# Methodology

## Phase 1: Problem analysis and general solution

First, I analyzed the information that could be extracted from the html of Carrefour's product pages, and that could also be present in pages from other supermarkets, such as Auchan.

For this, I studied the HTML of several pages through tools such as the Requests and BeautifulSoup library, as well as the Chrome DevTools tool.

After this period of analysis, I found that the html tag "title" (which is quite popular in html) is used on many sites. By using it I could create a classifier that processes different formats of websites and products, but that have the "title" tag.

## Phase 2: Scrapping

Then I made a Scrapping code to iterate over product category pages (selected manually) to extract product links. I selected 31 categories that I used to see on shopping sites (baby items, computers, food, appliances...) as mentioned in the previous section.

By iterating over the category pages, I collected over 1700 links.

## Phase 3: Preprocessing

After this collection, the title of these product pages was treated with text processing algorithms used in NLP projects, in order to prepare the data that would be used for training the classification model. Algorithms were made for character cleaning, case sensitive removal, stopword removal, lemmatization. These algorithms were transformed into functions and are available in the project\_functions.py file.

With the texts of the titles processed, these titles were converted into a matrix of TF-IDF features. I defined that only the 200 most important words in the product titles would participate in this experiment, that is, useful to classify products, which appear in at least 1 product, but at most 70% of the products. I reached number 200 after an empirical study testing other values ​​and analyzing some evaluation metrics (mentioned later) of the model.

These features along with the target defined as the categories from which the products were taken were used as input to our modeling process.

To be able to preprocess new urls as they were informed by the client, it was necessary to store the TF-IDF vectorizer model (the tfidfconverter pickle).

## Phase 4: Model

Studies were carried out on different algorithms for classification, including Decision Trees, KNN, distance to the category cluster centroid. However, the algorithm that presented the best results was RandomForrest, presenting an average of 85% of precision, recall and f1-score considering all the categories studied (see more details in Classification.ipynb)

## Phase 5: Using the solution

Using the Tkinter library, a graphical interface was created for the solution. To use it, just download the dist folder from this link:

<https://drive.google.com/file/d/1i3kNjiK82QjjqFnVIPCGCQALpmW_aON3/view?usp=sharing>

In this folder are the pickles for the model and tfidfconverter (necessary to run the program) and the executable file gui.exe.

**Attention:** It is necessary to have Python installed on the machine and Chrome version 104.

I also created a Jupyter notebook that features the GUI, called GUI.ipynb.

## Conclusion:

As expected the project was quite challenging!

I had some limitations that hampered the development of the solution such as:

- With my computer setup, the scrapping of 1700 pages took 5h to run.

- I tried to use the pythonanywhere platform to provide access to the solution via api made in flask, but the platform does not allow access to a third-party website (Carrefour)

- At first I had a lot of problems with Carrefour's Cloudflare, it was necessary to create several selenium hacks to overcome this barrier

- I also tried using the Heroku platform to create the api, but it had RAM memory limitations for using free accounts.

As future steps, I believe we can add more business-related features to the model, add more product categories, and work with more product pages.