MIDAS@IIITD Summer Internship Task 2021

Task 3: NLP

Problem Statement

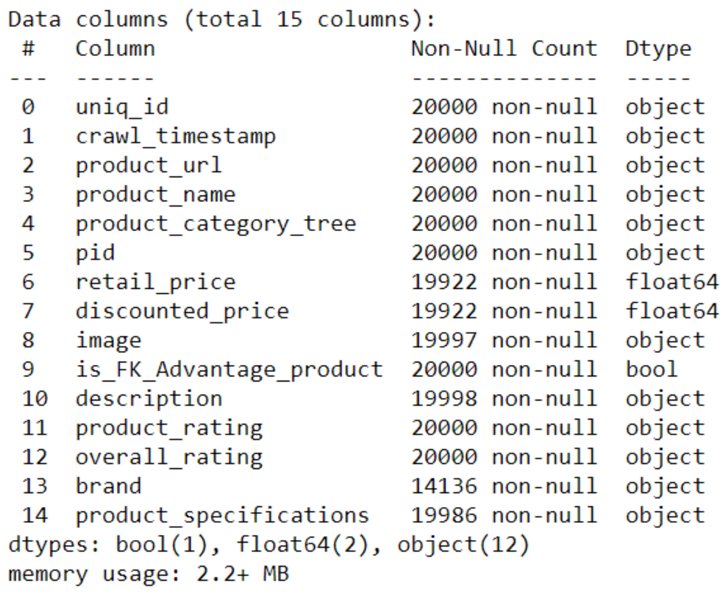
Use the Flipkart ecommerce dataset to build a model to predict the primary category of a product using the description.

About the task

- Clean and visualize the data.
- Separate all categories from the product category tree.
- Figure out the primary category from all the categories.
- Build a model to predict the primary category keeping description as the main feature.
- Measure the accuracy of the model.

Dataset

The dataset had 20000 items each having 15 attributes.
The structure of the dataset is as follows:



```
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   uniq_id                  20000 non-null  object
 1   crawl_timestamp          20000 non-null  object
 2   product_url              20000 non-null  object
 3   product_name             20000 non-null  object
 4   product_category_tree    20000 non-null  object
 5   pid                      20000 non-null  object
 6   retail_price             19922 non-null  float64
 7   discounted_price         19922 non-null  float64
 8   image                    19997 non-null  object
 9   is_FK_Advantage_product  20000 non-null  bool
 10  description              19998 non-null  object
 11  product_rating           20000 non-null  object
 12  overall_rating           20000 non-null  object
 13  brand                    14136 non-null  object
 14  product_specifications   19986 non-null  object
dtypes: bool(1), float64(2), object(12)
memory usage: 2.2+ MB
```

Figure 1: Structure of the dataset

Initial cleaning of the dataset

1. The task required just 2 columns from the whole dataset, so I first removed the rest of the columns.
2. There were 2 rows which had NULL value for "description". So, I removed these 2 rows.

Separating the categories from the product category tree

1. To achieve this, I converted the item present in the product_category_tree column into a list by splitting the elements using **" >> "** as our separator.
2. I found out the **maximum and minimum number of levels** to check how many columns are required to be created to accommodate all the values of the product_category_tree.
3. I created lists to store the values of categories at respective levels and added it to the dataset.

Visualizing the dataset and further cleaning it.

First, I visualized the percentage of items having different numbers of levels in the product_category_tree.

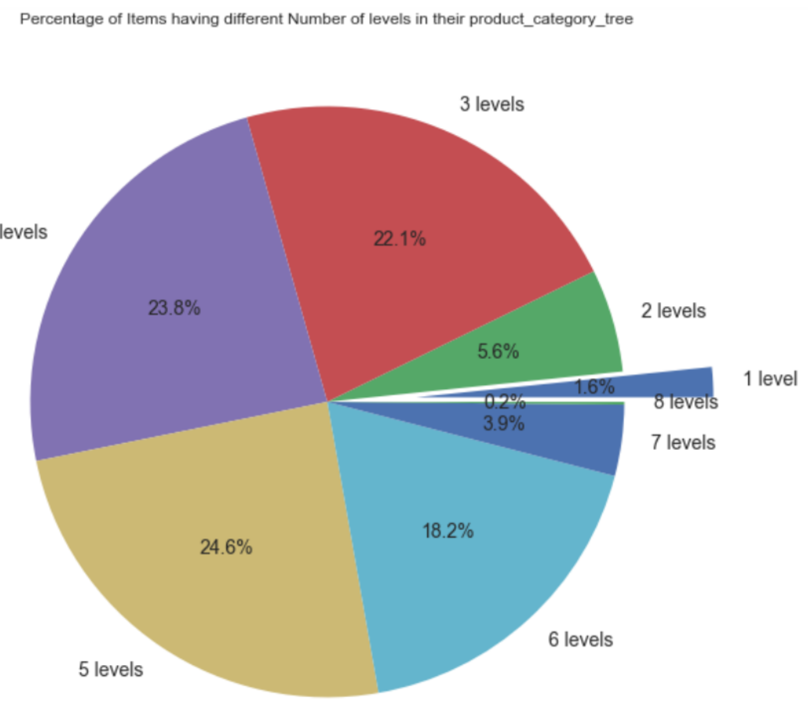


Figure 2: Pie chart showing percentage of items having different number of levels

I also printed some random data of rows having just 1 level in their product category tree. From this, I observed that the data items which contain level 1 only are very specific in nature, thus they cannot be the primary category.

Therefore, I removed those items which only had a single level of categorization.
They formed just **1.2%** of the whole dataset, thus removing them didn't affect the dataset much.

Cleaning the text in description column

I performed the following operations to clean the data:

1. Removed the numerical values and special characters using **Regular Expression Tokenizer**.
2. Converted all the data into **lowercase**.
3. Removed all the **stopwords** (such as the, he, have).
4. **Stemming** all the words i.e. converting the words into their root form.

Figuring out the primary category

First Approach

Use the whole product_category_tree as the primary category.

This approach is **very naive** and **not meant to be used** in real-life scenarios as in reality, the end product is always different and thus no two items will ever have the same product_category_tree.
Therefore, automatic classification according to the primary category would be impossible.

Second Approach

Use the level 1 categorization as the primary category for each product.

Although this approach reduces the number of unique categories, and can be used to train a classification model, it is **not good** enough as it still does not give the exact primary category for a particular item.

Let us consider an example to justify this, Clothing >> Women's Clothing >> Lingerie, Sleep & Swimwear >> Shorts >> Alisha Shorts >> Alisha Solid Women's Cycling Shorts

The second approach will give Clothing as the primary category. But here, Clothing is the **Product Group.** A product group is basically like an umbrella for related primary categories.

For this item, the primary category should be **Shorts** as the item **Alisha Solid Women's Cycling Shorts** is a specific kind of shorts.

Thus, we cannot use the second approach to find the primary category.

Third Approach

I visualized some data and made the following **observations**:

1. The product groups formed a huge part of the dataset while primary categories didn't have that broad range of items.
2. When the number of items in a particular category started decreasing in number, we started moving from product groups towards the primary key.

I used these observations to figure out an algorithm to find the primary category. I observed that while moving in the product category tree, when the number of items in a particular category became approximately 30% equal to the total items at that level, that specific level was our primary category. But if we reached the last level before that, we have to consider the second last level as the primary category.

Also I observed that after level 5, the categories became very specific, thus we didn't have to go beyond that level. Therefore, I used this algorithm only till level 5.

Algorithm:

1. Set a threshold percentage value.
2. Start traversing through all the products.
3. Start traversing through the product category tree.
4. Check if we have reached the last element in product_category_tree.
5. If yes, set the previous level as the primary category.
6. If no, go further in the tree to check if the number of items (n) are more than the threshold percentage of total items (th).
7. n < th : set the primary category to current level.
8. n > th : go further in the product_category_tree.

Then, I used the third approach to find the primary category of each of the products and store them in the dataset.

Preprocessing the data

1. Used **LabelEncoder()** to label the primary category for training.
2. Used **CountVectorizer()** to convert collection of text descriptions to a vector of token counts.
3. Divided the data into training and testing sets.

Building a model and making predictions

I trained three different types of models on the training set : Naive Bayes Classifier, Support Vector Machine (SVM) and Random Forest Classifier.

I optimized the hyperparameters for all the models and got the following accuracy on the testing set.

Table 1: Accuracy of various models

Model	**Accuracy**
Naive Bayes	89.17%
SVM	96.28%
Random Forest	95.22%

Results

Thus, from table 1, we can observe that we achieved highest accuracy i.e, **96%** using **SVM Classifier** on testing data**.**

Other models that can be used

Since the problem statement is a text classification problem, we can also use the following models: Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and Long Short Term Memory (LSTM) Model.

References

[1] Leopold, E., Kindermann, J. Text Categorization with Support Vector Machines. How to Represent Texts in Input Space?. Machine Learning 46, 423–444 (2002).

[2] Changsung Kang, Jeehaeng Lee, and Yi Chang. 2012. Predicting primary categories of business listings for local search. Association for Computing Machinery, New York, NY, USA, 2591–2594.

[3] Erik D. Wiener Andreas S. Weigend and Jan O. Pederson. 1999. Exploiting Hierarchy in Text Categorization. Information Retrieval 1, 3 (October 1999), 193–216.

[4] Tan, Liling and Li, Maggie Yundi and Kok, Stanley E-Commerce Product Categorization via Machine Translation July 2020 Article No.: 11.

[5] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In Proc. of the 13th ACM International Conference on Information and Knowledge Management(CIKM), pages 78--87, 2004.