**ML-Powered Product categorization for smart shopping options**

# **Backgroud**

Aim: a particular product is categorized in exactly the same way across retailers.

Approaches to perform the mapping(map the products to universal categories across retailer):

1. Manual mapping:  looking at product titles and assigning categories from a universal taxonomy manually.
2. Rule-based categorization: write categorization rules based on retailer categories and product titles.

# **Machine Learning Approach→ for hierarchical product categorization**

1.challenges

+Availability of training data: data created manually or through some heuristic method(used a mix of manual effort and rule-based heuristics applications like feedonomics)

+Large scale data: huge storage, text classification needs text pre-processing, computation intensive.

+Hierarchical taxonomy: the categories we want to predict are hierarchical with different levels

2.→solution 1: All the levels together as one category---------the number of categories would increase exponentially

→solution 2: Different classification for different levels----------This method entails a nested/iterative approach. In the first pass, level 1 of hierarchy is predicted. In the 2nd pass, a separate model is run for each category in level 1 to predict level 2 category.

(If there are 10 distinct categories across products in level 1, 15 in level 2 and 20 in level 3, there would be 1 model in pass 1, 10 models in pass 2 and 150 (10\*15) models in pass 3.)

# **Implementing ML: Steps**

1. step1: Random Sampling

→Instead random sampling, Stratified Random Sampling to be precise, is used to ensure equitable representation from each category

1. Step 2: Data Pre-processing

→Pre-processing for text data; [n-grams/removing stop-words/stemming/lemmetisation]

→converted to a numeric representation(text vectorization methods)[Bag of words/TF-IDF/Word2Vec/Term-Document matrix];

1. Step 3: Training the Classification model

→algorithms used for text classification: Multinomial Naïve Bayes/Support Vector Machines/Random Forests

1. Step 4: Optimising the model parameters

→The performance/accuracy of the models depends a lot on the right values of the parameter

→fine-tune and find the optimum values of model parameters

1. Step 5: Nesting the models

→the model moves from predicting the main category level to predicting sub-level.

→The same model is trained separately for each group with its own train data consisting of products from that group.

→The accuracy is calculated for each of the models separately

→average accuracy across all the models increases as a result of the nesting

→The accuracy increases further as we go deeper in the nesting.

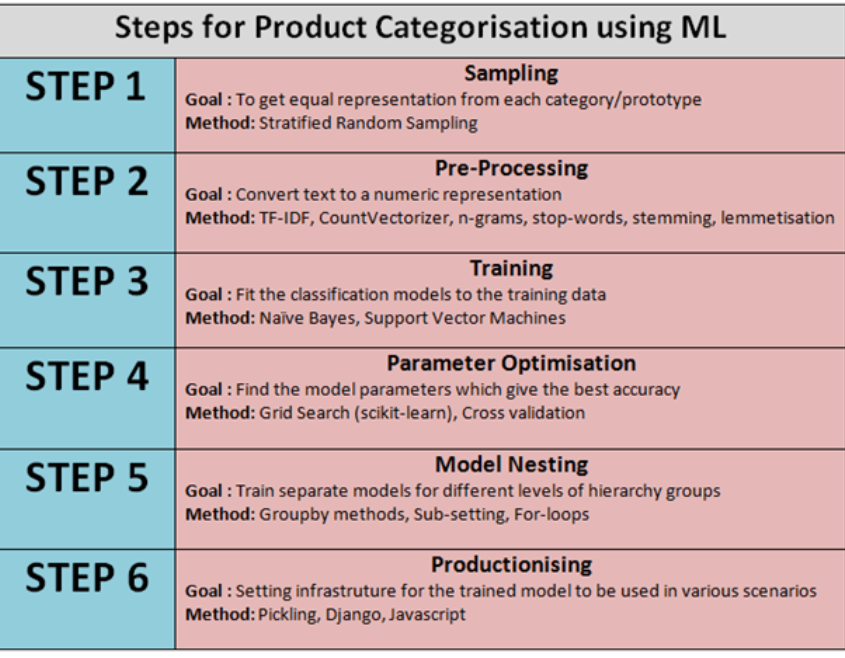
→The overall accuracy increase once we move from a single block category prediction to a hierarchy based category level prediction

1. Step 6: Productionising the model

→The trained model can be used to predict from a file in batch mode

→The trained model can be pickled(saves the attributes/parameters of the trained model in a physical file) for later use

Conclusion:



# **Tools and technologies used**

1.Tools — Python, Scikit Learn, Jupyter Notebook

2.Libraries



# **Applications and scope**

Case 1: Finding generic and used options for a product

Problem: UPC is universal identifier of a product across retailers. But, some of the retailers who sell new branded products and most of the retailers who sell generic and used products don’t have the UPC for products. The matching of similar products becomes really difficult in such a case. A free product title based text-matching (using Elastic Search) gives poor results on a large set.

Solution: Once the products are categorized, we can find the category of the query (new branded) product title and do a product title based text-matching only on the products from that category. This fetches more relevant results. This approach has improved our searches by a great margin.

Case 2: Categorising newly added products

Problem: The retailers constantly keep updating their product inventory. The frequency of these updates varies from daily to monthly. The number of products updated in one update range to thousands of products. For an e-commerce aggregator like us, it translates to 100K product updates across retailers (200+ retailers). Categorising them manually or through heuristics is prone to the problems discussed before.

Solution: The trained categorization model (saved as a pickle file) can be used to predict the categories for these new products within minutes. The prediction from the pickle can also be inserted into the ETL flow at the starting of the process so that every product has a universally (across retailers) identifiable category before it enters the system.

Case 3: Categorising products on the fly

Problem: There are many cases when a user lands on the retailer’s product display page, one would like to get their universal product category on the fly(即时)

Solution: This can be achieved by placing the pickled file in a web server and sending HTTP request with product title every time a page loads.