Credit Card Segmentation

Bhupendra Kumar

April 2020

Contents

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.

1.1 Data:

We are given 18 features and 8950 points(Rows) in the data.

Given below is a sample of the data set, we are using to cluster the behaviour of credit card holders.

Table 1.1: Credit Card Sample Data (Columns: 1-8)

A	B	C	D	E	F	G	H
CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY
C10001	40.900749	0.818182	95.4	0	95.4	0	0.166667
C10002	3202.467416	0.909091	0	0	0	6442.945483	0
C10003	2495.148862	1	773.17	773.17	0	0	1
C10004	1666.670542	0.636364	1499	1499	0	205.788017	0.083333

Table 1.1: Credit Card Sample Data (Columns: 9-14)

H	I	J	K	L	M
PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURCHASES_INSTALLMENTS_FREQUENCY	CASH_ADVANCE_FREQUENCY	CASH_ADVANCE_TRX	PURCHASES_TRX
0.166667	0	0.083333	0	0	2
0	0	0	0.25	4	0
1	1	0	0	0	12
0.083333	0.083333	0	0.083333	1	1

Table 1.1: Credit Card Sample Data (Columns: 15-19)

N	O	P	Q	R
CREDIT_LIMIT	PAYMENTS	MINIMUM_PAYMENTS	PRC_FULL_PAYMENT	TENURE
1000	201.802084	139.509787	0	12
7000	4103.032597	1072.340217	0.222222	12
7500	622.066742	627.284787	0	12
7500	0		0	12

Below are the given variables, will help us to cluster the different behaviours of credit card holders.

Table 1.3: Predictor variables

Sr.no.	**Variables**
1	CUST_ID
2	BALANCE
3	BALANCE_FREQUENCY
4	PURCHASES
5	ONEOFF_PURCHASES
6	INSTALLMENTS_PURCHASES
7	CASH_ADVANCE
8	PURCHASES_ FREQUENCY
9	ONEOFF_PURCHASES_FREQUENCY
10	PURCHASES_INSTALLMENTS_FREQUENCY
11	CASH_ADVANCE_ FREQUENCY
12	AVERAGE_PURCHASE_TRX
13	CASH_ADVANCE_TRX
14	PURCHASES_TRX
15	CREDIT_LIMIT
16	PAYMENTS
17	MINIMUM_PAYMENTS
18	PRC_FULL_PAYMENT
19	TENURE

2. Methodology:

2.1 Data Pre-Processing

2.1.1 Missing Value Analysis

Missing values in data is a common real world problem, we face while analysing and fitting the model. In this case, I have imputed missing values using the median of the active feature.

The table below shows us missing value counts for each feature in our data.

Variables	**Total #. Of missing data**
CUST_ID	0
BALANCE	0
BALANCE_FREQUENCY	0
PURCHASES	0
ONEOFF_PURCHASES	0
INSTALLMENTS_PURCHASES	0
CASH_ADVANCE	0
PURCHASES_ FREQUENCY	0
ONEOFF_PURCHASES_FREQUENCY	0
PURCHASES_INSTALLMENTS_FREQUENCY	0
CASH_ADVANCE_ FREQUENCY	0
AVERAGE_PURCHASE_TRX	0
CASH_ADVANCE_TRX	0
PURCHASES_TRX	0
CREDIT_LIMIT	1
PAYMENTS	0
MINIMUM_PAYMENTS	313

PRC_FULL_PAYMENT	0
TENURE	0

2.1.2 Outlier Analysis

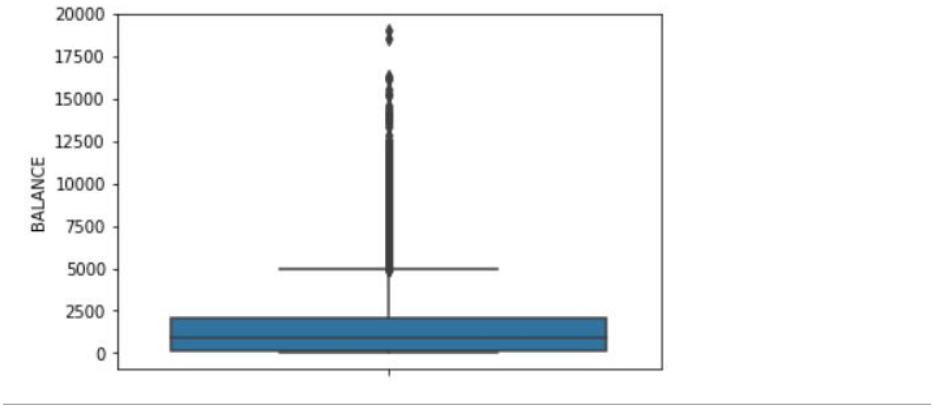
This is one of the most important data pre-processing parts which needs good understanding of the data and careful analysis. Outliers are the points which are far away from the other observations/or from their distribution.

We are going to observe outliers with the help of Box-Plot. While EDA we observed outliers in features “BALANCE” and “PURCHASES”, boxplot of ‘BALANCE” in **(Fig 2.4)**.

After removing the outliers from ‘BALANCE’ feature, total points left are 8255 and if we remove ‘PURCHASES’ feature total points(rows) comes down to 7590. Which is a huge data loss, since all features have outliers in it.

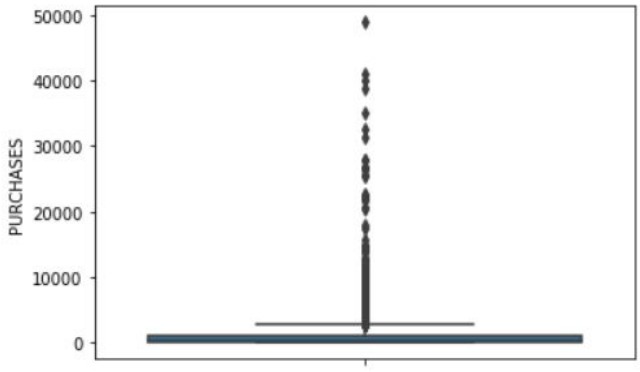
So, instead of removing outliers, we will apply log function on each variable. Which will help us in reducing the effect of outliers.

```
[0.0, 128.2819155, 873.385231, 2054.1400355]
points are outliers if below -2760.5052645 and above 4942.9272155
```



Fig

```
[0.0, 39.635, 361.28, 1110.13]
points are outliers if below -1566.1075000000003 and above 2715.8725000000004
```



Fig

Code Snippet:

```
In [12]: # Applying logarithmic function to reeduce the effect of outliers
         credit_data=credit_data.drop(["CUST_ID"],axis=1).applymap(lambda x: np.log(x+1))
```

2.1.3 Feature Selection

When the number of features are very large. We can't visualize or create correlation heat map to observe which features are important and which are not. In our case we have known that have only 18 features out of which 16 are continuous features(Relevant ones).

Observations:

i. It's clearly visible through the heat map that most of the features are highly correlated to each other..

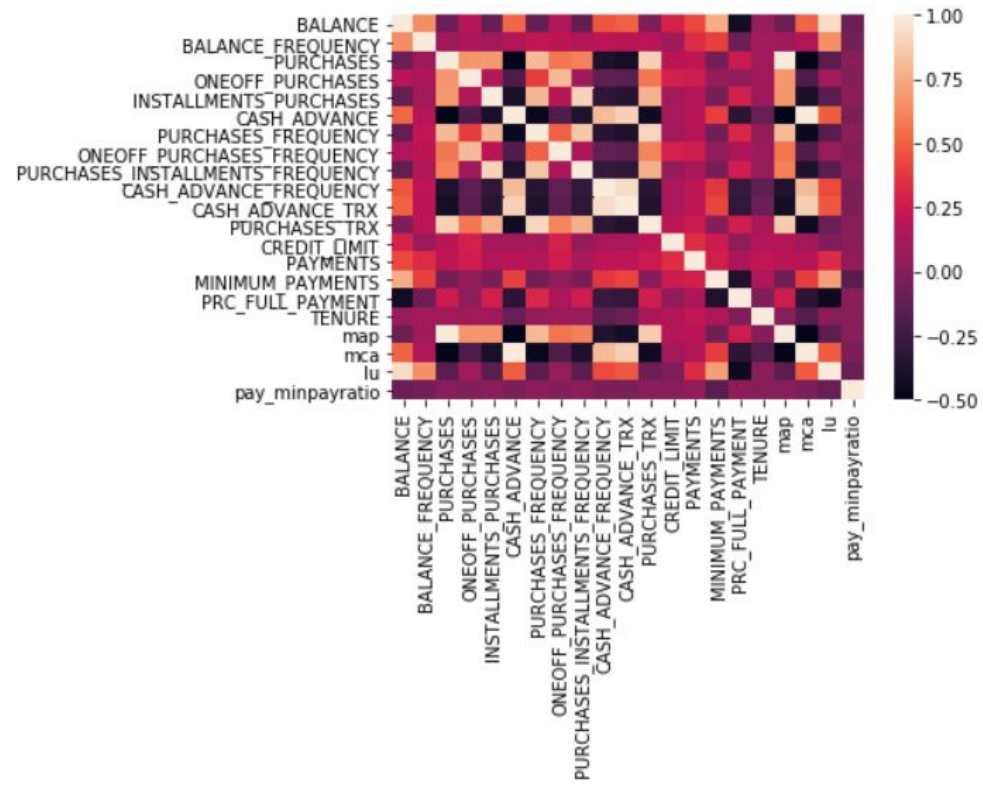
ii. Features mentioned below are highly correlated to each other and we will reduce the dimensions

a. lu - Balance, mca-Purchases, mca - Installments_Purchases, mca - Purchases_Frequency, map - Cash_Advance, mca - Purchases_Installments-Frequency, map - Cash_Advance_Frequency, map - Cash_Advance_TRX and map - Prc_Full_Payment

Below fig 2.6 illustrates that relationship between all numeric variables using Correlation heat map

Figure 2.6 correlation heat map of numeric variables (R code in Appendix B)

<matplotlib.axes._subplots.AxesSubplot at 0x223796ee7f0>



2.1.4 Deriving New KPI

Key Performance Indicator (KPI) is a type of performance measurement, used to evaluate the performance of an organization, projects or products in which it engages.

i. Monthly Average Purchases(map):

map = Purchases/Tenure

Name itself says everything gives us an average purchase in a month by the customer.

ii. Monthly Cash Advancement(mca):

mca = Cash Advance/Tenure

Name itself says everything gives us an average advance cash taken by customer monthly wise.

iii. Limit Usage(lu):

lu = Balance/Credit Limit

If lu is high => Balance is higher than credit limit else Balance has crossed Credit limit. In other words lower values means good credit score.

iv. Payments to minimum payment ratio(pay_minpayratio):

pay_minpayratio= Payments/Minimum Payments

If the ratio is high then the customer has been paying bills more than Minimum Payments on time.

2.1.5 Feature Scaling

Feature scaling is one of the most important pre-processing techniques. It majorly involves two techniques named as Normalization and standardization.

Link to Random forest classifier based feature importance is given below,
https://towardsdatascience.com/running-random-forests-inspect-the-feature-importances-with-this-code-2b00dd72b92e
It's important to rescale features else it may lead to wrong predictions, especially in the case of regression problems.

Rescaling data between 0 and 1 is known as feature scaling. In our case we will normalize all the features in giving data.

Code snippet:

```
# All of the columns need to be scaled
for ithCol in Final_CreditData.columns:
    Final_CreditData[ithCol]=(Final_CreditData[ithCol] - min(Final_CreditData[ithCol]))/(max(Final_CreditData[ithCol]) -
                             min(Final_CreditData[ithCol]))
Final_CreditData.head()
```

2.2 Clustering

Clustering means grouping of objects based on information found in the data, describing the objects or their relationship. The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data.

Organizing data into clusters shows internal structure of the data. Sometimes, partitioning is the goal.

2.2.1 KMean Clustering:

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. To achieve this objective, K-means looks for a fixed number (k) of clusters in a dataset.

Where **K is Hyperparameter.**

Code Snippet:

```
from sklearn.cluster import KMeans
from sklearn import metrics
```

```
# I am going to use elbow method to classify the clusters
Final_CreditData.shape
```

```
(8950, 10)
```

```
cluster_range = range( 2, 11 )
cluster_errors = []
#This list will contain silhouette scores
scores = []
for num_clusters in cluster_range:
    clusters = KMeans( n_clusters=num_clusters, random_state=123 )
    clusters.fit( Final_CreditData )
    cluster_errors.append( clusters.inertia_ )
    scores.append(metrics.silhouette_score(Final_CreditData, clusters.labels_))
```

2.2.2 DBSCAN Clustering:

The main concept of DBSCAN algorithm is to locate regions of high density that are separated from one another by regions of low density.

i. *Density at a point P: Number of points within a circle of Radius Eps (ϵ) from point P.*

ii. *Dense Region: For each point in the cluster, the circle with radius ϵ contains at least minimum number of points (MinPts).*

Code Snippet:

DBSCAN Algorithm

```
from sklearn.cluster import DBSCAN
```

```
model = DBSCAN(eps=0.4, min_samples=4).fit(df_X)
pred = model.fit_predict(Y)
df_Y['PREDICTED_CLUSTER'] = pred
train_summary = df_Y.groupby(by='PREDICTED_CLUSTER').mean()
train_summary
```

Conclusion

DBSCAN algo is giving quite accurate results and easy to classify them, below are clusters and a set of strategies can be applied in business to grow w.r.t the cluster properties.

a. Group -1

- They are potential target customers who are paying dues and doing purchases and maintaining comparatively good credit score(Limit Usage) comparatively it's low to other clusters. Monthly Average purchase is also not very high but average(Still Balance is low)
  - -- we can increase credit limit or can lower down interest rate
  - -- Can be given premium card /loyalty cards to increase transactions

b. Group 0

- Their credit score close to average (0.44), we need more information on this set to make further decisions. (Since their Credit limit and payments is above average.)

c. Group 1

- This group has a minimum paying ratio and uses cards for just one off transactions (may be for utility bills only). This group seems to be risky group.(And they have high credit limit, so definitely something i not going right for this group in recent times)

d. Group 2

- This group is performing best among all as customers are maintaining good credit scores and paying dues on time.
  - -- Increasing credit limit or giving premium cards offer will be justifiable for this group.

e. Group 3

- Their credit score close to average (0.44), we need more information on this set to make further decisions. (Since their Credit limit, payments are above average and Balance frequency is also quite high.)

Appendix A (Complete R File)

EDA (Exploratory Data Analysis)

```
##################################Exploratory Data Analysis#############################################

#Col names
colnames(credit_data)

#Structure of the data
str(credit_data)
head(credit_data)

#Summary of the input data
summary(credit_data)

#Hist plot of BALANCE feature
hist(credit_data$BALANCE)

#Hist plot of BALANCE_FREQUENCY feature
hist(credit_data$BALANCE_FREQUENCY)

#Hist plot of BALANCE_FREQUENCY feature
hist(credit_data$PURCHASES)

#Hist plot of BALANCE_FREQUENCY feature
hist(credit_data$ONEOFF_PURCHASES)

#Hist plot of BALANCE_FREQUENCY feature
hist(credit_data$INSTALLMENTS_PURCHASES)

#Hist plot of BALANCE_FREQUENCY feature
hist(credit_data$CASH_ADVANCE)

#Hist plot of BALANCE_FREQUENCY feature
hist(credit_data$PURCHASES_FREQUENCY)

#Hist plot of BALANCE_FREQUENCY feature
hist(credit_data$ONEOFF_PURCHASES_FREQUENCY)

#Hist plot of BALANCE_FREQUENCY feature
hist(credit_data$PURCHASES_INSTALLMENTS_FREQUENCY)

#Hist plot of BALANCE_FREQUENCY feature
hist(credit_data$CASH_ADVANCE_FREQUENCY)
```

Treating Outliers and Missing Value

```
27
28 library(imputeMissings)
29 credit_data<- impute(credit_data,method = "median/mode")
30 sum(is.na(credit_data$CREDIT_LIMIT))
31 sum(is.na(credit_data$MINIMUM_PAYMENTS))
32
33 library(ggplot2)
34 # Outlier Analysis
35 ggplot(data = credit_data, aes(x = "", y = MINIMUM_PAYMENTS)) +
36   geom_boxplot() # No outliers
37
38 ggplot(data = credit_data, aes(x = "", y = CREDIT_LIMIT)) +
39   geom_boxplot() # No outliers
40
41 #As we can see that most of the features has outliers, if we exclude outlier, loss of data is huge. Instead, let's normalize the data and apply log function.
42
```

KPI

```
################################################### KPI ##########################################################################
#New Variables creation#

credit_data$Monthly_Avg_PURCHASES <- credit_data$PURCHASES/credit_data$TENURE
credit_data$Monthly_CASH_ADVANCE <- credit_data$CASH_ADVANCE/credit_data$TENURE
credit_data$LIMIT_USAGE <- credit_data$BALANCE/credit_data$CREDIT_LIMIT
credit_data$MIN_PAYMENTS_RATIO <- credit_data$PAYMENTS/credit_data$MINIMUM_PAYMENTS
```

Feature Scaling and Feature Selection

```
#####################################Feature Scaling and Feature Selection#########################################################
Scaled_data = scale(credit_data)

View(Scaled_data)

Scaled_data<- impute(Scaled_data,method = "median/mode")

data_New<-subset(Scaled_data,select = c('BALANCE_FREQUENCY', 'ONEOFF_PURCHASES', 'ONEOFF_PURCHASES_FREQUENCY',
                                        'CREDIT_LIMIT', 'PAYMENTS', 'MINIMUM_PAYMENTS', 'Monthly_Avg_PURCHASES', 'Monthly_CASH_ADVANCE', 'LIMIT_USAGE',
                                        'MIN_PAYMENTS_RATIO'))

View(data_New)
```

Clustering

```
# Using KMeans algorithm. Going for 3,4,5 and 6 means ckustering
cluster_three <- kmeans(data_New,3)
cluster_four <- kmeans(data_New,4)
cluster_five <- kmeans(data_New,5)
cluster_six <- kmeans(data_New,6)

clust<-cluster_three$cluster

install.packages("animation")
library(animation)

View(clust)
plot(clust)
rm(c3)
d3<-data_New
c3<-cbind(d3,clust)
View(c3)
plot(c3)

clust<-cluster_four$cluster
d4<-data_New
c4<-cbind(d4,clust)
View(c3)
plot(c4)
```