```
#Remove all the objects from the R environment
rm(list = ls())

#Set working directory
setwd("G:/Edwiser material/Project/buffer project")

# check the set directory
getwd()

#Load the required Libraries
x = c('ggplot2', 'corrgram', 'DMwR', 'caret', 'unbalanced', 'dummies', 'e1071', 'Information',
    "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'NbClust','fastDummies')

#Install the loaded packages(x)
lapply(x, require, character.only = TRUE)
rm(x)

#load the data
df=read.csv("credit-card-data.csv")

##############Data pre-processing########################
#Remove the cust_ID column as it is nothing but ID
df$CUST_ID = NULL

#Check for the missing values and imputing with knn imputation
#missing_val = data.frame(apply(df,2,function(x){sum(is.na(x))}))
#missing_val$Columns = row.names(missing_val)
#names(missing_val)[1] =  "Missing_percentage"
#missing_val$Missing_percentage = (missing_val$Missing_percentage/nrow(df)) * 100
#missing_val = missing_val[order(-missing_val$Missing_percentage),]
#row.names(missing_val) = NULL
```

```
#missing_val = missing_val[,c(2,1)]

#write.csv(missing_val, "Missing_perc.csv", row.names = F)


#class(df$CREDIT_LIMIT)

#class(df$MINIMUM_PAYMENTS)


#unique(df$CREDIT_LIMIT)

#unique(df$MINIMUM_PAYMENTS)


#There are missing values present in CREDIT_LIMIT & MINIMUM_PAYMENTS columns

#Check for the right method of missing value imputation

#Real value =6250

#mean method imputation = 4494.253

#median method imputation = 3000

#knn imputation = 5362.473


#df$CREDIT_LIMIT[71]

#df$CREDIT_LIMIT[71] = NA


# Mean Method

#mean(df$CREDIT_LIMIT, na.rm = T)


#Median Method

#median(df$CREDIT_LIMIT, na.rm = T)


# kNN Imputation

#df=knnImputation(df, k = 3)


#Real value = 2180.882

#mean value = 864.0541

#median value = 312.2556
```

```
#knn imputation = 2157.051


#df$MINIMUM_PAYMENTS[71]

#df$MINIMUM_PAYMENTS[71] = NA


# Mean Method

#mean(df$MINIMUM_PAYMENTS, na.rm = T)


#Median Method

#median(df$MINIMUM_PAYMENTS, na.rm = T)


# kNN Imputation

df=knnImputation(df, k = 3)

dim(df)

str(df)

unique(df$TENURE)

unique(df$PURCHASES_TRX)


#########Exploratory data Analysis#####################

#####Advanced data preparation

##Derive new KPI of Monthly average Purchases

#MONTHLY avg purchases Derivation

df$MONTH_AVG_PURCHASES = df$PURCHASES/df$TENURE


#Monthly cash advance derivation

df$MONTHLY_CASH_ADVANCE = df$CASH_ADVANCE/df$TENURE


#Check for the Purchases by type (one-off, instalments) number of both

#one-off purchases and instalment purchases which are greater than zero and are equal to zero

nrow(df[which(df$ONEOFF_PURCHASES > 0 & df$INSTALLMENTS_PURCHASES > 0),])
```

```
#no of both one-off purchases and instalments purchases which are qual to zero
nrow(df[which(df$ONEOFF_PURCHASES==0 & df$INSTALLMENTS_PURCHASES==0),])

#no of one-off purchases are zero and instalment purchases are greater than zero
nrow(df[which(df$ONEOFF_PURCHASES == 0 & df$INSTALLMENTS_PURCHASES > 0),])

#no of one-off purchases are higher than zero and instalments are zero
nrow(df[which(df$ONEOFF_PURCHASES >0 & df$INSTALLMENTS_PURCHASES== 0),])

#With the above details it will be clear that there will be four different types of transactions which are used for
#derive a new feature
#create a definition for new features
df$PURCHASE_TYPE[df$ONEOFF_PURCHASES == 0 & df$INSTALLMENTS_PURCHASES ==0]= "NONE"
df$PURCHASE_TYPE[df$ONEOFF_PURCHASES >0 & df$INSTALLMENTS_PURCHASES > 0]= "ONEOFF_INSTALLMENT"
df$PURCHASE_TYPE[df$ONEOFF_PURCHASES >0 & df$INSTALLMENTS_PURCHASES == 0]= "ONEOFF"
df$PURCHASE_TYPE[df$ONEOFF_PURCHASES ==0 & df$INSTALLMENTS_PURCHASES >0]= "INSTALLMENT"

# Limit usage calculation from balance to credit ratio
df$limit_usage = df$BALANCE/df$CREDIT_LIMIT

###Payments to minimum payments ratio
#Payments to minimum payments ratio calculation
df$Payment_minpay_Ratio = df$PAYMENTS/df$MINIMUM_PAYMENTS

#Separate PURCHASE_TYPE variable from the data to convert the all the data into log
#transformation so that outliers can be removed
df2=data.frame(df$PURCHASE_TYPE)
df$PURCHASE_TYPE=NULL
df=log(df+1)
```

```
dim(df)
sum(is.na(df))

############Feature Selection#########################
#Extract numeric index of the data to check the Correlation
numeric_index = sapply(df,is.numeric) #selecting only numeric
numeric_data = df[,numeric_index]
cnames = colnames(numeric_data)

##Correlation Plot
corrgram(df[,numeric_index], order = F,
        upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

####################### Dimensionality Reduction############################
#Remove the highly positive and negatively correlated variables
df = subset(df, select = -c(BALANCE,CASH_ADVANCE,PURCHASES_FREQUENCY,
                            CASH_ADVANCE_FREQUENCY,TENURE))
##############################Feature Scaling#########################
#Normality check
hist(df$BALANCE_FREQUENCY)

#Extract the column names
cnames=colnames(df)

#Normalise
for(i in cnames){
#  print(i)
  df[,i] = (df[,i] - min(df[,i]))/
    (max(df[,i] - min(df[,i])))
}
```

```
#Rename the column name
names(df2)[1]="PURCHASE_TYPE"

#Get dummies for each category of the column
df2=dummy_columns(df2$df.PURCHASE_TYPE)

#Remove the variable used for extraction of dummy variables
df2$.data=NULL

#Join the dummy variables extracted data to the main data
d= cbind(df, df2)
##############Implement machine learning model#########
#Extract the number of clusters to be build
NBclust_res = NbClust(d, min.nc=2, max.nc=15, method = "kmeans")

#Plot a Barplot to analyse the optimum clusters
barplot(table(NBclust_res$Best.n[1,]),
      xlab="Number of Clusters", ylab="Number of Criteria",
      main="Number of Clusters Chosen by 26 Criteria")
#Apply the K-mean clustering with Four clusters
kmeans_model = kmeans(d, 4, nstart=25)

#Summarize the clustering output
kmeans_model
```