Machine Learning Challenge, Group 23

Daniel Fidrmuc (snr: 2050307), Javier Torralba (snr: 2042878), Anja Han (snr: 2052202), Erion Mediu (snr: 2066763)

We were tasked to predict the year of publication of papers based on their metadata. We started with **exploratory data analysis**. This helped us understand types, structures, missing values, and general visualisation of the data. Next, we evaluated the baseline model to build on it further. From here, we decided the best approach was to build on the baseline algorithm, by first conducting appropriate data preprocessing, followed by feature engineering, algorithm selection and hyperparameter tuning. For this approach, we took inspiration from Molnar (2023), which was shared from our professor in class.

Starting with **data preprocessing**, we started by correcting the structure of author and editor features. Previously composed as lists of names, we transformed them into strings where the author's names were separated by semicolons. This alteration made it possible to apply vectorizing methods on each individual author name. Next, we filled all missing values of the dataset with "unknown". This provided a reference to vectorizing methods for features including strings. Building on this, we hypothesised that the number of authors, title length and abstract length might contribute to our prediction model, therefore, we created these additional features. We also created a binary feature indicating the presence or absence of an editor, however, it did not improve the MAE, so we removed it in the final model.

Continuing to **feature engineering**, we started by adding features one by one into the column transformer function, which guides the algorithm to assess and transform specific features through designated transformer objects (Müller and Guido, 2016). If a feature helped reduce the validation MAE by more than 0.1, it likely was a sign of a relevant feature to add to the model. We started using CountVectorizer, which splits a text and counts the frequencies of words (Scikit-learn.org, n.d.) as our vectorization method for: title, author, publisher, and abstract. Evaluating these features one by one, all of them significantly helped reduce the validation MAE by at least 0.2. Furthermore, in accordance with Probierz, Kozak, and Hrabia (2022), we employed the TfidVectorizer in our model. This vectorizer measures the importance of specific words in a text, assigning different weights, which helps highlight influential terms. This vectorizer showed significant improvement on the features: title, author, and abstract, but not on publisher. The features entrytype, abstract length, number of authors, and editor offered some reduction of the validation MAE, although small. Initially, these features tested individually helped by improving the MAE by about 0.01-0.08. All of them together helped reduce the CV MAE of our final model by about 0.1; therefore, we decided to include them in our final algorithm. Lastly, we tested title length as a new feature and improved the final MAE by 0.06, thus, we also included this feature on the final algorithm. Throughout the feature engineering and data preprocessing, we used OpenAI (2023) to help us find new things to test and suggest improvements.

Despite our successful feature engineering, we encountered several **failed feature engineering ideas**. These include genre clustering by k-means and PCA analysis, neither significantly improving the model. We also hypothesised that since abstract length seemed to improve the validation MAE almost as much as using a vectorizing method on abstract, we could impute the missing values on the abstract length, since about half of the values were missing. These ideas worsened the validation MAE. Also, we attempted to adjust different arguments in the vectorizers, such as adjusting the ngrams or building our own tokenizer, but these did not improve the validation MAE.

For **model selection**, we evaluated different algorithms on the set of features we engineered above. We tried several different algorithms, namely: Ridge regression, Lasso regression, XGBoost, Logistic regression, Random Forests, Support Vector Regression, and Gradient Boosting regression. From these, we chose the four algorithms to hyperparameter tune, based on validation MAE. We hyperparameter-tuned XGBoost, Logistic regression, Random Forests, and Ridge regression. We used a grid-search with cross-validation to do this. We looked up the documentation for each of these algorithms and selected the hyperparameters we believed to be the most relevant. XGBoost, Ridge regression, and Logistic regression were relatively quick to hyperparameter tune, whereas Random Forests took very long to produce an output, so hyperparameter tuning took much longer due to the high dimensionality of text data. After all hyperparameter tuning, the best-performing model was logistic regression with an inverse regularisation of 1, a solver of lbfgs, and max iterations of 1000. We made sure to use the full training data (i.e, no validation set) to predict on the testing set. This produced an official MAE of 2.87.

The name of the account under which you submitted your results to the competition on Codalab: Javier_Torralba

Detailed specification of the work done by group members:

- **Dan:** Explored stochastic regression for abstract length and other missing values, helped with the hyperparameter tuning of logistic regression, wrote the report, and edited/finalised the report.
- **Javi:** Improved the performance of the baseline regression model with feature engineering. Explored different vector methods, namely countVectorizer and TFIDVectorizer, along with clustering algorithms and PCA possibilities. Explored seven different algorithms: Ridge, Lasso, Logistic, XGBoost, Gradient Boosting, Support vector regression, and Random forests. Designed the hyperparameter tuning for the best-performing models. Helped write and edit the report.
- **Anja:** Explored natural language processing models such as BERT, Word2Vec and TFIDVectorizer, helped with hyperparameter tuning, worked on random forests and MLPregressor model, and helped write the report.
- **Erion:** I performed multiple steps in the EDA process. I evaluated several models, random forest, XGBoost with gradient descent, logistic regression with SGD, and support vector machines. I also attempted to engineer a new feature named "topic similarities," by transforming tittle with Word2Vec, and then applying KNN to devide topics into groups. However, the MAE did not improve significantly enough to be included in the final model.

References

Molnar, C. 2023. "A grandmaster's Guide to Machine Learning Challenges." URL `https://mindfulmodeler.substack.com/p/a-grandmasters-guide-to-machine-learning`. [Online; accessed 4 December 2023].

Müller, A. and S. Guido. 2016. *Introduction to Machine Learning with Python : A Guide for Data Scientists*. O'Reilly Media, Incorporated. URL `http://ebookcentral.proquest.com/lib/uvtilburg-ebooks/detail.action?docID=4698164`.

OpenAI. 2023. "Assistance Provided by ChatGPT." `https://www.openai.com`. Accessed on 7 December 2023.

Probierz, B., J. Kozak, and A. Hrabia. 2022. "Clustering of scientific articles using natural language processing." *Procedia Computer Science* URL `https://doi.org/10.1016/j.procs.2022.09.403`.

Scikit-learn.org. n.d. "Sklearn feature extraction text CountVectorizer." URL `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`. [Online; accessed 4 December 2023].