DataScientest • com

Unified Multimodal Product Classification for Rakuten's E-Commerce Catalog

Submitted By:

Samyuktha Soundararaj
Balamurugan Thirukonda Subramanian Balakrishnan
Syed Suhel

DataScientest.com
Training organization approval 11755665975
09 80 80 79 49
2 place de Barcelona, 75016 Paris

DataScientest

Contents

List of Figures

List of Tables

1 Introduction

Rakuten Group, Inc., a Japanese technology conglomerate, was founded by Hiroshi Mikitani in Tokyo in 1997. Initially centered around its flagship e-commerce platform, Rakuten Ichiba, the company has since expanded into a global leader in online retail, financial services, and digital content, earning its reputation as the "Amazon of Japan." Operating in over 30 countries, Rakuten has a significant presence in Europe, particularly in France, where it manages a thriving online marketplace.

Rakuten France serves as a cornerstone of the company's European operations, offering a dynamic platform where products from both professional and non-professional merchants are listed. The marketplace stands out for its diverse product catalog, encompassing both new and used items, catering to a wide customer base. Efficient and accurate cataloging of products is essential for maintaining Rakuten France's competitive edge, supporting key functionalities like personalized search, recommendations, and query understanding.

To drive innovation and address challenges like product categorization at scale, Rakuten established the Rakuten Institute of Technology (RIT). RIT is the company's dedicated research and innovation division with teams based in Tokyo, Paris, Boston, Singapore, and Bengaluru. The institute focuses on applied research in computer vision, natural language processing (NLP), machine and deep learning, and human-computer interaction. These technologies enable scalable solutions for e-commerce challenges, ensuring Rakuten remains at the forefront of technological advancement.

In Rakuten France, cataloging millions of product listings efficiently is a complex task. Products sourced from both professional and non-professional merchants come with titles, images, and additional descriptions that must be classified into predefined product type codes. This classification underpins the platform's e-commerce functionalities but is complicated by noisy data, unbalanced class distributions, and the scale of modern catalogs.

This project addresses the challenge of large-scale multimodal product type code classification, where the goal is to accurately predict product type codes using both text and image data, along with additional descriptions when available. For instance, a product titled Klarstein Présentoir 2 Montres Optique Fibre with a corresponding image is classified under a specific product type code. By developing a scalable classifier, the project aims to automate the categorization process, improving efficiency and reducing duplication across the platform.

The dataset used in this project was sourced from the Rakuten Institute of Technology as part of their challenge data. It is freely available, accessible, and open to everyone. The dataset is recommended by DataScientest for use in machine learning tasks.

Performance is evaluated using the weighted-F1 score, a robust metric that accounts for imbalanced data and provides a comprehensive measure of model effectiveness. Leveraging advancements in NLP and computer vision, this project not only supports Rakuten France's mission to optimize product cataloging but also aligns with RIT's goals of creating innovative, data-driven solutions that enhance user experience and operational efficiency.

Project Team Expertise:

The entire team made an equal contribution to this project. Every team member had a variety of abilities, including problem-solving, statistics, logical thinking, basic Python, data exploration, analytical skills, accountability, and cooperation.

Member	**Expertise**
Samyuktha Soundararaj	Beginner
Balamurugan Thirukonda Subramanian Balakrishnan	Beginner
Syed Suhel	Beginner

2 Objectives

The main objective of this project is to develop a Multimodal Product Data Classification system that combines Natural Language Processing (NLP) and Convolutional Neural Networks (CNN) to predict the product type based on either the product's description title or its image. The steps carried out in the project are listed below:

1. Understanding the context of the problem
2. Data pre-processing (cleaning, translation of text, categorizing)
3. Data visualization
4. Feature Engineering (data transformation, lemmatization/stemming, tokenization, vectorization)
5. Modeling (simple models, transfer models)
6. Hyperparameter optimization
7. Feature fusion of text and image model
8. Documentation on GitHub

3 Understanding and Manipulation of Data

3.1 Textual Dataset

The textual data used for training the model includes **X_train** and **Y_train**, containing a total of **84,916 records**.

X_train: This dataset includes textual information describing various products, such as the **index ID (Unnamed: 0)**, **designation**, **description**, **productid**, and **imageid**.

Y_train: This dataset contains the **product type code** for the classification task, which is the target variable.
The **X_train** and **Y_train** datasets contain the following columns:

- **Unnamed: 0 (index ID)**: An integer ID for the product. This ID is used to associate the product with its corresponding product type code.
- **Designation**: The product title, a short text summarizing the product.
- **Description**: A more detailed text describing the product. Not all merchants use this field, so some products may have **NaN** values in this column.
- **ProductID**: A unique ID for the product.
- **ImageID**: A unique ID for the image associated with the product.
- **prdtypecode**: The product category used for the classification task (the target variable).

3.2 Image Dataset

The image dataset is provided as a compressed file containing **84,916 images**. Each image is labelled with both a **product ID** and an **image ID**, which correspond to the entries in the **X_train** dataset.

4 Important Features for the Classification Tasks

4.1 Textual Features

The textual data in the description and designation columns plays a crucial role in providing context for product classification. These features vary in length and complexity across different products, contributing significant information for determining the product's category.

1. **Description**: The **description** column contains important keywords, features, and detailed information about the product. This text is essential for understanding the product’s characteristics and its associated category (**Product_type_code**).

2. **Designation**: The **designation** column consists of the product's title, which is a concise summary of the product. It provides critical clues for classification, such as the **brand**, **model**, and specific **features** of the product, which are helpful for categorizing the product into the correct product type (**Product_type_code**).

4.2 Image Features

The image dataset consists of images named with corresponding imageid and productid, which are then used to retrieve the associated product_type_code for classification. The image features are integral for identifying the category of the product.

1. **Image Resolution**: The images have a resolution of 500x500 pixels.

2. **Variability**: The images vary in terms of sharpness, lighting, angles, and other visual factors. These variations provide valuable features such as colors, shapes, edges, and sharpness, all of which help the model learn and accurately classify the images into specific product categories.

4.3 Target Variable

The **target variable** for the classification task is the **product type code** (**prdtypecode**), which is a categorical label used to categorize products into 27 distinct product categories (e.g., **pool accessories**, **books**, **kids' toys**, etc.).

Given the large number of product categories, this makes the classification task both **challenging** and **interesting**, requiring careful attention to both the textual and image data for effective model training.

5 Limitations Observed in the Dataset

Even though the provided dataset is enormous, following are some of the limitations and pre-processing tasks required on this dataset:

1. **Imbalanced Data**: Some product categories have more samples than others, which could lead to model bias where the model favors overrepresented categories. This may impact the model's ability to generalize to rare or less-represented categories.

2. **Image Quality**: Some product images are of low-quality, poorly lit, or blank/solid color background or have an inconsistent background, which may limit the model's ability to extract meaningful features from them.

3. **Missing or Inconsistent Data**: Some products do not have associated images, and certain product descriptions are incomplete or missing, which could reduce the overall dataset size for training.

4. **Multilingual Text Data**: The product descriptions and names are primarily provided in French with some descriptions in other European languages. This poses challenges to train the NLP model with such a dataset and thus it requires language specific pre-processing

5. **Noisy Text Data**: The product descriptions have variations in language and quality, such as spelling errors, missing information, and irrelevant terms (special characters and html tags), which could affect the model's ability to understand and classify the products accurately.

6 Pre-processing and Feature Engineering

The following steps are carried out in the pre-processing and feature engineering for textual and image data

6.1 Textual Data Pre-processing

1. **Data Cleaning**
   Checked for duplicates and missing (NaN) values in the dataset.

2. **Handling Missing Data**
   35.09% missing data was observed in the product description column. Given the dataset's imbalance and the importance of this feature for classification, a new column, **'title_description'**, was generated by concatenating the product designation and description columns.

3. **Text Pre-processing**

   - **Translation**: As the text data is multilingual (French/German and English), GoogleTrans API was used to translate the content in the **'title_description'** column to English language and is stored in the column '**description_english'**.
   - **Data Cleaning**: Before translation, the text data was cleaned by applying Latin encoding to restore special characters, removing HTML tags, special characters, and punctuation using regular expressions, and converting the text to lowercase.

4. **Pre-processing steps for Exploratory Data Analysis**

   - **Tokenization**: Performed using a simple tokenizer and converted all tokens to lowercase.
   - **Stop Words Removal**: Common stop words (e.g., "and", "the", "a", etc.) were removed, as they do not contribute significantly to model learning.
   - **WordCloud**: Created a WordCloud to visualize the top 200 most frequent words in the product title/description.

5. **Pre-processing steps for Text Classification Model**

   - **For machine learning models:**

     ➢ Performed tokenization on the translated text, converted to lowercase, and applied stemming and lemmatization for reducing words to their root forms.
     ➢ Applied vectorization using CountVectorizer or TF-IDF (as these methods do not require normalization or standardization).

- **For transfer learning models:**

  - Used the DistilBert-base-uncased Tokenizer for tokenization on the translated text.
  - As DistilBert-base-uncased is case-insensitive, converting text to lowercase was not critical.

6.2 Image Data Pre-processing

1. **Image Filename Information Extraction**
   Each image filename contains two key pieces of information: image ID and product ID (e.g., image_1263597046_product_3804725264).

2. **Linking Images to Product Categories**
   - The image ID and product ID were extracted from the filenames to link each image to its corresponding product category (or prdtypecode).
   - The following steps were carried out to associate the images with the correct product category

     - Extracted the image path and filename for each image from the image folder.
     - From each filename, extracted the image ID and product ID, storing them in the new 'image_ID' and 'product_ID' columns.

3. **Image Feature Extraction**
   Extracted features such as image size, pixel resolution, sharpness (calculated using Laplacian variance), and storing them in respective new columns.

4. **Handling Blank Images**
   Identified and removed 7 blank/solid images based on pixel value comparison.

6.3 Target Variable

For exploratory data analysis, a new textual category column ('product category') was created for each product type code **(**prdtypecode), reflecting the product categorization used on the Rakuten.

7 Visualizations and Statistics

7.1 Relationship between variables

The relationships between the text variables and the target classes were thoroughly analyzed, revealing significant correlations between specific terms and product categories. Advanced text embedding techniques were employed to effectively capture these relationships. Additionally, image data was incorporated, providing valuable contextual features that contributed to improving classification accuracy.

7.2 Textual Data Exploration

7.2.1 Imbalance in product category or prdtypecode

The dataset is categorized into 27 product type categories, and the number of products in each category is depicted in the figure below.

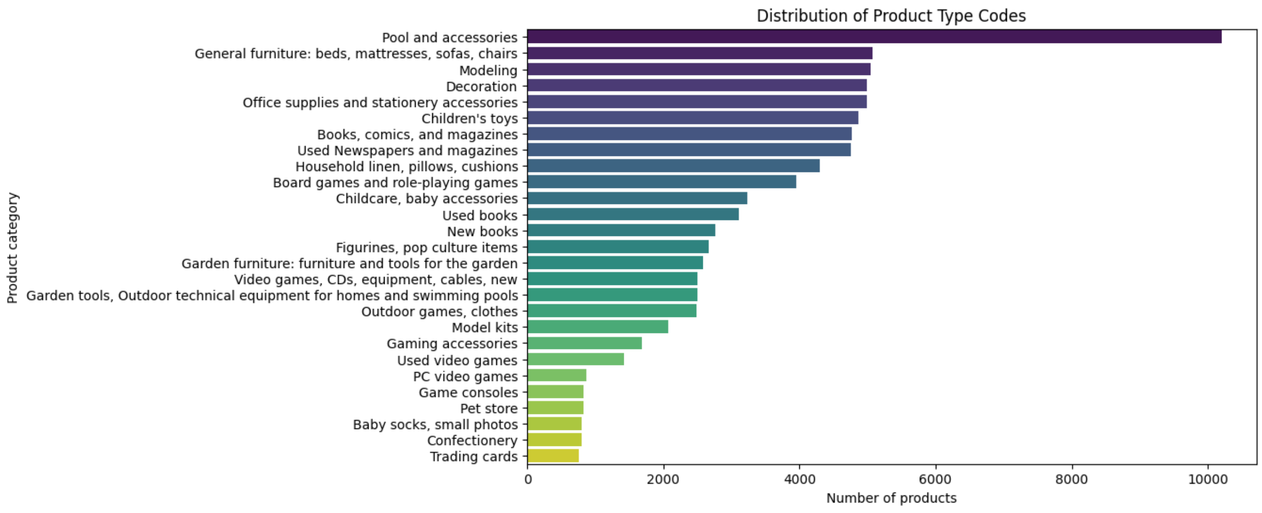


Figure 1: Distribution of product designation length in each category

From the figure, the following observations can be made:

- The **"Pool and Accessories"** category has the highest number of products, with a total of 10,209 items.
- The **"Trading Cards"** category has the least number of products, with only 764 items, among all the other categories.

7.2.2 Distribution of Product Title & Description Text

The box plots below illustrate the distribution of product title (designation) and description text lengths across different product categories in the dataset.

From the below figure, the following observations have been made:

- The **"Games Consoles"** category has the longest designation length (high median value) with the fewest outliers (i.e., only 1).
- The **"Used Newspaper and Magazines"** category exhibits a wide range of designation lengths.
- The **"Used Books"** category has a smaller median designation length but also shows larger variance (outliers) up to 250 characters.
- The **"New Books"** category has the shortest designation length on average.
- Product category "**New Books**" has minimum characters in designation

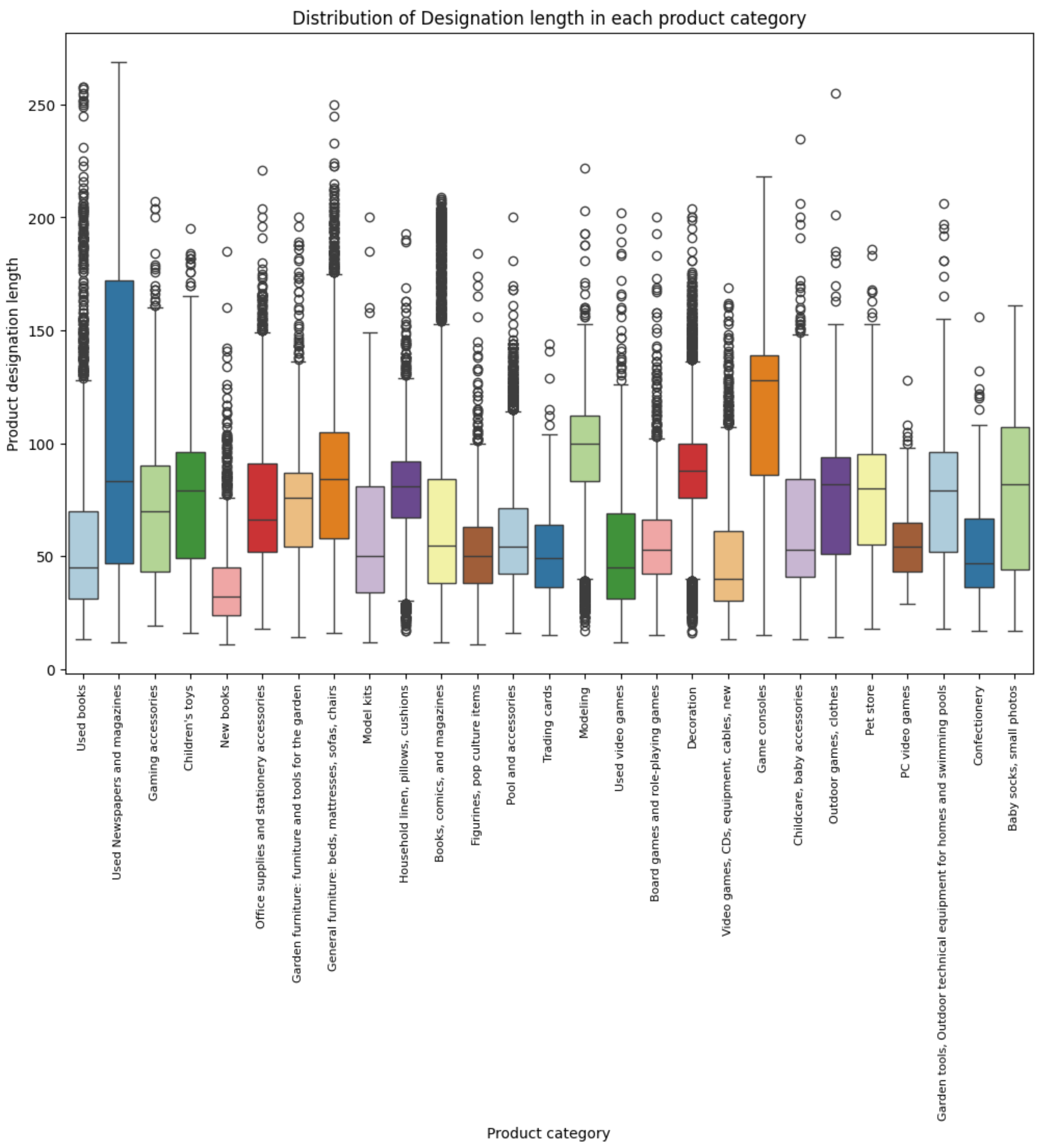


Figure 2: Distribution of product designation length in each category

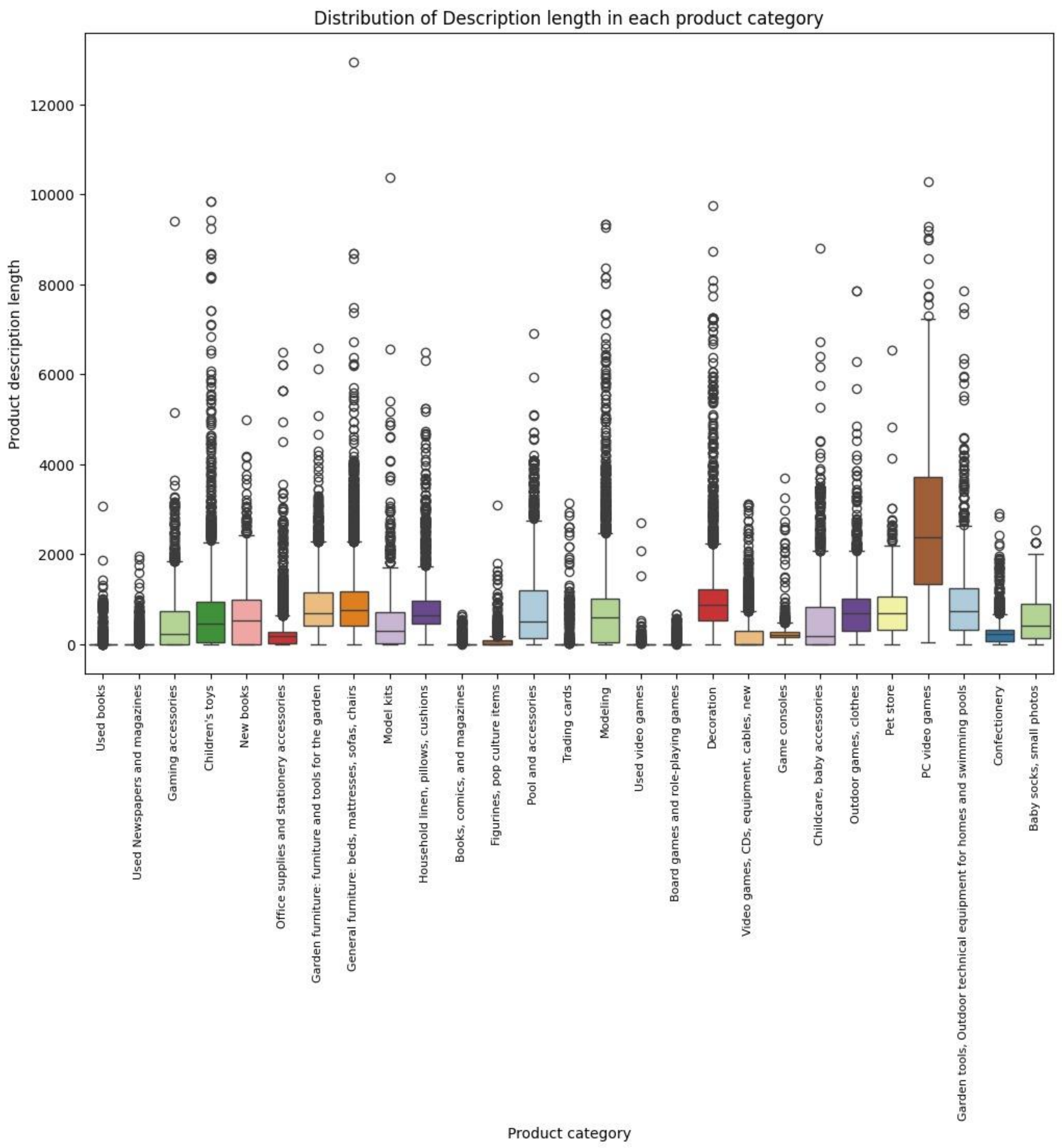


Figure 3: Distribution of product description length in each category

From the above figure, the following observations have been made:

- The **"PC Video Games"** category has the longest description length (high median value), with an average of 2,500 characters.

- The **"Used Newspaper and Magazines"** category again shows a wide range of description lengths.
- Most products in the following categories lack descriptions (median value is 0): **"Used Books," "Used Newspapers and Magazines," "Books, Comics and Magazines," "Used Video Games,"** and **"Board Games, CDs, Equipment, Cables, New."**
- The **"Children's Toys," "Modeling,"** and **"Decoration"** categories contain a large number of outliers in terms of description length.
- The longest description was found in a product from the **"General Furniture, Beds, Mattresses, Sofas, Chairs"** category.

7.2.3 Books and Video Games Categories has less Descriptions

As observed in **Figure 3**, some products do not have descriptions, as indicated by a median value of 0. To further illustrate this, a bar plot is presented to show the proportion of products with and without descriptions across different product categories.

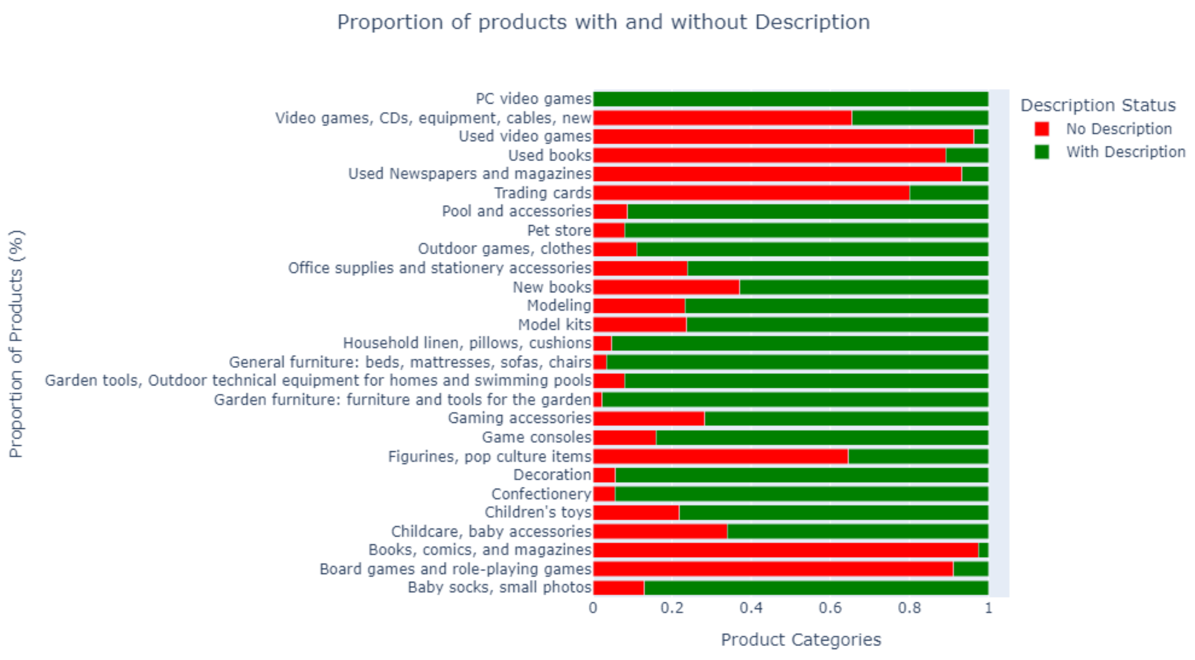


Figure 4: Proportion of products with and without description

The plot clearly shows that certain product categories are missing product descriptions. The **'Board Games and Role-Playing Games'** category has the highest proportion of products without descriptions, while **'PC Video Games'** contains only products with descriptions.

7.2.4 Stainless Steel, package include words are frequently used

After pre-processing the textual data, which involved the removal of special characters, HTML tags, and punctuation (using the regular expressions library), conversion to lowercase, removal of common stop words, stemming, and translation to English, a WordCloud was generated to visualize the most frequent words in the translated product titles for all products.

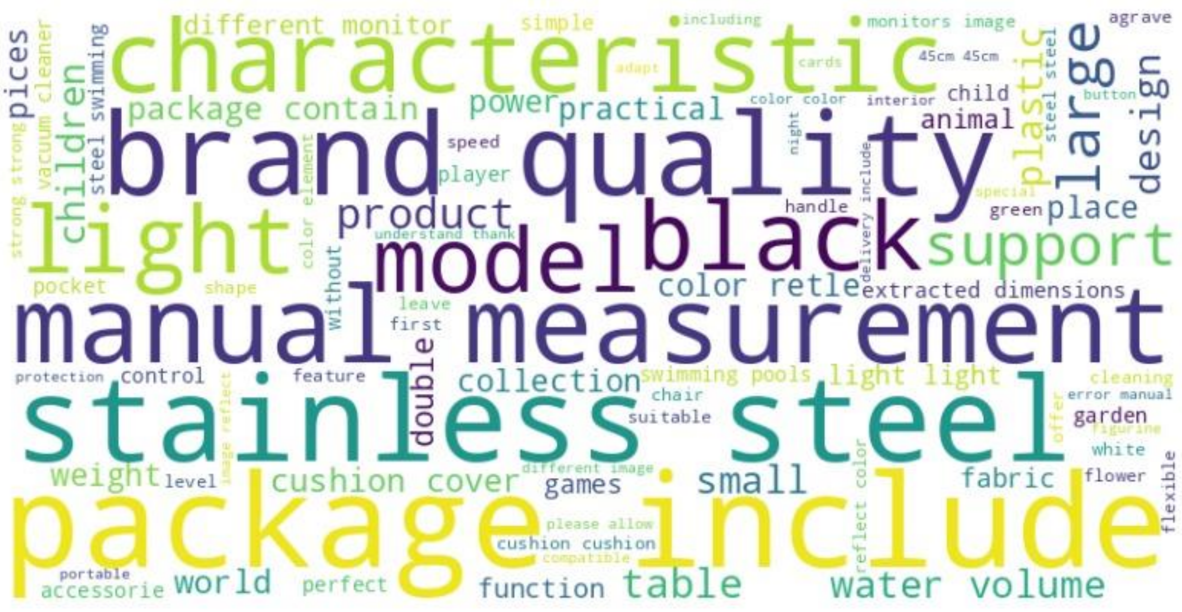


Figure 5: WordCloud of frequent words in product title/description

The WordCloud above highlights the 200 most frequent words in the product titles/descriptions. Notably, the words ‘package,’ ‘include,’ ‘stainless steel,’ and ‘brand’ appear frequently and are among the most repeated words across all product titles and descriptions.

7.3 Image Data Exploration

7.3.1 Removal of 7 blank Image from the Image Dataset

During the exploration of the image dataset, it was discovered that some product images were either blank or consisted of a solid background. To identify and omit these blank or solid images, each image was categorized as "blank" or "not blank" by comparing the pixel values. A bar plot was generated to visualize the distribution of blank images across all product categories.

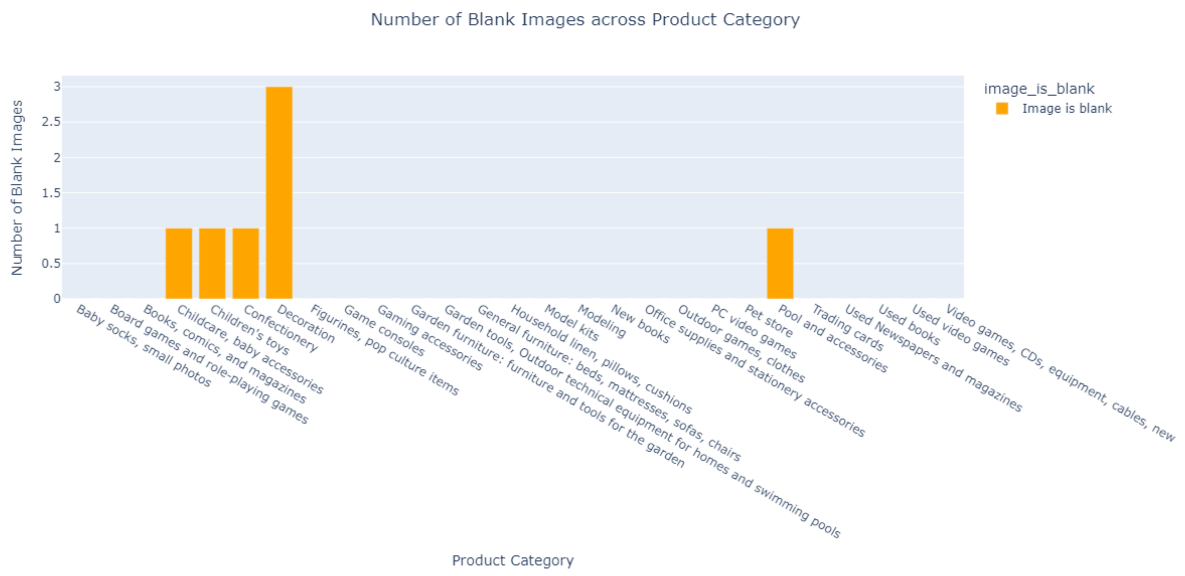


Figure 6: Number of blank images across product categories

7.3.2 91% of Product Images are of High Quality

Since image quality impacts the performance of learning models, a study was conducted to assess the quality of the images. The quality of each image was measured by calculating the Laplacian variance, which measures the rate of change in pixel intensity. Based on the Laplacian variance value, the images were categorized into the following categories: sharp (≥ 200), moderately sharp (199 - 100), slightly blurry (99 - 50), and blurry (< 50). The proportion of images in each sharpness category is illustrated in the pie chart below.

It can be observed from the below pie-chart:

- The majority (91%) exhibited sharp, high-quality characteristics
- Among the images, a small proportion (1.51%, or 1,280 images) were classified as blurry and 5.21% were moderately sharp, and 2.34% were slightly blurry

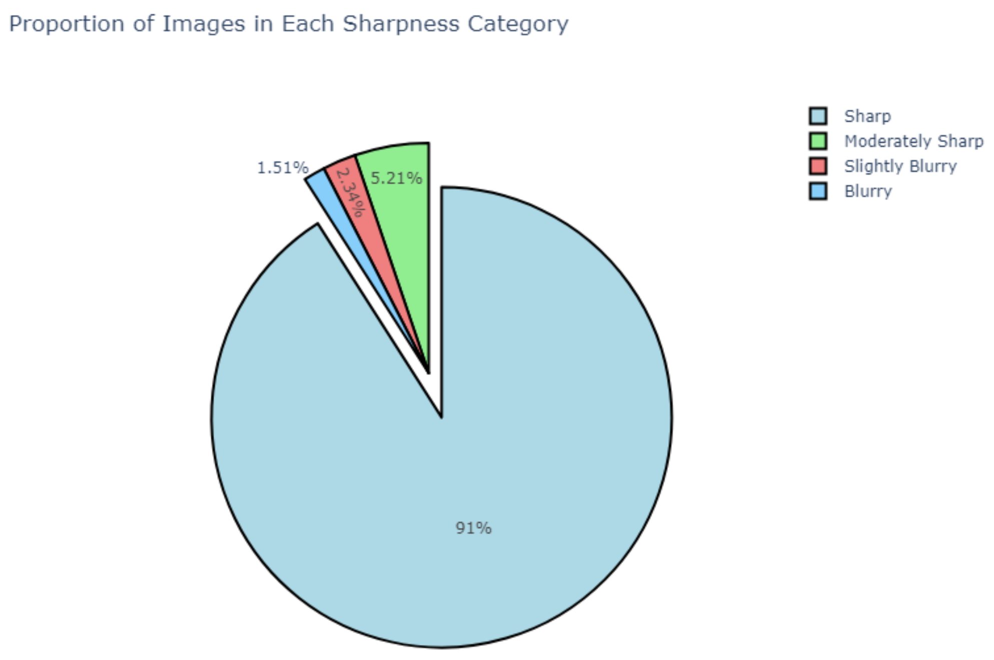


Figure 7: Proportion of Images by sharpness category

7.3.3 PC Video Games category contains highest proportion of sharp Images

It can be observed from the Box-plot below:

- The product category **'Garden Tools, Outdoor Technical Equipment for Homes and Swimming Pools'** has the highest sharpness score (i.e., 60,000)
- The **'Pool and Accessories'** category has the lowest image sharpness
- The **'PC Video Games'** category exhibits the highest median value for image sharpness scores, indicating that a larger proportion of images in this category are of higher quality

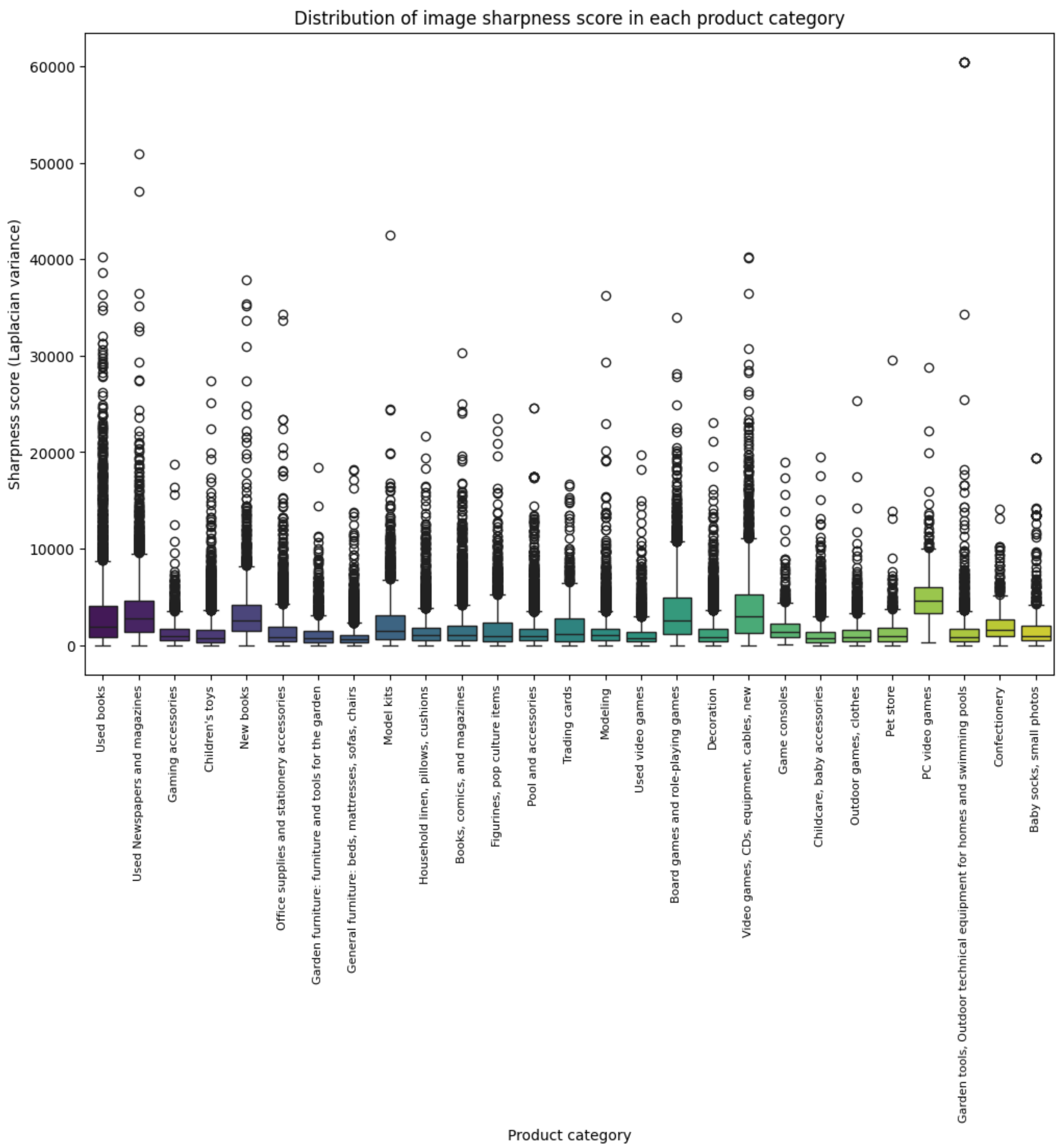


Figure 8: Distribution of Image sharpness across all product categories

8 Features and Target variables

For Text classification model, the column **'description_english'** is used as the feature and the column **'prdtypcode'** is used as the target variable.

For image classification model, the **'image_path'** which directs to the image file was used as feature and **'prdtypecode'** as target variable.

9 Handling Class Imbalance

For the textual dataset, after pre-processing, the class imbalance is addressed by applying the **SMOTE** (Synthetic Minority Over-Sampling Technique) method for machine learning models, and the **class_weight** method is used for transfer learning models.

For the image dataset, the imbalance is tackled by generating three augmented images for each image in the minority classes. After splitting the dataset into training, validation, and test sets, **data augmentation** is applied to the training set, irrespective of class imbalance, to enhance classification performance.

10 Classification of the Problem

The Rakuten E-commerce project relates to **multimodal classification**, combining both **text classification** and **image classification** tasks. The **DistilBERT** model is used for text classification, where it classifies textual data into predefined categories. The **VGG16** model is applied for image classification, categorizing images into various classes. This approach enables the processing and classification of both text and image data for a more comprehensive analysis and decision-making.

10.1 Metric used to compare the Models

The main performance metrics used to compare the models are **validation accuracy**, **validation loss**, and **F1-score**. These metrics are chosen to provide a comprehensive evaluation of model performance. **Validation accuracy** measures the overall correctness of the model on the validation set, while **validation loss** helps assess how well the model minimizes error during training. The **F1-score** is particularly useful for evaluating performance on imbalanced datasets, as it considers both precision and recall, providing a balanced measure of the model's ability to correctly classify positive and negative instances.

11 Model Choice and Optimization

11.1 Parameter Optimization Techniques

For machine learning models such as Logistic Regression, SVC, Random Forest Classifier (RFC), K-Nearest Neighbors (KNN), Multinomial Naive Bayes (NB), XGBoost, and Gaussian Naive Bayes (NB), GridSearchCV was employed to optimize the hyperparameters. In contrast, for transfer learning models like DistilBERT-base-uncased (NLP model), and VGG16 and ResNet50 (CNN models), various optimization techniques were applied, including different learning rates and callback functions such as EarlyStopping, ReduceLROnPlateau, and Learning Rate Decay, to fine-tune the performance of the text and image classification models.

11.2 Algorithms Tested for Text and Image Classification

For **Text Classification**, several pre-processing techniques were applied, including stemming and lemmatization. Both datasets were split into a training set (80%) and a test set (20%) using the **train_test_split** method from Scikit-learn. To address the imbalance in the dataset, the **SMOTE** resampling method was used for simple machine learning models, while the **class_weight** method was employed for transfer learning models. The results from the stemmed dataset showed lower accuracy and F1-score compared to the lemmatized dataset. Various trails with different Machine learning models were carried out and the results can be referred in Appendix table(a).

As a result, the lemmatized dataset was selected for further experimentation. Two vectorization methods, **CountVectorizer** and **TF-IDF**, were used with various machine learning models and transfer learning models. Among the models tested with **TF-IDF vectorization**, **SVC**, **Logistic Regression**, and **Random Forest Classifier** delivered the best accuracy and F1-scores, outperforming others like **KNN**, **Multinomial Naive Bayes**, **XGBoost**, and **GaussianNB**.

Table 1. Machine learning models and results for text classification

Tokenization/Vectorization	Models	Validation accuracy	F1-score
Simple Tokenizer, Countvectorizer	SVC (Support Vector Classifier)	0.6477	0.6579
Simple Tokenizer, Countvectorizer	LogisticRegression	0.6864	0.6923
Simple Tokenizer, Countvectorizer	RFC	0.6741	0.6744
Simple Tokenizer, Countvectorizer	Multinomial NB	0.6652	0.6643
Simple Tokenizer, Countvectorizer	XGBoosting	0.6100	0.6333
Simple Tokenizer, Countvectorizer	KNN	0.5313	0.5471
Simple Tokenizer, TF-IDF Vectorizer (features= 5000)	SVC	0.7555	0.7586
Simple Tokenizer, TF-IDF Vectorizer (features= 5000)	LogisticRegression (max_iter=200)	0.7333	0.7350
Simple Tokenizer, TF-IDF Vectorizer (features= 5000)	RFC (n_estimaters=50)	0.7148	0.7176
Simple Tokenizer, TF-IDF Vectorizer (features= 5000)	KNN	0.5457	0.5682
Simple Tokenizer, TF-IDF Vectorizer (features= 5000)	GaussianNB	0.5253	0.5299

* SVC - Support Vector Classifier, RFC – Random Forest Classifier, KNN – K-Nearest Neighbor

11.3 Testing of Advanced Models for Text Classification

The advanced models such as deep learning models were tested for both text and image classification. For text classification, **BERT variants** (DistilBERT, ALBERT, and RoBERTa) were used due to their superior performance in natural language processing tasks. Here we used 70% or 80% of dataset for training, and the remaining was split into half, where the first half is used for validation and second half is used for testing the model. Of the three, **DistilBERT** yielded the best performance with an accuracy of 0.83 as shown in Table 2.

Table 2. NLP BERT variants and results for text classification

Tokenization/Vectorization	Models	Validation accuracy
DistilBERT-base-uncased Tokenizer	DistilBERT-base-uncased (learning rate= 3e-5) (Epoch 4)	**0.8300**
ALBERT Tokenizer	Albert(learning rate= 1e-5) (Epoch 3)	0.7800
RoBERTa Tokenizer	Roberta(learning rate=1e-5) (Epoch 3)	0.8087

11.4 DistilBERT outperformed other BERT variants

Various learning rates, batch sizes, and train-validation-test split sizes were experimented to optimize the DistilBERT-base-uncased model and improve its F1 score. The results, including accuracy and F1 scores from the optimization process, are summarized in Table 2.

Table 3. Optimization results for DistilBERT-base-uncased model

Tokenization/Vectorization	Models	Test accuracy	F1-score
DistilBERT-base-uncased Tokenizer	DistilBERT-base-uncased (learning rate= 4e-5), batch =32(70, 15, 15 split)	0.8321	0.8174
DistilBERT-base-uncased Tokenizer	DistilBERT-base-uncased (learning rate= 1e-6), batch =32(80, 10, 10 split)	0.744	0.6817
DistilBERT-base-uncased Tokenizer	DistilBERT-base-uncased (learning rate= 4e-5), batch =16(70, 15, 15 split)	0.835	0.8209
DistilBERT-base-uncased Tokenizer	DistilBERT-base-uncased (learning rate= 3e-5), batch =16(80, 10, 10 split)	0.84	0.8236
DistilBERT-base-uncased Tokenizer	DistilBERT-base-uncased (learning rate= 3e-5), batch =32(80, 10, 10 split)	0.8387	0.8236
DistilBERT-base-uncased Tokenizer	DistilBERT-base-uncased (learning rate= 4e-5), batch =32(80, 10, 10 split)	0.8434	0.8294
DistilBERT-base-uncased Tokenizer	**DistilBERT-base-uncased (learning rate= 5e-5), batch =32(80, 10, 10 split)**	**0.8474**	**0.8382**

Among the different configurations, the best performance was achieved with an F1 score of 83.82, using a data split of 80% train, 10% validation, and 10% test, learning rate of 5e-5, a batch size of 32, and epoch 10.

11.5 Advanced Models for Image Classification

For image classification, **CNN models** like **ResNet50** and **VGG16** were utilized. These deep learning models were chosen because they are pre-trained on large datasets like **ImageNet**, which contains millions of images and thousands of classes, making them more efficient and accurate for classification tasks.

11.6 VGG16 performed better for Image Classification

For **Image Classification**, we used 80% of dataset for training and the remaining was split into half, where the first half is used for validation and second half is used for testing the model. The initial approach used a custom **CNN model**, but its performance was suboptimal with a very low validation accuracy (< 0.20). As a result, pre-trained transfer learning models such as **ResNet50** and **VGG16** were employed, both with their original and reduced base layers. To address class imbalance in the dataset, **ImageDataGenerator** was used to generate augmented data for the minority classes in the training set, following the **train_test_split**. Among the tested models, the results from **ResNet50** and **VGG16** showed varying levels of accuracy, with **VGG16** outperforming **ResNet50** in terms of accuracy, F1-score and validation loss under specific conditions. The validation accuracy and loss obtained from the ResNet50 and VGG16 models with different base layers and learning rates is shown in the table below.

Table 4. Image classification models and results

Model	**learning rate**	**Accuracy**	**Accuracy loss**
Resnet50 (Epoch 10)	0.0001	0.2668	2.5473
Resnet50(base_layer:-10) (Epoch 10)	2.50E-05	0.414	2.0206
Resnet50(base_layer:-10) (Epoch 10)	3.13E-06	0.4316	1.9658
Resnet50(base_layer:-10) (Epoch 10)	1.95E-07	0.4345	1.9563
VGG16 (Epoch 25)	1e-4	0.5500	1.5377

Among the ResNet50 and VGG16 models, the best accuracy 0.5500 and F1 score **0.5374** were achieved using the VGG16 model with a learning rate of 1e-4, a batch size of 16, and 25 epochs.

12 Interpretation of Results

12.1 Misclassification Statistics of Text Classification Model

The analysis of misclassified records provides key insights into the model's behavior. The confusion matrix obtained from the test dataset evaluation shown in Figure 10.

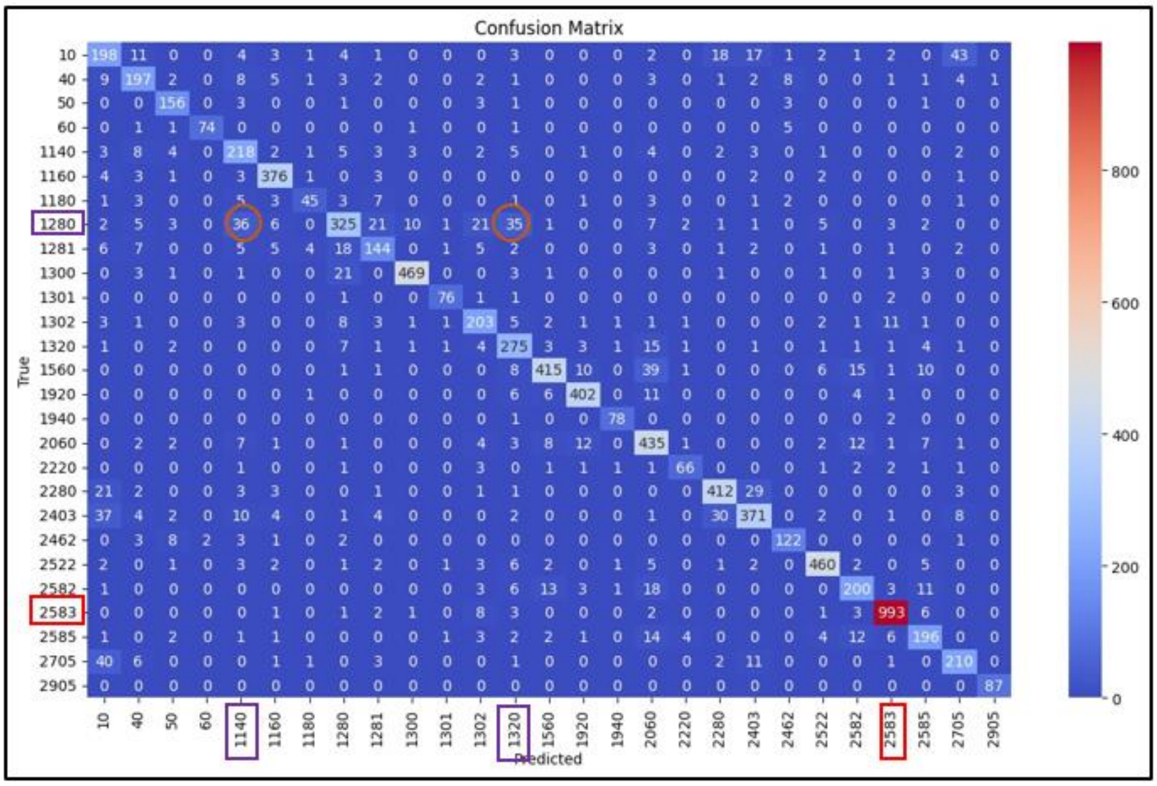


Figure 9: Confusion matrix for text classification model

12.1.1 Classification Statistics

The highest number of correctly classified products was observed in prdtypcode 2583 ("**Pool and Accessories**"). This category represents the majority class, with a total of 10,209 records in the dataset.

12.1.2 Misclassification Statistics

Analysing misclassified records provides valuable insights into the model's performance and limitations. Among the misclassified records, the Children's Toys category (prdtypecode: 1280) has the highest number of errors, with 162 instances. Most of these are misclassified into the Childcare and Baby Accessories category (prdtypecode: 1320). This misclassification is understandable since both categories deal with children's products.

Interestingly, the Children's Toys category is not a minor class and it accounts for a significant 4,870 records in the dataset. From the confusion matrix, another notable observation is that the highest number of products misclassified into a single category belongs to the General Furniture category (prdtypecode: 1560). Specifically, 39 products were incorrectly classified as part of the Decoration category (prdtypecode: 2060).

12.2 Mismatch Analysis and Image Comparison:

The confusion matrix obtained for the test dataset from the VGG16 Image model is given below:

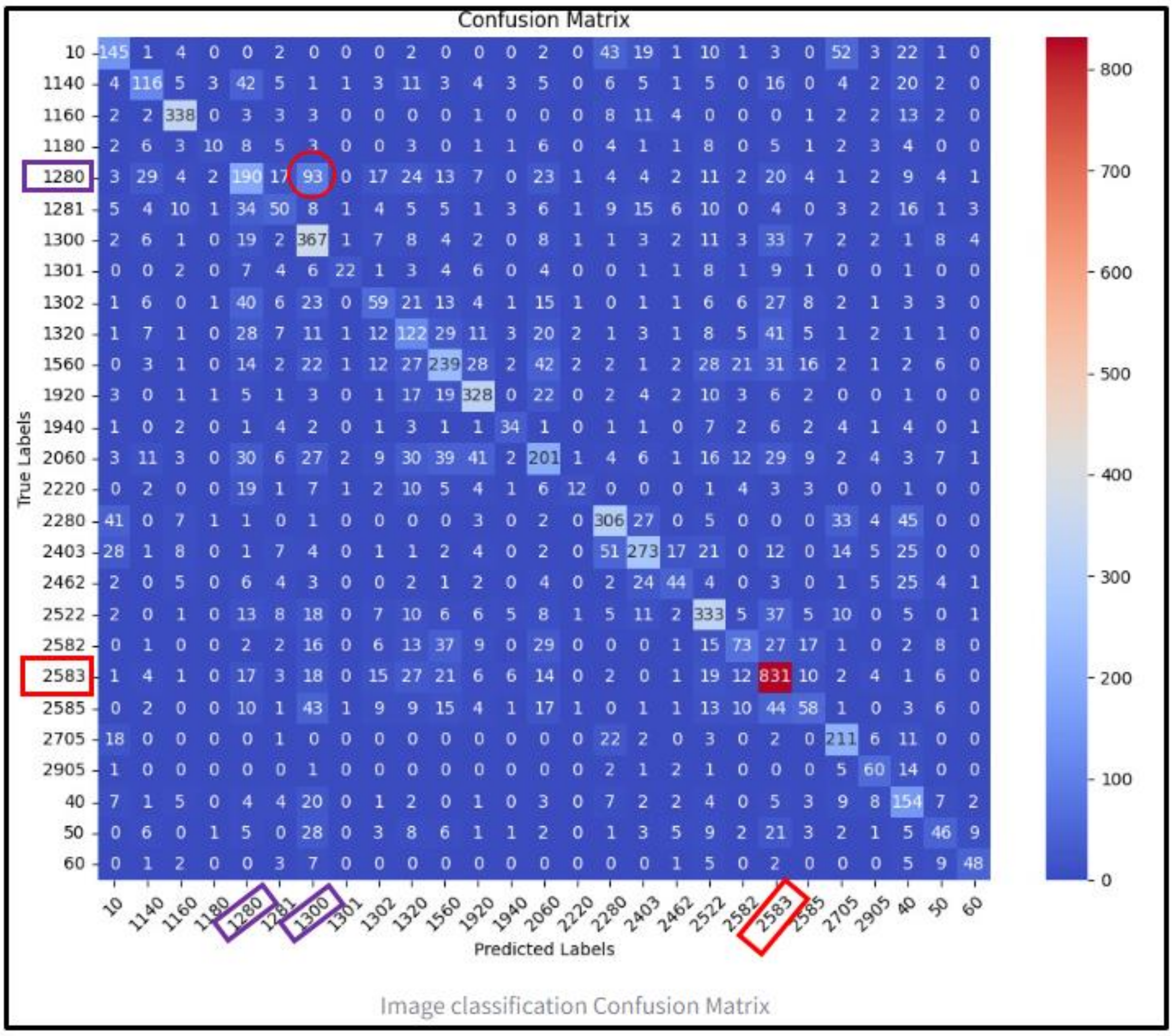


Figure 10: Confusion matrix for Image classification model

12.2.1 Classification Statistics

The highest number of correctly classified products was observed in prdtypcode 2583 ("**Pool and Accessories**"). This category represents the majority class, with a total of 10,209 records in the dataset.

12.2.2 Misclassification Statistics

The class with the highest number of mismatches is Class 1280 (Children's Toys), with a total of 606 mismatches. Notably, most of these mismatches are due to the model incorrectly classifying 93 images as belonging to Class 1300 (Modeling), which suggests that the model has difficulty distinguishing between children's toys and models (figurines). Despite Class 1280 being a majority class, this indicates that there might be significant similarities in features between the two classes, such as shape, color, or context within the images. This observation can guide further analysis of the visual characteristics of both categories to address the issue.

On the other hand, Class 60 (Game Consoles), which is a minority class with only 832 instances, has the lowest number of mismatches at 58 mismatches. This indicates that the model is highly effective at recognizing game consoles, despite their minority status in the dataset. This could be due to distinctive features that help the model easily differentiate game consoles from other categories.

12.3 Grad-CAM (Gradient-weighted Class Activation Mapping)

Grad-CAM (Gradient-weighted Class Activation Mapping) is a visualization technique that helps interpret and understand the decisions made by convolutional neural networks (CNNs). It generates a heatmap that highlights the important regions in an image that contribute most to the model's prediction. It works by calculating the gradients of the target class with respect to the feature maps of the last convolutional layer in the model, then using these gradients to weight the feature maps.
The result is a heatmap that is overlaid on the image, revealing the areas the model focused on when making its prediction.

The highlighted regions for some of the images is shown below and the area highlighted will differ across each image.

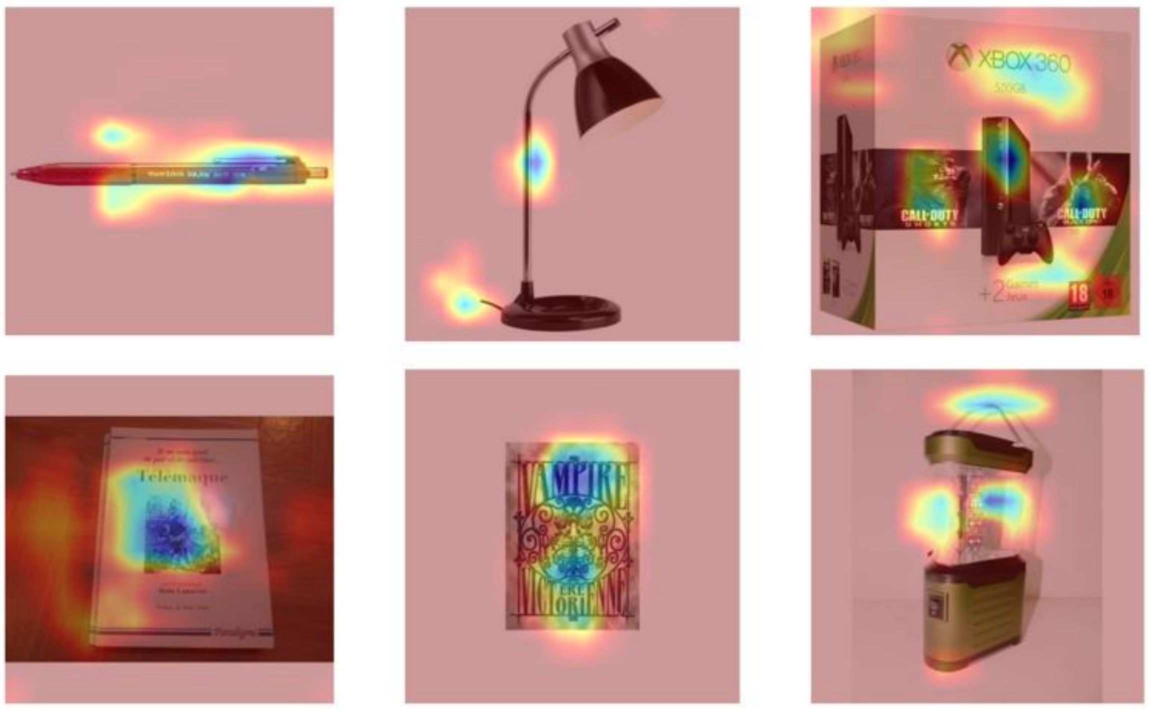


Figure 11: Grad-CAM examples

13 Fusion model

13.1 Multimodal Model Integration

To further improve classification accuracy, the text and image classifiers were integrated into a unified multimodal system. The late fusion technique was employed to combine the predictions of both models, leveraging the complementary strengths of text and image data. This approach led to better overall performance and contributed significantly to the success of the project.

13.2 Feature Fusion

Feature fusion is a technique where multiple feature sets from different models or sources are combined to improve model performance. In the context of machine learning, feature fusion can be used to integrate different types of features, such as those derived from text, image, or other modalities, to enhance the predictive power of the classifier.

13.3 Feature Fusion using Voting Classifier:

The steps used in feature fusion of DistilBERT-base-uncased and VGG16 model are listed below:

13.4 Model Selection for Multimodal Fusion

From both text and image classification models, the models with highest F1-score were chosen for building a fusion model. Therefore, from text classification models, DistilBERT-base-uncased model with a learning rate of 4e-5 is selected. Likewise, for image classification VGG16 with learning rate of 1e-6 is chosen for the fusion.

Since a higher F1-score is obtained on the text classification model (**0.8343**) than image classification model (0.5374), normalized weights will be added to the later fusion model to have better accuracy as text model performs relatively better. The technique used for the fusion of both text classification and image classification model is late fusion with voting classifier using soft voting technique.

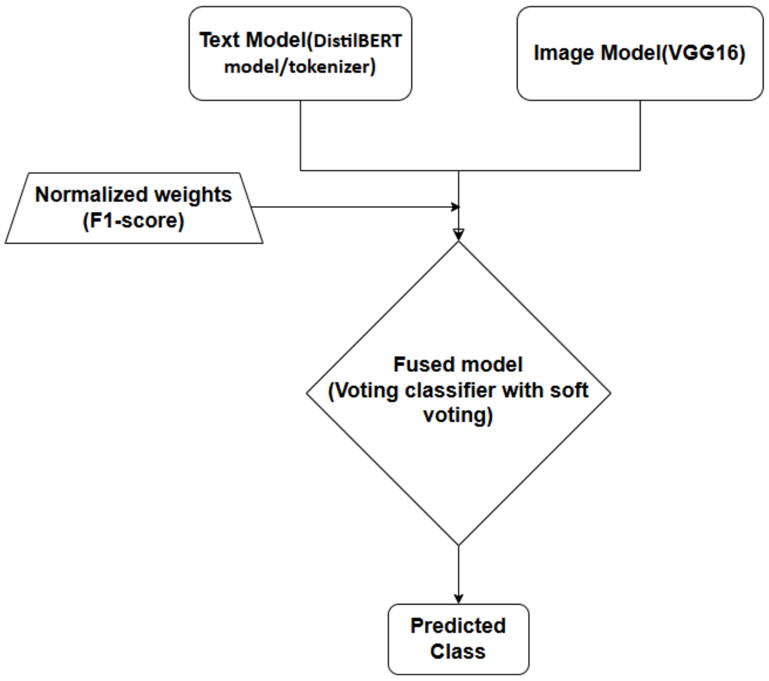

Text Model(DistilBERT model/tokenizer)
Image Model(VGG16)
Normalized weights (F1-score)
Fused model (Voting classifier with soft voting)
Predicted Class

Figure 12: Multimodal development with Late fusion and normalized weights

14 Results Comparison

The models chosen as benchmarks for this project were a simple CNN classifier for text classification and ResNet50 for image classification. Instead of simply improving upon these models, a more detailed exploration was conducted to achieve the project's goals, which involved experimenting with a variety of models, including both traditional machine learning techniques and advanced deep learning architectures for text and image classification.

Multiple models were evaluated based on their accuracy and F1 scores, and the most effective models for each modality (text and image) were selected. Afterward, hyperparameters for the chosen models were fine-tuned to enhance their performance. It is important to note that the optimized model did not undergo architectural changes; the improvements were primarily in the pre-processing steps.

Table 5. Model Performance Comparison

	F1-Score -Text Model	F1-Score Image Model
Benchmark	0.8113	0.5534
Our Models	**0.8343**	0.5374

Notably, the text classification model (DistilBERT-base-uncased) exhibited an improvement in classification accuracy, particularly for minority classes (those comprising less than 10% of the majority class). This includes categories such as prdtypcode 60 (Game Consoles), 2220 (Pet Store), 1940 (Confectionery), 1301 (Baby Socks, Small Photos), 1180 (Trading Cards), and 2905 (PC Video Games). This suggests that the model has enhanced its ability to correctly classify less-represented categories.

In contrast, despite incorporating image augmentation for minor classes during pre-processing, the image classification model did not show significant improvements for these minority classes. Some of the minor classes remained misclassified, indicating that the current augmentation strategy was not effective for all underrepresented categories.

14.1 Application and Future Potential of the Model

The multimodal classification system developed in this project has broad applications, particularly in areas such as product categorization for e-commerce platforms and image-based content tagging. By effectively integrating text and image data, the system can offer more accurate product classifications, benefiting businesses and enhancing the user experience.

15 Challenges encountered during the project

- Training the models took longer than anticipated due to our hardware computational limitations and huge datasets, which resulted in more computational time to test the model

- There was a lag between the module topic and respective project related task, which led to exploration of the modules before the actual sprints

16 Future Scope of the Models

For text classification, employing advanced models combined with varied pre-processing techniques can significantly enhance prediction accuracy.

In the case of the image classification model, most misclassifications arise due to the presence of similar products in multiple classes. Specifically, some classes are subcategories of broader categories, leading to overlap and confusion. To improve prediction accuracy, these subcategories should be merged under a single general category.

For instance, classes such as 10 (Used Books), 2280 (Used Newspapers and Magazines), 2403 (Books, Comics, and Magazines), and 2705 (New Books) all include images of books, making it challenging for the model to correctly distinguish between them.

Enhancing the performance of the image classification model can involve additional pre-processing steps, such as reducing image dimensions, increasing image augmentation, applying image cropping, and utilizing models with deeper CNN architectures, which are better equipped to achieve higher accuracy.

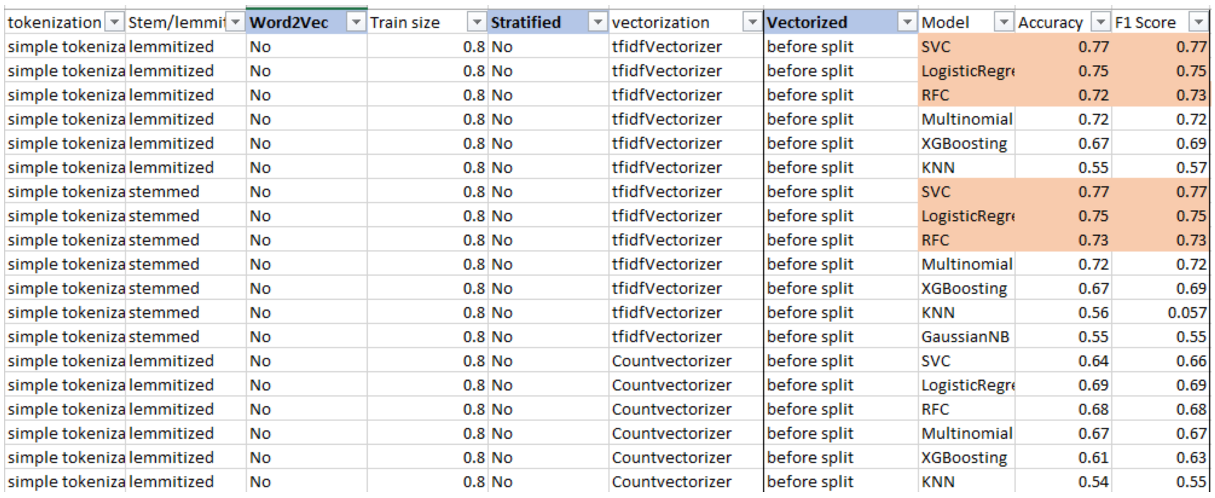
17 Bibliography

- "Rakuten set to launch a wireless mobile network in Japan - Tamebay". tamebay.com. 17 May 2018. Archived from the original on 2020-06-16. Retrieved 2019-05-22.
- "About Us". Rakuten Group, Inc. Retrieved 2023-02-04.
- "Financial Data". Rakuten Group, Inc. Retrieved 2023-02-04.
- Evelyn M. Rusli, "Pinterest on Wish List of Rakuten, Japan's Amazon", The New York Times, July 12, 2012.
- Rakuten abandonne la marque PriceMinister", Le Monde, November 6, 2019

18 Appendices

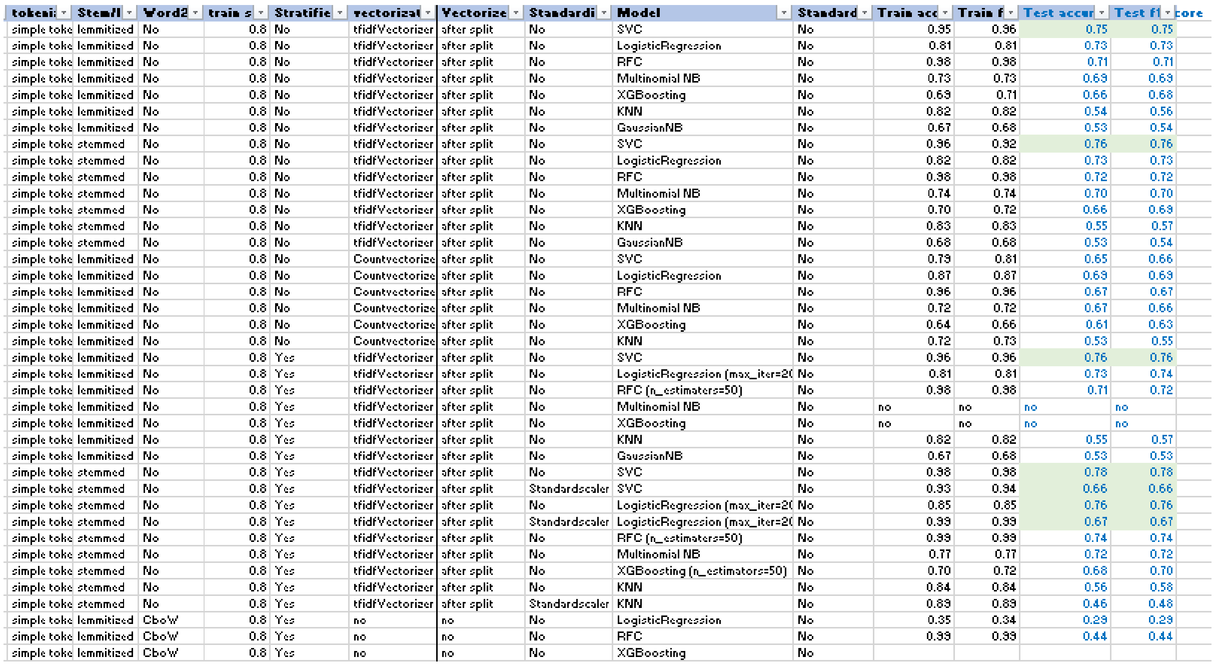
18.1 Machine learning models for Text classification

Table a. Machine learning models for Text classification



tokenization	Stem/lemmit	Word2Vec	Train size	Stratified	vectorization	Vectorized	Model	Accuracy	F1 Score
simple tokeniza	lemmitized	No	0.8	No	tfidfVectorizer	before split	SVC	0.77	0.77
simple tokeniza	lemmitized	No	0.8	No	tfidfVectorizer	before split	LogisticRegre	0.75	0.75
simple tokeniza	lemmitized	No	0.8	No	tfidfVectorizer	before split	RFC	0.72	0.73
simple tokeniza	lemmitized	No	0.8	No	tfidfVectorizer	before split	Multinomial	0.72	0.72
simple tokeniza	lemmitized	No	0.8	No	tfidfVectorizer	before split	XGBoosting	0.67	0.69
simple tokeniza	lemmitized	No	0.8	No	tfidfVectorizer	before split	KNN	0.55	0.57
simple tokeniza	stemmed	No	0.8	No	tfidfVectorizer	before split	SVC	0.77	0.77
simple tokeniza	stemmed	No	0.8	No	tfidfVectorizer	before split	LogisticRegre	0.75	0.75
simple tokeniza	stemmed	No	0.8	No	tfidfVectorizer	before split	RFC	0.73	0.73
simple tokeniza	stemmed	No	0.8	No	tfidfVectorizer	before split	Multinomial	0.72	0.72
simple tokeniza	stemmed	No	0.8	No	tfidfVectorizer	before split	XGBoosting	0.67	0.69
simple tokeniza	stemmed	No	0.8	No	tfidfVectorizer	before split	KNN	0.56	0.057
simple tokeniza	stemmed	No	0.8	No	tfidfVectorizer	before split	GaussianNB	0.55	0.55
simple tokeniza	lemmitized	No	0.8	No	Countvectorizer	before split	SVC	0.64	0.66
simple tokeniza	lemmitized	No	0.8	No	Countvectorizer	before split	LogisticRegre	0.69	0.69
simple tokeniza	lemmitized	No	0.8	No	Countvectorizer	before split	RFC	0.68	0.68
simple tokeniza	lemmitized	No	0.8	No	Countvectorizer	before split	Multinomial	0.67	0.67
simple tokeniza	lemmitized	No	0.8	No	Countvectorizer	before split	XGBoosting	0.61	0.63
simple tokeniza	lemmitized	No	0.8	No	Countvectorizer	before split	KNN	0.54	0.55

Table b. Models with vectorized feature after splitting test and train data



tokeni:	Stem/l	Word2	train s	Stratifie	vectorizat	Vectorize	Standardi	Model	Standard	Train acc	Train f	Test accur	Test f1	core
simple toke	lemmitized	No	0.8	No	tfidfVectorizer	after split	No	SVC	No	0.95	0.96	0.75	0.75	
simple toke	lemmitized	No	0.8	No	tfidfVectorizer	after split	No	LogisticRegression	No	0.81	0.81	0.73	0.73	
simple toke	lemmitized	No	0.8	No	tfidfVectorizer	after split	No	RFC	No	0.98	0.98	0.71	0.71	
simple toke	lemmitized	No	0.8	No	tfidfVectorizer	after split	No	Multinomial NB	No	0.73	0.73	0.69	0.69	
simple toke	lemmitized	No	0.8	No	tfidfVectorizer	after split	No	XGBoosting	No	0.69	0.71	0.66	0.68	
simple toke	lemmitized	No	0.8	No	tfidfVectorizer	after split	No	KNN	No	0.82	0.82	0.54	0.56	
simple toke	lemmitized	No	0.8	No	tfidfVectorizer	after split	No	GaussianNB	No	0.67	0.68	0.53	0.54	
simple toke	stemmed	No	0.8	No	tfidfVectorizer	after split	No	SVC	No	0.96	0.92	0.76	0.76	
simple toke	stemmed	No	0.8	No	tfidfVectorizer	after split	No	LogisticRegression	No	0.82	0.82	0.73	0.73	
simple toke	stemmed	No	0.8	No	tfidfVectorizer	after split	No	RFC	No	0.98	0.98	0.72	0.72	
simple toke	stemmed	No	0.8	No	tfidfVectorizer	after split	No	Multinomial NB	No	0.74	0.74	0.70	0.70	
simple toke	stemmed	No	0.8	No	tfidfVectorizer	after split	No	XGBoosting	No	0.70	0.72	0.66	0.69	
simple toke	stemmed	No	0.8	No	tfidfVectorizer	after split	No	KNN	No	0.83	0.83	0.55	0.57	
simple toke	stemmed	No	0.8	No	tfidfVectorizer	after split	No	GaussianNB	No	0.68	0.68	0.53	0.54	
simple toke	lemmitized	No	0.8	No	Countvectorize	after split	No	SVC	No	0.79	0.81	0.65	0.66	
simple toke	lemmitized	No	0.8	No	Countvectorize	after split	No	LogisticRegression	No	0.87	0.87	0.69	0.69	
simple toke	lemmitized	No	0.8	No	Countvectorize	after split	No	RFC	No	0.96	0.96	0.67	0.67	
simple toke	lemmitized	No	0.8	No	Countvectorize	after split	No	Multinomial NB	No	0.72	0.72	0.67	0.66	
simple toke	lemmitized	No	0.8	No	Countvectorize	after split	No	XGBoosting	No	0.64	0.66	0.61	0.63	
simple toke	lemmitized	No	0.8	No	Countvectorize	after split	No	KNN	No	0.72	0.73	0.53	0.55	
simple toke	lemmitized	No	0.8	Yes	tfidfVectorizer	after split	No	SVC	No	0.96	0.96	0.76	0.76	
simple toke	lemmitized	No	0.8	Yes	tfidfVectorizer	after split	No	LogisticRegression (max_iter=2(	No	0.81	0.81	0.73	0.74	
simple toke	lemmitized	No	0.8	Yes	tfidfVectorizer	after split	No	RFC (n_estimaters=50)	No	0.98	0.98	0.71	0.72	
simple toke	lemmitized	No	0.8	Yes	tfidfVectorizer	after split	No	Multinomial NB	No	no	no	no	no	
simple toke	lemmitized	No	0.8	Yes	tfidfVectorizer	after split	No	XGBoosting	No	no	no	no	no	
simple toke	lemmitized	No	0.8	Yes	tfidfVectorizer	after split	No	KNN	No	0.82	0.82	0.55	0.57	
simple toke	lemmitized	No	0.8	Yes	tfidfVectorizer	after split	No	GaussianNB	No	0.67	0.68	0.53	0.53	
simple toke	stemmed	No	0.8	Yes	tfidfVectorizer	after split	No	SVC	No	0.98	0.98	0.78	0.78	
simple toke	stemmed	No	0.8	Yes	tfidfVectorizer	after split	Standardscaler	SVC	No	0.93	0.94	0.66	0.66	
simple toke	stemmed	No	0.8	Yes	tfidfVectorizer	after split	No	LogisticRegression (max_iter=2(	No	0.85	0.85	0.76	0.76	
simple toke	stemmed	No	0.8	Yes	tfidfVectorizer	after split	Standardscaler	LogisticRegression (max_iter=2(	No	0.99	0.99	0.67	0.67	
simple toke	stemmed	No	0.8	Yes	tfidfVectorizer	after split	No	RFC (n_estimaters=50)	No	0.99	0.99	0.74	0.74	
simple toke	stemmed	No	0.8	Yes	tfidfVectorizer	after split	No	Multinomial NB	No	0.77	0.77	0.72	0.72	
simple toke	stemmed	No	0.8	Yes	tfidfVectorizer	after split	No	XGBoosting (n_estimators=50)	No	0.70	0.72	0.68	0.70	
simple toke	stemmed	No	0.8	Yes	tfidfVectorizer	after split	No	KNN	No	0.84	0.84	0.56	0.58	
simple toke	stemmed	No	0.8	Yes	tfidfVectorizer	after split	Standardscaler	KNN	No	0.89	0.89	0.46	0.48	
simple toke	lemmitized	CboW	0.8	Yes	no	no	No	LogisticRegression	No	0.35	0.34	0.29	0.29	
simple toke	lemmitized	CboW	0.8	Yes	no	no	No	RFC	No	0.99	0.99	0.44	0.44	
simple toke	lemmitized	CboW	0.8	Yes	no	no	No	XGBoosting	No					

Table c. Models with Stratified split and vectorization

tokenization	Stem/lemmi	Word2Vec	train size	Stratified	vectorization	Vectorization	Model	Train accuracy	Train f1 scor	Test accuracy	Test f1 score
simple toker	lemmitized	No	0.8	Yes	tfidfVectorizer	after split	SVC	0.9552	0.9562	0.7555	0.7586
simple toker	lemmitized	No	0.8	Yes	tfidfVectorizer	after split	LogisticRegressi	0.8108	0.8115	0.7333	0.735
simple toker	lemmitized	No	0.8	Yes	tfidfVectorizer	after split	RFC	0.9758	0.9766	0.7148	0.7176
simple toker	lemmitized	No	0.8	Yes	tfidfVectorizer	after split	KNN	0.8196	0.8234	0.5457	0.5682
simple toker	lemmitized	No	0.8	Yes	tfidfVectorizer	after split	GaussianNB	0.6716	0.6795	0.5253	0.5299
simple toker	stemmed	No	0.8	Yes	tfidfVectorizer	after split	LogisticRegressi	0.8482	0.8482	0.7571	0.758
simple toker	stemmed	No	0.8	Yes	tfidfVectorizer	after split	RFC	0.6854	0.7119	0.588	0.6109

DataScientest

18.2 Descriptions of Code Files

1. **Exploratory_analysis.ipynb:**

Purpose: Conducts exploratory data analysis (EDA) on a combined text and image dataset.

Source Files Used:

- X_train_update.csv: Training data for features.
- Y_train_CVw08PX.csv: Labels corresponding to training data.
- X_test_update.csv: Test dataset for features.
- translated_text.csv: File containing preprocessed or translated text data.
- Train_merged.csv: A combined dataset with features and labels.
- df_image_data_20241024.csv: Metadata or extracted features from image data.
- Additional Inputs: Image data available in the Challenge (link provided).

2. **DistilBERT_base_uncased_model.ipynb:**

Purpose: Implements a text classification model using the DistilBERT base uncased pre-trained model.

Source Files Used:

- BERT_dataset.csv: Dataset prepared for training and testing the text classifier.
- Output: Trained DistilBERT model, tokenizer, and evaluation metrics.

3. **Vgg16_image_model. ipynb:**

Purpose: Implements an image classification model using the VGG16 convolutional neural network architecture.

Source Files Used:

- image_train folder (within the image data folder): Contains training images for the model.
- Output: Trained VGG16 model and evaluation metrics.

4. **Grad_CAM.ipynb:**

Purpose: Generates visual explanations for VGG16 model predictions using Gradient-weighted Class Activation Mapping (Grad-CAM).

Source Files Used:

- Selected images from the image_train folder.
- Output: Heatmaps overlaid on input images to visualize model focus areas.

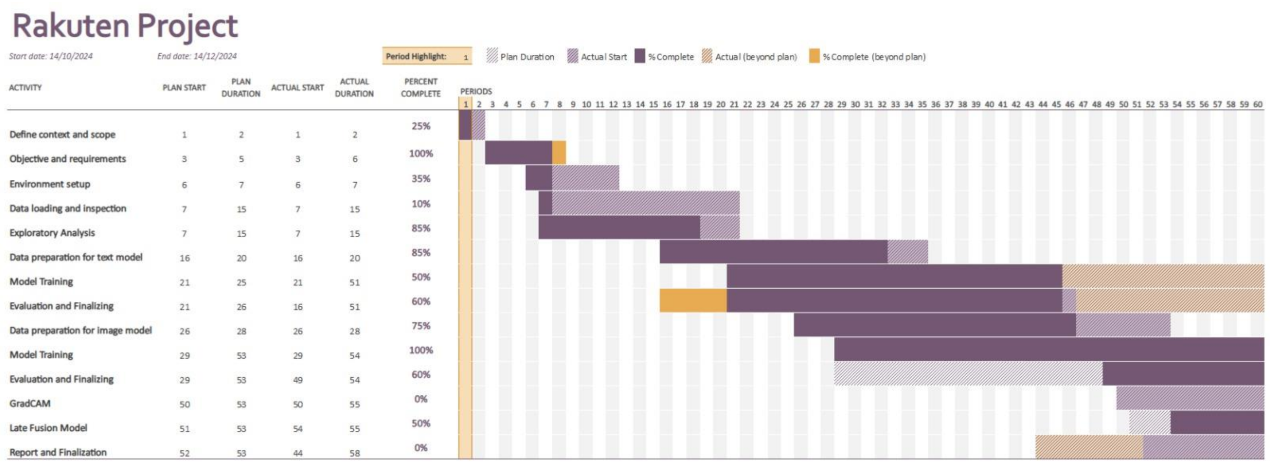
5. **late_fusion_model.ipynb:**

Purpose: Combines predictions from the DistilBERT text model and the VGG16 image model using a voting classifier (weighted and soft voting).

Source Files Used:

- Saved DistilBERT and VGG16 models.
- Tokenizer and label encoder used during text and image model training.
- Output: Final classification results, combining text and image modalities.

18.3 Project Timeline


Rakuten Project

Start date: 14/10/2024 End date: 14/12/2024

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Define context and scope	1	2	1	2	25%
Objective and requirements	3	5	3	6	100%
Environment setup	6	7	6	7	35%
Data loading and inspection	7	15	7	15	10%
Exploratory Analysis	7	15	7	15	85%
Data preparation for text model	16	20	16	20	85%
Model Training	21	25	21	51	50%
Evaluation and Finalizing	21	26	16	51	60%
Data preparation for image model	26	28	26	28	75%
Model Training	29	53	29	54	100%
Evaluation and Finalizing	29	53	49	54	60%
GradCAM	50	53	50	55	0%
Late Fusion Model	51	53	54	55	50%
Report and Finalization	52	53	44	58	0%