Predicting Essay Scores from Keystroke Data

ANONYMOUS (ONLY ONE AUTHOR), University of California - San Diego, USA

This project explores the prediction of an assigned essay score based on the dataset of keypresses. The dataset is from a Kaggle competition *Linking Writing Processes to Writing Quality* [Franklin et al. 2023]. The methods and feature selection are largely inspired by the paper *Uncovering the writing process through keystroke analysis* [Allen et al. 2016].

ACM Reference Format:
ANONYMOUS (only one author). 2024. Predicting Essay Scores from Keystroke Data. 1, 1 (December 2024), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Keystroke Analysis Dataset

The dataset used by this project is a collection of keystroke data from 2,470 writing sessions and their assigned scores by a panel of essay graders. The goal of this project is to use the keystroke data to predict the essay scores, minimizing mean squared error (MSE).

1.1 Data Format

The dataset is separated into two separate CSV files for convenience.

In the first csv **train_logs.csv**, all of the events to create the dataset are stored. In total, there are more than 8,400,000 events across the 2,470 writers. This means feature extraction must be very intentional. Of the data each event stores, the values I used to calculate my features were the **id** (essay id), **down_time** / **up_time** (the time the keypress / mouse click went down / up), **text_change** (the text that changed as a result of the event (all letter characters are anonymized to 'q' to keep the focus on keystroke analysis), and **word_count**, the count of words after the event, and **activity**, the category of activity the event belongs to.

The second csv **train_scores.csv** contains **id** (essay id) and **score** (the assigned score).

1.2 Exploratory Data Analysis

Our target variable is the **score** for each essay. I will first plot this variable to make sure the distribution seems statistically significant and a good target variable to train on.

Additionally, I need to extract and engineer features from the logs from each essay in order to predict the score. Three features that seem like they may tie in with essay quality and are relatively easy to source are the total number of events in an essay, the total number of deletions in the essay, and the total number of words in the essay (word count at the end of the essay). I created helper methods to extract these features from each essay, and plotted them in histograms as well.

These histograms display a notable trend - all three of the variables are skewed right. This isn't great for a linear regression, as we would want normally distributed features to predict a normally distributed target variable. So, for more accuracy, I'll engineer three more features that take the log of these basic features.

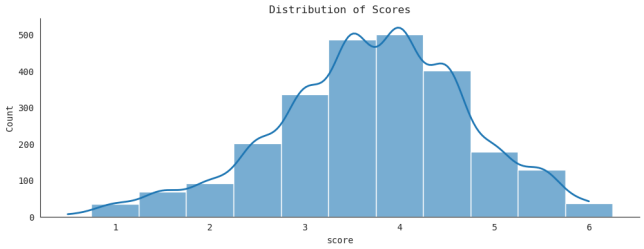

Fig. 1. Histogram & KDE plot of essay scores across the dataset

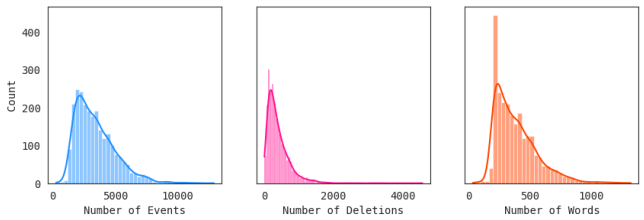

Fig. 2. Histograms for number of events, number of deletions, and number of words

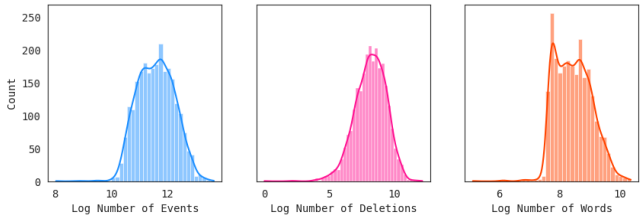

Fig. 3. Histograms for the log of the number of events, number of deletions, and number of words

Now that these are displayed, we can move into fitting the first model in this paper: scikit-learn's linear regression.

2 Prediction

This section will outline my full feature engineering and the models I trained on the data. I started with the basic features from exploratory data analysis and few different SKLearn Linear Regressions, but then I greatly expanded the feature list, and decided to train on the following gradient boosting models: Light Gradient Boosting Machine (LGBM), Extreme Gradient Boosting (XGBoost), Categorical Boosting (CatBoost), and as a bonus, I implemented basic Neural Network architectures from both Tensorflow's Keras and PyTorch.

Author's Contact Information: ANONYMOUS (only one author), anonymous@ucsd.edu, University of California - San Diego, city, state, USA.

ACM XXXX-XXXX/2024/12-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1. Mean Squared Error by feature (as-is and log)

Feature	As Is	Log Base 2
Number of Events	0.68	0.62
Number of Deletions	0.962	0.891
Number of Words	0.603	0.53

For my predictions with the expanded feature set, I implemented a K-Fold Cross Validation setup with five randomized folds.

2.1 Basic Linear Regressions

I trained six different linear regressions on one feature variable each just to get a general idea of feature correlations with target scores. The Mean Squared Errors (MSE)s are displayed in a table.

The MSE is lowest for the 'Number of Words' feature, and I also see that across the board, taking the log of these features has improved their predictive accuracy as I expected. This first model will guide feature selection for future models - we can clearly see that the score of essays was very heavily influenced by word count (likely, higher word count means more effort was put in to producing the writing).

Combining log of all features into one list and running a linear regression with three features, the MSE is **0.523**, not much better than just using the number of words. I should definitely use other models.

2.2 Cross Validation Methodology

This is a good time to talk about the validation method I decided to use for the rest of the papers. As opposed to just creating one validation set and one training set (say, proportions of 0.8 to 0.2), it is generally considered more accurate to use K-Fold cross validation, where all data is used for training and validation. How is this done? With chosen splits of 0.8 and 0.2, I used SKLearn's library to split the data into five folds, and we will validate each model five times, training on the four folds not in the validation set, and validating on the last fold. This gives us five MSEs for each output, and helps to test for overfitting and not leave validation to chance.

2.3 Feature Engineering

There are a lot of features that can be extracted from the keystroke data, and my exploratory data analysis barely scratched the surface. The features I decided to extract take into account the word lengths, typing speed, pauses (do they think while writing?) and punctuation use, where I hope my model can find relations between these variables and the score received.

Here is a more detailed breakdown of all of the features I used for modeling:

2.3.1 Word Length. These features were extracted by extracting all rows of activity **insert** from the dataframe, extracting the character inserted (typed), which was either an anonymized q, a space, or some other punctuation, creating a (rough) string to describe the essay, and using regex to list the "words" (sequences of qs). Then, these statistics would be calculated on this list for every essay.

- **avg_word_length**: The average word length of an essay
- **max_word_length**: The maximum word length of an essay
- **std_word_length**: The standard deviation of word length of an essay
- **skew_word_length**: The skew of the word length of an essay
- **kurtosis_word_length**: The kurtosis of word length of an essay
- **words_longer_4** to **words_longer_9** : Six features counting the number of words used that are longer than 4 all the way to 9.

2.3.2 Typing Speed. These features were extracted by examining differences between **down_time** values for keystrokes within an essay. It's important not to incorporate gaps for thinking into typing speed, so any differences larger than one second were discarded. Then, these statistics were calculated on that list in differences between keystrokes.

- **avg_typing_speed**: The average typing speed of an essay
- **std_typing_speed**: The standard deviation of typing speed of an essay
- **skew_typing_speed**: The skew of the typing speed in an essay
- **kurtosis_typing_speed**: The kurtosis of typing speed of an essay

2.3.3 Pauses. These features were also extracted by using the differences between **down_time** for keypresses, but these merely count the number of gaps above a certain threshold. Maybe writers who have a lot of gaps are thinking a lot, indicating better writing.

- **pauses_0_5_second**: The number of pauses greater than 0.5 seconds when writing an essay
- **pauses_1_second**: The number of pauses greater than 1 second when writing an essay
- **pauses_1_5_second**: The number of pauses greater than 1.5 seconds when writing an essay
- **pauses_2_second**: The number of pauses greater than 2 seconds when writing an essay
- **pauses_2_5_second**: The number of pauses greater than 2.5 seconds when writing an essay
- **pauses_3_second**: The number of pauses greater than 3 seconds when writing an essay

2.3.4 Punctuation Usage. These features were extracted by using the **text_change** feature, and extracting counts for the total number of a specific kind of puntuation in an essay.

- **comma_count**: The number of commas in an essay
- **period_count**: The number of periods in an essay
- **question_mark_count**: The number of question marks in an essay
- **apostrophe_count**: The number of apostrophes in an essay
- **quote_count**: The number of quote characters in an essay
- **exclamation_count**: The number of exclamation marks in an essay

These features, plus the three features from the beginning, total to 30 features for our models.

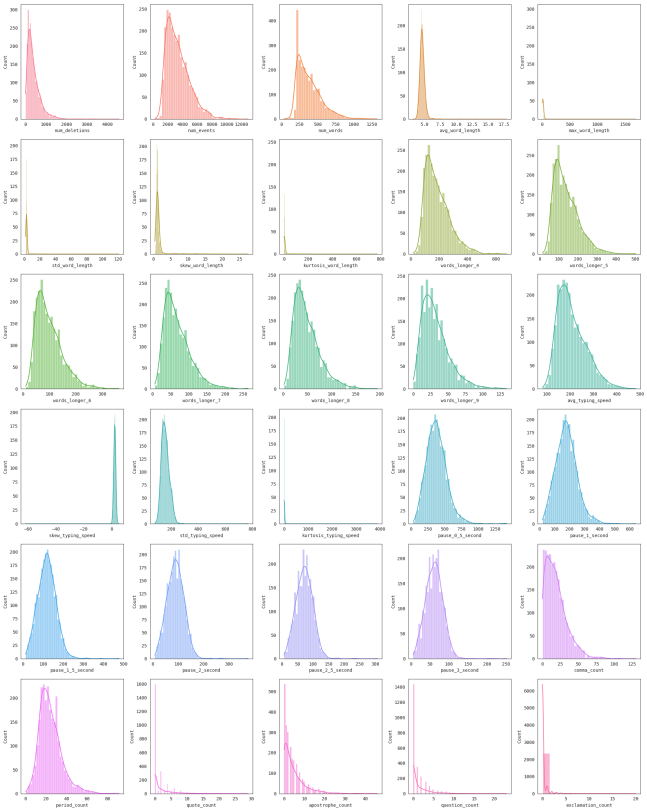

Fig. 4. Histogram & KDE plots for all of the features used for model training

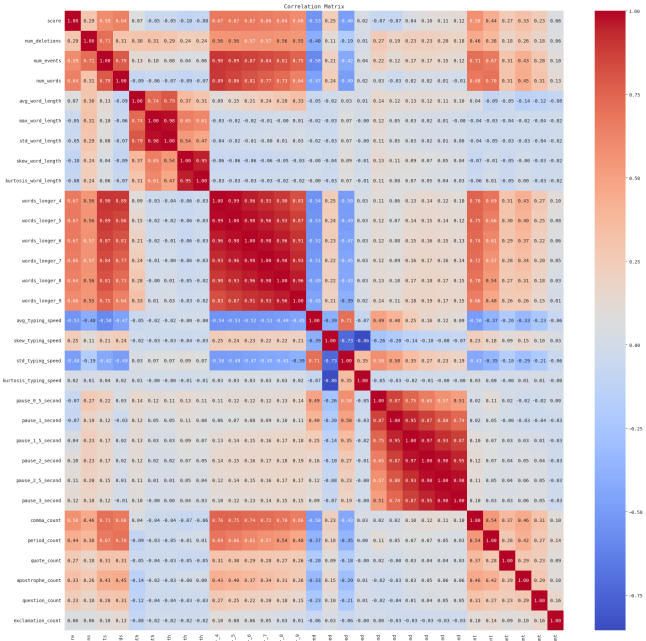

Fig. 5. Correlation Matrix for all of the features used for training

The correlation matrix has some notable trends that likely explain later results for feature selection. I'll spare details, but in general we see strong correlations for all of the features you would expect to increase with essay length (word count, comma count, period count), and then some interesting near zero correlation between average word length / high length words and these essay-length features.

3 Models

3.1 Linear Regression with the New Features

It was interesting to build these features, so I thought a good baseline would be to chuck the thirty features into a linear regression model. I decided to take the log of everything first because we saw it work earlier.

I first did a five-fold cross validation, receiving MSEs of 0.883, 116.35, 0.457, 0.396, and 0.487. Finding the variance and outliers very weird, I then did a two-fold validation, getting MSEs of 0.48 and 5.41 for each of the folds. Clearly, the linear regression was super susceptible to certain outliers, but I didn't want to continue optimizing the model further - there were better ways to address this with other models.

Table 2. Mean Squared Error 5-Fold for SKLearn Linear Regression

Fold:	1	2	3	4	5
MSE:	0.883	116.35	0.457	0.396	0.487

Table 3. 5-Fold Cross Validation Mean Squared Error for LGBM

Fold:	1	2	3	4	5
MSE:	0.37	0.435	0.38	0.41	0.4

3.2 Gradient boosting Algorithms

3.2.1 Hyperparameter Tuning. For all of the gradient boosting models), I decided to use **Optuna** to optimize the parameters for mean squared error. **Optuna** is very useful, because it can automatically perform a search within a search space for given hyperparameters, and it performs pruning based on learning curves, so if the learning curve is bad it can halt the trial without running all the way through and save time.

3.2.2 Light Gradient Boosting Machine (LGBM). LGBM is a gradient boosting algorithm that uses many weak learners to create weightings for different features and predict the target variables. I chose LGBM Regression as my first gradient boosting model because it's very fast and it will give me an interpretable feature importance graph, which I had iteratively used to reach the current feature list.

I fit the model and optimized the parameters using Optuna, and the final 5-fold MSEs averaged out to 0.399.

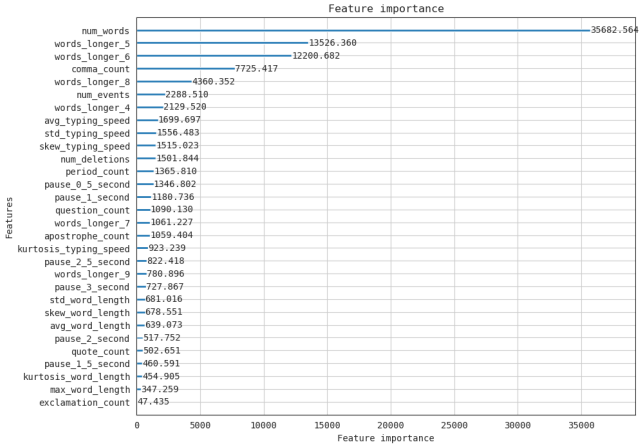

Fig. 6. Feature importance for LGBM Regressor

Table 4. 5-Fold Cross Validation Mean Squared Error for XGBoost

Fold:	1	2	3	4	5
MSE:	0.36	0.441	0.375	0.397	0.401

If we look at the feature importances, we see that LGBM really prioritized the number of words, and then certain features about word lengths, and also the comma count to predict writing quality. Typing speed features were also pretty important.

3.2.3 Extreme Gradient Boosting (XGBoost). XGBoost is similar to LGBM, but doesn't prune trees and only grow its most accurate leaves, but instead grows all branches each level. This makes it more robust against overfitting, so I decided to also tune XGBoost hyperparameters and validate it on the data.

The 5-fold MSEs for XGBoost averaged to 0.395, which is very accurate!

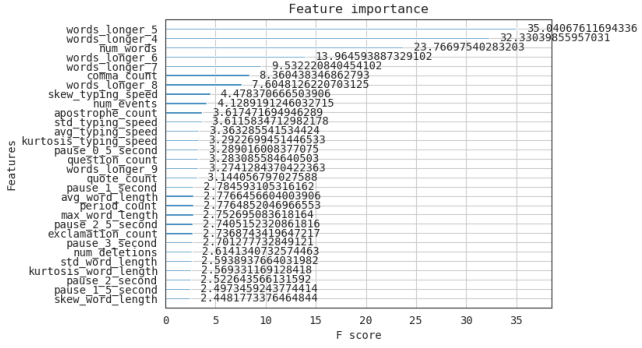

Fig. 7. Feature importance for XGBoost Regressor

From XGBoost's feature importance graph, we can observe the odd phenomenon where XGBoost prioritized features significantly differently from LGBM (words longer than features were the most important by a lot, before number of words), which is puzzling, as MSES for each fold (where we used the same seed just for comparability) were very similar.

Table 5. 5-Fold Cross Validation Mean Squared Error for CatBoost

Fold:	1	2	3	4	5
MSE:	0.364	0.442	0.375	0.392	0.402

3.2.4 Categorical Gradient Boosting (CatBoost). Categorical Gradient Boosting is the original gradient boosting method, and it's the most robust and standard one. It uses symmetrical trees and is very good against overfitting. I tuned the hyperparameters with **Optuna** and then cross validated it.

The 5-fold MSEs for CatBoost averaged to 0.396, very similar to the previous two.

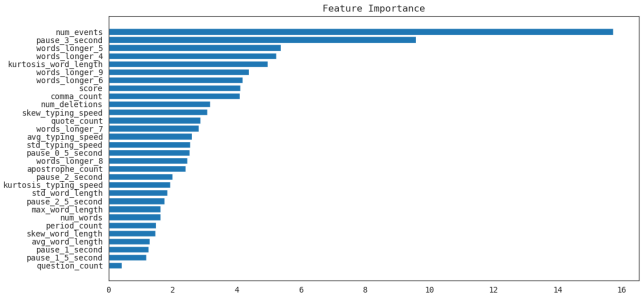

Fig. 8. Feature importance for CatBoost Regressor

CatBoost prioritized still different features from the previous two GBMs, while reaching very similar MSEs. so odd!

3.3 Feed-Forward Neural Networks

Because the three boosters prioritized different features, I wondered if a neural network could make more sense of the features. I trained two neural networks from two different libraries on the features, and

3.3.1 Pytorch. I used pytorch's sequential neural network to fit on the 30 features and try to minimize MSE. Cross validation was not performed with this as it would take too long with neural networks. the lowest validation MSE achieved was 0.438.

3.3.2 Tensorflow (Keras). I also used Tensorflow with Keras in a nearly identical neural network architecture, although it seemed to perform better with different hyperparameters. The lowest MSE achieved was 0.43.

4 Related Works

The primary related work the analysis methods and feature selection is based on is *Uncovering the writing process through keystroke analysis* [Allen et al. 2016]. This paper examined a study of 131 participants, and identifies some feature selection attributes that I utilized in this prediction exercise as well.

This work examined basic aggregated metrics such as verbosity, total pauses, and the number of backspaces, which provided a high-level view of writing behaviors. The linear regressions examined in this paper closely mirror my results, where features expected to correlate with writing length and higher word length were very predictive of writing quality.

This work inspired the incorporation of pause counting features (which CatBoost found to be useful), and the inclusion of standard deviation / skew based features to examine more temporal differences in certain features. Both of these were useful for the machine learning models, and contributed to a more accurate study.

5 Results and Conclusion

5.1 Methods Comparison

We see from the data collected that boosting methods all performed similarly, and all performed significantly better than the neural networks. This can partially be explained by blaming me (I'm not too experienced with neural networks), but I believe the boosting trees' robustness against outliers also likely explains the higher accuracy. Especially if we take into account the linear regression (which if you can remember had loss of 100 on one fold, which devastated its average here), it seems a particular small amount of outliers threw off some predictions. Of the boosting methods, XGBoost performed the best, although the differences are minimal and can likely be corrected with tiny hyperparameter adjustments.

5.2 Feature Comparison

The Boosting methods' feature importance graphs reveal some key insights to draw between writing processes and writing quality. From just the number of words, number of keypresses (events), and the length of words, we can see that machine learning models are able to draw strong predictions for essay quality. As human writers, we know these are not, in and of themselves, definitive of a top quality essay - but they definitely appear to be correlated. Some features that did not show to be as important were the number of words, or the time spent writing the essay. This indicates that good writers were largely differentiated from poor writers by their writing quantity *and* their vocabulary (word lengths). We also do see modest usage of typing speed in all of the boosting models' feature selections.

Of course - the curse of neural networks is we do not know how they achieved their predictions.

5.3 Alternative Analysis Comparison

This data comes from a Kaggle competition, so there are a lot of predictions that I found were far more accurate than mine. They used RMSE in the competition, but converting the top solutions' RMSE to MSE gives a low end MSE of 0.325. Their methods largely involved extracting the words and sentence structures and feeding them into a pre-trained LLM (usually Deberta) to get results. Alas, while this was more accurate, I felt it was also sidestepping my goal of this project which was to examine keypresses (And not writing structure), so I neglected to adopt this method. With the Kaggle competition, most implementations were based on copying others' implementations and adding a little bit or making some differnce - I wanted to work from scratch so I only represented around 30 features, but some complex representations had upwards of 200. Still, the largest correlations of features were similar to mine. [Franklin et al. 2023]

To speak on methods - other Kagglers found similar results, where ensemble models were most accurate with the temporal NLP with deberta in conjunction with a boosting method, and not a neural network.

References

Laura K. Allen, Matthew E. Jacovina, Mihai Dascalu, Rod D. Roscoe, K. Michelle Kent, Aaron D. Likens, and Danielle S. McNamara. 2016. ENTERing the time series SPACE: Uncovering the writing process through keystroke analysis. In *Proceedings of the 9th International Conference on Educational Data Mining (EDM 2016)*, Tiffany Barnes, Min Chi, and Mingyu Feng (Eds.). International Educational Data Mining Society, Raleigh, NC, 22–29.

Alex Franklin, Jules King, Maggie Demkin, Perpetual Baffour, Ryan Holbrook, and Scott Crossley. 2023. Linking Writing Processes to Writing Quality. https://kaggle.com/competitions/linking-writing-processes-to-writing-quality. Kaggle.

Table 6. Final Mean Squared Errors for Different Methods

Method:	Lin. Reg.	LGBM	XGBoost	CatBoost	Pytorch NN	Tensorflow NN
MSE:	23.6	0.399	0.396	0.395	0.438	0.43