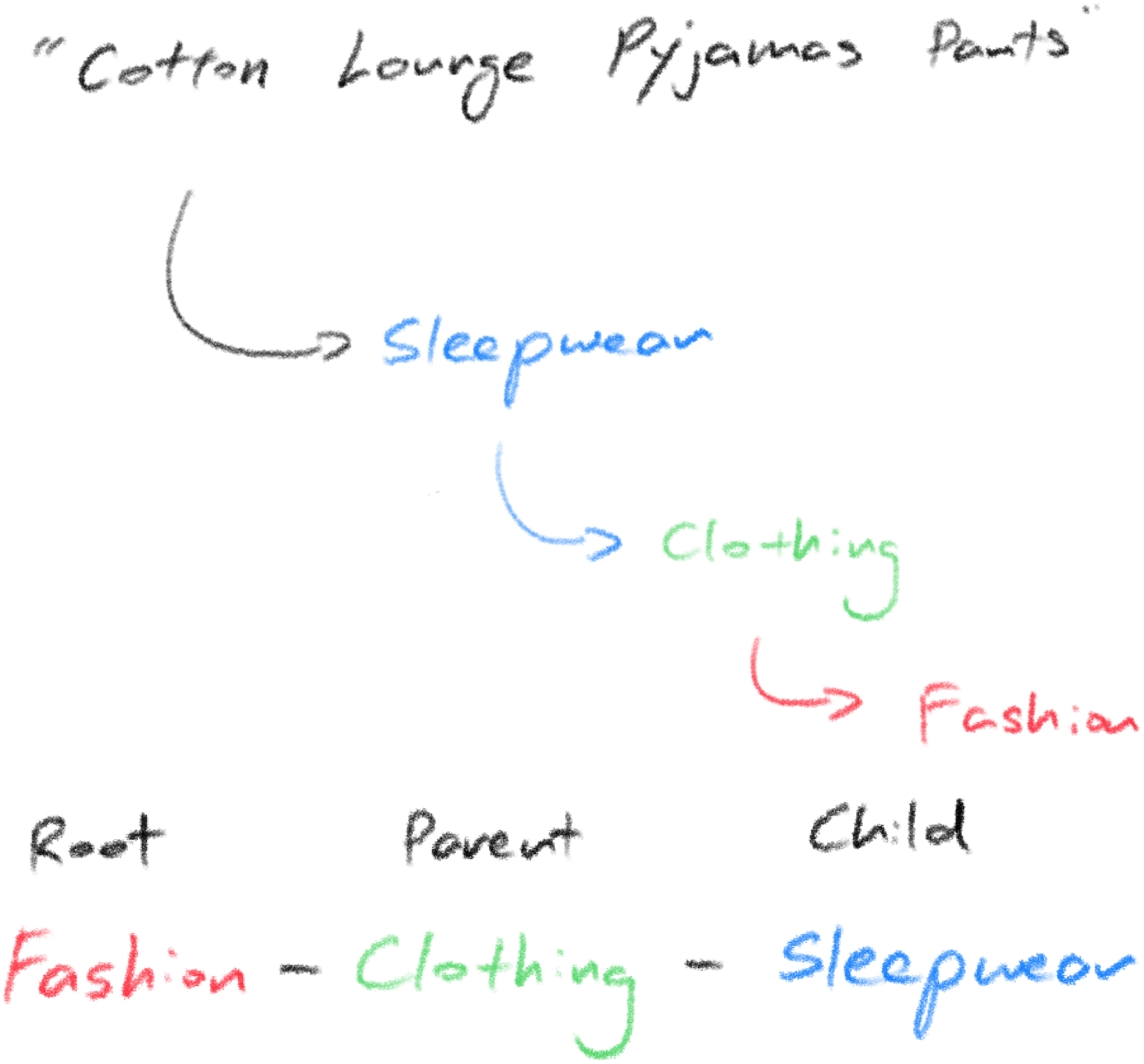
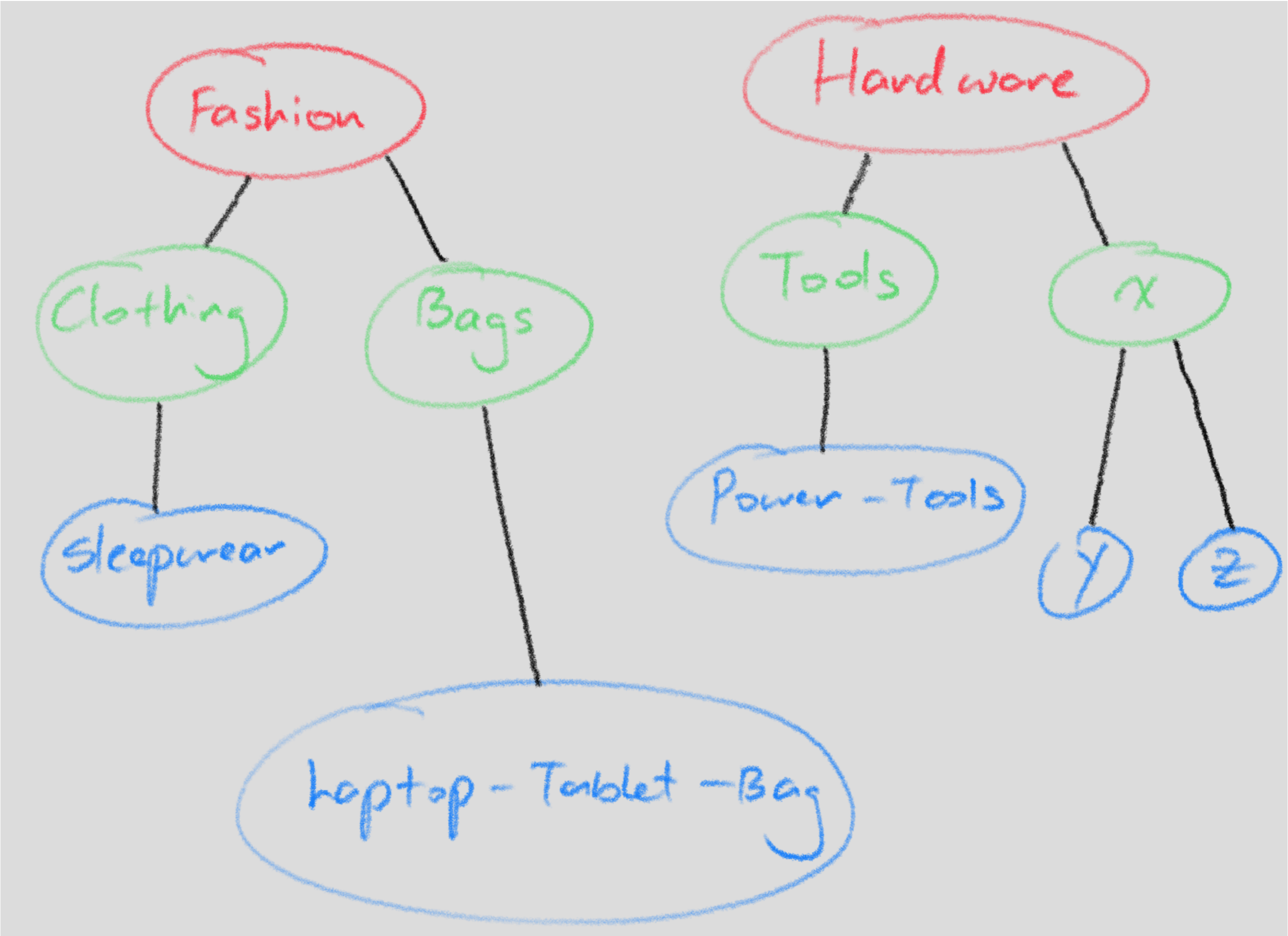
# Data Scientist - Technical Challenge

As the MarTech team in Zip, our mission is to simplify the search experience for our users. In order to achieve this goal we require not only an understanding of our users but also the enormous amount of products that our affiliated merchants have.

We have vast amounts of product data from a variety of websites created by website crawlers that we are able to utilise. In order to make sense of this trove of data, the first step would be to place each product into a “category”. For example, a clothing item with a description of *“Cotton Lounge Pyjamas Pants”* would sit in the category of “sleepwear” which might sit under a parent category of “clothing” which sits under the root category of “fashion”.



If we imagine this as a tree diagram we could imagine it might have multiple roots and parent nodes.



Naturally, we would like to **implement a model to identify the categories/collection of products** based on the features of the product we have from the crawling data.

The current method used to identify the categories involves REGEX and hard-coded rules. Therefore, some errors are expected during the categorisation process.

The dataset provided (download link at the end of this document) is a zipped JSON file containing crawled data from various fashion websites with 125344 entries and 146 feature/attribute columns.

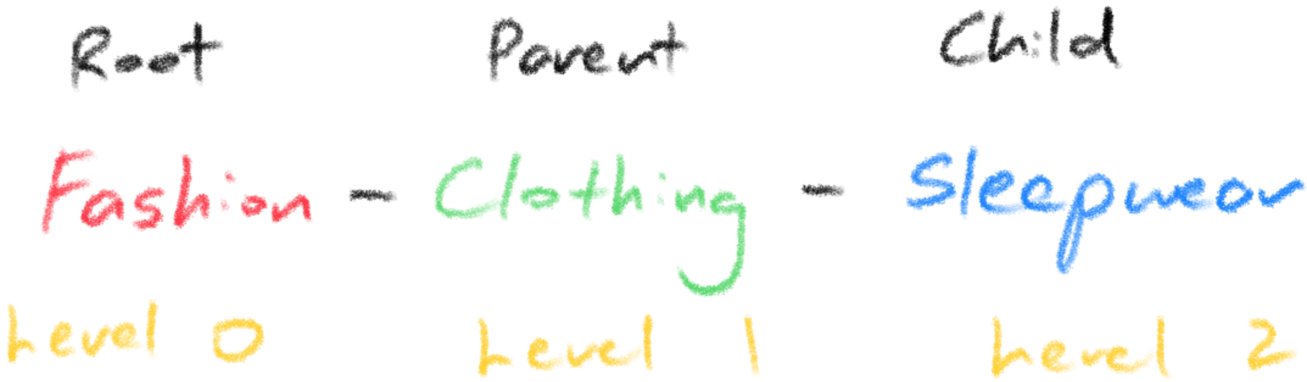
Some notable columns contained in the dataset are as follow:

| Name | Context | Data Type |
|---|---|---|
| availability | whether this product was in stock | string: ‘in-stock’, ‘out-of-stock’ |
| brand | the brand that the item belongs to | string |
| gender | gender for the item | string: 'male', 'female', None, 'unisex' |
| long_description | text description of the product | string and int |
| product_name | name of the product | string and int |
| retailer_price | price of the product | float |
| retailer_url | URL link for the product | string and int |
| e_price | price of the product (could be different due to discount) | float |
| e_product_name | name of the product | string |
| e_brand_formatted | the brand that the item belongs to | string |

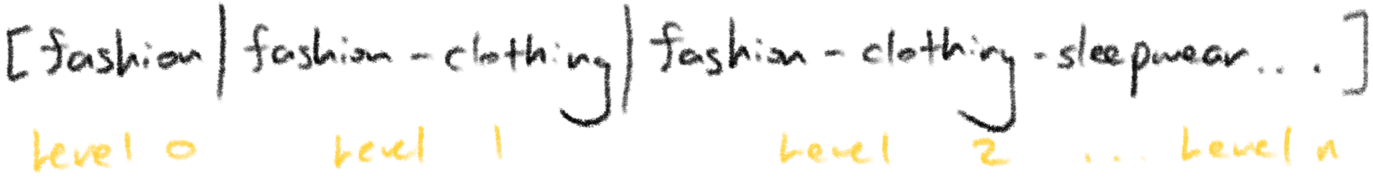
| | | |
|---|---|---|
| e_brand_formatted_slug | formatted brand name (lower-cased, removed spaces and special characters…etc) | string and int |
| id | generated id to identify the product and retailer and code | string and int |
| e_material | the material of the product | string |
| e_color | color of the product | string |
| e_matched_tokens_categories_formatted | The piece of text extracted from the long_description or product_name that helps our internal regex-based classifier to identify the actual category for this product | list of string/s |
| e_color_parent | the hierarchical parent colour | string |
| e_image_urls_square_jpg | url with link to the image of the product | list of string/s |
| **e_cat_l2 or e_categories_path** | **the category of the product identified by our internal regex classifier** | **string** |

As mentioned above, the internal REGEX classifier has produced both the **e_cat_l2** and **e_categories_path** columns to classify what category the product falls under. There are some differences between these two columns though:

**e_cat_l2:** Only contains information that is two levels from the root, so for example if we use the above example then Fashion would be the root category/Level 0, Clothing would be the next one/Level 1 and Sleepwear would be Level 2.



**e_categories_path:** This column contains a list of string with a "|" symbol to seperate the different levels for that particular product. So this would include Level 0, Level 1, Level 2…. Level n


[fashion | fashion - clothing | fashion - clothing - sleepwear . . . ]
level 0
level 1
level 2
. . . level n

So technically, **e_cat_l2** can be treated as a subset of **e_categories_path** and we can treat **e_categories_path** as a more detailed representation of the product as it contains the full "n" levels of child categories.

You are free to be creative and surprise us with your solution! You could use the text description of the data to classify the product or perhaps you could build an image classifier to visually identify the product (there are no images provided in the zip file but the links to the images are there). Maybe it might make more sense to take an unsupervised learning approach to identify the clusters of categories…?

You could implement this on the cloud or locally, but please store the code somewhere like GitHub/GitLab/etc. as we are interested in your coding practices as much as in the final result.

The input file can be found here: https://assets-us.theurge.com/exercise3.jl.zip Good luck!

The below is a guideline for markers

If we think about how we would do it at Zip, we would probably implement these steps:

- The first step will read and parse the JSON file with product data.
- The second step will perform an EDA on the dataset to identify missing data, errors in the data, skewness…etc.
- The third step could be data preparation, including any pre-processing, train/test splits that needs to be done.
- Fourth step would be to build the model/s, optimise parameters and evaluate the results.
- Final step could be generating a report/short summary of anything of particular interest found while doing the task.