Carrefour

Daisy Lynn

2022-04-01

```
library(readr)
library(caret)
```

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

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

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(ggplot2)
library(tidyverse)
```

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

```
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v purrr   0.3.4     v forcats 0.5.1
```

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

```
library(dplyr)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(Rtsne)
library(superml)
```

```
## Loading required package: R6
```

```r
library(clustvarsel)
```

```
## Loading required package: mclust

## Package 'mclust' version 5.4.9
## Type 'citation("mclust")' for citing this R package in publications.

##
## Attaching package: 'mclust'

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

## Package 'clustvarsel' version 2.3.4

## Type 'citation("clustvarsel")' for citing this R package in publications.
```

```r
library(mclust)
```

Loading dataset

Below dataset will be used for the practice of dimensionality reduction and feature selection

```r
data1 <- read_csv("http://bit.ly/CarreFourDataset")
```

```
## Rows: 1000 Columns: 16
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): Invoice ID, Branch, Customer type, Gender, Product line, Date, Pay...
## dbl  (8): Unit price, Quantity, Tax, cogs, gross margin percentage, gross in...
## time (1): Time
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
head(data1)
```

```
## # A tibble: 6 x 16
##   `Invoice ID` Branch `Customer type` Gender `Product line`         `Unit price`
##   <chr>        <chr>  <chr>           <chr>  <chr>                         <dbl>
## 1 750-67-8428  A      Member          Female Health and beauty              74.7
## 2 226-31-3081  C      Normal          Female Electronic accessories         15.3
## 3 631-41-3108  A      Normal          Male   Home and lifestyle             46.3
## 4 123-19-1176  A      Member          Male   Health and beauty              58.2
## 5 373-73-7910  A      Normal          Male   Sports and travel              86.3
## 6 699-14-3026  C      Normal          Male   Electronic accessories         85.4
## # ... with 10 more variables: Quantity <dbl>, Tax <dbl>, Date <chr>,
## #   Time <time>, Payment <chr>, cogs <dbl>, `gross margin percentage` <dbl>,
## #   `gross income` <dbl>, Rating <dbl>, Total <dbl>
```

```r
#checking size of dataframe

dim(data1)
```

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

There are 1000 records and 16 variables

```r
# Identifying missing data in dataset

colSums(is.na(data1))
```

```
##              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 missing values

Dimensionality reduction

```r
Label<-data1$Branch
data1$Branch<-as.factor(data1$Branch)

# Assign colors
colors = rainbow(length(unique(data1$Branch)))
names(colors) = unique(data1$Branch)
```

```r
# Creating a new dataframe which specifies which features to be used

data_use <- data1[, c(2:8, 11, 12, 14, 15, 16)]

head(data_use)
```

```
## # A tibble: 6 x 12
##   Branch `Customer type` Gender `Product line`       `Unit price` Quantity   Tax
##   <fct>  <chr>           <chr>  <chr>                       <dbl>    <dbl> <dbl>
## 1 A      Member          Female Health and beauty            74.7        7 26.1
## 2 C      Normal          Female Electronic accessor~         15.3        5  3.82
## 3 A      Normal          Male   Home and lifestyle           46.3        7 16.2
## 4 A      Member          Male   Health and beauty            58.2        8 23.3
```

```
## 5 A      Normal          Male   Sports and travel            86.3        7 30.2
## 6 C      Normal          Male   Electronic accessor~         85.4        7 29.9
## # ... with 5 more variables: Payment <chr>, cogs <dbl>, ‘gross income‘ <dbl>,
## #   Rating <dbl>, Total <dbl>
```

```
unique(data_use$Payment)
```

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

```
lbl <- LabelEncoder$new()


data_new <- data_use %>%
  mutate(`Customer type` = factor(lbl$fit_transform(.$`Customer type`)),
         Gender = factor(lbl$fit_transform(.$Gender)),
         `Product line` = factor(lbl$fit_transform(.$`Product line`)),
         Payment = factor(lbl$fit_transform(.$Payment)))
```

```
#viewing the new dataset
head(data_new)
```

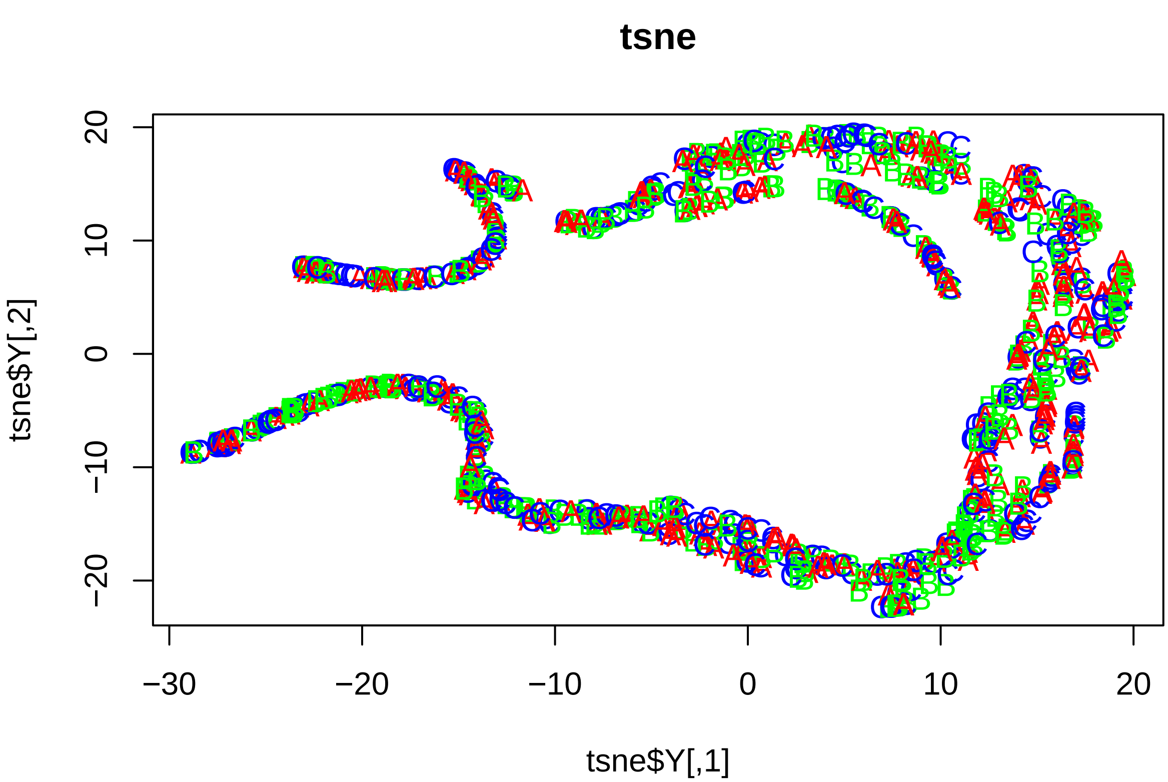
```
## # A tibble: 6 x 12
##   Branch ‘Customer type‘ Gender ‘Product line‘ ‘Unit price‘ Quantity   Tax
##   <fct>  <fct>           <fct>  <fct>                 <dbl>    <dbl> <dbl>
## 1 A      0               0      0                      74.7        7 26.1
## 2 C      1               0      1                      15.3        5  3.82
## 3 A      1               1      2                      46.3        7 16.2
## 4 A      0               1      0                      58.2        8 23.3
## 5 A      1               1      3                      86.3        7 30.2
## 6 C      1               1      1                      85.4        7 29.9
## # ... with 5 more variables: Payment <fct>, cogs <dbl>, ‘gross income‘ <dbl>,
## #   Rating <dbl>, Total <dbl>
```

```
# Executing the algorithm
#
tsne <- Rtsne(data_new,dims = 2, perplexity=30, verbose=TRUE, max_iter = 500)
```

```
## Performing PCA
## Read the 1000 x 19 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 0.20 seconds (sparsity = 0.101260)!
## Learning embedding...
## Iteration 50: error is 58.761062 (50 iterations in 0.12 seconds)
## Iteration 100: error is 51.508937 (50 iterations in 0.09 seconds)
## Iteration 150: error is 50.183960 (50 iterations in 0.11 seconds)
## Iteration 200: error is 49.655824 (50 iterations in 0.10 seconds)
## Iteration 250: error is 49.399839 (50 iterations in 0.11 seconds)
## Iteration 300: error is 0.574251 (50 iterations in 0.10 seconds)
```

```
## Iteration 350: error is 0.415194 (50 iterations in 0.10 seconds)
## Iteration 400: error is 0.365160 (50 iterations in 0.10 seconds)
## Iteration 450: error is 0.347704 (50 iterations in 0.10 seconds)
## Iteration 500: error is 0.338604 (50 iterations in 0.12 seconds)
## Fitting performed in 1.04 seconds.
```

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



model was a success and the dimension has been reduced to a lower one, this helps to flexibly and easily work with the data

Feature Selection

```
feature<- data_new
feature
```

```
## # A tibble: 1,000 x 12
##    Branch `Customer type` Gender `Product line` `Unit price` Quantity   Tax
##    <fct>  <fct>           <fct>  <fct>                 <dbl>    <dbl> <dbl>
##  1 A      0               0      0                      74.7        7  26.1
```

```
##   2 C      1              0      1                      15.3        5  3.82
##   3 A      1              1      2                      46.3        7 16.2
##   4 A      0              1      0                      58.2        8 23.3
##   5 A      1              1      3                      86.3        7 30.2
##   6 C      1              1      1                      85.4        7 29.9
##   7 A      0              0      1                      68.8        6 20.7
##   8 C      1              0      2                      73.6       10 36.8
##   9 A      0              0      0                      36.3        2  3.63
## 10 B      0              0      4                      54.8        3  8.23
## # ... with 990 more rows, and 5 more variables: Payment <fct>, cogs <dbl>,
## #   `gross income` <dbl>, Rating <dbl>, Total <dbl>
```

```
#Changing the factor columns to numeric

#Branch
feature$Branch <- factor(feature$Branch)
feature$Branch <- as.numeric(feature$Branch)

# Customer type
feature$`Customer type` <- factor(feature$`Customer type`)
feature$`Customer type` <- as.numeric(feature$`Customer type`)

# Gender
feature$Gender <- factor(feature$Gender)
feature$Gender <- as.numeric(feature$Gender)

# Product line
feature$`Product line` <- factor(feature$`Product line`)
feature$`Product line` <- as.numeric(feature$`Product line`)

#Payment
feature$Payment <- factor(feature$Payment)
feature$Payment <- as.numeric(feature$Payment)
```

```
#checking if the data type has changed
head(feature)
```

```
## # A tibble: 6 x 12
##   Branch `Customer type` Gender `Product line` `Unit price` Quantity   Tax
##    <dbl>           <dbl>  <dbl>          <dbl>        <dbl>    <dbl> <dbl>
## 1      1               1      1              1         74.7        7 26.1
## 2      3               2      1              2         15.3        5  3.82
## 3      1               2      2              3         46.3        7 16.2
## 4      1               1      2              1         58.2        8 23.3
## 5      1               2      2              4         86.3        7 30.2
## 6      3               2      2              2         85.4        7 29.9
## # ... with 5 more variables: Payment <dbl>, cogs <dbl>, `gross income` <dbl>,
## #   Rating <dbl>, Total <dbl>
```

the data types have been changed

Filter Method

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

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

```
## [1]  9 12  7
```

```
names(feature[,highlyCorrelated])
```

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

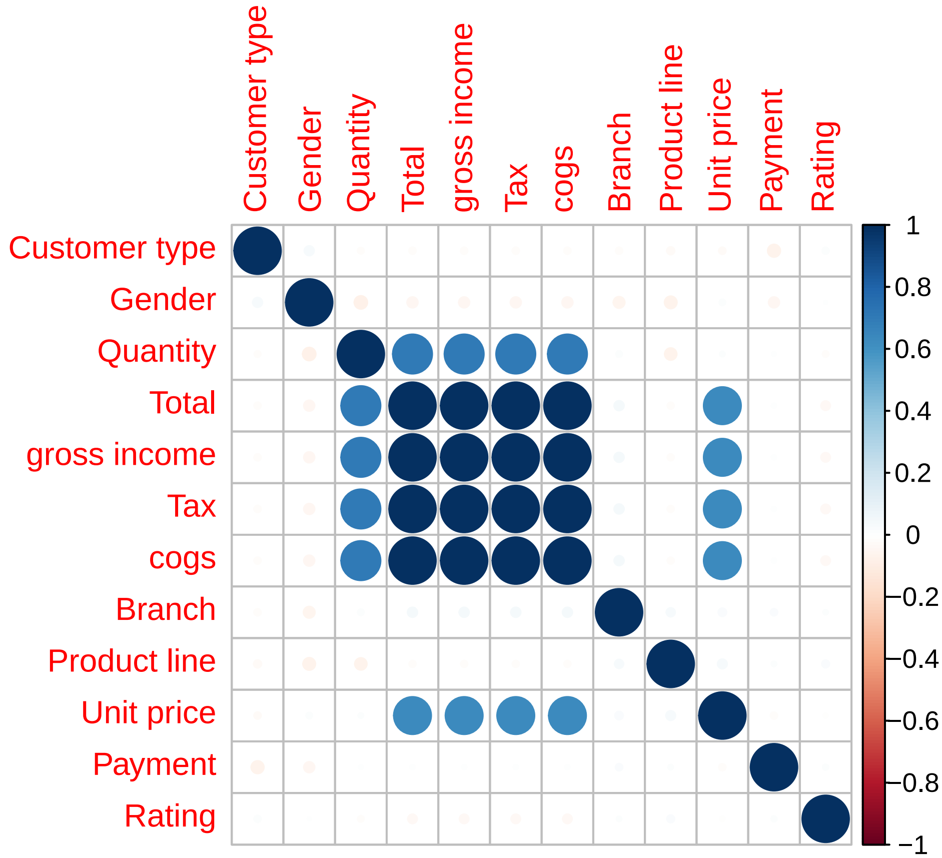
cogs, total and tax have a high correlation

```
#removing the variables with a higher correlation
feature_clean <- feature[-highlyCorrelated]

head(feature_clean)
```

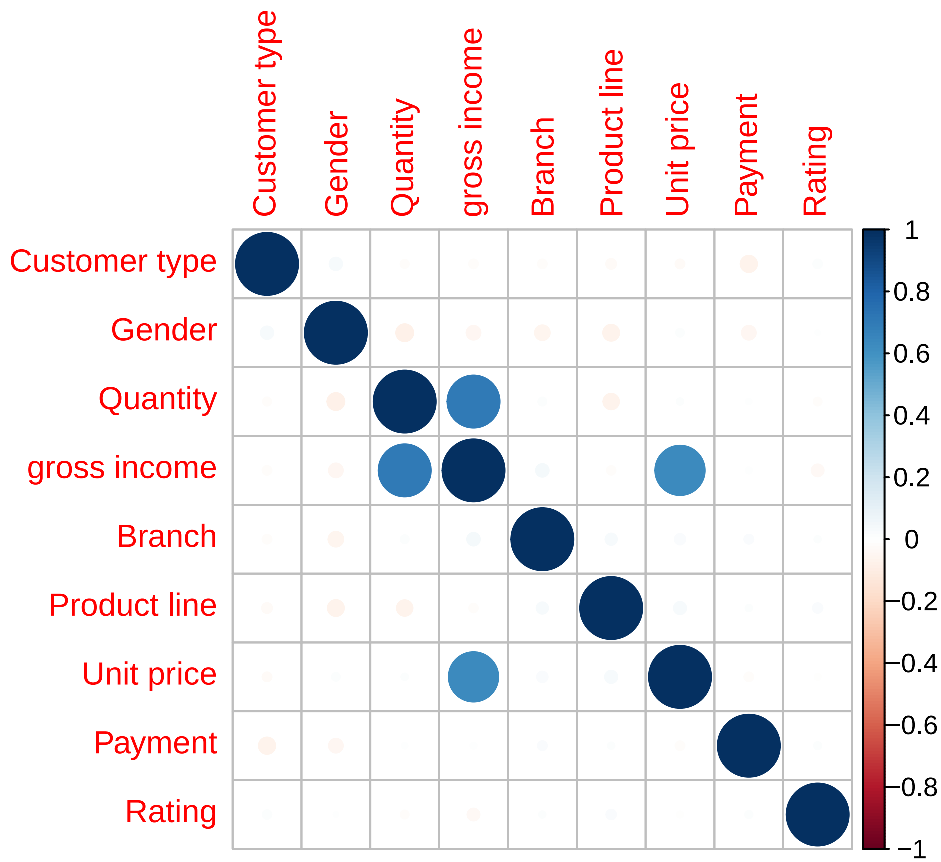
```
## # A tibble: 6 x 9
##   Branch `Customer type` Gender `Product line` `Unit price` Quantity Payment
##    <dbl>           <dbl>  <dbl>          <dbl>        <dbl>    <dbl>   <dbl>
## 1      1               1      1              1         74.7        7       1
## 2      3               2      1              2         15.3        5       2
## 3      1               2      2              3         46.3        7       3
## 4      1               1      2              1         58.2        8       1
## 5      1               2      2              4         86.3        7       1
## 6      3               2      2              2         85.4        7       1
## # ... with 2 more variables: `gross income` <dbl>, Rating <dbl>
```

cogs, total and tax have been removed

```
#lets check how the original correlation was

corrplot(cor(feature), order = 'hclust')
```




```
#lets check how the correlation is after removing the highly correlated variables

corrplot(cor(feature_clean), order = 'hclust')
```



Comparing the original correlation to the clean correlation, our correlation matrix has improved

The above correlation for the clean dataset shows variables with a high enough significance level

Wrapper Method

```
#greedy search

greedy = clustvarsel(feature, G = 1:5)
greedy
```

```
## ------------------------------------------------------
## Variable selection for Gaussian model-based clustering
## Stepwise (forward/backward) greedy search
## ------------------------------------------------------
##
##  Variable proposed Type of step   BICclust Model G     BICdiff Decision
##                Tax          Add  -7382.354     V 4    389.0238 Accepted
##       gross income          Add  55117.386   VEV 3   2502.9883 Accepted
##           Quantity          Add -16164.602   VVI 5 -66967.5199 Rejected
##                Tax       Remove  -7392.222     V 3   2512.8564 Rejected
##
## Selected subset: Tax, gross income
```

algorithm selected Tax and gross income, they'll be used for the model

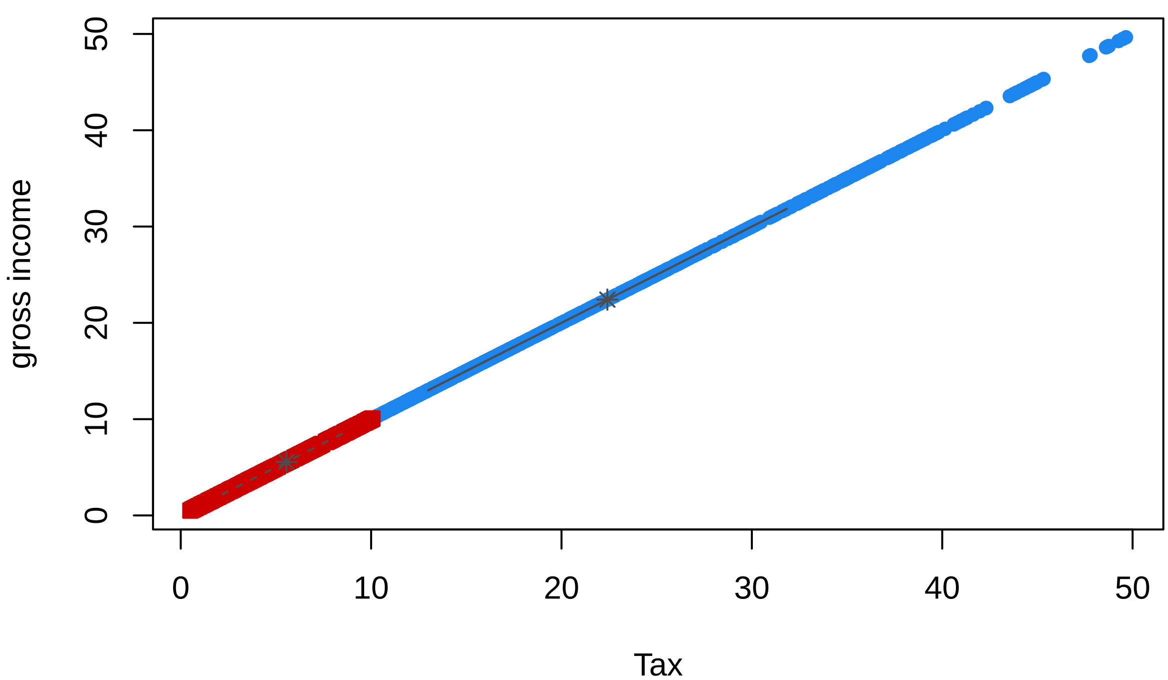
```r
Subset1 = feature[,greedy$subset]
model = Mclust(Subset1, G = 1:5)
summary(model)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEV (ellipsoidal, equal shape) model with 2 components:
##
##  log-likelihood    n df      BIC      ICL
##        27364.17 1000 10 54659.26 54524.45
##
## Clustering table:
##   1   2
## 564 436
```

model has chosen 2clusters 1 with 564, 2 with 436

```r
plot(model,c("classification"))
```

```
## Warning in sqrt(rev(sort(ev$values))): NaNs produced
```



our two variables chosen by the greedy algorithm have a good linear relationship

The model can be considered a success since it was able to pick variables and compare the well

Conclusions

two methods were used to determine which features contribute the most information to the dataset some of these features are gross income , tax, quantity