I have downloaded a multilabel database at Kaggle (<https://www.kaggle.com/datasets/shivanandmn/multilabel-classification-dataset/data>). It is a textual database in English about scientific articles and their topics. It consists of nine columns with the ID, title, and abstract, and six columns about the topics: Computer Science, Physics, Mathematics, Statistics, Quantitative Biology, and Quantitative Finance. The training set in “train.csv” is annotated, while the test one has only ithe d, title, and abstract.

I chose a textual database because I have little experience in NLP and want to challenge myself with a new class of problems.

The database has no missing values, and it is balanced, so there is no necessity for data cleaning, data imputation, handling imbalanced classes, and so on.

The data preprocessing starts with the concatenation of the “TITLE” and the “ABSTRACT” columns to obtain the working feature “full\_text” of both datasets. Then, I have lemmatized this new column via spaCy. The target variable is composed of the columns of the labels. After that, I divided the original training dataframe in training and validation sets via the classic “train\_test\_split” function of scikit-learn. The data preprocessing is finished via vectorization with TfidfVectorizer, a class from scikit-learn.

I followed the scientific literature on the topic; therefore, I have chosen binary relevance to tackle the multilabel problem. This approach is implemented in OneVsRestClassifier class of scikit-learn, where an independent classifier is trained for each label. I have explored 3 types of classifiers: logistic regression (the simplest one), random forest, and supporting vector machine. I adopted the f1 score as the evaluation metric to capture both the precision and recall of the classification.

I have performed the training on the training set with all the 3 classifiers, and I have evaluated the performance on the validation set, where the supporting vector machine classifier proves to be the best.

Now, the models are ready to perform the inference on the data in “test.csv”, and send the results to a Kaggle submission.

The f1 score on the validation set is near 80%, quite good for these simple classifiers. What can be done to improve?

1. Try different approaches to the data preprocessing: lemmatization vs stemming, using a customized set of stop words, and so on.
2. Tuning the parameters of each classifier via a grid search or an automatic hyperparameter optimization (like an Asynchronous Successive Halving)
3. Exploring other types of classifiers: for instance, Deep Learning ones.