UNSUPERVISED LEARNING TECHNIQUES USING R PROGRAMMING

CHEROTICH FAITH

2022-06-10

INTRODUCTION

Identifying the Research Question

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).

Specifying the Research Question

To identify the most relevant marketing strategies that will result in the highest no. of sales (total price including tax) in the marketing department at Carrefour Market in Kenya.

Defining the Metric of Success

Our project will be considered a success if we can successfully identify the most relevant marketing strategies which will help in making the highest number of sales at CarreFour Market.

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.

Recording the Experimental Design

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.

DATA PREPARATION

```r
#Loading the dataset
setwd("C:/Users/user/Downloads/")

supermarket <- read.csv("Supermarket_Dataset_1 - Sales Data.csv")
```

```r
#preview the top 6 records of the dataset
head(supermarket)
```

```
##    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
```

DATA UNDERSTANDING

```r
#let's preview the number of rows and columns in our dataset using the dimension function, dim()

dim(supermarket)
```

```
## [1] 1000   16
```

Our dataset has 1000 rows and 16 columns

```r
#let's preview our data types using the str() function

str(supermarket)
```

```
## '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" "
##  $ 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 ...
```

```
#let's use the sapply() function to get a better summary of our variables datatypes
sapply(supermarket,class)
```

```
##              Invoice.ID                  Branch           Customer.type 
##             "character"             "character"             "character" 
##                  Gender            Product.line              Unit.price 
##             "character"             "character"               "numeric" 
##                Quantity                     Tax                    Date 
##               "integer"               "numeric"             "character" 
##                    Time                 Payment                    cogs 
##             "character"             "character"               "numeric" 
## gross.margin.percentage            gross.income                  Rating 
##               "numeric"               "numeric"               "numeric" 
##                   Total 
##               "numeric"
```

Most of our variables have the appropriate data types.Therefore, we proceed with our analysis

DATA CLEANING

Checking for Missing values

```
#Let's check for missing values in each column using the ColSums()function
colSums(is.na(supermarket))
```

```
##              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
```

```
#hence print the sum of missing values
sum(is.na(supermarket))
```

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

There are no missing values in our dataset

Checking for duplicated Values

```
#we use the duplicated()function to check for duplicated values in our dataset
duplicated_values <- supermarket[duplicated(supermarket),]


#hence we print the sum of our duplicates
sum(duplicated(supermarket))
```

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

From the output, there is zero duplicated variables

Checking for Outliers

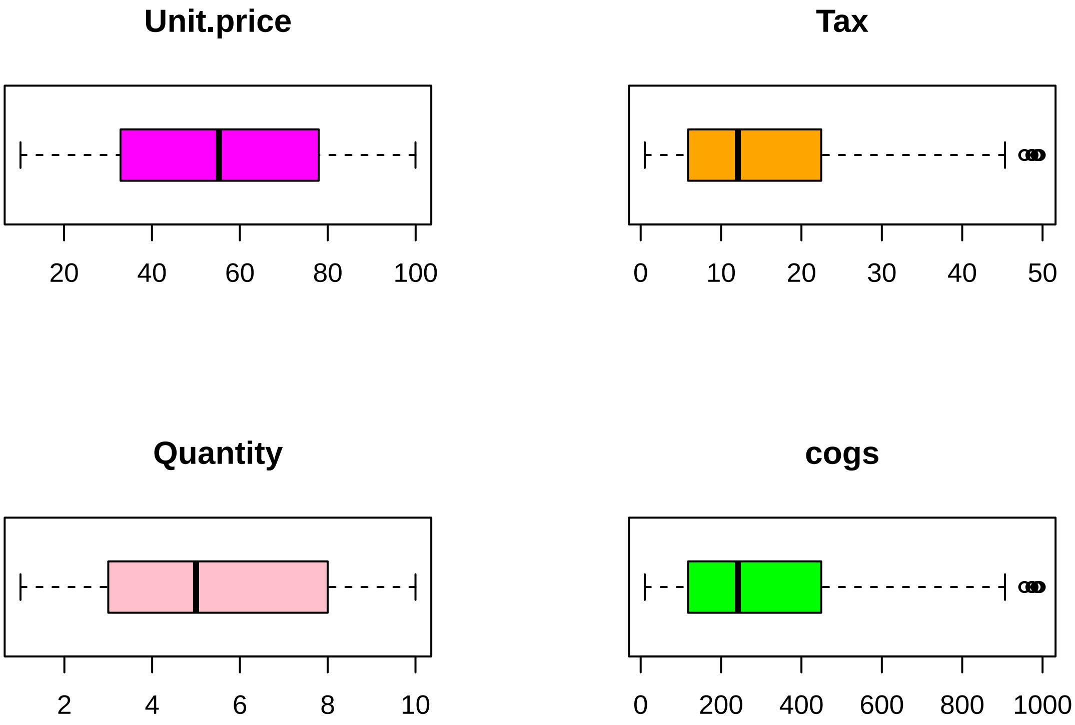
```
#checking for outliers in all the numerical variables of our dataset and visualizing them using the box

#Let's first, recall our datatypes
sapply(supermarket,class)
```

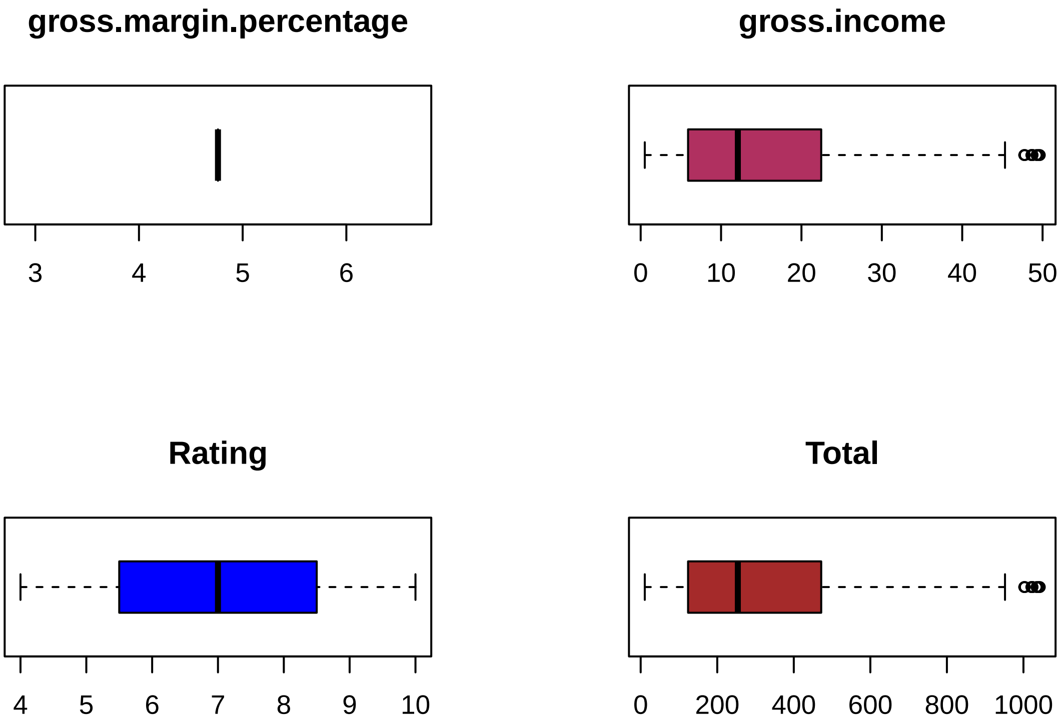
```
##              Invoice.ID                  Branch           Customer.type
##             "character"             "character"             "character"
##                  Gender            Product.line              Unit.price
##             "character"             "character"               "numeric"
##                Quantity                     Tax                    Date
##               "integer"               "numeric"             "character"
##                    Time                 Payment                    cogs
##             "character"             "character"               "numeric"
## gross.margin.percentage            gross.income                  Rating
##               "numeric"               "numeric"               "numeric"
##                   Total
##               "numeric"
```

```
#plotting boxplots for the numerical variables

par(mfrow=c(2,2))
boxplot((supermarket$`Unit.price`),horizontal=TRUE, col='Magenta', main="Unit.price")
boxplot((supermarket$`Tax`), horizontal =TRUE, col='orange', main="Tax")
boxplot((supermarket$`Quantity`), horizontal =TRUE, col='pink', main="Quantity")
boxplot((supermarket$`cogs`),horizontal= TRUE, col='green', main="cogs")
```



```
boxplot((supermarket$`gross.margin.percentage`),horizontal=TRUE, col='yellow', main="gross.margin.percen
boxplot((supermarket$`gross.income`),horizontal= TRUE, col='maroon', main="gross.income")
boxplot((supermarket$`Rating`),horizontal=TRUE, col='blue', main="Rating")
boxplot((supermarket$`Total`),horizontal=TRUE, col='brown', main="Total")
```



a. There are no outliers in the unit.price column

b. There are outliers in the tax column. This is because different products have different tax rates based on their quality, quantity and their prices.

c.There are no oultiers in the Quantity column

d.There are outliers in the Cogs(cost of goods sold) column. This is because different good are sold at different prices and at different seasons.

e. In the dataset, we were given the same value for the gross margin percentage. This explains the above boxplot output.

f. Gross income column has outliers as a result of different sale of different products with different price rates

g.There are no oultiers in the rating column

h. The Total column also has outliers which is due to the different unit prices, gross income, cost of goods sold and sale of different products.

Dropping the unnecessary columns

```r
# dropping the "Invoice.ID" column
supermarket$Invoice.ID <- NULL
```

Splitting the date column

```
# we split the Date column into Day, Month, and Year columns, and store the results as factors
supermarket$Day <- as.factor(format(as.POSIXct(supermarket$Date, format="%m/%d/%Y"), "%d"))
supermarket$Month <- as.factor(format(as.POSIXct(supermarket$Date, format="%m/%d/%Y"), "%m"))
supermarket$Year <- as.factor(format(as.POSIXct(supermarket$Date, format="%m/%d/%Y"), "%Y"))

# we also split the Time variable into Hour and Minute, and store the results as factors
supermarket$Hour <- as.factor(format(as.POSIXct(supermarket$Time, format="%H:%M"), "%H"))
supermarket$Minute <- as.factor(format(as.POSIXct(supermarket$Time, format="%H:%M"), "%M"))

# Further, we drop "Date" and "Time" since they are no longer useful
supermarket$Date <- NULL
supermarket$Time <- NULL

# we then preview our dataset after the changes
head(supermarket)
```

```
##   Branch Customer.type Gender           Product.line Unit.price Quantity
## 1      A        Member Female      Health and beauty      74.69        7
## 2      C        Normal Female Electronic accessories      15.28        5
## 3      A        Normal   Male     Home and lifestyle      46.33        7
## 4      A        Member   Male      Health and beauty      58.22        8
## 5      A        Normal   Male      Sports and travel      86.31        7
## 6      C        Normal   Male Electronic accessories      85.39        7
##       Tax     Payment   cogs gross.margin.percentage gross.income Rating
## 1 26.1415     Ewallet 522.83                4.761905      26.1415    9.1
## 2  3.8200        Cash  76.40                4.761905       3.8200    9.6
## 3 16.2155 Credit card 324.31                4.761905      16.2155    7.4
## 4 23.2880     Ewallet 465.76                4.761905      23.2880    8.4
## 5 30.2085     Ewallet 604.17                4.761905      30.2085    5.3
## 6 29.8865     Ewallet 597.73                4.761905      29.8865    4.1
##      Total Day Month Year Hour Minute
## 1 548.9715  05    01 2019   13     08
## 2  80.2200  08    03 2019   10     29
## 3 340.5255  03    03 2019   13     23
## 4 489.0480  27    01 2019   20     33
## 5 634.3785  08    02 2019   10     37
## 6 627.6165  25    03 2019   18     30
```

Previewing the unique entries in the non-numeric columns

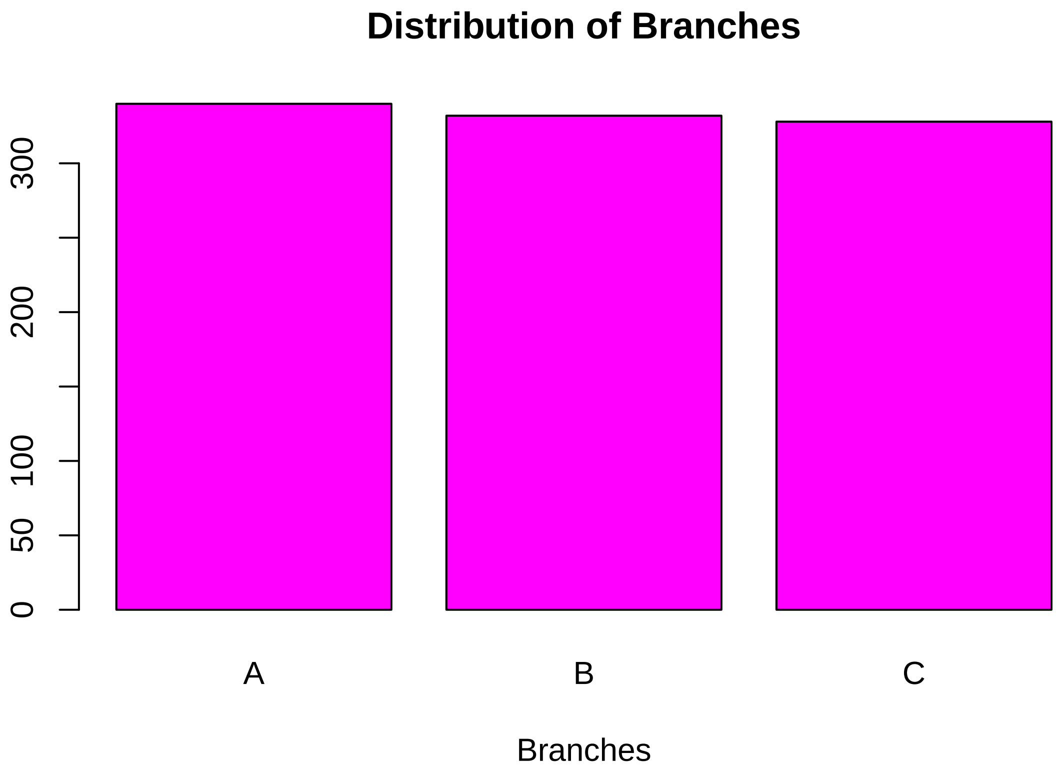
```
#Branch column
unique(supermarket$Branch)
```

```
## [1] "A" "C" "B"
```

```
branch <- supermarket$Branch
branch_freq <- table(branch)
branch_freq
```

```
## branch
##   A   B   C
## 340 332 328
```

```r
#visualize using frequency tables
branch <- supermarket$Branch
branch_freq <- table(branch)
barplot(branch_freq,main="Distribution of Branches", xlab="Branches",col="Magenta")
```



There are three branches, A, B and C. Branch A (340) has a slight difference from B and C which are almost equal.

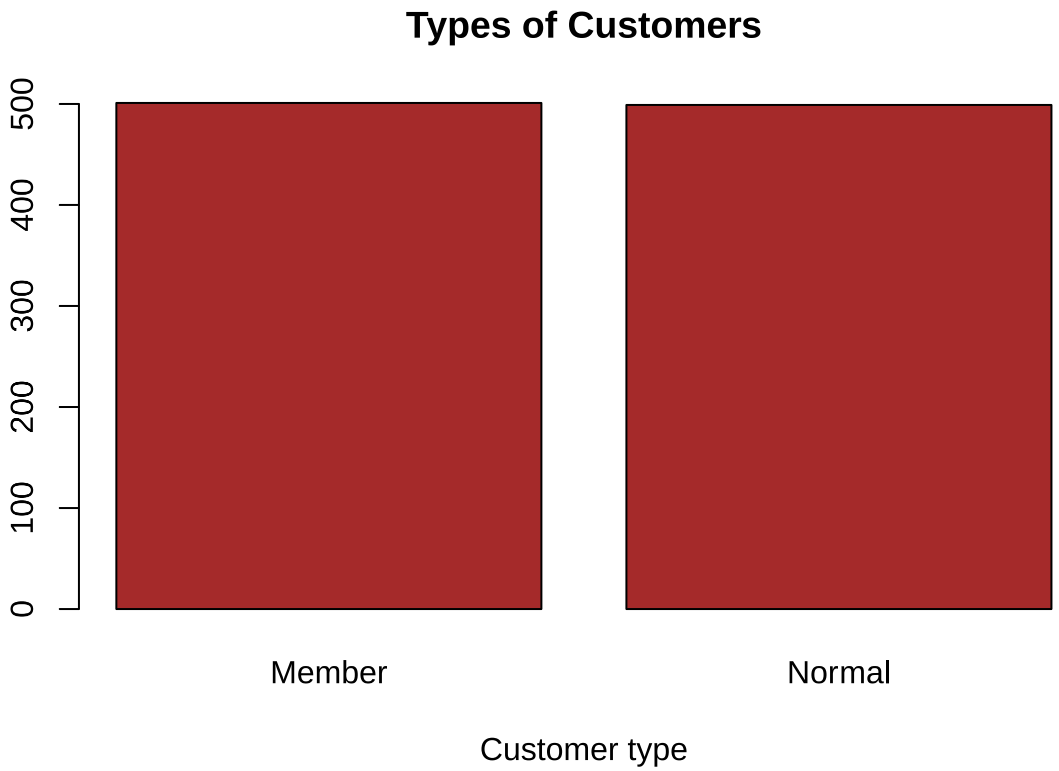
```r
# Customer.type column
unique(supermarket$Customer.type)
```

```
## [1] "Member" "Normal"
```

```r
customer<- supermarket$Customer.type
customer_freq <- table(customer)
customer_freq
```

```
## customer
## Member Normal
##    501    499
```

```r
#visualize using frequency tables
customer<- supermarket$Customer.type
customer_freq <- table(customer)
barplot(customer_freq,main="Types of Customers", xlab="Customer type",col="brown")
```



There are two types of customers; Member and Normal customer Both customer types have a slightly similar distribution.

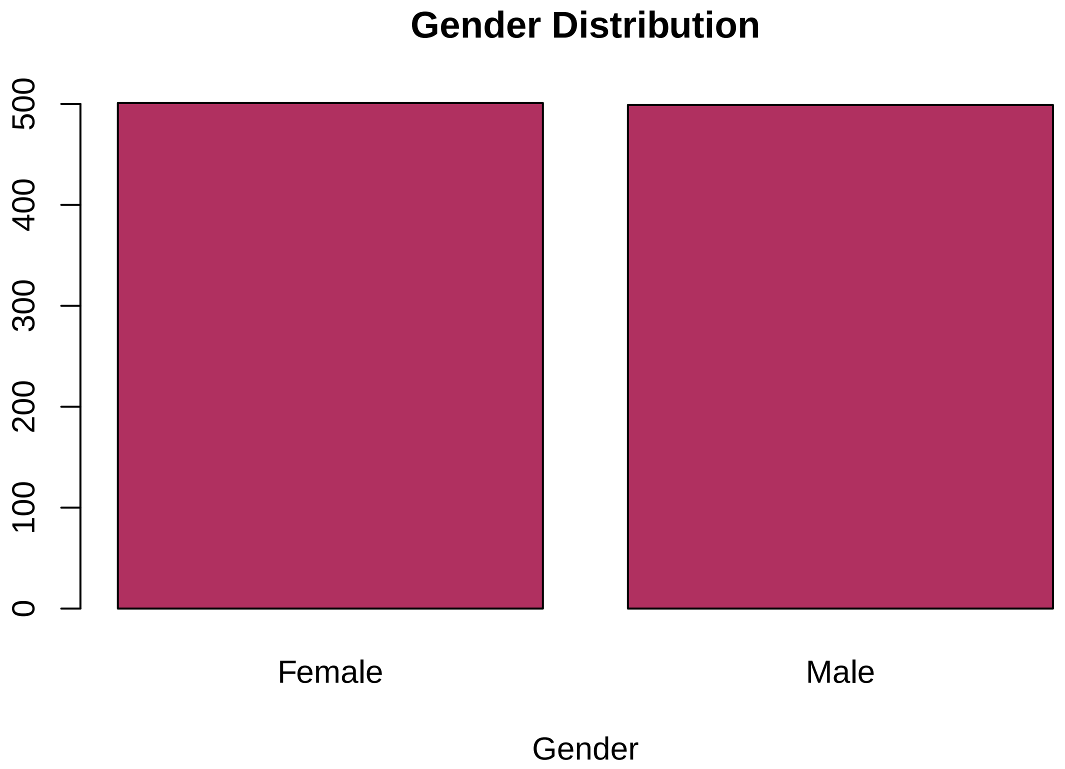
```
#Gender column
unique(supermarket$Gender)
```

```
## [1] "Female" "Male"
```

```
gender<- supermarket$Gender
gender_freq <- table(gender)
gender_freq
```

```
## gender
## Female   Male
##    501    499
```

```
#visualize using frequency tables
gender<- supermarket$Gender
gender_freq <- table(gender)
barplot(gender_freq,main="Gender Distribution", xlab="Gender",col="maroon")
```



The gender type are male and female and both have a slightly equal distribution.

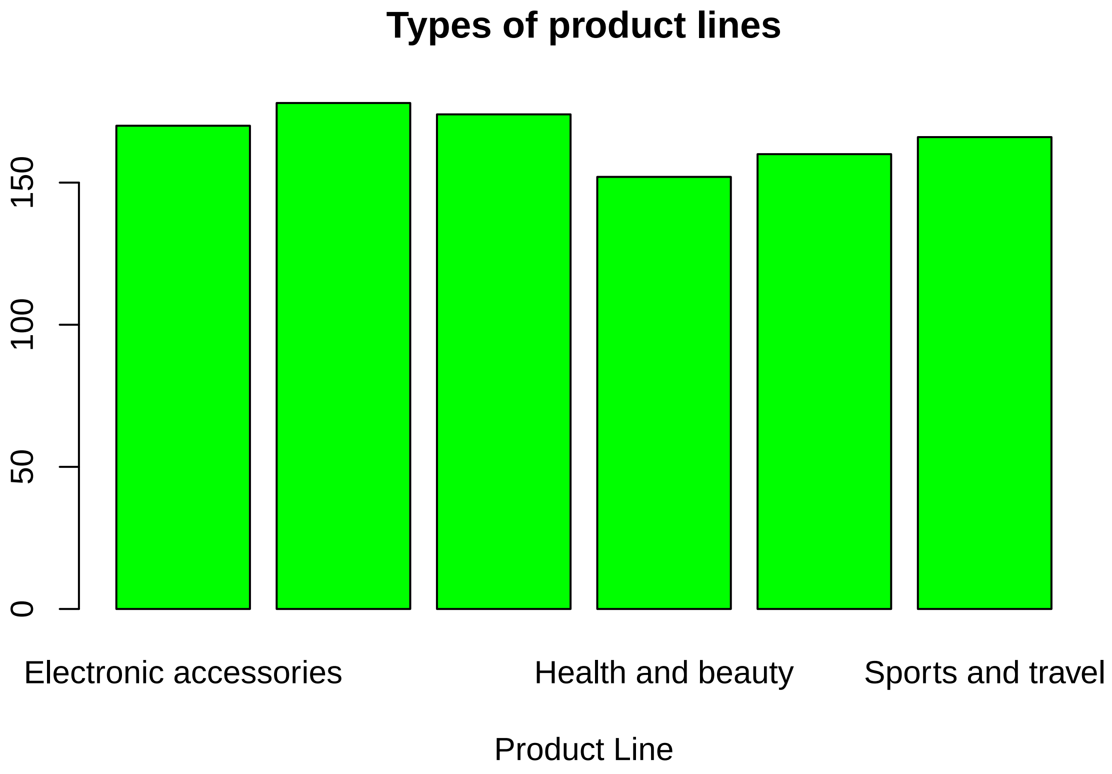
```
#product.line column
unique(supermarket$Product.line)
```

```
## [1] "Health and beauty"      "Electronic accessories" "Home and lifestyle"
## [4] "Sports and travel"      "Food and beverages"     "Fashion accessories"
```

```
product_line <- supermarket$Product.line
productline_freq <- table(product_line)
productline_freq
```

```
## product_line
## Electronic accessories    Fashion accessories     Food and beverages
##                    170                    178                    174
##      Health and beauty     Home and lifestyle      Sports and travel
##                    152                    160                    166
```

```
#visualize using frequency tables
product<- supermarket$Product.line
product_freq <- table(product)
barplot(product_freq,main="Types of product lines", xlab="Product Line",col="green")
```



There are six product lines; Health and beauty, Electronic accessories, Home and lifestyle, Sports and travel, Food and beverages, and Fashion accessories. The product line with more customers is the Fashion accessories line with 178, followed by Food and beverages with 174, Electronic accessories with 170, then sports and travel(166), home and lifestyle(160) and the least is Health and beauty with 152.

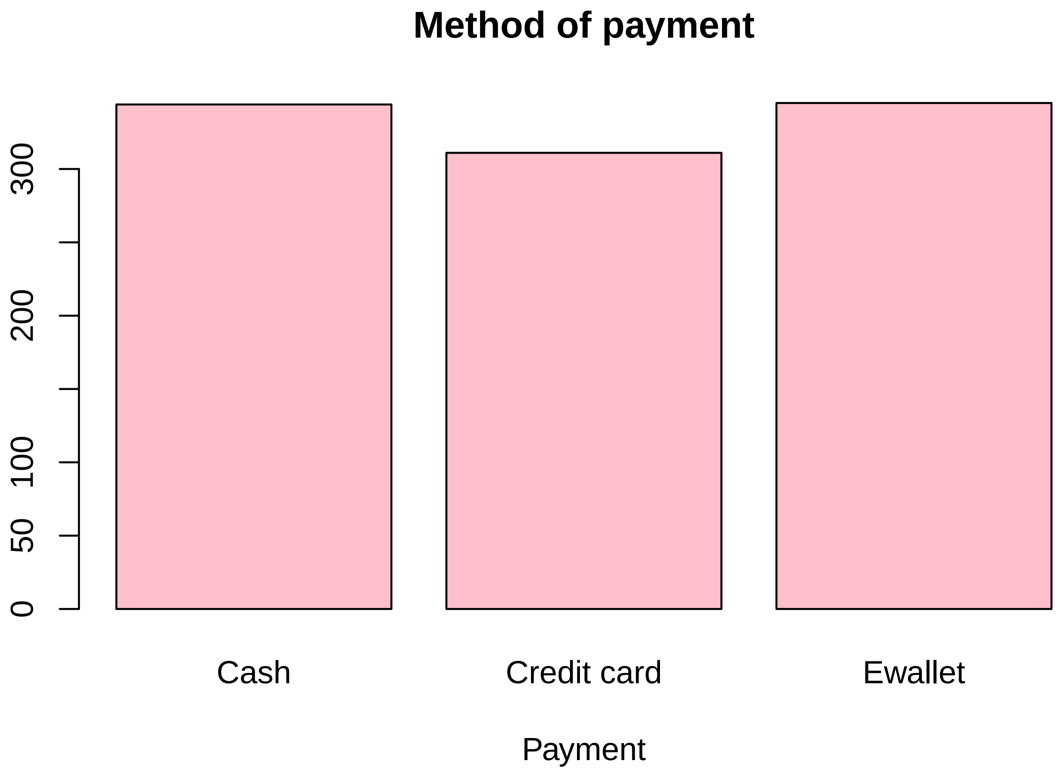
```
#payment column
unique(supermarket$Payment)
```

```
## [1] "Ewallet"     "Cash"        "Credit card"
```

```
payment <- supermarket$Payment
payment_freq <- table(payment)
payment_freq
```

```
## payment
##        Cash Credit card     Ewallet 
##         344         311         345
```

```
#visualize using frequency tables
payment <- supermarket$Payment
payment_freq <- table(payment)
barplot(payment_freq,main="Method of payment", xlab="Payment",col="pink")
```



There are three methods of payment; cash, credit card and Ewallet. Cash and Ewallet have almost similar distribution while credit card is the least used method.

Label Encoding

We are going to encode the categorical columns to integers. This is because integers cannot take decimals.

```
# Branch column
supermarket$Branch<-as.integer(supermarket$Branch)
```

```
## Warning: NAs introduced by coercion
```

```
# Customer Type column
supermarket$Customer.type<-as.integer(supermarket$Customer.type)
```

```
## Warning: NAs introduced by coercion
```

```
# Gender column
supermarket$Gender<-as.integer(supermarket$Gender)
```

```
## Warning: NAs introduced by coercion
```

```
# Product.line column
supermarket$Product.line<-as.integer(supermarket$Product.line)
```

```
## Warning: NAs introduced by coercion
```

```
#Payment column
supermarket$Payment<-as.integer(supermarket$Payment)
```

```
## Warning: NAs introduced by coercion
```

DIMENSIONALITY REDUCTION

Using Principal Comment Analysis

PCA is the unsupervised learning technique where data is converted from a g=high dimension to lower or equal number of dimensions. We shall apply PCA to numerical data.

```
#we first install the packages we shall need
#install.packages("Rtsne")
#library(Rtsne)

library(lattice)

#install.packages("dplyr")
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
#install packages("tidyverse")
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v stringr 1.4.0
## v tidyr   1.2.0     v forcats 0.5.1
## v readr   2.1.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
#install.packages("ggplot2")
library(ggplot2)

#install.packages("devtools", dependencies=TRUE)
library(devtools)
```

```
## Loading required package: usethis
```

```
#install_github("vqv/ggbiplot", force = TRUE) For plotting PCA
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 object is masked from 'package:purrr':
##
##     compact

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

## 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
```

```
#Extract numerical and integer columns only
super <- select_if(supermarket,is.numeric)
str(super)
```

```
## ’data.frame’:    1000 obs. of  13 variables:
##  $ Branch                 : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ Customer.type          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ Gender                 : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ Product.line           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 ...
##  $ Payment                : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 ...
```

```
#replace the missing values;
super[is.na(super)]=0
head(super)
```

```
##   Branch Customer.type Gender Product.line Unit.price Quantity     Tax Payment
## 1      0             0      0            0      74.69        7 26.1415       0
## 2      0             0      0            0      15.28        5  3.8200       0
## 3      0             0      0            0      46.33        7 16.2155       0
## 4      0             0      0            0      58.22        8 23.2880       0
## 5      0             0      0            0      86.31        7 30.2085       0
## 6      0             0      0            0      85.39        7 29.8865       0
##     cogs gross.margin.percentage gross.income Rating    Total
## 1 522.83                4.761905      26.1415    9.1 548.9715
## 2  76.40                4.761905       3.8200    9.6  80.2200
## 3 324.31                4.761905      16.2155    7.4 340.5255
## 4 465.76                4.761905      23.2880    8.4 489.0480
## 5 604.17                4.761905      30.2085    5.3 634.3785
## 6 597.73                4.761905      29.8865    4.1 627.6165
```

```
# We remove the constant/zero column to avoid an error which comes about by including them
names(super[, sapply(super, function(v) var(v, na.rm=TRUE)==0)])
```

```
## [1] "Branch"                  "Customer.type"
## [3] "Gender"                  "Product.line"
## [5] "Payment"                 "gross.margin.percentage"
```

```
# Then we drop the columns as they result to a ("cannot rescale a constant/zero column to unit variance
super <- subset(super, select = -c(gross.margin.percentage, Branch, Customer.type, Gender, Product.line
```

```
# We can now conduct our PCA with center and scale set to true
super.pca <- prcomp(super, center = TRUE, scale. = TRUE)

# previewing our PCA summary
summary(super.pca)
```

```
## Importance of components:
```

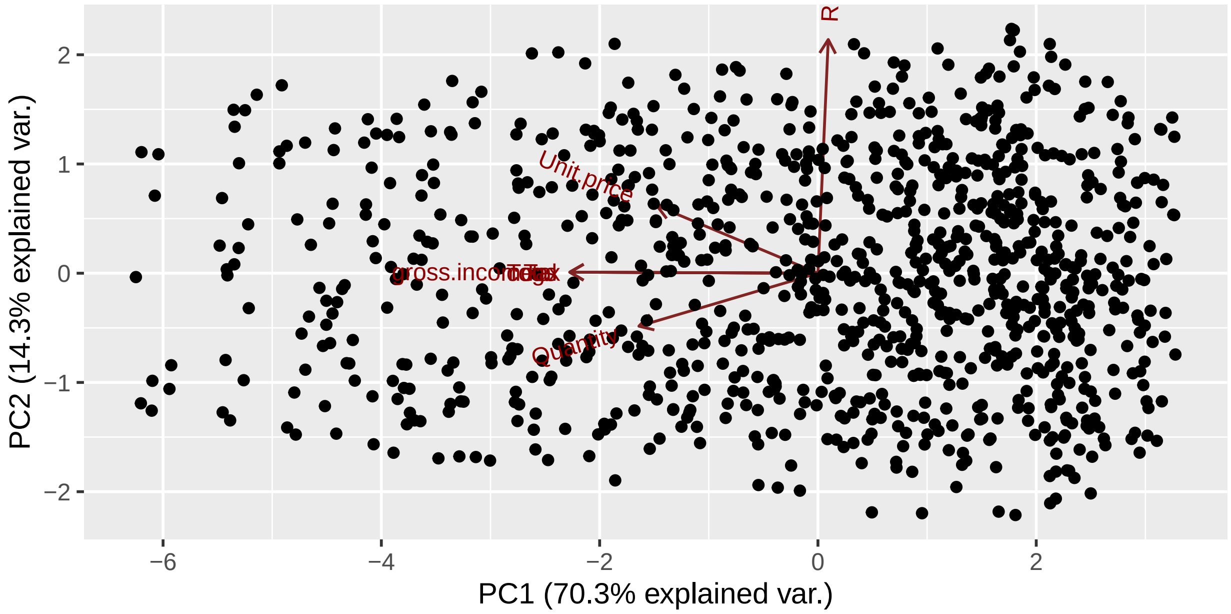
```
##                           PC1    PC2    PC3     PC4       PC5       PC6
## Standard deviation     2.2185 1.0002 0.9939 0.30001 2.981e-16 1.493e-16
## Proportion of Variance 0.7031 0.1429 0.1411 0.01286 0.000e+00 0.000e+00
## Cumulative Proportion  0.7031 0.8460 0.9871 1.00000 1.000e+00 1.000e+00
##                              PC7
## Standard deviation     9.831e-17
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```

From the Output, we have a total of 7 components after dropping 6 with missing values out of the 13 components. PC1 gives the total variance of 70.31% while PC2 gives 14.29%. We can see that the variance percentage decreases across the PCA table.

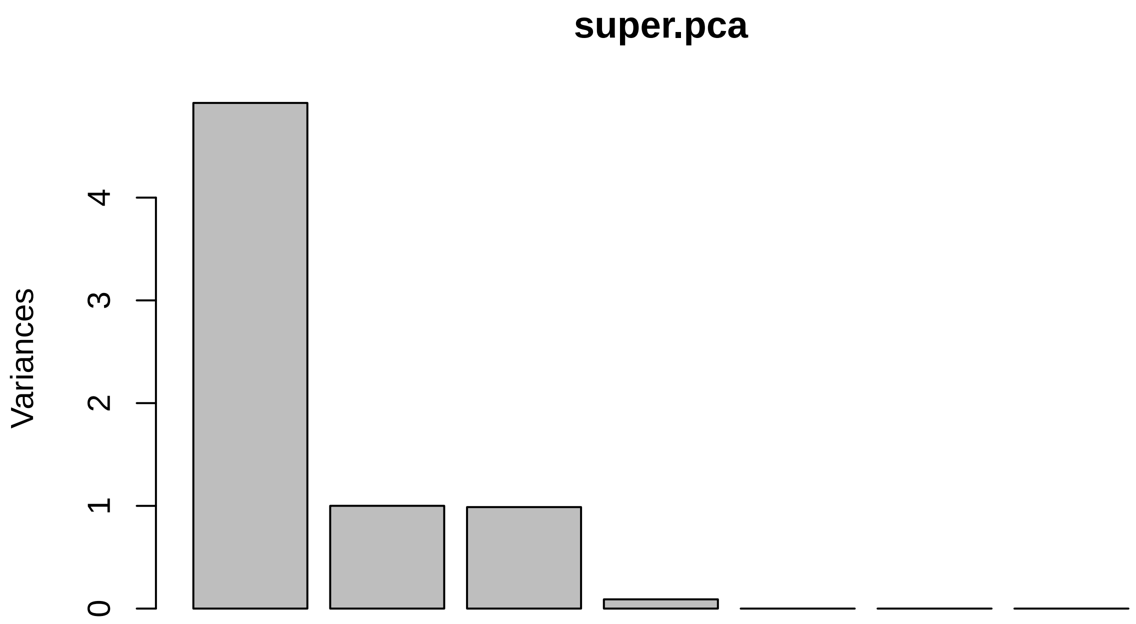
```
#preview the PCA structure using the str()function
str(super.pca)
```

```
## List of 5
##  $ sdev    : num [1:7] 2.22 1.00 9.94e-01 3.00e-01 2.98e-16 ...
##  $ rotation: num [1:7, 1:7] -0.292 -0.325 -0.45 -0.45 -0.45 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:7] "Unit.price" "Quantity" "Tax" "cogs" ...
##   .. ..$ : chr [1:7] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : Named num [1:7] 55.67 5.51 15.38 307.59 15.38 ...
##   ..- attr(*, "names")= chr [1:7] "Unit.price" "Quantity" "Tax" "cogs" ...
##  $ scale   : Named num [1:7] 26.49 2.92 11.71 234.18 11.71 ...
##   ..- attr(*, "names")= chr [1:7] "Unit.price" "Quantity" "Tax" "cogs" ...
##  $ x       : num [1:1000, 1:7] -2.005 2.306 -0.186 -1.504 -2.8 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:1000] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:7] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"
```

```
#we then plot the ggbiplot to visualize the PCA results
ggbiplot(super.pca, labels=rownames(super.pca),ellipse = TRUE,obs.scale=1,var.scale=1)
```



```
plot(super.pca, type="b")
```



PART TWO: FEATURE SELECTION

Feature Selection is the method of reducing the input variable to your model by using only relevant data and getting rid of noise in data.

It is the process of automatically choosing relevant features for your machine learning model based on the type of problem you are trying to solve. We do this by including or excluding important features without changing them. It helps in cutting down the noise in our data and reducing the size of our input data.

Using the Filter Method

```
#we start by installing the necessary packages
#install.packages("caret")
library(caret)
```

```
##
## Attaching package: 'caret'

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

```
#install.packages("corrplot")
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
#Checking the correlations
corr <- cor(super)
corr
```

```
##                Unit.price    Quantity        Tax       cogs gross.income
## Unit.price    1.000000000  0.01077756  0.6339621  0.6339621    0.6339621
## Quantity      0.010777564  1.00000000  0.7055102  0.7055102    0.7055102
## Tax           0.633962089  0.70551019  1.0000000  1.0000000    1.0000000
## cogs          0.633962089  0.70551019  1.0000000  1.0000000    1.0000000
## gross.income  0.633962089  0.70551019  1.0000000  1.0000000    1.0000000
## Rating       -0.008777507 -0.01581490 -0.0364417 -0.0364417   -0.0364417
## Total         0.633962089  0.70551019  1.0000000  1.0000000    1.0000000
##                    Rating      Total
## Unit.price   -0.008777507  0.6339621
## Quantity     -0.015814905  0.7055102
## Tax          -0.036441705  1.0000000
## cogs         -0.036441705  1.0000000
## gross.income -0.036441705  1.0000000
## Rating        1.000000000 -0.0364417
## Total        -0.036441705  1.0000000
```

```
#getting the highly correlated variables

highcorr <- findCorrelation(corr, cutoff=0.75)
names(super[,highcorr])
```

```
## [1] "cogs"  "Total" "Tax"
```

Cost of goods sold(cogs), total and tax variables are highly correlated. Hence we remove them

```
#removing the highlycorrelated variables, to remove redundancy
filter_super <- super[-highcorr]
head(filter_super)
```

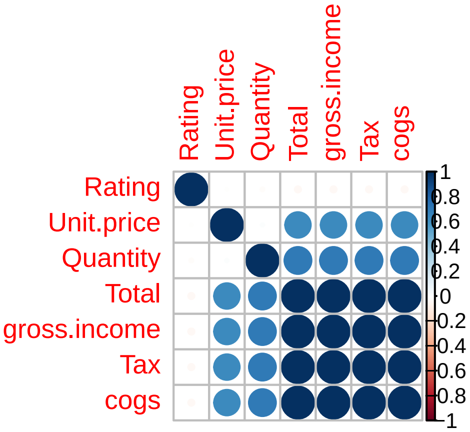
```
##   Unit.price Quantity gross.income Rating
## 1      74.69        7      26.1415    9.1
## 2      15.28        5       3.8200    9.6
## 3      46.33        7      16.2155    7.4
## 4      58.22        8      23.2880    8.4
## 5      86.31        7      30.2085    5.3
## 6      85.39        7      29.8865    4.1
```

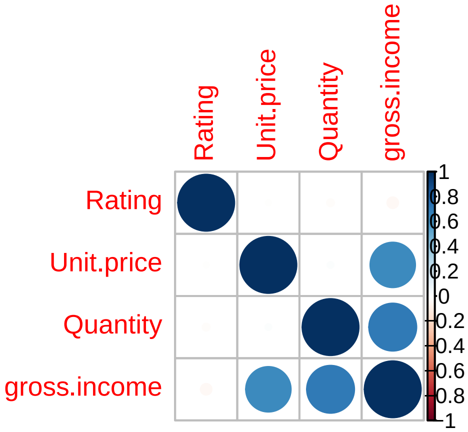
```
#graphical visualization to compare the correlation

par(mfrow=c(2,2))


corrplot(corr,order="hclust")#original dataset

corrplot(cor(filter_super),order="hclust")#filtered dataset
```


Rating
Unit.price
Quantity
Total
gross.income
Tax
cogs
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1


Rating
Unit.price
Quantity
gross.income
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1