Automated product classification

Introduction

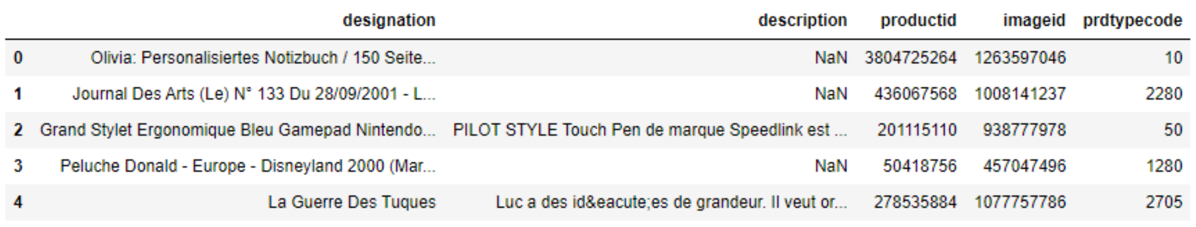
The digital buying or selling of products and services, referred to as e-commerce, is one of the many industry fields that utilizes artificial intelligence (AI). This helps for process automations, and improving marketing strategies and profitability. Information on the deliverables is collected and modified inside electronic catalogs or PIM (product information management) systems. Each deliverable or group of deliverables belongs to a specific category or class, which can be further organized into subcategories that meet given criteria. This structure allows for a narrower and faster search of items in the catalog, requiring, however, their accurate classification. Presently, in some companies the classification is still performed manually, which is challenging and time consuming when a large amount of data must be processed. Moreover, manual data handling is prone to human errors, which may take additional time and resources to identify and fix. Hence, the number of firms that turn to data science to automate this procedure is continuously increasing.

E-commerce data comes in the form of text and images for describing the deliverables. The branches of computer science, applied for analyzing and treating this form of data are various machine and deep learning algorithms, such as natural language processing (NLP), convolutional neural network (CNN), etc. Often the input data is noisy (containing typos, special characters etc.) or incomplete which requires a thorough pre-processing before feeding to the models.

In this project we are introduced to an e-commerce problem related to the multiclass classification of products, based on their description (text) and image. We approached this task by treating the text and images separately. For text processing, we used NLP tools and further tested various machine learning algorithms. The image classification was performed using convolutional neural networks (CNN) by varying the input parameters. The final class of the product was obtained through the combination of text and image modeling.

Dataset

The information provided to us is in the form of tabular data (dataframe) and images, available on the Rakuten platform. The dataframe used for applying the models is provided in a CSV format and divided into X (variables) and Y (target), containing the product features and the product classes, respectively. An extract of the combined dataframe is shown in Figure 1.



	designation	description	productid	imageid	prdtypecode
0	Olivia: Personalisiertes Notizbuch / 150 Seite...	NaN	3804725264	1263597046	10
1	Journal Des Arts (Le) N° 133 Du 28/09/2001 - L...	NaN	436067568	1008141237	2280
2	Grand Stylet Ergonomique Bleu Gamepad Nintendo...	PILOT STYLE Touch Pen de marque Speedlink est ...	201115110	938777978	50
3	Peluche Donald - Europe - Disneyland 2000 (Mar...	NaN	50418756	457047496	1280
4	La Guerre Des Tuques	Luc a des id&eacute;es de grandeur. Il veut or...	278535884	1077757786	2705

Figure 1. Extraction of the training dataframe containing the products' designation, description, product ID, image ID and the product type codes.

The columns are:

1)	**designation**	Information about the product
2)	**description**	Information about the product
3)	**productid**	Unique ID of the product
4)	**imageid**	Unique ID of the image associated with the product.
5)	**prdtypecode**	The class that the product belongs to

The dataset contains 84916 entries, 27 distinct product classes and roughly 35% missing values in the description column.

The image dataset is provided in jpg format, containing 84916 (2.10 GB) images.

Approach

To solve the problem of product classification, we treated the text and images separately, and carried out the following steps:

1. Data exploration and visualization
2. Creation of vectors and word embeddings
3. Applying modeling techniques
4. Analyzing the results

Detailed explanation and results

(A) Text

1. Data exploration and visualization

To begin with, the text data of the set was investigated and pre-processed thoroughly. It involved the checking of missing values, and the products' distribution over the various classes, as shown in Figure 2.

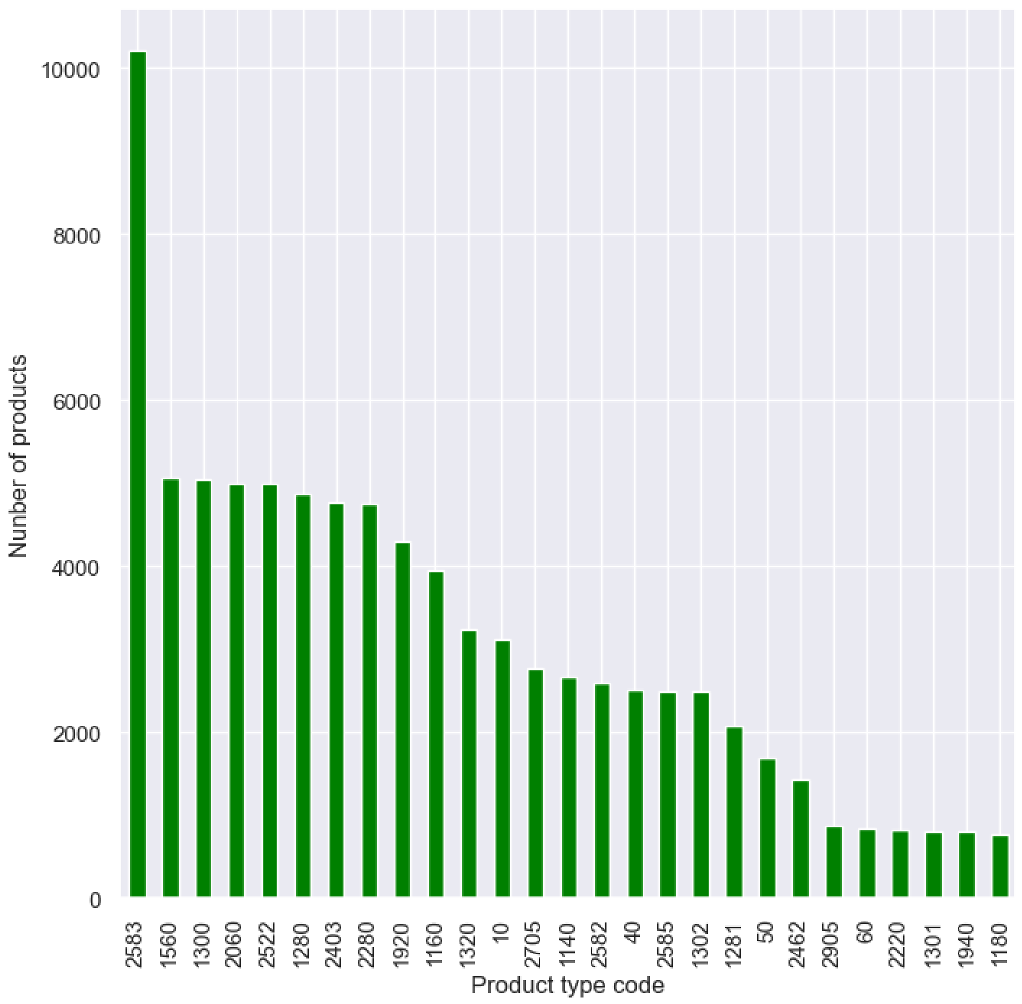


Figure 2. Distribution of the product classes

From this, we see that the data set is strongly unbalanced, which often leads to biases and poor generalization of the applied models, and should be considered when analyzing the results. Due to the large fraction of missing values in the 'description' column, we decided to merge both 'designation' and 'description', removing the overlapping information.

Further, we performed the following text mining techniques:

- converting strings to lowercase
- removing web links/html tags/e-mails/numbers through regular expressions
- removing punctuations and white spaces
- checking the languages present in the text
- removing stop words
- applying word tokenization
- stemming
- retaining only words above a given length of letters

From the cleaned data frame we were able to extract some metadata (words per entry, number of emails and websites), the languages present in the text and the most frequent words. Figure 3 and 4 show the detected languages and the most frequent words in the form of a word cloud, respectively.

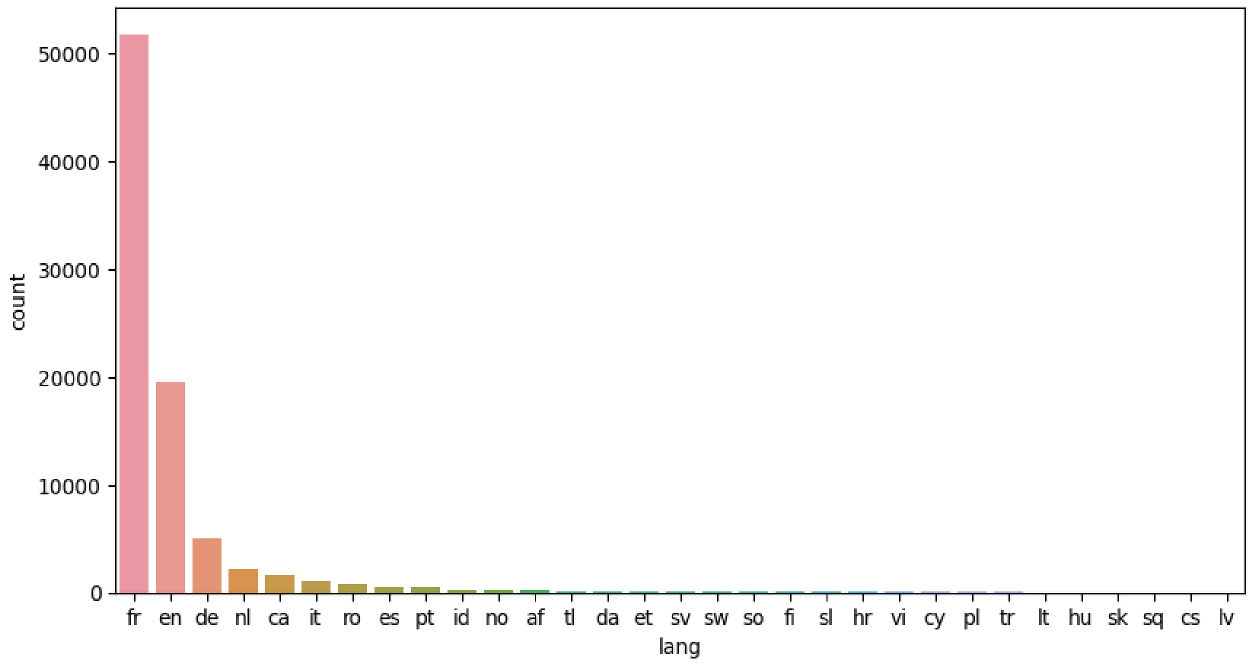


Figure 3. Languages detected in the text.

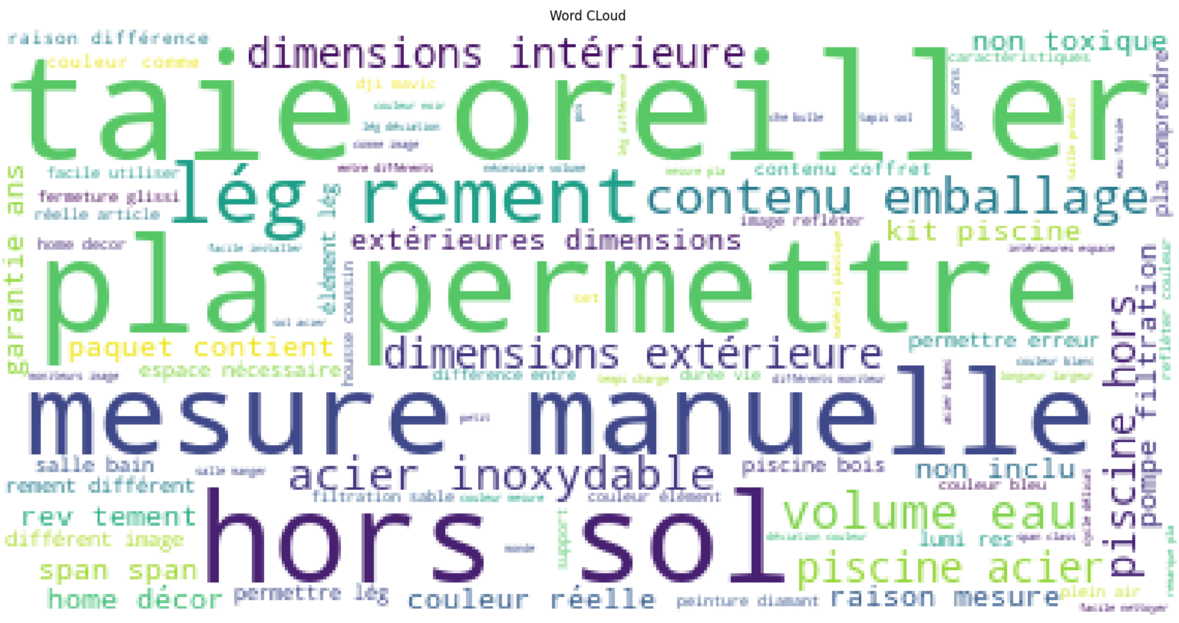


Figure 4. Word cloud of the most frequent words after cleaning the text data.

2. Creation of vectors and word embeddings

In order to provide input to the ML/deep learning algorithm, it is mandatory to translate the text into numerical data in the form of vectors. For this purpose, we used the count vectorizing and word2vec techniques. The difference between them is summarized below.

Count vectorization:

- Takes the full text data (from all columns) and counts the amount of unique words
- The rows are transformed to sequences of 0s and 1s, where each unique word corresponds to a single column and given a value 0 (if the word is absent) or 1 (if it is present).
- This method creates orthogonal vectors which do not provide a relationship between the different words.

Word2Vec:

- This technique also vectorizes the data but it provides the semantic and arithmetic properties of the words, owing to the dot product between the vectors.
- It gives the orientation/position of the words in space.
- It reduces the size of the data to be processed.

This model was applied on the column that contained the stemmed text from both the description and designation columns of our dataset, and each word was denoted by a

100-dimensional vector. A dictionary containing the vocabulary was created. To visualize the word embeddings, the 100-dimensional vector space was transformed into 2D space by using Principal Component Analysis (PCA), which reduces the number of dimensions. Some of the words are depicted in Figure 5.

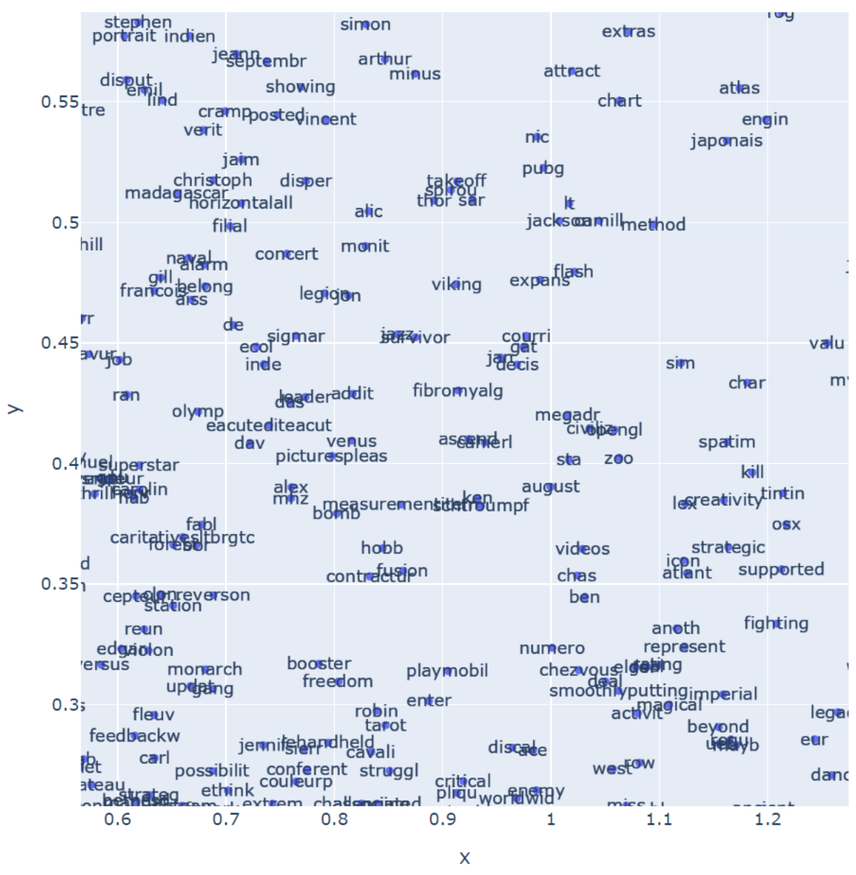


Figure 5. Representation of words in 2D space after applying PCA.

To form the final representation of each row, the corresponding vectors for that row were summed up and averaged, thus creating a sentence.

3. Modeling

After the pre-processing and creation of sentence vectors, the data was splitted into training and validation sets with the validation part chosen 20% of the data. To get the same splitting of the data set for each model execution, the random state was fixed. Various algorithms were trained on the data to predict the product classes. The reason for this was to test which algorithm gives a better score, and a good generalization. A brief description of a few of these algorithms is given below:

(a) K-Nearest Neighbours: It assigns the class to the unlabeled data depending on the majority class among its nearest neighbors. We tested a number of n-neighbors ranging from 1-20 with the help of GridSearchCV, which returns the best parameters for the algorithm.

(b) Random Forest: The classification is achieved by organizing the data in a multitude of decision trees.The output class is the one that is selected by most of the trees.

(c) Dense Neural Networks: This model is similar to the working of neurons in the human brain, where a single neuron in a layer receives an input from all the neurons present in the previous layer (dense connection).

Table 1 summarizes the parameters and scores the used models.

Model name	Vectorizers	Parameters	Train score	Overall accuracy score
K-Nearest Neighbours	Count vectorizer	n_neighbors = 1	0.99	0.60
	word2vec	n_neighbors = 7	0.80	0.75
Random Forest	word2vec	default	0.99	0.76
Support Vector Machine	word2vec	default	0.85	0.79
Dense Neural Networks	4 hidden layers, sparse_categorical_crossentropy" loss, adam optimizer, "accuracy" metrics, 20 epochs	word2vec	0.76	0.75

Table 1. The scores obtained by various data algorithms

(B) Image processing

After importing the images and checking their size, we reshaped them to 100x100 and normalized their pixel intensities, in order to speed up the computation. Preliminary results obtained with a CNN model, applied directly after, showed poor overall performance on both training and validation sets and null F1-score for many classes. One of the reasons for the low score is suggested to be the unbalanced data. In an attempt to improve the score, we applied data augmentation. Due to limited resources, we were not able to run a model on the new data. Next, we created a bounding box to enclose the represented object and minimize the noise. A CNN model was again trained on the new dataset. However, this did not improve the score significantly. Thereafter, we re-weighted the classes, which involves the assignment of a higher cost to the minority categories. We noticed an overall improvement in the score and a better agreement between training and validation accuracy. The evolution of the loss and accuracy with each training step is shown in Figure 6.

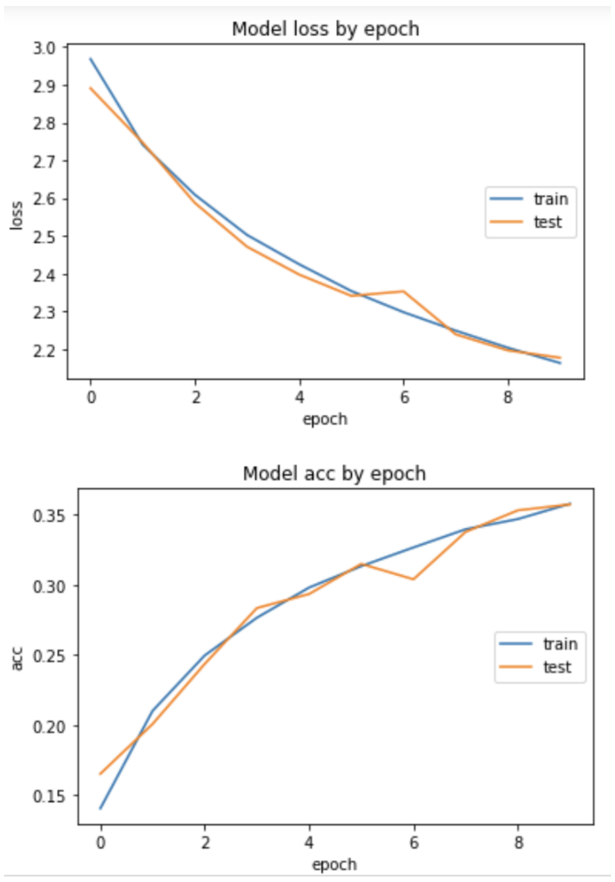
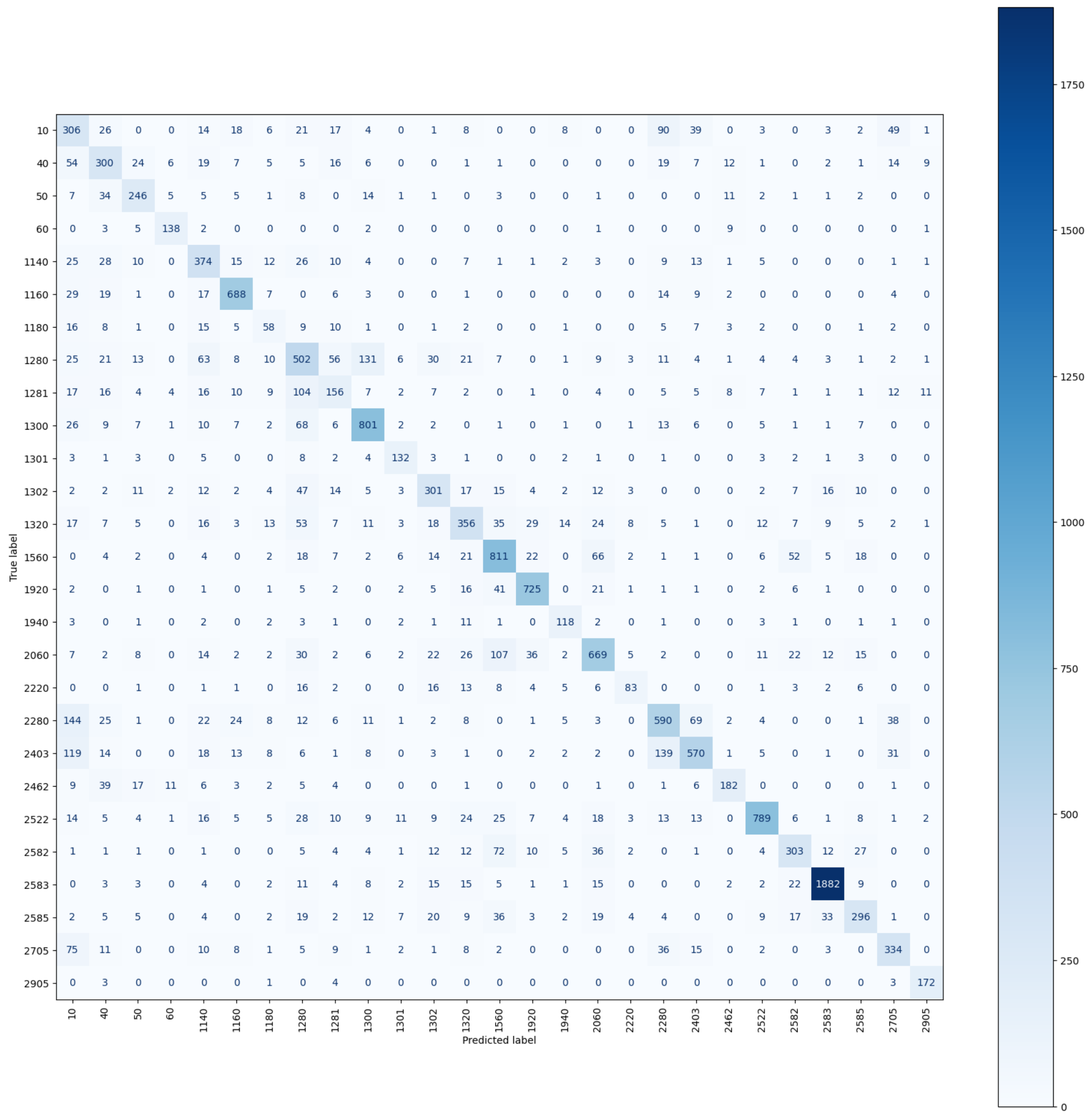


Figure 6. Model loss and accuracy of the training and validation sets according to the epoch.

(C) Combination of text and images

The probabilities of a product belonging to a particular class were obtained for both text and images using KNN and CNN, respectively. To attain the final probability per class, we assumed that text and image data are independent of each other. Therefore, we averaged the probabilities obtained for both data types. Finally, we trained a KNN model on the merged dataframe of word2vec vectors and probabilities from image modeling. Figure 7 shows the predicted and actual classes in the form of confusion matrix.



True label \ Predicted label	10	40	50	60	1140	1160	1180	1280	1281	1300	1301	1302	1320	1560	1920	1940	2060	2220	2280	2403	2462	2522	2582	2583	2585	2705	2905
10	306	26	0	0	14	18	6	21	17	4	0	1	8	0	0	8	0	0	90	39	0	3	0	3	2	49	1
40	54	300	24	6	19	7	5	5	16	6	0	0	1	1	0	0	0	0	19	7	12	1	0	2	1	14	9
50	7	34	246	5	5	5	1	8	0	14	1	1	0	3	0	0	1	0	0	0	11	2	1	1	2	0	0
60	0	3	5	138	2	0	0	0	0	2	0	0	0	0	0	0	1	0	0	0	9	0	0	0	0	0	1
1140	25	28	10	0	374	15	12	26	10	4	0	0	7	1	1	2	3	0	9	13	1	5	0	0	0	1	1
1160	29	19	1	0	17	688	7	0	6	3	0	0	1	0	0	0	0	0	14	9	2	0	0	0	0	4	0
1180	16	8	1	0	15	5	58	9	10	1	0	1	2	0	0	1	0	0	5	7	3	2	0	0	1	2	0
1280	25	21	13	0	63	8	10	502	56	131	6	30	21	7	0	1	9	3	11	4	1	4	4	3	1	2	1
1281	17	16	4	4	16	10	9	104	156	7	2	7	2	0	1	0	4	0	5	5	8	7	1	1	1	12	11
1300	26	9	7	1	10	7	2	68	6	801	2	2	0	1	0	1	0	1	13	6	0	5	1	1	7	0	0
1301	3	1	3	0	5	0	0	8	2	4	132	3	1	0	0	2	1	0	1	0	0	3	2	1	3	0	0
1302	2	2	11	2	12	2	4	47	14	5	3	301	17	15	4	2	12	3	0	0	0	2	7	16	10	0	0
1320	17	7	5	0	16	3	13	53	7	11	3	18	356	35	29	14	24	8	5	1	0	12	7	9	5	2	1
1560	0	4	2	0	4	0	2	18	7	2	6	14	21	811	22	0	66	2	1	1	0	6	52	5	18	0	0
1920	2	0	1	0	1	0	1	5	2	0	2	5	16	41	725	0	21	1	1	1	0	2	6	1	0	0	0
1940	3	0	1	0	2	0	2	3	1	0	2	1	11	1	0	118	2	0	1	0	0	3	1	0	1	1	0
2060	7	2	8	0	14	2	2	30	2	6	2	22	26	107	36	2	669	5	2	0	0	11	22	12	15	0	0
2220	0	0	1	0	1	1	0	16	2	0	0	16	13	8	4	5	6	83	0	0	0	1	3	2	6	0	0
2280	144	25	1	0	22	24	8	12	6	11	1	2	8	0	1	5	3	0	590	69	2	4	0	0	1	38	0
2403	119	14	0	0	18	13	8	6	1	8	0	3	1	0	2	2	2	0	139	570	1	5	0	1	0	31	0
2462	9	39	17	11	6	3	2	5	4	0	0	0	1	0	0	0	1	0	1	6	182	0	0	0	0	1	0
2522	14	5	4	1	16	5	5	28	10	9	11	9	24	25	7	4	18	3	13	13	0	789	6	1	8	1	2
2582	1	1	1	0	1	0	0	5	4	4	1	12	12	72	10	5	36	2	0	1	0	4	303	12	27	0	0
2583	0	3	3	0	4	0	2	11	4	8	2	15	15	5	1	1	15	0	0	0	2	2	22	1882	9	0	0
2585	2	5	5	0	4	0	2	19	2	12	7	20	9	36	3	2	19	4	4	0	0	9	17	33	296	1	0
2705	75	11	0	0	10	8	1	5	9	1	2	1	8	2	0	0	0	0	36	15	0	2	0	3	0	334	0
2905	0	3	0	0	0	0	1	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	172

Conclusion

In this work, we pre-processed the text and converted it into numerical data using two vectorizing techniques, namely the count vectorizer and word2vec. Further, we trained various machine learning models to check their performance. The image data was trained using a CNN model. The overall probability for belonging to a particular class was calculated by combining the results from text and image data. To improve the obtained score, we suggest training a model using the original size of the images and applying augmentation for balancing the data.