DESCRIPTION

Background of Problem Statement:

# A UK-based online retail store has captured the sales data for different products for the period of one year (Nov 2016 to Dec 2017). The organization sells gifts primarily on the online platform.The customers who make a purchase consume directly for themselves.There are small businesses that buy in bulk and sell to other customers through the retail outlet channel.

Project Objective:

Find significant customers for the business who make high purchases of their favourite products.The organization wants to roll out a loyalty program to the high-value customers after identificationof segments. Use the clustering methodology to segment customers into groups:

Domain: E-commerce

Dataset Description:

This is a transnational dataset that contains all the transactions occurring between Nov-2016 to Dec-2017 for a UK-based online retail store.

Attribute Description

InvoiceNo Invoice number (A 6-digit integral number uniquely assigned to each transaction)

StockCode Product (item) code

Description Product (item) name

Quantity The quantities of each product (item) per transaction

InvoiceDate The day when each transaction was generated

UnitPrice Unit price (Product price per unit)

CustomerID Customer number (Unique ID assigned to each customer)

Country Country name (The name of the country where each customer resides)

# Installing necessary packages

installpackages("dplyr",dependencies = T)

install.packages("ggplot2", dependencies = T)

install.packages("scales", dependencies = T)

install.packages("NbClust", dependencies = T)

# reading installed packages and atttaching it to project

library(dplyr)

library(ggplot2)

library(NbClust)

library(scales)

# reading data from file

ecom\_data <- read.csv("Ecommerce.csv",header = T)

# data exploration

class(ecom\_data) # class of data

View(ecom\_data)

str(ecom\_data) # type of data stored by each columns

summary(ecom\_data) # mean,median,min,max etc

head(ecom\_data) # first 6 data

dim(ecom\_data) # 541909 9

# Removing redundant column X

ecom\_data\_subset <- subset(ecom\_data, select = -X)

View(ecom\_data\_subset)

# Checking for missing values

length(unique(ecom\_data\_subset$CustomerID)) # 4373

sum(is.na(ecom\_data\_subset$CustomerID)) # 135080

mean(is.na(ecom\_data\_subset)) # 0.02769633 only 2.76 % data are having missing values so we can ignore it since its less than 5 %

# InvoiceNo StockCode Description Quantity InvoiceDate UnitPrice CustomerID

# 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 24.92669

# Country amount\_spent

# 0.00000 0.00000

# From the output we can see that CustomerID has around 25% of missing values which is way too high.

# In order to visualize missing values

install.packages("VIM",dependencies = T)

library(VIM)

aggr\_plot <- aggr(ecom\_data\_subset, col=c('navyblue','red'), numbers=TRUE, sortVars=TRUE, labels=names(ecom\_data\_subset), cex.axis=.7, gap=3,

ylab=c("Histogram of missing data","Pattern"))

A screenshot of a cell phone

Description automatically generated

# Let's see the number of unique invoices and unique customers.

length(unique(ecom\_data\_subset$InvoiceNo))

length(unique(ecom\_data\_subset$CustomerID))

# We now have a dataset of 23,494 unique invoices and 3,951 unique customers.

# Remove Quantity with negative values

pos\_quant <- ecom\_data\_subset[ecom\_data\_subset$Quantity > 0,]

nrow(pos\_quant) # 5,31,285

# changing date format

ecom\_data\_subset$InvoiceDate <- as.Date(ecom\_data\_subset$InvoiceDate,

format = "%d-%b-%y") #23-Nov-17

str(ecom\_data\_subset$InvoiceDate) # Now data class is changed from factor to Date

ecom\_data\_subset$InvoiceNo <- as.integer(ecom\_data\_subset$InvoiceNo)

# Changing Invoice No from factor to int

# Customer clusters vary geographically

# So here we’ll restrict the data to one geographical unit.

table(ecom\_data\_subset$Country)

# Let's see the number of unique invoices and unique customers.

length(unique(ecom\_data\_subset$InvoiceNo))

length(unique(ecom\_data\_subset$CustomerID))

# We now have a dataset of 25,900 unique invoices and 4,373 unique customers.

# We will calculate recency which is the no of days elapsed since customer last order

# and frequency which will refer to the no of invoices with purchase during the year

# It is necessary to distinguish invoices with purchases from invoices with returns.

ecom\_data\_subset$item.return <- grepl("C", ecom\_data\_subset$InvoiceNo, fixed=TRUE)

ecom\_data\_subset$purchase.invoice <- ifelse(ecom\_data\_subset$item.return == "TRUE", 0,1) # Identify returns

# Creating Customer Level Dataset

customers <- as.data.frame(unique(ecom\_data\_subset$CustomerID))

names(customers) <- "CustomerID"

# Adding a recency column by substracting the InvoiceDate from the (last InvoiceDate+1)

ecom\_data\_subset$recency <- as.Date("2017-12-08") - as.Date(ecom\_data\_subset$InvoiceDate)

# remove returns so only consider the data of most recent "purchase"

temp <- subset(ecom\_data\_subset, purchase.invoice == 1)

# Obtain no of days since most recent purchase

recency <- aggregate(recency ~ CustomerID, data=temp, FUN=min, na.rm=TRUE)

remove(temp)

# Add recency to customer data

customers <- merge(customers, recency, by="CustomerID", all=TRUE, sort=TRUE)

remove(recency)

customers$recency <- as.numeric(customers$recency)

### frequency ##########

customer.invoices <- subset(ecom\_data\_subset, select = c("CustomerID","InvoiceNo", "purchase.invoice"))

customer.invoices <- customer.invoices[!duplicated(customer.invoices), ]

customer.invoices <- customer.invoices[order(customer.invoices$CustomerID),]

row.names(customer.invoices) <- NULL

# Number of invoices/year (purchases only)

annual.invoices <- aggregate(purchase.invoice ~ CustomerID, data=customer.invoices, FUN=sum, na.rm=TRUE)

names(annual.invoices)[names(annual.invoices)=="purchase.invoice"] <- "frequency"

# Adding of invoices to customers data

customers <- merge(customers, annual.invoices, by="CustomerID", all=TRUE, sort=TRUE)

remove(customer.invoices, annual.invoices)

range(customers$frequency) # NA NA

table(customers$frequency) # displays some values

# Remove customers who have not made any purchases in the past year

customers <- subset(customers, frequency > 0)

# Calculating Monetary Value for the customers

# Add the column - amount\_spent

ecom\_data\_subset['amount\_spent'] = ecom\_data\_subset['Quantity'] \* ecom\_data\_subset['UnitPrice']

# Aggregated total sales to customer and assigning it to monetary field

total.sales <- aggregate(amount\_spent ~ CustomerID, data = ecom\_data\_subset, FUN=sum, na.rm=TRUE)

names(total.sales)[names(total.sales)=="amount\_spent"] <- "monetary"

# Adding monetary value to customers dataset

customers <- merge(customers, total.sales, by="CustomerID", all.x=TRUE, sort=TRUE)

remove(total.sales)

# Identify customers with negative monetary value numbers, as they were presumably

# returning purchases from the preceding year

hist(customers$monetary)

A screenshot of a cell phone

Description automatically generated

customers$monetary <- ifelse(customers$monetary < 0, 0, customers$monetary)

# reset negative numbers to zero

hist(customers$monetary)

A screenshot of a cell phone

Description automatically generated

# highly valued top 20% and bottom 80 % customers

customers <- customers[order(-customers$monetary),]

high.cutoff <- 0.8 \* sum(customers$monetary)

customers$high <- ifelse(cumsum(customers$monetary) <= high.cutoff, "Top 20%", "Bottom 80%")

customers$high <- factor(customers$high, levels=c("Top 20%", "Bottom 80%"), ordered=TRUE)

levels(customers$high) # "Top 20%" "Bottom 80%"

round(prop.table(table(customers$high)), 2) # Top 20% Bottom 80%

0.27 0.73

remove(high.cutoff)

customers <- customers[order(customers$CustomerID),]

###################

# Preprocessing Of DataSet #

###################

# Log-transform positively-skewed variables

customers$recency.log <- log(customers$recency)

customers$frequency.log <- log(customers$frequency)

customers$monetary.log <- customers$monetary + 0.1 # can't take log(0), so add a small value to remove zeros

customers$monetary.log <- log(customers$monetary.log)

# Z-scores to scale each of the data

customers$recency.z <- scale(customers$recency.log, center=TRUE, scale=TRUE)

customers$frequency.z <- scale(customers$frequency.log, center=TRUE, scale=TRUE)

customers$monetary.z <- scale(customers$monetary.log, center=TRUE, scale=TRUE)

View(customers)

A close up of a piece of paper

Description automatically generated

### Data Visualization ##########

library(ggplot2)

library(scales)

# Original scale

plot1 <- ggplot(customers, aes(x = frequency, y = monetary))

plot1 <- plot1 + geom\_point(aes(colour = recency, shape = pareto))

plot1 <- plot1 + scale\_shape\_manual(name = "80/20 Designation", values=c(17, 16))

plot1 <- plot1 + scale\_colour\_gradient(name="Recency\n(Days since Last Purchase))")

plot1 <- plot1 + scale\_y\_continuous(label=dollar)

plot1 <- plot1 + xlab("Frequency (Number of Purchases)")

plot1 <- plot1 + ylab("Monetary Value of Customer (Annual Sales)")

plot1

A screenshot of a cell phone

Description automatically generated

# This first graph uses the variables’ original metrics and is almost completely uninterpretable.There’s a clump of data points in the lower left-hand corner of the plot, and then a few outliers.This is why we log-transformed the input variables.

# Log-transformed

plot2 <- ggplot(customers, aes(x = frequency.log, y = monetary.log))

plot2 <- plot2+ geom\_point(aes(colour = recency.log, shape = pareto))

plot2<- plot2+ scale\_shape\_manual(name = "80/20 Designation", values=c(17, 16))

plot2<- plot2+ scale\_colour\_gradient(name="Log-transformed Recency")

plot2<- plot2+ xlab("Log-transformed Frequency")

plot2<- plot2+ ylab("Log-transformed Monetary Value of Customer")

plot2

A screenshot of a computer

Description automatically generated

# Scaled variables

plot3 <- ggplot(customers, aes(x = frequency.z, y = monetary.z))

plot3 <- plot3 + geom\_point(aes(colour = recency.z, shape = pareto))

plot3 <- plot3 + scale\_shape\_manual(name = "80/20 Designation", values=c(17, 16))

plot3 <- plot3 + scale\_colour\_gradient(name="Z-scored Recency")

plot3 <- plot3 + xlab("Z-scored Frequency")

plot3 <- plot3 + ylab("Z-scored Monetary Value of Customer")

plot3

A screenshot of a cell phone

Description automatically generated

Analysis tasks to be performed:

Use the clustering methodology to segment customers into groups:

Use the following clustering algorithms:

K means

Hierarchical

• Identify the right number of customer segments.

• Provide the number of customers who are highly valued.

• Identify the clustering algorithm that gives maximum accuracy and explains robust clusters.

• If the number of observations is loaded in one of the clusters, break down that cluster further using the clustering algorithm.

[ hint: Here loaded means if any cluster has more number of data points as compared to other clusters then split that clusters by increasing the number of clusters and observe, compare the results with previous results.]

########## Determining number of clusters through K-Means #####################

preprocessed <- customers[,9:11]

clustmax <- 10 # specify the maximum number of clusters you want to try out

models <- data.frame(k=integer(),

tot.withinss=numeric(),

betweenss=numeric(),

totss=numeric(),

rsquared=numeric())

for (k in 1: clustmax)

{

print(k)

# Run kmeans

# nstart = number of initial configurations; the best one is used

# $iter will return the iteration used for the final model

output <- kmeans(preprocessed, centers = k, nstart = 20)

# Add cluster membership to customers dataset

var.name <- paste("cluster", k, sep="\_")

customers[,(var.name)] <- output$cluster

customers[,(var.name)] <- factor(customers[,(var.name)], levels = c(1:k))

# Graph clusters

cluster\_graph <- ggplot(customers, aes(x = frequency.log, y = monetary.log))

cluster\_graph <- cluster\_graph + geom\_point(aes(colour = customers[,(var.name)]))

colors <-c('red','orange','green3','deepskyblue','blue','darkorchid4','violet','pink1','tan3','black')

cluster\_graph <- cluster\_graph + scale\_colour\_manual(name = "Cluster Group", values=colors)

cluster\_graph <- cluster\_graph + xlab("Log-transformed Frequency")

cluster\_graph <- cluster\_graph + ylab("Log-transformed Monetary Value of Customer")

title <- paste("k-means Solution with", k, sep=" ")

title <- paste(title, "Clusters", sep=" ")

cluster\_graph <- cluster\_graph + ggtitle(title)

print(cluster\_graph)

# Cluster centers in original metrics

library(plyr)

print(title)

cluster\_centers <- ddply(customers, .(customers[,(var.name)]), summarize,

monetary=round(median(monetary),2),# use median b/c this is the raw, heavily-skewed data

frequency=round(median(frequency),1),

recency=round(median(recency), 0))

names(cluster\_centers)[names(cluster\_centers)=="customers[, (var.name)]"] <- "Cluster"

print(cluster\_centers)

cat("\n")

cat("\n")

# Collect model information

models[k,("k")] <- k

models[k,("tot.withinss")] <- output$tot.withinss # the sum of all within sum of squares

models[k,("betweenss")] <- output$betweenss

models[k,("totss")] <- output$totss # betweenss + tot.withinss

# percentage of variance explained by cluster membership

models[k,("rsquared")] <- round(output$betweenss/output$totss, 3)

assign("models", models, envir = .GlobalEnv)

}

# Graph variance explained by number of clusters

r2\_graph <- ggplot(models, aes(x = k, y = rsquared))

r2\_graph <- r2\_graph + geom\_point() + geom\_line()

r2\_graph <- r2\_graph + scale\_y\_continuous(labels = scales::percent)

r2\_graph <- r2\_graph + scale\_x\_continuous(breaks = 1:clustmax)

r2\_graph <- r2\_graph + xlab("k (Number of Clusters)")

r2\_graph <- r2\_graph + ylab("Variance Explained")

r2\_graph

A picture containing water, table, sitting, white

Description automatically generated

# From the graphs we can see that a 2 cluster solution explains only 48% of variance and that cannot be taken into consideration with business strategy targeted for customers.

Also 5 cluster solutions provides 76% variance around which is good but there is no break at that point as evident from the graph

# Graph within sums of squares by number of clusters

# Look for a "bend" in the graph, as with a scree plot

ss\_graph <- ggplot(models, aes(x = k, y = tot.withinss))

ss\_graph <- ss\_graph + geom\_point() + geom\_line()

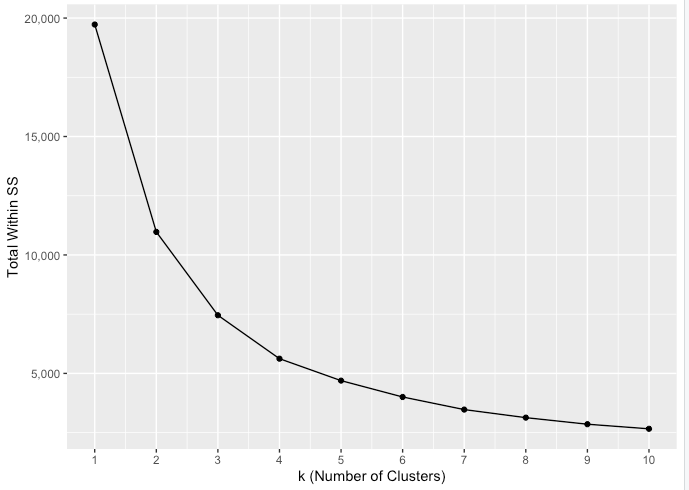
ss\_graph <- ss\_graph + scale\_x\_continuous(breaks = 1:clustmax)

ss\_graph <- ss\_graph + scale\_y\_continuous(labels = scales::comma)

ss\_graph <- ss\_graph + xlab("k (Number of Clusters)")

ss\_graph <- ss\_graph + ylab("Total Within SS")

ss\_graph

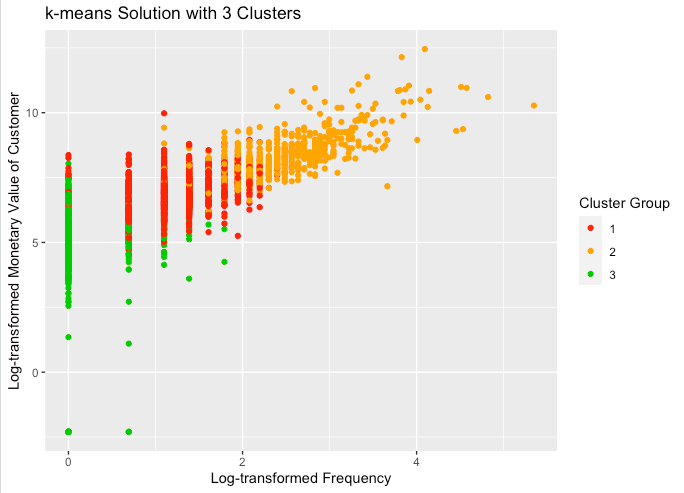


A picture containing table, white

Description automatically generated

A screenshot of a cell phone

Description automatically generated



A screenshot of a cell phone

Description automatically generated

A picture containing table

Description automatically generated

A screenshot of text

Description automatically generated

A picture containing text, receipt

Description automatically generated

A close up of text on a white background

Description automatically generated

A picture containing white

Description automatically generated

#########################################################

# Using NbClust metrics to determine number of clusters #

library(NbClust)

set.seed(1)

nc <- NbClust(preprocessed, min.nc=2, max.nc=7, method="kmeans")

\*\*\* : The Hubert index is a graphical method of determining the number of clusters.

In the plot of Hubert index, we seek a significant knee that corresponds to a

significant increase of the value of the measure i.e the significant peak in Hubert

index second differences plot.

\*\*\* : The D index is a graphical method of determining the number of clusters.

In the plot of D index, we seek a significant knee (the significant peak in Dindex

second differences plot) that corresponds to a significant increase of the value of

the measure.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Among all indices:

\* 8 proposed 2 as the best number of clusters

\* 2 proposed 3 as the best number of clusters

\* 2 proposed 4 as the best number of clusters

\* 9 proposed 5 as the best number of clusters

\* 2 proposed 6 as the best number of clusters

\*\*\*\*\* Conclusion \*\*\*\*\*

A close up of a map

Description automatically generated\* According to the majority rule, the best number of clusters is 5

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

table(nc$Best.n[1,])

0 1 2 3 4 5 6

2 1 8 2 2 9 2

nc$All.index # estimates for each number of clusters on 26 different metrics of model fit

A close up of a newspaper

Description automatically generated

barplot(table(nc$Best.n[1,]),

xlab="Number of Clusters", ylab="Number of Criteria",

main="Number of Clusters Chosen by Criteria")

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Inference : Cluster No 5 is high monetary value,high frequency ,recent purchase group and hence can be identified as a high valued customer segment and should be the most ideal to roll out the loyalty program