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). In this part we are going to use Dimensionality Reduction and Feature selection to gain insights from the dataset that will help the marketing department.

Reading the dataset

```
#Importing the library
#Reading the data
#Previewing the first six entries of the data.
library("data.table")
data = fread('http://bit.ly/CarreFourDataset')
head(data)
```

```
#Previewing the last six entries of the dataset
tail(data)
```

Checking the dataset

```
#Checking the dimensions of the dataset.
dim(data)
```

```
#Checking the structure of the data
str(data)
```

```
Classes ‘data.table’ and '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
 $ 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 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

```
#Removing white spaces from the column names
names(data)<-make.names(names(data),unique = TRUE)


#Assigning the correct data types to columns with wrong data types.
data$Branch <- as.factor(data$Branch)
data$Customer.type <- as.factor(data$Customer.type)
data$Gender <- as.factor(data$Gender)
data$Product.line <- as.factor(data$Product.line)
data$Payment <- as.factor(data$Payment)



# splitting the Date column into Day, Month, and Year columns, and storing the results as fac
data$Day <- as.factor(format(as.POSIXct(data$Date, format="%m/%d/%Y"), "%d"))
data$Month <- as.factor(format(as.POSIXct(data$Date, format="%m/%d/%Y"), "%m"))
data$Year <- as.factor(format(as.POSIXct(data$Date, format="%m/%d/%Y"), "%Y"))

# splitting the Time variable into Hour and Minute, and storing the results as factors
data$Hour <- as.factor(format(as.POSIXct(data$Time, format="%H:%M"), "%H"))
data$Minute <- as.factor(format(as.POSIXct(data$Time, format="%H:%M"), "%M"))


#Checking the column names.
names(data)




#Confirming if the variables have been assigned the correct data type
str(data)
```

```
Classes ‘data.table’ and 'data.frame':  1000 obs. of  21 variables:
 $ Invoice.ID            : chr  "750-67-8428" "226-31-3081" "631-41-3108" "123-19-1176
 $ Branch                : Factor w/ 3 levels "A","B","C": 1 3 1 1 1 3 1 3 1 2 ...
 $ Customer.type         : Factor w/ 2 levels "Member","Normal": 1 2 2 1 2 2 1 2 1 1 .
 $ Gender                : Factor w/ 2 levels "Female","Male": 1 1 2 2 2 2 1 1 1 1 ...
 $ Product.line          : Factor w/ 6 levels "Electronic accessories",..: 4 1 5 4 6 1
 $ 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               : Factor w/ 3 levels "Cash","Credit card",..: 3 1 2 3 3 3 3 3
 $ 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 ...
 $ Day                   : Factor w/ 31 levels "01","02","03",..: 5 8 3 27 8 25 25 24
 $ Month                 : Factor w/ 3 levels "01","02","03": 1 3 3 1 2 3 2 2 1 2 ...
```

```
 $ Year                    : Factor w/ 1 level "2019": 1 1 1 1 1 1 1 1 1 1 ...
 $ Hour                    : Factor w/ 11 levels "10","11","12",..: 4 1 4 11 1 9 5 2 8 4
 $ Minute                  : Factor w/ 60 levels "00","01","02",..: 9 30 24 34 38 31 37
 - attr(*, ".internal.selfref")=<externalptr>
```

```
#Previewing the dataset
head(data)
```

```
#Checking the number of columns in the dataset
ncol(data)
```

```
#Checking the number of rows in the dataset
nrow(data)
```

Data Cleaning

Data Completeness

```
#Finding the total missing values in each column
```

```
colSums(is.na(data))
```

There are no missing values in the dataset.

Data Consistency

```
#Checking for duplicates
anyDuplicated(data)
```

There are no duplicates in the dataset.

Data Validity

```
#Dropping unnecessary columns
data$Time <- NULL
data$Date <- NULL
data$Invoice.ID <- NULL
colnames(data)
```

Dimensionality Reduction

Principal Component Analysis(PCA)

```
#Changing all variables to numeric
data_num <- data
data_num$Branch <- as.numeric(data_num$Branch)
data_num$Customer.type <- as.numeric(data_num$Customer.type)
data_num$Gender <- as.numeric(data_num$Gender)
data_num$Product.line <- as.numeric(data_num$Product.line)
data_num$Payment <- as.numeric(data_num$Payment)
data_num$Day <- as.numeric(data_num$Day)
data_num$Month <- as.numeric(data_num$Month)
data_num$Year <- as.numeric(data_num$Year)
data_num$Hour <- as.numeric(data_num$Hour)
```

```
data_num$Minute <- as.numeric(data_num$Minute)
data_num$Quantity <- as.numeric(data_num$Quantity)
```

```
#Confirming if the variables have been changed to numeric.
str(data_num)
```

```
'data.frame':	1000 obs. of  18 variables:
 $ Branch                 : num  1 3 1 1 1 3 1 3 1 2 ...
 $ Customer.type          : num  1 2 2 1 2 2 1 2 1 1 ...
 $ Gender                 : num  1 1 2 2 2 2 1 1 1 1 ...
 $ Product.line           : num  4 1 5 4 6 1 1 5 4 3 ...
 $ Unit.price             : num  74.7 15.3 46.3 58.2 86.3 ...
 $ Quantity               : num  7 5 7 8 7 7 6 10 2 3 ...
 $ Tax                    : num  26.14 3.82 16.22 23.29 30.21 ...
 $ Payment                : num  3 1 2 3 3 3 3 3 2 2 ...
 $ 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 ...
 $ Day                    : num  5 8 3 27 8 25 25 24 10 20 ...
 $ Month                  : num  1 3 3 1 2 3 2 2 1 2 ...
 $ Year                   : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Hour                   : num  4 1 4 11 1 9 5 2 8 4 ...
 $ Minute                 : num  9 30 24 34 38 31 37 39 16 28 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

```
#Previewing the dataset
head(data_num)
```

```
#Checking the standard deviation of different variables.
sd(data_num$Branch)
sd(data_num$Customer.type)
```

```
sd(data_num$Gender)
sd(data_num$Product.line)
sd(data_num$Payment)
sd(data_num$Unit.price)
sd(data_num$Quantity)
sd(data_num$Tax)
sd(data_num$Total)
sd(data_num$Day)
sd(data_num$Month)
sd(data_num$Minute)
sd(data_num$Year)
sd(data_num$Hour)
sd(data_num$cogs)
sd(data_num$gross.income)
sd(data_num$gross.margin.percentage)
sd(data_num$Rating)
```

```
#Dropping variables which have a standard deviation of 0
data_num$gross.margin.percentage <- NULL
data_num$Year <- NULL
```

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

PC1 explains 30.81% of the total variance and PC2 explains 7.39% of the total variance.

```
# Calling str() to have a look at your PCA object
str(data.pca)
```

```
List of 5
 $ sdev    : num [1:16] 2.22 1.09 1.08 1.05 1.02 ...
 $ rotation: num [1:16, 1:16] 0.0224 -0.0125 -0.0283 0.0174 0.2911 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:16] "Branch" "Customer.type" "Gender" "Product.line" ...
  .. ..$ : chr [1:16] "PC1" "PC2" "PC3" "PC4" ...
 $ center  : Named num [1:16] 1.99 1.5 1.5 3.45 55.67 ...
  ..- attr(*, "names")= chr [1:16] "Branch" "Customer.type" "Gender" "Product.line" ...
 $ scale   : Named num [1:16] 0.818 0.5 0.5 1.715 26.495 ...
  ..- attr(*, "names")= chr [1:16] "Branch" "Customer.type" "Gender" "Product.line" ...
 $ x       : num [1:1000, 1:16] 2.05 -2.287 0.126 1.466 2.743 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:16] "PC1" "PC2" "PC3" "PC4" ...
 - attr(*, "class")= chr "prcomp"
```

```
# Installing our ggbiplot visualisation package
#
library(devtools)
install_github("vqv/ggbiplot")
# Then Loading our ggbiplot library
#
library(ggbiplot)
options(repr.plot.width = 10, repr.plot.height = 10)
ggbiplot(data.pca)
```

We can observe that unit price, total income and quantity contribute to PCA1.

```
# Adding more detail to the plot, we provide arguments rownames as labels
#
options(repr.plot.width = 20, repr.plot.height = 20)
ggbiplot(data.pca, labels=rownames(data), obs.scale = 1, var.scale = 1)
```

t-Distributed Stochastic Neighbor Embedding (t-SNE)

```
# Installing Rtnse package
#
install.packages("Rtsne")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
# Loading our tnse library
#
library(Rtsne)


# Curating the database for analysis
# Using Branch as the factor.
data_branch <- data_num
data_branch$Branch<-as.factor(data_branch$Branch)

# For plotting
#
colors = rainbow(length(unique(data_branch$Branch)))
names(colors) = unique(data_branch$Branch)


# Executing the algorithm on curated data
tsne <- Rtsne(data_branch[,-1], dims = 2, perplexity=30, verbose=TRUE, max_iter = 500)
# Getting the duration of execution
exeTimeTsne <- system.time(Rtsne(data_branch[,-1], dims = 2, perplexity=30, verbose=TRUE, max
```

```
Performing PCA
Read the 1000 x 15 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 0.09 seconds (sparsity = 0.101692)!
Learning embedding...
Iteration 50: error is 62.372482 (50 iterations in 0.14 seconds)
Iteration 100: error is 54.432705 (50 iterations in 0.12 seconds)
Iteration 150: error is 52.730028 (50 iterations in 0.12 seconds)
Iteration 200: error is 52.196947 (50 iterations in 0.12 seconds)
Iteration 250: error is 51.971683 (50 iterations in 0.12 seconds)
Iteration 300: error is 0.721679 (50 iterations in 0.13 seconds)
Iteration 350: error is 0.551896 (50 iterations in 0.13 seconds)
Iteration 400: error is 0.510869 (50 iterations in 0.12 seconds)
Iteration 450: error is 0.495838 (50 iterations in 0.12 seconds)
Iteration 500: error is 0.482765 (50 iterations in 0.12 seconds)
Fitting performed in 1.24 seconds.
Performing PCA
Read the 1000 x 15 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 0.09 seconds (sparsity = 0.101692)!
Learning embedding...
Iteration 50: error is 59.951980 (50 iterations in 0.15 seconds)
Iteration 100: error is 53.315151 (50 iterations in 0.12 seconds)
Iteration 150: error is 52.252961 (50 iterations in 0.12 seconds)
Iteration 200: error is 51.977148 (50 iterations in 0.12 seconds)
Iteration 250: error is 51.925500 (50 iterations in 0.12 seconds)
```

```
Iteration 300: error is 0.721104 (50 iterations in 0.12 seconds)
Iteration 350: error is 0.548494 (50 iterations in 0.12 seconds)
Iteration 400: error is 0.505661 (50 iterations in 0.12 seconds)
Iteration 450: error is 0.490794 (50 iterations in 0.12 seconds)
Iteration 500: error is 0.480143 (50 iterations in 0.12 seconds)
Fitting performed in 1.24 seconds.
```

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

```
# Curating the database for analysis
# Using Customer.type as the factor.
data_customer <- data_num
data_customer$Customer.type<-as.factor(data_customer$Customer.type)

# For plotting
#
colors = rainbow(length(unique(data_customer$Customer.type)))
names(colors) = unique(data_customer$Customer.type)

# Executing the algorithm on curated data
tsne <- Rtsne(data_customer[,-1], dims = 2, perplexity=30, verbose=TRUE, max_iter = 500)
# Getting the duration of execution
exeTimeTsne <- system.time(Rtsne(data_customer[,-1], dims = 2, perplexity=30, verbose=TRUE, m
# Plotting our graph and closely examining the graph
#
plot(tsne$Y, t='n', main="tsne")
text(tsne$Y, labels=data_customer$Customer.type, col=colors[data_customer$Customer.type])
```

```
# Curating the database for analysis
# Using Gender as the factor.
data_gender <- data_num
data_gender$Gender<-as.factor(data_gender$Gender)

# For plotting
#
colors = rainbow(length(unique(data_gender$Gender)))
names(colors) = unique(data_gender$Gender)

# Executing the algorithm on curated data
tsne <- Rtsne(data_gender[,-1], dims = 2, perplexity=30, verbose=TRUE, max_iter = 500)
# Getting the duration of execution
exeTimeTsne <- system.time(Rtsne(data_gender[,-1], dims = 2, perplexity=30, verbose=TRUE, ma
# Plotting our graph and closely examining the graph
#
```

```
    plot(tsne$Y, t='n', main="tsne")
    text(tsne$Y, labels=data_gender$Gender, col=colors[data_gender$Gender])
```

```
# Curating the database for analysis
# Using Product.line as the factor.
data_product <- data_num
data_product$Product.line<-as.factor(data_product$Product.line)

# For plotting
#
colors = rainbow(length(unique(data_product$Product.line)))
names(colors) = unique(data_product$Product.line)

# Executing the algorithm on curated data
tsne <- Rtsne(data_product[,-1], dims = 2, perplexity=30, verbose=TRUE, max_iter = 500)
# Getting the duration of execution
exeTimeTsne <- system.time(Rtsne(data_product[,-1], dims = 2, perplexity=30, verbose=TRUE, ma
# Plotting our graph and closely examining the graph
#
plot(tsne$Y, t='n', main="tsne")
text(tsne$Y, labels=data_product$Product.line, col=colors[data_product$Product.line])
```

```
# Curating the database for analysis
# Using Payment as the factor.
data_payment <- data_num
data_payment$Payment<-as.factor(data_payment$Payment)

# For plotting
#
colors = rainbow(length(unique(data_payment$Payment)))
names(colors) = unique(data_payment$Payment)

# Executing the algorithm on curated data
tsne <- Rtsne(data_payment[,-1], dims = 2, perplexity=30, verbose=TRUE, max_iter = 500)
# Getting the duration of execution
exeTimeTsne <- system.time(Rtsne(data_payment[,-1], dims = 2, perplexity=30, verbose=TRUE, ma
# Plotting our graph and closely examining the graph
#
plot(tsne$Y, t='n', main="tsne")
text(tsne$Y, labels=data_payment$Payment, col=colors[data_payment$Payment])
```

Feature Selection

Filter Method

```
# Installing and loading our caret package
# ---
#
suppressWarnings(
        suppressMessages(if
                         (!require(caret, quietly=TRUE))
                 install.packages("caret")))
library(caret)


# Installing and loading the corrplot package for plotting
# ---
#
suppressWarnings(
        suppressMessages(if
                         (!require(corrplot, quietly=TRUE))
                 install.packages("corrplot")))
library(corrplot)


# Calculating the correlation matrix
correlationMatrix <- cor(data_num)
correlationMatrix
```

```
# Find attributes that are highly correlated
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.75)
highlyCorrelated
```

Variables number 7, 9 and 12 are highly correlated.

```
#Checking the names of the highly correlated variables.
names(data_num[,..highlyCorrelated])
```

```
# Removing Redundant Features
data_filtered<-data_num[,-..highlyCorrelated]
head(data_filtered)
```

Dropping variables cogs, total and tax since they are highly correlated.

```
# Performing our graphical comparison
# ---
#
par(mfrow = c(1, 2))
```

```
corrplot(correlationMatrix, order = "hclust")
corrplot(cor(data_filtered), order = "hclust")
```

Wrapper Method

```
# Installing and loading our clustvarsel package
# ---
#
suppressWarnings(
        suppressMessages(if
                         (!require(clustvarsel, quietly=TRUE))
                 install.packages("clustvarsel")))

library(clustvarsel)



# Installing and loading our mclust package
# ---
#
suppressWarnings(
        suppressMessages(if
                         (!require(mclust, quietly=TRUE))
                 install.packages("mclust")))
library(mclust)


#Previewing the data
head(data)
```

```
#Dropping columns that are not numeric
data_num1 <- data
data_num1$Branch <- NULL
data_num1$Customer.type <- NULL
data_num1$Gender <- NULL
data_num1$Product.line <- NULL
data_num1$Payment <- NULL
data_num1$Day <- NULL
data_num1$Month <- NULL
data_num1$Minute <- NULL
data_num1$Year <- NULL
data_num1$Hour <- NULL
data_num1$gross.margin.percentage <- NULL
head(data_num1)
```

```
#Checking the structure of the data
str(data_num1)
```

```
Classes ‘data.table’ and 'data.frame':  1000 obs. of  7 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.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 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

```
#Scaling the data
data_sc <- scale(data_num1)
```

```
# Sequential forward greedy search (default)
# ---
```

```
#
out = clustvarsel(data_sc, G = 1:9)
out
```

Quality and Total have been accepted while the other variables have been rejected.

```
out$subset
```

```
#Building a clustering model
Subset1 = data_sc[, out$subset]
mod = Mclust(Subset1, G = 1:9)
summary(mod)
```

```
# plotting
options(repr.plot.width = 10, repr.plot.height = 10)
plot(mod ,c("classification"))
```

Embedded Method

```
# We install and load our wskm package
# ---
#
suppressWarnings(
        suppressMessages(if
                         (!require(wskm, quietly=TRUE))
                 install.packages("wskm")))
library(wskm)

set.seed(2)
model <- ewkm(data_num[1:16], 3, lambda=2, maxiter=1000)


# Loading and installing our cluster package
suppressWarnings(
        suppressMessages(if
                         (!require(cluster, quietly=TRUE))
                 install.packages("cluster")))
library("cluster")
```

```
# Cluster Plot against 1st 2 principal components
# ---
#
clusplot(data_num[1:16], model$cluster, color=TRUE, shade=TRUE,
         labels=2, lines=1,main='Cluster Analysis for The Carefour dataset.')
```

These two components explain 55.25% of the point variability.

```
#Checking the weights stored in the model.
round(model$weights*100,2)
```

Feature Ranking

```
# We install and load the required packages
# ---
#
suppressWarnings(
        suppressMessages(if
                         (!require(FSelector, quietly=TRUE))
                 install.packages("FSelector")))
library(FSelector)


# From the FSelector package, we use the correlation coefficient as a unit of valuation.
Scores <- linear.correlation(data_num)
Scores
```

```
# cutoff.k: The algorithms select a subset from a ranked attributes.
# ---
#
Subset <- cutoff.k(Scores, 5)
as.data.frame(Subset)
```

```
Subset2 <-cutoff.k.percent(Scores, 0.4)
as.data.frame(Subset2)
```

```
Scores2 <- information.gain(data_num)

# Choosing Variables by cutoffSubset <- cutoff.k(Scores2, 5)
# ---
#
Subset3 <- cutoff.k(Scores2, 5)
as.data.frame(Subset3)
```

Using feature ranking we find out that the variables that rank first(based on the correlation coefficient) are Customer.type, Gender and Product.line.

Conclusion and Recommendation

From the above analysis we find out that Tax, cogs and gross.income are highly correlated variables and they may not give insights to the marketing department and hence the department shouild consider avoiding those variables when preparing data to be used for marketing purposes.