Algorithms for Big Data - Recommendation System

Francisco Barros 82608, Francisco Caldeira 79191, Nuno Pires 96073

May 23, 2021

1 Introduction

A recommendation system combines several computational techniques to select personalized items based on the interests of users and conforms to the context in which they are inserted. Items can be any product, such as movies, books, music, videos, electronic devices, advertisements, etc. Companies like Google, Youtube, HBO are recognized for their intensive use of recommendation systems to keep customers using their products, either by suggesting purchases to make or content to use. This kind of systems gives these companies a huge competitive advantage.

Recommendation systems arose in response to people's difficulty in choosing a wide variety of products and services. These systems have been evolving and nowadays they present a complexity and capacity to make recommendations better than the human recommendations themselves. Recommendation systems ingest ever increasingly large datasets in order to make the assumptions, and thus posing a difficulty of the analysis.

Several approaches to recommendation algorithms exist, from predicting user ratings to predicting the probality of different sets of occurring together.

A recommendation algorithm will try to predict a given user response to a certain item(a news article, a movie seen, ...) based of the history of user, after predicting the rating the ones rated higher are expected to be of the like of the user.

It makes predictions (filtering) about the interests of users and collects the preferences of the various users (collaboration). For example, if user A has the same opinion as user B about one product then it is more likely that they share the same opinion in other products than user A sharing the same opinion with a random user. The similarity between two items is determined by the interest / rating given by the users.

This report will analyze the following dataset[1] collected by Open CDP from an eCommerce store with more than 15 million visitors per month, from October 2020 to November 2020, it features data from items viewed, purchased and placed in the shopping cart.

2 Dataset

The dataset rows represent an event by a given user, either view, purchasing or changing the items in the cart. The dataset was already anonymized which means the personal data was removed.

The dataset is structured with 9 columns:

Property	**Type**	**Description**
event_time	String	Timestamp of event in UTC
event_type	String	Event type
product_id	Number	ID of a product
category_id	Number	Product category ID
category_code	String	Product category code name
brand	String	Brand name
price	Number	Price of a product, unknown currency.
user_id	Number	User ID.
user_session	String	Temporary session ID, changed every time a user accesses the store

Event types can be:

- **view** - a user viewed a product.
- **cart** - a user added/removed a product to the shopping cart.
- **purchase** - a user purchased a product.

[1] `https://www.kaggle.com/mkechinov/ecommerce-behavior-data-from-multi-category-store`

An example of a row

Property	Value
event_time	2019-10-01 00:00:00 UTC
event_type	view
product_id	3900821
category_id	2053013552326770905
category_code	appliances.environment.water_heater
brand	aqua
price	33.20
user_id	554748717
user_session	9333dfbd-b87a-4708-9857-6336556b0fcc

3 Data analysis

Before executing the algorithm, we performed a detailed analysis of the dataset. Each dataset line is an event specific to an eCommerce store. Altogether we have 109 950 743 records. The first step was to identify if there were any null fields. For this, we built a table with a single line, which shows the field count to null for each column, as shown in the table below.

category_code	brand	user_session
35 413 780	15 341 158	12

We found many records with **brand** and **category_code** at null and 12 records with **user_session** to null. We decided to remove these lines and not consider them for our recommendation system. With these changes, we are left with a dataset of 68 650 184 records. For running the algorithm we left the category code be null since we had the category id so this didn't impact the result of the algorithm.

3.1 Brand analysis

For the analysis of brands, the quantity of products purchased and viewed from the various brands was counted. Below are two graphs with this count:

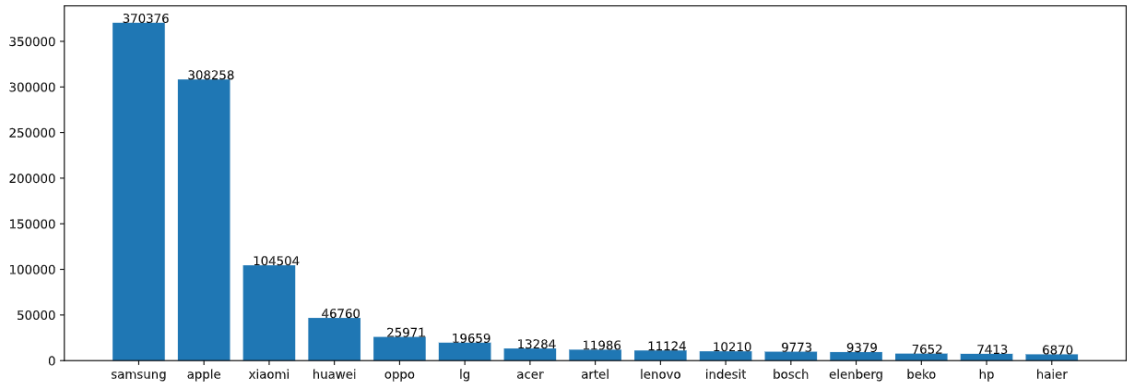

Figure 1: Count of brand products purchased.

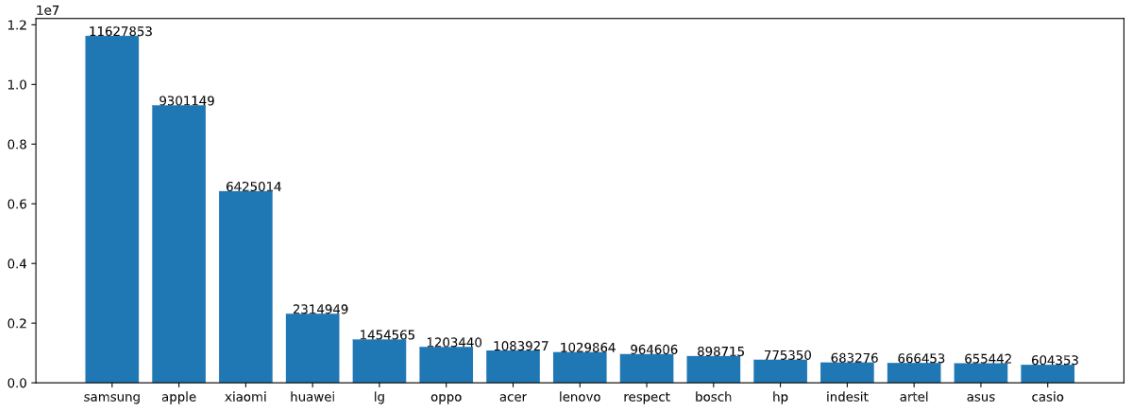

Figure 2: Count of brand products viewed.

As we can see, Samsung was the most viewed and purchased brand. Apple and Xiaomi are also well represented in this store. An interesting analysis that we can see now is the number of times on average a brand is seen before being purchased.

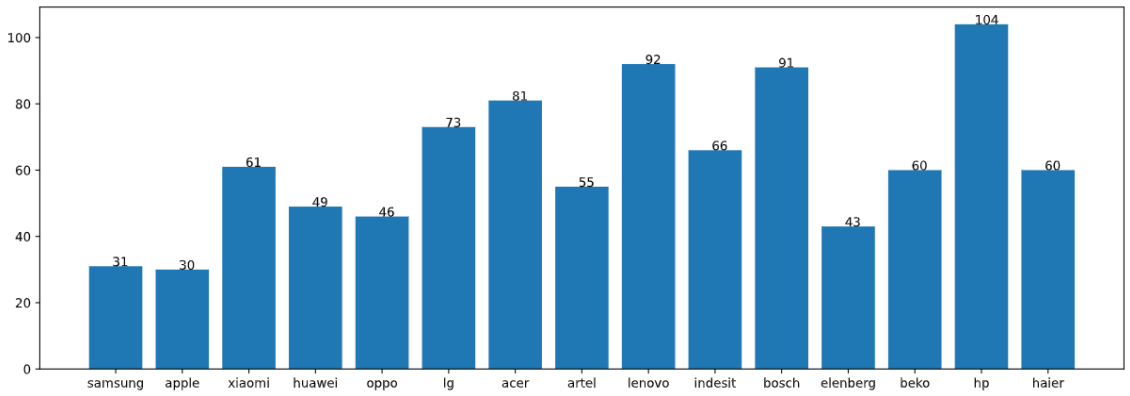

Figure 3: Number of times on average a brand is seen before being purchased.

We were able to observe that Samsung and Apple do not need to be seen very often to be purchased. On the other hand, HP needs to be seen, on average, 104 times before it is purchased.

3.2 Category analysis

For the analysis of categories, the number of products by category that were purchased and viewed was analyzed. Below are two graphs with this count:

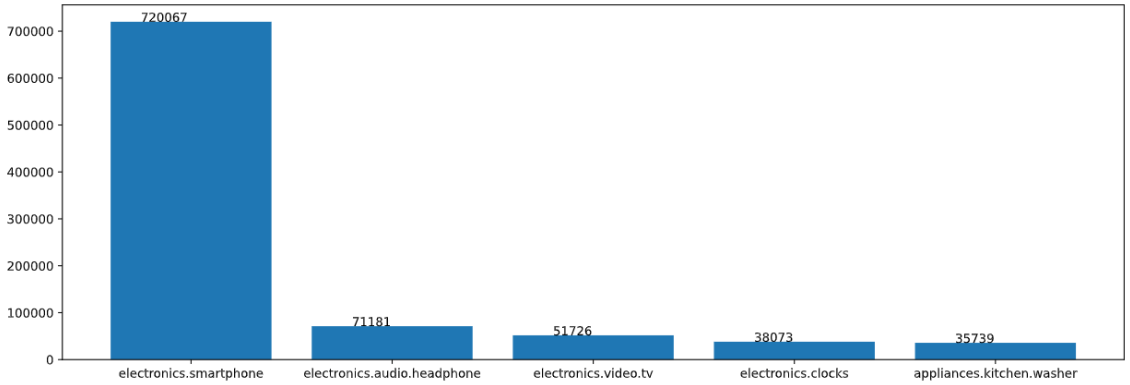

Figure 4: Count of category products purchased.

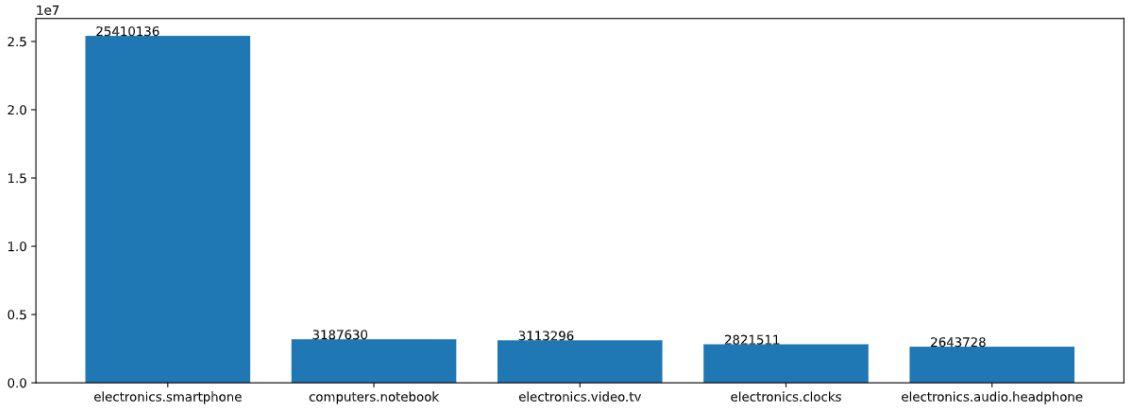

Figure 5: Count of category products viewed.

As we can see, smartphones was the most viewed and purchased category by far, followed by computers, TVs, headphones, clocks and kitchen appliances. An interesting analysis that we can see now is the number of times on average a category is seen before being purchased.

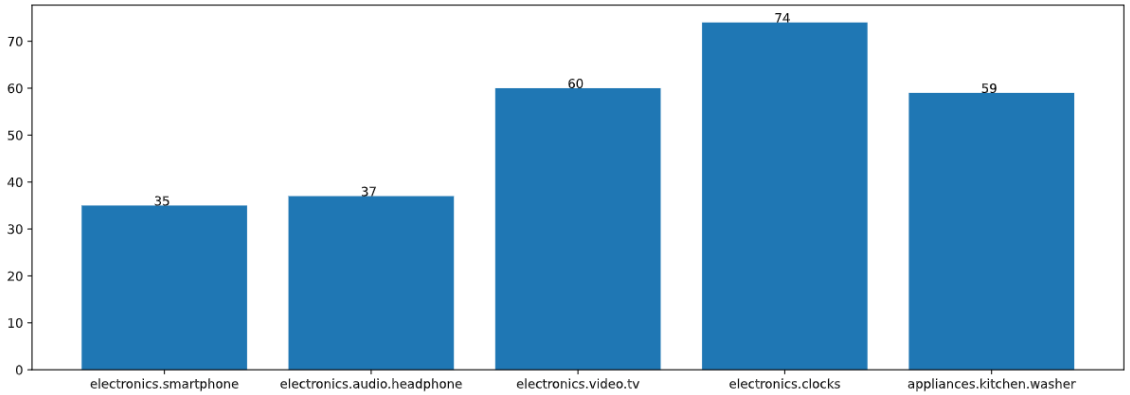

Figure 6: Number of times on average a category is seen before being purchased.

We were able to observe that smartphones and headphones do not need to be seen very often to be purchased. On the other hand, clocks need to be seen, on average, 74 times before it is purchased.

4 Data Preparation

To better handle the algorithm input data some changes had to be done to the dataset, some columns could be removed and a new one added.

First, three columns were excluded:

- **event_time** - The event timestamp was removed because it isn't a relevant data for the algorithm used.
- **category_code** - This column has null values and the dataset already provides a category_id column with a numeric value to represent the category. In order to avoid having redundant information, we decided to exclude it from our dataset.
- **price** - This data is not relevant for our recommendation system, since the algorithm doesn't take the price into consideration. The main focus is to establish a relationship between users and baskets of products / categories / brands.

The new column, named brand_category, was the concatenation of two others, the category id and brand. This was done to better understand the user preference when purchasing items of a given category and their brand of choice.

An example below:

```
+-------------------+--------+------------------------+
|category_id        |brand   |brand_category          |
+-------------------+--------+------------------------+
|2053013552326770905|aqua    |2053013552326770905.aqua|
+-------------------+--------+------------------------+
```

For this project we decided to just consider the events of *view* and *purchase* type since in this dataset the *cart* event type doesn't allow to distinguish between the items added to the cart and the removed ones. Then the dataset was split by event types mencioned above. Then to prepare the data for the algorithm several tables were created by grouping the dataset by user_id and session_id and joining the product_id, category_id, brand and the new column brand_category. This yielded 14 tables that served as an input for the algorithm. The tables ending with *_v* represent the views and the others the purchases. Each table was then saved into a file.

- **user_products** - user_id × product_id
- **user_products_v** - user_id × product_id
- **user_brands** - user_id × brand
- **user_brands_v** - user_id × brand
- **user_categories** - user_id × category_id
- **user_categories_v** - user_id × category_id
- **user_category_brand** - user_id × brand_category
- **user_category_brand_v** - user_id × brand_category
- **session_products** - session_id × brand_category
- **session_products_v** - session_id × brand_category
- **session_brands** - session_id × brand_category
- **session_brands_v** - session_id × brand_category
- **session_categories** - session_id × brand_category
- **session_categories_v** - session_id × brand_category
- **session_category_brand** - session_id × brand_category
- **session_category_brand_v** - session_id × brand_category

5 Algorithm

The chosen algorithm was FPGrowth, for extracting frequent itemsets and generating association rules. This algorithm has been used in favor of the Apriori algorithm seen in the lectures. The algorithm takes as parameters, the data to analyse, the minimum support and minimum confidence.

The minimum support is the threshold for a set of items to be considered frequent, this is used to get the frequent items from the data. The minimum confidence is the minimum confidence needed for a given association rule to be considered as important. We used values between 0.001 and 0.0001 for the minimum support and 0.20 for the minimum confidence in order to achieve results. For some of the tables a higher value of minimum support resulted in no association rules so we had to lower it until the algorithm returned some. A minimum confidence value less than 0.20 we think that it is to low to consider but in order to get results out of the session_products table we had to lower this value.

After generating the frequent items the algorithm calculates the association rules for them. The association rules then give us the consequent (item the user is likely to add to his basket) based on the antecedent (items that the user already has), also it gives us two variable results, confidence and lift.

The confidence is the probability of seeing item B on a basket when it already contains A. A confidence value of 1 means that all baskets that have item A also have item B.

The lift is a metric that reflects the increase in probability of having item B in basket if the user has already item A.

A pair of antecedent and consequent with high values of confidence and lift means that we can recommend item B to all users that already have A and they will most likely add it to the basket (in this case buy/view).

6 Results

The output of the algorithm for the table **session_products** showed only 1 row inspite of the many rows the original table had. This is because people tend to only buy one item at most per session. As we can see below the confidence value of the most frequent basket is really low so we think this should not be considered, we just wanted to show the kind of values we got from this table.

```
+----------+----------+-----------------+------------------+
|antecedent|consequent|confidence       |lift              |
+----------+----------+-----------------+------------------+
|[1004209] |[1004856] |0.067280163599182|1.5578528839942554|
+----------+----------+-----------------+------------------+
```

Below we can see the top 5 results of the algorithm for the **user_category_brand** table but instead of seeing the id we replaced it with the category code just for readability, this also applies to the tables to come. In the results' table we can see people tend to buy various smartphone brands and people who buy apple products and a smartphone also have a high probability of buying an apple smartphone.

```
+-----------------------------------+--------------------+-------------------+------------------+
|antecedent                         |consequent          |confidence         |lift              |
+-----------------------------------+--------------------+-------------------+------------------+
|[apple.clocks, samsung.smartphone] |[apple.smartphone]  |0.6340388007054674 |3.8623420398517830|
|[oppo.smartphone, apple.smartphone]|[samsung.smartphone]|0.5538461538461539 |2.3693545938950447|
|[oppo.smartphone, huawei.smartphone]|[samsung.smartphone]|0.5155185465556397 |2.2053890381605720|
|[huawei.smartphone, apple.smartphone]|[samsung.smartphone]|0.5133333333333333 |2.1960406930453145|
|[apple.headphone, samsung.smartphone]|[apple.smartphone]  |0.5129121970599920 |3.1244812450804744|
+-----------------------------------+--------------------+-------------------+------------------+
```

In the result of the algorithm for the table **session_categories** we can see that people tend to buy products of similar category.

```
+-------------------------+-----------------------+-------------------+------------------+
|antecedent               |consequent             |confidence         |lift              |
+-------------------------+-----------------------+-------------------+------------------+
|[components.cpu]         |[components.motherboard]|0.2171875          |439.27017620716515|
|[components.motherboard] |[components.cpu]       |0.21651090342679127|439.27017620716515|
|[kitchen.hob]            |[kitchen.oven]         |0.14145031333930170|32.623266138842470|
|[appliances.ironing_board]|[appliances.iron]     |0.11859649122807017|19.416717433477782|
+-------------------------+-----------------------+-------------------+------------------+
```

On the table above we can see that the lift is high for cpu-motherboard which means that a purchase of cpu will highly impact the probability of the user also purchasing a motherboard and vice-versa. In the output of the algorithm for the table **session_categories_v** we can observe that the lift is low in comparison with the results above, which means that looking up telephones almost doesn't impact looking up smartphones.

```
+---------------------------+-------------------------+-------------------+------------------+
|antecedent                 |consequent               |confidence         |lift              |
+---------------------------+-------------------------+-------------------+------------------+
|[electronics.telephone]    |[electronics.smartphone] |0.3605470258465718 |1.049254090651665 |
|[apparel.shoes]            |[apparel.shoes]          |0.3066989578233196 |30.38944127045021 |
|[apparel.shoes]            |[apparel.shoes.keds]     |0.22444619914183056|26.381189859860445|
|[apparel.shoes.keds]       |[apparel.shoes]          |0.2029590351309315 |26.381189859860445|
|[electronics.audio.subwoofer]|[auto.accessories.player]|0.2001501317482689 |24.808248133550173|
+---------------------------+-------------------------+-------------------+------------------+
```

Although the category ids for the second row are different they both map the same category code that's why they appear twice.

After working with this algorithm we can say that it was easy to use and understand. Also, it gave us interesting results that can be used, for example, to make a promotion on items that have high lift meaning they are frequently bought together. It is also possible to use this algorithm to suggest items to a specific user based on his views/purchases.