ylab = 'Average Limit Usage', main = 'Average Limit Usage for each
Purchase Type')
# Insight 2: Clients with Purchase type Installment have good Credit
Score

# Average Monthly Cash Advance for each Purchase Type
a3 = data.frame(data_KPI %>% group_by(.$Purchase_Type) %>%
summarise(Monthly_Cash_Advance__ = mean(Monthly_Cash_Advanced_Amount)))
# Plotting Bar Graph for Average Monthly Cash Advance VS Purchase Type
barplot(unlist(a3[2]), names.arg = unlist(a3[1]), xlab = 'Purchase Type',
ylab = 'Average Monthly Cash Advance', main = 'Average Monthly Cash
Advance for each Purchase Type')
# Insight 3: Clients who dont do any type of Purchases(OneOff,
Installment), Take more Cash on Advance.
```

#Outliers Analysis#

```
# boxplot for credit_data
boxplot(credit_data, horizontal = F)

# col_names = names(credit_data)
# removing outliers for variables
# for(i in col_names){
#   val = credit_data[,i][credit_data[,i] %in%
boxplot.stats(credit_data[,i])$out]
#   credit_data = credit_data[which(!credit_data[,i] %in% val), ]
# }

print('Since it removes 90% of the data, lets not remove it.')
```

#Feature Selection #

```
# Selecting features for Clustering
library(caret)
correlation_matrix = cor(credit_data[,2:ncol(credit_data)])
heatmap(correlation_matrix, main = 'Correlation Matrix')
highly_corelated = findCorrelation(correlation_matrix, cutoff=0.8,
verbose = T, names = FALSE)

View(credit_data[,c('PURCHASES_FREQUENCY',
'PURCHASES_INSTALLMENTS_FREQUENCY')])
```

```
View(credit_data[,c('PURCHASES', 'ONEOFF_PURCHASES')])

# Since PURCHASES_FREQUENCY and PURCHASES_INSTALLMENTS_FREQUENCY have
high correlation, remove any one.
# Sameway.. Since PURCHASES and ONEOFF_PURCHASES has high correlation,
remove any one.
# Also remove 1st column since it is customer_ID and has no dependency on
clustering
# Index 1 = CUST_ID, Index 5 = ONEOFF_PURCHASE, Index 10 =
PURCHASES_INSTALLMENTS_FREQUENCY
cnames = names(credit_data)[-c(1,5,10)]

credit_data_ForClustering = credit_data[, cnames]
```

#Feature Scaling #

```
# Plotting histogram for each column
for(i in cnames){
  hist(credit_data_ForClustering[, i])
}

# Since data is nor uniformly/normally distributed, we apply
normalization
# First convert all columns to numeric
credit_data_ForClustering[] <- lapply(credit_data_ForClustering[],
as.numeric) # convert to numeric
str(credit_data_ForClustering) # check whether all columns are converted
to numeric

Normalized_Credit_data_ForClustering <- credit_data_ForClustering
#copying from one variable to another

for(i in cnames){
  print(i)
  Normalized_Credit_data_ForClustering[,i] =
(Normalized_Credit_data_ForClustering[,i] -
min(Normalized_Credit_data_ForClustering[,i]))/
    (max(Normalized_Credit_data_ForClustering[,i]) -
min(Normalized_Credit_data_ForClustering[,i]))
}

View(calMissingValues(Normalized_Credit_data_ForClustering),
title='Normalized Values')
```

#Cluster Analysis#

```
library(NbClust)

# NbClust is used to find the Optimal number of clusters for k-means
NB_Clust_Result =
NbClust(Normalized_Credit_data_ForClustering[,2:ncol(Normalized_Credit_da
ta_ForClustering)], min.nc = 2, max.nc = 15, method = 'kmeans')

# K-means clustering
NumberOfClusters = 4
kmeans_model = kmeans(Normalized_Credit_data_ForClustering,
NumberOfClusters, nstart = 25)

credit_data$CLUSTER = kmeans_model$cluster

credit_data = credit_data[, c(1, 19, 2:18)] # re-arranging columns
credit_data = credit_data[order(credit_data$CLUSTER), ] # re-arranging
rows

# Number of Clients is each cluster
table(credit_data$CLUSTER)

for(i in 1:(NumberOfClusters+1)){
  assign(paste0("Cluster_", i), credit_data[credit_data['CLUSTER'] ==
i,])
}

# we can use the summary to gain information on each cluster.
summary(Cluster_1)
summary(Cluster_2)
summary(Cluster_3)
summary(Cluster_4)
```

References:-

1. For Data Cleaning and Model Development - https://edwisor.com/career-data-scientist

2. For Visualization – https://www.udemy.com/python-for-data-science-and-machine-learning-bootcamp/

THANK YOU