Improving the Customers' In-Store Experience using Apriori Algorithm

Ioana DAVID
Department of Economic Informatics and Cybernetics
Bucharest University of Economic Studies
ioana.david4@yahoo.com

The 21st century is the era of technology and digital development. That's the reason why mobile applications nowadays became an important support for businesses and a significant part of our daily activities. The usage of smart devices and the ease of access to technology lead to obvious changes in consumer behavior. Therefore, as it has been remarked a decrease of in-store shopping, improving the shopping experience in traditional stores has become of high interest for many retailers. In this paper, we propose a mobile application which helps people to optimize the time they spend inside of hypermarkets and which suggests an optimal placement for aisles in a store.
Keywords: *in-store experience, aisles placement, mobile application development*

1 Introduction

Grocery shopping is a continuous activity in everyone's life. One of the effects of technological evolution is the desire to optimize daily activities by using mobile phones. Thereby, online shopping becomes more popular than in-store shopping. As online shopping is focused mainly on non-perishable products, it gives room to traditional stores to maintain or even grow their businesses. Due to the wide range of products, hypermarkets allow customers to buy everything that can satisfy their routine shopping needs.

The process of buying products for the current consumption became a challenge both for the clients and for the stores. The time spent searching products is the most important problem that people take into consideration when deciding to visit a hypermarket. On the opposite side, retail stores are preoccupied with strategies to optimize products positioning in order to improve sales and revenue.

Providing a large variety of grocery and household products, hypermarkets accentuate in-store environmental stimuli, such as shelf-space allocation and product display. These strategies can lead to positive effects like maximizing profit, improving stock control or improving customer satisfaction. [1] The problem of deciding how to stock products among the aisles of a store can be resolved by extracting valuable information from the store's transactions history. Different analytical tools and algorithms help companies to study cutomer purchasing behaviour.

A well-known technique to understand the way customers make decisions is the *Association rule learning* which helps to discover relationships between items bought toghether. This is an algorithm that helps companies to study and to identify purchasing patterns that can be used in order to establish certain marketing actions or strategies. [2]

The application proposed in this article combines companies' necessity to position products in the most effective way and clients' interest in optimizing the time spent inside hypermarket. For these reasons, the application has a preliminary module in which an algorithm is applied to place products based on antecedent transactions. The main module consists in a mobile application with a user-friendly interface, where people can view details about many hypermarkets or can organize their future shopping sessions.

2 Market Basket Analysis

The concept of *affinity analysis* is a data analysis and data mining principle, which

presents the coexistence relationships between different entities. It is used also in the retail industry, where it is known as *Market Basket Analysis*, which extracts rules that associate products based on the frequency of purchasing them together in sets of different sizes. When using the term of *market basket*, we only refer to item sets purchased by a customer during a single shopping session. The process of frequent item sets analysis begins from the premise that no consumer takes isolated decisions, so he rarely buys a single item and he always inclines to buy numerous products from distinct categories. Consequently, analysts' purpose is to discover the products which lead certainly to buying other products. Revealing this information allows managers to develop methods of influencing the shopping behavior, increasing the demand for some products through promotional offers. In addition, companies focus on optimizing prices of products in order to achieve higher cross-selling rates and on high-margin products, because these are the main prices that must be adapted to purchasing patterns. [2]

Depending on a high volume of data which involves all the transactions realized in a hypermarket, market basket analysis can be considered a difficult operation which is based on the analysis of data sets which exceed the processing capacity of traditional software. This challenge is correlated to the concept of *big data*, which regards the predictive analysis, the human behavior analysis and extraction of information from high amounts of data.

The systematization of the extracted data from a hypermarket's transactions and the analysis of sold products, both from the point of view of the associations and from the point of view of prices and quantities, help in the process of product placement. Once the whole analysis is completed, a hierarchy f~~or~~ of the aisles from a store department can be established in order to join products with a high affinity probability.

The most used algorithm for estimating associations is the *Apriori algorithm* proposed in 1994 by three researchers who defined the problem of market basket analysis: Rakesh Agrawal, Tomasz Imieliński and Arun Swami. [3] This algorithm returns the associated itemsets using a high complexity in time and space, because it searches all the possible associations (2^n - 1).

3 Application Design

The application fulfills two main goals, one reffering to an optimal layout for hypermarkets' aisles and the other one to minimize the time spent by customers at shopping, whithout decreasing the company's profit.

First of all, the application suggests a specific placement for all the aisles from a store and identifies each aisle from the system with the position indicated after applying the Apriori algorithm. At this step, user interaction with the application is not necessary, because the store organizing process takes place before the application ~~to~~ be available.

Second of all, the application is ready to be used by hypermarkets' customers during theirs shopping sessions. Users have access to many functionalities designed to improve every shopping experience. Among these it can be mentioned the possibility to set and modify a favourite store on the basis of which are developed all the future actions, such as an interactive map with the hypermarkets' layout, a list with all the available products, a search engine to find a particular product or a particular category of products, a module to view all the information regarding a specific hypermarket and an option to locate a store using Google Maps. In addition, users can define their own shopping list which can always be modified and which can contain products from all the stores available on this platform or they can add products to a list with their favorite products. To ensure the

transparency of all prices and sales amounts during the last year, the application provides a module named *Statistics* which presents different graphs and hierarchies based on users' filters.

A user's interaction with the system can be visually represented in a use case diagram which helps to create a technical and functional perspective over the entire application. Thus, the diagram illustrated in Figure 1 presents the system's functionalities.

Fig. 1. Use case diagram

In the next phase - database design - the entities needed to implement the desired functionalities were identified: *Hypermarket, Department, Aisle, Category, Product, Receipt, Product History, Favorites List* and *Shopping List.*

Based on the relationships between entities, the structure of the database can be defined. Thereby, between these entities there are different kinds of relationships and cardinalities. For example, this model contains multiple one-to-many relationships, but there are also two many-to-many relationships. Whereas relational database paradigm does not accept the existence of many-to-many logical connections, it is necessary to transform this relationship into an additional table. This design pattern provides a consistent database model for future data operations and ensures the clarity of the system. The many-to-many relationships identified inside of this model are those between *Aisles* and *Products*, respectively the one between *Products* and *Receipts.*

Regarding *Favorites List* and *Shopping List* entities, it is not necessary to store the corresponding data inside the general database, because these contain information about a specific user, individualized by his own smartphone. Thus, data about favorite products and the shopping list will be stored locally on user's device.

The final database structure is shown in Figure 2, which represents the database schema, including all the attributes.

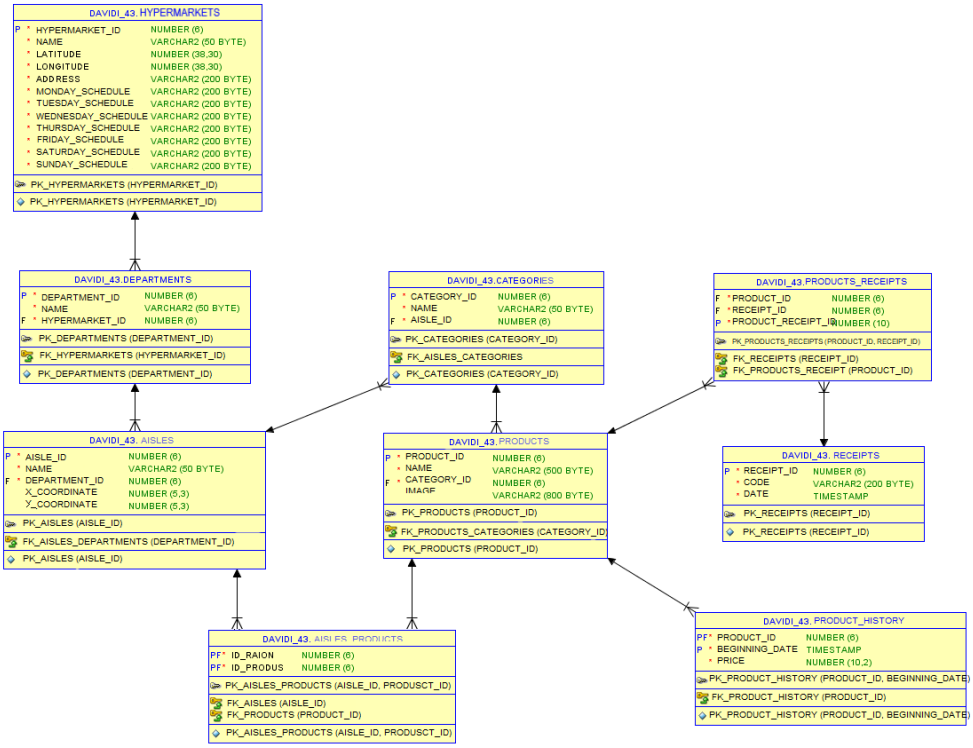

Fig. 2. Database schema

4 Software technologies

The development of this application is based on several technologies, such as Python language, Flask micro-framework, Amazon Web Services Cloud9 integrated development environment, SQL and PL/SQL languages, Java language and Android Studio integrated development environment.

Python is an open-source programming language, distinguished by its simplicity and its applicability in many areas. [4] That represents one of the reasons why Python was used both in data scraping and in developing the application's server. In order to perform the in-store experience analysis and optimization, the development of this application included the process of collecting data from a hypermarket's website. For Web scraping two Python libraries were used: BeautifulSoup and Selenium-Webdriver, which allow to automatically parse the data in HTML containers into Comma Separated Values files, which represent an intermediate stage before inserting it into an Oracle database. The functions used to extract data implied the launching of a virtual browser from which the elements from the Document Object Model (DOM) can be loaded. The DOM represents an interface which exposes an XML document as an arborescent structure. Python was also used to insert data into an Oracle database through the cx_Oracle library, which allows to establish a database connection and to execute queries with the help of a cursor.

Flask is a Python micro-framework used to develop the application's backend. It provides the possibility to develop a REST web service, with an obvious distinction between the client and the server. Through the web services developed in the REST API, create, read, update and delete operations (CRUD) can be executed using the four main standard methods of the HTTP protocol: *POST*, *GET*, *PUT* and *DELETE*. The server represents the middleware between the mobile application and the database, and the data is passed between the endpoints in JSON format.

Amazon Web Services Cloud9 is an integrated development environment in the cloud which enables the writing, compiling and debugging of programs within a web

browser. [5] It supports the Python language and provides access to a terminal. The API server was developed using this platform because it provides an external IP which is needed to be able to access to the web services from anywhere.

SQL and PL/SQL languages were used through the Oracle SQL Developer tool and helped to transform the database structure defined in the analysis process into a relational database. The high volume of data existent for storing all the products from a hypermarket leads to the necessity to model an efficient method for accessing and processing. The creation of the database was done using SQL's Data Definition Language to define and describe the nine tables and the relationships between them. During the development and the use of the mobile application, the SQL's Data Manipulation Language was used to insert data or query the tables. Additionally, the PL/SQL language was used to define sequences and triggers.

The application was designed for devices running the Android Operating System because it is the most popular operating system for mobile devices, covering 88% of the mobile devices as of the end of 2018. [6] From the point of view of mobile application development, Android is easier than other mobile operating systems, because it provides open-source tools and it uses the Java language.

The Java programming language is recommended in developing Android applications due to the variety of API tools and libraries which are available. For the development of the application the Java language and the Android Studio IDE were used thanks to the possibility to use multiple interactive instruments and to the intuitive development platform. Although the standard Java and Android libraries provide a lot of classes and methods, such as data structures, visual components and data processing methods, the use of some external open-source libraries extends the possibility to implement and simplify the effort to build an application. The libraries and APIs that were used are: OkHttp, Picasso, MPAndroidChart, ClickableAreasImages, ArcLayout and Google Maps API.

5 Input data depiction

Containing the products sold by hypermarkets, input data is relevant also for the analysis process and for the final solution. Being impossible to access a retail store database, the solution was to collect data from a hypermarket's web site. For this purpose, the chosen hypermarket was Carrefour, because its web site presents the entire architecture of the store, from departments to product categories allocated to an aisle. Thus, the three hypermarkets that can be found in this application have the same organizational structure as Carrefour.

Using the Python programming language and the two mentioned libraries the data in each page of the store was extracted. The *Selenium* library provides a *web driver* which allows duplicating a web page to parse data. Two distinct situations of data parsing were found: one with crossing data in a single page and one with crossing multiple pages. The second situation takes place when the chosen category has more than one page with products. It was necessary to separate these cases because their URLs do not observe the same rule: in the first case the URL contains only identifying names, while the URL from the second case has the number of the current page appended at the end. Because web sites implement some security restrictions, it was required to apply a function which enforces the scraping method to wait some seconds before the next processing, in order not to overcharge the site and to block the access. After this step, the function *findAll* from BeautifulSoup library authorizes the access to the HTML structure of the page. This way, parsing data is realized navigating through the tree structure of the page, choosing the useful data from different element types as: *class, div* or *src* and

storing data temporarily into *.csv* files. The method of accessing the entire set of products from a page is presented below and it is based on selecting data from an item list whose class attribute is named "product isProduct product-item ng-scope".

```
url = link
browser = webdriver.Chrome
                    ('chromedriver')
browser.implicitly_wait(60)
browser.get(url)
html = browser.page_source
soup = BeautifulSoup(html,
                    'html.parser')
products = soup.findAll('li',
        attrs={'class':'product
        isProduct product-item
        ng-scope'})
```

Another important phase of creating input data consisted in the process of adding records to the database. Mostly it was fulfilled using the Python language to insert the data from the .csv files into an Oracle database through cx_Oracle connection and its cursor. To automate the process of parsing products the *os* module - which allows the communication with the operating system of the computer - and the *glob* module - which finds all the files' paths that correspond to *.csv* extension - can be used.

```
path = 'C:/Users/user/Desktop/licenta
        /Python/Scraping/Carrefour'
extension = 'csv'
os.chdir(path)
files = glob.glob('*.{}'
        .format(extension))
```

In the same function that inserts products into the database, prices variations are also defined by inserting random historical changes with an oscillation of 1 to 4 percent at an interval of 15 days. The chosen probability for a price to be changed is 30% and it is assured by generating a random number between 1 and 100.

Another essential function generates fictive shopping receipts for a period longer than one year, with a generating frequency of 30 minutes. This function takes into consideration the hour on which a transaction is done, ensuring that it doesn't take place outside the time slot 8-22. The selection of products included on a receipt is also randomly performed, using a random number of products and selecting random IDs of products.

6 Algorithm implementation

Aisles placement inside hypermarkets is accomplished relying on the transactions previously performed and applying the Apriori algorithm (as in Figure 3). The volume of the analyzed receipts depends on the system's performance, because the running time of the Apriori function is directly linked to its high complexity of $O(2^n)$.

To organize the aisles, it is necessary to know the available positions for each department before applying the algorithm. The coordinates of the available positions are considered input data for the algorithm, along with the ID of the current department and the maps dimensions.

The positioning function itself (Figure 4) calls another function which creates the array of tuples with aisles associations after extracting all the transactions and after applying the *apriori* function from *Mlxtend.frequent_patterns* library. To establish the coordinates for each aisle, it is defined an array with the aisles sorted descending from the ones with the strongest affinity to the ones with the weakest affinity. So, the main purpose is to associate aisles with an as high as possible association relationship. On the opposite side, the coordinates array is sorted first using the coordinates from the Ox axis of the map and secondly using the coordinates from the Oy axis. Finally, using the two arrays, the coordinates of an array position correspond to the aisle at the same position of the

second array. The values added into the database are estimated as percentages of the map by dividing each coordinate to the corresponding map dimension.

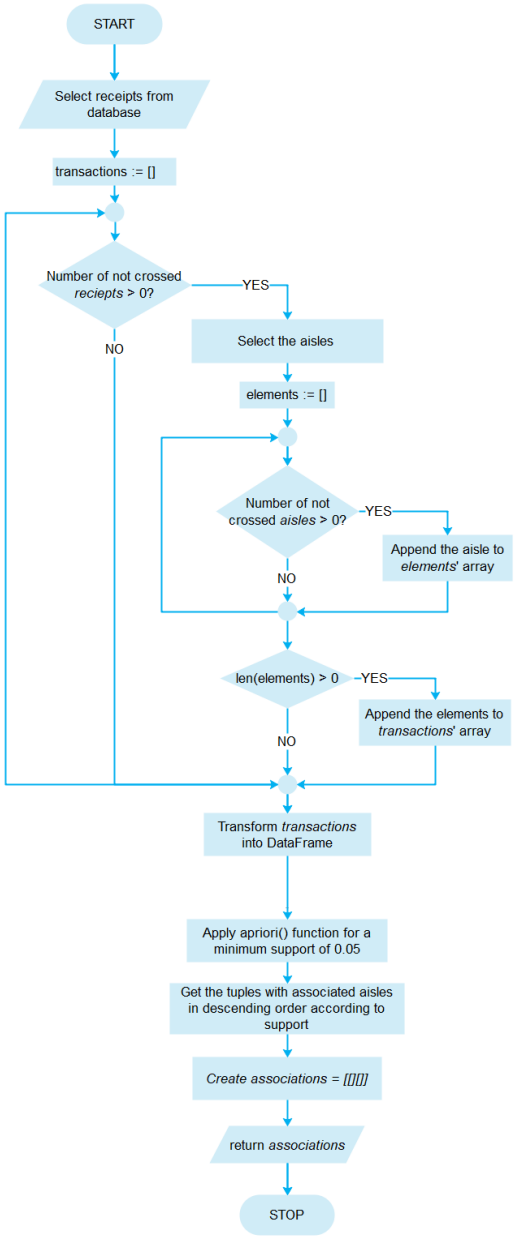

Fig. 3. Logical Schema for applying the Apriori function

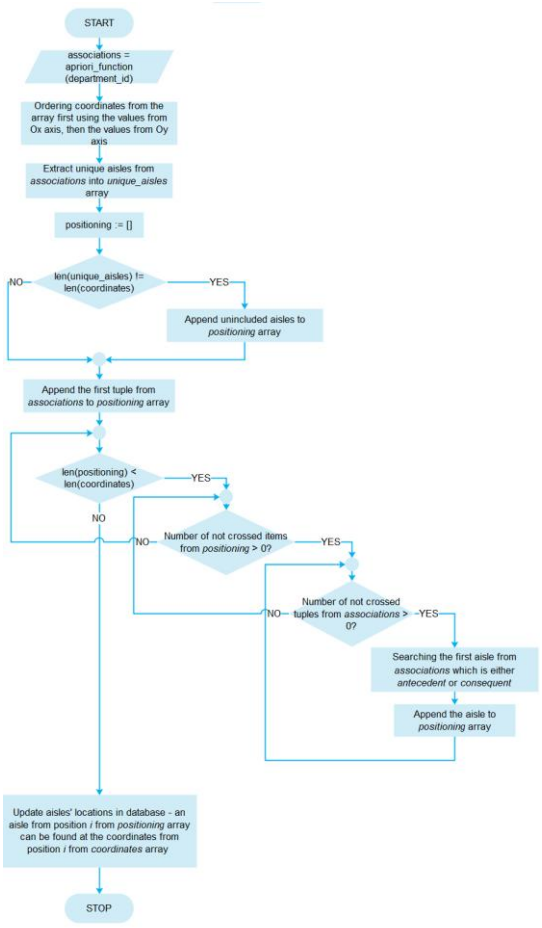

Fig. 4. Logical Schema for positioning function

7 Application interface description

The mobile application starts with the home screen that presents a series of six buttons circularly arranged and a menu at the bottom of the screen which is kept in all the screens (as in Figure 5). The six buttons have the role to link the application main functionalities to the corresponding application screens. The menu allows the users to access four essential screens from every point of the application. In the upper part of the screen it can be remarked an information icon which redirects to the application's description.

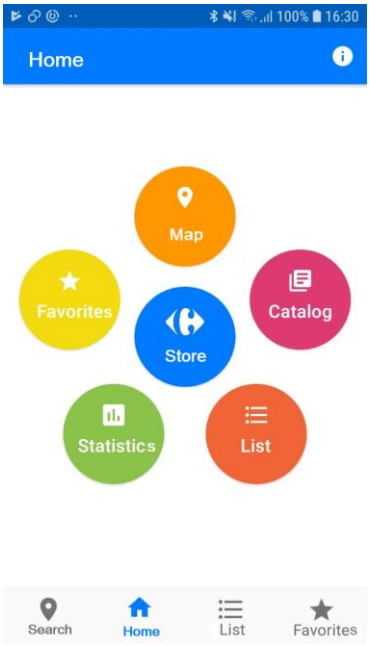

Fig. 5. Home screen

Whereas the application's main idea is to set a favorite hypermarket and to perform some actions based on it, a useful screen is the one that is opened after selecting the *Store* option from the home screen. This screen presents information about the favorite store and contains two buttons designed to locate the store on Google Maps and to change the favorite store as in Figure 6.

Fig. 6. Store

In addition, the users can access the *Map* section where they can navigate on the map of the favorite hypermarket. Each area can be selected and after that a message will be displayed at the bottom of the screen to indicate the department.

Accesing the *Catalog* screen gives the possibility to observe all the available products from a specific category selected by the user. The filter is applied after the gradual selection in three different screens which illustrate departments, aisles and categories. Each product from can be added either to the shopping list by selecting the associated button or to the favorites list by activating the star icon. At the same time, by selecting a product, it will be located on the hypermarket's map.

The screen named *Shopping list* allows users to keep a register of the products they need to buy during the next shopping session. In this screen, the user can see products from all the stores, not only from the favorite one, by expanding the list from the left side. The list calculates the total payment taking into consideration the quantities from each product. During the shopping session, the user can check the products already put in the basket, in order to be removed from the list. Furthermore, to optimize the time spent to search products in the hypermarket, the user can locate every product just by clicking on it (as in Figure 7).

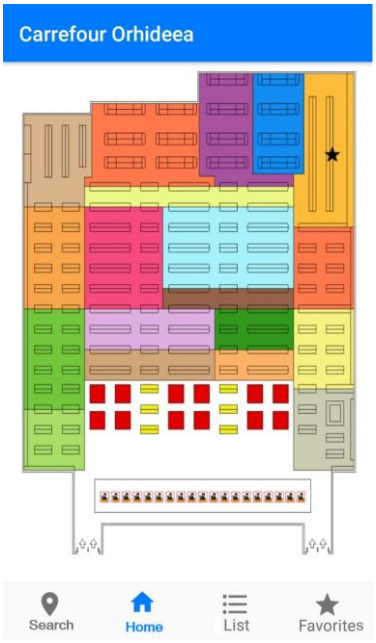

Fig. 7 Locate a product

Fig. 8 Statistics

The Favorites section offers the possibility to view the products selected as favorites to ease the process of adding product to the shopping list. The products can be eliminated from the list by disabling the star icon or added to the shopping list.

To search as fast as possible for a specified product from the favorite store, users have access to a search engine that returns the products which correspond to their requirements.

Another functionality is included in the *Statistics* screen (as in Figure 8) where prices and sales evolutions are presented in the form of a bar graph, a line graph and a hierarchy. When accessing this screen, the standard statistics are referring to all the products from the store, but users can personalize the results by either just mentioning the aisle or mentioning both the aisle and the category. Following that, the bar graph illustrates the amounts of sales in the recent 12 months and the line graph illustrates prices evolution in the last 12 months. The hierarchy presents the top three cheapest products from the selected category or the three most sold products.

8 Conclusions

The developed application represents a system which incorporates facilities for all the parts involved in a buy and sale process. Its main purposes are to increase retail stores' sales and to improve customers' in-store experience.

The application can be further extended. The main two directions in which the application can be improved are: adding an application module to be accessed by companies to refresh the database and implementing a complex process of positioning products inside aisles, taking into consideration the prices, the profit margin and other marketing strategies.

References

[1] G. Aloysius, "An approach to products placement in supermarkets using PrefixSpan algorithm," *Journal of King Saud University - Computer and Information Science*, vol. 25, pp. 77-87, 2013.

[2] M. Kouzis-Loukas, "Analysing Customer Baskets - A Business-to-Business Case Study," Erasmus School of Economics, Rotterdam, 2014.

[3] "Apriori algorithm," Wikipedia,

[Online]. Available: https://en.wikipedia.org/wiki/Apriori_algorithm..

[4] "Python Website," Python Software Foundation, [Online]. Available: https://www.python.org.

[5] "Amazon Web Services Cloud9," Amazon Web Services, Inc., [Online]. Available: https://aws.amazon.com/cloud9.

[6] "Global Market Share Held by Smartphone Operating Systems," [Online]. Available: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems.

Ioana DAVID (b. February 4, 1998) has graduated the Faculty of Cybernetics, Statistics and Economic Informatics of the Bucharest University of Economic Studies with a bachelor's degree in 2019. She studied Mathematics and Computer Science at the National College of Computer Science "Tudor Vianu" from Bucharest.