Week 14 IP

DImensionaity Reduction.

Specifying the Question

Carrefour Kenya requires information about the marketing department, to identify the most relevant marketing strategies that will result in increased sales.

Defining the Metrics for Success.

The metric for success should be to beable to provide an marketing strategy that can improve the sales at Carrefour Kenya.

Understanding the Context

You are a Data analyst at Carrefour Kenya and are currently undertaking a project that will inform the marketing department on the most relevant marketing strategies that will result in the highest no. of sales (total price including tax). Your project has been divided into four parts where you'll explore a recent marketing dataset by performing various unsupervised learning techniques and later providing recommendations based on your insights.

Part 1: Dimensionality Reduction

This section of the project entails reducing your dataset to a low dimensional dataset using the t-SNE algorithm or PCA. You will be required to perform your analysis and provide insights gained from your analysis.

Part 2: Feature Selection

This section requires you to perform feature selection through the use of the unsupervised learning methods learned earlier this week. You will be required to perform your analysis and provide insights on the features that contribute the most information to the dataset.

Part 1

Loading Libraries

```
library(data.table)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse
1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
```

```
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.0.5

## -- Conflicts ---------------------------------------
tidyverse_conflicts() --
## x dplyr::between()   masks data.table::between()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::transpose() masks data.table::transpose()

library(dplyr)
library(tibble)
library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at
https://goo.gl/ve3WBa

library(caret)

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

Loading the dataset

```
# Loading the dataset for dimensionality reduction
c4 <- fread('http://bit.ly/CarreFourDataset')
# Loading the dataset as a dataframe
df = as.data.frame(c4)
head(df)

##    Invoice ID Branch Customer type Gender           Product line Unit
price
## 1 750-67-8428      A        Member Female      Health and beauty
74.69
## 2 226-31-3081      C        Normal Female Electronic accessories
15.28
## 3 631-41-3108      A        Normal   Male     Home and lifestyle
46.33
## 4 123-19-1176      A        Member   Male      Health and beauty
58.22
## 5 373-73-7910      A        Normal   Male      Sports and travel
86.31
## 6 699-14-3026      C        Normal   Male Electronic accessories
```

```
85.39
##   Quantity     Tax      Date  Time     Payment   cogs gross margin
percentage
## 1        7 26.1415  1/5/2019 13:08     Ewallet 522.83
4.761905
## 2        5  3.8200  3/8/2019 10:29        Cash  76.40
4.761905
## 3        7 16.2155  3/3/2019 13:23 Credit card 324.31
4.761905
## 4        8 23.2880 1/27/2019 20:33     Ewallet 465.76
4.761905
## 5        7 30.2085  2/8/2019 10:37     Ewallet 604.17
4.761905
## 6        7 29.8865 3/25/2019 18:30     Ewallet 597.73
4.761905
##   gross income Rating    Total
## 1      26.1415    9.1 548.9715
## 2       3.8200    9.6  80.2200
## 3      16.2155    7.4 340.5255
## 4      23.2880    8.4 489.0480
## 5      30.2085    5.3 634.3785
## 6      29.8865    4.1 627.6165

# Size of the dataset
glimpse(df)

## Rows: 1,000
## Columns: 16
## $ `Invoice ID`              <chr> "750-67-8428", "226-31-3081", "631-41-
3108",~
## $ Branch                    <chr> "A", "C", "A", "A", "A", "C", "A", "C",
"A",~
## $ `Customer type`           <chr> "Member", "Normal", "Normal", "Member",
"Nor~
## $ Gender                    <chr> "Female", "Female", "Male", "Male",
"Male", ~
## $ `Product line`            <chr> "Health and beauty", "Electronic
accessories~
## $ `Unit price`              <dbl> 74.69, 15.28, 46.33, 58.22, 86.31,
85.39, 68~
## $ Quantity                  <int> 7, 5, 7, 8, 7, 7, 6, 10, 2, 3, 4, 4, 5,
10, ~
## $ Tax                       <dbl> 26.1415, 3.8200, 16.2155, 23.2880,
30.2085, ~
## $ Date                      <chr> "1/5/2019", "3/8/2019", "3/3/2019",
"1/27/20~
## $ Time                      <chr> "13:08", "10:29", "13:23", "20:33",
"10:37",~
## $ Payment                   <chr> "Ewallet", "Cash", "Credit card",
"Ewallet",~
```

```
## $ cogs                      <dbl> 522.83, 76.40, 324.31, 465.76, 604.17,
597.7~
## $ `gross margin percentage` <dbl> 4.761905, 4.761905, 4.761905, 4.761905,
4.76~
## $ `gross income`            <dbl> 26.1415, 3.8200, 16.2155, 23.2880,
30.2085, ~
## $ Rating                    <dbl> 9.1, 9.6, 7.4, 8.4, 5.3, 4.1, 5.8, 8.0,
7.2,~
## $ Total                     <dbl> 548.9715, 80.2200, 340.5255, 489.0480,
634.3~
```

- The dataset contains 1000 rows and 16 columns.

```
# Structure of the dataframe
str(df)
```

```
## 'data.frame':    1000 obs. of  16 variables:
##  $ Invoice ID            : chr  "750-67-8428" "226-31-3081" "631-41-3108"
"123-19-1176" ...
##  $ Branch                : chr  "A" "C" "A" "A" ...
##  $ Customer type         : chr  "Member" "Normal" "Normal" "Member" ...
##  $ Gender                : chr  "Female" "Female" "Male" "Male" ...
##  $ Product line          : chr  "Health and beauty" "Electronic
accessories" "Home and lifestyle" "Health and beauty" ...
##  $ Unit price            : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ Quantity              : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ Tax                   : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ Date                  : chr  "1/5/2019" "3/8/2019" "3/3/2019"
"1/27/2019" ...
##  $ Time                  : chr  "13:08" "10:29" "13:23" "20:33" ...
##  $ Payment               : chr  "Ewallet" "Cash" "Credit card" "Ewallet"
...
##  $ cogs                  : num  522.8 76.4 324.3 465.8 604.2 ...
##  $ gross margin percentage: num  4.76 4.76 4.76 4.76 4.76 ...
##  $ gross income          : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ Rating                : num  9.1 9.6 7.4 8.4 5.3 4.1 5.8 8 7.2 5.9 ...
##  $ Total                 : num  549 80.2 340.5 489 634.4 ...
```

- The dataframe contains character, numeric and integer datatypes

```
# Changing date column to datetime format
#library(anytime)
#df$date <- anytime::anydate(df$date)
# Checking the datatype for the date column
str(df$date)
```

```
##  NULL
```

Data cleaning.

Columns.

```
# Changing column names to lower case, and replacing spaces with underscores
colnames(df) = tolower(str_replace_all(colnames(df), c(' ' = '_')))
# Checking column names.
colnames(df)
```

```
##  [1] "invoice_id"             "branch"
##  [3] "customer_type"          "gender"
##  [5] "product_line"           "unit_price"
##  [7] "quantity"               "tax"
##  [9] "date"                   "time"
## [11] "payment"                "cogs"
## [13] "gross_margin_percentage" "gross_income"
## [15] "rating"                 "total"
```

- Column names have been changed to lower case and spaces replaced with underscores

Missing Values

```
# Checking for null values
colSums(is.na(df))
```

```
##              invoice_id                  branch           customer_type
##                       0                       0                       0
##                  gender            product_line              unit_price
##                       0                       0                       0
##                quantity                     tax                    date
##                       0                       0                       0
##                    time                 payment                    cogs
##                       0                       0                       0
## gross_margin_percentage            gross_income                  rating
##                       0                       0                       0
##                   total
##                       0
```

- No null values were found

Duplicates.

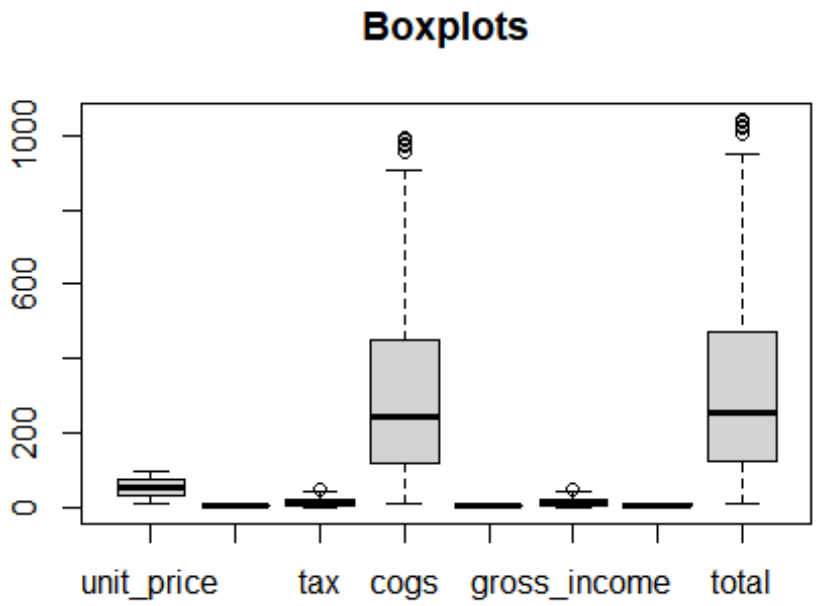
```
# Checking for duplicates
sum(duplicated(df))
```

```
## [1] 0
```

- No duplicates were found

Outliers

```
# Isolating numerical columns
df_num <- df[,!sapply(df, is.character)]
boxplot(df_num,main='Boxplots')
```



* Some outliers exist in the tax, cogs, gross_income and Total columns.

PCA

```
# Dummify the data
#dmy <- dummyVars(" ~ .", data = df)
#df_dummy <- data.frame(predict(dmy, newdata = df))
#head(df_dummy)

# We then pass df to the prcomp(). We also set two arguments, center and
scale,
# to be TRUE then preview our object with summary
## df_pca <- prcomp(df_dummy, center = TRUE, scale. = TRUE)
#summary(df_pca)

#rlang::last_error()
```

DImensionality Reduction: PCA

```
str(df_num)
```

```
## 'data.frame':    1000 obs. of  8 variables:
##  $ unit_price             : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ quantity               : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ tax                    : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ cogs                   : num  522.8 76.4 324.3 465.8 604.2 ...
##  $ gross_margin_percentage: num  4.76 4.76 4.76 4.76 4.76 ...
##  $ gross_income           : num  26.14 3.82 16.22 23.29 30.21 ...
```

```
##  $ rating                 : num  9.1 9.6 7.4 8.4 5.3 4.1 5.8 8 7.2 5.9 ...
##  $ total                  : num  549 80.2 340.5 489 634.4 ...

# Checking the structure of the dataframe
str(df)

## 'data.frame':    1000 obs. of  16 variables:
##  $ invoice_id             : chr  "750-67-8428" "226-31-3081" "631-41-3108"
"123-19-1176" ...
##  $ branch                 : chr  "A" "C" "A" "A" ...
##  $ customer_type          : chr  "Member" "Normal" "Normal" "Member" ...
##  $ gender                 : chr  "Female" "Female" "Male" "Male" ...
##  $ product_line           : chr  "Health and beauty" "Electronic
accessories" "Home and lifestyle" "Health and beauty" ...
##  $ unit_price             : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ quantity               : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ tax                    : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ date                   : chr  "1/5/2019" "3/8/2019" "3/3/2019"
"1/27/2019" ...
##  $ time                   : chr  "13:08" "10:29" "13:23" "20:33" ...
##  $ payment                : chr  "Ewallet" "Cash" "Credit card" "Ewallet"
...
##  $ cogs                   : num  522.8 76.4 324.3 465.8 604.2 ...
##  $ gross_margin_percentage: num  4.76 4.76 4.76 4.76 4.76 ...
##  $ gross_income           : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ rating                 : num  9.1 9.6 7.4 8.4 5.3 4.1 5.8 8 7.2 5.9 ...
##  $ total                  : num  549 80.2 340.5 489 634.4 ...

# Creating clusters for the total column, which is our target column
summary(df$total)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.68  124.42  253.85  322.97  471.35 1042.65

# Identifying the breaks using quantiles.
quantiles<-c(0,124.42,471.35,Inf)
# Assigning names for the clusters
clusters<-c("low","medium","high")
# Creating the clusters for total
df$total_clusters <-cut(df$total, breaks=quantiles, labels = clusters)
head(df$total_clusters)

## [1] high   low    medium high   high   high
## Levels: low medium high
```

- The clusters have been successfully created

```
str(df)

## 'data.frame':    1000 obs. of  17 variables:
##  $ invoice_id             : chr  "750-67-8428" "226-31-3081" "631-41-3108"
"123-19-1176" ...
```

```
##  $ branch                 : chr  "A" "C" "A" "A" ...
##  $ customer_type          : chr  "Member" "Normal" "Normal" "Member" ...
##  $ gender                 : chr  "Female" "Female" "Male" "Male" ...
##  $ product_line           : chr  "Health and beauty" "Electronic
accessories" "Home and lifestyle" "Health and beauty" ...
##  $ unit_price             : num  74.7 15.3 46.3 58.2 86.3 ...
##  $ quantity               : int  7 5 7 8 7 7 6 10 2 3 ...
##  $ tax                    : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ date                   : chr  "1/5/2019" "3/8/2019" "3/3/2019"
"1/27/2019" ...
##  $ time                   : chr  "13:08" "10:29" "13:23" "20:33" ...
##  $ payment                : chr  "Ewallet" "Cash" "Credit card" "Ewallet"
...
##  $ cogs                   : num  522.8 76.4 324.3 465.8 604.2 ...
##  $ gross_margin_percentage: num  4.76 4.76 4.76 4.76 4.76 ...
##  $ gross_income           : num  26.14 3.82 16.22 23.29 30.21 ...
##  $ rating                 : num  9.1 9.6 7.4 8.4 5.3 4.1 5.8 8 7.2 5.9 ...
##  $ total                  : num  549 80.2 340.5 489 634.4 ...
##  $ total_clusters         : Factor w/ 3 levels "low","medium",..: 3 1 2 3
3 3 2 3 1 2 ...
```

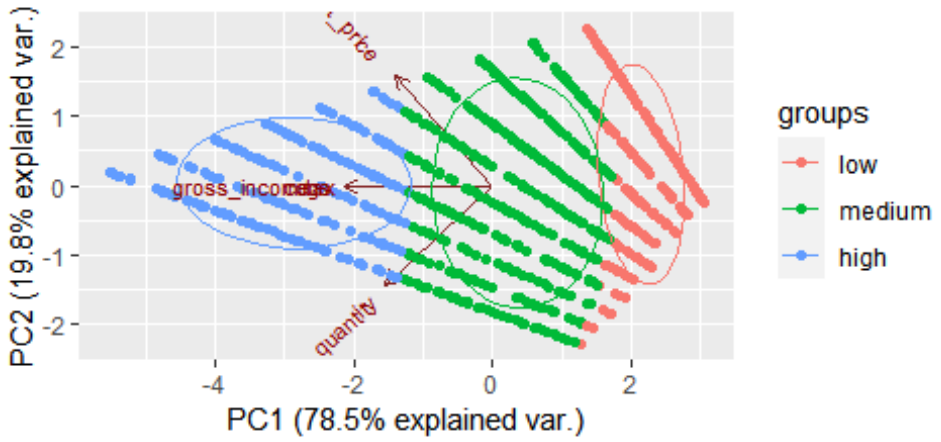
```
# Selecting columns for PCA
dfnum <- df[,c(6,7,8,12,14)]
# Applying PCA on the numerical columns
df.pca <- prcomp(dfnum, center = TRUE, scale. = TRUE)
summary(df.pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3       PC4       PC5
## Standard deviation     1.9814 0.9946 0.29132 2.511e-16 1.472e-16
## Proportion of Variance 0.7852 0.1979 0.01697 0.000e+00 0.000e+00
## Cumulative Proportion  0.7852 0.9830 1.00000 1.000e+00 1.000e+00
```

- The first two principal components, i.e, PC1 and PC2 contribute the highest percentage of variance.

```
# Plotting the PCA
library(devtools)

## Warning: package 'devtools' was built under R version 4.0.5

## Loading required package: usethis

## Warning: package 'usethis' was built under R version 4.0.5

library(ggbiplot)

## Loading required package: plyr

## ------------------------------------------------------------------------
----
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first,
then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------
----

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:purrr':
##
##     compact

## Loading required package: scales

##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor

## Loading required package: grid

ggbiplot(df.pca, groups=df$total_clusters, ellipse=TRUE, obs.scale=1,
var.scale=1)
```



* The clusters for total spending have been created, with definition of customers who were low, medium or high spenders. The marketing strategy should be formulated so as to identify the best commodities or package of commodities that should be marketed to these three groups of customers. The three groups have been created based on their * unit_price
* quantity
* tax * cogs * gross_income * rating

t-SNE

```
# Calling the dataframe
head(df)
```

```
##    invoice_id branch customer_type gender           product_line
unit_price
## 1 750-67-8428      A        Member Female      Health and beauty
74.69
## 2 226-31-3081      C        Normal Female Electronic accessories
15.28
## 3 631-41-3108      A        Normal   Male     Home and lifestyle
46.33
## 4 123-19-1176      A        Member   Male      Health and beauty
58.22
## 5 373-73-7910      A        Normal   Male      Sports and travel
86.31
## 6 699-14-3026      C        Normal   Male Electronic accessories
85.39
##   quantity     tax      date  time     payment   cogs
gross_margin_percentage
```

```
## 1        7 26.1415  1/5/2019 13:08     Ewallet 522.83
4.761905
## 2        5  3.8200  3/8/2019 10:29        Cash  76.40
4.761905
## 3        7 16.2155  3/3/2019 13:23 Credit card 324.31
4.761905
## 4        8 23.2880 1/27/2019 20:33     Ewallet 465.76
4.761905
## 5        7 30.2085  2/8/2019 10:37     Ewallet 604.17
4.761905
## 6        7 29.8865 3/25/2019 18:30     Ewallet 597.73
4.761905
##   gross_income rating    total total_clusters
## 1      26.1415    9.1 548.9715           high
## 2       3.8200    9.6  80.2200            low
## 3      16.2155    7.4 340.5255         medium
## 4      23.2880    8.4 489.0480           high
## 5      30.2085    5.3 634.3785           high
## 6      29.8865    4.1 627.6165           high
```

```
# Loading our t-SNE library
#
library(Rtsne)
```

```
## Warning: package 'Rtsne' was built under R version 4.0.5
```

```
# Curating the database for analysis
#
Labels<-df$total
df$total<-as.factor(df$total)

# For plotting
#
colors = rainbow(length(unique(df$total)))
names(colors) = unique(df$total)

# Executing the algorithm on curated data
#
tsne <- Rtsne(df[,-8], dims = 2, perplexity=30, verbose=TRUE, max_iter = 500)
```

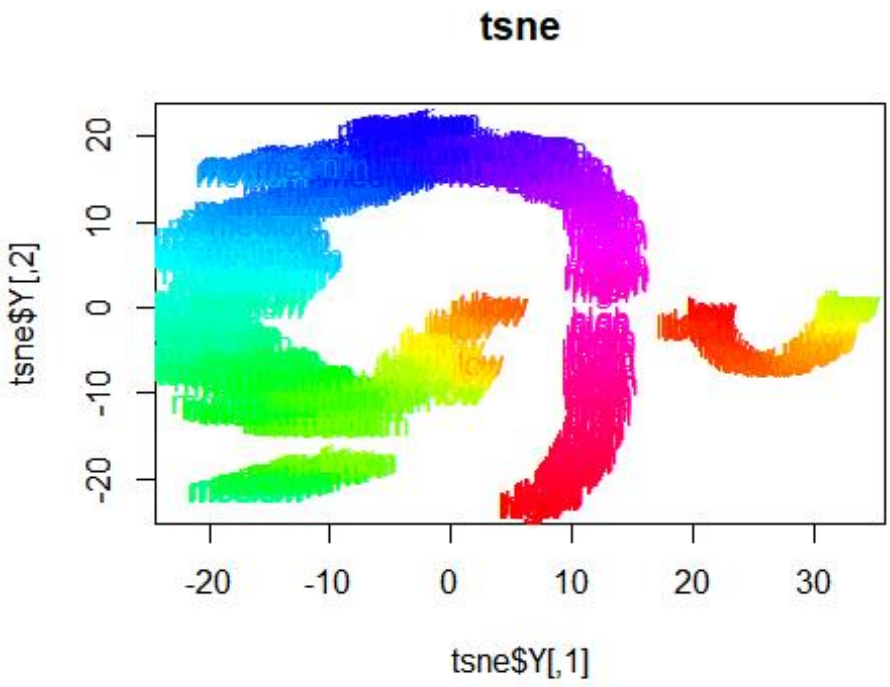
```
## Performing PCA
## Read the 1000 x 50 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.25 seconds (sparsity = 0.102670)!
## Learning embedding...
## Iteration 50: error is 60.345115 (50 iterations in 0.24 seconds)
## Iteration 100: error is 53.322531 (50 iterations in 0.19 seconds)
## Iteration 150: error is 52.385699 (50 iterations in 0.19 seconds)
```

```
## Iteration 200: error is 52.092779 (50 iterations in 0.20 seconds)
## Iteration 250: error is 51.978535 (50 iterations in 0.21 seconds)
## Iteration 300: error is 0.601193 (50 iterations in 0.19 seconds)
## Iteration 350: error is 0.438332 (50 iterations in 0.20 seconds)
## Iteration 400: error is 0.397449 (50 iterations in 0.22 seconds)
## Iteration 450: error is 0.381205 (50 iterations in 0.24 seconds)
## Iteration 500: error is 0.373477 (50 iterations in 0.22 seconds)
## Fitting performed in 2.11 seconds.
```

```
# Getting the duration of execution
#
exeTimeTsne <- system.time(Rtsne(df[,-8], dims = 2, perplexity=30,
verbose=TRUE, max_iter = 500))
```

```
## Performing PCA
## Read the 1000 x 50 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.26 seconds (sparsity = 0.102670)!
## Learning embedding...
## Iteration 50: error is 60.386778 (50 iterations in 0.24 seconds)
## Iteration 100: error is 52.775527 (50 iterations in 0.19 seconds)
## Iteration 150: error is 51.715687 (50 iterations in 0.21 seconds)
## Iteration 200: error is 51.304083 (50 iterations in 0.24 seconds)
## Iteration 250: error is 51.040660 (50 iterations in 0.21 seconds)
## Iteration 300: error is 0.563582 (50 iterations in 0.22 seconds)
## Iteration 350: error is 0.417526 (50 iterations in 0.20 seconds)
## Iteration 400: error is 0.371671 (50 iterations in 0.20 seconds)
## Iteration 450: error is 0.366291 (50 iterations in 0.23 seconds)
## Iteration 500: error is 0.359583 (50 iterations in 0.21 seconds)
## Fitting performed in 2.16 seconds.
```

```
# Plotting our graph and closely examining the graph
#
plot(tsne$Y, t='n', main="tsne")
text(tsne$Y, labels=df$total_clusters, col=colors[df$total])
```



- The t-SNE has created multiple clusters for the different categories of customers, as indicated by the different colours in the plot