Association

Daniel Ryba Zanardini de Oliveira

Association utilities

Why?

In retail the most of purchases are bought on impulse. Some analysis provides clues as to what a customer might have bought for "impulse". This information can be used to decide the location and promotion of goods within a store.

If a rule is found in some store and it is not observed in another (equal stores), it is a point of attention. It may be a store that has different customers or that the organization of the products is proposing a "more profitable" way. This type of research can generate profitable insights.

It is important to realize that there are many areas in which it can be applied: Analysis of credit card purchases. Analysis of telephone calling patterns. Identification of fraudulent medical insurance claims. Analysis of telecom service purchases and others.

Introduction

In this report I use rules of association to analize a market basket.

I will use an Apriori Algorithm to do Market Basket Analysis of Customers purchasing behaviours. The goal is to predict what the customer will buy by looking at the products he is buying.

An Eclat algorithm will be used to provide a list of the most often bought together.

Data set: Market Basket Optimization (Data Mining for Grocery Stores) - https://www.kaggle.com/roshansharma/market-basket-optimization/data

Data set

```
# set directory
setwd("C:\\Users\\Daniel\\Documents\\BI - MASTER (MBA)\\DM\\AULA_05_ASSOCIACAO")

# load data
df <- read.csv('Market_Basket_Optimisation.csv', header = FALSE)
```

```
# look for data
str(df)
```

```
## 'data.frame':    7501 obs. of  20 variables:
##  $ V1 : chr  "shrimp" "burgers" "chutney" "turkey" ...
##  $ V2 : chr  "almonds" "meatballs" "" "avocado" ...
##  $ V3 : chr  "avocado" "eggs" "" "" ...
##  $ V4 : chr  "vegetables mix" "" "" "" ...
##  $ V5 : chr  "green grapes" "" "" "" ...
##  $ V6 : chr  "whole weat flour" "" "" "" ...
```

```
##  $ V7 : chr  "yams" "" "" "" ...
##  $ V8 : chr  "cottage cheese" "" "" "" ...
##  $ V9 : chr  "energy drink" "" "" "" ...
##  $ V10: chr  "tomato juice" "" "" "" ...
##  $ V11: chr  "low fat yogurt" "" "" "" ...
##  $ V12: chr  "green tea" "" "" "" ...
##  $ V13: chr  "honey" "" "" "" ...
##  $ V14: chr  "salad" "" "" "" ...
##  $ V15: chr  "mineral water" "" "" "" ...
##  $ V16: chr  "salmon" "" "" "" ...
##  $ V17: chr  "antioxydant juice" "" "" "" ...
##  $ V18: chr  "frozen smoothie" "" "" "" ...
##  $ V19: chr  "spinach" "" "" "" ...
##  $ V20: chr  "olive oil" "" "" "" ...
```

Apriori

The Apriori algorithm was proposed by Agrawal and Srikant in 1994. Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation or IP addresses). Each transaction is seen as a set of items (an itemset). Given a threshold x, the Apriori algorithm identifies the item sets which are subsets of at least x transactions in the database.

In another words, frequent itemsets are sets of items that appear together at least less than $x\%$ of transactions. The number x witch needs to be provided to the algorithm is called support data.

For example, the rule of association: A $\Rightarrow$ B where A, B, C and D are items. The confidence (c) is the proportion of transactions that include A and also include B. So if the rule confidence is 60%, it can be read as 60% of the people who bought A also bought B.

$confidence(A \Rightarrow B) = \frac{suport(AB)}{suport(A)}$

Example

	coffee	no coffee	total
tea	150	50	200
no tea	650	150	800
total	800	200	1000

- the rule $tea \Rightarrow coffe$ have support(s) $s = \frac{(150)}{(1000)} = 0.15$
- the confidence of this rule is $c = \frac{(0.15)}{(0.2)} = 0.75$
- but the rule is NOT interesting because the rule $\{\} \Rightarrow coffee$ have a confidence 80% (80% of people already drink coffee). The fact that someone drinks tea decreases the probability of drinking coffee!

Lift measures how much the two sides of the rule are not independent.

$lift(A \Rightarrow B) = \frac{P(AB)}{P(A)P(B)}$

- lift = 1 indicate that A and B are independent (bad).
- lift > 1 indicate a positive correlation between A and B.

Back to our study

```
# install.packages('arules')
library(arules)
```

```r
# numbers of digits
options(digits = 5)

# sparse matrix
dataset = read.transactions('Market_Basket_Optimisation.csv', sep = ',',
                            rm.duplicates = TRUE) #removes duplicate entries in each transaction
```

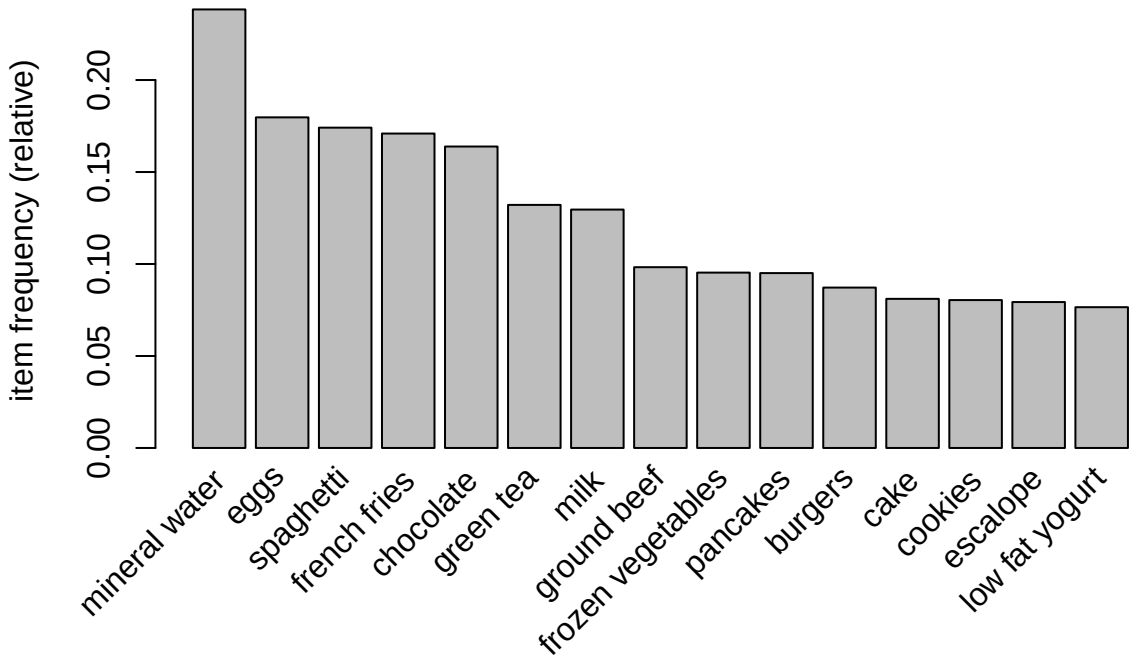
```
## distribution of transactions with duplicates:
## 1
## 5
```

```r
summary(dataset) # matrix details
```

```
## transactions as itemMatrix in sparse format with
##  7501 rows (elements/itemsets/transactions) and
##  119 columns (items) and a density of 0.03289
##
## most frequent items:
## mineral water          eggs     spaghetti  french fries     chocolate
##          1788          1348          1306          1282          1229
##       (Other)
##         22405
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 1754 1358 1044  816  667  493  391  324  259  139  102   67   40   22   17    4
##   18   19   20
##    1    2    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    2.00    3.00    3.91    5.00   20.00
##
## includes extended item information - examples:
##              labels
## 1           almonds
## 2 antioxydant juice
## 3         asparagus
```

Now print the 15 most frequent products in the market basket.

```r
itemFrequencyPlot(dataset, topN = 15) #topN (choose your number)
```



Create the rules indicating the support and the confidence,

- Support - optimize the sale of products purchased 3 times a week $\frac{(3*7)}{(7501)} = 0.003$.
- Confidence - Low: rules that don't make sense; High: obvious rules. *ATTENTION*, a value of 0.8 means that all rules generated must be correct in 80% of transactions

I put a confidence of 0.2 but we must do several tests and understand the "business" questions.

```
# create rules
rules = apriori(data = dataset, parameter = list(support = 0.003, confidence = 0.2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.2    0.1    1 none FALSE            TRUE       5   0.003      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: 22
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.00s].
## sorting and recoding items ... [115 item(s)] done [0.00s].
```

```
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 done [0.00s].
## writing ... [1348 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

Visualization 15 rules created, ordered by lift.

```
library(knitr)
library(kableExtra)

# inspection rules
wide <- inspect(sort(rules, by = 'lift')[1:15])
```

```
# visualization
knitr::kable(wide, "markdown", booktabs = T) %>%
kable_styling(font_size = 4)
```

	lhs		rhs	support	confidence	coverage	lift	count
[1]	{mineral water,whole wheat pasta}	=>	{olive oil}	0.00387	0.40278	0.00960	6.1159	29
[2]	{frozen vegetables,milk,mineral water}	=>	{soup}	0.00307	0.27711	0.01107	5.4844	23
[3]	{fromage blanc}	=>	{honey}	0.00333	0.24510	0.01360	5.1643	25
[4]	{spaghetti,tomato sauce}	=>	{ground beef}	0.00307	0.48936	0.00627	4.9806	23
[5]	{light cream}	=>	{chicken}	0.00453	0.29060	0.01560	4.8440	34
[6]	{pasta}	=>	{escalope}	0.00587	0.37288	0.01573	4.7008	44
[7]	{french fries,herb & pepper}	=>	{ground beef}	0.00320	0.46154	0.00693	4.6974	24
[8]	{cereals,spaghetti}	=>	{ground beef}	0.00307	0.46000	0.00667	4.6818	23
[9]	{frozen vegetables,mineral water,soup}	=>	{milk}	0.00307	0.60526	0.00507	4.6709	23
[10]	{french fries,ground beef}	=>	{herb & pepper}	0.00320	0.23077	0.01386	4.6658	24
[11]	{chocolate,frozen vegetables,mineral water}	=>	{shrimp}	0.00320	0.32877	0.00973	4.6009	24
[12]	{frozen vegetables,milk,mineral water}	=>	{olive oil}	0.00333	0.30120	0.01107	4.5736	25
[13]	{pasta}	=>	{shrimp}	0.00507	0.32203	0.01573	4.5067	38
[14]	{chocolate,herb & pepper}	=>	{ground beef}	0.00400	0.44118	0.00907	4.4902	30
[15]	{chocolate,mineral water,shrimp}	=>	{frozen vegetables}	0.00320	0.42105	0.00760	4.4172	24

In this table we can check the rules found. The rules are described, for example: *mineral water, whole wheat pasta* $\Rightarrow$ *olive oil* (the support, the confidence, the lift, coverage and the number of times that appears in the base).

Another way to help the analysis is utilize an algorithm that shows the frequency of which products are purchased together.

ECLAT

Eclat(alt. ECLAT, stands for Equivalence Class Transformation) is a depth-first search algorithm based on set intersection. Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.

```r
# create rules of frequency
rules2 = eclat(data = dataset, parameter = list(support = 0.003, minlen = 2))
```

```
## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen            target  ext
##     FALSE   0.003      2     10 frequent itemsets TRUE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 22
##
## create itemset ...
## set transactions ...[119 item(s), 7501 transaction(s)] done [0.00s].
## sorting and recoding items ... [115 item(s)] done [0.00s].
## creating sparse bit matrix ... [115 row(s), 7501 column(s)] done [0.00s].
## writing  ... [1328 set(s)] done [0.01s].
## Creating S4 object  ... done [0.00s].
```

```r
# minlen = minimum number of products in the group
```

Visualization 15 rules created, ordered by support.

```r
# visualization
knitr::kable(wide2, "markdown", booktabs = T) %>%
kable_styling(font_size = 4)
```

	items	support	transIdenticalToItemsets	count
[1]	{mineral water,spaghetti}	0.05973	448	448
[2]	{chocolate,mineral water}	0.05266	395	395
[3]	{eggs,mineral water}	0.05093	382	382
[4]	{milk,mineral water}	0.04799	360	360
[5]	{ground beef,mineral water}	0.04093	307	307
[6]	{ground beef,spaghetti}	0.03919	294	294
[7]	{chocolate,spaghetti}	0.03919	294	294
[8]	{eggs,spaghetti}	0.03653	274	274
[9]	{eggs,french fries}	0.03640	273	273
[10]	{frozen vegetables,mineral water}	0.03573	268	268
[11]	{milk,spaghetti}	0.03546	266	266
[12]	{chocolate,french fries}	0.03440	258	258
[13]	{mineral water,pancakes}	0.03373	253	253
[14]	{french fries,mineral water}	0.03373	253	253
[15]	{chocolate,eggs}	0.03320	249	249

In this table we can check the associations found. We can see the most purchased products together.

Conclusion

These analyzes are quite interesting, since with them you can evaluate the operational organization, marketing, among others. Understanding the business problems we can, with help of this information, optimize the

business profit making some improvements.

In this example, we can see products that are purchased according to predetermined rules and products that are frequently purchased together.