SDU

UNIVERSITY OF SOUTHERN DENMARK

Faculty of Engineering - Course 2019/2020

What can a webshop learn from Amazon data?

Leroux Louis-Clément - Molnar Akos - Ngigi Kinungi Francis - Tallaa Alexandre

Contents

Abstract

In this paper, we want to show how managing and analyzing big data leads to results that would not be achieved by other tools. We do all this through a practical problem that provides the opportunity to show results alongside the codes, queries, and insights used. To describe these, we processing large-scale data using a separated environment (Docker), a distributed file system (Hadoop), and an analytical engine (Spark). We make this more efficient with the help of Spark libraries (Spark SQL, MLlib). The Amazon Web Store data was used for the investigation and to describe the planned package. Our goal was to use data from a proven successful store for learning purposes. So, the focus of this paper is on data analysis capabilities that can break down a large piece of data so that information can be got from it, quickly and easily. Our findings shoe the power of the used techniques and demonstrates the value of using such a tool.

1 Introduction

Online shopping is a determining point nowadays as the forecast shows that this market will reach 4 trillion in 2020 [1], which means for example 25% of Americans shop online at least once per month. Online websites allows for a faster, easier, more convenient shopping process, but it is still assumed that many people are afraid that they may not receive the desired product when they receive their order. We find it important that the offer of a webshop, the description and presentation of the products should be met with the needs of the users. Negative comments resulting from misunderstandings and wrong choices can have a significant impact on new customers' perceptions and attitudes towards buying. To prevent this, we have created a package for webshops that contains reports and analyzes that provide essential and useful information for store operators, as they get insight into users' reviews, commitment to product categories, product types, and in general, user interest and their needs and intentions regarding the products. In order to make this package relevant and easy to use, we have based on one of the world's largest online retailers, the Amazon Webshop. Using their data, we learned how to analyze comments and product data to generate understandable, usable and visual report results to the company. At the end of the learning process, we are able to execute the package we generated on the data of other web stores, helping them to grow and achieve greater user engagement. The following documentation describes the preparation process for this product. First you will read in detail about the specific objective and the workflow we have done in the **Problem description** section. Then, in the **Design** section, we explain how each workflow was designed in advance of teamwork for efficiency and organization. This is followed by the **Implementation** section, where specific filtering and insights performed on the selected learning data set are described, along with a description of the visual representation associated with them. Finally, in the **Discussion section**, we discuss whether the package we have created has met its stated goal and what improvement ideas have emerged during the project work.

2 Background

Nowadays online shopping rules the world, but it is very important to give the best user experience to engage the user. We strongly hope it depends on the product they get. We believe that if a webshop can ensure its users get what they really want then both customer satisfaction and the shopping experience can be maintained.

With this in mind, we will explore the implementation of a package that may be useful for analyzing webshop data to improve webshop acceptance.

2.1 Problem description

The most important thing for a webshop is to match the offering products and the user habits, in other words, to make balance between business goals and user needs. However, maintaining this balance is not an easy task, as the business can basically build its own product offering based on its own statistics, but they cannot know in advance what kind of product will the users like and why, and which ones they will not and why. In addition, we would like to point out the fact that a negative comment also can change users' mind, which may encourage customers to not buy. It is necessary to pay attention to this because incorrect description and user reviews that conflict with the description can cause misunderstanding and that customers will not get what they want.

2.2 Aim

We intend to reduce negative user experience and increase customer satisfaction and engagement in the same time by analyzing user reviews and product data. In this way, we can provide webshops a business offer that provides them information without investing a great deal of resources, as they can obtain those by give the public data from the webshop.

2.3 Training data

To build the analytics for our package, we used a relevant data set to learn how to extract information from webshop data that is useful and can help improve business processes. To do this, we chose Amazon data, more specifically product metadata and customer reviews from 1996 to 2018. For us, two specific pieces of data were the starting point for us:

- Dataset A: 5-core review data
- Dataset B: Product metadata

Data set A contains user ratings that have been saved in the marked period by users who have posted at least 5 comments and products that have received at least 5 comments. Using the 5-core instead of using the entire raw data set, the 83.68 million comments are reduced to 41 million, but we believe it is more credible to use data that is not an individual opinion on a product that has not been much appreciated, but rather opinions that can provide a comprehensive view of the purchase situation for that product. Data set B contains general information on the products available during the period, which represents 9.4 million products. The two sets can be linked together by the product identifier "asin" found in both.

It can be said that we are dealing with big data as they correspond to "3V" in the following way:

- Volume: Dataset A is 9.9 GB and contains 41.13 million reviews. Dataset B is 3.1 GB and contains 9.4 million product's metadata
- Variety: Combining dataset A and B we get a variety of sources which contains different but correlated things.
- Velocity: Dataset A and B spans from 1996 - 2018. Dataset A has a velocity of 4 added rows per minute. Dataset B adds a new product every third minute on average, but might have a higher velocity as row may have been updated. Eg. a row contains information about what other users have bought with the product. this might have changed during the dataset's lifetime, though we cannot tell when it or if it changed.

This large amount of data made it possible to search for representative samples and to find out the analytical methods we can use to achieve our goal.

3 Design

In this chapter, we will explain in more detail the data available, the environment in which we examined it, and how the workflow was structured throughout the project.

3.1 Amazon dataset

As described earlier, we worked with two datasets to get the most detailed picture possible. The first, which includes customer reviews, has the following attributes:

- reviewerID: which identifies the contributor
- asin: which indicates the related product
- reviewerName: the name of the customer
- helpful: helpfulness rating of the review
- reviewText: textual evaluation of the product
- overall: rating of the product
- summary: summary/short title of the review
- unixReviewTime: time of the review in unix format
- reviewTime: time of the review in raw format

These values enabled evaluation, time and text-based analysis. The second dataset, which contains product metadate, has the following attributes:

- asin: product ID
- title: title of the product
- price: price of the product
- imgUrl: path to the product image
- related: related products
  - also_bought: those who bought this product also bought these products
  - also_viewed: products viewed when that product was viewed as well
  - bought_together: products purchased with that product
- salesRank: sales rank information
- brand: name of the product's brand
- categories: list of categories the product belongs to

These values enabled price, brand and category based analysis.

3.2 Workflow design

We have identified five types of bases along which insights can be made, once again these was the rank value, the time when the review was made, the text of the review, the product price, and finally the brand and category to which the product belongs. To get familiar with the data and the data handling process each of the group members chose one basis and made insights of them. However, these data set properties of course cannot be completely treated separately, as they alone provide little information, but combining them can give more detailed, important insights into the data. These will be introduce in the Implementation chapter. To set up a basic structure, the chapter was broken down by the identified 5 attributes, and the combined queries were placed where the attribute is more pronounced.

One more important thing about the workflow was that every team members in their own "field" also decided how they would like to visualize the specific results, as an important point of the project was to make the information understandable and visible to everyone, so that the business can respond clearly and immediately to any issues that arise.

We did not devote a separate chapter to visualization, as we found it appropriate to talk about how and why it is presented after giving an description of a specific insight in the Implementation chapter.

3.3 Environment

This section presents the setup and tools used for the implementations of our project solution. The first section beings by describing the programming environment. The following parts proceed with the description of why and how tools were selected and the overall usages.

3.3.1 Programming language

Python is one of the most powerful and widely used programming languages in data science and Machine Learning. Python has a huge set of libraries applicable to data science analysis (e.g. Numpy, pandas visualization etc.) which developers use to almost do anything and everything with the dataset such as organizing the data, cleaning, sorting, filtering, aggregating, calculating among other functionalities. For these reasons, the programming language used to implement the solution for this project is Python because of its ease of use, but also its wealth of many libraries.

3.3.2 Docker

Docker allows us to package and run applications in an isolated environment that we call containers. The containers use the same Linux kernel of hosting operating system. The main requirement for this project was to work with big data. Usually, working with such data involves working with many isolated tools such as Hadoop, apache-spark, programming tools coupled with the hardware used to set up and manage big data clusters. These are some of the basic sets of tools that are required to complete the processing of a large dataset and to perform computations. Since this a group project consisting of multiple students. The problem is that each one of us had specific tools in mind to work on the tasks, this, however, would have resulted in the distribution of many tools and their dependencies to Hadoop clusters. Dependency issues would have occurred as a result of using different tools and would have caused the malfunction. Thanks to the arrival of the docker technology. Docker offers a way to overcome these dependency issues by allowing to build a big data ecosystem in which each tool is self-contained, along with all of its dependencies. This way, each one of use had possibilities

to do experimentations with our tools for the different tasks without worrying about conflicting with other tools because each tool is isolated with a container.

3.3.3 Hadoop

Hadoop can be thought of as to distribute very large files across multiple machines. The issues of storing small files to a local machine or a standalone server are quite trivial, but the idea of storing very large files into one machine and distribute across multiple machines was a quite complicated issue back in the days of traditional relational database management systems. Such complications are some of the reasons which led to the creation of the Hadoop. The HDFS allows the users to work with really large data sets across a distributed system. HDFS duplicates blocks of data for fault tolerance which is the idea if the one machine goes down, the copy of the same file is duplicated on some other machine. Besides that, it has supported many tools that help to perform computations across the distributed dataset. Such tools include MapReduce, apache-spark among others.

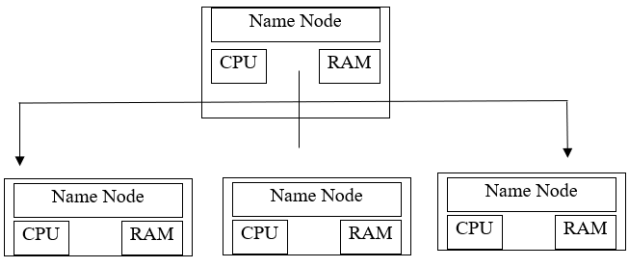

Figure 1: High level of Hadoop

The Master Node or Name Node having its CPU and RAM and that controls the process of either distributing the storage or keeping the information concerning the performance of slave nodes. The effect of HDFS is that it uses a block of data with a size of 128 megabytes by default. And each of these blocks is replicated three times and the blocks are distributed in a way support fault tolerance. So a cluster can go down but the data will go on so small blocks provide more polarization during processing and multiple copies of a block prevent loss of data due to the failure of a node and a map.

3.3.4 Spark

Spark as a framework is one of the latest technologies being used to quickly and easily handle big data. Its ease of use and speed has made it explode in popularity. Spark can be thought of as a flexible alternative to map-reduce. Spark can work with data stored in a variety of formats including datasets that are stored on Google Cloud, Cassandra, Databricks, Hdfs and more. Unlike MapReduce that only works with data stored in HDFS. More importantly, the spark can also perform operations up to 100 times faster than MapReduce. As MapReduce writes most data to disk after each map and reduces operation, spark it keep most of that data in memory after each transformation example df = spark.read.json(xxx.json). df.show(). And it can also split over to disk if the memory is filled. Because the core engine of spark is very fast, it powers multiple higher-level components specialized for various workloads, such as Spark SQL, machine learning. Also, the core of spark is the idea of a resilient distribution dataset (RDD) which means it supports distribution collection of data, fault tolerance, parallel operations and the ability to use many data sources

3.3.5 Spark DataFrames

Spark DataFrames hold data in column and row format. Each DataFrame column represents some feature or variable and each row represents an individual data point. We can then use these DataFrame to apply various transformations on the dataset. At the end can either show or collect the result to display or for some final processing. To work with spark data frames, we first need to start a spark session by importing the pyspark APIs. From there we could be able to read the data and perform operations

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
df = read.json('metadata.json')
```

4 Implementation

In this chapter we will describe the concrete implementation of the insights with the associated query code and explanation of the visual presentation.

4.1 Rank

In this part, we will analyze the data about the rank. On each review, the user mark a grade from 1 to 5. We will see what we can do with this data, which in a whole number, in order to get useful information about the product, category, or user.

Firstly, the first thought about this data is to compare the number of each grade, to see if there are more 5 star or another grade. We will take it for example the category about "Video Games". Here the graph we obtain:

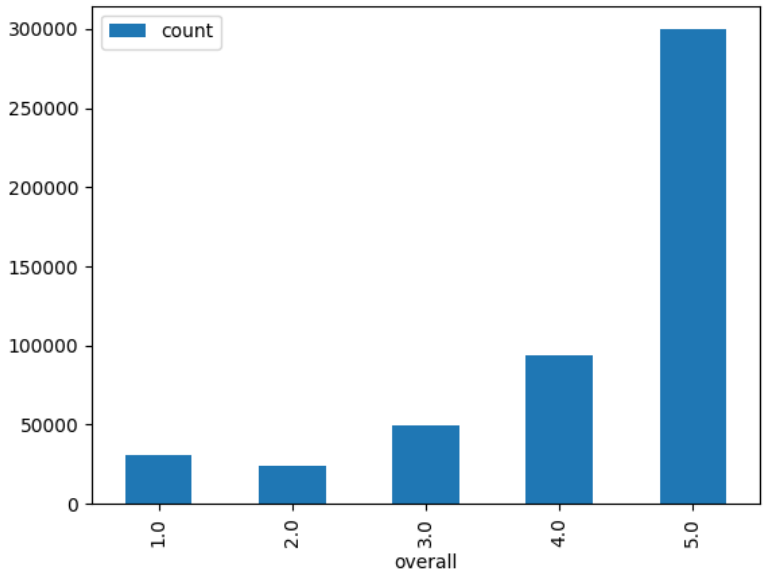

Figure 2: Grades of the Video Games category

We choose to represent it with bars, in grade's croissant order. So, we can easily visualize the difference between the amount of each mark. (We can't trace a line between these bars because the order of the bars is not fixed. It shouldn't matter if we change their place.) If we want some precises numbers, like the percentage of each grade of the video games category, it's better to plot it on a pie chart.

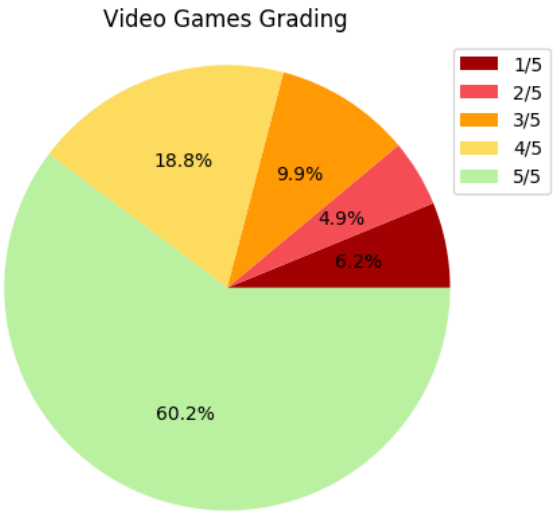

Figure 3: Grades of the Video Games category

With this chart, we can easily get the percentage of each grade. These graphs were pretty basic, and it gives only information about a precise category, here "Video Games". So the webshop can get the information about the global appreciation of this category. To get this chart, the core of the program is:

```
dataGraph = df.groupby('overall').count()
dataGraphP = dataGraph.toPandas()
dataGraphP = dataGraphP.sort_values(by=['overall'])
```

We group the review by their grade and we count them. The pie chart plot take care about the percentage.

We will now analyse again the percentage of each mark, but this time through multiple categories. So we take some categories. We choose not to take every category, there are too many. But it shows what we can do with pre-selected categories, in order to compare them.

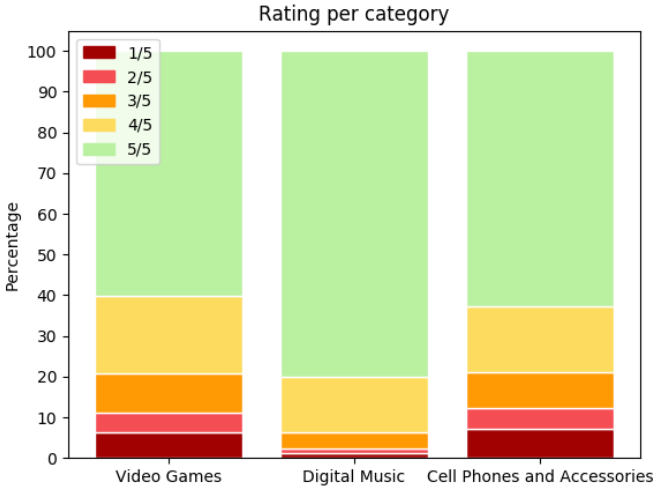

Figure 4: Comparison of category's grading

Here we choose to plot with a 100% stacked column chart. Firstly, a stacked column chart is a chart where we can easily compare some categories. The stacked column chart lets us divide the column with different values, here the percentage of each mark. It is a 100% stacked column chart because we use percentages, so the columns will have the same height. We could otherwise use a classic stacked column chart with the count of each mark. But the number of reviews for each chart is not what it is interesting here. It would have just made different height column and we couldn't compare properly the height of each mark, biased by the number of reviews.

Thus, the chart shows us the global appreciation of each category. The webshop can see the categories that don't have good appreciation and focus on to see what is happening. For more information about that, the webshop could analyse the feedback text. We will see later that we have some useful tool to get easily the keywords of all these texts.

To finish, and to introduce the next part about the time review, we can combine the grade analyses with the time. For example, for a single category or all of them, we can draw the mean over the years, to see its fluctuation.

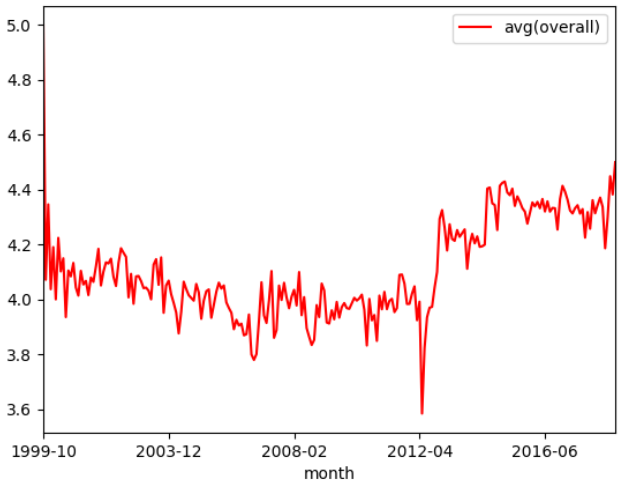

Figure 5: Grading's mean over the years

With the code's core:

```
df2 = df.withColumn("date",F.to_timestamp(df["unixReviewTime"]))
df2 = df2.select(F.date_format('date','yyyy-MM').alias('month'),"overall")
.groupby('month').agg({"overall":"avg"}).orderBy("month")
```

We use is the unixReviewTime to get the date, that we group by month and year. So for each different month, we compute the mean of the grading.

Thus, the webshop can learn the different variations that a category had. He can see if recently this category grading is rapidly increasing, but he can also see in the history some falls in the grading. For example here, the grade is falling in April 2012. So I can look for the reviews on that time to understand what happened and try to avoid it in the future.

And one more interesting thing would be to use machine learning algorithm to predict the mean on the future. The algorithms will learn from the past the different variations and periods over the months, and compute with a variant probability of error a prevision.

4.2 Time

For this part, we perform exploratory analysis of products and reviews to find how the distribution of the number of reviews over the years and months is represented. This means that we had to work with the timestamp and the users' reviews dataset. Time series on a high level can be defined as any data set where the values are measured at a different point in time. Many times series are uniformly spaced at a specific frequency, for our dataset the frequency is distributed as daily count of product reviews, monthly and yearly. To be able to execute the said operations. Firstly, the data type 'unixReviewTime' of columns was converted in the DataFrame 'df' into date time format.

```
df['unixReviewTime'] = pd.to_datetime(df['unixReviewTime'])
```

Additional columns were created in DataFrame 'df' for Year and Month by filtering year and month part of the 'unixReviewTime' column

```
df['Month']=df['unixReviewTime'].dt.month.
df['Year']=df['unixReviewTime'].dt.year
```

Grouping by year and taking the count of reviews for each year

```
Yearly=dataset.groupby(['Year'])['reviewerID'].count().reset_index()
```

Renemaed the column reviewerID to Number_of_Reviews

```
Yearly=Yearly.rename(columns={'reviewerID': 'Number_Of_Reviews'})
```

Line Plot for number of reviews over the years.

```
Yearly.plot(x="Year",y="Number_Of_Reviews",kind="line",title="NUMBER OF REVIEWS OVER
    ↪ THE YEARS")
```

The following results shows that there were no reviews as from year 1996 up to year 2006. A general assumption is that there were no many palmtop devices by then

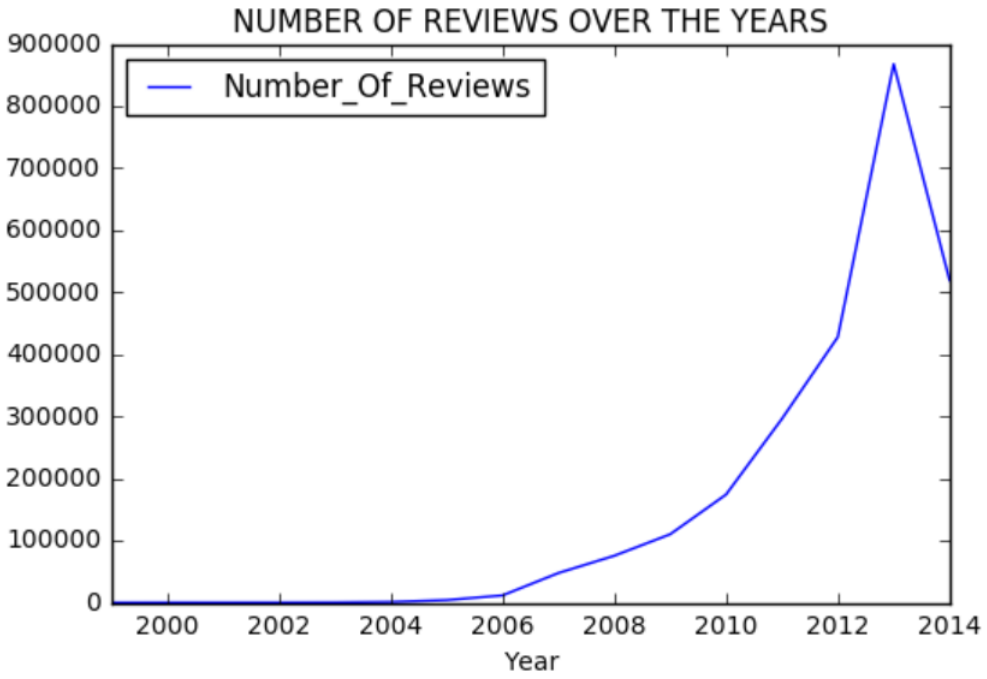

Figure 6: Distribution of reviews

```
Monthly=df.groupby(['Month'])['reviewerID'].count().reset_index()
Monthly['Month'] = Monthly['Month'].apply(lambda X: calendar.month_name[x])
Monthly=Monthly.rename(columns = {'reviewerID': 'Number_of_Reviews'})

Monthly.head()
```

	Month	Number$_O f_R eviews$
1	January	306079
2	February	228285
3	March	233105
4	April	204342
5	May	209967

The timestamp with the number of reviews reveals that the highest reviews occurred in December as well as in January. The results are not so surprising because in most cases, people do a lot of shopping in December. Buying gifts to families and friends.

```
Yearly.plot(x="Month",y="Number_Of_Reviews",kind="bar",title="NUMBER OF REVIEWS OVER
    ↪ THE MONTH")
```

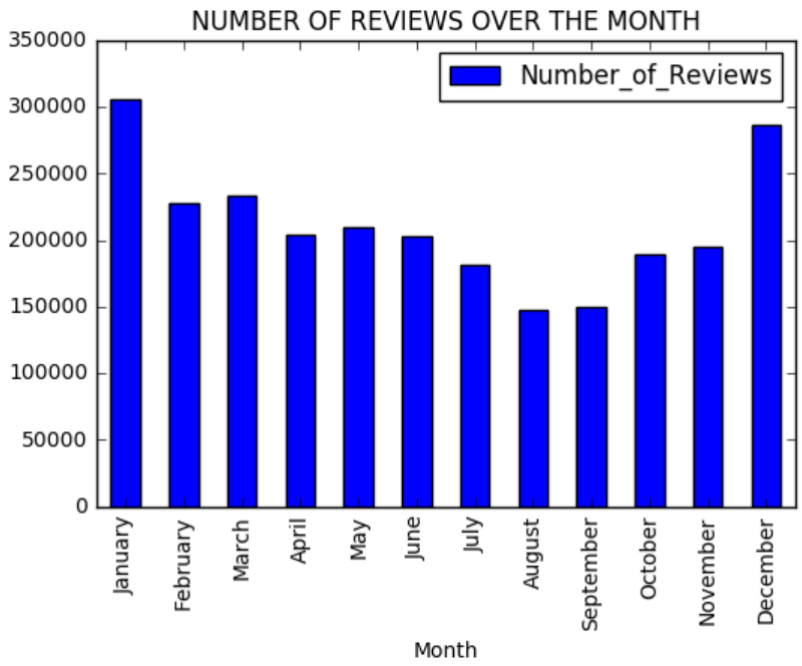

Figure 7: Distribution of reviews

4.3 Price

We intend to look at the pricing on Amazon and how the prices are distributed, and as a first step, basic statistics have been extracted. The minimum price and maximum price show the breadth of pricing, while the average price and mode show where the values are shifting in the distribution. The importance of all this is to find out what price ranges users prefer when shopping online.

The query used for minimum, maximum, and average values was as follows:

```
json.createOrReplaceTempView("data")
dataAgg = spark.sql("SELECT Min(price) as min_price, Max(price) as max_price,
    ↪ Avg(price) as avarage_price FROM data")
```

and for the mode was this one:

```
dataMode = spark.sql("SELECT count(price) as quantity, price FROM data GROUP BY
    ↪ price HAVING quantity = (SELECT max(quantity) FROM (SELECT count(price) as
    ↪ quantity FROM data GROUP BY price))")
```

The following values were obtained after running the queries:

Minimum price	0.01$
Maximum price	999.99$
Average price	34.01$
Mode of price	9.99$

In order to illustrate the values, a funnel graph has been created which clearly shows that, although there are more, expensive items in the product range, less expensive products are more emphasized.

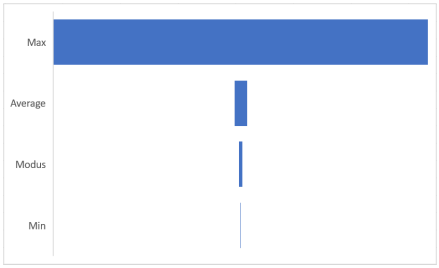

Figure 8: Aggregated price values

With this in mind, the question arose as how the distribution of this wide price range looks like. To find out, another sparkSQL query was run, here the scale had been divided into 10 sections and the values for each one were calculated. The associated code:

```
dataAgg = spark.sql("
    SELECT
        tmp.range as range,
        COUNT(*) as quantity
    FROM (
        SELECT
            CASE
            WHEN price BETWEEN 0.00 AND 199.99 THEN '0-199'
            WHEN price BETWEEN 200.00 AND 399.99 THEN '200-399'
            WHEN price BETWEEN 400.00 AND 599.99 THEN '400-599'
            WHEN price BETWEEN 600.00 AND 799.99 THEN '600-799'
            WHEN price BETWEEN 800.00 AND 999.99 THEN '800-999'
            END as range
        FROM data) as tmp
    GROUP BY tmp.range")
```

The results are represented by a bar chart, which clearly shows that the í$0-199 price range is the highest, which means users want to buy cheaper products online. Also included in the graph the fact that many product data does not include a price, presumably products that are no longer sold have a null value in the dataset.

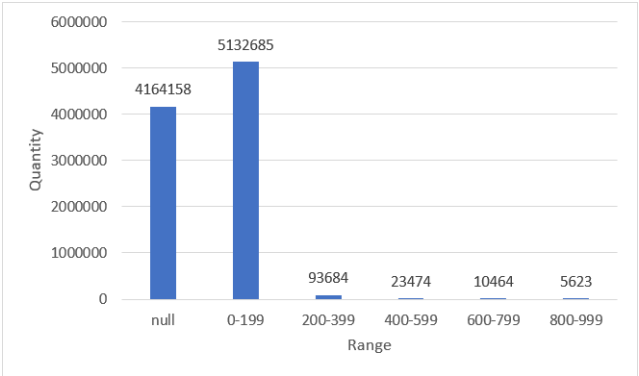

Figure 9: Distribution of price

We found it interesting to look at the largest interval in more detail, so we divided the $0-199 range into 10 more sections and ran the query again. The generated pie chart clearly demonstrates that inside the cheapest range of the previous insight, also the cheapest range is the biggest. This leads to the assumption that selling cheap items is the biggest success for Amazon.

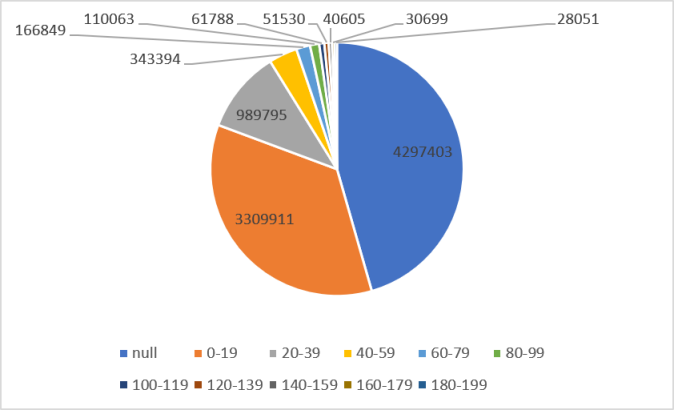

Figure 10: Distribution of price

In the following price insight, we were curious as to which categories had the highest average price. However, as it can be seen from the table below, these values can be found in categories where there is only one product.

Minimum price	Maximum price	Average price	Category
999.99	999.99	999.99	Clothing ... Fashion Watches
999.95	999.95	999.95	Clothing ... New Arrivals
999.0	999.0	999.0	Sports ... Sport Watches
...	...	...	...

Thus, the query was repeated with the condition that the number of elements in the category should be at least 30. The following code was ran for this:

```
data = spark.sql("
       SELECT
           min(price) min_price,
           max(price) max_price,
           avg(price) avg_price,
           COUNT(*) as quantity,
           categories
       FROM data
       GROUP BY categories
       HAVING quantity > 30
       ORDER BY avg_price DESC
       LIMIT 20")
```

The result can be seen in the following bar chart.

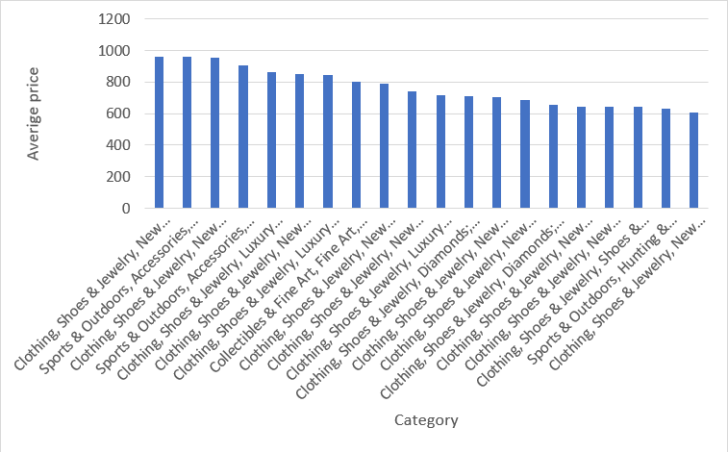

Figure 11: Top 20 average price based on categories

For this result, if we display the number of pieces measured in the category, we get the following bubble chart.

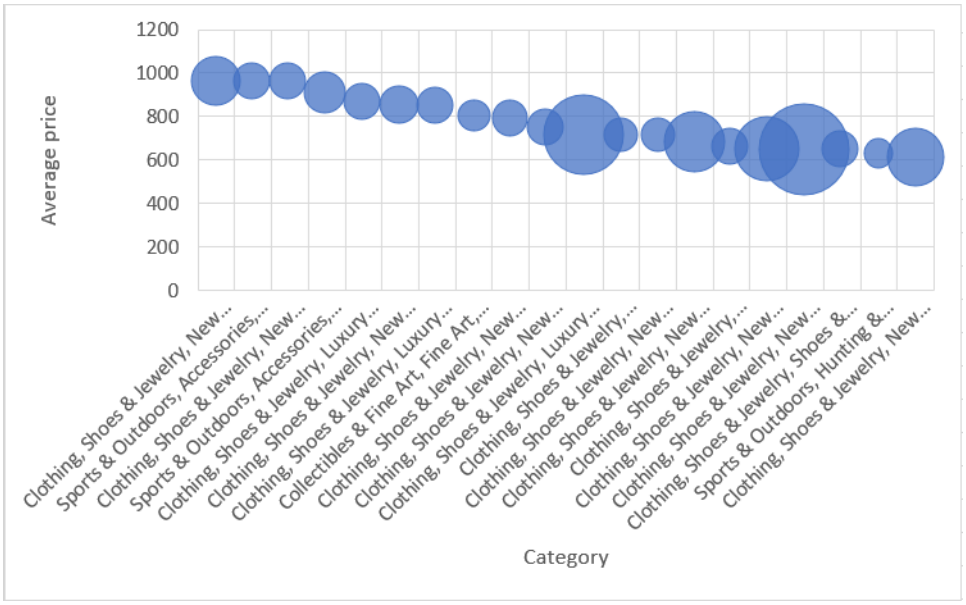

Figure 12: Top average price by categories with quantity

We can read the two categories with the highest volume and average price from the graph, which are the followings:

- Clothing, Shoes and Jewelry > New Arrivals > Women's Luxury Brands
- Clothing, Shoes and Jewelry > Luxury Watches > Women's Luxury Brands

That is, women's luxury products can be sold in the largest quantities at the highest average price online.

Price has been examined in one more aspect, the following insight looking for the answer to what is the relationship between the average price of a brand and the ratings given to the product has the specific brand. For this, prices had to be obtained with the rating number and grouped by brand. The two used dataset had been linked together through the related product.

```
data = spark.sql("
    SELECT
        avg(overall),
        avg(price),
        brand
    From data INNER JOIN data2 ON data.asin = data2.asin
    GROUP BY brand")
```

From the comparison of the two values, the following scatter plot was drawn

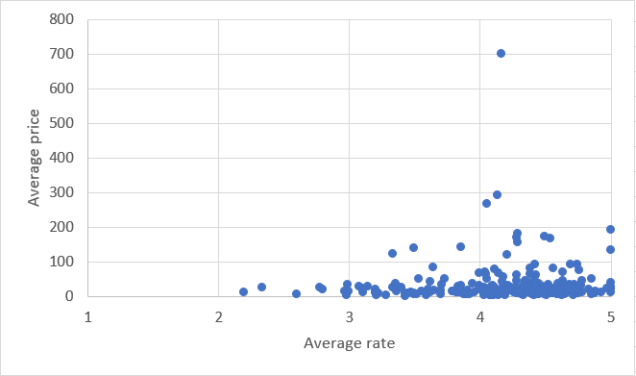

Figure 13: Price against overall value

The chart concludes that the average ratings of a brand is higher for cheaper products. This also shows that in terms of online shopping habits, it can be stated that shoppers prefer cheaper products and this is not significantly influenced by the brand of the product.

4.4 Brands

One of the most important part of building an online webshop is to decide with which brand you should work with. Indeed, to be an online retailer you first need to find providers for your shop. That's where the Amazon review dataset can give you useful insight on what brands are the best for your commerce. One of the first way of doing that is to simply search which brand have the most product listed in the dataset B.

The query for that is as follow :

```
data = spark.sql("
        SELECT
            brand,
            COUNT(*) AS Total
        FROM data
        GROUP BY brand
        ORDER BY COUNT(*) DESC")
```

We've decided to put the result from this query in a bar chat, limiting the number of brands only to 15 (Figure 14).

From this data, a webshop can learn which brands are the most open for the business of online retailing. It seems that Disney, Mattel and Dorman are the brands with the most

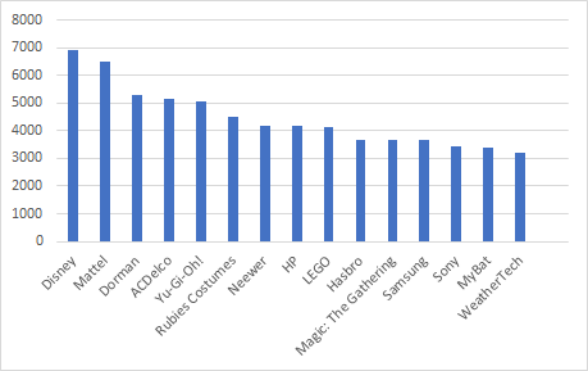

Figure 14: Top 15 brands with the most products on dataset B

products sold on Amazon. Which means that they are the one with which it will be easier to come into contact with, to buy their product and resell them.

Then, when it has finally decided which brand to cooperate with, it can once again use the Amazon data to get an estimation on what kind of product that brand is selling. It can then decide whether or not those products are aligned with its goals. For example, if you want to make a construction webshop, you most likely won't buy Disney's products. Speaking of which, as an example, we can try to get what kind of products they are selling on Amazon with the following query :

```
data = spark.sql("
        SELECT
            categories,
            COUNT(*) AS Total
        FROM data
        WHERE brand='Disney'
        GROUP BY categories
        ORDER BY COUNT(*) DESC")
```

The results will be shown in a pie chart for the 5 biggest category in which Disney propose their products (Figure 15).

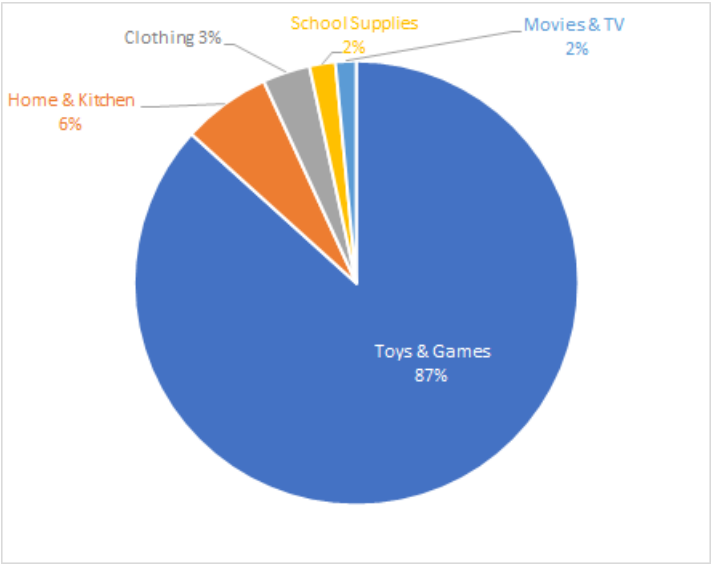

Figure 15: Disney's products categories on Amazon

From Figure 15 we can tell that Disney have a huge part of their online sales in Toys Games,

and almost nothing in Movies TV in Amazon. That's something to keep in mind if you want a partnership with them.

Finally, you can do the exact opposite by asking the dataset which brands have the most of the product that you need. For example, if you want to open a book webshop, you can then use this query :

```
data = spark.sql("
           SELECT
               brand,
               COUNT(*) AS Total
           FROM data
           WHERE categories[0][0]='Books'
           GROUP BY brand
           ORDER BY COUNT(*) DESC")
```

And you get a list of all the brands that sell the most book on Amazon in Figure 16

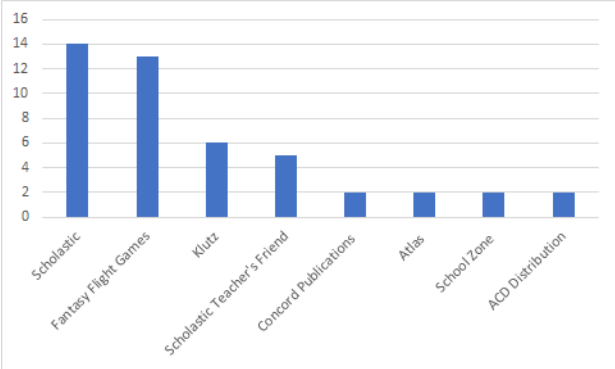

Figure 16: Brands that sell books on Amazon

With that list in mind, you now know that you have to go to Scholastic if you need anything related to books.

4.5 Text review

The little feedback that the user write on the review is a source of very interesting analyses. In fact, to get the main elements of every text review, either you read all of them, or you use machine learning algorithms that give you the important elements. For that, a common tool is the keyword extraction. We want keywords that reflect all the feedback, but keywords also that can be unusual. For example, the TF-IDF method compare the term frequency in a text with the frequency in the whole data set, to get keywords that stand out.

In this project, we use the python package NLTK (Natural Language ToolKit). It is one of the best-known and most-used NLP libraries.

The steps of your keyword extraction, using NLTK, are:

- Word tokenize: the text is split into words.
- Remove Stopwords: we remove every 'stopword'. It means common words like 'a', 'the',...
- Remove punctuations
- Lemmatization: we put the words into their common base form
- Text Classification: we can sort our words according to their frequency

The utility of analysing text feedback is multiple. For example, we can get the percentage of user that are saying something about the price in their feedback. It is also useful to summarize the text, or generate tag clouds according to the text. We see that there are still a lot of potential in this text review that we can develop.

5 Discussion

Implementation chapter showed which information can be obtained through public data of Amazon webshop. We received a comprehensive picture of the strength and characteristics of the web store's product offering. User-provided ratings, pricing, and brand analyzes were described in more detailed. As stated in our goal, we wanted to answer how it is possible to improve the offering of a webshop and make it more user-friendly to engage the customers and gaining their trust and encouraging them to shop online.

The results are the foundation of the package we wanted to produce, because by doing these analyzes, we are able to provide meaningful, business-impacting advice on other companies' data that gives them a preference that they would not be able to exploit through their own resources.

The weakness of the project is that a substantial part of the planned package, the analysis of the written text requires further research and insights. Removing this gap will be a future goal as well as a more transparent and structured exploration of the implementation described above, which also requires further development.

However, it can be said that the results obtained are credible because they are valid data analysis results and can be transferred to other webshops and replicated by the development environment used.

6 Conclusion

Through this project, we learned the basis of Data Sciences. We put in practice the theories that we saw in class, through a problematic about a set of data. The choice of the dataset and the problematic was a key of the project and give us motivation to work on it. The Amazon Review Data is a very interesting dataset with a lot of potential. Even with all of our work, we still don't explore all the possibility of analyse this dataset can give to us.

This project taught us many steps of Data Sciences, as building our cluster on Docker, the utilisation of Hadoop and Spark, analyse the data, and get the best visualized for them,

This project gave us a professional experience, but also a personal one working on a team. We have now some notions about Data Sciences that we will certainly use in our future.

We can also say that we have laid the foundation for producing a valuable, usable and useful product through the practical use of the knowledge we have gained.

7 Reference

[1] Ouellette, C. (2020). Online Shopping Statistics You Need to Know in 2020. [online] OptinMonster. Available at: https://optinmonster.com/online-shopping-statistics/ [Accessed 11 Jan. 2020].