Independent_Project_Week_14_Part2

Jonah okiru

2022-06-10

Part 3: Association Rules

1. Problem definition.

a). Specify the question.

To create association rules that will allow one to identify the relationship among the variables in the dataset.

b). The metric of success

The rule model should be in position to identify the relationship between the variables with a confidence level of at least 90%.

c). The context

Supermarket stock 100 of thousands of items , for the purpose of the convenient to there customers, items that supplement each other are stocked together in other to ease their customers with the burden of long search for the items. In order to achieve the above goal the supermarket management needs to identify the relationship between the items in the supermarket so that it could be stocked appropriately.

d). Experimental design

- Load the data

-Association analysis

2. Source the data

The data that was used in the project was provided by the education institution.

3. Load the data

```
df = read.csv("D:/R studio/week3 R/Supermarket_Sales_Dataset II.csv", header = 0)
```

4. Check the data.

```
#Load the data.table and library
library(data.table)
library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
#Preview the 3 rows of the data
path = "D:/R studio/week3 R/Supermarket_Sales_Dataset II.csv"
#Load the transaction dataset from csv file.
Transactions <- read.transactions(path, header = 0)
```

```
## Warning in asMethod(object): removing duplicated items in transactions
```

```
Transactions
```

```
## transactions in sparse format with
##  7501 transactions (rows) and
##  5729 items (columns)
```

```
# Verifying the object's class
class(Transactions)
```

```
## [1] "transactions"
## attr(,"package")
## [1] "arules"
```

```
#Preview the first seven transactions
inspect(Transactions[1:7])
```

```
##     items
## [1] {cheese,energy,
##      drink,tomato,
##      fat,
##      flour,yams,cottage,
##      grapes,whole,
##      juice,frozen,
##      juice,low,
##      mix,green,
##      oil,
##      shrimp,almonds,avocado,vegetables,
##      smoothie,spinach,olive,
##      tea,honey,salad,mineral,
##      water,salmon,antioxydant,
##      weat,
##      yogurt,green}
## [2] {burgers,meatballs,eggs}
## [3] {chutney}
## [4] {turkey,avocado}
## [5] {bar,whole,
##      mineral,
##      rice,green,
##      tea,
##      water,milk,energy,
##      wheat}
## [6] {fat,
##      low,
##      yogurt}
## [7] {fries,
##      pasta,french,
##      wheat,
##      whole}
```

From the print out above the first item of the transactions are cheese, energy, drink, tomato, fat, flour. etc.

```
#Preview items that make up our datasets
items <- as.data.frame(itemLabels(Transactions))
colnames(items) <- "item"
#Preview the first seven items
head(items, 7)
```

```
##                                       item
## 1                                        &
## 2                              accessories
## 3                  accessories,antioxydant
## 4              accessories,champagne,fresh
## 5            accessories,champagne,protein
## 6                    accessories,chocolate
## 7 accessories,chocolate,champagne,frozen
```

The items that makes top two of the data sets are as follows; &, accessories

```
#Summary of the transactions dataset
summary(Transactions)
```

```
## transactions as itemMatrix in sparse format with
##  7501 rows (elements/itemsets/transactions) and
##  5729 columns (items) and a density of 0.0005421748
```

```
##
## most frequent items:
##     tea   wheat mineral     fat  yogurt (Other)
##     803     645     577     574     543   20157
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   15   16
## 1603 2007 1382  942  651  407  228  151   70   39   13    5    1    1    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   3.106   4.000  16.000
##
## includes extended item information - examples:
##                    labels
## 1                        &
## 2             accessories
## 3 accessories,antioxydant
```

from the summary above of the transaction , -the transaction has 7501 rows, 5729 columns. -The most frequent item is tea at 803 followed by wheat at 645. -The (itemset/transaction) are as follows size “1” is 1603 items, “2” is 2007. etc

```
# Exploring the frequency of some articles
# i.e. transacations ranging from 3040 to 3050 and performing
# some operation in percentage terms of the total transactions
itemFrequency(Transactions[, 3040:3041],type = "absolute")
```

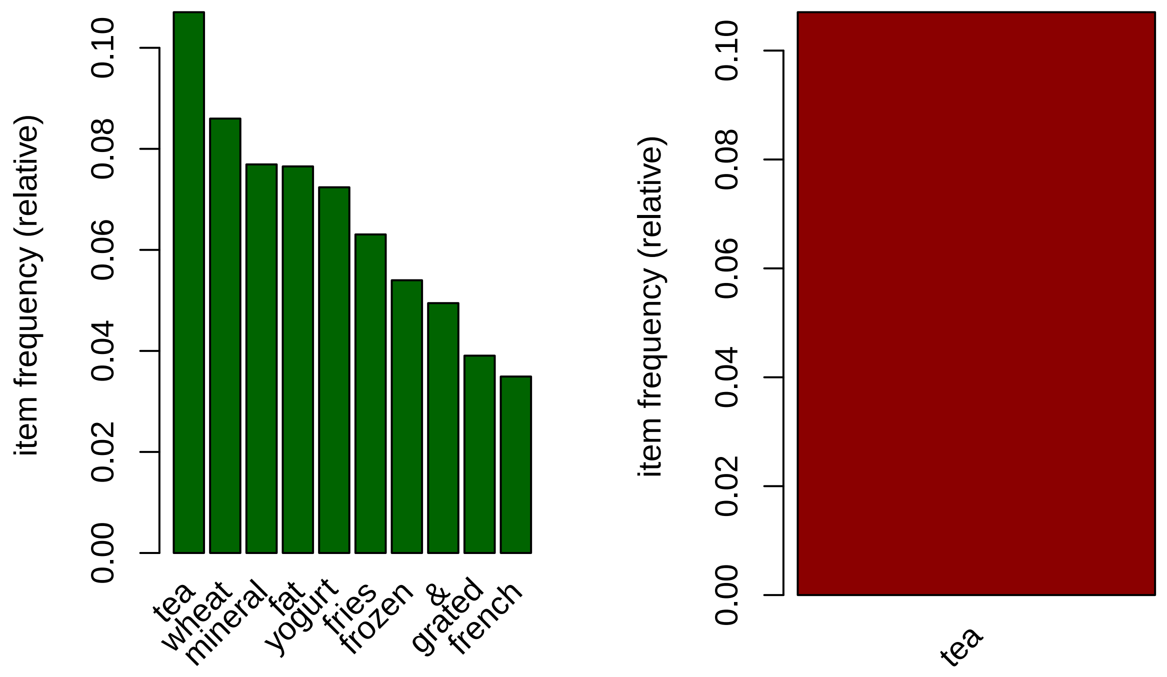
```
## pepper,spaghetti,pancakes,low pepper,spaghetti,rice,cooking
##                             1                            1
```

```
round(itemFrequency(Transactions[, 3040:3050],type = "relative")*100,2)
```

```
##                                    pepper,spaghetti,pancakes,low
##                                                             0.01
##                                    pepper,spaghetti,rice,cooking
##                                                             0.01
## pepper,spaghetti,salmon,honey,cake,chili,carrots,cereals,french
##                                                             0.01
##                                      pepper,spaghetti,soup,olive
##                                                             0.01
##                                          pepper,spaghetti,strong
##                                                             0.03
##                                    pepper,spaghetti,yams,clothes
##                                                             0.01
##                                                    pepper,tomato
##                                                             0.09
##                                           pepper,tomatoes,ground
##                                                             0.08
##                                          pepper,tomatoes,mineral
##                                                             0.04
##                                   pepper,tomatoes,pepper,mineral
##                                                             0.01
##                                  pepper,tomatoes,spaghetti,mineral
##                                                               0.03
```

```
# Producing a chart of frequencies and fitering
# to consider only items with a minimum percentage
# of support/ considering a top x of items
# ---
# Displaying top 10 most common items in the transactions dataset
# and the items whose relative importance is at least 10%
#
par(mfrow = c(1, 2))

# plot the frequency of items
itemFrequencyPlot(Transactions, topN = 10,col="darkgreen")
itemFrequencyPlot(Transactions, support = 0.1,col="darkred")
```



From the bar chart above the top most common items in the data sets are as follows tea, wheat, nineral, fat, yogurt, fries etc.

```
#Build the model based on association rules using apriori function.
#I will use the minimum support as 0.002 and confidence as 0.85

rules <- apriori(Transactions, parameter = list(supp = 0.001, conf = 0.85))
```

```
## Apriori
```

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.85    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 7
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[5729 item(s), 7501 transaction(s)] done [0.01s].
## sorting and recoding items ... [354 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [247 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules
```

```
## set of 247 rules
```

The model built has 247 rules.

```
#Perform exploration of the model using the summary function
summary(rules)
```

```
## set of 247 rules
##
## rule length distribution (lhs + rhs):sizes
##   2   3   4
##  96 134  17
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    2.00    3.00    2.68    3.00    4.00
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.001067   Min.   :0.8571   Min.   :0.001067   Min.   :  8.096
##  1st Qu.:0.001200   1st Qu.:1.0000   1st Qu.:0.001200   1st Qu.: 11.630
##  Median :0.001466   Median :1.0000   Median :0.001600   Median : 13.068
##  Mean   :0.002907   Mean   :0.9775   Mean   :0.003014   Mean   : 22.846
##  3rd Qu.:0.002666   3rd Qu.:1.0000   3rd Qu.:0.002800   3rd Qu.: 20.218
##  Max.   :0.068391   Max.   :1.0000   Max.   :0.076523   Max.   :613.718
##      count
##  Min.   :  8.00
##  1st Qu.:  9.00
##  Median : 11.00
##  Mean   : 21.81
##  3rd Qu.: 20.00
##  Max.   :513.00
##
## mining info:
##          data ntransactions support confidence
##  Transactions          7501   0.001       0.85
##                                                                        call
##  apriori(data = Transactions, parameter = list(supp = 0.001, conf = 0.85))
```

rule length distribution (lhs + rhs):sizes 2 3 4 96 134 17

Min. 1st Qu. Median Mean 3rd Qu. Max. 2.00 2.00 3.00 2.68 3.00 4.00

summary of quality measures: support confidence coverage lift count
Min. :0.001067 Min. :0.8571 Min. :0.001067 Min. : 8.096 Min. : 8.00
1st Qu.:0.001200 1st Qu.:1.0000 1st Qu.:0.001200 1st Qu.: 11.630 1st Qu.: 9.00
Median :0.001466 Median :1.0000 Median :0.001600 Median : 13.068 Median : 11.00
Mean :0.002907 Mean :0.9775 Mean :0.003014 Mean : 22.846 Mean : 21.81
3rd Qu.:0.002666 3rd Qu.:1.0000 3rd Qu.:0.002800 3rd Qu.: 20.218 3rd Qu.: 20.00
Max. :0.068391 Max. :1.0000 Max. :0.076523 Max. :613.718 Max. :513.00

```r
#Check the first 7 build rules of the model
inspect(rules[1:7])
```

```
##     lhs                                 rhs      support     confidence
## [1] {cookies,low}                    => {yogurt} 0.001066524 1
## [2] {cookies,low}                    => {fat}    0.001066524 1
## [3] {extra}                          => {dark}   0.001066524 1
## [4] {burgers,whole}                  => {wheat}  0.001199840 1
## [5] {fries,escalope,pasta,mushroom} => {cream}  0.001066524 1
## [6] {fries,cookies,green}            => {tea}    0.001333156 1
## [7] {shrimp,whole}                   => {wheat}  0.001066524 1
##     coverage    lift     count
## [1] 0.001066524 13.81400  8
## [2] 0.001066524 13.06794  8
## [3] 0.001066524 83.34444  8
## [4] 0.001199840 11.62946  9
## [5] 0.001066524 47.77707  8
## [6] 0.001333156  9.34122 10
## [7] 0.001066524 11.62946  8
```

The interpretation of the rules is as follows; 1. rule number one states that if someone buy cookies and low they are 100% likely to buy yogurt. 2. The second rule states that if someone buys cookies and low they are 100% likely to buy fat.

```r
#order the rules by support, preview the first 7 rules
rules<-sort(rules, by="support", decreasing=TRUE)
inspect(rules[1:7])
```

```
##     lhs                   rhs      support     confidence coverage    lift
## [1] {yogurt}           => {fat}    0.068390881 0.9447514  0.072390348 12.34596
## [2] {fat}              => {yogurt} 0.068390881 0.8937282  0.076523130 12.34596
## [3] {herb}             => {&}      0.030929209 1.0000000  0.030929209 20.21833
## [4] {whole}            => {wheat}  0.018930809 0.9466667  0.019997334 11.00922
## [5] {rice}            => {wheat}  0.012265031 0.9583333  0.012798294 11.14490
## [6] {pepper,ground}   => {&}      0.009065458 1.0000000  0.009065458 20.21833
## [7] {mineral, yogurt} => {fat}    0.007732302 1.0000000  0.007732302 13.06794
##     count
## [1] 513
## [2] 513
## [3] 232
## [4] 142
## [5]  92
## [6]  68
## [7]  58
```

The rules above are ordered by the size of their support in the decreasing order.

```
#The items the customer purchased before buying fat
fat <- subset(rules, subset = rhs %pin% "fat")

# Then order by confidence
fat<-sort(fat, by="confidence", decreasing=TRUE)
inspect(fat[1:5])
```

```
##     lhs                  rhs   support     confidence coverage    lift
## [1] {mineral, yogurt} => {fat} 0.007732302 1          0.007732302 13.06794
## [2] {wheat, yogurt}   => {fat} 0.006932409 1          0.006932409 13.06794
## [3] {low}             => {fat} 0.006132516 1          0.006132516 13.06794
## [4] {low, yogurt}     => {fat} 0.005999200 1          0.005999200 13.06794
## [5] {&, yogurt}       => {fat} 0.005199307 1          0.005199307 13.06794
##     count
## [1] 58
## [2] 52
## [3] 46
## [4] 45
## [5] 39
```

The above rules are ordered by the level of confidence the top items the customer purchased before purchasing the fat.

```
#the items that customers might buy who have previously bought fat
# Subset the rules
fat <- subset(rules, subset = lhs %pin% "yogurt")

# Order by confidence
fat<-sort(fat, by="confidence", decreasing=TRUE)

# inspect top 5
inspect(fat[15:19])
```

```
##     lhs                             rhs   support     confidence coverage
## [1] {smoothie,low, yogurt}       => {fat} 0.002532996 1          0.002532996
## [2] {juice,low, yogurt}          => {fat} 0.002532996 1          0.002532996
## [3] {oil,low, yogurt}            => {fat} 0.002399680 1          0.002399680
## [4] {vegetables,ground, yogurt} => {fat} 0.002133049 1          0.002133049
## [5] {beef,mineral, yogurt}      => {fat} 0.002133049 1          0.002133049
##     lift     count
## [1] 13.06794 19
## [2] 13.06794 19
## [3] 13.06794 18
## [4] 13.06794 16
## [5] 13.06794 16
```

The Above rules displays the items the customer might buy who previously purchased the fat.

4. The insight of the analysis

1. The top 5 items the customer who previously purchased fat might bought are

as follows:{smoothie,low, yogurt}, {juice,low, yogurt}, {oil,low, yogurt}, {vegetables,ground, yogurt} and {beef,mineral, yogurt}.
2.The items the customer purchased before buying fat are as follows; {mineral, yogurt}, {wheat, yogurt}, {low}, {low, yogurt} and {&, yogurt} respectively.

5. The recomendation

1. The fats should be stored near the following items in the supermarket

shelves; yogurt, smoothie, juice, low, oil, low, vegetables, grounds, beef, minerals and wheat.

6. Challenge the solution.

The metric of success of the model was to identify the relationship between the variables with the confidence level of 90% and above but some variables relationship have been identified with the confidence level of less than 90%.

7. Follow up question.

a). Do we have the right data?

```
Yes, the data was appropriate
```

b). Do we need another data?

No, the data was appropriate.

c).Do we have the right question?

Yes, the question is clear.

Part 4 : Anomaly Detection

1. Problem definition

a). Specify the question

To check for the anomalies in the sales dataset, with the aim of detecting fraud.

b). The metric of success

There is no model built , so there' is no metric of success

c). The context

In the sales data sometimes, the un usual situations happens, either the figure is too much than the usual or the to small than normal figure. All of these discrepancy could be detected as abnormal in the dataset which helps to detect fraud in the sales data

d). Experimental design

- Load the data

-Anormaly detection.

2. Source the data

The data that was used in the project was provided by the education

institution.

3. Load the data.

```r
#Load the data
carrefour_forescast <- read.csv("http://bit.ly/CarreFourSalesDataset")
#Preview the dataset
head(carrefour_forescast, 2)
```

```
##       Date    Sales
## 1 1/5/2019 548.9715
## 2 3/8/2019  80.2200
```

4. Implement the solution

```r
#Check the datatypes of the data columns of the dataset
str(carrefour_forescast)
```

```
## 'data.frame':    1000 obs. of  2 variables:
##  $ Date : chr  "1/5/2019" "3/8/2019" "3/3/2019" "1/27/2019" ...
##  $ Sales: num  549 80.2 340.5 489 634.4 ...
```

```
#Convert the date from string to date data type
carrefour_forescast$Date <- as.Date(carrefour_forescast$Date, "%m/%d/%Y")
```

```
#Sort the date column
carrefour_forescast$Date <- sort(carrefour_forescast$Date)
```

```
#Load the library
library(tibbletime)
```

```
##
## Attaching package: 'tibbletime'

## The following object is masked from 'package:stats':
##
##     filter
```

```
#Convert the dataset to tibble
carrefour_forescast <- as_tbl_time(carrefour_forescast, index= Date)
#Preview the dataset
head(carrefour_forescast, 3)
```

```
## # A time tibble: 3 x 2
## # Index: Date
##   Date       Sales
##   <date>     <dbl>
## 1 2019-01-01 549.
## 2 2019-01-01  80.2
## 3 2019-01-01 341.
```

```
#load the library
library(anomalize)
```

```
## == Use anomalize to improve your Forecasts by 50%! =============================
## Business Science offers a 1-hour course - Lab #18: Time Series Anomaly Detection!
## </> Learn more at: https://university.business-science.io/p/learning-labs-pro </>
```

```
library(tidyverse)
```

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

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()   masks data.table::between()
## x tidyr::expand()    masks Matrix::expand()
## x dplyr::filter()    masks tibbletime::filter(), stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x tidyr::pack()      masks Matrix::pack()
## x dplyr::recode()    masks arules::recode()
## x purrr::transpose() masks data.table::transpose()
## x tidyr::unpack()    masks Matrix::unpack()
```

```
library(tibbletime)
```

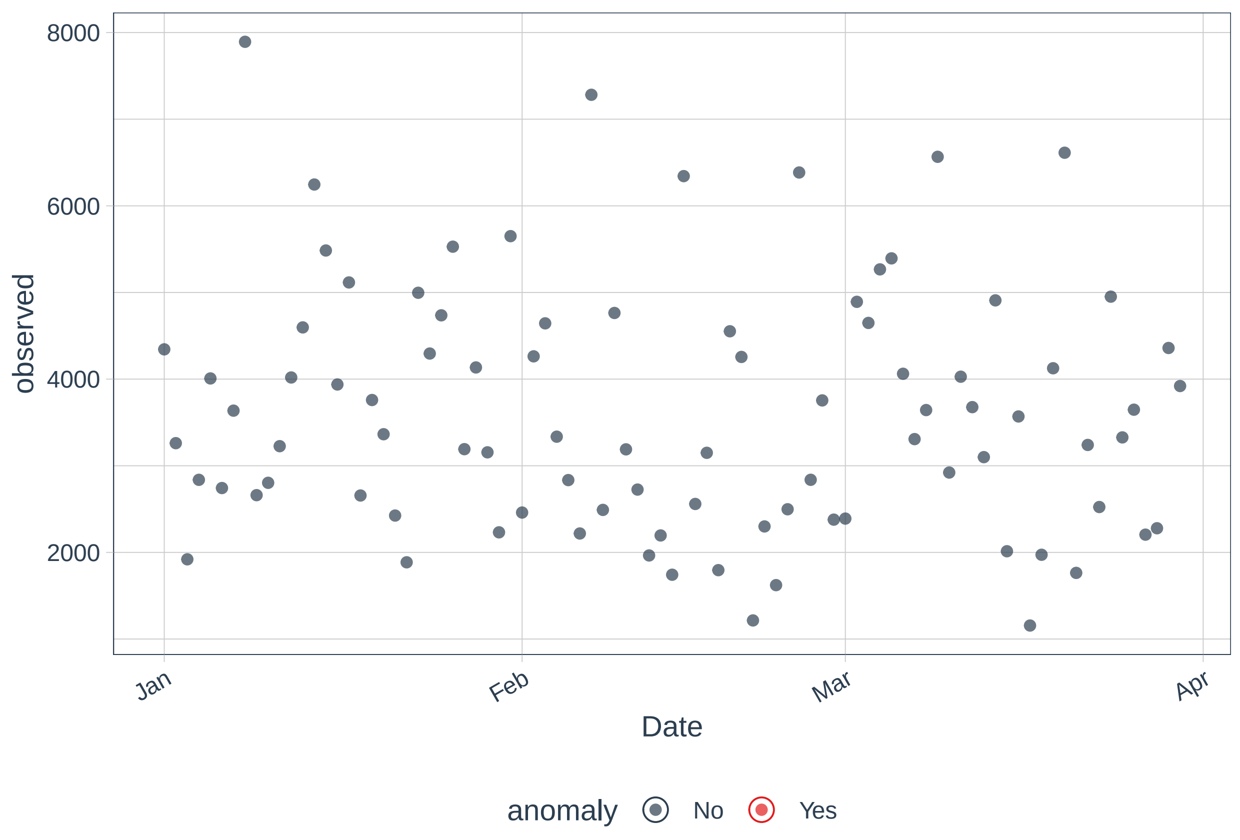
```
#Check for a normally
df_Anormally <- carrefour_forescast %>%
  group_by(Date) %>%
  summarise(Sales = sum(Sales)) %>%
  time_decompose(Sales) %>%
  anomalize(remainder) %>%
  time_recompose()
```

```
## frequency = 7 days
```

```
## trend = 30 days
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
#plot the df_Anormally
df_Anormally %>% plot_anomalies(ncol = 2, alpha_dots = 0.7)
```



## 5. The insight from the analysis

from the analysis there's no any a normally detected in the dataset from the month of January to April.

4. Challenge the solution.

There was no model built, we were just detecting the anomaly in the dataset.

5. Follow up question.

a). Do we have the right data?

Yes, the data was appropriate

b). Do we need another data?

No, the data was appropriate.

c).Do we have the right question?

Yes, the question is clear.