ML Challenge: Authorship Prediction | Group 13

Learning objective
In this assignment a dataset with metadata of scientific papers was provided, with a task on authorship prediction using machine learning models. Via this assignment understanding of theory and implementing different algorithms, using Python libraries such as Numpy and Scikit-learn, are being showcased.

Feature engineering
Before preprocessing the authorship metadata an exploratory data analysis was conducted. All paperId's were unique, there were no duplicates and the amount of unique authorId's was around half of the number of entries in the dataset. The test dataset consists of the same features but without author name and authorId. Since the paperId in de train dataset did not have duplicates, the four remaining features (abstract, title, venue and year) were used for training. A function was made for the preprocessing stage. The function removes special characters, punctuation, leading and trailing whitespaces and has tokenizer and lemmatizer. This function transforms all the columns into one list, with all string inputs. After that the input was vectorized using TF-IDF vectorizer (term frequency-inverse document frequency). TF-IDF reflects how important a word is to a document in a collection of words (Dua et al. 2017).

Learning algorithms
After preprocessing data, the data was split in train and validation sets. Based of the highest accuracy on the validation set, the LinearSVC algorithm was chosen for the prediction task. Other models considered included BERT (Bidirectional Encoder Representations and Transformers), Logistic Regression and Gradient Booster classifier, unfortunately running these models resulted in memory errors. Therefore, there had been decided to proceed with KNN, SVC, SGD and LinearSVC. The SGD and LinearSVC had the highest scores on validation with 9.7% and 10,9% respectively. (Appendix – Table 2)

Hyperparameter tuning
After training, Gridsearch was used to find optimal hyperparameters to see if the predictions could be made more accurately. A dictionary was created with two parameters for loss, two for penalty and four for regularization. The final hyperparameters were set at: loss= squared_hinge, penalty=l2 and C=10. This resulted in an increase in accuracy of 0.1% on validation data and 0.11% on test data.

Discussion performance model
Using the finished model on the test data showed an accuracy score of 20.4%, which is better than the baseline by 5%. Since this dataset contains imbalanced data classes, using just accuracy would not be very reliable. Precision, recall and F1 score can provide better insights to evaluate the results (Boyle, 2019).

Table 1 Classification report

	Precision	Recall	F1-score	Support
Accuracy			0.11	2426
Macro avg	0.06	0.07	0.06	2426
Weighted avg	0.10	0.11	0.10	2426

In the classification report of the model, there is a difference in performance between the macro and weighted average. The weighted average gives a more representative view of the model's performance because it considers how heavily the different classes should weight (Leung, 2022).

Different models have been used with different hyperparameters but since some of them scored around 4% or even 1%, it was hard to tell after tuning if this was some randomization or something structural in the way the data was preprocessed or the algorithm itself.
Another thing that is noticeable, is that the test accuracy is around twice as high as the validation set. The reason for this is that in the validation set it is possible to have new authorId's that have never been seen before, while in the test data all authorId's have been trained on the model at least once.

Codalab submission name: Yasmin_Lin

Detailed specification of the work done by group members

Yasmin Lin 2070056	Research on NLP feature engineering, preprocessing data, build multiple models (KNN, SVC, LinearSVC, DecisionTree, BERT, Logistic Regression, SGD Classifier), making the predictions, making the report.
Pieter Hendriks 2106712	Research NLP, data cleaning, n-grams, feature engineering. Build KNN, Decision Tree, model. GridSearch hyperparameter tuning for KNN and SVC.
Agata Rapiej 2086695	Research NLP, stylometry, text data cleaning, n-grams, building models (Multinominal Naive Bayes, KNN, Linear Regression, SVM, LinearSVC, GradientBooster classifier), tfdif, hyperparameter tuning for the final model.
Femke van Verseveld 2070487	Research on NLP feature engineering, preprocessing data, and data cleaning. Building KNN. GridSearch hyperparameter tuning for KNN. Report documentation.

Appendix

Table 2 Validation accuracy results

Algorithm	Features/ Preprocessing	Score on validation
KNN	All features	5.07%
Decision Tree	All features	2.27%
SVC	All features	0.74%
Linear SVC	All features	10.92%
SGD	All features	9.73%
Random Forrest	Abstract with POS tagging	36.76%
SVC	Without TFIDF	0.29%

References

Boyle, T. (2019). Dealing with imbalanced data. https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18

Dua, R., Gotra, S. & Pentreath, N. (2017). Machine learning with Spark – second edition. https://www.oreilly.com/library/view/machine-learning-with/9781785889936/c9033d19-dac5-4dfa-ac90-2151288db5e6.xhtml

Leung, K. (2022). Micro, Macro, & Weighted Averages of F1 Score, Clearly explained. https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f