| Problem Chosen | 2023 MCM/ICM Summary Sheet | Team Control Number |
|---|---|---|
| C | | 2318804 |

# The Sparks When Flipping the Wordle


## Summary

Players try to guess a five-letter word several rounds in the Wordle game. Wordle's simple rules and moderate difficulty attracted many players in a short time. This paper gives a methodology to quantify the word attributes and models the changing trend of the number of reported results. We use this information to build models to study the attributes affect the difficulty of guessed words and predict the result distribution for any given solution word.

The trend of the number of reported results over time shows various patterns at different stages. We propose a breakpoint detection regression algorithm to find the breakpoints between stages, and then make reasonable assumptions about the growth rate based on the characteristics of the data growth trend. We fit the historical data to estimate the model parameters. The model predicts that the number of reported results of March 1, 2023, is 14614, and the prediction interval with 95% confidence is [11133, 18095].

The percentage of Hard Mode increases over time and shows a stable trend in tail. We find that the word attributes of the past few days will affect the percentage of the next day through an AR(1) process. The coefficient of Gini index (used to describe the dispersion of letters) in the attributes in the model is significant, and the AR(1) model itself can also pass the significance test. The impact of word attributes will decay exponentially in a few days. We give an intuitive explanation of this mechanism, that is, several easy puzzles in a row may drive players to challenge the Hard Mode next time; In the same way, continuous difficult puzzles will make players give up Hard Mode.

We model the results distribution using the multinomial logits and estimate the parameters by minimizing the cross-entropy loss. The model needs the true distribution data in the past to construct features, we use the rolling prediction method to fill the gap from now to the target date. The uncertainty is mainly due to the unknown of the answer words and the true results distribution in the future. We use the Bootstrap technique to solve this problem by sampling the unused solution words and aggregating. The point estimation of distribution prediction for the word EERIE on March 1, 2023, is [1e-6, 1e-3, 0.050, 0.301, 0.434. 0.196, 0.018].

Finally, we use the average number of tries to split the words into Easy, Normal, and Hard, and use attributes constructed to establish a Random Forest classification model. The validation F1 Score can reach 0.75. We use Shapley value to identify how the attribute of a word contributes to the classification task. Among all attributes, Gain (that is, the average decline of the system uncertainty for all possible tries) has the powerful explanation ability for the difficulty. The classification result for the difficulty of EERIE is Normal.



**Key words:** Breakpoint Detection, Time Series Analysis, Multinomial Logits Model, Rolling Prediction, Bootstrap, Shapley Value

# Contents

# 1. Introduction

The Wordle game will take a word containing five letters as the solution word every day. After each round of guessing, the player receives a vector about the correctness of the letter in each position, which contains three states (green indicates correct, yellow indicates that the word contains the letter but is not placed in the right position, and gray indicates that the word does not contain the letter). After each round of guessing, players can adjust their strategies based on feedback to reduce the search space for feasible solutions. Most players can win the game after four rounds of guessing.

If we only start from the rules of the game, we can study the guessing process of Wordle word as a random system, and the feedback received from each guess provides information to reduce the uncertainty of the system. If all five letters are gray in a guess, novice players may feel that this is a "failed" try. In fact, this situation is very helpful from the perspective of reducing the randomness of the system. It provides a great hint that we do not have to consider any words containing these five letters in the subsequent tries. If we know the set of all possible answers, based on the idea of minimizing the uncertainty of the system each round, we can get a locally optimal greedy strategy to play the Wordle game.

However, the purpose of this paper is not to find the optimal strategy of Wordle game. We are more concerned about which word attributes determine the difficulty of guessing a word, and model players' decisions in the game based on these attributes. Our core work is how to transform a given word into a numerical vector that can describe its characteristics. We construct some “natural” attributes of words from the structure, part of speech and the frequency of occurrence of the corpus. At the same time, we construct two attributes from the Wordle rule, which have strong explanatory power for the difficulty of guessing. We define them as “Gain”, that is, the average of reduction of uncertainty when a word is guessed. We have designed three algorithms to help us model the problems we need to solve.

The structure of this paper is as follows. Section 2 and Section 3 introduce the core assumptions of modeling and some symbols most used in the paper. Section 4 introduces our ideas and methods of constructing word attributes. Section 5 models the trend for the number of reported results and the percentage of scores reported that were played in Hard Mode. Section 6 discusses the modeling the distribution of daily results and how to predict the distribution of a given date and a given word. In Section 7, we define the word difficulty based on the average number of tries required to win the game and use our constructed attributes to do classification. Section 8 shows some interesting insights in the process of modeling. Section 9 summarizes the modeling process and discusses the possible improvement of the model.

# 2. Model Assumptions

1. Assume that the time trend we modeled will not change in the future.
2. Assume that the Wordle game contains a limited set of valid words, that is, the set of words allowed to be recognized by the game, and a set of all possible answers. Wordle randomly selects an unused word from the answer set every day as the target word of the day [2].

# 3. Notations

The following are some notations that are often used in this paper.

| Symbol | Description |
|---|---|
| $\mathcal{V}$ | The valid words set of Wordle game |
| $\mathcal{A}$ | The answer words set of Wordle game |
| $l_i$ | The $i$-th letter of the given word, in which $i=1,2,\cdots,5$ |
| $f(W)$ | The word frequency in valid words set |
| $f_{pos}$ | The letter frequency in valid words set aggregate by position |
| $f_{let}$ | The letter frequency in valid words set aggregate by letter |
| $gini(W)$ | The Gini index of word $W$ |
| $rc(W)$ | Indicator describes whether $W$ contains repeated and continuous letters |
| $g_i(W)$ | The Gain of word $W$ after $i$ rounds of guess |
| $\mathcal{F}$ | The feasible set of answer word |
| $N(t)$ | The number of reported results over time $t$ |
| $bp$ | The breakpoint of a given time series |
| $P(t)$ | The percentage of scores reported that were played in Hard Mode |
| $X_t$ | The vector of attributes of for the given word $W_t$ when the date is $t$ |
| $p_k(t)$ | The results distribution when the date is $t$ |
| $AP_k(t)$ | The average value of the distribution of the past few days |
| $M$ | The window size parameter to calculate $AP_k(t)$ |
| $kl$ | The KL divergence |
| $S_t$ | The static features used to predict results distribution when date is $t$ |
| $D_t$ | The dynamic features used to predict results distribution when date is |

# 4. Construct Word Features

To analyze the target word in the subsequent modeling process, we hope to convert the string of each word into a numerical vector (i.e., features) to represent the characteristics of the word, including the structure of the word, the difficulty of guessing the word, etc.

We put forward two ways to construct the features of words. On the one hand, we consider the "nature" of words, such as the composition, the frequency of occurrence in the corpus, which we call the words' natural attributes; On the other hand, starting from the Wordle game itself, we measure the uncertainty of the system from the perspective of entropy, and inspired by the information gain of the decision tree, we define the gain of Wordle word to describe how difficult it is to guess a given word, which is called words' Wordle attributes.

## 4.1 Words' Natural Attributes

### 4.1.1 Word Frequency

For a guessed word $W$, intuitively, if the word is used more frequently in people's daily life, the player will be most likely to associate these words after receiving certain feedback. According to the network, we found that the Wordle game contains nearly 13,000 legal words (we call it the valid words set $\mathcal{V}$), and only about 2,300 of them will be the answers to the Wordle game (we call it the answer words set $\mathcal{A}$) [2].

Based on Google Web Trillion Word Corpus [1], we counted the frequency of each word $f(W)$ in valid words set and normalized to make the sum of frequencies equal to 1.

$$\sum_{W\in\mathcal{V}} f(W)=1 \tag{4.1}$$

### 4.1.2 Letter Frequency

We now focus on each letter $l_i, i=1,\cdots,5$ that constitutes a word $W=(l_1,l_2,l_3,l_4,l_5)$. We will count the number of occurrences $c(i,l)$ of each letter in each position in valid words set $\mathcal{V}$, in which, $i=1,\cdots,5, l\in\{\mathrm{A},\mathrm{B},\cdots,\mathrm{Z}\}$, and then use two methods to normalize the frequency.

First, we fix the position $i$ and count the frequency of 26 letters (formula (4.2)). Second, we can fix the letter $l$ and count the frequency of each letter in 5 positions (formula (4.3)).

$$f_{pos}(i,l)=\frac{c(i,l)}{\sum_{l'\in\{\mathrm{A},\mathrm{B},\cdots,\mathrm{Z}\}} c(i,l')},\quad i=1,\cdots,5,\quad l\in\{\mathrm{A},\mathrm{B},\cdots,\mathrm{Z}\} \tag{4.2}$$

$$f_{let}(i,l)=\frac{c(i,l)}{\sum_{j=1,2,3,4,5} c(j,l)},\quad i=1,\cdots,5,\quad l\in\{\mathrm{A},\mathrm{B},\cdots,\mathrm{Z}\} \tag{4.3}$$

For a given word $W=(l_1,l_2,l_3,l_4,l_5)$, we can construct a letter frequency vector with 10 elements in it as some attributes of $W$ using letters $l_i$.

$$\left[f_{pos}(1,l_1)\quad f_{pos}(2,l_2)\quad\cdots\quad f_{pos}(5,l_5)\quad f_{let}(1,l_1)\quad f_{let}(2,l_2)\quad\cdots\quad f_{let}(5,l_5)\right]\in\mathbb{R}^{10}$$

### 4.1.3 Word Gini Index

For a discrete distribution $p=\{p_i\}_{i=1}^{K}$, we can define the Gini index of $p$ as:

$$gini(p)=1-\sum_{i=1}^{K}p_i^2. \tag{4.4}$$

Gini index is like entropy and can be used to indicate the dispersion of a probability distribution. If a probability distribution is more dispersed, the Gini index is larger. As shown in (4.5), we count the frequency of letters in $W$ and use it to calculate the Gini index $gini(W)$ of a word:

$$gini(W)=1-\sum_{l\in W}\left[p(l)\right]^2,\quad p(l)=\frac{1}{5}\sum_{i=1,2,3,4,5}\mathbb{I}(l_i=l). \tag{4.5}$$

For one of the most "complex" words, that is, the word does not contain duplicate letters, the Gini index $gini(W)$ takes the maximum value:

$$\max_W gini(W)=1-5\times\frac{1}{5^2}=0.8$$

For those words that contain repeated letters, the Gini index will be less than 0.8, for example $gini(\text{"APPLE"})=0.72$. The Gini index must be greater than 0, because no word containing five letters is all composed of the same letter.

### 4.1.4 Repeated and Consecutive Letters

Both words "APPLE" and "FEVER" have repeated letters ("P" and "E"), so they have the same Gini index $gini(W)$. However, the repetition is different. The two letters "P" in word "APPLE" appear continuously, and an attribute $rc(W)$ is defined to indicate whether the word $W$ contains repeated and continuous letters, as shown in (4.6).

$$rc(W)=\begin{cases}1, & \exists i,\ \text{s.t.}\quad l_i=l_{i+1},\\ 0, & \forall i,\quad l_i\neq l_{i+1}.\end{cases} \tag{4.6}$$

### 4.1.5 The Part of Speech

We assume that part of speech, as an attribute of a word, may also affect the difficulty of the Wordle game. We use 0-1 vector $ps(W)$ to encode the part of speech of the word $W$ [4].

## 4.2 Words' Wordle Features

Next, we construct the attribute of the word $W$ from the rules of the Wordle game. We hope this attribute can measure the difficulty of guessing the word. The basic idea is to use entropy to measure our current uncertainty.

For a given word $W$, assuming that there are total $N = |\mathcal{A}|$ answers and $M = |\mathcal{V}|$ valid guessing words in Wordle, the guess strategy for a player may follow as:

- At the beginning, we have no information about the guessed word $W$. The uncertainty of the system is the largest, and the entropy of the system is:

$$H = \log_2 N$$

- In the first round, for any guess $A_1$, players will receive feedback containing an array of five elements (each has 3 color states), which provides us with information to reduce the uncertainty of the system.
  - In short, the size of the feasible state space of the system will be reduced from $N$ to $N_1$. Notice that $N_1$ is related to guess $A_1$ and target word $W$.

$$N \to N_1(A_1, W), \quad N > N_1(A_1, W)$$

  - Inspired by the decision tree and we define the gain $g_1(A_1, W)$ brought by guess $A_1$ as the reduction of system uncertainty as shown in formula (4.7):

$$g_1(A_1, W) = \log_2 N - \log_2 N_1(A_1, W) \tag{4.7}$$

  - If we try each word in the valid words set $\mathcal{V}$ in turn, we can define the possible average gain $g_1(W)$ of guessing for the word $W$ in the first round. In formula (4.8), the weight $w_{A_1}$ represents the possibility that players choose $A_1$ from the valid words set $\mathcal{V}$, which can be estimated by word frequency.

$$g_1(W) = \sum_{A_1 \in \mathcal{V}} w_{A_1} \cdot g_1(A_1, W) \tag{4.8}$$

  - Notice that $g_1(W)$ various with word $W$. For a difficult target word $W$, the possible gain $g_1(W)$ from trying all guesses is also small, and more guesses are needed to obtain more information in the next few rounds. $g_1(W)$ can be used as an attribute to describe the difficulty of the word.
- The above process can be deduced to the next few rounds. We search all possible guesses in the subset in turn, and then calculate the average gain brought by these guesses. There, we can define such as $g_2(W), g_3(W), \cdots$, the subscript represents the round of guess. This can be easily achieved through a recursive algorithm. Notice that the computational complexity of the recursive algorithm increases exponentially with the number of rounds. The time required to calculate $g_3(W)$ of one word will exceed several hours.

Table 1 shows the recursive algorithm of calculating gain $g_i(W)$ for a given word $W$ and a required round $i$ (Algorithm 1).

**Table 1**: The recursive algorithm of calculating gain.

**Algorithm 1**: GainRecursive($W, i, \mathcal{F}$)

**Input:** target word $W$, search rounds $i \geq 1$, feasible word set $\mathcal{F}$, word frequency $\{f(W)\}_{W \in \mathcal{V}}$, valid words set $\mathcal{V}$ and answer word set $\mathcal{A}$

**Output:** Average gain of word $W$ after $i$ rounds of guessing

1 Initialize the set used to save the gain, $G \leftarrow []$ and $N \leftarrow \text{length}(\mathcal{F})$
2 Initialize the set used to save the weights, $w \leftarrow []$
3 **for** guess word $A$ **in** $\mathcal{F}$
4 Calculate the pattern $P$ obtained when the target is $W$ and the guess is $A$
5 $\mathcal{F}_1 \leftarrow \{W \in \mathcal{A} \mid \text{When the guess is } A, \text{ the received pattern is } P\}$, $N_1 \leftarrow \text{length}(\mathcal{F}_1)$
6 **if** $i > 1$ **and** $N_1 > 0$
7 $G$.add($\log_2 N - \log_2 N_1 + \text{GainRecursive}(W, i\text{-}1, \mathcal{F}_1)$)
8 **else**
9 $G$.add($\log_2 N - \log_2 N_1$)
10 **end if**
11 Add word frequency $w$.add($f(A)$)
12 **end for**
13 Normalize the weights $w$, $w \leftarrow w / \text{sum}(w)$
14 Calculated the average gain $g$, $g \leftarrow \text{sum}(w \cdot G)$
15 **return** $g$

We will supplement the implementation details of Algorithm 1. Because the calculation of gain requires a lot of search and recursion, it will take a lot of time. In the algorithm:

- given the target word $W$ and the player guess $A$, to calculate the pattern $P$ that the player will receive (line 4), and
- for a given guess $A$, find which answers can get the same pattern $P$ to from constructing a new feasible set $\mathcal{F}_1$ (line 5).

These two steps can greatly be accelerated by storing the pre-calculated results in a hash table. In this way, only $\mathcal{O}(1)$ time complexity is required for table lookup operation, without recalculation each time. Figure 1 shows the calculation principle of gain $g_i(W)$.

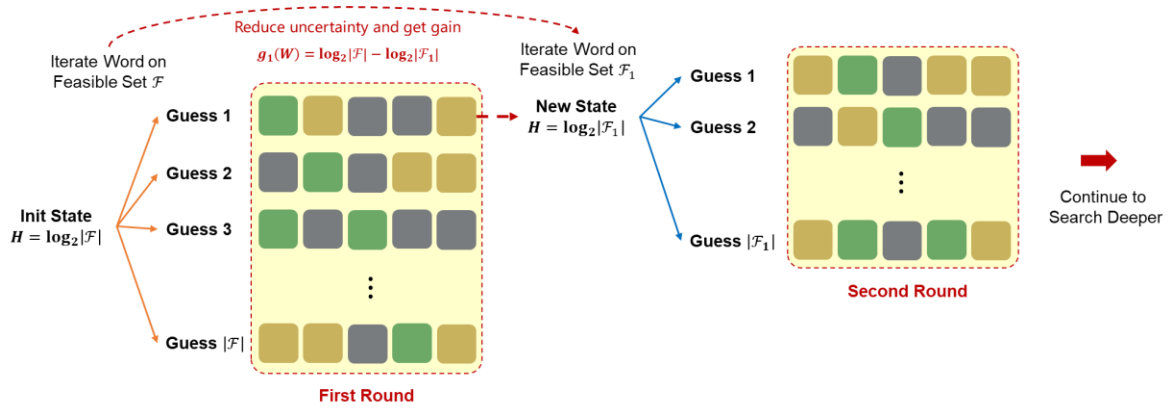


**Figure 1**: Recursive diagram of gain calculation.

# 5. Trend for The Number of Reported Results

In this section, we will discuss the modeling of the number of reported results and the percentage of scores reported that were played in Hard Mode, both can be regarded as time series analysis problems.

We introduce the breakpoint detection regression algorithm and decompose the number of reported results changing into four parts for modeling according to the change characteristics. In addition, we find that the attributes of the word in the previous few days will affect the percentage of Hard Mode scores in the future through an $\mathrm{AR}(1)$ process, which is intuitive. We will discuss the phenomenon in detail in section 4.2.

## 5.1 Modeling the Number of Reported Results with Breakpoint Regression

Visualize the change trend for the number of reported results $N(t)$ over time $t$, we find that the change of $N(t)$ presents different trends at different stages.

- At the beginning, Wordle received people's attention. $N(t)$ increases rapidly.
- After the $N(t)$ rose to the peak, the popularity of the game gradually decreased and $N(t)$ declined rapidly.
- Wordle games entered a stable attenuator in the second half of 2022, and $N(t)$ was still declining, but the decay rate was very slow.

We design a breakpoint detection regression algorithm (Algorithm 2) to split the time series into different parts, and then model each part separately. For a time series $\{N(t)\}$, the core idea of Algorithm 2 is to find a breakpoint $bp$ to divide the series into two sub-series $\{N_1(t)\}$ and $\{N_2(t)\}$, then we run OLS regression for two sub-series with respect to time index $t$. Assume that the square error of OLS on the two series is $SE_1$ and $SE_2$ respectively, the algorithm adjusts $bp$ to minimize $SE_1 + SE_2$ through dichotomy search.

Algorithm 2 is very efficient due to the use of dichotomy search. There is no need to iterate on every possible breakpoint and the time complexity of the algorithm is $\mathcal{O}(\log n)$, in which, $n$ is the length of the time series.

Figure 2 shows the two breakpoints obtained by using breakpoint detection regression. Next, we will start from 2022-01-07 and divide the time series into four parts for modeling. The assumptions and corresponding models of each part are as follows:

- Part I: (From 2022-01-07 to $bp_1$=2022-02-01) Assumption of fast rising stage, $N(t)$ can be expressed by the following linear model:

$$N(t) = \beta_1 \cdot t + \beta_0 + \varepsilon_1, \quad \varepsilon_1 \sim N(0, \sigma_1^2). \tag{5.1}$$

- Part II: (From $bp_1$=2022-02-01 to $bp_2$=2022-07-26) Assumption of fast exponential decay stage, $N(t)$ can be expressed by the following model. Assume the decay rate satisfies the following differential equation:

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = -c_1 \cdot \beta_2 \cdot e^{-\beta_2 \cdot t}.$$

Then $N(t)$ has the following expression:

$$N(t) = N_0 + c_1 e^{-\beta_2 \cdot t} + \varepsilon_2, \quad \varepsilon_2 \sim N(0, \sigma_2^2). \tag{5.2}$$

- Part III: (From $bp_2$=2022-07-26 to 2022-12-31) Assumption of slow exponential decay stage, $N(t)$ can be expressed by the following model. Assume the decay rate satisfies the following differential equation:

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = \beta_3 \cdot N(t) \cdot \left(1 - \frac{N(t)}{N_1}\right)$$

Then $N(t)$ has the following expression:

$$N(t) = \frac{N_1}{1 + e^{c_2 - \beta_3 \cdot t}} + \varepsilon_3, \quad \varepsilon_3 \sim N(0, \sigma_3^2). \tag{5.3}$$

- Part IV: (After a long time on 2022-12-31, **Not Observed**) Assume there is a stable number of reported results $\mu$ when time index $t$ is large enough. Then $N(t)$ satisfies:

$$N(t) = \mu + \varepsilon_4, \quad \varepsilon_4 \sim N(0, \sigma_4^2) \tag{5.4}$$

The Part IV is the assumption of the future change trend for $N(t)$. The model parameters of the Part I, II, III models are estimated by minimizing the square error.

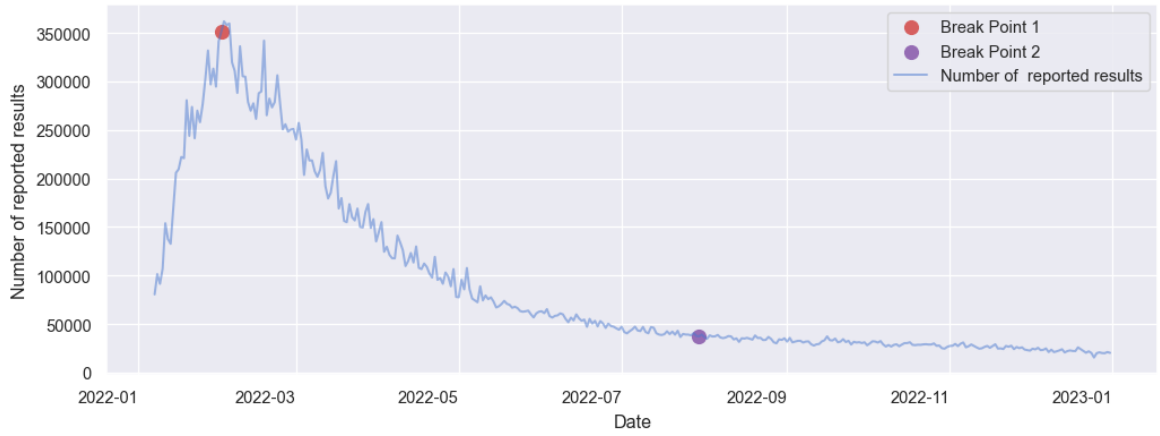


**Figure 2**: The two breakpoints found by the breakpoint detection regression algorithm. It can be seen in the figure that the three curves obtained by breakpoints have significantly different change trends.

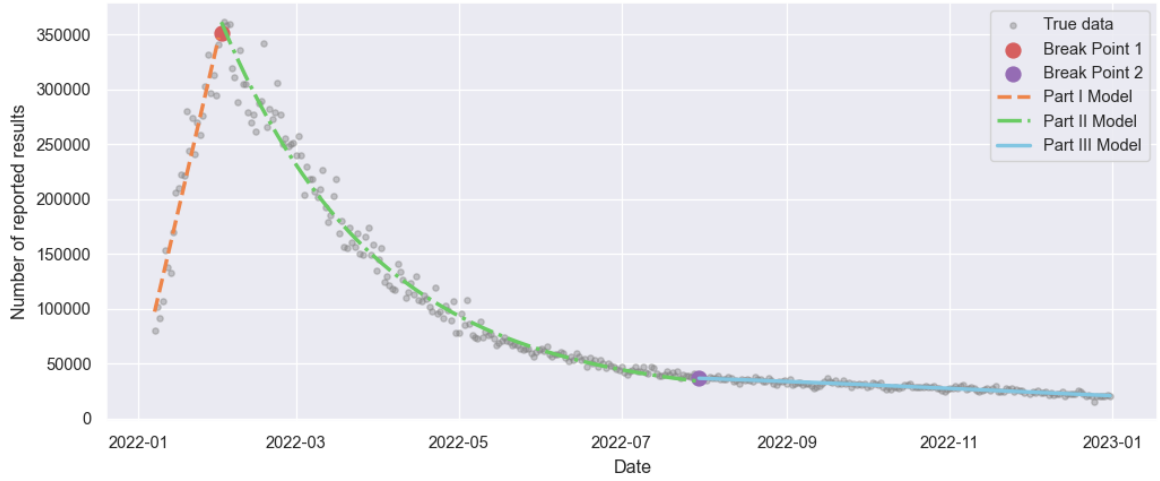


**Figure 3**: The fitting result of the three models (Part I, II, III) to the data.

Figure 3 shows the estimated model and the real data. The curve fits the trend of data change well. Test the residuals $r(t) = N(t) - \hat{N}(t)$. The residuals $r(t)$ can pass the normality test at the significance level of 10%, but from the ACF plot of the residuals, the 1-Lag tail can be observed. Therefore, the residuals series do not conform to the independence assumption, which can be improved by establishing an $\mathrm{AR}(p)$ model for the residuals $r(t)$. For the sake of simplicity, we will continue the main clue of modeling and do not discuss it in detail here.

As for the prediction of future $N(t)$, we assume that the change trend of $N(t)$ will continue to maintain the assumption of Part III model (formula (5.3)) until March 1, 2023. According to the estimated model parameters, we can obtain the point estimate of $\hat{N}(t_0)$:

$$\hat{N}(t_0) = \frac{\hat{N}_1}{1 + e^{\hat{c}_2 - \hat{\beta}_3 \cdot t_0}}. \tag{5.5}$$

Then we use the historical data to calculate the realized residual $r(t)$. Next, we can estimate the standard deviation $\hat{\sigma}_3$ of the residual. Finally, for the given confidence level $\alpha$, We can construct confidence interval from point estimation $\hat{N}(t_0)$ as shown in formula (5.6):

$$\left[\hat{N}(t_0)-\Phi^{-1}(0.5+\alpha/2)\cdot\hat{\sigma}_3 \quad \hat{N}(t_0)+\Phi^{-1}(0.5+\alpha/2)\cdot\hat{\sigma}_3\right], \tag{5.6}$$

In which, $\Phi$ is the CDF of standard normal distribution.

After calculation, the model predicts that the number of reported results of March 1, 2023, is $N(t)=14614$, and the prediction interval with 95% confidence level is $[11133 \quad 18095]$.

**Table 2**: The breakpoint detection regression algorithm.

**Algorithm 2**: BreakpointDetection($A$, *iters*)

**Input:** time series array $A$, the number of iterations *iters*

**Output:** the best breakpoint $bp$

Initialize the search range, $l \leftarrow 1$, $r \leftarrow A.\text{size}()$. Initialize $bp \leftarrow \text{int}\left((l+r)/2\right)$
Initialize the set used to save the square error, $SE \leftarrow []$
**while** $iters > 0$
    Split the array into two parts, $L \leftarrow A[l:bp]$, $R \leftarrow A[bp:r]$
    Run OLS for $L$ and $R$ respectively to obtain the square error $SE_1$, $SE_2$
    **if** $SE_1 > SE_2$
        $bp \leftarrow \text{int}\left((l+bp)/2\right)+1$
    **else**
        $bp \leftarrow \text{int}\left((bp+r)/2\right)+1$
    **end if**
    Save square error, $SE[bp] \leftarrow SE_1 + SE_2$, update iteration index $i \leftarrow i-1$
**end while**
**return** $bp \leftarrow \arg\min\{SE\}$

## 5.2 Modeling the Percentage of Hard Mode

We use $X_t \in \mathbb{R}^p$ represents the word attributes of the word $W_t$ when the date is $t$, and we use $P(t)$ to describe the percentage of scores reported that were played in Hard Mode. Notice that $P(t)$ has nothing to do with $X_t$ because $X_t$ is an "afterthought attributes". It is impossible for players to see the word $W_t$ to guess today in advance (otherwise, they can obtain information $X_t$ about it), and then decide whether to play the Hard Mode.

In fact, our assumption is that at date $t$, the word attributes for the past few days, i.e. $X_{t-1}, X_{t-2}, \cdots$ will have an impact on $P(t)$. The intuition of this conjecture is that:

- If the words in the past few days are easy to guess, more people will choose the Hard Mode to challenge themselves the next day.
- If the words in the past few days are difficult, most people will choose to give up the Hard Mode in the next day.

Next, we will verify the correctness of the above assumption through time series analysis.

First, it is observed that $P(t)$ has a trend of growth over time. The growth rate gradually slows down and finally approaches to a steady state. We assume that $P(t)$ meets the model,

$$P(t) = \mu(t) + \varepsilon_t, \tag{5.7}$$

and use the exponential model to model the trend term $\mu(t)$ as follows:

$$\mu(t) = \mu_0 - a \cdot e^{-\beta \cdot t}. \tag{5.8}$$

The above model has the following two intuitive realistic explanations:

- The number of players who join in the fun gradually decreases.
- The percentage of hardcore players among the remaining players continues to rise, and finally reaches a stable state.

Figure 4 shows the fitted result of trend term $\mu(t)$. The next step, we move to check the residuals time series $\hat{\varepsilon}(t) = P(t) - \hat{P}(t)$. Figure 5 shows the histogram, time varying characteristic, ACF and PACF plot of residuals. After checking we notice that:

- After normality test, $\hat{\varepsilon}(t)$ does not obey the normal distribution.
- After ADF test, $\hat{\varepsilon}(t)$ is not a stationary time series.
- In the ACF, $\hat{\varepsilon}(t)$ has a serious lag, $\hat{\varepsilon}(t)$ is autocorrelated.

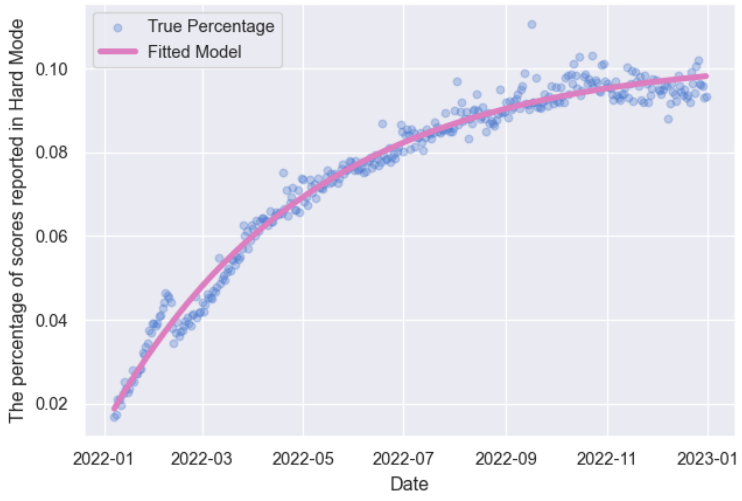


**Figure 4:** The fitted trend term model for the percentage of Hard Mode.

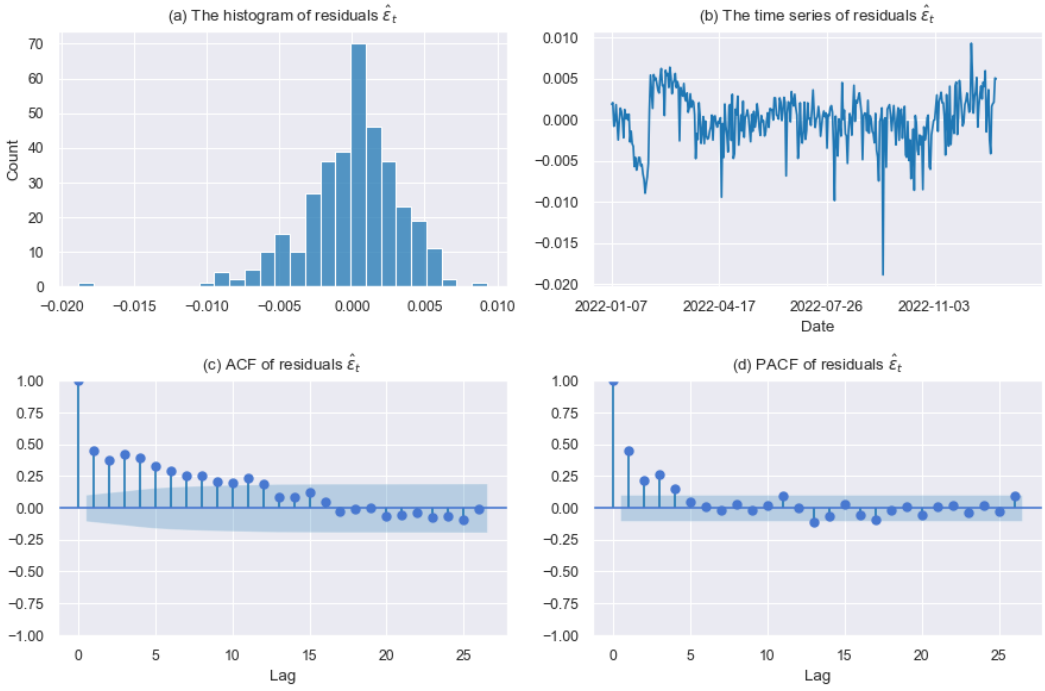


**Figure 5:** The (a) histogram, (b) time series, (c) ACF and (d) PACF plot of residuals. It can be clearly observed that the residual time series is not stationary, and there is a serious lag in the ACF.

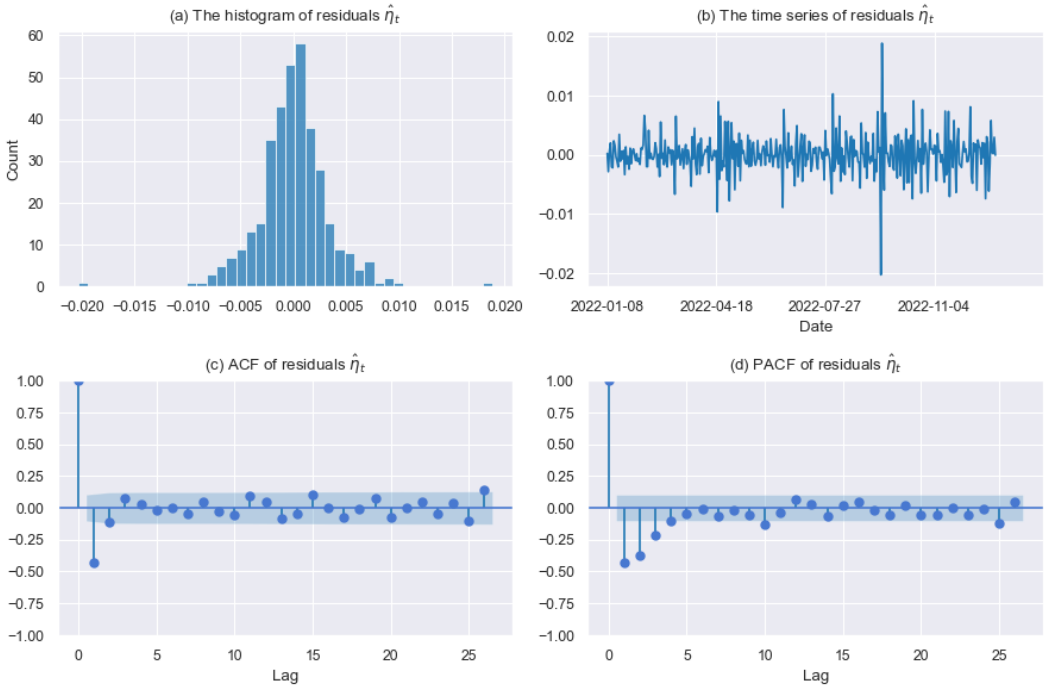


**Figure 6**: The (a) histogram, (b) time series, (c) ACF and (d) PACF plot of residual changes. At this time, the time series becomes normal and is stationary, we can see an obvious AR(1) characteristics in ACF.

We carry out first-order difference for residual $\hat{\varepsilon}(t)$ and define sequence $\eta_t$ and $\hat{\eta}_t$:

$$\eta_t = \varepsilon_t - \varepsilon_{t-1}, \quad \hat{\eta}_t = \hat{\varepsilon}_t - \hat{\varepsilon}_{t-1}, \tag{5.9}$$

As shown in Figure 6, the same test is performed on $\hat{\eta}_t$ again. The sequence after difference can pass the normality test and stationarity test. It is observed from ACF that $\hat{\eta}_t$ still has first order autocorrelation.

We assume that the random variable $\eta_t$ can be explained by two parts of information, one is about the auto-regression of $\eta_{t-1}$ (i.e. an $\mathrm{AR}(1)$ process), and the other is the word attribute of the previous day $X_{t-1}$. We build the model as shown in formula (5.10):

$$\eta_t \sim \alpha \cdot \eta_{t-1} + \beta^T X_{t-1}, \tag{5.10}$$

notice that we use the historical word attributes $X_{t-1}$, so there is no lookahead bias in (5.10).

Now we run an OLS to fit residual changes $\{\hat{\eta}_t\}$. Filter out most of the insignificant word attributes, and the final fitted model statistics are shown in Table 3. From the results, $\eta_t$ is most relevant to the information of its 1-order lag $\eta_{t-1}$. Only Gini index $Gini(W)$ is significant (level 5%) in the word attributes. The significance of $rc(W)$ (whether there are repeated and continuous letters in $W$) and $g_2(W)$ (2-round gain) is close to 10%. The other attributes have no explanatory power. The whole model adjusted R-squared is 0.219. According to the F-statistic, the regression model is significant.

Attributes (especially $Gini(W)$) do have an impact on $\eta_t$, combine (5.9) and (5.10), we have:

$$\begin{aligned} \varepsilon_t &= \varepsilon_{t-1} + \alpha \cdot \eta_{t-1} + \beta^T X_{t-1} \\ &= \varepsilon_{t-1} + \alpha \cdot \left(\alpha \cdot \eta_{t-2} + \beta^T X_{t-2}\right) + \beta^T X_{t-1} \\ &= \cdots \\ &= \varepsilon_{t-1} + \beta^T \sum_{k=1}^{K} \alpha^{k-1} X_{t-k} + \alpha^K \eta_{t-K} \end{aligned} \tag{5.11}$$

We substitute the above model into the expression (5.7), and the final model is:

$$P(t) = \mu(t) + \varepsilon_{t-1} + \beta^T \sum_{k=1}^{K} \alpha^{k-1} X_{t-k} + \alpha^K \eta_{t-K}, \tag{5.12}$$

in which, $|\alpha| < 1$. Therefore, over time, the impact of the target word attributes $X_{t-k}$ in the previous few days on the current percentage $P(t)$ decreases exponentially. In our model, $|\alpha|$ is about 0.354, so the influence of the word attributes five days ago $X_{t-5}$ on the current $P(t)$ is only close to 1%.

**Table 3:** The OLS results of residual changes fitting.

| **OLS** | Adj. R-squared: | 0.219 | P (F-statistic): | 3e-18 *** |
|---|---|---|---|---|
| **Features** | | Coef | std,err | p-value |
| **Lag Term** | $\eta_{t-1}$ | -0.3540 | 0.050 | 0.000 *** |
| **Word Attributes** | $Gini(W)$ | -0.1229 | 0.062 | 0.049 ** |
| | $rc(W)$ | -0.2945 | 0.185 | 0.113 |
| | $g_2(W)$ | -0.1744 | 0.115 | 0.131 |

To sum up, from (5.12), we can see that the word attributes of the past few days are superimposed through an $\mathrm{AR}(1)$ process, thus affecting the Hard Mode percentage $P(t)$ at the time of date $t$, which will decay exponentially in several days. The above discussion verifies our conjecture at the beginning of this section. At the same time, we get from the regression model (5.10) that the Gini index has the greatest impact on the $P(t)$ of all defined word attributes.

# 6. Model the Results Distribution

In this section, we will discuss the Wordle results distribution modeling. There are 7 possible states for each day's results (1, 2, 3, 4, 5, 6 and X). Figure 7 shows the distribution various with time. We use a linear function to model the contribution of the features to the labels, and then use the softmax function to convert the output into a probability distribution. Finally, the model parameters are obtained by maximum likelihood estimation (MLE).

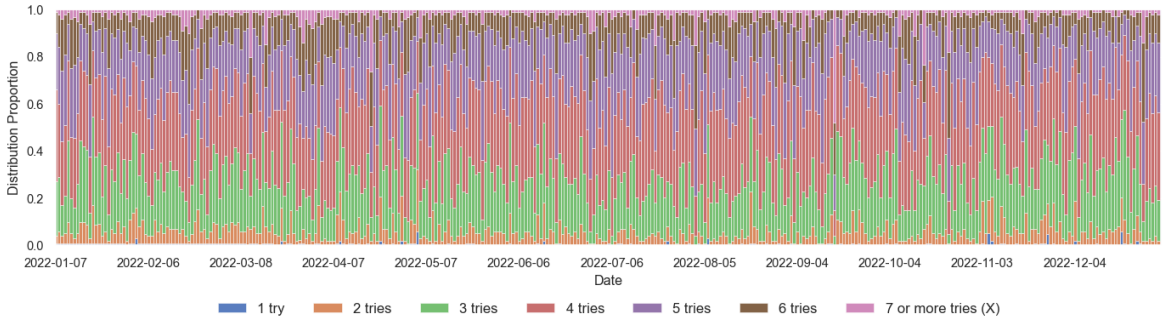


**Figure 7:** The distribution of daily results changes over time. This change may be related to the difficulty of Wordle words and the level of players.

## 6.1 Modeling Distribution Using Static and Dynamic Features

For each possible state $k=1,2,\cdots,7$, each word $W_t$ and time index $t$, assume the distribution we want wo predict is $p_k(t)$:

$$\sum_k p_k(t)=1, \quad \forall t, \tag{6.1}$$

we define the logits $L_k(W_t,t)$, i.e., the contribution of the features to the distribution:

$$L_k(W_t,t)=(\beta_S^{(k)})^T S_t+(\beta_D^{(k)})^T D_t. \tag{6.2}$$

in which, $\beta_S^{(k)},\beta_D^{(k)},k=1,2,\cdots,7$ is the corresponding coefficient of state $k$.

- $S_t \in \mathbb{R}^{p_1}$ is the static feature of the model. $S_t$ contains the word attribute of the word $W_t$, In other words, we have $S_t = X_t$. Every time the word $W_t$ is given, $S_t$ is determined. $S_t$ is only related to the word and has nothing to do with the player's level.
- $D_t \in \mathbb{R}^{p_2}$ is the dynamic feature of the model, which is related to players and date $t$. We design the following three features:
  - Current time index $t$ is used to capture the trend term in model.
  - Hard Mode percentage $P(t)$ discussed in Section 5.2. We assume that the more players in Hard Mode, the higher the overall level of the players. On average, players only need to try less guesses to win, so the distribution is skewed to the right.
  - In the past period (taking the window size parameter $M$ ), the player's average performance $AP_k(t)$ is calculated:

$$AP_k(t) = \frac{1}{M}\sum_{i=1}^{M} p_k(t-i), \quad k = 1,2,\cdots,7 \tag{6.3}$$

$AP_k(t)$ is the average value of the distribution of the past $M$ days. It can be used to measure the ability (game level) of players in the recent period.

Now, we can define our distribution model using softmax transform:

$$\hat{p}_k(t) = \frac{\exp L_k(W_t, t)}{\sum_{j=1}^{7} L_j(W_t, t)} \in (0,1), \quad k = 1,\cdots,7 \quad \text{and} \quad \sum_{k=1}^{7} \hat{p}_k(t) = 1, \tag{6.4}$$

we estimate the model parameters $\{\beta_S^{(k)}, \beta_D^{(k)}\}_{k=1}^{7}$ by minimizing the Cross Entropy loss $\ell$:

$$\ell = \sum_t \text{CrossEntropy}(p(t), \hat{p}(t)) = -\sum_t \left( \sum_{k=1}^{7} p_k(t) \cdot \log \hat{p}_k(t) \right), \tag{6.5}$$

and use KL divergence $kl > 0$ to measure the error between the predicted distribution and the real distribution. The smaller the value of $kl$, the more accurate the distribution prediction is.

$$kl = \sum_t \left( \sum_{k=1}^{7} P_k(t) \cdot \log \hat{P}_k(t) \right) - \sum_t \left( \sum_{k=1}^{7} P_k(t) \cdot \log P_k(t) \right) \tag{6.6}$$

## 6.2 Evaluation of Fitting Model

When fitting, we split the whole time series into training set and validation set. This kind of random split ensures that the training set is always in front, and the validation set is behind, so there will be no lookahead bias.

Figure 8 shows the model coefficient matrix estimated when $M = 5$. The horizontal axis represents the features (29 in total), and the vertical axis represents the components of the states (7 in total). The brighter the color of the part in the heatmap represents the stronger the interpretation ability of the feature to the distribution.

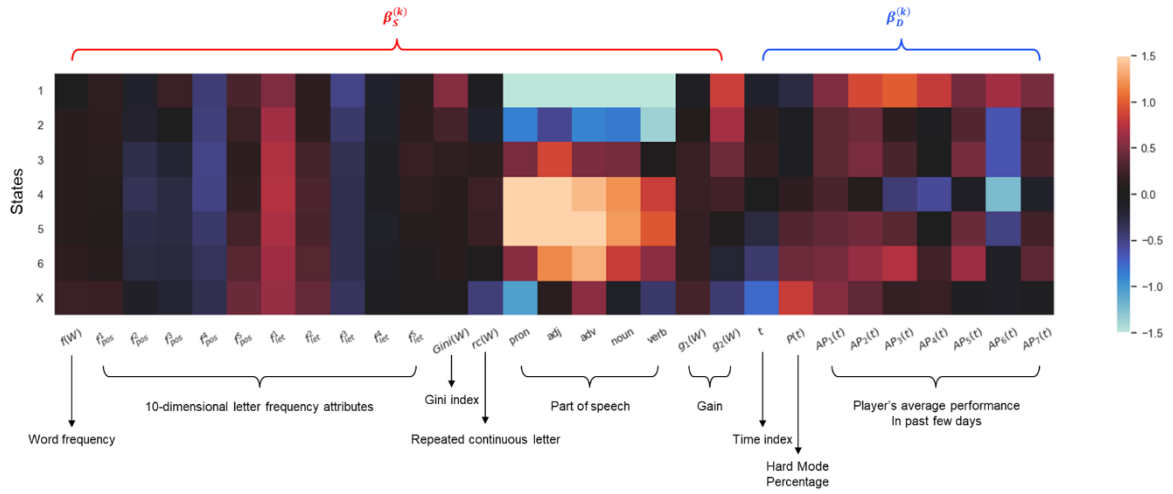


**Figure 8**: The heatmap of estimated model parameters. The brighter the color is, the more significant the coefficient is, and the more important the corresponding feature is for distribution prediction. The darker the color is, the closer the absolute value of the coefficient is to 0, and the less important the feature is.

We can also get some direct insights from Figure 8. For example, from the perspective of letter frequency, the frequency of the first letter is more important than the frequency of the letter at the end of the word. The word frequency itself does not play much role. The part of speech of words and the average performance of players in the past few days are very helpful for prediction.

We use cross validation to select the window size parameter $M$. Figure 9 shows the change trend of KL divergence $kl$ of training set and validation set with $M$. Considering bias-variance tradeoff, with the increase of $M$, the $kl$ of the training and validation set decreases, the model fitting ability increases. When $M$ continues to increase, the $kl$ of the training set continues to decrease, but that of the validation set increases. Currently, the generalization error increases, the model appears overfitting. We think $M = 7$ is a good choice.

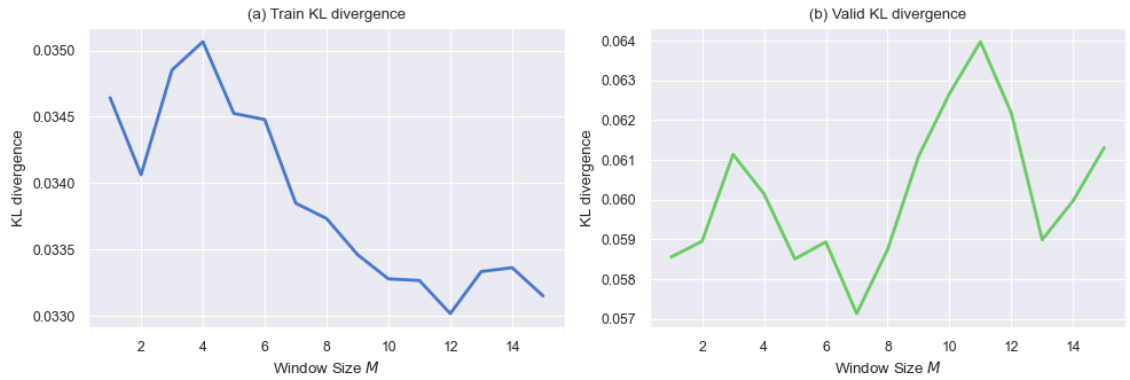


**Figure 9**: The KL divergence of training set and validation set with different window size parameter. When the length of the window is greater than 7, we can observe overfitting.

Figure 10 shows the comparison between the predicted results and the real distribution of the two samples in the training set and the validation set. The model can grasp the overall characteristics of the distribution and there is a small amount of error in each state space.

We consider the errors $e_k(t) = p_k(t) - \hat{p}_k(t)$ of distribution prediction. Figure 11 shows the boxplot of the residuals. The estimation of the model in the training set is accurate and the performance in the validation set is slightly biased, but the error distribution is concentrated and acceptable. The mean absolute error of $e_k(t)$ is 0.0313. The possible reason for this bias is that we do not model the time series trend of distribution changes in detail. We will discuss it in the model improvement (section.


**Figure 10**: Comparison of predicted distribution and real distribution.

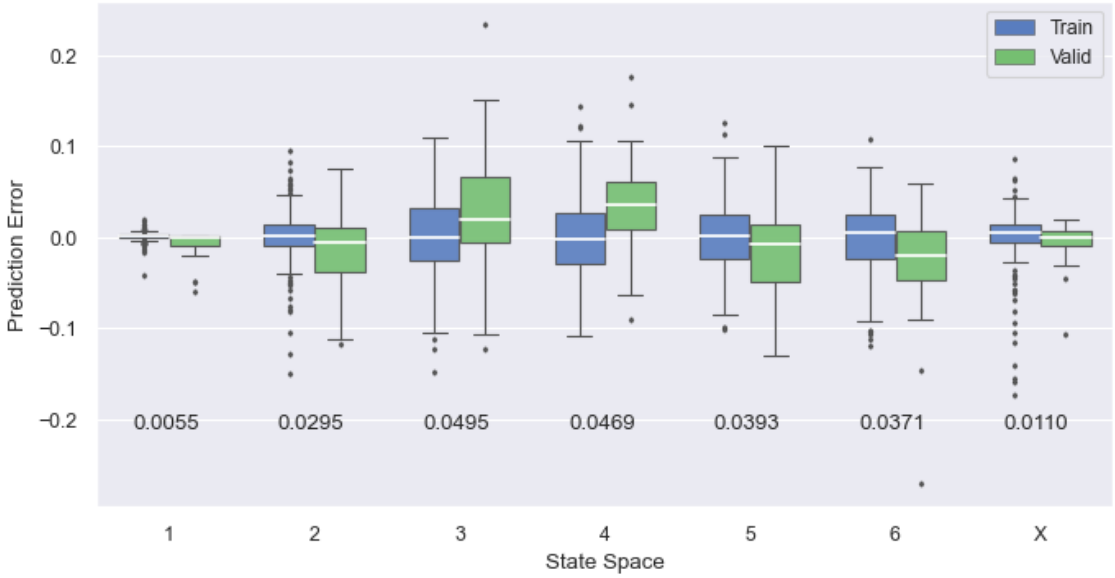


**Figure 11**: The boxplot of distribution prediction error. We can observe that the point estimation of the training set is accurate, and the point estimation of the validation set is slightly biased, but the error distribution is relatively concentrated. The validation set mean absolute error of each state is displayed.

## 6.3 Rolling Prediction for Future and Uncertainties

For a given date $t$ and a given target word $W_t$, we can construct the static features $S_t$ using $W$, and get a rough prediction for Hard Mode percentage $\hat{P}(t)$ using model (5.8). However, in order to predict the distribution at $t$, model (6.4) needs all the previous $M$ days' distribution information $\{p_k(t-i)\}_{k,i}, k=1,\cdots,7, i=1,\cdots,M$. This part belongs to future information and cannot be obtained at present. We can only fill in the distribution between the last day $t_0$ of the dataset and the target date $t$ step by step through rolling prediction method.

For rolling prediction procedure, we start from the last day $t_0$ of the dataset. Suppose we know the target words $\{W_{t_0+1}, W_{t_0+2}, \cdots, W_t\}$ to the Wordle game every day. Then, we can use the word $W_{t_0+1}$ to construct static features $S_{t_0+1}$, and use $\{p_k(t_0), \cdots, p_k(t_0-M+1)\}_{k=1}^7$ (the past $M$ days distribution) to construct dynamic features $D_{t_0+1}$ with predicted Hard Mode percentage $\hat{P}(t)$. Combined with $S_{t_0+1}$ and $D_{t_0+1}$, we can predict $\{p_k(t_0+1)\}_{k=1}^7$ using (6.4). Next, we update the time index to $t_0+1$, take the predicted $\{p_k(t_0+1)\}_{k=1}^7$ as the historical distribution, use the same process to predict $\{p_k(t_0+2)\}_{k=1}^7$, and repeat the above rolling prediction procedure until reach the target date $t$.

The above rolling prediction method includes two main uncertainties:

- (6.4) needs true history results distributions to build $D_{t_0+1}$. We use the predicted value to fill in the missing information about future distribution. This may cause uncontrollable error accumulation, to reduce the reliability of the results of long-term prediction.
- In practice, we don't know the target word list $\{W_{t_0+1}, W_{t_0+2}, \cdots, W_t\}$ for each day in the future. Different word list, or words in the same list appear in different order, will make the final prediction result different.

Therefore, for the given date $t$ and word $W_t$, we have to assume the target word list $\{W_{t_0+1}, W_{t_0+2}, \cdots, W_t\}$ between $t_0$ and $t$. We sample from the unused answer set $\mathcal{A}$ of Wordle game to construct the list. The predicted distribution of the date $t$ obtained from each sampling are different. Then we repeat the above process $B$ times through the bootstrap method, and finally take the mean value as the point estimate of the distribution. We can also construct confidence intervals from the prediction results of the $B$ bootstrap predictions to obtain the knowledge about the reliability of the results.

We take bootstrap times $B=500$, and Figure 12 shows the prediction for the word "EERIE" on March 1, 2023. From the result, "EERIE" is a difficult word to guess. Only about 5% of players can win the game within three tries, and more than 40% of players need five attempts to win the game. Figure 12(b) shows the bootstrap standard deviation, which describes the confidence level of the model for each state prediction. The maximum standard deviation is 0.007, which indicates that the rolling prediction method is stable, the prediction result does not fluctuate too much. The point estimation of distribution prediction and its 95% confidence interval prediction (as a subscript) is as follows:

$$\left[1\mathrm{e}-6_{\pm 6\mathrm{e}-7} \quad 1\mathrm{e}-3_{\pm 7\mathrm{e}-5} \quad 0.050_{\pm 0.002} \quad 0.301_{\pm 0.015} \quad 0.434_{\pm 0.005} \quad 0.196_{\pm 0.014} \quad 0.018_{\pm 0.005}\right]$$

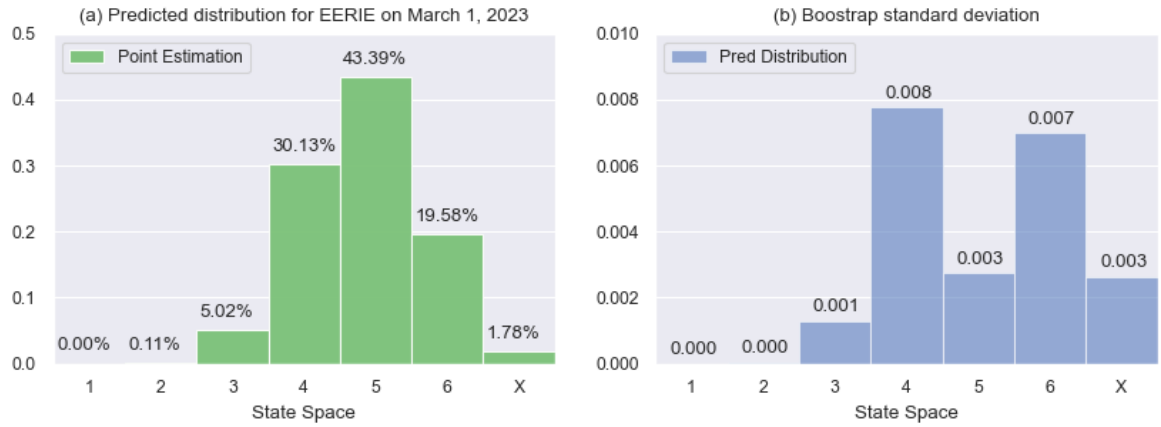


**Figure 12**: The predicted distribution for the word "EERIE" on March 1, 2023. From the (b), the rolling prediction results are stable, without too large numerical fluctuations, so the prediction results are reliable.

# 7. The Difficulty of Wordle Puzzle

In this section we discuss the classification of solution words by difficulty and identify how the attribute of a word contributes to the classification. We extract a measure of difficulty for the words based on the daily players' average performance, and then turn the problem into a multi-classification task by assigning difficulty labels to each word.

We discuss the influence of different assignment strategies on the accuracy of the model and use the best one to classify the difficulty level of the given word. Also, in Section7.1, we will evaluate and get an overall view of the importance level of each word attribute to the classification model.

## 7.1 Multi-Classification Model

The difficulty of the solution word has an implicit relationship, which we're going to unveil later in this section, with attributes constructed in former sections. However, based on the players' performance (the results distribution) every day, we can use the average number of tries taken to win as a metric of the difficulty for a given word. The value of the word difficulty indicator can be expressed as:

$$D(t) = \sum_{k=1}^{7} k \cdot p_k(t), \tag{7.1}$$

in which, $p_k(t) \in (0,1)$ is the distribution of state $k$ at date $t$ defined in (6.1). $k$ represent the number of tries. Noticed that, for those who takes more than 6 tries or not be able to solve the Wordle puzzle, we use 7 to represent their number of tries.

In this way, the difficulty of the word can be represented by a float number $D(t)$. Intuitively, the smaller $D(t)$ is, the less difficult the word is. The larger $D(t)$ is, the more difficult it is to guess. Players need more rounds of tries to get enough information for judgment. To make the results of the classification easier to understand and use, we assign difficulty level Easy, Normal and Hard to each word according to indicator $D(t)$ we constructed above.

For two given quantiles $q_1, q_2, q_1 < q_2$ of $D(t)$, we define the difficult level $Y(t)$ as shown in (7.2):

$$Y(t) = \begin{cases} \text{Easy}, & D(t) \le q_1, \\ \text{Normal}, & q_1 < D(t) \le q_2, \\ \text{Hard}, & D(t) > q_2. \end{cases} \tag{7.2}$$

We use Random Forest to build our classification model. However, the classification model's accuracy is related to how we split the range of $D(t)$, i.e., the choice of quantiles $q_1, q_2$. In extreme cases, if $q_1$ is taken as 1% quantile and $q_2$ is taken as 99% quantile, the model which always predicts Normal will reach accuracy 98%. This unreasonable split will not only bring the illusion of false high accuracy, mislead model selection, but also have no meaning in practical application.

Therefore, we jointly consider the distribution of $D(t)$, the actual application meaning of segmentation results and the classification performance of the model, determine the optimal quantiles $q_1^*, q_2^*$ using validation set through grid search. The result shows that $q_1^*$ is taken as 10% quantile and $q_2^*$ is taken as 70% quantile to assign difficulty level of words. The actual meaning of the above scheme is that in our average number of tries distribution, the first 10% of words are defined as Easy, and the last 30% are defined as Hard. The ratio of Easy, Normal and Hard is 1:6:3.

The parameters of the Random Forest are adjusted through cross validation. Figure 13 shows the confusion matrix of the fitted model in the training set and the validation set. From the confusion matrix, we can find that our model displays a good performance of identifying the difficulty level. It won't classify Easy words as Hard ones, nor will it classify the Hard words as Easy ones. The number of misclassifications is relatively quite small as well.

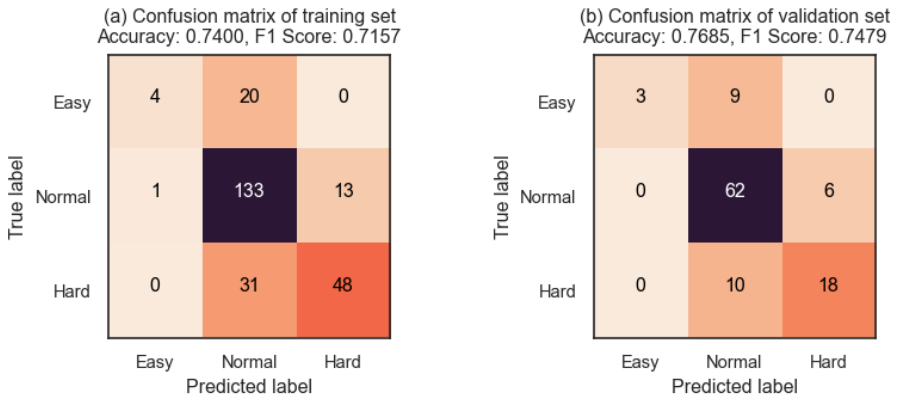


**Figure 13**: Confusion matrix of random forest model in training set and validation set. The model is easy to confuse Easy, Hard with Normal, but it is good for distinguishing Easy and Hard.

From the values of feature importance in the model, we find that word attributes that are most associated with the classification are Gain $g_1(W), g_2(W)$. We show their relationship with the average number of tries $D(t)$ in Figure 16 in Section 8.

To get a more explicit illustration of the difference among the influences of the word attributes to the classification, we introduce Shapley value [5], which is created to help interpretate the outputs of machine learning algorithms. We take the average of the Shapley value for each feature as the contribution explaining the model in classification task. Figure 14 gives the top 10 most important attributes and their Shapley value when predicting each class.

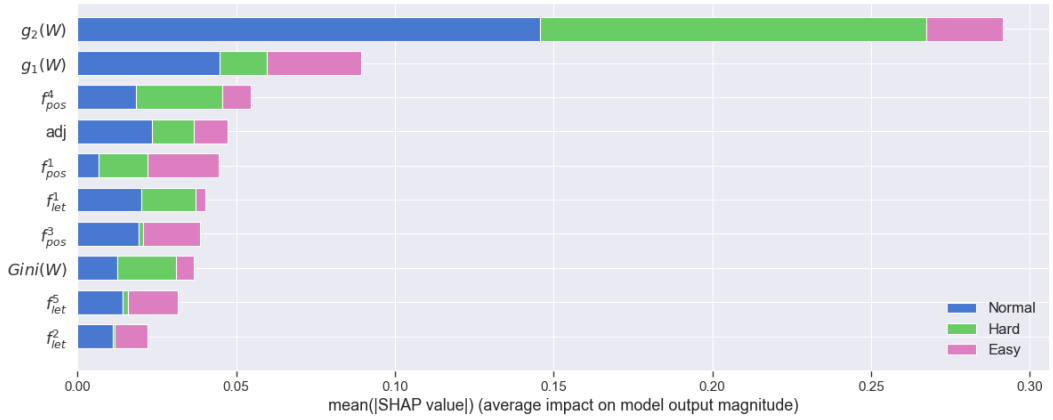


**Figure 14:** The average contribution of the top 10 important features in the model prediction of each class.

Based on the model above; we can now make classification for word "EERIE". The classification of the difficulty level for it is **Normal**. Figure 15 shows the contribution of the attributes when classifying "EERIE" for each class through Shapley value.

Both Gain $g_1(W), g_2(W)$ and letter frequency $f^1_{pos}, f^5_{let}$ play a role in determining whether to classify word into Easy. When judging the Normal class, model almost rely on Gain $g_1(W), g_2(W)$. When judging the Hard class, $g_2(W)$ and $f^4_{pos}$ contribute the most.


**Figure 15:** Shapley value of word attributes when predicting "EERIE" for each class.

# 8. Interesting Insights of Dataset

Inspired by [3], we counted the frequency of letters as the first letter of a word in the valid words set $\mathcal{V}$, and counted the distribution of the letter in five positions, shown in Figure X.

- About 30% of all valid words begin with the letters S, E and A, while almost no words begin with the letters X and Q.
- Compared with the average number of guesses of words, it is also found that the higher the frequency of the first letter, the less the average number of guesses required.

Figure 16 contains a large amount of deep information about the word structure that can be mined, which will affect people's decision-making when playing Wordle.


**Figure 16**: The frequency of letters as the first letter of a word in the valid words set and their distribution.

We have not intuitively presented the relationship between the difficulty of words and the constructed gain attribute $g_1(W), g_2(W)$. Figure 17 shows the strong interpretation ability of feature $g_1(W), g_2(W)$ to the average number of tries.


**Figure 17**: The relationship between the difficulty of words and the constructed gain attributes.

# 9. Conclusion and Improvement

We construct word attributes from the perspective of word structure, part of speech, language characteristics and Wordle game rules, to convert a given word into a numerical vector. We combined time series analysis and breakpoint detection regression to model the change trend of the number of reported results. When modeling the time series of the percentage of Hard Mode, we found that the past few days' word attributes will affect the percentage in the next period through an $AR(1)$ process, and we give the insights and explanation for it.

We use the multinomial distribution to model the distribution of the results and solve the model parameters by minimizing the cross-entropy loss function. a Using rolling prediction and Bootstrap technique, we can predict the distribution of a given date and a given word.

Finally, we use the average number of tries required to win the game to define the difficulty of the word, and divide it into easy, normal, and hard according to its quantiles. The word gain attributes $g_1(W), g_2(W)$ constructed has strong prediction ability for difficulty. We use the Shapley value to identify how the attribute of a word contributes to the classification task.

Several aspects of the modeling can be improved, and more reasonable results could be obtained:

- The residual in the modeling of the number of reports is not carefully handled. The residual may have autocorrelation. It can be modeled using AR or more complex models (such as ARIMA) to make the trend model more accurate.
- The results distribution model (6.4) does not consider the significance of the estimated coefficients, so we ignore the features selection step. In fact, it is necessary to filter out effective features before modeling. The significance of coefficients can be calculated through Bootstrap technique.
- We do not use time series analysis technology for distribution (6.4). By introducing more time series features into the model or considering time series modeling based on the original one, we could improve the model performance and correct the bias.

# 10. References

[1] Rachael Tatman. English Word Frequency in Google Web Trillion Word Corpus. Kaggle, February 18, 2023, https://www.kaggle.com/datasets/rtatman/english-word-frequency

[2] Owen Yin. Here Lies Wordle: Full Answer List (NYT Update). Medium, February 18, 2023, https://medium.com/@owenyin/here-lies-wordle-2021-2027-full-answer-list-52017ee99e86

[3] @neilrakye. Frequency of letters in 5 letter English words and where they occur in the word. February 18, 2023, http://www.gwicks.net/dictionaries.htm

[4] 1eez lordly, abcdea. Word part of speech database in 1eez / 103976. Github, February 1, 2023, https://github.com/1eez/103976

[5] Lundberg, Scott M., and Su-In Lee. "A unified approach to interpreting model predictions." Advances in Neural Information Processing Systems (2017)

# 11. Letter to the Puzzle Editor

Dear Puzzle Editor of the New York Times,

We are the TEAM#2318804 from MCM. Through the research on the 2022 Wordle player's performance data and word attributes, we have some interesting findings.

1. After the popularity of Wordle, the number of visitors declined rapidly. So far. Although the number of players per day still has a downward trend, it is close to a stable state. We expect that the number of players will fluctuate at a constant value in the future. The model predicts that the number of reported results on March 1, 2023, will be 14614 and the prediction interval will be [11133, 18095].
2. Based on the analysis on the change of percentage of players in Hard Mode, we found that this percentage will be affected by the attributes of words in the previous few days. When going over a specific interval of time, the impact will decrease exponentially. For example, we found that the attributes of words 5 days ago will only have influences of 1% on the current percentage of Hard Mode players compared to the attributes of words yesterday.
3. Among the word attributes we created, the one that measures the complexity of the word has the greatest impact on the percentage of players in Hard Mode.
4. When modeling to get insights of the distribution of the reported results, we found that the frequency of the first letter is much more important than the one at the end of the word. Besides, the part of speech of the word and the average performance of players in previous days exhibit greatest influence on the prediction of the distribution.
5. According to the word attributes and the performance data we have, we can also make further analysis like classify the solution words by difficulty to have an estimation of the upcoming word, which could of some help to monitor the influence of change of difficulty level on the players' performance and then to design interesting new words and playing modes.

Sincerely,

Team#2318804