Feature Selection

Brian Onchweri

2022-06-10

We will have Four sections for this study as illustrated below:

Part 1: Feature Selection and Dimensionality Reduction

a) Feature Selection

This section requires us to perform feature selection through the use of the unsupervisedlearning methods. We will be required to perform our analysis and provide insights on the features that contribute the most information to the dataset.

b) Dimensionality Reduction

This section of the project entails reducing our dataset to a low dimensional dataset using thet-SNE algorithm or PCA. We will be required to perform our analysis and provide insightsgained from our analysis.

```
#Installing and reading the required libraries
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
```

```
library(tidyr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

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

```
library(ggbiplot)
```

```
## Loading required package: ggplot2
```

```
## 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
```

```
## Loading required package: scales
```

```
## Loading required package: grid
```

```
library(ggplot2)
library('devtools')
```

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

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
#install_github("vqv/ggbiplot")
```

Loading the dataset

```
data <- read.csv('http://bit.ly/CarreFourDataset')
# Checking the first 6 rows of the dataset
head(data)
```

```
##    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
```

```
summary(data)
```

```
##   Invoice.ID           Branch           Customer.type        Gender
##  Length:1000        Length:1000        Length:1000        Length:1000
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  Product.line         Unit.price       Quantity          Tax
##  Length:1000        Min.   :10.08   Min.   : 1.00   Min.   : 0.5085
##  Class :character   1st Qu.:32.88   1st Qu.: 3.00   1st Qu.: 5.9249
##  Mode  :character   Median :55.23   Median : 5.00   Median :12.0880
##                     Mean   :55.67   Mean   : 5.51   Mean   :15.3794
##                     3rd Qu.:77.94   3rd Qu.: 8.00   3rd Qu.:22.4453
##                     Max.   :99.96   Max.   :10.00   Max.   :49.6500
##      Date               Time             Payment               cogs
##  Length:1000        Length:1000        Length:1000        Min.   : 10.17
##  Class :character   Class :character   Class :character   1st Qu.:118.50
##  Mode  :character   Mode  :character   Mode  :character   Median :241.76
##                                                           Mean   :307.59
##                                                           3rd Qu.:448.90
##                                                           Max.   :993.00
##  gross.margin.percentage  gross.income         Rating           Total
```

```
##  Min.   :4.762          Min.   : 0.5085   Min.   : 4.000   Min.   :  10.68
##  1st Qu.:4.762          1st Qu.: 5.9249   1st Qu.: 5.500   1st Qu.: 124.42
##  Median :4.762          Median :12.0880   Median : 7.000   Median : 253.85
##  Mean   :4.762          Mean   :15.3794   Mean   : 6.973   Mean   : 322.97
##  3rd Qu.:4.762          3rd Qu.:22.4453   3rd Qu.: 8.500   3rd Qu.: 471.35
##  Max.   :4.762          Max.   :49.6500   Max.   :10.000   Max.   :1042.65
```

```
# Checking the data types in each column
sapply(data,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"
```

```
# Changing the date column to Datetime Object
data$Date <-as.Date(as.character(data$Date))
```

```
# Changing the time column from character type to time type
#strptime(data$Time, format="%Y-%m-%d %H:%M:%S")
```

```
# Checking the number of unique elements in the dataset
lapply(data, function(x) length(table(x)))
```

```
## $Invoice.ID
## [1] 1000
##
## $Branch
## [1] 3
##
## $Customer.type
## [1] 2
##
## $Gender
## [1] 2
##
## $Product.line
## [1] 6
##
## $Unit.price
## [1] 943
##
## $Quantity
## [1] 10
```

```
##
## $Tax
## [1] 990
##
## $Date
## [1] 36
##
## $Time
## [1] 506
##
## $Payment
## [1] 3
##
## $cogs
## [1] 990
##
## $gross.margin.percentage
## [1] 1
##
## $gross.income
## [1] 990
##
## $Rating
## [1] 61
##
## $Total
## [1] 990
```

```
# Checking for duplicate values in dataset
duplicated_rows <- data[duplicated(data),]
print(duplicated_rows)
```

```
##  [1] Invoice.ID              Branch                  Customer.type
##  [4] Gender                  Product.line            Unit.price
##  [7] Quantity                Tax                     Date
## [10] Time                    Payment                 cogs
## [13] gross.margin.percentage gross.income            Rating
## [16] Total
## <0 rows> (or 0-length row.names)
```

```
# Checking for null values in the dataset
is.null(data)
```

```
## [1] FALSE
```

```
# Checking the numerical data in the dataset
str(data)
```

```
## '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                   : Date, format: "0001-05-20" "0003-08-20" ...
##  $ 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 ...
```

```
# Selecting the numerical columns
df <- data[c(6:8,12:16)]
head(df)
```

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

```
# Ensuring our variances is not 0
ndf <- df[ , which(apply(df, 2, var) != 0)]
head(ndf)
```

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

Preview the Principal Components

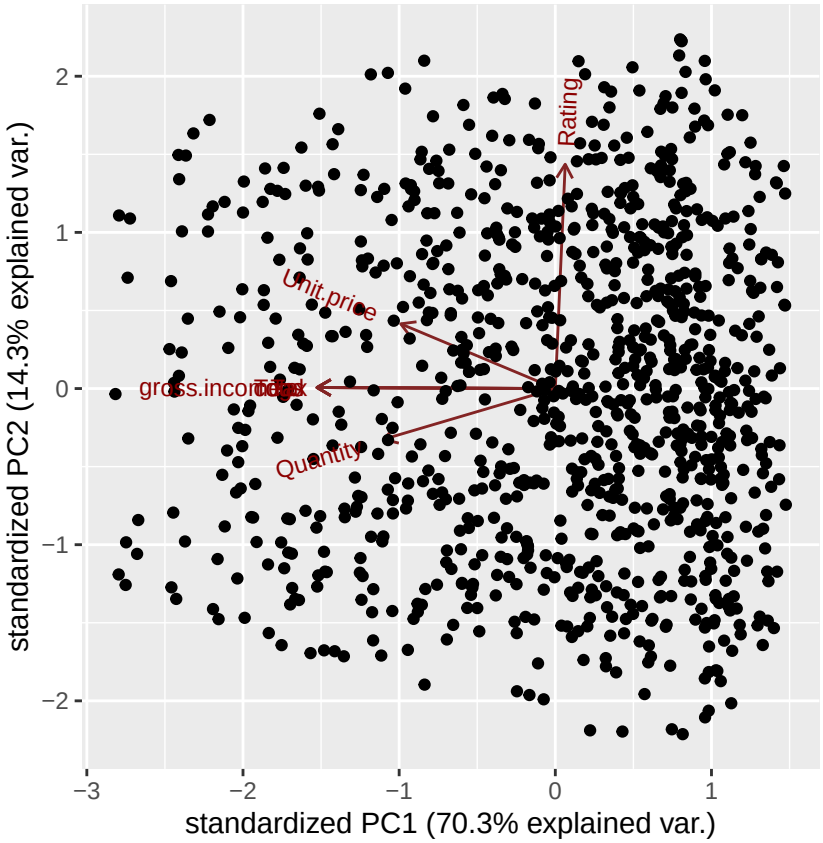
```
# Previewing our PCAs
ndf.pca <- prcomp(ndf, center = TRUE, scale. = TRUE)
summary(ndf.pca)
```

```
## 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
```

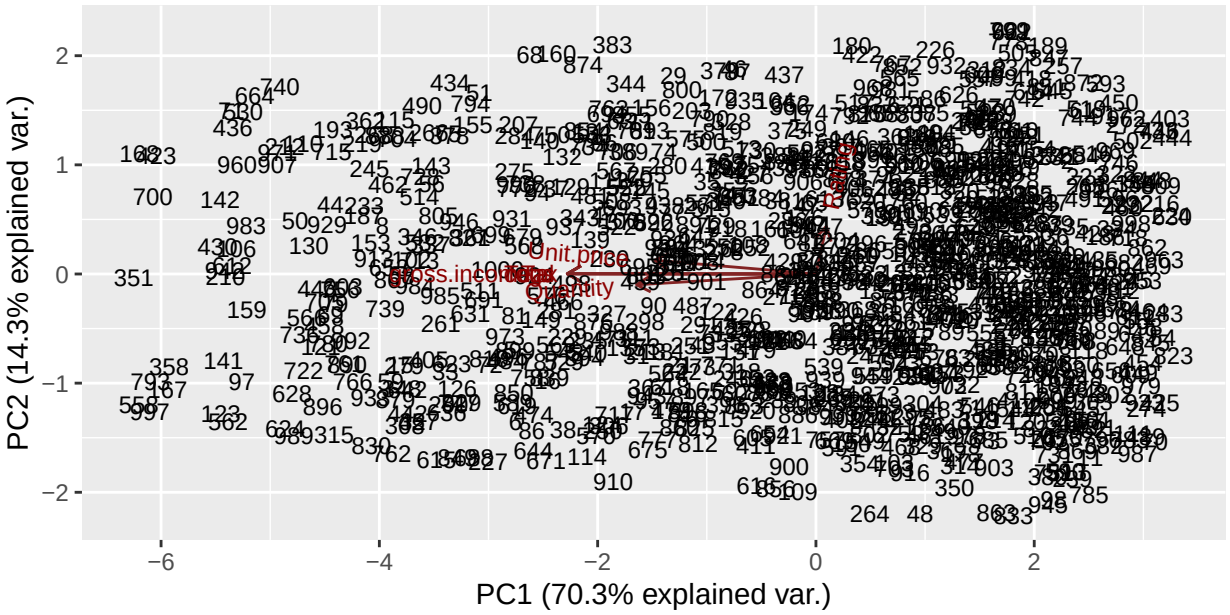
```
# Calling str() to have a look at your PCA object
# ---
#
str(ndf.pca)
```

```
## 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
##   .. ..$ : NULL
##   .. ..$ : chr [1:7] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"
```

```
ggbiplot(ndf.pca)
```




```
# Adding more details to the plot
ggbiplot(ndf.pca, labels=rownames(ndf), obs.scale = 1, var.scale = 3)
```



PART 2: FEATURE SELECTION

```
#Using the numeric Dataset
head(df)
```

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

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

```
## Warning in cor(df): the standard deviation is zero
```

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

# Highly correlated attributes
# ---
#
highlycorrelated
```

```
## [1] 4 8 3
```

```
names(df[,highlycorrelated])
```

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

```
# We can then remove the features that are highly correlated then compare graphically
df2<-df[-highlycorrelated]
head(df2)
```

```
##   Unit.price Quantity gross.margin.percentage gross.income Rating
## 1      74.69        7                4.761905      26.1415    9.1
## 2      15.28        5                4.761905       3.8200    9.6
## 3      46.33        7                4.761905      16.2155    7.4
## 4      58.22        8                4.761905      23.2880    8.4
## 5      86.31        7                4.761905      30.2085    5.3
## 6      85.39        7                4.761905      29.8865    4.1
```

Clusterverse as we will see below uses WRAPPER methods, it uses variable section methodology

to find the optimal subset of variables in a Dataset.

```
# Installing the above mentioned 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)
```

```
# Sequential forward g-search (default)
# ---
#
#out = clustvarsel(df, G = 1:5)
```

```
# Loading data from our csv file
# ---
#
path<-"http://bit.ly/FeatureSelectionDataset2"
Dataset<-read.csv(path, sep = ",", dec = ".",row.names = 1)
head(Dataset)
```

```
##            X1         X2          X3          X4       X5
## 1 -0.71807482 -0.1137642 -1.02888833  0.45394435 1.725742
## 2  3.75019220  3.3638854  3.43319075 -2.12665936 2.469371
## 3 -0.44627119 -0.6971258  2.25009635  4.21566323 1.063453
## 4  0.08522441  0.1547583  0.09926313  0.07124757 1.691745
## 5  4.36181004  2.0209057  4.06428491  0.41207853 1.462167
## 6 -0.52709715  1.2428107 -0.75457360  0.99417826 2.209369
```

```
# Loading data from our csv file
# ---
#
path<-"http://bit.ly/FeatureSelectionDataset2"
Dataset<-read.csv(path, sep = ",", dec = ".",row.names = 1)
head(Dataset)
```

```
##            X1         X2          X3          X4       X5
## 1 -0.71807482 -0.1137642 -1.02888833  0.45394435 1.725742
## 2  3.75019220  3.3638854  3.43319075 -2.12665936 2.469371
## 3 -0.44627119 -0.6971258  2.25009635  4.21566323 1.063453
## 4  0.08522441  0.1547583  0.09926313  0.07124757 1.691745
## 5  4.36181004  2.0209057  4.06428491  0.41207853 1.462167
## 6 -0.52709715  1.2428107 -0.75457360  0.99417826 2.209369
```

```
# Loading data from our csv file
# ---
#
path<-"http://bit.ly/FeatureSelectionDataset2"
Dataset<-read.csv(path, sep = ",", dec = ".",row.names = 1)
head(Dataset)
```

```
##            X1         X2          X3          X4       X5
## 1 -0.71807482 -0.1137642 -1.02888833  0.45394435 1.725742
## 2  3.75019220  3.3638854  3.43319075 -2.12665936 2.469371
## 3 -0.44627119 -0.6971258  2.25009635  4.21566323 1.063453
```

```
## 4  0.08522441  0.1547583  0.09926313  0.07124757 1.691745
## 5  4.36181004  2.0209057  4.06428491  0.41207853 1.462167
## 6 -0.52709715  1.2428107 -0.75457360  0.99417826 2.209369
```

```
# Sequential forward greedy search (default)
# ---
#
out = clustvarsel(Dataset, G = 1:5)
```

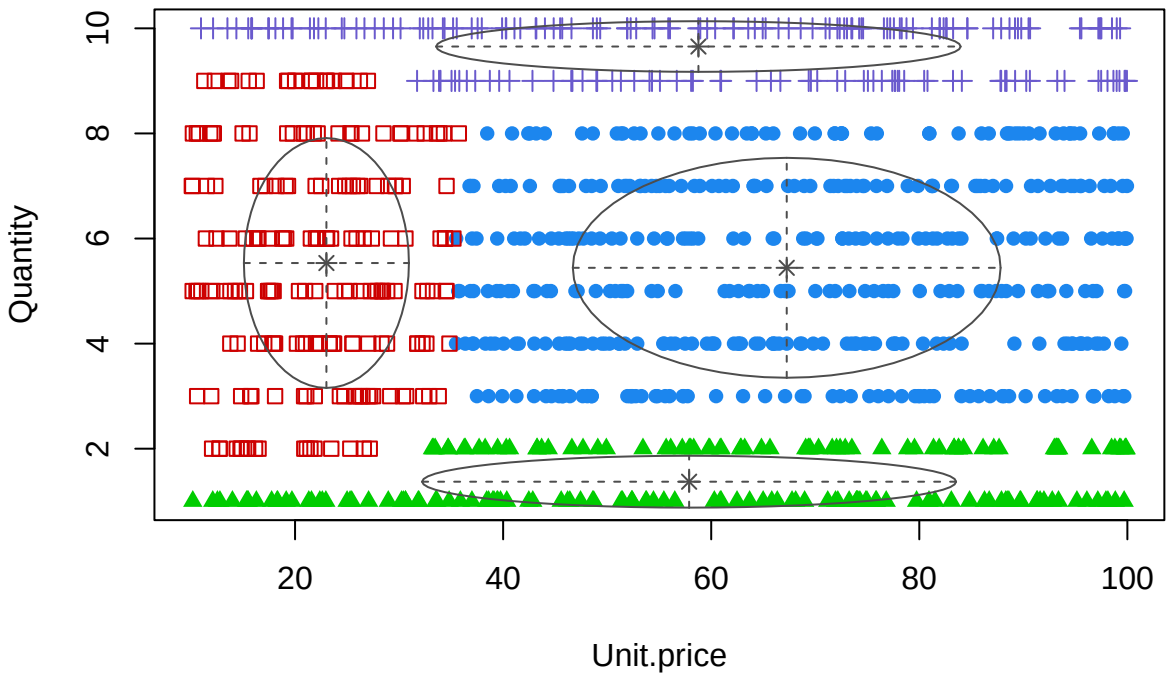
```
# The selection algorithm would indicate that the subset
# we use for the clustering model is composed of variables X1 and X2
# and that other variables should be rejected.
# Having identified the variables that we use, we proceed to build the clustering model:
# ---
#

Subset1 = df[,out$subset]
mod = Mclust(Subset1, G = 1:5)
```

```
summary(mod)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VVI (diagonal, varying volume and shape) model with 4 components:
##
##  log-likelihood    n df       BIC       ICL
##       -6969.135 1000 19 -14069.52 -14409.77
##
## Clustering table:
##   1   2   3   4
## 421 205 185 189
```

```
plot(mod,c("classification"))
```


Quantity
10
8
6
4
2
20
40
60
80
100
Unit.price