Prediction of product success on Amazon
Machine Learning for Natural Language Processing 2020

First Author
Yann Boulo
yann.boulo@ensae.fr

Second Author
Tristan Loisel
tristan.loisel@ensae.fr

Abstract

We are a vendor on Amazon and we would like to know if the product we put on the platform will be sucessful or not, taking into account the data we give on the product. We aim to predict a label which is 1 if the product will be successful, and 0 otherwise. This information could be interesting for both the vendor and Amazon. To do so, we first build an index of success and then create a classification model. Our work can be found on github [1].

1 Problem Framing

How can we choose the index of success ? We can base it on the overall ratings given by the clients. Taking the average rating would not be satisfying, because we would not know how to rate a product with no rating, and because a product with one rating equal to 5/5 would be more valuable to us than another one sold 1 million times and with an average rating of 4,99/5. We must also take into account the fact that some products are very recent compared to some others (we obviously have less ratings for them). Finally, we took the **number N_i of ratings equal to 5/5 during the four years following the release of the product** as our measure of success, and transform it into a binary variable to get a simple variable:

$$Y_i = \mathbb{1}\{N_i > a\}$$

where $a \in \mathbb{N}$. In our classification model, we only take as known the data that is given by the vendor when he puts the product online (title, description, price...). We do not use the client reviews or the product rank given by Amazon, as they are highly correlated with the ratings, on which our label is based. However, we take advantage of the fact that Amazon has data on many products in the past, and therefore has data on products which were successful and to which we can compare the new products that appear on the website (for instance using doc2vec).

[1] https://github.com/TristanLoisel/NLP_Project_ENSAE

2 Experiments Protocol

Data: Amazon Video Games sales (84,893 products): we used the reviews to produce the labels $Y_i \in \{0, 1\}$, and the metadata (title, description, price...) to produce features for our model. The Video Games dataset has medium size, hence computations take a reasonable amount of time.
We split the metadata into two disjoint sets: `data_for_model` which is the set of product metadata we use to train and test our model, and `data_for_features` which is the metadata of additional products that we can use to produce features which require labelised data (see Brand Score and Nearest Neighbors features in the following).

Model: We tested several methods from the labs, such as the LSTM networks, but they do not work well in our case. The product descriptions can be very different in terms of content, length, etc. Moreover, even if the description provides information on the quality of the product, it is not enough: we need to use additional data (price, brand information...). Our final model is based on the **XGBoost classifier**. The features are both textual and non textual. Here are examples of features we implemented:

- **Word2Vec similarity with some key words:** Thanks to a pre-trained Word2Vec model, we computed the similarity of each word of the description with some key word that appeared to be relevant in the context (e.g. "guarantee", "delivery", "best"...) and took

the maximum of these quantities. For each product i with a description d, and each key word k_w, the feature is given by $\max \sum_{w \in \mathrm{d}} \text{Word2Vec.similarity}(w, k_w)$

- **Brand score:** For each brand, we computed the average number of ratings equal to 5/5 that the brand obtains. As this score is correlated to the index we want to predict, we computed it on a separate data set: `data_for_features` (disjoint from `data_for_model`).

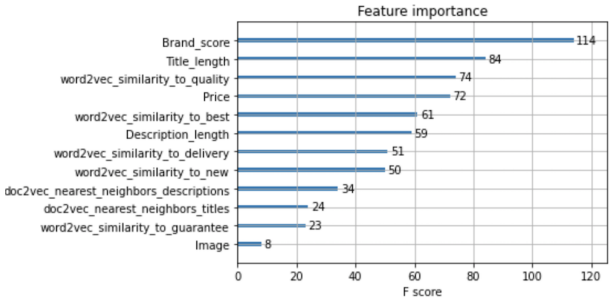
- **Nearest Neighbors using Doc2Vec:** We build a doc2vec model from the products' descriptions. Each description gets a corresponding vector. For each product, we take its 10 nearest neighbors (in terms of description) among the products from `data_for_features` (which are labelised). Our feature is the number of products having $Y_i = 1$ within these 10 neighbors.

- **Price, Length of the title/description...**

Training and evaluation: We split `data_for_model` into a training set X_{train} and a testing set X_{test} (respectively 70% and 30% of the data) to train and evaluate the accuracy of the model. Finally, we implemented a greedy algorithm to select the best set of features.
We chose the parameter a in the index $Y_i = \mathbb{1}\{N_i > a\}$ so that there are approximately 50% of the products with each label. This way, the constant predictors (the one predicting always 0 and the one predicting always 1) have an accuracy close to 50%. The choice of a is important because if it is such that 99% of the data have the same label, then it is very difficult to outperform the best constant predictor.

3 Results

Using both textual and non textual features: One of our tests gave us an accuracy of 77.29% (we always obtain around 70-75% accuracy on the cleaned Video Games data set). As an element of comparison, the best constant estimator (in this case the one always predicting 0) has an accuracy of 52.9%. We computed the features importance (in terms of F-score) in this context and obtained the following graph.



Some non-textual features are more performant than the textual features. One can assume that the title and description lengths reflect the care the producer is giving to its product and thus more serious products. Then the brand score is also crucial, which can be explained by the dominance of big brands over the video games world. Finally we were really surprised that the presence of a picture almost did not have any importance. It would indicate that a majority of products have the same value for this feature. We confirmed it, as we found that only about 17% of the products have a picture.

Using only textual features: Our test gave us an accuracy of 63.30%. Compared with the former 77.29%, we see that non textual features remain an important source of information to predict the quality of a product.

Robustness of the results: We tested our model on larger or less clean data sets (for example on the Toys and Games data set or without cleaning the price column in the Video Games data). In general, we obtained similar results on these data sets (almost 70% accuracy in general, with all the features). A lower accuracy was nevertheless observed when the number of reviews per product was too low.

4 Discussion/Conclusion

The conclusion of this project is that one can predict with a **high accuracy** the success of a product, **using only information entered by the vendor**.

However, our model requires very clean data. After cleaning the data, we only have a few thousands products left, because the metadata contains many NaNs. Therefore, if the data we delete is quite different from the one we have left, our model might have to be adapted to work in a more general context.