MCM Problem C

Marcus Lenz, Aidan Shinfeld, Taimur Shaikh

1/8/2024

Abstract

We chose to analyze MCM problem C, with the goal of understanding and replicating the results of winning papers, that is check the mathematical model, verify the sources for any additional data used, program the simulations and validate the results. We were successful in our efforts and this write up will analyze the key methodologies used by the winning authors.

1 Intro??

2 Gated Recurrent Unit

the relative error rate of 2.1569relative RESE of
6.4957%, indicating a good accuracy of the model predictions. The predicted interval for the number of reported results on March 1, 2023 is
20367 ± 2.01569%

The GRU model was effective in fitting to the data related to the number of reported results per day. This is essentially a time-series problem, akin to predicting future stock price closes, and so models with built-in memory states (like LSTM or GRU) are well-suited to this sub-problem. An LSTM might have been able to make more accurate predictions due to it separating short-term and long-term state whilst sacrificing compute speed, but for a dataset of this size, the GRU performed well. After preprocessing the data as per the specifications of Paper 1, we began implementing a GRU model in Python using PyTorch, pandas, numpy and scikit-learn. We took heavy inspiration from online sources, notably a YouTube tutorial detailing how to use PyTorch to make an Amazon stock close prediction model. We also consulted DataCamp on explaining the inner workings of GRU cells and how they different from normal RNNs as well as LSTM cells (sources all cited).

Our model fit to the dataset in a similar way to that shown in the paper. There was some guesswork, tweaking and intuition involved in certain aspects, such as choosing the model hyperparameters. We ended up choosing standard, logical values for things like learning rate, choice of optimizer and number of epochs to train for. We also accounted for the author's choice to drop Christmas Day as it is an outlier. Doing this, our statistical analysis lined up closely with what was presented in the original paper. Our model did end up doing worse than the original one, as is proven by the higher RMSE calculations. This is likely due to the aforementioned guesswork for hyperparameters; given enough time, some optimization could be done on these parameters to find the best combination, rather than us manually tweaking them to find what works best. Just by looking at the results for our training set, our model also seemed to besubject to more overfitting than the original data. **NOTE: it doesn't yet because of the percentage point stuff**

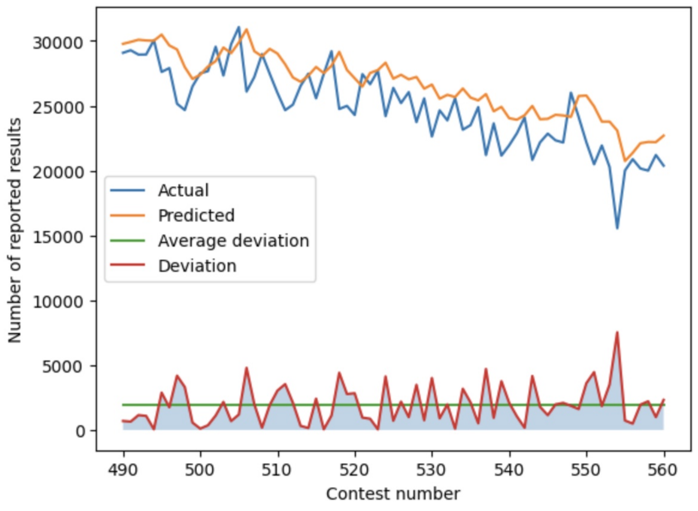


Figure 1:

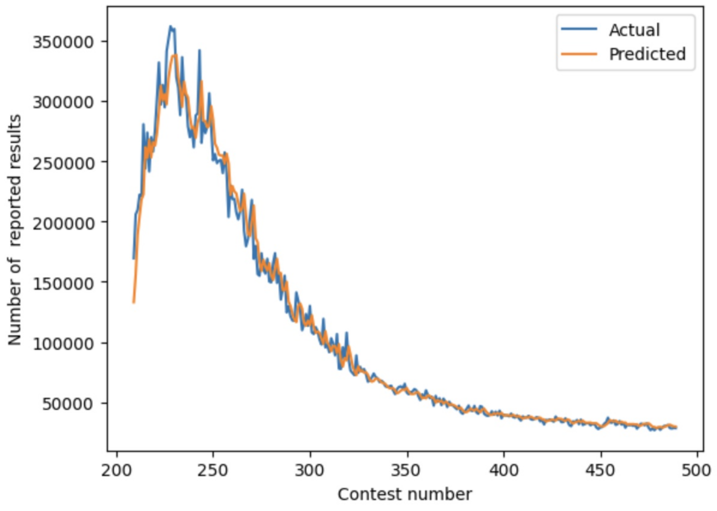


Figure 2:

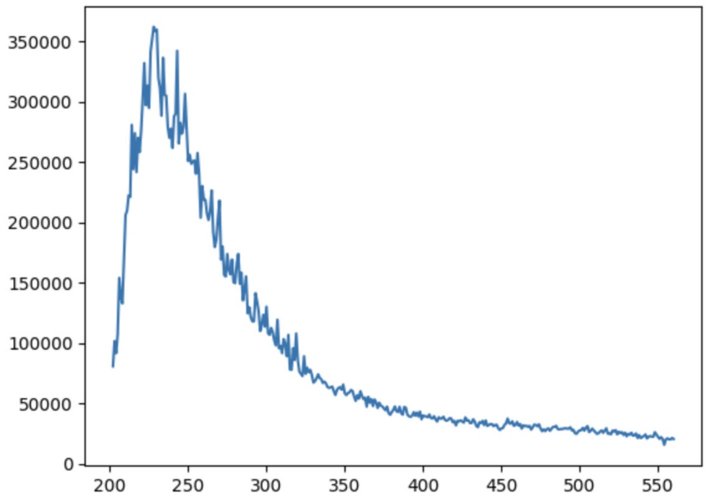


Figure 3:

3 Relationship of Word Attributes and Scores from Percentage

In this section, the paper conducted a data analysis on four attributes of words and *score*, which is derived from the percentage of scores. Through regression analysis, they found found that word frequency and letter frequency of words are linearly correlated with score. Through boxplot analysis, they found that score of words with different letter repetition patterns has some differences, while the score of words with different parts of speech does not show significant differences. We setout to confirm their results.

First, we calculated the score using the same equation they used:

$$score = \omega_1 * \sum_{i=1}^{6} i * p_i + \omega_2 * p_X \tag{1}$$

where, p_i denoted the percentage of i tries and ω_1 and ω_2 equaled 0.5.

They state they then collected the frequencies of the words and letters using Google Ngram Viewer `https://books.google.com/ngrams/` and Python. So I built a Python script to go to the website and pull the data from 2018 or 2019 of the percentages. There were discrepancies in the percentages for the letters frequencies. So we decided to use the values from `https://en.wikipedia.org/wiki/Letter_frequency`.

Now with these values, we then calculated the letter frequencies for each word, which was a summation of the percentage for each character in the words.

Then we ran Pearson and Spearman correlation tests on the Word Frequencies and Letter Frequencies against the scores. The results are in Table 1 below.

Regression with *score*	Pearson	Spearman
Log(Word Frequency)	-0.252	-0.256
Word Frequency	-0.179	-0.256
Letter Frequency	-0.377	-0.353

Table 1: Regression analysis between word attributes and *score*

We found that our correlation values differed from the values of the paper, but had similar characteristic. For the word frequency and score correlation, the Pearson was larger than the Spearman. For the letter frequency and score correlation, the Spearman was larger than the Pearson.

We then plotted these regressions and they plotted the log of the Word Frequency. Which, allows their Pearson correlation to increase significantly.

We concluded from the Pearson and Spearman values that a certain linear correlation between word frequency and *score* and letter frequency and *score*. This was consistent with the paper.

They then did a comparison of words with repetitions and no repetitions. We then filtered the words that had repetitions. We then plotted a box plot to compare the scores for words with repetitions and words with none. We also plotted a density plot.

Median Difference	Box Length Difference	Outlier Difference
0.13	0.02	2

Table 2: Boxplot analysis between Repetition of Letter and *score*

Our results were consistent with the papers, however, the difference between the box length difference was the only thing that was significantly different. However, our plots look exactly identical so they could've made a small mistake in their calculation.

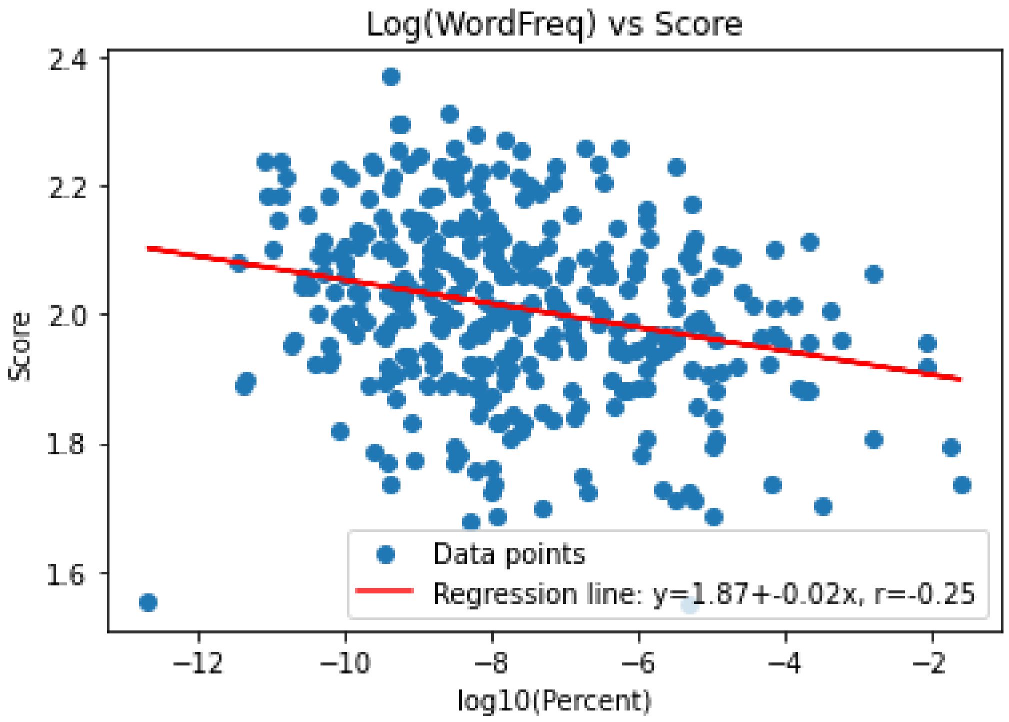


Figure 4: Word Frequency

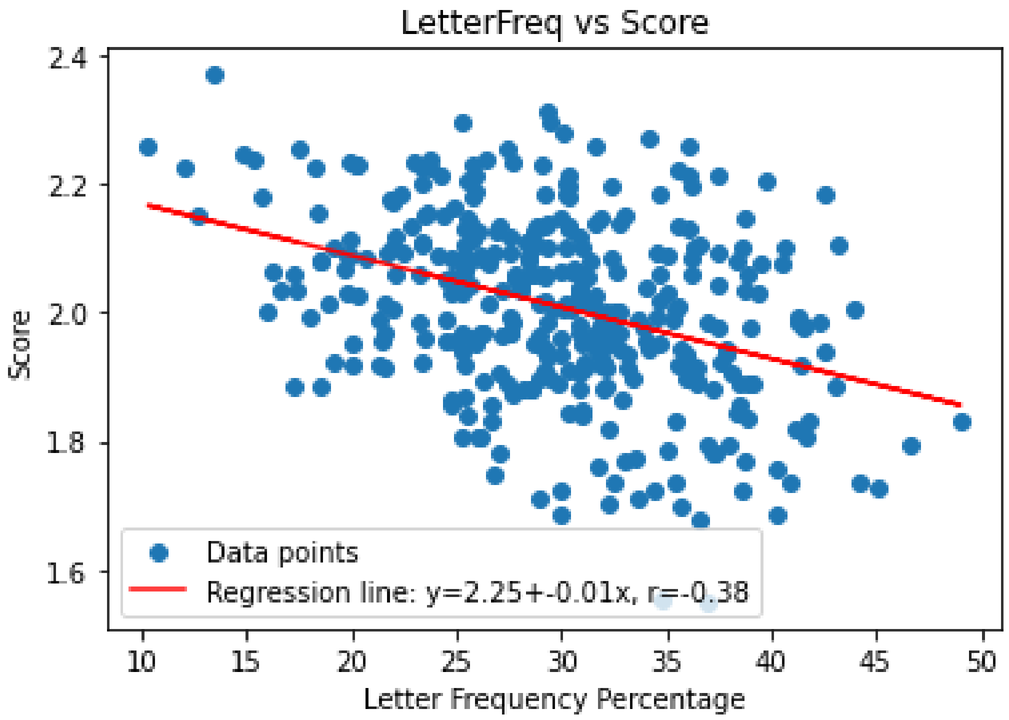


Figure 5: Letter Frequency

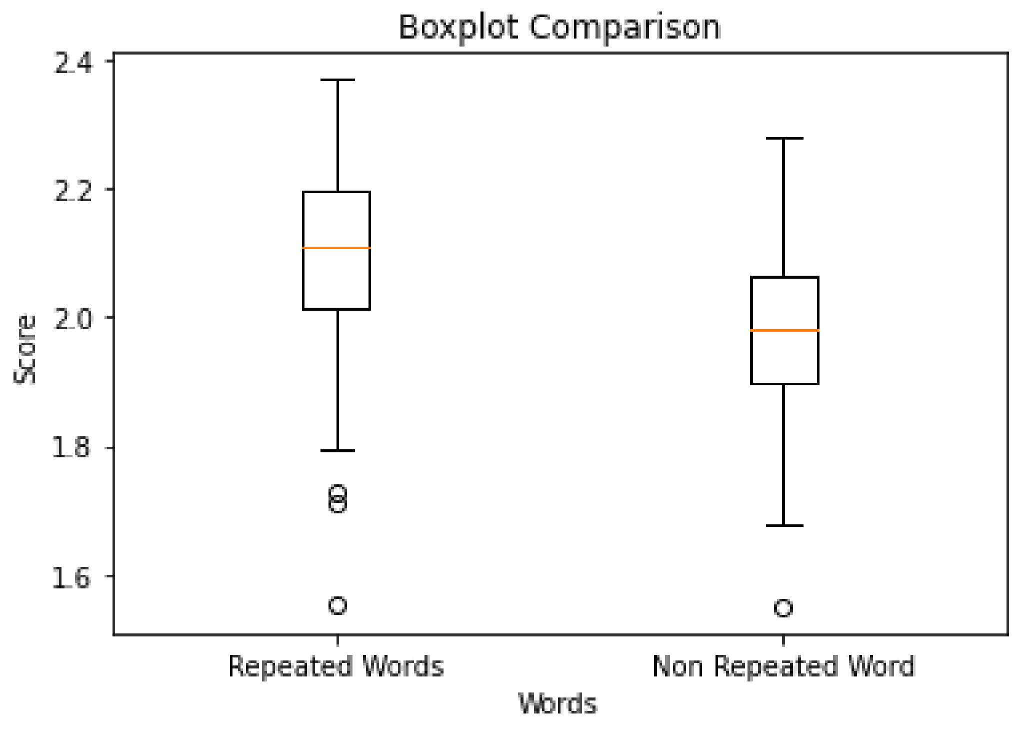


Figure 6: Word Frequency

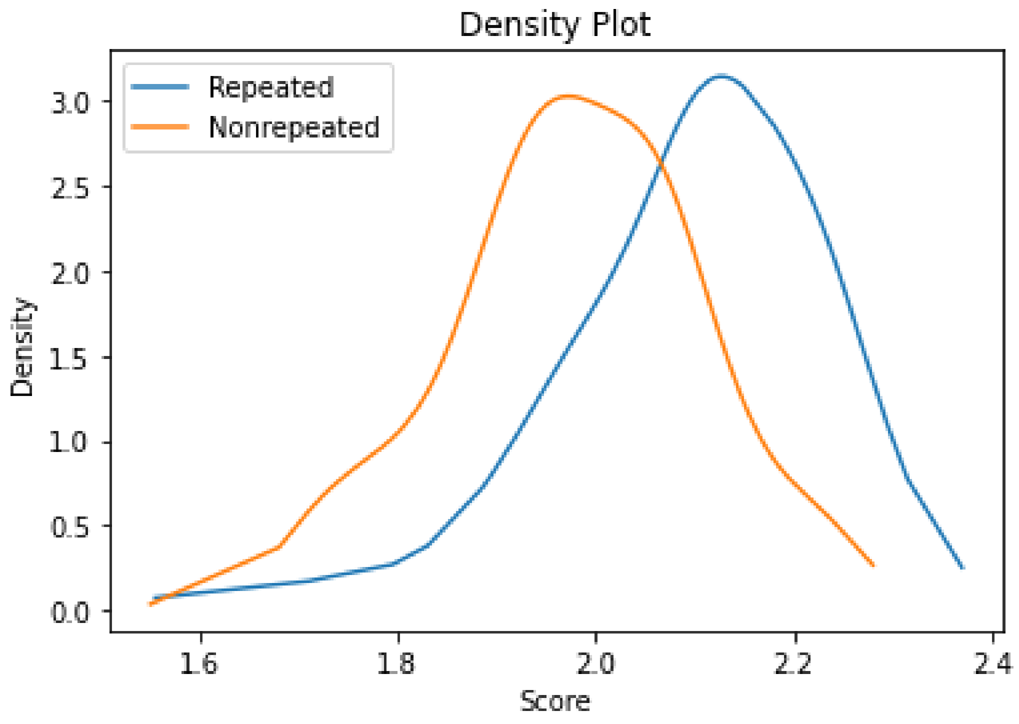


Figure 7: Word Frequency

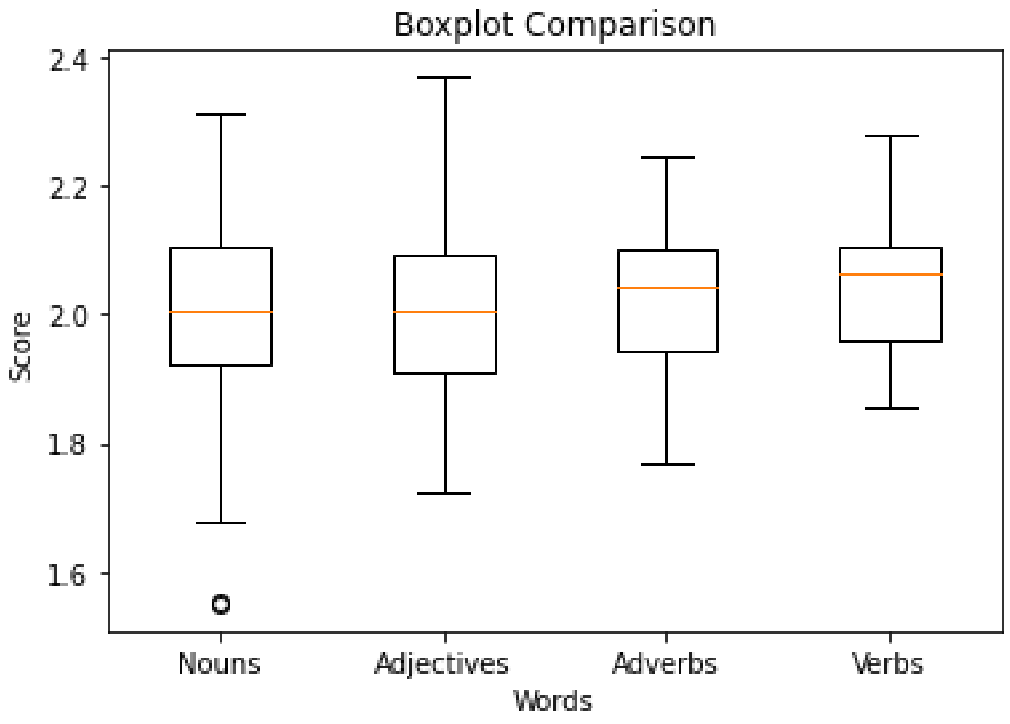


Figure 8: Word Type (POS) Box Plot

It can be seen from the density plots that there is a significant difference in distributions, therefore, we also believe like the paper that there is a certain difference in the *score* situation between words with and without repetitions.

Lastly, they compared the part of speech attribute of the word. They split the words into four categories: Nouns, Adjectives, Adverbs and Verbs. They did this using a Python library called Natural Language Toolkit.

Using this library, we filtered the words into these four categories. However, we saw that only 340 words fell into those categories when the data set included 359 words. They did not mentions anything about this so we did not include those 19 words.

We then plotted a box plot for each of the categories as well as density plots.

Median Diff1	Median Diff2	Median Diff3
0.0	-0.03	-0.05

Table 3: Boxplot analysis between word type and *score*

Note: Median Diff1 is the Noun vs Adjective, Median Diff2 is Noun vs Adverb and Median Diff3 is Noun vs Verb.

We found our results to differ from the paper for the box plots. However, our curves were similar with the Verb having the highest density but the curves were not a one to one correspondence. We were not able to conclude if the *score* of the words had significant differences across difference POS. This was a similar conclusion to the paper.

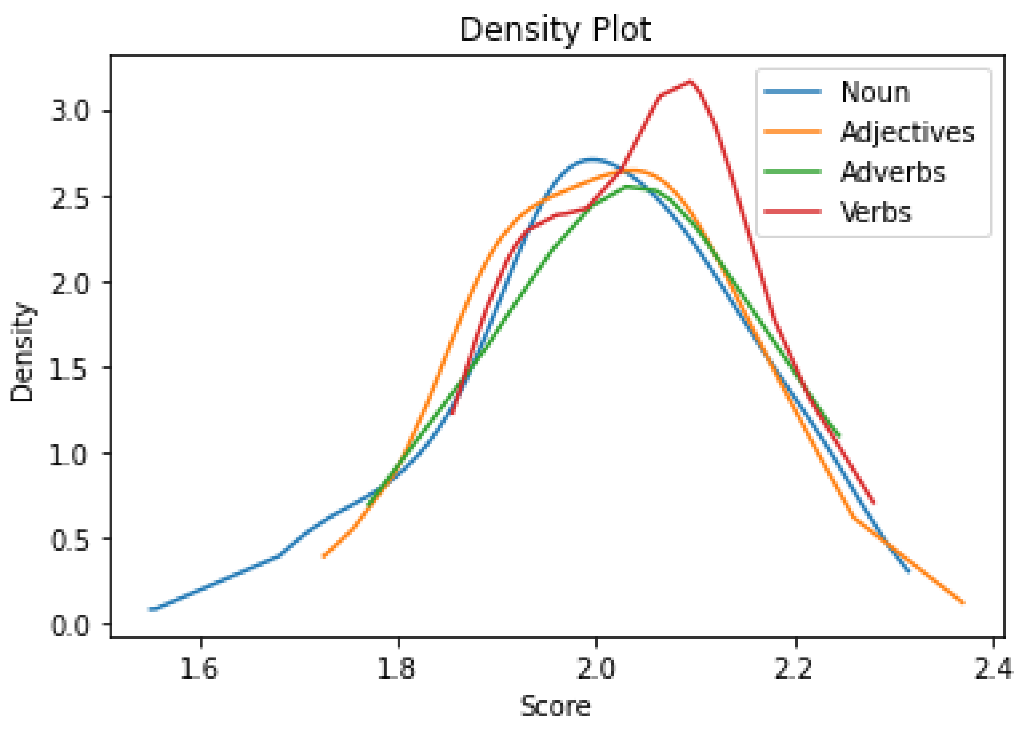


Figure 9: Word Type (POS) Density Curve

4 Prediction for ’EERIE’ on March 1st,2023

To make a prediction for the results of the word ”Eerie” on a given day, the authors of the original paper used a method referred to as Grid Search CV.

GridSearchCV is a parameter optimization algorithm commonly used to fine-tune hyperparam- eters in machine learning models to optimize their performance. It iterates through a specified parameter grid and trains and evaluates the model for each possible parameter combination, ulti- mately outputting the best parameter set and corresponding model performance metric.

Random Forest is a machine learning algorithm that is an ensemble of multiple decision trees. The training process of Random Forest is based on multiple decision trees, with the algorithm randomly selecting a subset of features for training in each decision tree. During prediction, Random Forest aggregates the predictions from each decision tree by averaging or voting to obtain the final prediction.

Combining these methods we are able to train and predict our random forest machine learning model using the best hyper parameters which.

4.1 Predicting EERIE

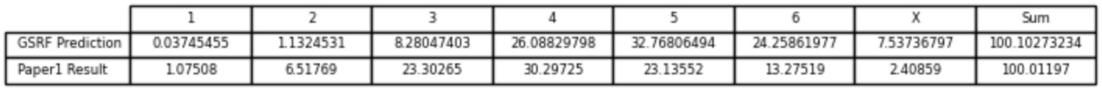
The input parameters for random forest were $(f_{word}, f_{letter}, rep)$ using GSRF algorithm to predict $(1, 2, 3, 4, 5, 6, X)$ for the ’eerie’ on the given date.

$$(f_{word}, f_{letter}, rep) \xrightarrow{GSRF} (1, 2, 3, 4, 5, 6, X) \tag{2}$$

Given the attributes for the word 'eerie'

$$f_{word} = 0.00023\%, f_{letter} = 0.418799, rep = 1.5 \tag{3}$$

Where $rep = 1.5$ represents a word with repeated letters. Executing the algorithm we obtained the following results;



	1	2	3	4	5	6	X	Sum
GSRF Prediction	0.03745455	1.1324531	8.28047403	26.08829798	32.76806494	24.25861977	7.53736797	100.10273234
Paper1 Result	1.07508	6.51769	23.30265	30.29725	23.13552	13.27519	2.40859	100.01197

Figure 10: 'Eerie' Prediction Results for March 1st,2023

We can see though our results follow the same trends, we do not agree in our numbers. This is a direct result of the data used to train our GSRF model. Where our data experienced discrepancies in f_{word} and f_{letter}. Further visualizing our differences, and trend;

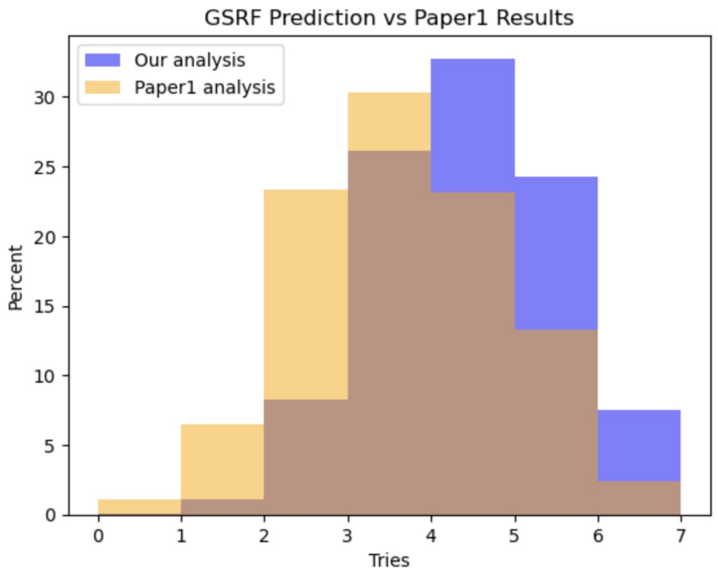


Figure 11: Histogram prediction results

This figure helps us visualize the discrepancies in our results. While different, the trend is the same and values are a direct result of our data, which also had discrepancies. Analyzing methods only we were successful in our efforts to replicate results by implementing the same methods.

5 Sources and Work

https://github.com/taimurshaikh/cummw-mcm-group-7/tree/main