Classification of Research Papers

Beatrice Tomassi, Lorenzo Rocchi

February 10, 2025

Contents

1 Introduction

The classification of research papers is a crucial task in the academic and scientific communities, allowing efficient organization, retrieval, and recommendation of knowledge. Usually, finding research papers that exactly match a specific topic or subject is an extremely time-consuming activity that an effective classification system can potentially improve by suggesting the most relevant works to the researchers.

In this project, we worked on the problem of multilabel classification of research papers, where each document may belong to multiple categories simultaneously.

In particular, the used dataset[3] consists of papers categorized into six domains:

- *Computer Science*,
- *Physics*,
- *Mathematics*,
- *Statistics*,
- *Quantitative Biology*,
- *Quantitative Finance*.

To identify the most effective model for accurately classifying research papers into their respective domains, four different classifiers have been explored and evaluated.

In the following sections, each phase of the work done will be explored.

2 Dataset visualization

The dataset consists of research papers classified into multiple scientific fields. Each category is represented as 0 or 1, where:

- **1** indicates that the paper **belongs** to that specific category;
- **0** indicates that the paper **does not belong** to that category.

	ID	TITLE	ABSTRACT	Computer Science	Physics	Mathematics	Statistics	Quantitative Biology	Quantitative Finance
0	1	Reconstructing Subject-Specific Effect Maps	Predictive models allow subject-specific inf...	1	0	0	0	0	0
1	2	Rotation Invariance Neural Network	Rotation invariance and translation invarian...	1	0	0	0	0	0
2	3	Spherical polyharmonics and Poisson kernels fo...	We introduce and develop the notion of spher...	0	0	1	0	0	0
3	4	A finite element approximation for the stochas...	The stochastic Landau--Lifshitz--Gilbert (LL...	0	0	1	0	0	0
4	5	Comparative study of Discrete Wavelet Transfor...	Fourier-transform infra-red (FTIR) spectra o...	1	0	0	1	0	0

Figure 1: The dataset.

As mentioned and as can be seen in Figure 1, each paper can belong to one or more categories.

Since before applying machine learning models is important to understand the dataset, we explored the data through different visualizations, in order to gain insight into different characteristics of the dataset.

Below, we present the visualizations performed and explain their relevance.

2.1 Distribution of categories

We choose to initially visualize if one or more categories have significantly more samples than others, meaning if the dataset is balanced or imbalanced in terms of category distribution.

The result in Figure 2 shows that the dataset is **highly imbalanced**, in particular, most articles are classified as *Computer Science* (8594), while very few are classified as *Quantitative Biology* (587) or *Quantitative Finance* (249).

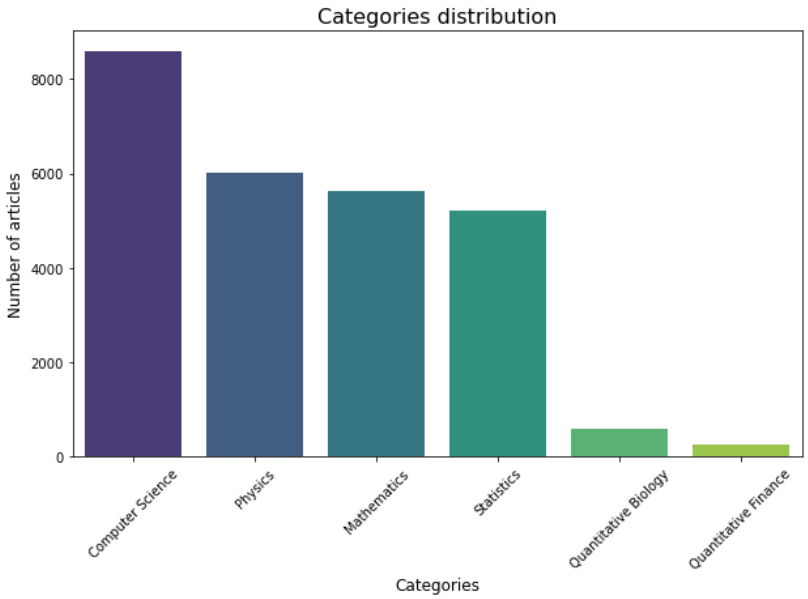

Figure 2: Distribution of categories.

2.2 Text length

Understanding the length distribution of texts, where for *text* we mean title and abstract, will help us to understand whether the preprocess phase will be performed correctly or not.

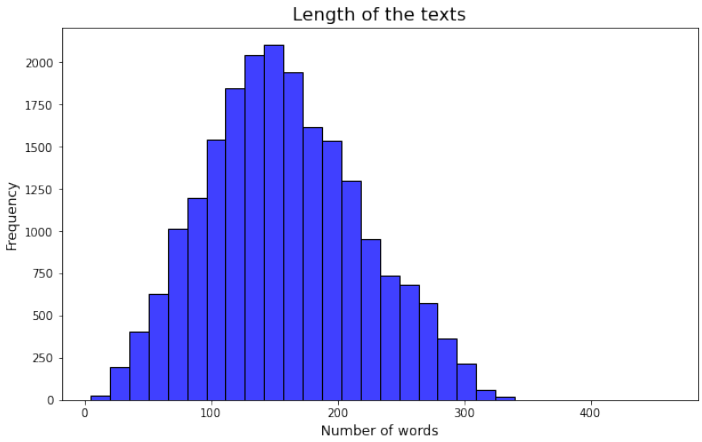

Figure 3: Length of texts.

As the dataset is right now, the average length of the texts is 157.92 words.

2.3 Most common words

By visualizing the most frequently occurring words, we gain insight into the main themes within the dataset. This also helps detect potential stopwords (e.g., *the* or *and*) that may need removal, ensuring that the most informative terms are mantained.

As expected and shown in Figure 4, at the moment the most common words do not contribute to the main purpose of the articles.

Figure 4: Most common words.

3 Dataset preprocessing

To prepare the text for classification, the following steps were performed:

1. **Lowercasing**: all words were converted to lowercase to ensure uniformity and avoid treating words with different cases as distinct (e.g., *Physics* and *physics*);
2. **Removing special characters**: non-alphanumeric characters (such as punctuation) were removed;
3. **Contraction expansion**: contracted words were expanded (e.g., *doesn't* becomes *does not*);
4. **Removing stopwords**: commonly used words (stopwords) that do not carry significant meaning (e.g., *the*, *and*, *of*) were removed to reduce noise;
5. **Lemmatization**: words were reduced to their base or dictionary form (e.g., *reconstructing* becomes *reconstruct*, *better* becomes *good*) to reduce word variation and improve model generalization.

To verify that the preprocessing was successful, we visualized the new dataset to inspect whether stopwords and special characters were correctly removed, contractions were expanded, and lemmatization was properly applied.

3.1 Preprocessed dataset visualization

In this step, two key aspects were analyzed again: the text length distribution and the most common words. In particular, we examined whether the average length of the texts had decreased after preprocessing, confirming the effective removal of stopwords and unnecessary characters. Then, a word cloud visualization was generated to ensure that the obtained words were relevant to the classification task.

By analyzing these visualizations (Figure 5 and Figure 6), we confirmed that the preprocessing steps successfully cleaned the dataset while preserving meaningful content.

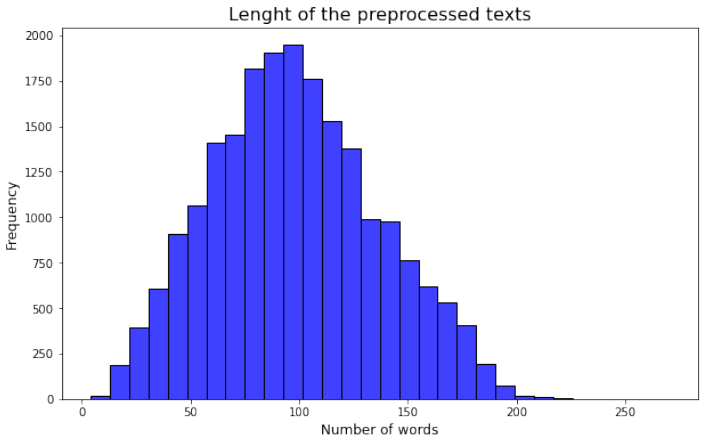

Figure 5: Length of texts after preprocessing.

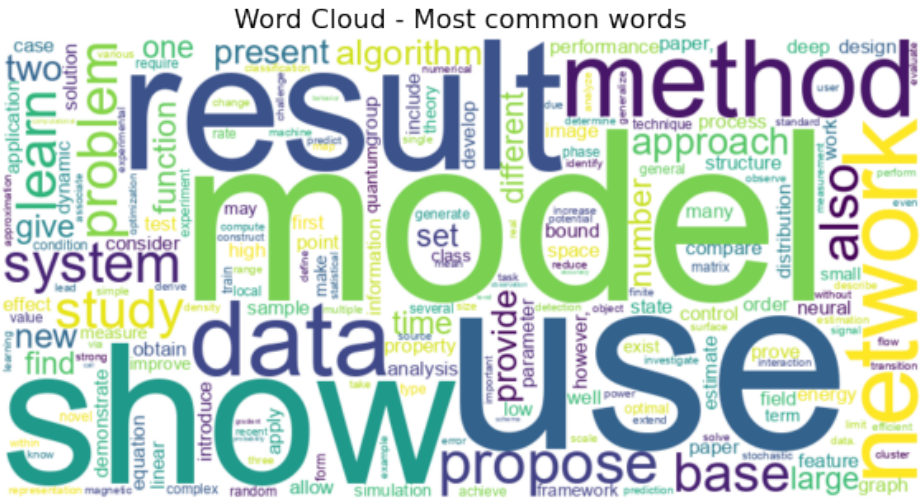

Figure 6: Most common words after preprocessing.

4 Training phase and evaluation metrics

Before the training phase itself, the data needed to be further processed in order to be correctly read by the Machine Learning models, therefore we performed two key steps:

- **Binarizing the labels:** We used the *MultiLabelBinarizer* from `sklearn.preprocessing` to transform the multilabel categories into a binary matrix format. Each category was converted to a separate binary feature, making the labels suitable for multilabel classification.
- **Vectorizing the text**: We applied the *TfidfVectorizer* from `sklearn.featureextraction.text` to convert the text data into a matrix of TF-IDF features. By setting `maxfeatures` to 10,000, we limited the number of features to the top 10,000 most important terms.

These steps ensure that both the labels and text data are in a format suitable for training the models.

4.1 Training the models

Once the dataset was ready, we moved on to the training phase. In particular, we chose to assess the impact of preprocessing on classification performance and train multiple models to evaluate different scenarios.

A common practice among all training sessions was the use of K-Fold Cross Validation to evaluate the different models. K-Fold Cross Validation partitions the dataset into multiple subsets (K subset) where some are used for the training and the others for the validation. This helps mitigate the risk of overfitting, ensuring that each data point is used for both training and validation and providing a more reliable estimation of model performance compared to a single train-test split.

We initially trained a **Logistic Regression classifier** wrapped in a **OneVsRest (OvR) strategy**. Logistic Regression is a widely used algorithm for classification tasks due to its *simplicity*, *efficiency*, and *interpretability*. However, since our problem involves multilabel classification, where a single instance can belong to multiple categories simultaneously, a traditional multiclass approach is not directly applicable. The OneVsRest strategy (OvR) was adopted to handle this issue, as it allows us to train a separate binary classifier for each label. Also, this approach is particularly useful when dealing with imbalanced datasets, as each classifier focuses on distinguishing a specific category from the rest.

The OneVsRest (OvR) strategy was preferred over other alternatives like OneVsOne (OvO) or Binary Relevance (BR) for three main reasons:

1. is computationally more efficient than the others,
2. is better suited for multilabel problems, since it independently predicts whether each class applies to a given instance,
3. each classifier in OvR can be analyzed separately, making it easier to debug and understand individual class predictions.

After establishing a baseline with Logistic Regression, we expanded our experimentation by training three additional models on the preprocessed dataset:

- **Support Vector Machine (SVM)** with a *LinearSVC classifier*, which is well-suited for high-dimensional feature spaces as TF-IDF vectorized text data.
- **LightGBM (LGBM)**, a gradient boosting model that is efficient in handling large-scale data and offers strong performance in multi-label tasks.
- **Multi-Layer Perceptron (MLP)**, a simple neural network architecture, to evaluate how a more complex, non-linear model compares to traditional classifiers.

Each model was evaluated using **F1-score**, **accuracy**, and **recall**. In particular, **F1-score** provides a balanced measure between precision and recall and it is a very important metric to study when the dataset is imbalanced, accuracy gives an overall measure of correctness, and recall is particularly important to ensure that relevant labels are not missed.

4.1.1 The impact of preprocessing

To evaluate the effect of preprocessing, we trained the Logistic Regression model on the raw dataset (using the original text) and then on the preprocessed dataset (where text cleaning, stopword removal, and lemmatization were applied). This allowed us to analyze how text processing influences classification performance.

```
Average F1 Score:  0.8046835068187619
Average Accuracy Score:  0.6535380528287474
Average Recall Score:  0.7531909238029411
```

Figure 7: Scores on the raw dataset.

```
Average F1 Score:  0.8048530242924674
Average Accuracy Score:  0.6531089137498771
Average Recall Score:  0.7537980623531562
```

Figure 8: Scores on the processed dataset.

As can be seen from Figure 11 and Figure 12, surprisingly the preprocessing does not significantly affect the performance of the model. Here are few possible explanations:

- The raw dataset is already well structured and has sufficient information to allow the classifier to make accurate predictions without needing extensive cleaning;

- Since Logistic Regression assigns weights to individual features, the preprocessing steps may not have significantly altered the most influential words, resulting in minimal changes to the model's performance.

We can conclude that even though text preprocessing is generally beneficial, in this case its minimal effect suggests that the dataset and model are already performing optimally.

4.1.2 Evaluating different models

As mentioned, we then expanded our experimentation by training three additional models on the preprocessed dataset.

```
Average F1 Score:  0.8048530242924674
Average Accuracy Score:  0.6531089137498771
Average Recall Score:  0.7537980623531562
```

Figure 9: Scores of Logistic Regression.

```
Average F1 Score:  0.8049358557741408
Average Accuracy Score:  0.6454326090453302
Average Recall Score:  0.7813181782303971
```

Figure 10: Scores of Support Vector Machine.

```
Average F1 Score:  0.8044515645630783
Average Accuracy Score:  0.6415221699880014
Average Recall Score:  0.7764345866804828
```

Figure 11: Scores of LightGBM.

```
Average F1 Score:  0.7747247507707448
Average Accuracy Score:  0.593171685755745
Average Recall Score:  0.7632861456345716
```

Figure 12: Scores for Multi-Layer Perceptron.

As can be seen, the scores does not significantly vary between different classifiers, suggesting that the models might have similar expressive power for this task: Logistic Regression, Support Vector Machines, LightGBM, and Multi-Layer Perceptron are different in complexity, but if the relationships between features and labels are relatively simple, their expressivness might be comparable. This means that in some cases even a linear model like Logistic Regression may perform just as well as more advanced models.

To observe greater differences, one could explore different data representations or more extensive hyperparameter tuning.

5 Comparison with other works

Since the comparison with the previously proposed classifiers did not fully meet our expectations, to gain a deeper understanding of the dataset's potential we compared our results with existing works available on Kaggle that have used the same dataset. We wanted to answer the following questions:

- What if we had balanced the dataset?
- What if we had chosen to use Transformers/PyTorch instead of Sklearn?

5.1 Multi-Label Text Classification balancing the dataset

The reference work found on Kaggle[1] balances the dataset, which is usually crucial in mutlilabel classification, since it helps ensure that all labels are adequately represented in the training process. This should prevent the model from being biased toward the majority classes.

Specifically, the referenced work applies oversampling (SMOTE) to balance the dataset, augmenting the number of examples of the underrepresented classes to match the frequency of overrepresented ones.

Then, it trains a MultiOutputClassifier on the balanced dataset. Table 1 outlines the differences between the results of the referenced work and ours.

From this comparison, we observe:

1. **Higher accuracy in the reference work**, suggesting that balancing the dataset helps the classifiers make more precise predictions. This could indicate that our model struggles with imbalanced classes, leading to misclassification of minority labels.

Table 1: Performance Comparison

Model	F1-score	Accuracy	Recall
Our Logistic Regression (OneVsRestClassifier)	0.804	0.653	0.753
Referenced Work (MultiOutputClassifier with MLSMOTE)	0.814	0.918	0.643

2. **Higher recall in our model**, meaning that our model is more sensitive to detecting positive lables.

3. **Similar F1-scores**, indicating a similar balance between precision and recall.

We can conclude that dataset balancing significantly improves accuracy, as seen in the referenced work, however it comes at the cost of recall, meaning the model might miss some positive cases. The similar F1-scores suggest that the overall classification capability is not drastically different, despite the different techniques used.

5.2 Multi-Label Text Classification using Transformers

The reference work found on Kaggle[2] employs Transformer-based models for multi-label text classification. Transformers captures contextual relationships in text, making them highly effective for natural language processing (NLP) tasks. In contrast to our approach, which relies on traditional Machine Learning, the referenced work uses a Transformer model to extract deep semantic features from the text before classification.

Table 2: Performance Comparison

Model	F1-score	Accuracy	Recall
Our Logistic Regression (OneVsRestClassifier)	0.804	0.653	0.753
Referenced Work (Transformer-based Model)	0.792	0.643	0.784

From the comparison in Table 2, we observe:

1. **Higher F1-score and accuracy in our model**, suggesting that the OneVsRest Logistic Regression approach performs slightly better in terms of balancing precision and recall, as well as overall correct classifications.

2. **Higher recall in the Transformer model**, suggesting it is better at capturing and identifying positive labels. This indicates that deep contextual understanding helps in retrieving more relevant labels, although it may come at the expense of precision.

We can conclude that while Transformer-based models have the advantage of capturing complex textual relationships, they do not always guarantee superior performance.

6 Conclusion

In this project, we explored the multi-label classification of research papers, focusing on six academic domains. By analyzing the dataset, we identified key characteristics (such as class imbalance), which guided our **preprocessing** decisions. The preprocessing steps, including lowercasing, stopword removal, and lemmatization, effectively **improved the quality of textual data**, as demonstrated by our visual analysis.

To classify the research papers, we trained **multiple models** using the K-Fold cross validation, including Logistic Regression (with OneVsRest strategy), Support Vector Machine, LightGBM and Multi-Layer Perceptron. The results of our evaluations did not confirm that preprocessing always enhances performance, encouraging us to compare our work to more complex ones.

However, this further analysis shows that not always complex models performs better than simpler ones.

We would like to conclude citing the **Occam's Razor Principle**: *"All things being equal, the simplest solution tends to be the best one"*.

References

[1] FELLAH Abdelnour. *Multi-Label Text Classification*. https://www.kaggle.com/code/fellahabdelnour13/multi-label-text-classification. Accessed: 10/02/2025.

[2] isseyice. *Transformer-Multi-Label-Classification*. https://www.kaggle.com/code/isseyice/transformer-multi-label-classification. Accessed: 10/02/2025.

[3] Shivanand. *Multi-Label Classification Dataset for Research Articles*. https://www.kaggle.com/datasets/shivanandmn/multilabel-classification-dataset. Accessed: 10/02/2025.