**CREDIT CARD**

**SEGMENTATION**

**Dudde Dharmatej**

5th April 2020

**Contents:**

1. Introduction
   1. Problem statement………………………………………….03
   2. Data………………………………………...………………..03
2. Methodology
   1. Exploratory data analysis...……………..…………………..05
   2. Data preprocessing………………………………..………...05
      1. Missing Value analysis………………….…………….06
      2. Data Scaling & Outlier analysis………..………….......08
      3. Feature selection……………………………….……...09
         1. KPI……………………………………………....09
         2. Principle Component Analysis(PCA):………….……13
   3. Modeling

2.3.1K-Means clustering……………………….…………….....14

1. Conclusion

3.1 Model Evaluation…………………………...………............17

3.1.1 Visualizing clusters………………….…………..…17

3.1.2 Silhouette Score…………………...……………….18

3.2 Cluster Interpretation……………………….…………..….19

4. Visualizations………………..…………………….………...…23

5. Code…………………….…………………………………....…25

**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.

**1.2 Data:**

**Number of attributes: 18**

**Number of Rows: 8950**

**Variable details:**

● 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

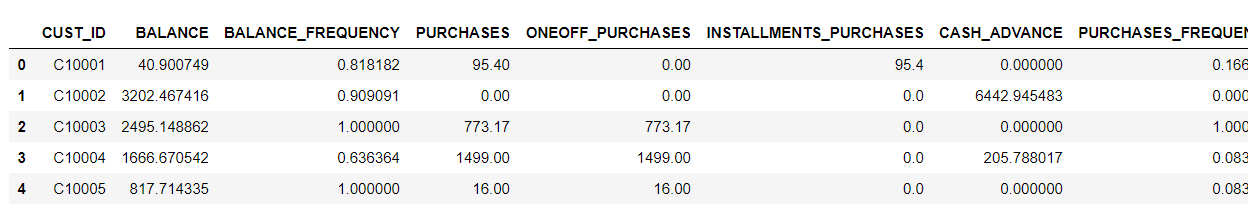
● MINIMUM\_PAYMENTS Total minimum payments due in the period.

● PRC\_FULL\_PAYMENT- Percentage of months with full payment of the due

Statement balance

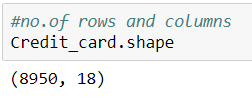
● TENURE Number of months as a customer

**Sample Data:**

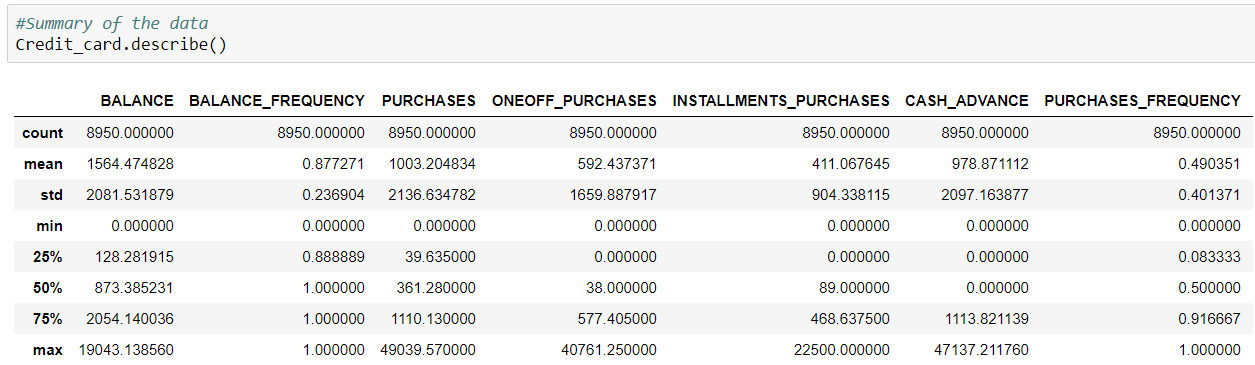
****

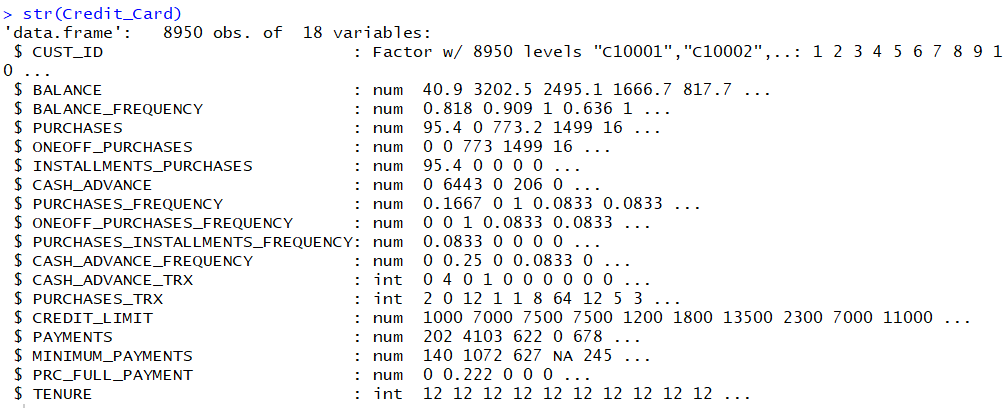
**2. Methodology:**

**2.1 Exploratory Data Analysis:**

Given data have dimensions of 8950 rows and 18 independent columns.

Summary of given data can be seen below



****

**2.2 Data Preprocessing:**

We can see that the given data only consists of numeric variables and only categorical variable present is CUST\_ID which we will anyways omit as its of no use for analysis and grouping.

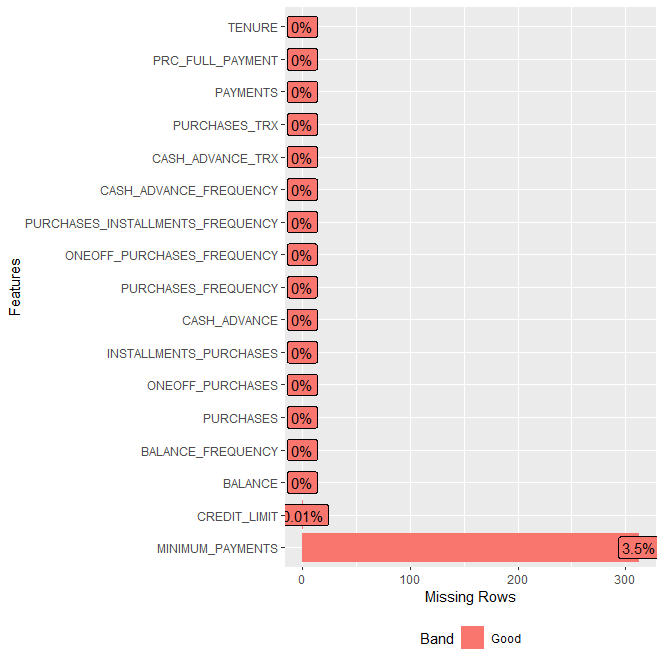
Dimensions of Data after removing CUST\_ID are 8950 X 17

**2.2.1 Missing Value Analysis:**

Missing value analysis is done to check is there any missing value present in given dataset. Missing values can be easily treated using various methods like mean, median method, knn method to impute missing value.

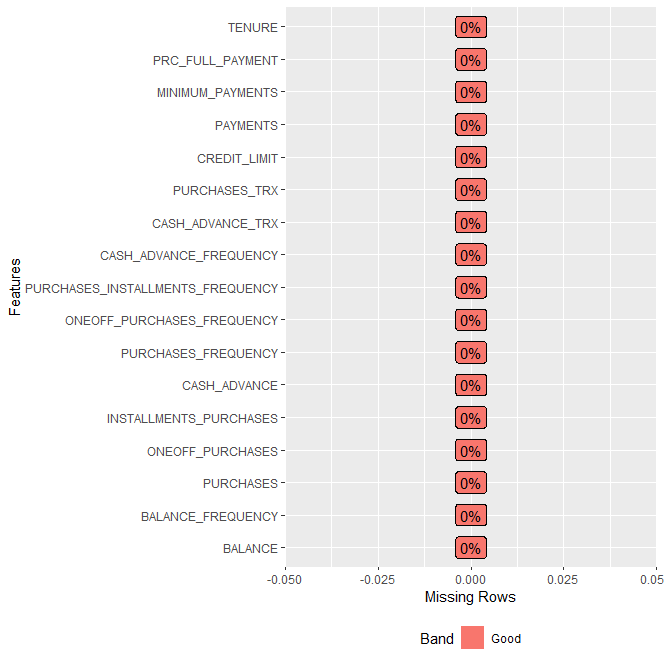
In R function(x){sum(is.na(x))} is the function used to check the sum of missing values.

In python Credit\_card.isnull().sum() is used to detect any missing value



From the plot we can see that only one value missing from CREDIT\_LIMIT and its 0.01% and then MINIMUM\_PAYMENTS has 3.5% of missing values. Which means its better to impute the values using mean value of the respective variable in this case.

After imputing the missing values with mean values, we can see that there are no more missing values.

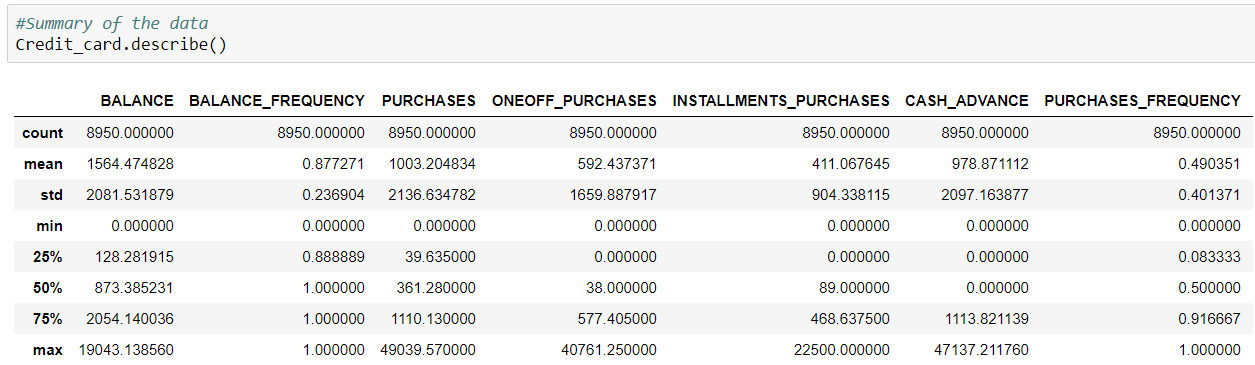


**2.2.2 Data Scaling & Outlier analysis:**

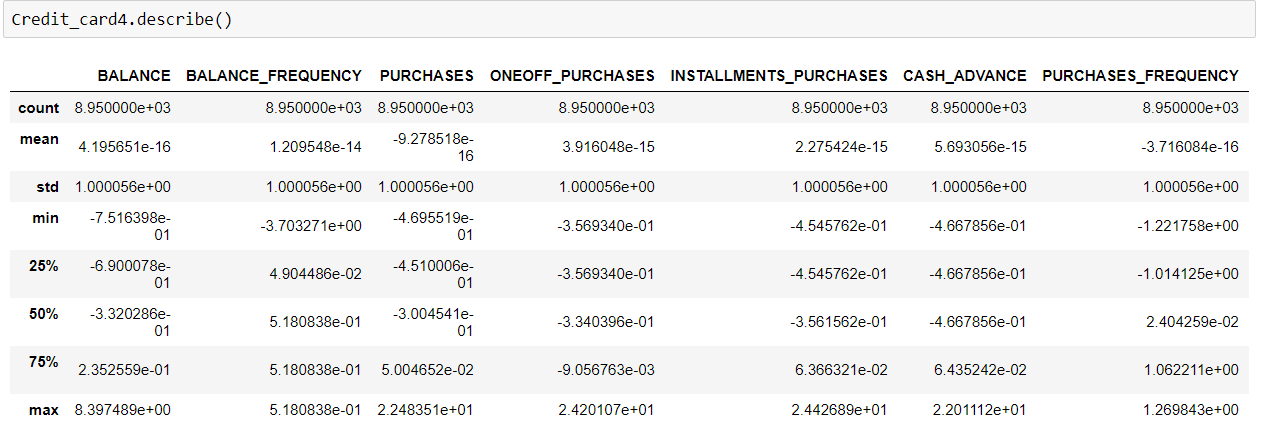
Some variables are of range 0 to 1 and few are of the range 0 to 30000, hence data scaling is important for grouping the given data.

Since there are variables having extreme values these can become potential outliers, hence standardizing the data can also reduce the effect of outliers in modeling the data

Summary of data before standardizing data:



Summary of data after standardizing data:

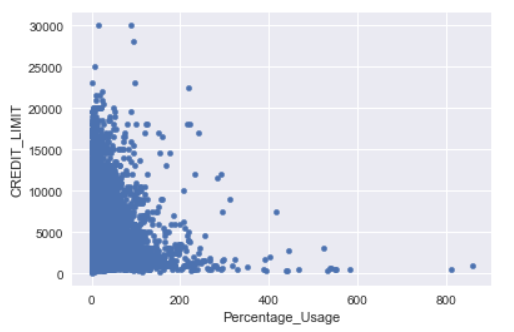


**2.2.3 Feature selection:**

**2.2.3.1 KPI and Visualization:**

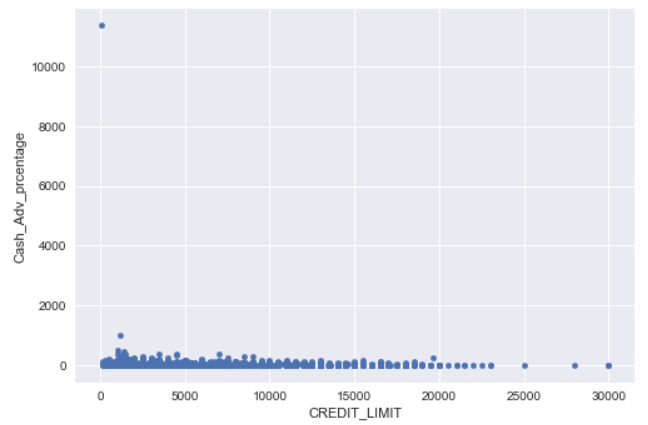
Key performance indicators (KPI) help in analysis by reducing variables or by developing new variables from existing ones.

**Limit Usage Percentage:**



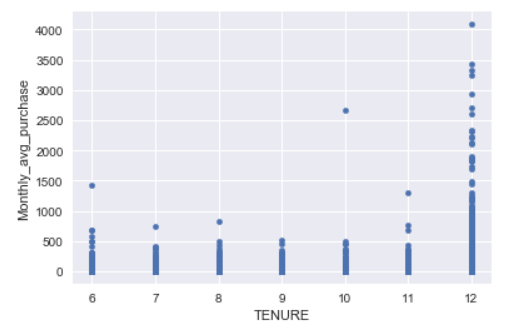
We can see that from the usage plot, people with less credit limit are using the amount to fullest and vice versa.

**Cash Advance percentage:**



From the above cash advance percentage we can see that people with low credit limit tend to take cash advance more than going for purchases.

**Monthly average purchase:**



From the above monthly average purchase graph we can see that people with higher tenure are like to purchase more and use a credit card.

We can see that there are 4 types of purchase behaviors in the customers from the given data set.

1. People who only do One-Off Purchases.

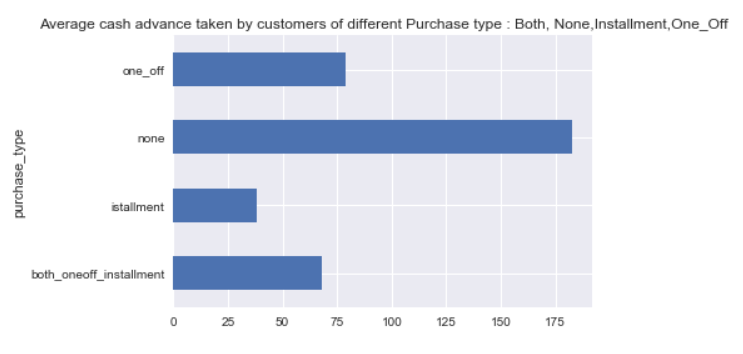
2. People who only do Installments Purchases.

3. People who do both.

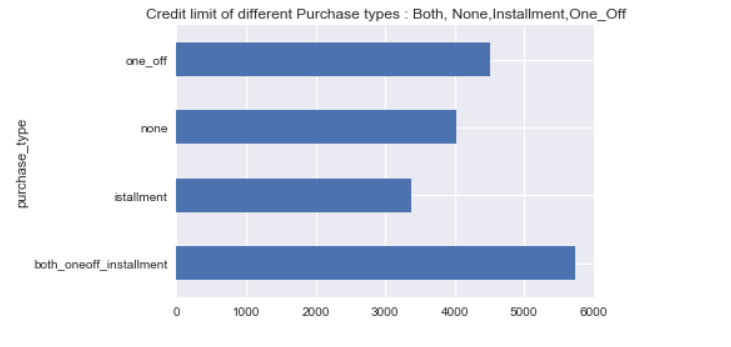
4. People who do none.

So we have derived a categorical variable based on the customer purchase behavior.

**Observations from the derived types of purchases:**

****

From the above graph we can see that the customers who don’t go for either one-off purchases nor installment purchases are likely to take cash advances.



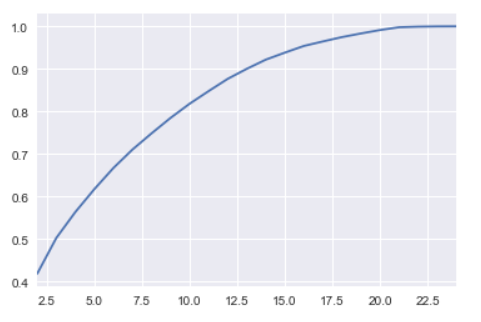
From the above graph we can see that people with high credit limit also have the higher purchases.

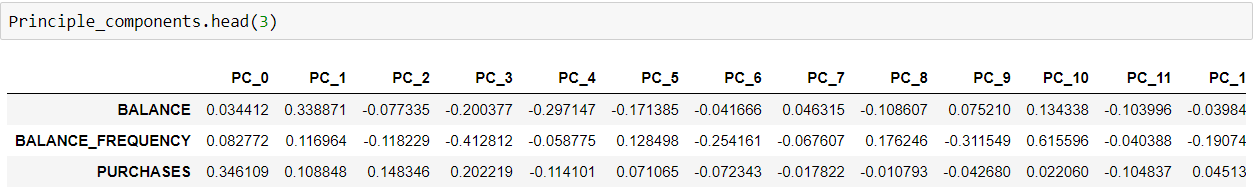
**2.2.3.2 Principle Component Analysis (PCA):**

PCA is a method used to reduce number of variables in the data by extracting important ones from a large pool. It reduces the dimension of data with the aim of retaining as much information as possible.

Let’s apply PCA before going for clustering the data, this also takes care of any correlation existing between the variables

After applying PCA, from the below graph we see that 13 components are explaining about 90% variance so we will use 13 components instead of given variables

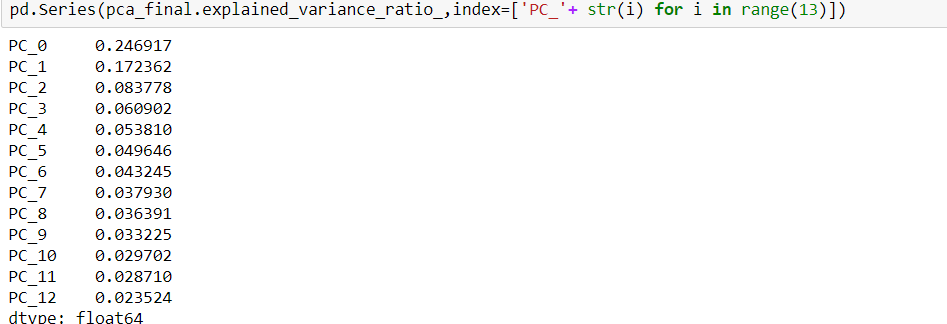
  
Here is how the sample of principle components looks like:



**Factor Analysis:**

Factor analysis is a statistical method used to describe variability among observed, correlated variables in terms of a potentially lower number of unobserved variables called factors or principle components with PCA.

Variance explained by each principle component is:



**2.3 Modeling:**

**K-Means clustering:**

Kmeans is a popular unsupervised machine learning algorithms. It’s 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 (far) as possible.

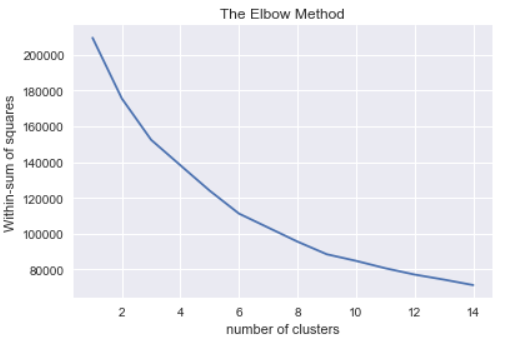
**The Elbow Method:**

This is probably the most well-known method for determining the optimal number of clusters.

Calculate the **Within-Cluster-Sum of Squared** Errors (WSS) for **different values of k**, and choose the k for which WSS becomes first starts to diminish. In the plot of WSS-versus-k, this is visible as an **elbow.**

We have used elbow method to find the optimum number of clusters to partition the whole data using K-means algorithm.

The below graph gives an illustration of elbow graph:



From the elbow graph we can see that K = 6 should be optimum number of clusters.

On performing K-means clustering with K as 6 we get the following value counts for each cluster

K-means clustering code in Python:

km=KMeans(n\_clusters=6,random\_state=123)

km.fit(Credit\_card6)

km.labels\_

**Values counts:**

**5 2410**

**3 2152**

**1 1840**

**2 1715**

**4 650**

**0 183**

**Total 8950**

**3. Conclusion:**

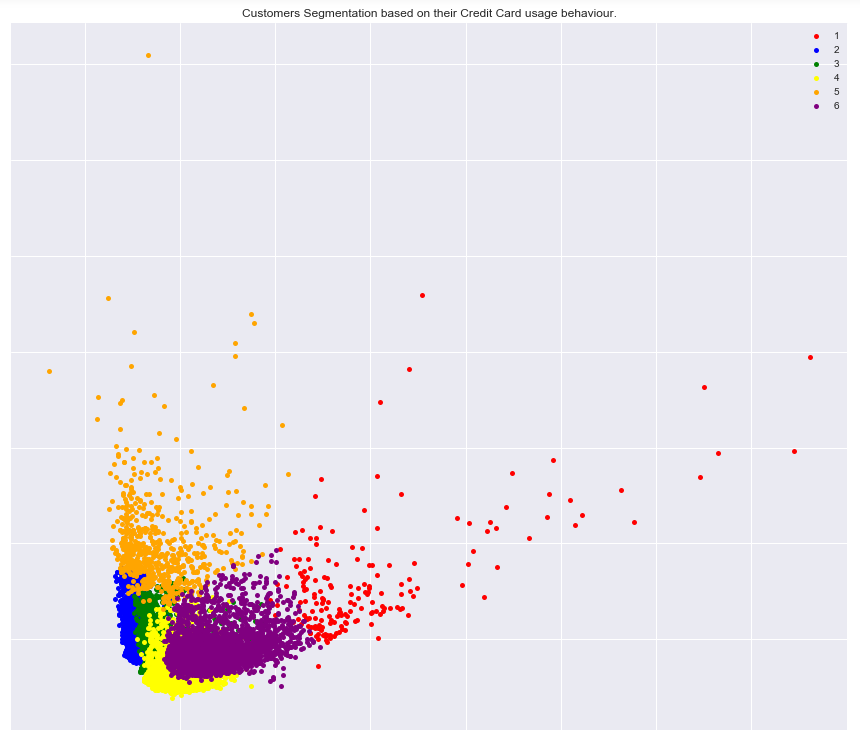
**3.1 Model Evaluation:**

The only gold standard of evaluating clusters is by visualizing the clusters and then a human expert studying it based on experience and domain knowledge. Otherwise theoretically there are few methods that can evaluate the quality of clustering, one of such method is silhouette\_score.

**3.1.1 Visualizing Clusters:**

Customers Segmentation based on their Credit Card usage behavior.

Using the below plot we can visualize the clusters:

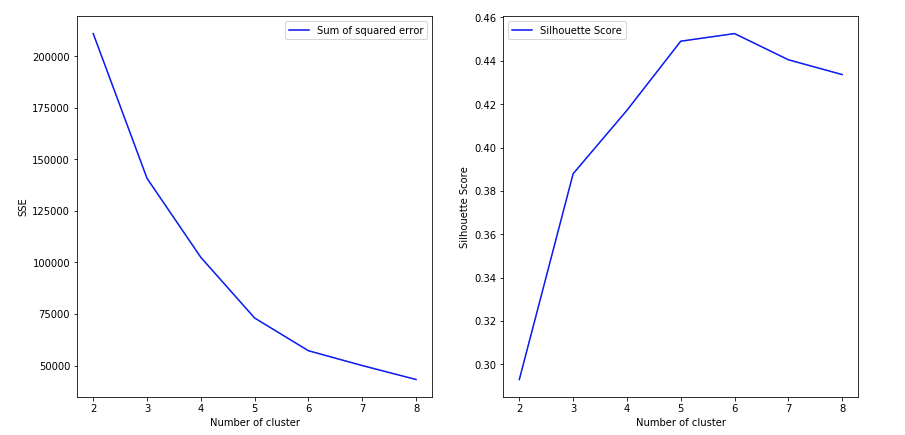


**3.1.2 Silhouette\_score Metric:**

The silhouette value measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation).

The range of the Silhouette value is between +1 and -1. A **high value is desirable** and indicates that the point is placed in the correct cluster. If many points have a negative Silhouette value, it may indicate that we have created too many or too few clusters.

I mentioned before that a high Silhouette Score is desirable. The Silhouette Score reaches its **global maximum at the optimal k**. This should ideally appear as a peak in the Silhouette Value-versus-k plot like in the below Silhouette score graph.

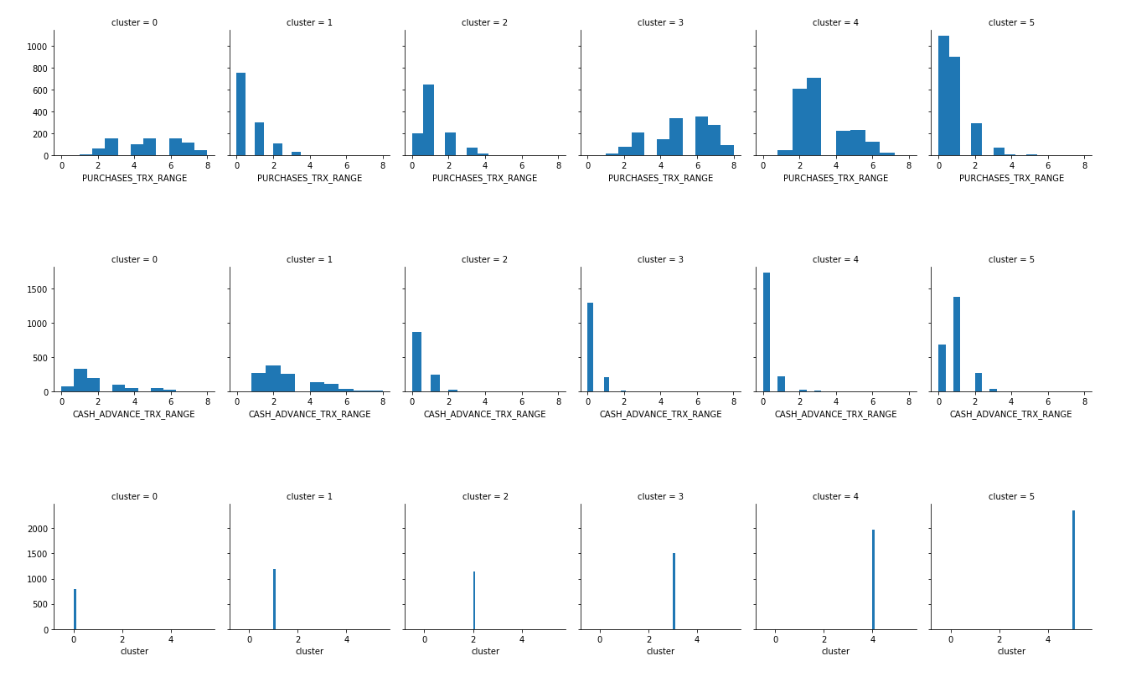
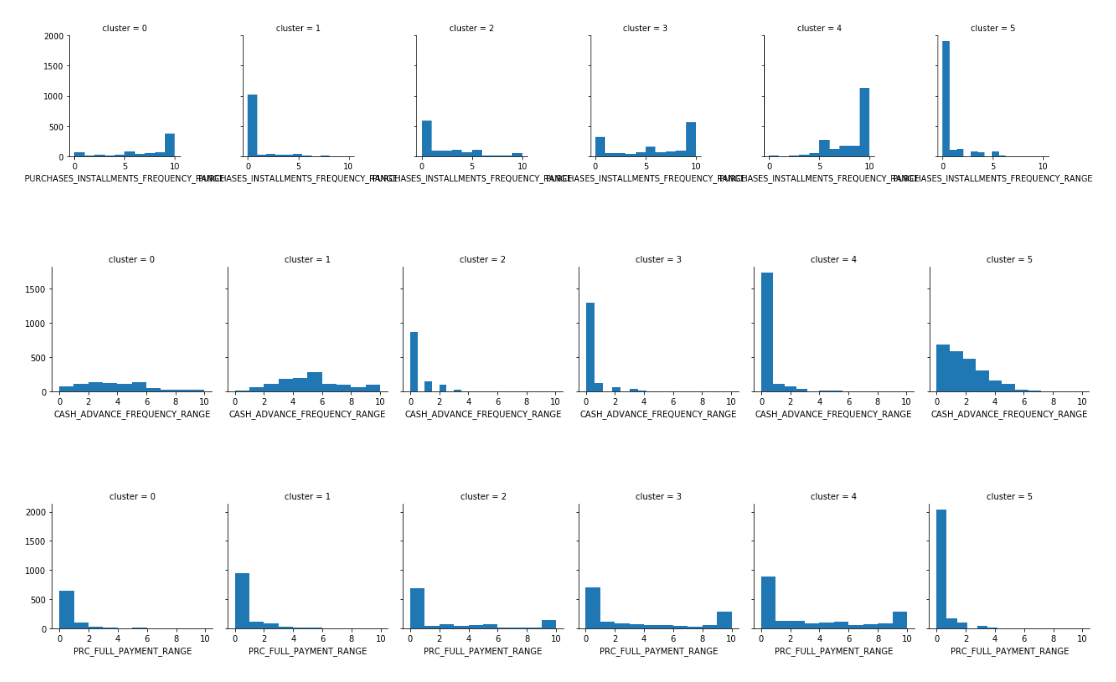
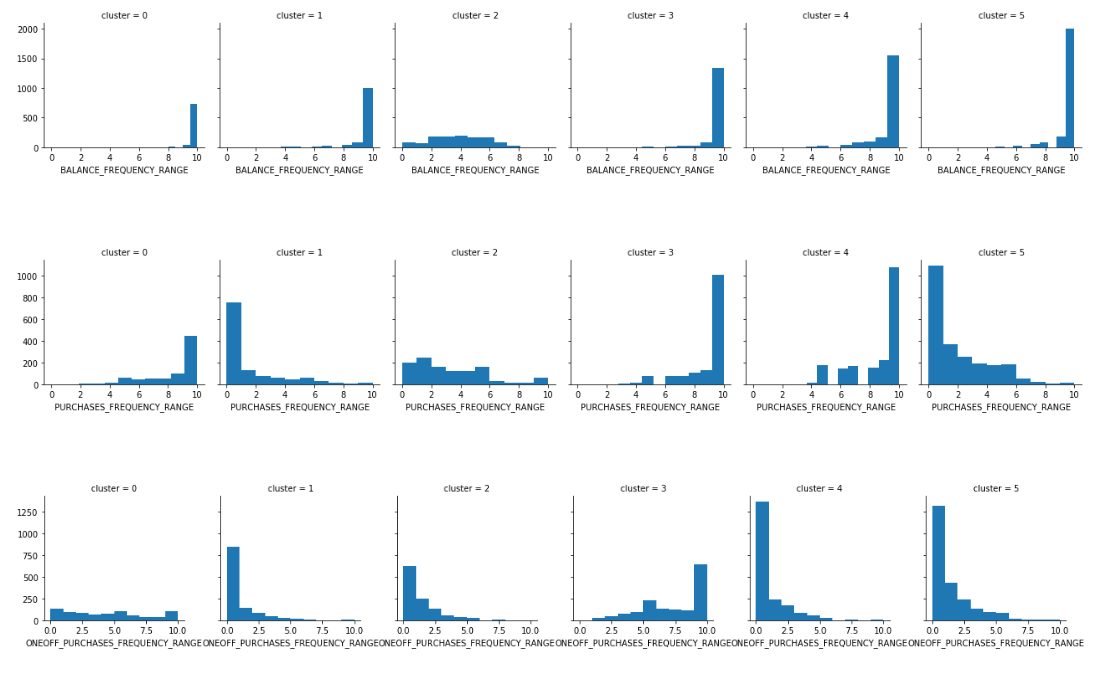
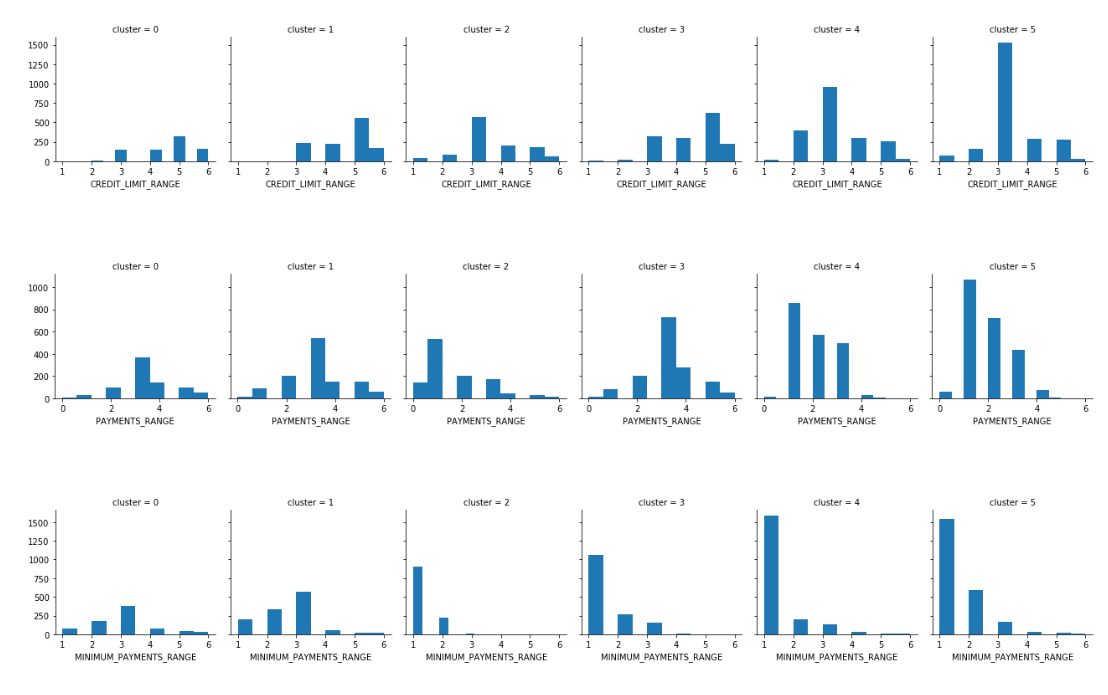
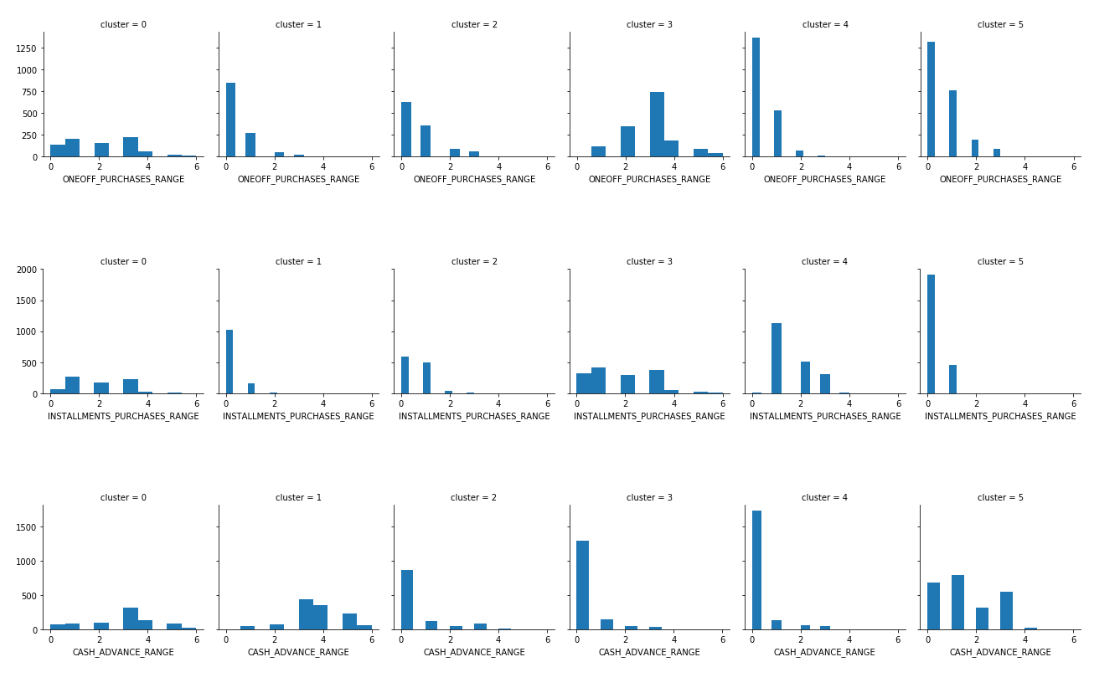
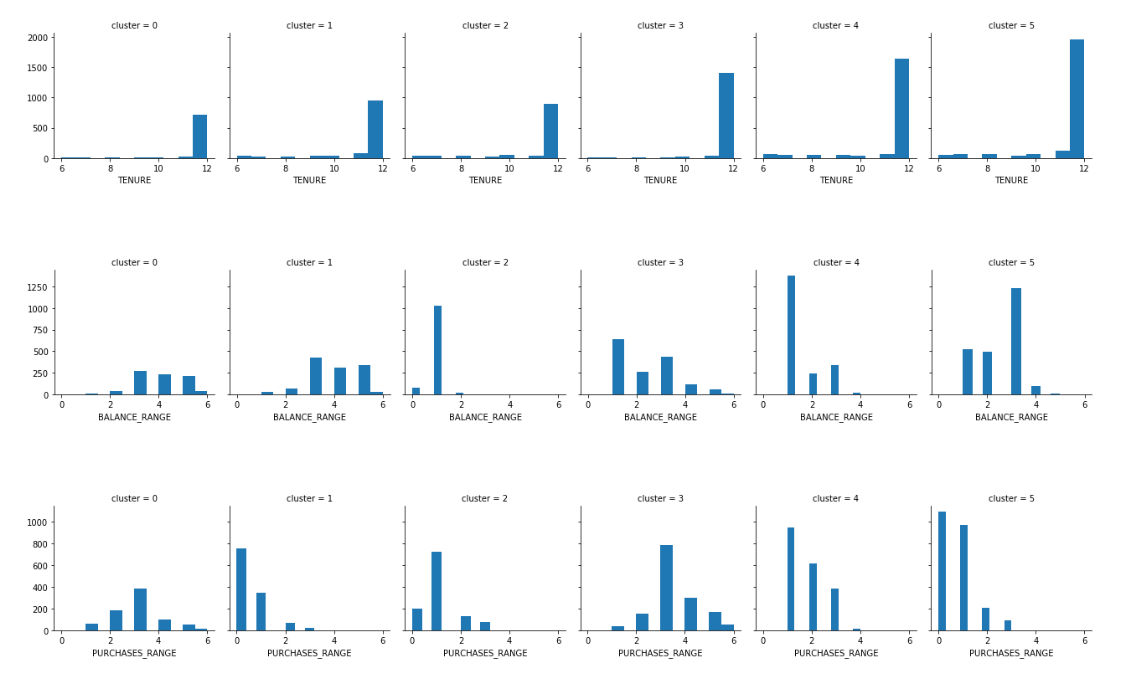


We can see that the silhouette score is maximum at k value 6.

**3.2 Cluster Interpretation:**

Following are the interpretations made by using derived key performance indicators (KPI) as mentioned above:

* People with less credit limit are using the amount to fullest and vice versa.
* People with low credit limit tend to take cash advance more than going for purchases.
* People with higher tenure are like to purchase more.
* Customers who don’t go for either one-off purchases nor installment purchases are likely to take cash advances.
* People with high credit limit also have the higher purchase ratio
* Majority of the one-off purchases are below 10000 amount indicating that people are going for installments for higher amount purchases.
* People are more inclined towards installments for higher amount purchases

Following are the interpretations derived from clusters using k-means clustering algorithm:****

* Cluster0 People with average to high credit limit who make all type of purchases
* Cluster1 This group has more people with due payments who take advance cash more often
* Cluster2 Less money spenders with average to high credit limits who purchases mostly in installments
* Cluster3 People with high credit limit who take more cash in advance
* Cluster4 High spenders with high credit limit who make expensive purchases
* Cluster5 People who don't spend much money and who have average to high credit limit

**4. Visualizations:**

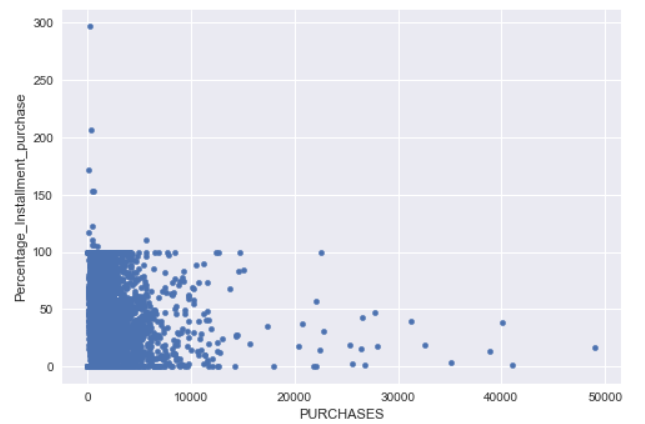
Following are some more visualizations from KPI:

**Percentage\_oneoff\_purchases:**



We can see from one off percentage purchase graph that majority of the one-off purchases are below 10000 amount indicating that people are going for installments for higher amount purchases.

**Percentage\_Installment\_purchase:**



We can see that people are more inclined towards installments for higher amount purchases

**5. Code:**

**R-Code:**

#This case requires trainees to develop a customer segmentation to define marketing strategy.

#clearing all the existing objects

rm(list=ls())

#set working directory

setwd("C:/Users/Sunny/Desktop/edwisor/Credit Card")

getwd()

#reading the data file

Credit\_Card = read.csv("credit-card-data.csv", header = TRUE, sep = ",")

#sample data

head(Credit\_Card)

#dimensions of data

dim(Credit\_Card)

#Summary of the data

summary(Credit\_Card)

#structure of the data

str(Credit\_Card)

#We can see that the given data only consists of numeric variables and only categorical

#vaiable present is CUST\_ID which we will anyways omit as its of no use for analysis and grouping.

#removing CUST\_ID from the data

Credit\_Card = Credit\_Card[-1]

names(Credit\_Card)

#finding the missing values

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

missing\_val

#We can see that the variables CREDIT\_LIMIT and MINIMUM\_PAYMENTS has missing values

#lets find percentage of missing values

install.packages("tidyr")

install.packages("DataExplorer")

library("tidyr")

library("DataExplorer")

plot\_missing(Credit\_Card)

#from the plot we can see that only one value missing from CREDIT\_LIMIT and its 0.01% and then

#MINIMUM\_PAYMENTS has 3.5% of missing values. Which means its better to

#impute the values using mean value of the respective variable in this case

#Treating missing values:

Credit\_Card$CREDIT\_LIMIT[which(is.na(Credit\_Card$CREDIT\_LIMIT))] <- mean(Credit\_Card$CREDIT\_LIMIT, na.rm=TRUE)

Credit\_Card$MINIMUM\_PAYMENTS[which(is.na(Credit\_Card$MINIMUM\_PAYMENTS))] <- mean(Credit\_Card$MINIMUM\_PAYMENTS, na.rm=TRUE)

plot\_missing(Credit\_Card)

#Now there are no missing values

#some variables are of range 0 to 1 and few are of the range 0 to 30000,

#hence data scaling is important for grouping the given data

#Data Scaling:(standardization)

Credit\_Card1 = scale(Credit\_Card)

summary(Credit\_Card1)

#K-means clustering:

#finding no of clusters to build using elbow graph

set.seed(123)

# Compute and plot within sum of squares(wss) for k = 2 to k = 15.

k.max <- 15

data <- Credit\_Card1

wss <- sapply(1:k.max,

function(k){kmeans(data, k, nstart=50,iter.max = 15 )$tot.withinss})

wss

plot(1:k.max, wss,

type="b", pch = 19, frame = FALSE,

xlab="Number of clusters K",

ylab="Total within-clusters sum of squares")

#using the elbow graph we can see that no of clusters to be user are 8

#K-means clustering

kmeans\_model = kmeans(data, 8, nstart = 50, iter.max = 15)

#we keep number of iter.max=15 to ensure the algorithm converges and nstart=50 to

#ensure that atleat 50 random sets are choosen

#As the final result of k-means clustering result is sensitive to the

#random starting assignments, we specify nstart = 25.

#This means that R will try 25 different random starting assignments

#and then select the best results corresponding to the one with the

#lowest within cluster variation.

#Summarize cluster output

kmeans\_model

#CLuster analysis:

Cluster\_data = cbind(Credit\_Card, clusterID = kmeans\_model$cluster)

Cluster\_data = data.frame(Cluster\_data)

head(Cluster\_data)

install.packages("cluster")

library(cluster)

clusplot(Credit\_Card, kmeans\_model$cluster, color=TRUE, shade=TRUE,labels=2, lines=0)

#plotting obtained clusters on existing data:

install.packages("ggplot2")

library("ggplot2")

p1<-ggplot(Cluster\_data, aes(x = clusterID, y = PURCHASES)) +

geom\_bar(fill = "#0073C2FF", stat = "identity") +

geom\_text(aes(label = PURCHASES), vjust = -0.3)

p2<-ggplot(Cluster\_data, aes(x = clusterID, y = ONEOFF\_PURCHASES)) +

geom\_bar(fill = "#0073C2FF", stat = "identity") +

geom\_text(aes(label = ONEOFF\_PURCHASES), vjust = -0.3)

p3<-ggplot(Cluster\_data, aes(x = clusterID, y = INSTALLMENTS\_PURCHASES)) +

geom\_bar(fill = "#0073C2FF", stat = "identity") +

geom\_text(aes(label = INSTALLMENTS\_PURCHASES), vjust = -0.3)

p4<-ggplot(Cluster\_data, aes(x = clusterID, y = CREDIT\_LIMIT)) +

geom\_bar(fill = "#0073C2FF", stat = "identity") +

geom\_text(aes(label = CREDIT\_LIMIT), vjust = -0.3)

p5<-ggplot(Cluster\_data, aes(x = clusterID, y = PAYMENTS)) +

geom\_bar(fill = "#0073C2FF", stat = "identity") +

geom\_text(aes(label = PAYMENTS), vjust = -0.3)

p6<-ggplot(Cluster\_data, aes(x = clusterID, y = PRC\_FULL\_PAYMENT)) +

geom\_bar(fill = "#0073C2FF", stat = "identity") +

geom\_text(aes(label = PRC\_FULL\_PAYMENT), vjust = -0.3)

gridExtra::grid.arrange(p1,p2,p3,p4,p5,p6, ncol=3)

#from the barplots we can conclude the following:

# out of 8 clusters, clusters like 1,3,8 are of high purchase and they tend to buy

# both one-off purchase and installment purchases high but they are good at

# installment payments but not one off payments. SO its better offer them good plans

# for installment payments

# clusters like 1,3,6,7 are having high credit limit and we can see that people who have

# high purchases also has high payments and full payments indirectly showing that high

# Credit limit comes with good payment history

# on the other hand from clusters like 2,4,5,8 we can see that people with low credit limit

# and likely to go for one-off purchase over installments and its commonsense to choose

# one off purchases when you have low credit limit as installments need mostly need high

# credit limit for bigger purchases

**Python Code:**

#lets import required packages:

import os

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

get\_ipython().magic('matplotlib inline')

import seaborn as sns

#lets Set working directory

os.chdir("C:\\Users\\Sunny\\Desktop\\edwisor\\Credit Card")

os.getcwd()

# reading data into dataframe

Credit\_card= pd.read\_csv("credit-card-data.csv")

#sample data

Credit\_card.head(5)

#data types of variables

Credit\_card.dtypes

#no.of rows and columns

Credit\_card.shape

#Summary of the data

Credit\_card.describe()

#lets find the missing values in the data

print(Credit\_card.isnull().sum())

# we can see that the variables "CREDIT\_LIMIT" and "MINIMUM\_PAYMENTS" has missing values

#treating missing values

#impute the missing values using the respectable variable mean

Credit\_card['CREDIT\_LIMIT'].fillna(Credit\_card['CREDIT\_LIMIT'].mean(),inplace=True)

Credit\_card['MINIMUM\_PAYMENTS'].fillna(Credit\_card['MINIMUM\_PAYMENTS'].mean(),inplace=True)

print (Credit\_card.isnull().sum())

#after treating missing values we can see that there are no mising values in the data

#lets drop the forst column "CUST\_ID" as it will not help in analysis

Credit\_card1 = Credit\_card.drop(['CUST\_ID'], axis = 1)

Credit\_card1.head(3)

#lets plot some variables and try getting some useful insights

#Percentage oneoff purchases:

Credit\_card1['Percentage\_oneoff\_purchases']=(Credit\_card1['ONEOFF\_PURCHASES']/Credit\_card1['PURCHASES'])\*100

Credit\_card1['Percentage\_oneoff\_purchases'].head(5)

Credit\_card1.plot.scatter(x='PURCHASES', y='Percentage\_oneoff\_purchases')

#Percentage\_Installment\_purchase:

Credit\_card1['Percentage\_Installment\_purchase']=(Credit\_card1['INSTALLMENTS\_PURCHASES']/Credit\_card1['PURCHASES'])\*100

Credit\_card1['Percentage\_Installment\_purchase'].head(5)

Credit\_card1.plot.scatter(x='PURCHASES', y='Percentage\_Installment\_purchase')

#Percentage\_Usage

Credit\_card1['Percentage\_Usage']=(Credit\_card1['PURCHASES']/Credit\_card1['CREDIT\_LIMIT'])\*100

Credit\_card1['Percentage\_Usage'].head(5)

Credit\_card1.plot.scatter(x='Percentage\_Usage', y='CREDIT\_LIMIT')

#Cash\_Advance\_percentage:

Credit\_card1['Cash\_Adv\_prcentage']=(Credit\_card1['CASH\_ADVANCE']/Credit\_card1['CREDIT\_LIMIT'])\*100

Credit\_card1['Cash\_Adv\_prcentage'].head(5)

Credit\_card1.plot.scatter(x='CREDIT\_LIMIT', y='Cash\_Adv\_prcentage')

#Monthly\_average\_purchase:

Credit\_card1['Monthly\_avg\_purchase']=Credit\_card1['PURCHASES']/Credit\_card1['TENURE']

Credit\_card1['Monthly\_avg\_purchase'].head(5)

Credit\_card1.plot.scatter(x='TENURE', y='Monthly\_avg\_purchase')

#Monthly\_cash\_advance:

Credit\_card1['Monthly\_cash\_advance']=Credit\_card1['CASH\_ADVANCE']/Credit\_card1['TENURE']

Credit\_card1['Monthly\_cash\_advance'].head(5)

Credit\_card1.plot.scatter(x='TENURE', y='Monthly\_cash\_advance')

#payments to minpayments:

Credit\_card1['payment\_minimumpayments']=Credit\_card1.apply(lambda x:x['PAYMENTS']/x['MINIMUM\_PAYMENTS'],axis=1)

Credit\_card1['payment\_minimumpayments'].head(5)

#We can see that there are 4 types of purchase behaviors in the customers from the given data set.

# 1.People who only do One-Off Purchases.

# 2.People who only do Installments Purchases.

# 3.People who do both.

# 4.People who do none.

#So deriving a categorical variable based on the customer behavior.

Credit\_card1[(Credit\_card1['ONEOFF\_PURCHASES']==0) & (Credit\_card1['INSTALLMENTS\_PURCHASES']==0)].shape

Credit\_card1[(Credit\_card1['ONEOFF\_PURCHASES']>0) & (Credit\_card1['INSTALLMENTS\_PURCHASES']>0)].shape

Credit\_card1[(Credit\_card1['ONEOFF\_PURCHASES']>0) & (Credit\_card1['INSTALLMENTS\_PURCHASES']==0)].shape

Credit\_card1[(Credit\_card1['ONEOFF\_PURCHASES']==0) & (Credit\_card1['INSTALLMENTS\_PURCHASES']>0)].shape

def purchase(Credit\_card1):

if (Credit\_card1['ONEOFF\_PURCHASES']==0) & (Credit\_card1['INSTALLMENTS\_PURCHASES']==0):

return 'none'

if (Credit\_card1['ONEOFF\_PURCHASES']>0) & (Credit\_card1['INSTALLMENTS\_PURCHASES']>0):

return 'both\_oneoff\_installment'

if (Credit\_card1['ONEOFF\_PURCHASES']>0) & (Credit\_card1['INSTALLMENTS\_PURCHASES']==0):

return 'one\_off'

if (Credit\_card1['ONEOFF\_PURCHASES']==0) & (Credit\_card1['INSTALLMENTS\_PURCHASES']>0):

return 'istallment'

Credit\_card1['purchase\_type']=Credit\_card1.apply(purchase,axis=1)

Credit\_card1.shape

Credit\_card1.head(5)

Credit\_card1['purchase\_type'].value\_counts()

Credit\_card1.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')

Credit\_card1.groupby('purchase\_type').apply(lambda x: np.mean(x['CREDIT\_LIMIT'])).plot.barh()

plt.title('Credit limit of different Purchase types : Both, None,Installment,One\_Off')

Credit\_card1.shape

Credit\_card1.dtypes

Credit\_card1.describe()

#creating Dummies for categorical variable

Credit\_card2=pd.concat([Credit\_card1,pd.get\_dummies(Credit\_card1['purchase\_type'])],axis=1)

Credit\_card2.head(3)

print(Credit\_card2.isnull().sum())

#lets remove the redundent variable "purchase\_type" as it is already conveyed using the dummy variables

x=['purchase\_type','Percentage\_oneoff\_purchases','Percentage\_Installment\_purchase']

Credit\_card3 = Credit\_card2.drop(x, axis = 1)

print(Credit\_card3.isnull().sum())

Credit\_card3.head(3)

#Since there are variables having extreme values these can become potential outliers, hence lets standardize the data

#which will also bring all the variables in to one standard range

from sklearn import preprocessing

names = Credit\_card3.columns

scaler = preprocessing.StandardScaler()

Credit\_card4 = scaler.fit\_transform(Credit\_card3)

Credit\_card4 = pd.DataFrame(Credit\_card4, columns=names)

Credit\_card4.describe()

Credit\_card4.shape

#lets apply PCA to reduse dimentionality before going for clustering the data, this also takes care of

#any correlation existing between the variables

from sklearn.decomposition import PCA

var\_ratio={}

for n in range(2,25):

pca=PCA(n\_components=n)

Credit\_card5=pca.fit(Credit\_card4)

var\_ratio[n]=sum(Credit\_card5.explained\_variance\_ratio\_)

var\_ratio

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

#we see that 13 components are explaining about 90% variance so we select 13 components

pca\_final=PCA(n\_components=13)

Credit\_card5=pca\_final.fit\_transform(Credit\_card4)

Credit\_card6=pd.DataFrame(Credit\_card5)

Credit\_card6.shape

Credit\_card6.head(5)

#lets name the principle components:

Principle\_components = pd.DataFrame(pca\_final.components\_.T, columns=['PC\_' +str(i) for i in range(13)],index=names)

Principle\_components.head(3)

Principle\_components.shape

# Factor Analysis : variance explained by each principle component-

pd.Series(pca\_final.explained\_variance\_ratio\_,index=['PC\_'+ str(i) for i in range(13)])

#lets apply clustering algorithm to find the clusters those can explain customer behaviour and profile

#K-means clustering

#lets fine the optimumm no of clusters to be formed using silhouette\_score method

from sklearn.cluster import KMeans

WSS = []

for i in range(1, 15):

kmeans = KMeans(i)

kmeans.fit(Credit\_card6)

WSS\_iter = kmeans.inertia\_

WSS.append(WSS\_iter)

WSS

number\_clusters = range(1,15)

plt.plot(number\_clusters,WSS)

plt.title('The Elbow Method')

plt.xlabel('number of clusters')

plt.ylabel('Within-sum of squares')

#from the elbow graph we can see that K is equal to 6 should be optimum number of clusters

#K-means clustering with K as 6:

km=KMeans(n\_clusters=6,random\_state=123)

km.fit(Credit\_card6)

km.labels\_

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

#lets add the obtained clusters to the dataset

clustersPCA=pd.concat([Credit\_card6, pd.DataFrame({'cluster':km.labels\_})], axis=1)

clustersPCA.head(3)

clusters=pd.concat([Credit\_card1, pd.DataFrame({'cluster':km.labels\_})], axis=1)

clusters.head(3)

#lets plot the clusters

x, y = Credit\_card5[:, 0], Credit\_card5[:, 1]

colors = {0:'red',1:'blue',2:'green',3:'yellow',4:'orange', 5:'purple'}

names = {0:'1', 1:'2', 2:'3', 3:'4', 4:'5',5:'6'}

plot = pd.DataFrame({'x': x, 'y':y, 'label':km.labels\_})

groups = plot.groupby('label')

fig, ax = plt.subplots(figsize=(15, 13))

for name, group in groups:

ax.plot(group.x, group.y, marker='o', linestyle='', ms=5,

color=colors[name],label=names[name], mec='none')

ax.set\_aspect('auto')

ax.tick\_params(axis='x',which='both',bottom='off',top='off',labelbottom='off')

ax.tick\_params(axis= 'y',which='both',left='off',top='off',labelleft='off')

ax.legend()

ax.set\_title("Customers Segmentation based on their Credit Card usage behaviour.")

plt.show()

#Evaluating k-meaans performance

from sklearn.metrics import silhouette\_score

sil = []

kmax = 15

for k in range(2, kmax+1):

kmeans1 = KMeans(n\_clusters = k).fit(Credit\_card6)

labels = kmeans.labels\_

sil.append(silhouette\_score(Credit\_card6, labels, metric = 'euclidean'))

number\_clusters = range(1,15)

plt.plot(number\_clusters,sil)

plt.title('The silhouette\_score Method')

plt.xlabel('number of clusters - K')

plt.ylabel('silhouette\_score')

#Performance metrics also suggest that K-means with 6 cluster is able to show distinguished characteristics of each cluster.

#cluster interpretation

for c in clusters:

grid= sns.FacetGrid(clusters, col='cluster')

grid.map(plt.hist, c)

#Cluster0 People with average to high credit limit who make all type of purchases

#Cluster1 This group has more people with due payments who take advance cash more often

#Cluster2 Less money spenders with average to high credit limits who purchases mostly in installments

#Cluster3 People with high credit limit who take more cash in advance

#Cluster4 High spenders with high credit limit who make expensive purchases

#Cluster5 People who don't spend much money and who have average to high credit limit