Report 1

SNumbers: u264332, u264443, u264202

Names: Levente Olivér Bódi, Riccardo Zamuner, Giada Izzo

Introduction

In this short report we

Code

Part 1

We would like to reflect on what we did in the code.

Section 1

For Section 1, we imported the necessary libraries first, alongside the nltk stopwords package. After, we made 2 functions, one for preprocessing textual data, and one for preprocessing numerical data. These are combined in the preprocess_document function, which preprocesses with the help of the functions described above. In the preprocess_text function we tokenize the text, convert it to lower case, eliminate stopwords, stem it, remove punctuation, just as described in the assignment.

Section 2

The dataset contains 2 columns, namely crawled_at, and images, which the assignment does not mention in the compulsory return fields, however, for now, we keep them. The fields are read into a pandas dataframe in Part 2.

Section 3

Pros of keeping separate: end user can easily filter products (i.e. if category / sub-category / brand fully matches user filter, product is relevant, otherwise it's not)

Cons of keeping separate: more computation to check general relevance of products

Pros of merging: every aspect of the product helps with relevance (e.g. user can query (not filter) for brands, and the most relevant products will be the one with the user requested brands, but other products, which might still be useful for the user, could still be relevant)

Cons of merging: you lose the option to filter

Maybe some fields could be saved in the inverted index tree using one-hot encoding, to try a hybrid approach, depending on how one wants to build its retrieval engine

We choose not to merge fields. It is understandable that it costs more computationally, however, we aim for better precision this way.

Section 4

We reflect on the section for Part 2, after doing the exploration.

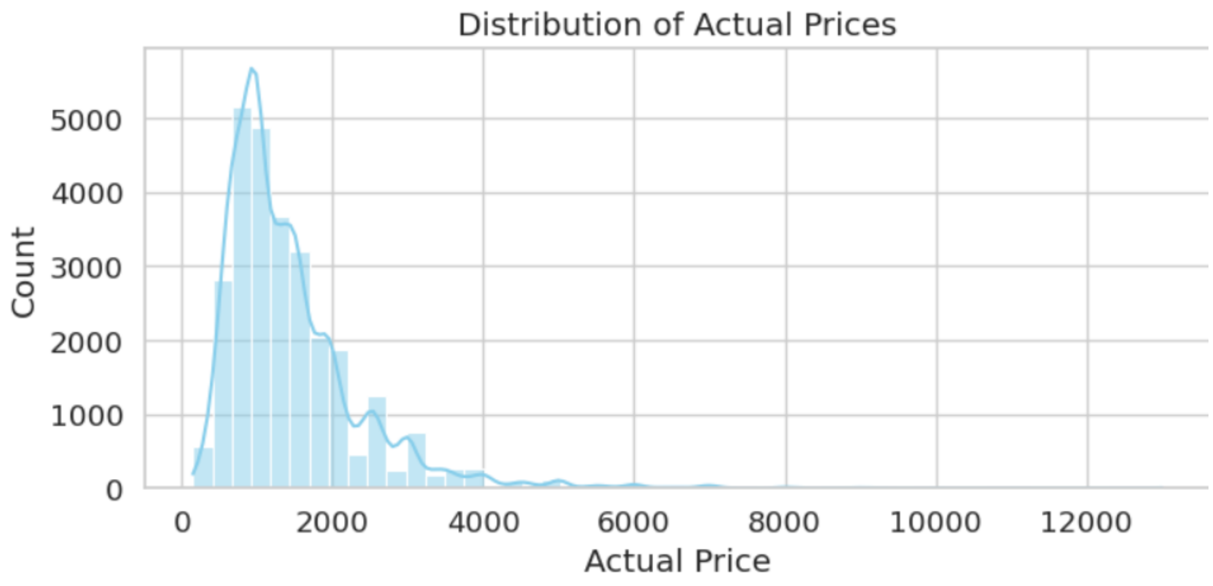
Part 2

After checking the plots and exploring the data, we found out that the dataset contains 855 fields which do not have a discount, 777 fields which do not have actual price, and 2 fields which do not have selling price. We impute 0 for the discount, because we hypothesize that if there is no data on discount, there is no actual discount existing. After that, we impute the selling prices to the missing actual price fields, since if discount = 0, then selling price = actual price. The 2 rows which do not have a selling price we drop, since price is crucial information.

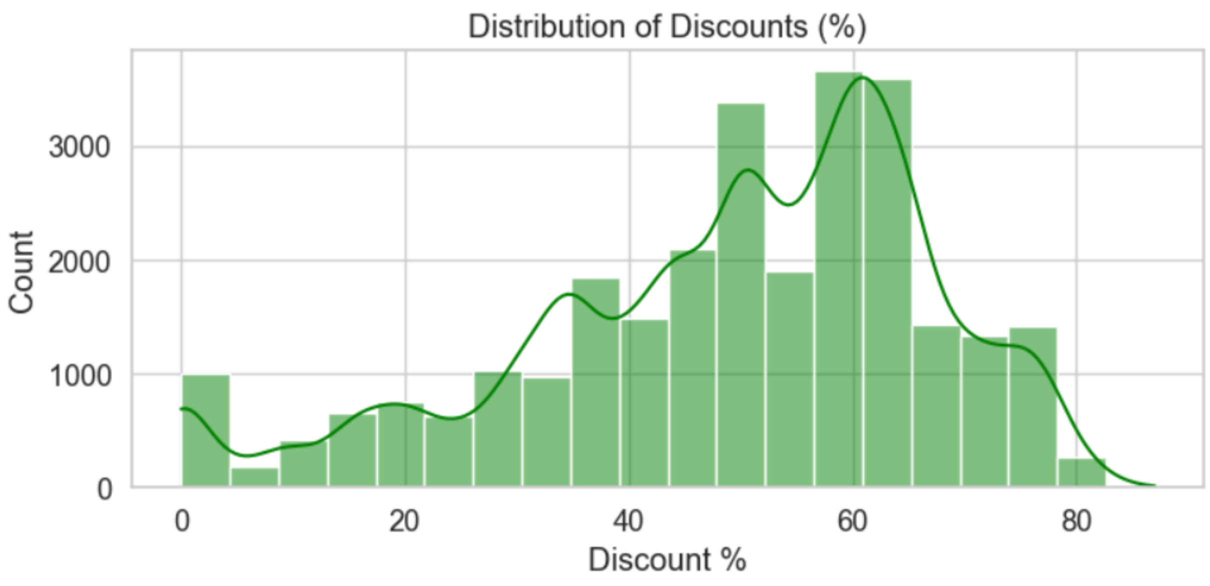
Another column which has a lot of missing records, namely 2259, is average rating. This we will leave for now, since we do not know yet what we need it for and we do not think it is sensible either to drop, or to impute it with zeroes or averages of other ratings.

We also found out that 11149 records are missing for description and 2009 missing for brand. The products without a brand field got 'no brand' imputed.

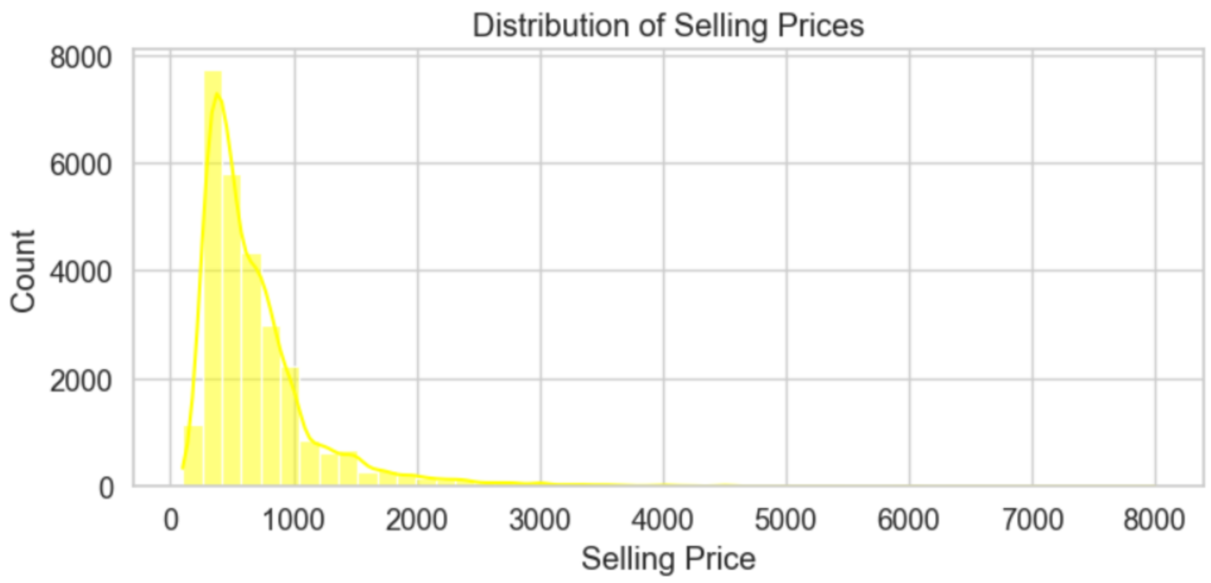
We also want to reflect on some distributions and plots of the data. The actual price of the data is approximately normally distributed with a mean of 1442.79.



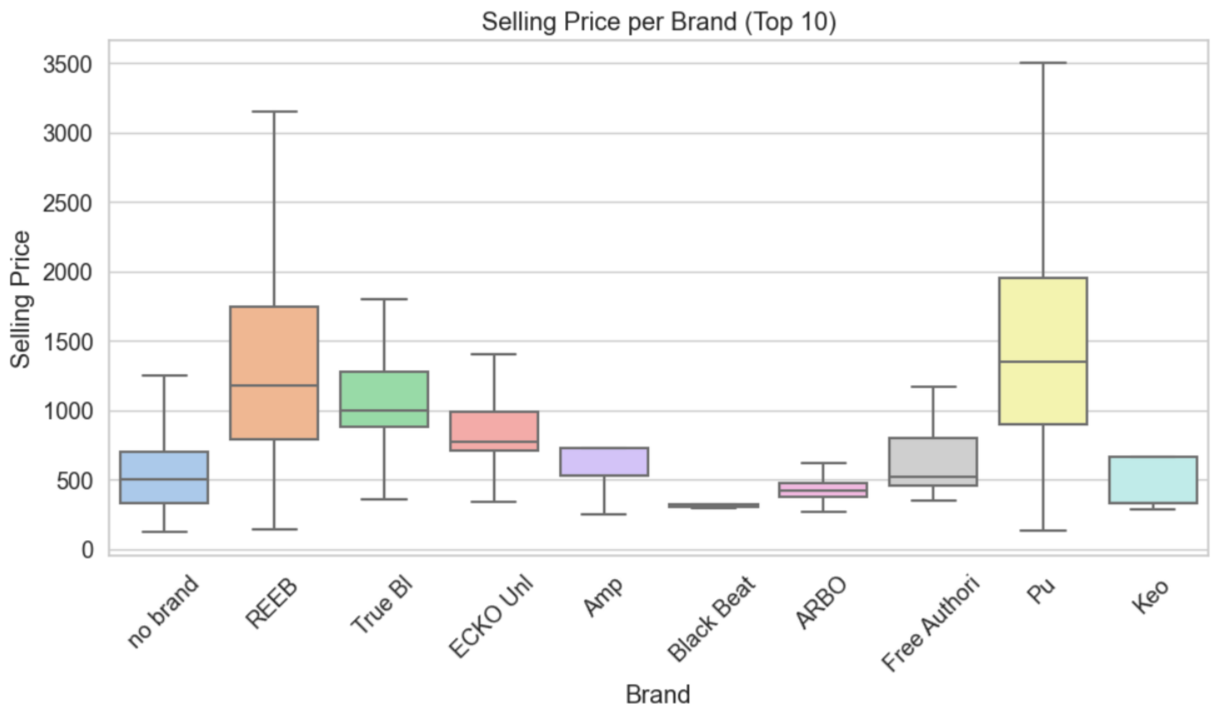
The discounts seem also more or less normally distributed, however, it has a little peak at 0, meaning no discounts, and it is a bit left skewed.

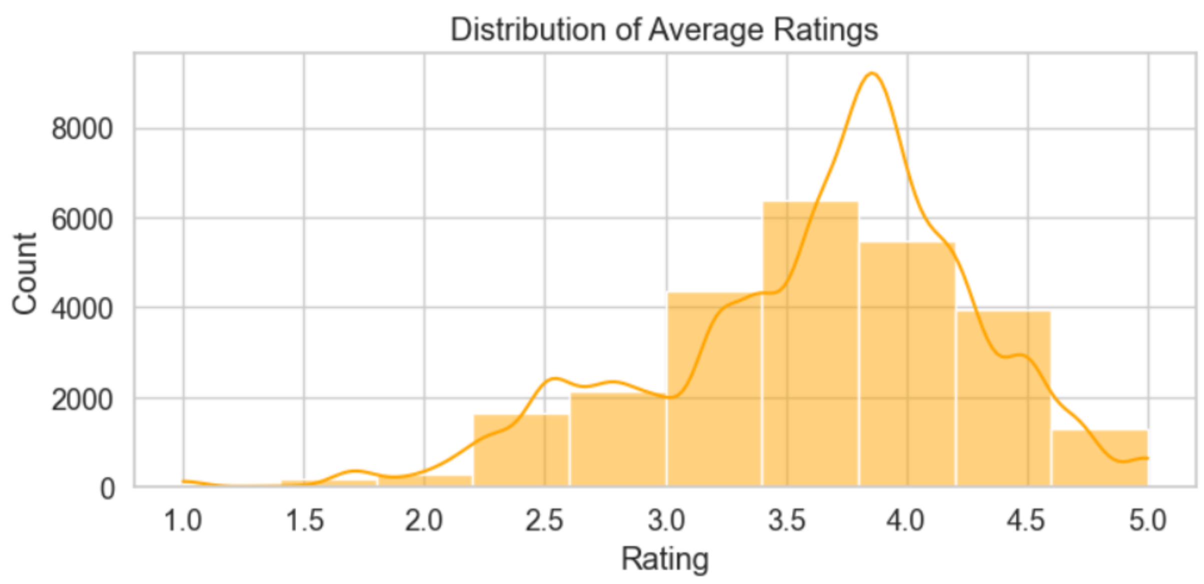


The selling price distribution is right skewed, with a mean of 705.64.



We also plotted the selling price of the top 10 brands in a boxplot, and the average rating distribution:





We also did an exploration word counting, average sentence length and vocabulary size. Besides that, we discovered the top products by specific categories, based on rating, price, and discount.

We think that out_of_stock should be a Boolean variable (either is in stock or not, consumes less space than string), selling_price, discount, actual_price, and average_rating should be floating point numbers, and not text fields, because they take less memory this way.