Machine Learning Assignment Report Group 4

Shan Cao: 2088399 Ivaylo Papazov:2051866 Wei Wei:2069283 Monica Venkataraman:2113957

Description

The machine learning challenge assignment aims to predict the year of publication. The evaluation metric for this assignment is MSE (Mean Squared Error). Our goal is to achieve a score below the baseline of 5.8. Initially, we explore and study the dataset, noting, for example, that 98% of the editors' information is missing, and the data type of 2399 authors is in string format. These steps assist us in feature engineering. Additionally, we explore potential regression models and attempt hyperparameter tuning. At the end of the report, we will discuss the performance of our solution.

Feature engineering

To begin with, we read the '.json' files corresponding to the train and test sets. After that, we did a couple of data processing techniques on the data such as the 'pd.to_numeric' function to convert the "year" column from categorical to numerical data (by applying errors='coerce',non-numeric values or errors are replaced with NaN and the downcast option is set to 'float' for memory efficiency) as well as transforming the 'author' and 'editor' columns, which initially contain lists of values, by concatenating the list elements into comma-separated strings. Furthermore, duplicate rows in the training data are dropped using the drop_duplicates method, which is done to ensure that each unique combination of features is considered only once during model training. Once that is done, we performed text feature engineering to merge all the text features of the dataset into one (namely "all_features"). we performed processing of the text data by removing the stopwords (using set(stopwords.words('english')) and applying TF-IDF transformation to it. Finally, we split the training data into training and validation sets using the 'train_test_split' function. The split is stratified based on the 'year' column, ensuring that the distribution of 'year' values is similar in both the training and validation sets.

Learning algorithm(s)

The target variable of the dataset we used is "year" which is definitely numbers, which means regression models need to be used in this prediction. We chose five regression models for this prediction. Out of all the five learning algorithms - Ridge Regression, Random Forest Regression, Linear Regression, LightGBM and XGBoost Regression, we found Random Forest Regression and LightGBM yield better results than others. Besides the fact that it produced the lowest MAE, this model also has the advantage of dealing with non-linear data, as well as providing feature importance, giving us the opportunity to see which were the most important features of the model. Random Forest belongs to ensemble learning methods, leveraging the construction of multiple decision trees (a fundamental method for classification and regression) for predictions. It performs well in high dimensional dataset, performing robustness capabilities of prediction. LightGBM is a machine learning model based on the gradient boosting framework, specifically designed for efficient high dimensional dataset as well. It utilizes a histogram-based algorithm that enables fast computation. Additionally, it supports parallel learning and can handle the missing values.

Hyperparameter tuning

Regarding the model selection, Random Forest Regression and LightGBM outperformed the other three models. According to those two algorithms, Grid search was applied to do the hyperparameter tuning. Using the Grid search function, we also discovered which were the best hyperparameters to use in our situation. Function Gridsearchcv() was used to do the Grid Search. After tuning, we found out that when n_estimators=100, max_depth=None, max_features='auto', the Random Forest Regression performed best. When learning_rate=0.05, n_estimators=100, LightGBM got the better

result. However, Random Forest Regression always performed better than LightGBM. Finally, the most successful model was used to make predictions on the given test dataset.

Discussion of the performance of your solution

We started by exploring the training data and then experimented with various models, but the results did not surpass the Ridge model. We considered modifying the features to be a priority. We adjusted feature selection and data cleaning, and indeed, the numerical results improved from 5.38 to 3.58 and then to 3.44. However, when we applied the code to the test set, the numerical value remained at 3.48. We suspected an overfitting issue. Attempts to address the overfitting with SMOTE were unsuccessful. Meanwhile, an error occurred during the upload due to using "drop_duplicates: test = test.drop_duplicates(keep='first')" in the test, resulting in a mismatch in the number of samples between the train and test data, causing us to miss two opportunities. Subsequently, we tried the 5 regression models, removed stop words, performed test preprocessing, and applied tokenization. The result dropped to around 3.2. In the final phase, we tuned the hyperparameters of TF-IDF transformation and the parameters of the models, resulting in a decrease in MAE from 3.2 to 3.13.

CodaLab account name

Group4

Detailed specification of the work done by group members

Shan Cao (Team Leader): Feature engineering: EDA, Preprocessing of the dataset, Merge all features together, Feature Transfer. Learning algorithm: Tried Linear Regression, Random Forest and SVM model. Hyperparameter Tuning: Use Grid Search to Tune the parameter of models. Created the account and made the submissions.

Ivaylo Papazov: Feature engineering: training and text data processing. Writing the report.

Wei Wei: Feature engineering: EDA, Data cleaning. Text data preprocessing. Learning algorithm: Random Forest Regression, XGBoost. Writing the report

Monica Venkataraman: Feature engineering: Selecting features and merge features, Tfidfvectorizer. Keeping track of the project progress and outline of the report

References

Das, T. (n.d.). How to Perform Feature Engineering From Text Data. (Medium) Retrieved December 2022, from Better

How to Perform Feature Engineering From Text Data | by Tapas Das | Better Programming

AndresHG. (2020). NLP GloVe, BERT, TF-IDF, LSTM... Explained. Retrieved from Kaggle

NLP GloVe, BERT, TF-IDF, LSTM... Explained | Kaggle

Rani, V. (2021, April 1). NLP Tutorial for Text Classification in Python. Retrieved December 2022, from Medium:

NLP Tutorial for Text Classification in Python | by Vijaya Rani | Analytics Vidhya | Medium

Nima Beheshti(2022),Random Forest Regression,A basic explanation and use case in 7 minutes

Random Forest Regression. A basic explanation and use case in 7… | by Nima Beheshti | Towards Data Science

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA

LightGBM: A Highly Efficient Gradient Boosting Decision Tree (neurips.cc)