

UDACITY

MARCEL JACQUES MACHADO

**MACHINE LEARNING IN PRODUCT CATEGORIZATION:
APPLICATION OF A SUPERVISED LEARNING MODEL
IN A REAL E-COMMERCE DATASET**

CURITIBA

2018

CONTENTS

1	DEFINITION	1
1.1	PROJECT OVERVIEW.....	1
1.2	PROBLEM STATEMENT.....	1
1.3	METRICS.....	2
2	ANALYSIS	4
2.1	DATA EXPLORATION	4
2.2	EXPLORATORY VISUALIZATION	6
2.3	ALGORITHMS AND TECHNIQUES.....	10
2.4	BENCHMARK.....	13
3	METHODOLOGY	14
3.1	DATA PREPROCESSING.....	14
3.2	IMPLEMENTATION.....	15
3.3	PRELIMINARY TESTS	15
3.4	REFINEMENT.....	16
4	RESULTS	20
4.1	MODEL EVALUATION AND VALIDATION.....	20
4.2	JUSTIFICATION.....	20
5	CONCLUSION	22
5.1	FREE-FORM VISUALIZATION.....	22
5.2	REFLECTION.....	23
5.3	IMPROVEMENT.....	24
	REFERENCES	25

1 DEFINITION

1.1 PROJECT OVERVIEW

Online stores have millions of different products that need to be organized in categories, otherwise the customers would have very difficulty in find what they are looking for. The stores would lose sales and money.

Using a machine learning approach, this project will show how to implement a simple and fast solution to the product categorization problem. And, as input of our model, we are going to use a real Walmart dataset that contains ~1.2 million available products.

Of course this problem was already explored before, but usually with smaller datasets. Here we have some links to recent solutions that make use of machine learning as well:

- [Boosting Product Categorization with Machine Learning;](#)
- [Classifying Marketplace Inventory at Scale With Machine Learning;](#)
- [Implementing a Machine-Learning Based eCommerce Product Classification System.](#)

The source code and the dataset of this project, developed in Python and with Scikit-learn, a Machine Learning library, are available on [GitHub](#).

1.2 PROBLEM STATEMENT

Let's say we are responsible for organizing the Walmart catalog of products in categories that will be shown to the final customer in the virtual store. It has ~30,000 product suppliers, such as Sony, Samsung, Dell, LG, Motorola etc. And every day these brands send us a list of their new releases in notebooks, TVs, smartphones etc.

So our job is to classify each new product in one of ~2,000 available categories. However, that may seem to be too much for a human and too intelligent for traditional programming, right?

That's because we can receive thousands of new products daily. And even though we had a traditional program to categorize that, if Apple creates a new product named "iSomething", the source code probably would need to be changed to bind correctly the "iSomething" to, for example, a hoverboard.

So we would like a solution to get the best of the two worlds: the intelligence of a human being and the speed of a computer. And fortunately, machine learning can provide us that.

Basically we are going to preprocess the dataset, keeping only the product name, brand and price as feature data and the product category as label. Some normalization steps will be required. Then we will use different supervised learning algorithms to train and test our data and compare their results using two metrics: accuracy and time. After that we will analyze the wrong predictions and try to implement some improvements based on that. To finish, we will compare the result to another product categorization model, even though the datasets aren't the same.

1.3 METRICS

To get the best solution, we will preprocess the dataset in different ways, try different supervised learning algorithms and then compare them using a very simple metric, the accuracy.

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

Where:

- *tp*: true positive;
- *tn*: true negative;
- *fp*: false positive;
- *fn*: false negative.

That couldn't be a good metric when you want to know, for example, how many predictions failed for each class. But for our product categorization model, that seems enough. Actually, accuracy was chosen among other classification metrics

because that's also our benchmark model choice.

By the way, it's known that a product categorizer would need an accuracy range of 95% to be accepted by small-to-medium sized companies. And maybe this can be our optimistic goal, because our real goal is to overcome the chosen benchmark, achieving the highest possible accuracy in the shortest possible training and prediction time.

2 ANALYSIS

2.1 DATA EXPLORATION

Our Walmart dataset is a CSV file of ~1.2 GB provided by ZanoX, a global affiliate network, and contains ~2.6 million products for sale on its Brazilian virtual store. This is an example of a product we have in the dataset:

	Column Name	Value
0	Program Id	12011
1	ZanoX Id	6267bf6a0ebb5fb08b708bbc4954c45c
2	Product Name	Smart TV Sony KDL55W955B LED 55' 3D Full HD Smart TV Wi-fi integrado Motionflow 480hz
3	Product Short Description	<null>
4	Product Price	4999.00
5	Product Price Old	<null>
6	Product Manufacturer Brand	Sony
7	Update Date	01/12/2017 11:03:00
8	Image Medium URL	https://static.wmobjects.com.br/imgres/arquivos/ids/5068145-250-250
9	Image Large URL	<null>
10	ZanoX Product Link	http://ad.zanoX.com/ppc/?...
11	Merchant Product Category Path	Eletrônicos / TVs / Smart TV
12	Merchant Product Main Category	<null>
13	Merchant Product Sub Category	<null>
14	Merchant Product Third Category	<null>
15	Product EAN	4905524959369
16	Image Small URL	<null>
17	Extra Text Two	Disponibilidade:false
18	Additional Image One	<null>
19	Gender	<null>

Taking a look in the whole dataset, the most useful columns to solve the

categorization problem seems to be:

	Column Name	Characteristics
2	Product Name	<ul style="list-style-type: none"> ■ VARCHAR(160) NOT NULL; ■ it's in Portuguese; ■ the characters can have accents; ■ contains non-alphanumeric characters; ■ contains irrelevant words, such as prepositions; ■ contains repeated words; ■ upper, lower and camel case; ■ singular and plural; ■ masculine, feminine and neuter words; ■ contains numbers, measures and codes; ■ it may contain the brand; ■ up to 34 words.
4	Product Price	<ul style="list-style-type: none"> ■ FLOAT(9, 2) NOT NULL; ■ value in Reais (R\$); ■ between 0.90 and 100,000,000.00.
6	Product Manufacturer Brand	<ul style="list-style-type: none"> ■ VARCHAR(80) NULL; ■ the characters can have accents; ■ contains non-alphanumeric characters; ■ upper, lower and camel case; ■ up to 13 words.
7	Image Medium URL	<ul style="list-style-type: none"> ■ VARCHAR(70) NOT NULL; ■ URL to the product image; ■ 250 x 250 px.
11	Merchant Product Category Path	<ul style="list-style-type: none"> ■ VARCHAR(140) NOT NULL; ■ formatted as: "main / sub / third category"; ■ 79 main categories; ■ 833 sub categories; ■ 3069 third categories; ■ up to 23 words.
15	Product EAN	<ul style="list-style-type: none"> ■ VARCHAR(13) NULL; ■ International Article Number; ■ contains digits only;

		<ul style="list-style-type: none"> ■ if two products have the same EAN, then they should be the same product.
17	Extra Text Two	<ul style="list-style-type: none"> ■ BOOLEAN NOT NULL; ■ product availability.

2.2 EXPLORATORY VISUALIZATION

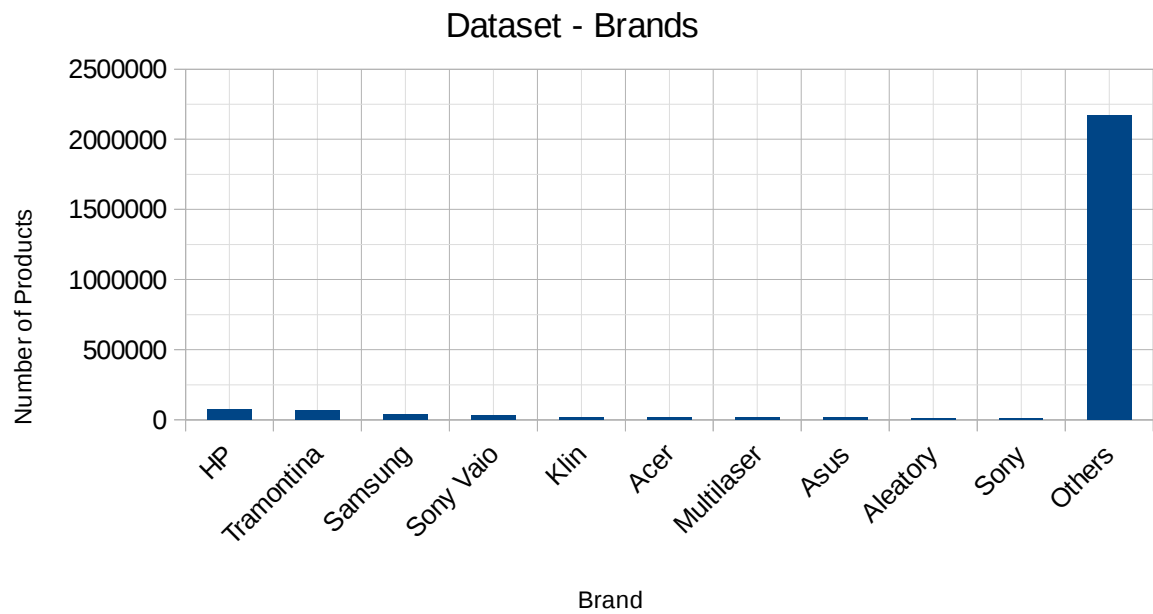
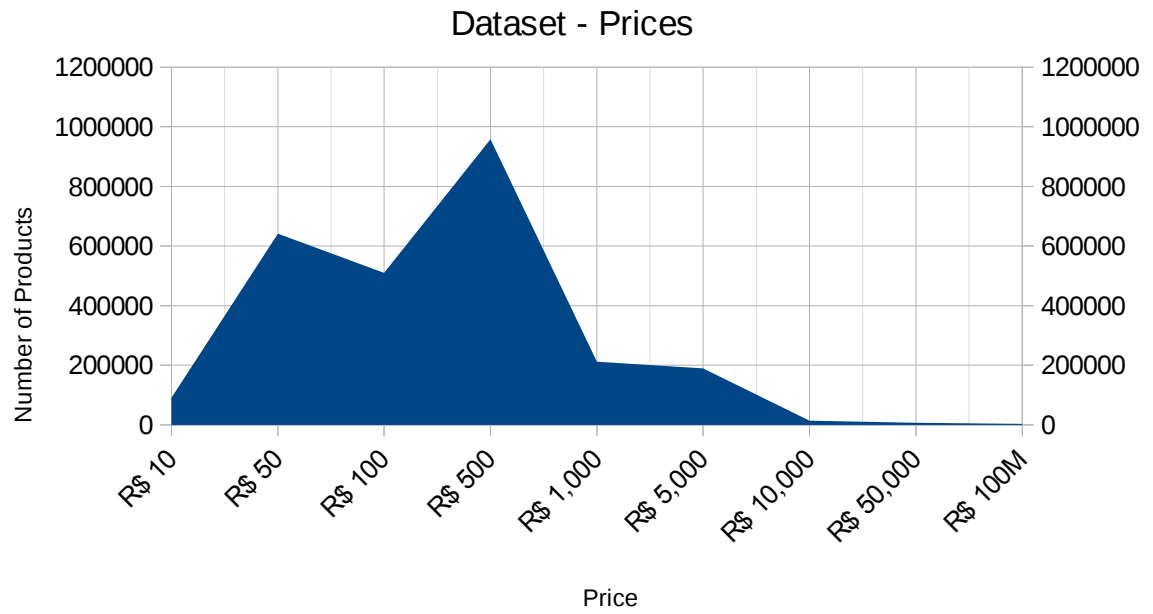
After studying the characteristics of our most relevant features, it was possible to draw some conclusions. They will help us to determine the final format of our dataset.

Column	Conclusions
Product Name	<ul style="list-style-type: none"> ■ the main feature we have; ■ contains not only the product model, but the most important product features; ■ needs to be normalized; ■ the most relevant words are at the beginning; ■ maybe some common product features, such as color names, could be moved from name to new columns.
Product Price	<ul style="list-style-type: none"> ■ there are a lot of products clearly with wrong prices: a t-shirt, for example, could be more expensive than a luxury yacht. However, the apparently purposeful error in the product prices happens mainly in products that are not available for sale anymore; ■ to be used as a prediction feature, it needs to be converted from string to decimal.
Product Manufacturer Brand	<ul style="list-style-type: none"> ■ in most cases it will be the commercial brand, but it's possible to find products where this column is filled with the complete product manufacturer name; ■ needs to be normalized.
Image Medium URL	<ul style="list-style-type: none"> ■ even though it would be possible to use deep learning to recognize the product image and use the result in category prediction, that's not too simple: we would have longer prediction times, more need for memory and an increase in

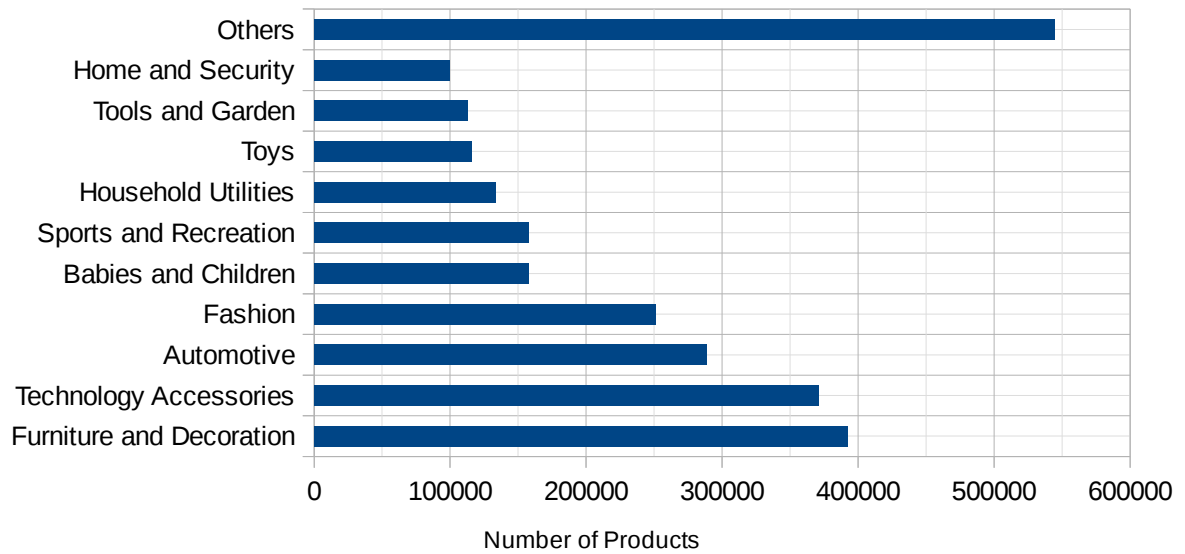
	code complexity, among other possible problems. So this column will be discarded from our dataset with the purpose of simplifying this project.
Merchant Product Category Path	<ul style="list-style-type: none"> ■ our label, the value that will be predicted; ■ the main problem found here is the ambiguity: there are products that could belong to more than one category. Sneakers, for example, have a category in Fashion and another in Sports. So after the prediction we may have sneakers from Fashion category being classified in the Sports category and vice versa. This may decrease our accuracy; ■ some categories, mainly when the product is not available, have an additional text, such as “inactive”, “don’t activate” or “don’t catalog”. Products with these warnings will be considered outliers and will be removed.
Product EAN	<ul style="list-style-type: none"> ■ it could be very useful if we were working with datasets from different stores, where we would have a lot of duplicated products. But in our case we are using only the Walmart dataset, where the most products are different, so doesn’t make sense to keep the EAN column there.
Extra Text Two	<ul style="list-style-type: none"> ■ this is the column that indicates if a product is available or not. So it will be used to keep only the available products (~1.2 million) in the dataset, once the unavailable products (~1.4 million) have problems in the price and category values, as we have discussed.

So in the end of the preprocessing of the original data we would have three features (name, price and brand) and one label (category). But that’s not definitive and can be changed later if our accuracy isn’t good enough. Maybe some features will need to be removed or splitted.

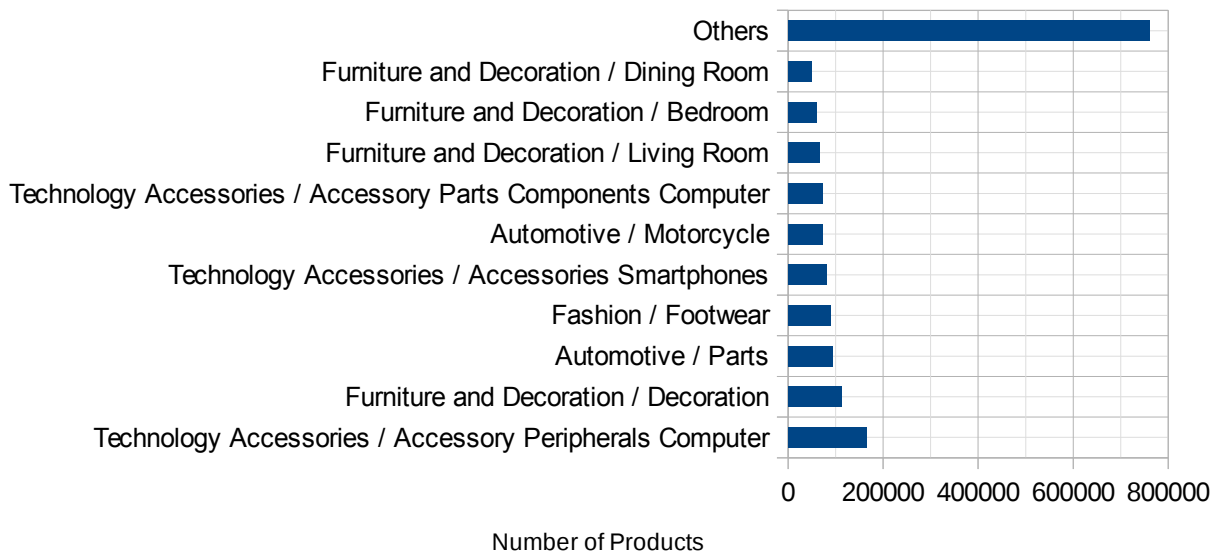
The following charts show the current distribution of these data.



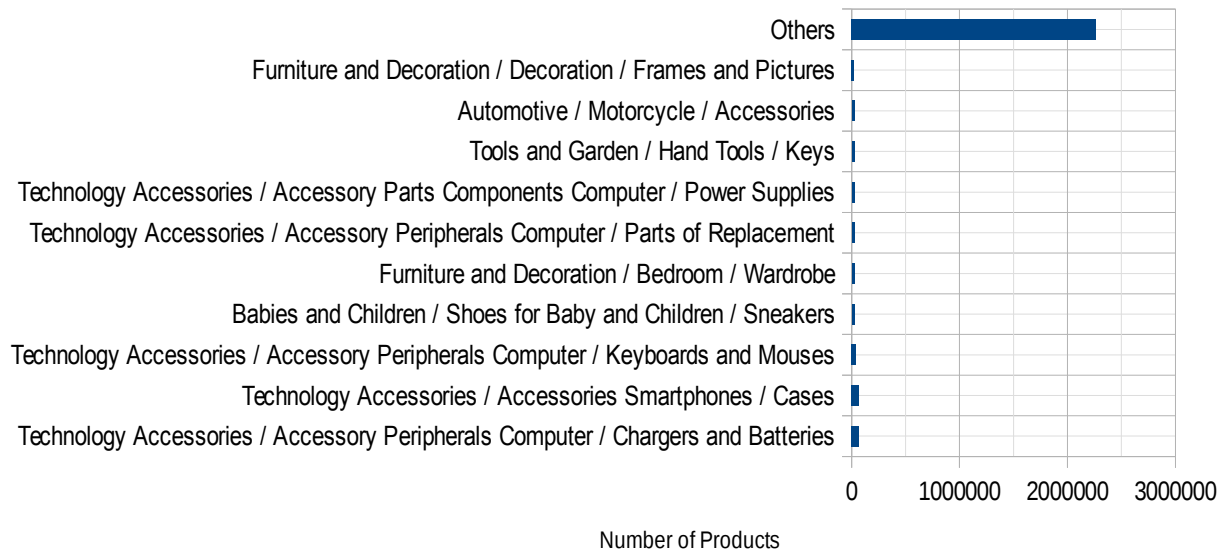
Dataset - Main Category



Dataset - Main / Sub Category



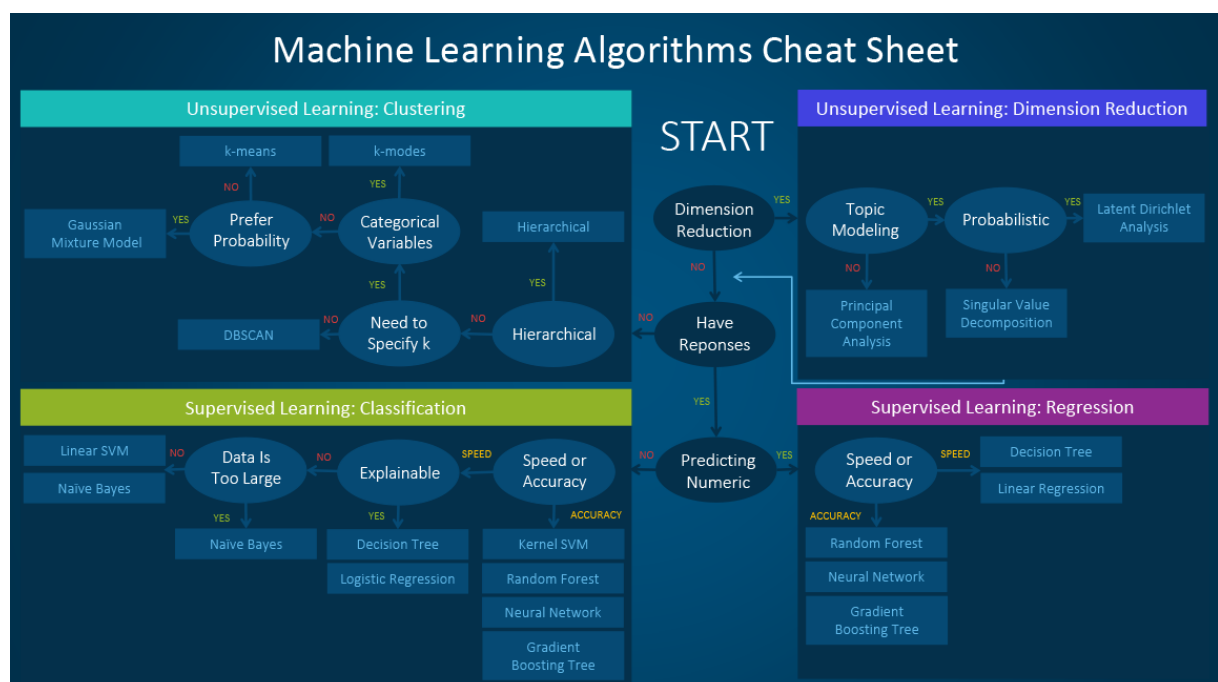
Dataset - Main / Sub / Third Category



2.3 ALGORITHMS AND TECHNIQUES

2.3.1 Algorithms

We will use this sheet to help us with our algorithm choice:



As we can see, the sheet confirms that product categorization is a classification problem once we don't have dimension reduction, but we have responses and we aren't predicting numbers. Moreover, the sheet suggests Kernel SVM, Random Forest, Neural Network or Gradient Boosting Tree if we are interested in accuracy. And Decision Tree, Logistic Regression, Naive Bayes or Linear SVM if we are interested in speed.

Once our dataset is large (~1.2 million products considering the available products only), let's give preference to speed and, among the faster algorithms, we will choose the one with the best accuracy, that will be Decision Tree, Logistic Regression or Naive Bayes. About Linear SVM, that's not recommended for large data.

2.3.1.1 Decision Tree

The Decision Tree algorithm makes use of a tree data structure to build a regression or classification model. Regression models are those that work with numbers, continuous values; classification models, with categorical, discrete values.

There are two node types in the tree: decision and leaf nodes. We put the features (predictors) in decision nodes, that can have two or more branches, and the labels (target) in the leaf nodes. The tree is built top-down and the topmost decision nodes are the best predictors. The distribution of the nodes in the tree is calculated based on Entropy (a measure of disorder) and Information Gain (a measure of relevance of a variable).



Mapping the tree from the root to the leaf nodes, we will get the rules that will be used to make predictions.

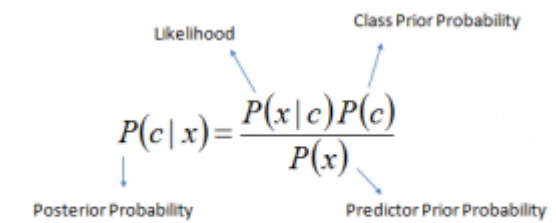
2.3.1.2 Logistic Regression

The Logistic Regression algorithm explores the natural logarithm function (unlike Linear Regression, that makes use of a linear equation) to find the best fitting model to describe the relationship among one or more independent variables.

The test data is used to find the coefficients that will be used in the Logistic equation to predict a binary outcome.

2.3.1.3 Naive Bayes

The Naive Bayes algorithm is a probabilistic classifier based on Bayes' Theorem. According to that, even the dataset features depend on each other, they contribute independently to the classification. For this reason it is called naive. The theorem is stated in this mathematical equation:



$$P(c | x) = \frac{P(x | c)P(c)}{P(x)}$$

$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$

Where:

- c : label (class);
- x : features (predictors);
- $P(c | x)$: conditional probability, the likelihood of c when x is true;
- $P(x | c)$: conditional probability, the likelihood of x when c is true;
- $P(c)$: individual probability of c ;
- $P(x)$: individual probability of x .

2.3.2 Techniques

With the algorithm chosen, we can use Grid Search, an approach to parameter tuning, to improve our model. Or its variation, Randomized Search. And maybe K-fold cross-validation as well, depending on the training time, that could help us to avoid overfitting.

Another technique is the Label Encoding, used to normalize labels, converting non-numerical to numeric labels, once some algorithms can handle only numeric inputs.

2.4 BENCHMARK

To help us to know if we are on the right track, we will use as a benchmark model the academic paper “Applying machine learning to Product Categorization” by Sushant Shankar and Irving Lin from the Department of Computer Science of the Stanford University. As the title says, they tried to solve a very similar problem using a very similar solution.

In that short but valuable paper we can find useful information about the datasets, data preprocessing, algorithms, impediments, perceptions and results, including accuracy and training/prediction time. There are also a list of suggestions that weren’t implemented but promise to improve the model, such as the usage of the product images as an input of the classifier.

About the results, to make a long story short, their model was able to classify ~5,000 products in 3 seconds with an accuracy of 79.6% using Naive Bayes algorithm; using Decision Tree, 8 hours and 86% of accuracy.

Algorithm	Number of Products	Accuracy	Time
Naive Bayes	~5,000	79.6%	~3 seconds
Decision Tree	~5,000	86.0%	~8 hours

3 METHODOLOGY

3.1 DATA PREPROCESSING

The preprocessing of the original data will follow a few steps:

- a) remove the unavailable products (~1.4 million);
- b) remove the ~7,000 products belonging to very small categories (less than 50 products);
- c) remove all columns, except name (Product Name), price (Product Price), brand (Product Manufacturer Brand) and category (Merchant Product Category Path);
- d) remove non-alphanumeric characters;
- e) convert to lower case;
- f) remove accents;
- g) merge whitespaces;
- h) convert the most common plural words to singular;
- i) convert the most common feminine words to masculine.

Then each one of the ~1.2 million products of our input dataset will be formatted as the following example:

Column	Before	After
Name	Smart TV LED 3D 65"" Nano Cristal Ultra HD 4K Samsung 65JS8500 com Conversor Digital 4 HDMI 3 USB Wi-Fi Integrado	smart tv led 3d 65 nano cristal ultra hd 4k samsung 65js8500 conversor digital 4 hdmi 3 usb wi fi integrado
Brand	Samsung	samsung
Price	10799.00	10799.00
Category	Eletrônicos / TVs / Smart TV	Eletrônicos / TVs / Smart TV

As you can see, there will be no changes in Price and Category once the first one is a number and the second one is the prediction label. But Name and Brand will be transformed while the other columns will be removed.

Anyway, that format won't be definitive. Actually, that will be only our first

preprocessed dataset and probably won't be the last. That's because there is no way to detect all the problems that this dataset format will have until we run a machine learning algorithm and calculate the accuracy.

3.2 IMPLEMENTATION

With the dataset ready, our category predictor implementation took this way:

- a) read the dataset;
- b) split the data into features and target data;
- c) encode the data;
- d) shuffle and split the encoded data into training (80%) and testing (20%) data;
- e) import a supervised learning model and initialize the classifier (algorithm);
- f) fit the classifier;
- g) predict the categories for testing data;
- h) decode and print the incorrect categorized data (for analysis purposes);
- i) calculate and print the time and accuracy.

The main problem faced during the implementation of the model was the lack of memory in the prediction phase, due the huge size of our dataset. And while I was thinking about to buy more memory (I have 8 GB only), a more creative idea allowed me to solve that in another way.

Instead of calling the predict method once, providing all the test data at once, it was created a new method to predict each product of the test data individually. Of course we couldn't keep to use the accuracy method provided by the library, we had to implement ours. But after some tests, we could see it working fine.

As additional information, the refinement process will lead to a decrease in the dataset size (in Bytes, not in number of products), that will allow us to make predictions in the traditional way, without using the new method.

3.3 PRELIMINARY TESTS

To test the first version of our model, we will try 3 different algorithms with 2 different dataset sizes.

We are starting the tests with 5,000 products, the same size of the benchmark dataset, and we will increase that to 50,000 products, a value 10 times greater just to check what will happen to the accuracy and time.

Algorithm	Number of Products	Accuracy (%)	Time (seconds)
Decision Tree	5,000	65.50	1.69
Logistic Regression	5,000	20.40	10.07
Naive Bayes	5,000	46.80	0.19

Algorithm	Number of Products	Accuracy (%)	Time (seconds)
Decision Tree	50,000	81.56	3.86
Logistic Regression	50,000	13.62	365.22
Naive Bayes	50,000	37.71	3.43

That's not hard to see that, among our choices, Decision Tree is the only algorithm that works well with our dataset. Increasing the number of products, the accuracy has improved while the training and prediction time remained low.

On the other hand, Logistic Regression and Naive Bayes had the accuracy decreased while the time increased too much. These classifiers definitely wouldn't be able to process our whole dataset in a reasonable time with a good accuracy. So let's discard them.

And before we start the refinement process, let's test our model with the complete dataset.

Algorithm	Number of Products	Accuracy (%)	Time (seconds)
Decision Tree	1,228,176	85.11	166.30

3.4 REFINEMENT

Taking a look in the first result, an accuracy of 85.11% doesn't seem so bad. However, that means that almost 190,000 products were misclassified, what doesn't seem so good.

Then we will test some changes in the dataset and in the implementation as

well to try to improve our accuracy without making the time worse. To do that we will use a reduced dataset of 600,000 products (~50% of total) and Decision Tree algorithm.

Algorithm	Number of Products	Initial Accuracy (%)	Initial Time (s)
Decision Tree	600,000	84.71	66.12

Change (implementation)	Accuracy (%)	Time (s)
Apply Randomized Search on hyper parameters: {'criterion':['gini','entropy'], 'splitter':['best','random'], 'max_features':['auto','sqrt','log2',None], 'class_weight':['balanced',None], 'presort':[True,False]}	-0.93	-43.88

Change (preprocessing)	Accuracy (%)	Time (s)
Remove Price column	+0.86	-0.43
Solve the ~100 most common ambiguity cases	+2.15	-1.59
Create Gender column	+0.21	+1.95
Create Room column	+1.31	+1.14
Create Vehicle column	+0.67	+0.83
Create Console column	+0.07	+1.26
Create Device column	+0.19	+1.04
Create Pet column	+0.09	+1.03
Create Mattress column	+0.05	+1.00
Create Cup column	+0.04	+1.39
Keep only the 3 first words of name	+0.39	-6.72

Algorithm	Number of Products	Final Accuracy (%)	Final Time (s)
Decision Tree	600,000	89.81	23.14

As you could see in the tables, the only proposed change in the implementation was the Randomized Search usage. And after a lot of tests with different parameter values, I decided to choose those that allowed us a great gain of

speed (66%), even losing some accuracy. It would be possible as well to improve our accuracy in 1% or 2% choosing other parameter values, but the time would increase too much.

The other changes were in the preprocessing and they changed our dataset format. In the product name, for example, we kept only the 3 first words and removed the other all. Indeed, the relevant words are in the beginning and removing the others we could improve our training and prediction time significantly.

We also removed the price column, that was decreasing the accuracy. Even the price was used as a continuous value and we have tried to scale that with a logarithmic function the price wasn't helpful. We already knew some prices were wrong and maybe that have made some difference here.

But we also have created some new columns, that can assume some certain values according the words present in the name or category columns. The idea to add new columns came after observing some similar categories with a lot of misclassified products. We had, for example, female perfumes categorized as male perfumes and vice versa, so the new feature gender (and others) were created to help the Decision Tree to choose the right category.

New Column	Possible values	Target
Gender	male, female, child	clothes, perfumes
Room	kitchen, dinner, office, living room, wc, laundry, room, external	house, furnitures (chair, table, carpet etc)
Vehicle	car, motorcycle	automotive
Console	xbox one, xbox 360, ps3, ps4, psp, wii, 3ds, vita, ds	games
Device	tablet, smartphone, notebook, pc	chargers, covers, parts, accessories
Pet	cat, dog, bird, reptile, rodent	pets
Mattress	single, double, queen, king, child	mattresses
Cup	beer, water, champagne, wine, dessert, juice	cups

An important point here is that many times we don't have, for example, the gender data in the product name, making it impossible to classify some products in their male or female subcategories, unless we look for the gender data at product

category as well. So what to do in this case? Could we extract that data from the product category to help us to fill those new columns that will improve our model in 2.63%? Or would we be cheating?

There is no exact answer to the question above but, in my opinion, we couldn't do that. Even so I decided to keep the new columns, at least to illustrate that sometimes the only way to improve our model is improving our dataset. And it may be including a little piece of data that is missing. By the way, in the original Walmart data we got from Zanox, there was already a Gender column, but unfortunately all values were null.

To finish, the last change was an attempt to mitigate the ambiguity problem, that happens when the dataset has the same type of products in more than one category. For example, there are a lot of keyboards in a category of replacement computer parts but also in the keyboard category. Then, during the preprocessing phase, we should move the products from the general category to the specific one. And doing that with products from ~100 categories we could increase our accuracy in 2.15%.

So applying the suggested refinements, we could improve the model this way:

Algorithm	Number of Products	Accuracy (%)	Time (seconds)
Decision Tree	1,228,176	85.11 (+4.94)	166.30 (-109.93)
		90.05	56.37

4 RESULTS

4.1 MODEL EVALUATION AND VALIDATION

So we have done a lot of refinements to improve the category prediction in our Walmart dataset. But would our model work with datasets from other stores? Let's test the model with the Brazilian datasets from other 3 big e-commerce companies.

Store	Number of Products	Accuracy (%)	Time (s)
Carrefour	106,715	89.98	3.07
Ricardo Eletro	9,353	86.05	0.26
Casas Bahia	173,161	59.97	7.90
Casas Bahia (removing books)	85,450	87.17	2.70

And as we can see, even with other datasets, the model continues to work with almost 90% of accuracy, at least for Carrefour and Ricardo Eletro, once we had to remove all books from Casas Bahia dataset to reach a similar accuracy. Indeed, our model is not prepared yet to categorize books in categories such as Astrology, Romance, Arts, History etc.

Despite some problems, I think we built a good machine learning model here. That's not perfect, of course, but the Zanox datasets aren't perfect either. And even though the model didn't reach our optimistic goal of 95%, with 90% we're almost there. If you are not convinced yet, realize that the entire datasets of some of the leading e-commerce companies in Brazil were trained and predicted here in 3 seconds only, using a personal computer. And to perform that we had only the name, brand and price of the products. What could we do if we had access to the real databases of those companies, with all the relevant data already organized in different features?

4.2 JUSTIFICATION

It's time to compare our solution to the benchmark model, the academic paper

"Applying Machine Learning to Product Categorization".

Model	Algorithm	Number of Products	Number of Categories	Main Category	Sub Category	Third Category	Accuracy (%)	Time (min)
Benchmark	Decision Tree	5,102	21	Yes	No	No	86.0	480
Ours	Decision Tree	1,228,176	48	Yes	No	No	95.01	0.62
Ours	Decision Tree	1,228,176	369	Yes	Yes	No	92.79	0.65
Ours	Decision Tree	1,228,176	1373	Yes	Yes	Yes	90.05	0.94

It's hard to believe that our Decision Tree trained and categorized a quantity of products 240 times greater in a time 510 times lower with a better accuracy. But that's true. Thanks, Scikit-learn. Sorry, Orange.

The machine learning library, however, is not the only factor that matters here. Our model underwent several transformations to overcome the benchmark accuracy, even though our Decision Tree was always faster.

We should remember as well that the benchmark was successful using Naive Bayes algorithm while we decided to discard that due the low accuracy we got.

5 CONCLUSION

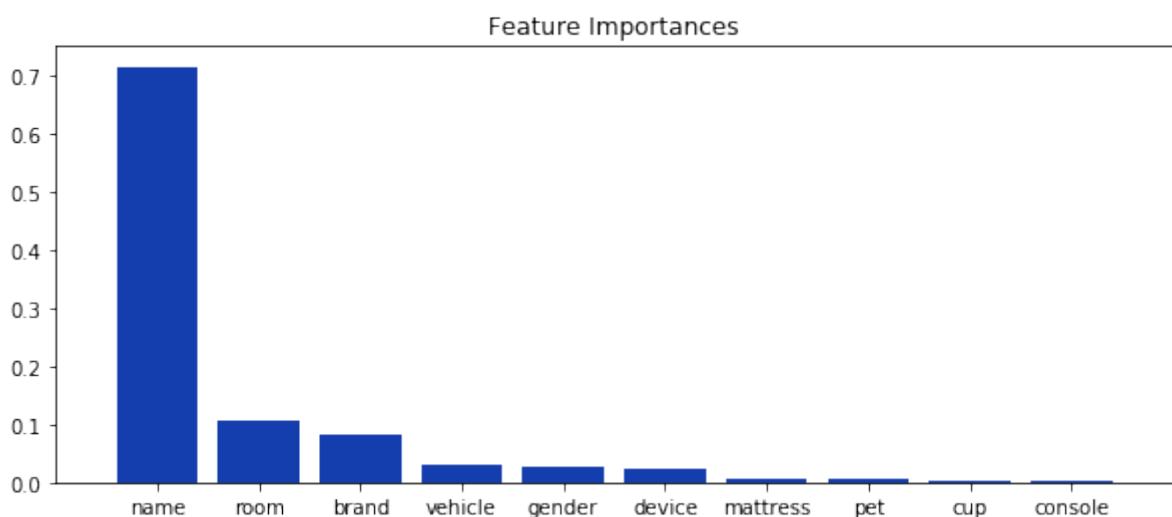
5.1 FREE-FORM VISUALIZATION

In my opinion, the best quality of our model is the speed. And it was totally necessary, once we are not running the model on Google Cloud Platform or AWS, but in a personal computer.

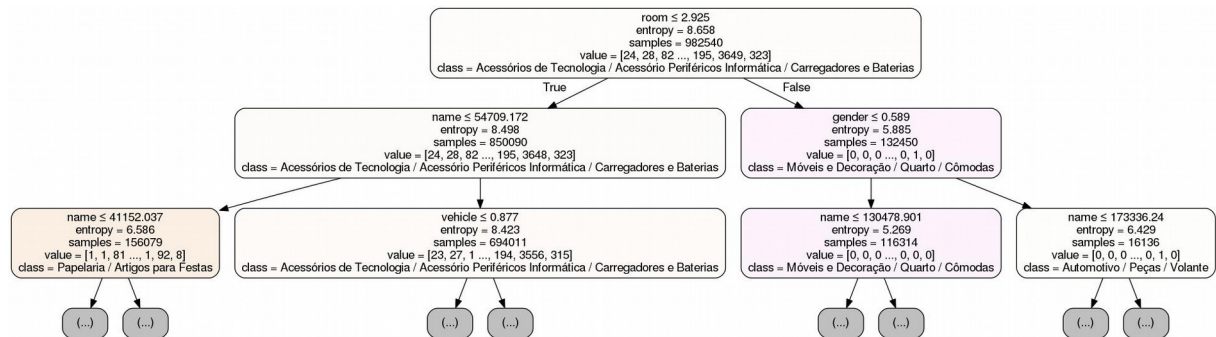
Then we preferred not use, for example, a Support Vector Machine classifier tuned for the best accuracy. Instead of that, we preferred to use a Decision Tree classifier, tuned for the best speed, only improving the accuracy, as much as possible, later. This way we could train and predict ~1.2 million products in less than 1 minute.

We also could try to use NLTK (Natural Language Toolkit) instead of implementing our own preprocessing methods, of course. But I'm sure we couldn't preprocess ~1.2 GB of data in ~5 minutes. Maybe there are better libraries for that out there. The spaCy library, for example, promises to be 400 times faster than NLTK (unfortunately, I just find out that now).

Anyway, in the end we have a really fast product categorizer model and a great accuracy (~90%), that unfortunately can no longer be greatly improved, unless we get more data (features) or make the product categories more generic. Below we can see the importance of each feature for our classifier, according to the data provided by the classifier itself.



And here we can visualize our tree classifier or, at least, its root and the top decision nodes, with the help of Graphviz library:



5.2 REFLECTION

The main problem we have faced during the development wasn't in the product classifier implementation. Actually, it was very easy to use the Scikit-learn library with all its machine learning algorithms and resources. And once our first classifier was ready, it practically didn't change anymore.

Of course we tested several different configurations there to try to improve the accuracy, but without much success. We soon discovered that the reason of our low accuracy was in the dataset.

Then analyzing what went wrong after each execution, it was possible to understand the many problems we had to solve in the dataset: products in wrong categories, two categories for the same type of product, products whose category was impossible to be predicted even for a human being etc.

But wait. If, as a human being, I am not able to read the product name and brand of a chair, for example, and define whether that's a chair for the kitchen, external area, dinner room or office, how could my machine learning model be able to do that?

The model and me are both going to classify the product correctly as a furniture (main category) and as a chair (subcategory), but we are both going to make a mistake as well, because we may not have the piece of data that allow us to know whether the chair is for kitchen, external area, dinner room or office (third category).

Our difference is that the model may have “learned” that the chair manufacturer (brand) produces mainly chairs for kitchen, while I didn’t know that. Actually I would need to read “kitchen” in the product name to classify that correctly, while the model doesn’t always need it. Then, even when it seems we don’t have enough data to classify a product correctly, the model still has a better chance of guessing than us. That’s an advantage of course. But it doesn’t mean that the model will get to do that all the time. So it still need a good dataset.

I believe that at least 90% of the time I spent doing this project was trying to build a better dataset. And after losing several nights changing the dataset to improve our accuracy in only ~5%, I can state this: doesn’t matter if you have the best software and hardware available, if you don’t have a perfect dataset, don’t expect a perfect result. If your are teaching your model wrong, then you should expect wrong answers. That’s all.

5.3 IMPROVEMENT

Of course it’s still possible to improve our model. Our main problem, the category ambiguity, wasn’t completely removed. And actually, we solved only the ~100 most common cases, even though that cost us a lot of work.

Indeed I don’t know any good solution for the ambiguity problem, including ours, but if I had to bet on something better, I would bet, again, in a machine learning approach.

So we would need to build a new machine learning model to identify the products with more than one category and move that to the most specific one. But this is for an upcoming project.

REFERENCES

ERICSON, Gary; ROHM, William A. **How to evaluate model performance in Azure Machine Learning**. 2017.

GITHUB. **Classification Metrics**. Accessed on February 13, 2018.

HEARTY, John. **Advanced Machine Learning with Python**. Birmingham: Packt Publishing, 2016.

LI, Hui. **Which machine learning algorithm should I use?**. 2017. Accessed on February 10, 2018.

LIN, Irving; SHANKAR, Sushant. **Applying Machine Learning to Product Categorization**. Department of Computer Science, Stanford University, Stanford, 2011.

MACHADO, Marcel J. **Model Evaluation & Validation: Predicting Boston Housing Prices**. Project – Machine Learning Engineer Nanodegree, Udacity, 2017.

MACHADO, Marcel J. **Supervised Learning: Building a Student Intervention System**. Project – Machine Learning Engineer Nanodegree, Udacity, 2017.

MEDCALC. **Logistic regression**. 2017. Accessed on February 16, 2018.

RAY, Sunil. **6 Easy Steps to Learn Naive Bayes Algorithm (with codes in Python and R)**. 2017. Accessed on February 17, 2018.

SAYAD, Saed. **An Introduction to Data Science**. 2018. Accessed on February 15, 2018.

WIKIPEDIA. **Bayes' theorem**. 2018. Accessed on February 17, 2018.

WIKIPEDIA. **Logistic Regression**. 2017. Accessed on February 16, 2018.

WIKIPEDIA. **Precision and recall**. 2017. Accessed on February 10, 2018.