Constrained Language Learning Techniques

Jonathan Bown

MSCS/Fall 2022

Case Studies in Machine Learning

The University of Texas at Austin

ABSTRACT

Feature engineering, dimensionality reduction, cross validation, and ensemble methods with NLP. The 'Feedback Prize' is a popular style of competition on Kaggle. The basic idea of the competition is to take a collection of student essays and score them according to various categories. The rules with these competitions tend to be stricter than the average Kaggle competition with restrictions on runtime, and internet usage. This presented an opportunity to explore alternative methods of prediction that did not rely on an existing pre-trained large language model as well as explore feature extraction techniques and compare performance. The goal was purely academic in comparing a baseline model to a more sophisticated style to better understand the value added of ensemble methods and feature engineering.

1. INTRODUCTION

In September of 2022 a new 'Feedback Prize' competition was launched on Kaggle [6]. The competition was hosted by Vanderbilt University and The Learning Agency Lab. The goal was to "asses the language proficiency of 8-12th grade English language learners" [6]. The competition host has a goal of obtaining a more automated way of evaluating student performance in certain areas and have more information for improving student outcomes. The rules were similar to previous competitions in that the notebook one submits has to run in less than 9 hours, internet must be disabled, and the test dataset had to be constructed from scratch. The competition did allow for external data and pre-trained models to be used. There have been several recent competitions that have a similar structure. The latest competition also added the code efficiency component to the submission evaluation so that those with the most efficient code could still be competitive.

Exploration of features using standard NLP libraries in python was of the most interest. Much research has been done to develop open source libraries that make more complicated features like ‘subjectivity’ or ‘sentiment’ more accessible [1]. Would these features have any bearing on the quality of the student essays? Would combining intuitive features with an ensemble of models have reasonable performance in the competition? These are questions I wanted to explore in depth without simply relying on a pre-trained model.

2. RESEARCH BACKGROUND or LITERATURE REVIEW

I have personally only been exposed to natural language processing techniques recently and am only familiar with how to use large language models like BERT from repositories like HuggingFace. Merging my previous experience with machine learning and natural language required me to start with the basics, experiment with different approaches and evaluate performance. The starting point was extracting basic information from text. This approach may seem naive but the error of the chosen models seemed to be very low right out of the box.

Many approaches to this type of problem simply rely on features extracted from the document-term matrix, term frequency matrix, count vectorizer, word2vec, or Latent Dirchlect allocation (LDA) [9]. These are then fed into a more standard algorithm like XGBoost or CatBoost to generate accurate predictions. LDA is a topic based model that is used to extract meaning from a given corpora [9]. I did not see a relevance here for topic based modeling although it would be interesting to see if there is any predictive power. Other more recent approaches are using transformer based large language models although they usually always mention the massive computational limitations to such a model [10]. This analysis is purely focused on extracting features relevant to evaluating the quality of the essay in a way that works within the competition and resource constraints.

Features from unstructured data like text can quickly enter large dimensions. Standard features like bag of words, TF/IDF, count vectorizers, would quickly hinder performance. Standard dimensionality reduction techniques like PCA are often employed both as an exploratory tool as well as a way to enhance predictions by generating independent components [8]. Many of these methods were explored as part of the analysis.

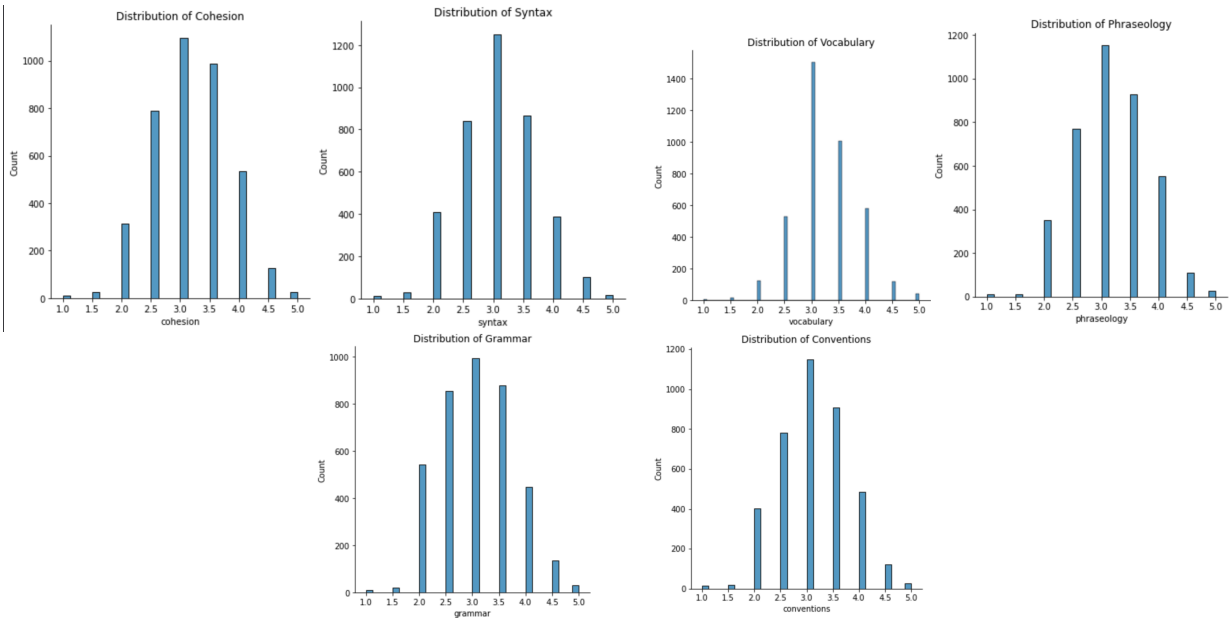
3. MATERIALS / DATA / SOURCES

Kaggle notebooks were used to develop and test competition submissions. Hardware was also an experimental component of this competition. There was a tradeoff between using a GPU and the standard setup on Kaggle notebooks. When the GPU on Kaggle notebooks is activated the access to CPU power is reduced. The GPU was really helpful when training CatBoost and XGBoost. The time to train was reduced by about 1/5. However, as more and more features were added to the model the demand on the CPU became greater than the speedup obtained by the GPU. Using the standard hardware allocation of 4 cores and 30 GB of RAM. Using the GPU kept the runtime at about 7-8 hours which was really close to the cutoff of 9 hours. Using the standard set up kept runtime to under 90 minutes.

The training data consisted of just under 4000 student essays with each essay receiving six different scores. The score categories are cohesion, syntax, vocabulary, phraseology, grammar, and conventions. The evaluation is performed by taking a separate set of test essays and predicting these six scores for each essay. Obtaining a score for a given notebook is performed first by running the entire notebook and then submitting the notebook for scoring. The notebook is given an alternative test file to run through the model and produce a score on the leaderboard. The notebook is a standard Jupyter notebook with a python kernel.

The test data included with the competition data only consists of three example essays that are mostly used to test the construction of the training/test data so that it works properly with the one that is inserted upon submission. The test data used throughout experimentation is just a portion of the test set that is used as the validation data. The distribution of the target variables is shown below. One major point to make from this is that the target variable is not continuous. The target variables span a range of 1-5 but only by intervals of 0.5. This key observation is important for producing accurate predictions. The expected output is discrete rather than continuous. The management of this situation is referred to ‘rounding’ in the competition submissions. It simply means to round up or down to the nearest half. Another point to make is that the distributions of the target variables are all very similar except for vocabulary as can be seen in Figure 1. Some of the features for the model were engineered specifically to capture the vocabulary.

Figure 1. Distributions of Target Variables



4. METHODS

Initial derived features included basic information like ‘number of unique words’, ‘number of characters’, ‘number of capital letters’, ‘number of stop words’, etc. These features were derived by applying lambda functions to the essay column in the training data.The experimentation quickly evolved into many different features after inspecting a lot of the essays one-by-one. There were several packages that already have many different methods for feature extraction from text. Table 1 lists the libraries used for advanced features.

Table 1. Python NLP Libraries

Library	Purpose	Example Features Extracted
TextBlob	Processing textual data	Spelling, parts of speech
NLTK (Natural Language Toolkit)	Standard NLP python library	Sentiment, polarity
Sklearn	Contains a breadth of feature extraction techniques	TF-IDF, count vectorizer
Gensim	NLP Semantic libraries	Word2Vec

Table 2. Feature Descriptions

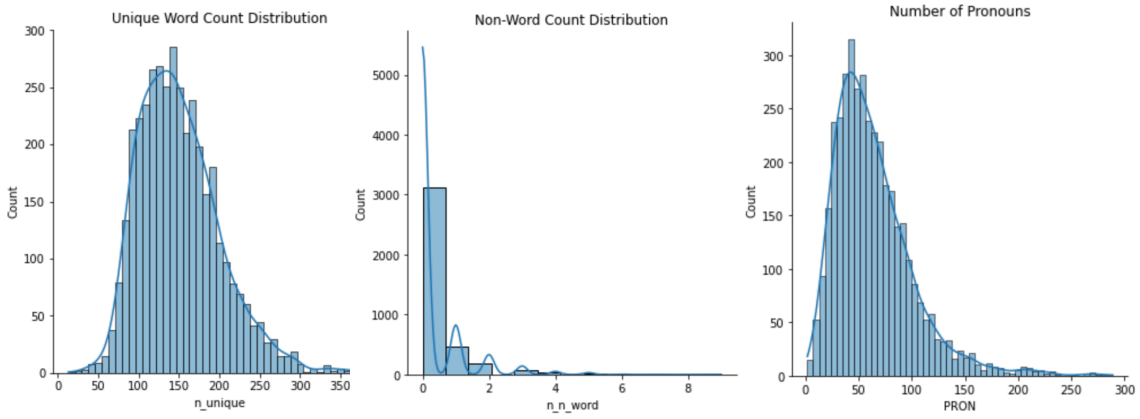
Feature	Description
n_unique	Number of unique words
n_capital	Number of capital letters
n_punct	Number of punctuation marks
n_unique_words_n_stop	Number of unique words less stop words
n_n_words	Number of non-words
noun_phrase_count	Number of noun phrases
Pron	Number of pronounds\
Verb	Number of verbs
Sconj	Subordinating conjunction
noun	Number of nouns
Polarity	Polarity sentiment score
Subjectivity	Subjectivity sentiment score
spell_score	Similarity to spell corrected essay
av_sent_length	Average sentence length
max_sent_length	Maximum length of sentences
min_sent_length	Minimum length of sentences
med_sent_length	Median length of sentences
std_sent_length	Standard deviation of sentence length
num_sent	Number of sentences
compound	Compound sentiment score
negative	Negative sentiment score
positive	Positive sentiment score
neutral	Neutrality sentiment score
char_len	Number of characters

Dimensionality reduction techniques were explored to reduce the complexity of the training data. Principal Components Analysis is a standard practice in NLP problems. There were two main approaches to determine if the method was useful in improving predictions. The first approach set the the desired explained variance ratio at various values to see how the number of components was changing. The second approach used two main components to visualize the loadings of each variable in a biplot shown in Figure 4. The training data used for PCA excluded

the TF_IDF, count vectorizer, and word2vec features. The idea here is that if the dimension of the more explainable features can't be reduced then it would only lose value by adding a much larger space to the training data. The training data was first scaled using a standard scaler in python. The desired ratio was first set to 85% and moved toward 99%.

Standard methods of normalization and standardization were employed during the model development process. Standardization was the better choice for principle components and normalization worked well for constructing the final training dataset used by the models. The MinMaxScaler and StandardScaler were fit to each variable ensuring the values of each variable weren't skewing the predictions. Most variable distributions were evenly spread but there were a few like the number of non-words that were potentially problematic as shown in Figure 2.

Figure 2. Feature distribution examples



The metric used to score submissions is MCRMSE or mean column-wise root mean squared error.

$$\text{MCRMSE} = \frac{1}{N_t}\sum_{j=1}^{N_t}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{ij} - \hat{y}_{ij})^2}$$

where N is the number of scored ground truth target columns and y are actual vs. predicted values. To evaluate model performance I am using RMSE because the overall evaluation metric is the average RMSE for each column.

Stacking is an ensemble machine learning technique that takes the output of multiple models and plugs them into a new dataset. Typically different models are used in each iteration to produce output. In this case stacking was used with the four best performing models out of the box on the training data. These models were applied to one of the columns and the best score was recorded. Table 5 shows the results of models that were attempted. Since MCRMSE is the evaluation metric for the competition the RMSE was used as the metric for comparing model predictions on a single column. The results are shown in Table 5. The dataset that is constructed by combining the output of the top four models is then fed into another regressor to create the submission file. Table 3 below is an example output from these four combined models.

Table 3. Example output of model stack before final ensemble model

Cohesion	Syntax	Vocabulary	Phraseology	Grammar	Conventions
2.723	3.42	3.59	2.92	3.04	2.98
3.34	3.24	3.61	3.39	3.19	3.22
3.44	3.29	3.59	3.42	3.2	3.27
2.68	3.11	3.17	2.41	2.72	2.52

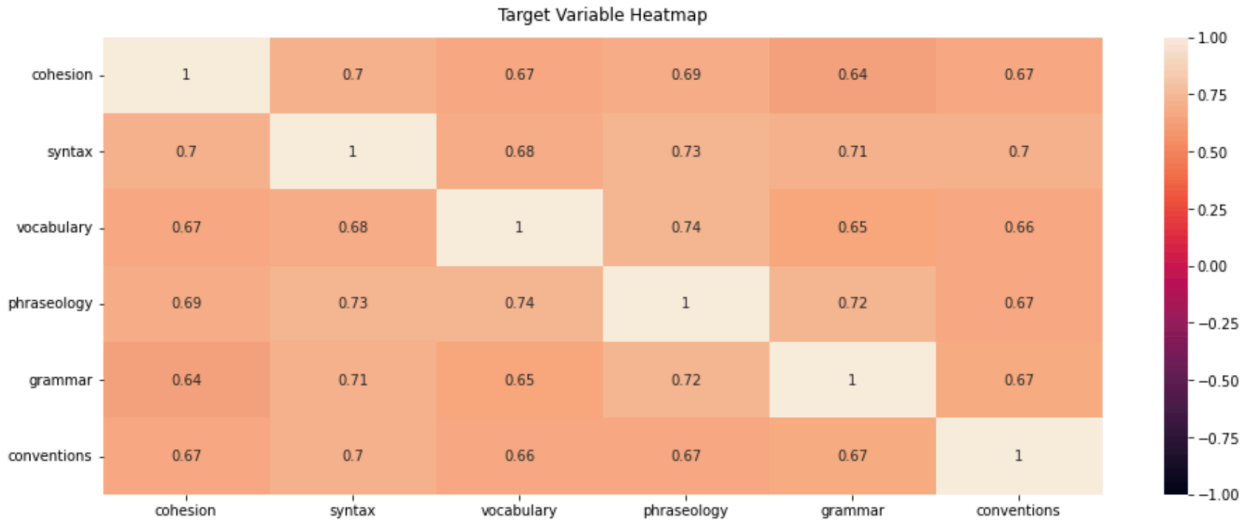
The thinking behind this is that the models by themselves actually do a pretty good job of creating points that are in aggregate close to the actual results.

Once the stacked model is built and has produced predictions those predictions are then used as a training dataset into the final model. The final model then makes six predictions for each essay.

The approach to producing six different prediction columns was a decision between using a standalone model for each column employing six different models or using a Multi-Output Regressor that uses one single model and produces six different output variables. One popular approach with multi-output regression problems is called the ‘direct’ approach. This approach divides the regression into a separate problem for each target variable. This assumes that each

target variable is independent. This may be a weak assumption given the following correlation map between the target variables.

Figure 3. Target variable correlation heat map



The balance struck here is that all of the work to cross validation and model tuning is done on a different variable for each model. The idea there is to prevent the stacked model to overfit onto one variable and just rely on correlations on the others to perform.

After the best models with default parameters are tested and evaluated, cross-validation was employed to optimize the parameter sets of each model to improve generalization. The models that were optimized included LGBM, XGBoost, and CatBoost. The linear regression model was excluded from this procedure. Common and influential parameter sets for these models were researched and a parameter space was developed for each model. The more influential the parameter on predictions such as 'n_estimators' for LGBM were given more possible values. The two approaches of randomized search and grid search were compared. Grid search quickly ran out of working memory for small parameter spaces. Randomized search was the more flexible option to iterate over larger spaces and also gave more flexibility on what percentage of the overall space to sample over. Three-fold cross validation was used on all three models with 100 iterations for each fold totaling 300 total fits. The target variables used for the three-fold cross validation were grammar, vocabulary, and cohesion.

5. RESULTS

Figure 4. PCA Biplot

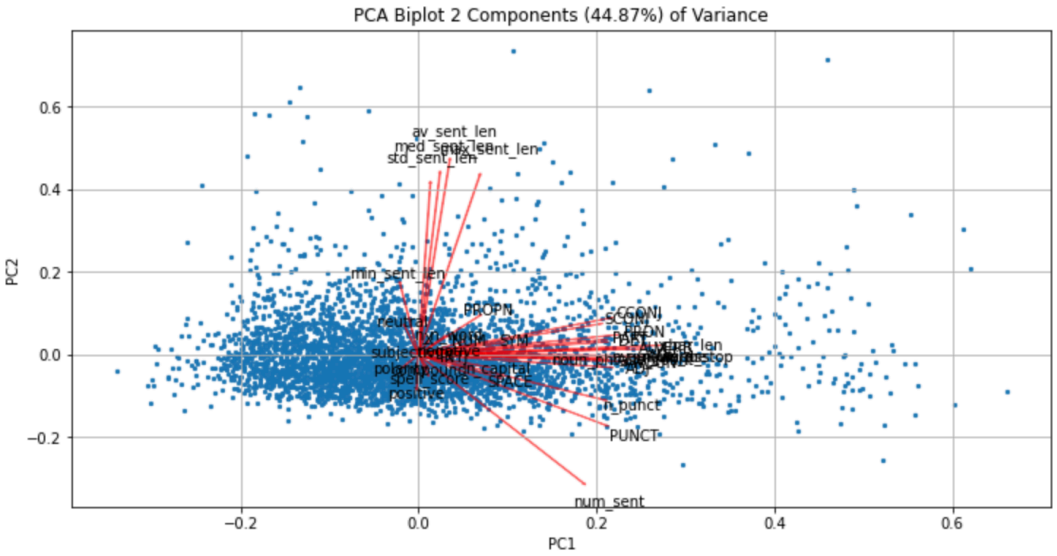


Figure 5. PCA Scatterplot

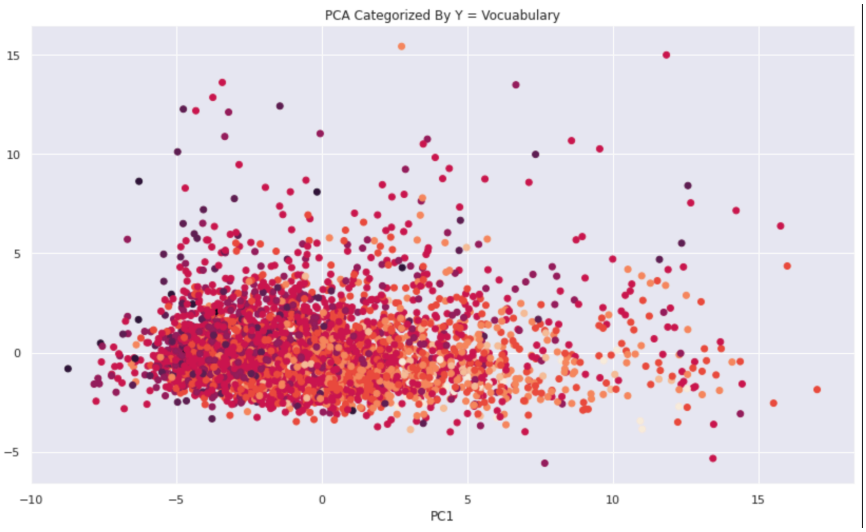


Table 4. PCA Number of Components

# Components	Explained Variance %	# Components	Explained Variance
3	50	4	55
5	60	6	65
9	75	14	85
23	95	30	99

Table 5. Regression Model Performance

Model	Initial RMSE	CV RMSE
Light GBM	0.0023	0.0021
Linear Regression	5.16E-06	-
XGBoost	0.023	0.0022
CatBoost	0.1954	0.0063
SGD	0.537	-
Kernel Ridge	0.533	-
Elastic Net	0.4388	-
Bayesian Ridge	0.347	-
GB Reg	0.401	-
SVR	0.433	-
Random Forest	0.475	-

Figure 6. Feature importance by target variable

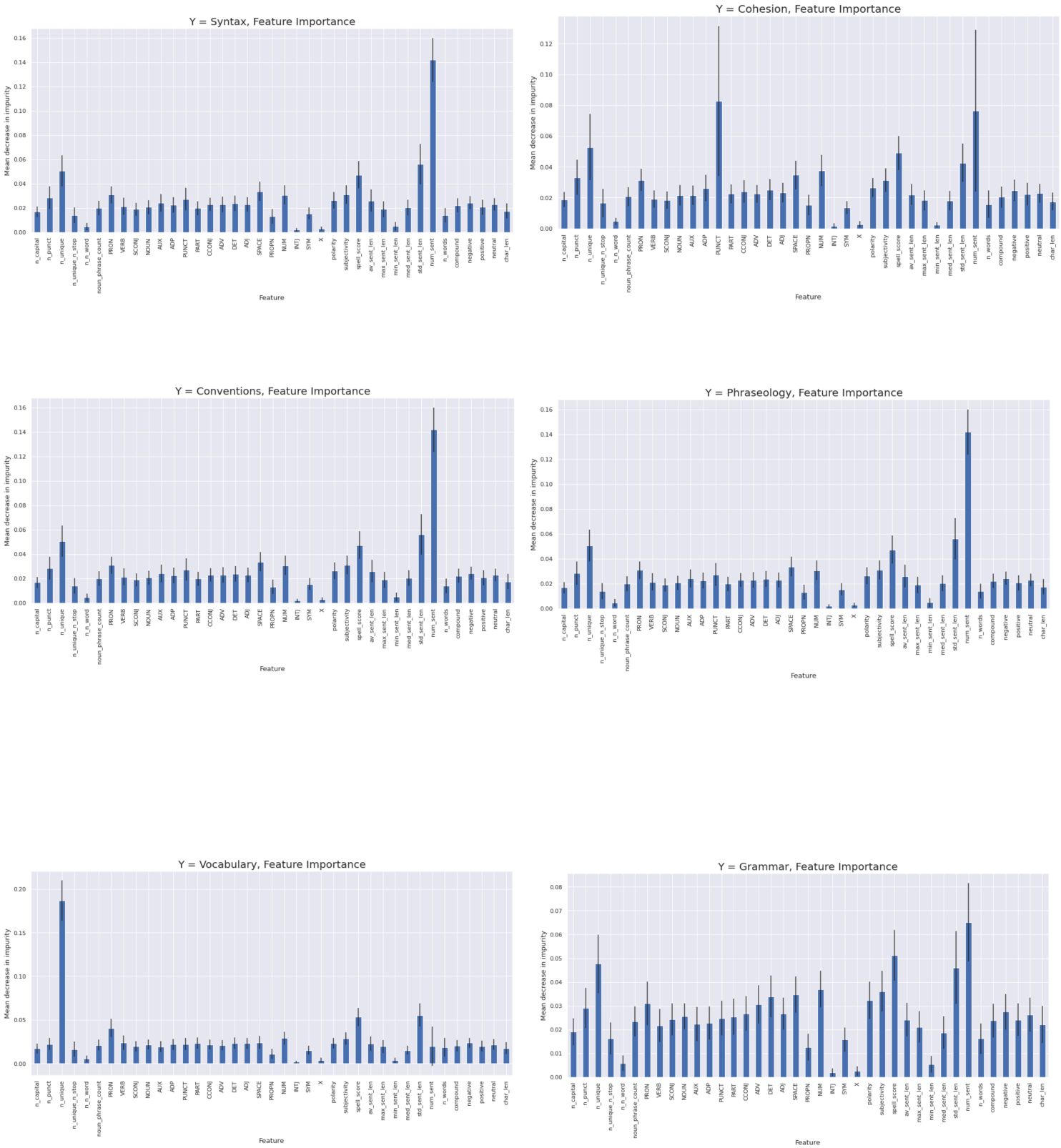


Figure 7. Multi Output Regressor Feature Importance

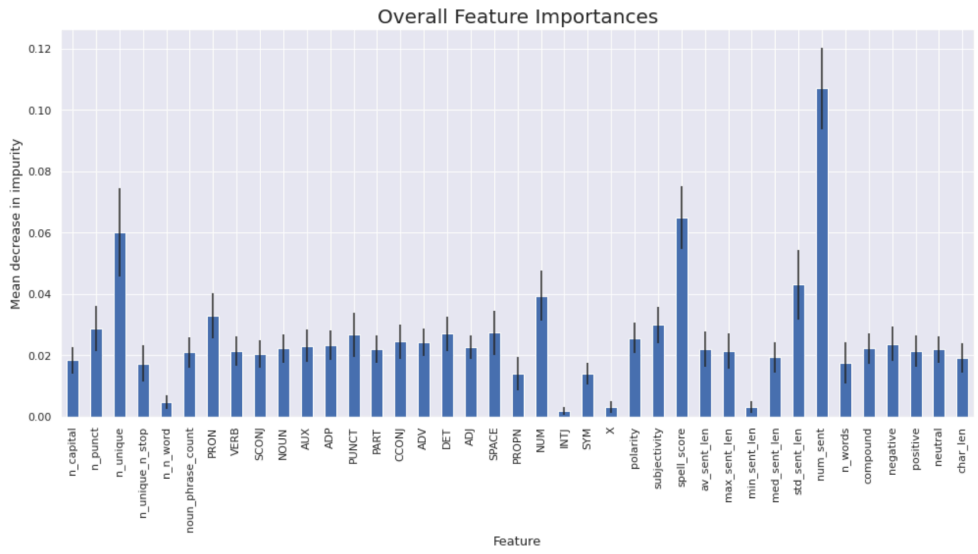


Table 6. Final Notebook Scores

Description	Public Score	Private Score
Single linear regression model, subset of engineers features (Baseline)	0.750361	0.738701
Stacked model w/ default parameters	0.550158	0.543464
Stacked model w/ rounding output	0.572441	0.561344
Stacked model w/ cv parameters	0.558848	0.553751

6. DISCUSSION & CONCLUSION

The best model scoring model turned out to be the default parameters of the stacked model. The difference between my best score and the winners was about 0.12 in MCRMSE. This was surprising given all the analysis that went into cross validation and model tuning. The expected result was that expanding the feature set along with building the ensemble model improved the model by about 20 basis points in MCRMSE. Another surprising result was that rounding the model output to the nearest half reduced the score slightly. Expanding the feature set was backed up by the PCA results. Table 4 shows why PCA was not employed in the final model. To get 99% of the explained variance, 30 components are needed despite having 39 features in total.

The overall goal of this research project was to explore different approaches to rating student essays using readily available nlp libraries. The approach of treating each target variable as independent was probably too generous given the high correlation. The vast differences in feature importances among the six different target variables supports the argument for building a separate model for each target variable and curate the features to exclude those below a certain threshold in the feature importance score. Clearly ‘vocabulary’ and ‘phraseology’ have better performance with variables like ‘num_sent’ than ‘cohesion’. The cross validation and inclusion of many features seemed to overfit to the training data and contribute to the lower scores on the leaderboard. Using PCA exclusively on the more interpretable features could have also been a limitation. Evaluating the potential reduction in complexity by using the principal components on the TF/IDF features could have contributed to the success of the model. Despite ignoring more popular methods like transformers or pre-trained models I am impressed with the performance of some of the variables. I did not expect any of them to perform as well as they did with certain target variables but the upside to using a model like this is very clear interpretability for the user. Model interpretability and explainability to those who eventually employ them will be crucial for competitions like this to continue to be popular and well funded.

I think there is a lot of potential with this approach to build a really fantastic model but it would take some more time to determine which variables are best for each target variable and which cross-validation approach would actual help generalization

REFERENCES

1. *Introduction to entity extraction: What is it and how it works*. MonkeyLearn Blog. (2020, November 9). Retrieved December 3, 2022, from https://monkeylearn.com/blog/entity-extraction/
2. Ma, E. (2018, November 17). *Essential text correction process for NLP tasks*. Medium. Retrieved December 3, 2022, from https://towardsdatascience.com/essential-text-correction-process-for-nlp-tasks-f731a025fcc3
3. Lass, D. (2019, September 4). *NLP punctuation, lower-case and StopWords pre-processing*. Medium. Retrieved December 3, 2022, from https://medium.com/@LauraHKahn/nlp-punctuation-lower-case-and-stopwords-pre-processing-d4888c4da940
4. *How to practice word2vec for NLP using python*. Built In. (n.d.). Retrieved December 3, 2022, from https://builtin.com/machine-learning/nlp-word2vec-python
5. Wu, J. D. (2021, November 18). *PCA - Implementation in Python - Damavis Blog*. Damavis Blog - Data - Machine Learning - Visualization. Retrieved December 3, 2022, from https://blog.damavis.com/en/principal-component-analysis-implementation-in-python/
6. *Feedback prize - english language learning*. Kaggle. (n.d.). Retrieved December 5, 2022, from https://www.kaggle.com/competitions/feedback-prize-english-language-learning
7. Brownlee, J. (2020, August 27). *How to use StandardScaler and MinMaxScaler transforms in Python*. MachineLearningMastery.com. Retrieved December 3, 2022, from https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/
8. Wambsganss, T., Engel, C., & Fromm, H. (2021). Improving explainability and accuracy through feature engineering: A taxonomy of features in nlp-based machine learning. In *Forty-Second International Conference on Information Systems*.
9. Gualberto, E. S., De Sousa, R. T., Thiago, P. D. B., Da Costa, J. P. C., & Duque, C. G. (2020). From feature engineering and topics models to enhanced prediction rates in phishing detection. *Ieee Access*, *8*, 76368-76385.

10. Ormerod, C. M., Malhotra, A., & Jafari, A. (2021, February 25). *Automated essay scoring using efficient Transformer-based language models*. arXiv.org. Retrieved December 3, 2022, from https://arxiv.org/abs/2102.13136