Problem Chosen	2023 MCM Summary Sheet	Team Control Number
C		2311888

Wordle Wars: The Battle to Understand Difficulty in the World's Favorite Word Game!

Abstract

Wordle is a popular word-guessing game, and understanding the factors that influence its difficulty is essential for improving its design, user experience, popularity, and entertainment value. To achieve this, this study employs data mining methods based on machine learning and linguistics to explore these factors.

In Model I, We used **XGBoost** with **RandomSearchCV** to optimize model training and testing for prediction tasks. **To address problem one**, we selected a stable data period after the 352nd round of the game for training and testing. We used the **Bootstrapping** method to estimate the **confidence interval** and obtained a 95%confidence interval of $(18289.66, 19917.65)$ for the predicted results, which ensures the credibility of our predictions. **For problem two**, word attributes from model two were used to predict answer attempt distribution.Here is the predicted distribution:$(1.89, 7.13, 25.01, 31.12, 24.09, 14.71, 5.26)$. Furthermore, We compared the performance of XGBoost with LSTM and MLP [1, 2] and found that XGBoost outperformed the other two models.

In Model II, We developed a model for **extracting word properties**, which includes seven features related to word structure and pronunciation, such as consonant clusters, vowel clusters, consonant-vowel-consonant structure, repeated letters, diacritic marks in the phonetic symbols, and structurally similar words. The similarity between words was defined using the **Jaro-Winkler similarity** measure, which was used to extract the feature of structurally similar words.

In Model III, We constructed three different functions using the Gaussian probability density function as a weight factor based on the reported distribution. We then weighted the word's attribute features based on the percentile of the word's difficulty within the corresponding word class and obtained the **differential distinguishability** of the attribute features for each word class: $VC_{\text{linear}} = 6.1798$, $VC_{\text{logarithmic}} = 6.05$, $VC_{\text{exponential}} = 5.45$, and found that the **linear function** performs the best. For the specific word "EERIE", we calculated its differential distinguishability of attribute features for different difficulty word classes under the linear function model, and obtained a difficulty evaluation of "Hard" with specific values of $(4.42, 4.62, 4.75, 4.34, 3.99)$.

In Problem IV,We explored three additional aspects of the dataset. Firstly, player attempts follow an approximately normal distribution, which can be controlled by extracting word features. Secondly, word properties are related to difficulty, the higher frequency and the simpler phonetic structure associated with the lower difficulty. Finally, player scores show a downward trend based on the weekly day, possibly due to differences in work and leisure patterns between weekdays and weekends.

Keywords: XGBoost, Bootstrapping, Feature Extraction, Jaro-Winkler similarity

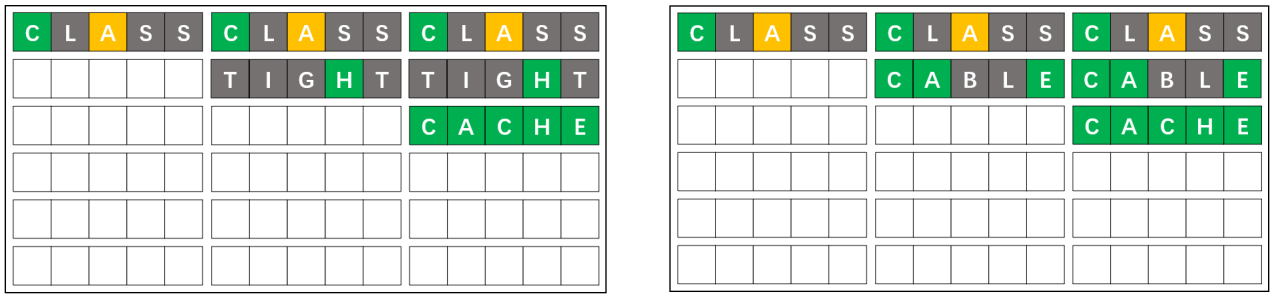
Contents

1 Introduction

1.1 Background

Word guessing games are a linguistic phenomenon, representing the use of language in different social and cultural contexts. Such games can help improve vocabulary and general language training. Wordle is a fun word guessing game developed by Josh Wardle that has gained tremendous popularity due to its simple gameplay and convenient access. Since February 1, 2022, Wordle has been acquired and maintained by The New York Times for daily operation.

The player's goal is to guess a five-letter English word within six or fewer attempts, and each guess must be a real English word recognized by the game's officials. After each guess, the colors of the tiles will change in three ways: gray, yellow, and green. Gray indicates that the mystery word does not contain the guessed letter, yellow indicates that the mystery word contains the guessed letter but in the wrong position, and green indicates that the mystery word contains the guessed letter in the correct position. Figure (a) shows an example solution that found the correct result on the third attempt.



(a) Regular mode (b) Hard mode

Figure 1: Wordle game schematic diagram

The game consists of two modes, and the above rules apply to the regular mode. The other mode is the hard mode, which requires the player to find a correct letter in a word (the color of the tile is yellow or green), and these letters must appear in the next word guessed by the player. Figure (b) is an example solution for the hard mode, where the second guessed word must include both C and A, in contrast to Figure 1.

1.2 Our works

In this paper, to predict and evaluate the word report count and distribution, as well as the difficulty level and classification of the word "EERIE" in the Wordle game, we established the mathematical model shown in **Figure 2** and completed the following works:

Work 1: Predicting the range of the result report numbers on March 1,2023;

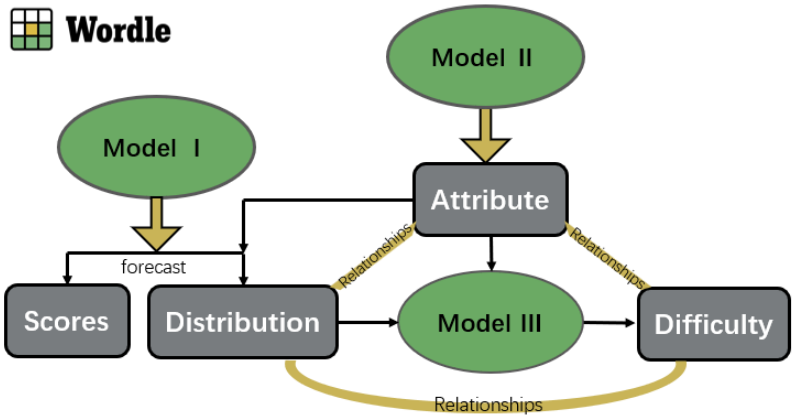


Figure 2: Overview of Mathematical Models

Work 2: Extracting the word attributes that can affect the game difficulty;

Work 3: Predicting the distribution of answer times using the extracted word attributes;

Work 4: Evaluating the difficulty level of words, classifying the difficulty, and evaluating the difficulty of "EERIE";

Work 5: Extracting the features of the dataset to explore the relationships among word attributes, time, answer times distribution, and difficulty.

2 Model Preparation

2.1 General Assumptions

(1) **Assumption 1:** We assume that the words provided by The New York Times for the Wordle game are randomly selected from a word database, and not influenced by editorial preferences or player performance. This assumption ensures that data analysis and predictions related to words will be more accurate, reducing the likelihood of incidental cases.

(2) **Assumption 2:** We assume that the number of players participating in the Wordle game remains stable on a daily basis. This ensures that the distribution of the number of attempts recorded for different words by players who successfully complete the game reflects word difficulty more accurately.

(3) **Assumption 3:** We assume that in the Wordle game, if a player fails to guess the answer within six attempts, the game will be considered as failed. Therefore, cases where a word is not successfully guessed within six attempts will be considered as seven attempts.

2.2 Symbol Explanation

The symbols we use in this article are shown in Table 1.

Table 1: Symbol Explanation

Symbol	Description	Unit
i	The number of attempts recorded when a game ends	-
p_i	The percentage of different attempt numbers in the reported results of a word	-
y_i	The difficulty factor corresponding to the counting of different attempt numbers	-
D	The difficulty level of a word	-
λ	The attribute feature vector corresponding to a word/word category	-
VC	The difference value of attribute feature vectors between words/word categories	-

3 Model I: XGBoost Algorithm

XGBoost is an efficient gradient boosting decision tree (GBDT) algorithm [3], which aims to improve the predictive ability of the model through weighted iterations. In this process, a decision tree is trained at each iteration and then combined with the previous trees to form a stronger ensemble model.

3.1 Principles of XGBoost Algorithm

Assuming we have N samples, M features, and the target variable is y, the training data can be represented as a matrix X, where each row represents a sample and each column represents a feature. The size of the matrix X is N x M, and the size of the vector y is N x 1.

The objective of XGBoost is to minimize the following loss function:

$$L(\theta) = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t\left(x_i\right)\right) + \sum_{i=1}^{t} \Omega\left(f_i\right) \tag{1}$$

where $l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i))$ is the loss function, $\hat{y}_i^{(t-1)}$ is the prediction of the first $t-1$ trees, $f_t(x_i)$ is the prediction of the t-th tree, and $\Omega(f_i)$ is the regularization term. The choice of the loss function and regularization term can be adapted to the specific problem, for example, the squared error loss and L2 regularization are commonly used in regression problems.

XGBoost adopts the gradient boosting algorithm, in which the gradient and second derivative of the loss function are computed at each iteration. Let g_{it} and h_{it} be the gradient and second derivative

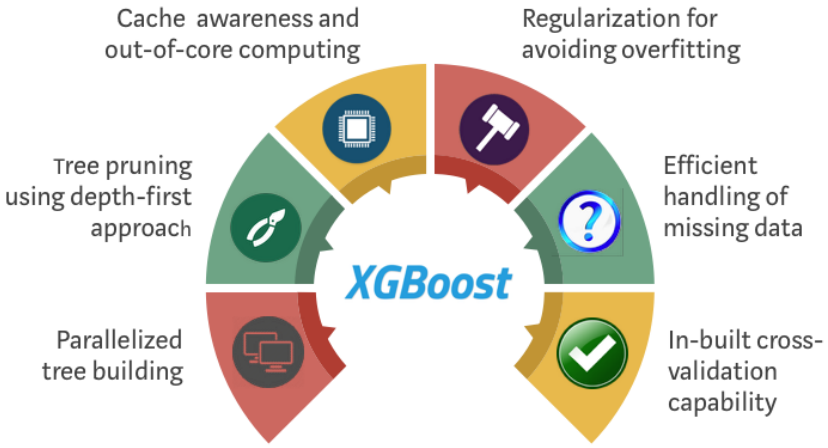


Figure 3: XGBoost Algorithm Illustration

of the loss function, respectively, for the i-th sample at iteration t. In order to construct the t-th tree, the following optimization problem needs to be solved:

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \left[g_{it}\theta_i + \frac{1}{2} h_{it}\theta_i^2 \right] + \gamma T + \frac{\lambda}{2} \sum_{j=1}^{T} w_j^2 \tag{2}$$

To solve the optimization problem mentioned above, a greedy algorithm is used to find the optimal partition at each leaf node. Specifically, for each feature, the corresponding gain is computed and the feature and split point with the maximum gain are chosen. The dataset is then split into two subsets, which are recursively entered into the left and right subtrees. To avoid overfitting, pruning is also performed.

Once all trees are trained, XGBoost combines them to form an ensemble model. Specifically, for a new sample x, the predicted value of XGBoost is:

$$\hat{y} = \sum_{t=1}^{T} f_t(x) \tag{3}$$

where T denotes the number of trees and $f_t(x)$ represents the predicted value of the t-th tree for the sample x.

Finally, the gradient descent algorithm is used to optimize the loss function. In each iteration, the first and second derivatives of the loss function are computed based on the current model's predicted values, and the model's parameters are updated according to the derivatives. XGBoost uses the second-order Newton method to accelerate the optimization process, which converges faster to the optimal solution.

The pseudocode for the XGBoost algorithm is described in **Algorithm** 1 below:

Algorithm 1: XGboost algorithm

Data: Dataset and hyperparameters

Initialize $f_0(x)$;

for $k = 1, 2, ..., M$ **do**

  Calculate $g_t = \frac{\partial L(y,f)}{\partial f}$;

  Calculate $h_t = \frac{\partial^2 L(y,f)}{\partial f^2}$;

  Determine the structure by choosing splits with maximized gain $\mathbf{A} = \frac{1}{2}\left[\frac{G_L^2}{H_L} + \frac{G_R^2}{H_R} - \frac{G^2}{H}\right]$;

  Determine the leaf weights $w^* = -\frac{G}{H}$;

  Determine the base learner $\hat{b}(x) = \sum_{j=1}^{T} wI$;

  Add trees $f_t(x) = f_{t-1}(x) + \hat{b}(x)$;

end

Result: $f(x) = \sum_{t=0}^{T} f_t(x)$

3.2 Experiment:

3.2.1 Experiment Setup:

Data Preprocessing

Outlier Handling: The box plot outlier detection method is a statistical method that calculates the distance between data points and the box boundaries by plotting the data in a box plot to detect outliers. Typically, a threshold K is set, and data points that are K times beyond the box boundaries are considered outliers. This method is simple and applicable to different types of datasets, and is not affected by the data distribution type. In this paper, the answer submission data for Contest 529 is considered an outlier, but its removal does not affect the data distribution. Similarly, the word "clen" in Contest 525 does not meet the contest requirements of having five letters and is removed.

Data Normalization: MinMaxScaler is a linear transformation normalization method that can scale the range of feature values in the original data to a fixed range. Specifically, for a feature x with a value range of $[x_{min}, x_{max}]$ in the original data, MinMaxScaler normalizes it to the range of $[0, 1]$ as follows:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

After normalization, the range of the feature values is scaled to between 0 and 1, which avoids huge differences in the value ranges of different features and improves the performance of the model. Normalization makes the weights of each feature more balanced, preventing certain features from having a disproportionately large impact on the model's prediction results. In addition, the normalized

data range is fixed between 0 and 1, reducing the dimensional differences between features, making it easier to compare and weight features. Therefore, MinMaxScaler is a commonly used preprocessing method that can improve the performance and interpretability of the model.

Kalman Filtering: In this paper, Kalman filtering is used during the data preprocessing stage to smooth the number of attempts data and eliminate large noise and fluctuations in the data. The Kalman filter is based on Bayesian probability theory and a dynamic model that describes the process of data changes, and combines measurement data to update the model, resulting in a more accurate state estimation. The algorithm has the advantages of high computational efficiency and real-time performance, making it suitable for filtering continuous data. After Kalman filtering, the fluctuation amplitude of the number of attempts data is significantly reduced, and the data is smoother, providing a more accurate input for subsequent model training.

Evaluation Matrix

To evaluate the performance of the proposed model, three commonly used metrics were used in this paper, namely mean squared error (MSE), root mean squared error (RMSE), and mean absolute error (MAE).

MSE is obtained by calculating the square of the difference between the predicted value and the actual value, and taking the mean of these values. Specifically, the calculation of MSE is as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value for the i-th instance, and n is the total number of instances.

RMSE is the square root of MSE, used to measure the average magnitude of errors. Specifically, the calculation of RMSE is as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

MAE is another commonly used metric for evaluating the performance of regression models. It is obtained by calculating the absolute difference between the predicted value and the actual value, and taking the mean of these values. Specifically, the calculation of MAE is as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

In the experiments conducted in this paper, MSE, RMSE, and MAE metrics were used to evaluate the performance of the proposed model. These metrics provide quantitative standards for measuring the ability of the model to predict the target variable. Smaller values of MSE, RMSE, or MAE indicate better model performance.

RandomizedSearchCV Algorithm

To improve the performance of deep learning, we considered using the RandomizedSearchCV algorithm for hyperparameter tuning.

The RandomizedSearchCV algorithm[4] is a cross-validation-based hyperparameter optimization algorithm that automatically finds the best combination of hyperparameters to optimize the performance of machine learning models. It uses a random search method to search for a set of hyperparameters in the hyperparameter space for training and evaluating models, avoiding the problem of predefined search grids while saving time and computing resources. The algorithm evaluates model performance through cross-validation and returns a model object with the best set of hyperparameters. When using this algorithm, parameters such as the algorithm model, hyperparameter distribution, sample size, evaluation metrics, cross-validation folds, and parallel job number need to be specified.

The pseudocode for describing the RandomizedSearchCV algorithm is presented in **Algorithm** 2 below:

Algorithm 2: RandomizedSearchCV algorithm

Data: Dataset and hyperparameters
Initialize $f_0(x)$;
for $k = 1, 2, ..., n$ **do**
    Sample a set of hyperparameters from the distribution;
        params = sample_hyperparameters(param_distribution);
    Train and evaluate the model with the sampled hyperparameters;
        model.set_params(**params);
        scores = cross_val_score(model, X, y, scoring=scoring, cv=cv);
        mean_score = np.mean(scores);
    Update the best hyperparameters if the score is better:
    if *mean_score* $\geqslant$ *best_score* **then**
        best_score = mean_score;
        best_params = params;
    end
    Return: best_score, best_params
end

3.2.2 Prediction Interval for the Number of Reports

Using RandomizedSearchCV to search for the optimal parameters, the XGBoost model constructed in this section uses MSE as the loss function, Contest Number as the input X, and the number

of reported results as the output y. The subsampling ratio is set to 0.8, the L2 and L1 regularization weights are set to 0.5 and 0, respectively, the number of decision trees is set to 50, the minimum weight of the leaf node is set to 0.5, the maximum depth of the decision tree is set to 9, the learning rate is set to 0.1, and the minimum loss reduction required for node splitting is set to 0. The column subsampling ratio is set to 0.6. All data is randomly divided into training and test sets, with a test set proportion of 0.3. The experimental platform uses a 14-core Intel(R) Xeon(R) Gold 6330 CPU and an NVIDIA GeForce RTX 3090 GPU with 24GB of GPU memory. The model hyperparameters are set as follows:

Table 2: hyperparameters

subsample	0.8	reg_lambda	0.5	reg_alpha	0
n_estimators	50	min_child_weight	5	max_depth	9
learning_rate	0.1	gamma	0	colsample_bytree	0.6

By observing the distribution of the number of reported results and the word index in **Figure 4**, it can be found that the number of reported results increases with the increase of the contest index, but the growth rate gradually slows down after contest index 150, and tends to be stable.

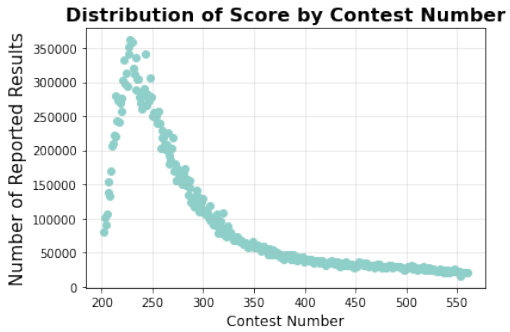


Figure 4: the number of reported results

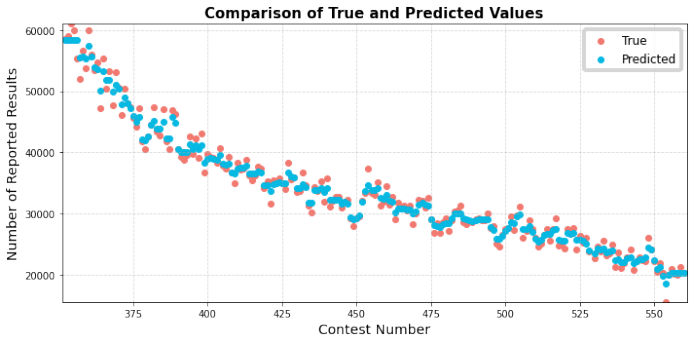


Figure 5: Comparison of True and Predicted Values

To predict the number of reported results on March 1, 2023, this paper first converted the date to a contest index. As the contest is held every day, the contest index for March 1, 2023 is 619. Next, we used the XGBoost model built in this paper to predict the number of reported results for this contest, and the predicted result was in the range of $(16645.494, 25407.666)$. However, this range does not directly indicate the confidence of the prediction, so further analysis and validation of the prediction results are needed.

To calculate the confidence interval of the predicted result, this paper used a commonly used non-parametric statistical method, namely Bootstrapping. The Bootstrapping method estimates the distribution and confidence interval of the statistics by repeatedly sampling the data. Compared with traditional parametric statistical methods, Bootstrapping does not require any assumptions about the

data distribution, making it more flexible and applicable to various types of data. In this paper, we used the Bootstrapping method to construct the confidence interval, assuming a sampling frequency of 1000 times and a confidence level of $1 - \alpha = 0.95$, to obtain the confidence interval of the number of reported results. With this method, we can comprehensively evaluate the confidence of the prediction results and improve the accuracy and reliability of the prediction results.

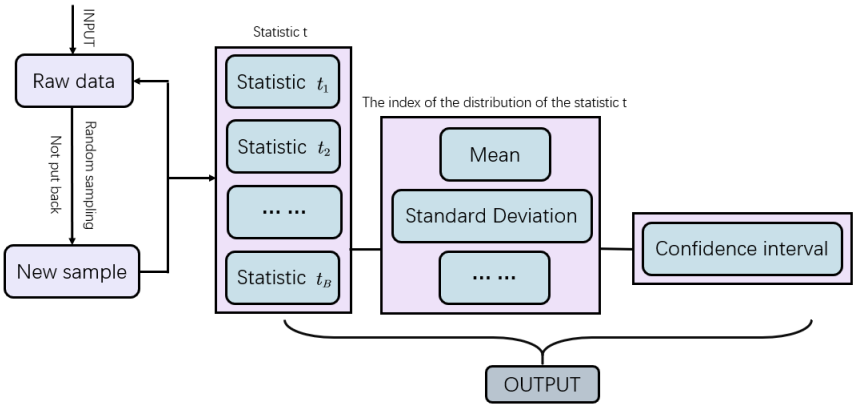


Figure 6: Bootstrapping Method

In this study, we used the Bootstrapping method to calculate a 95% confidence interval for the predicted results, which was found to be $(18289.66, 19917.65)$. This means that we have 95% confidence in the range of predicted values for the given dataset.

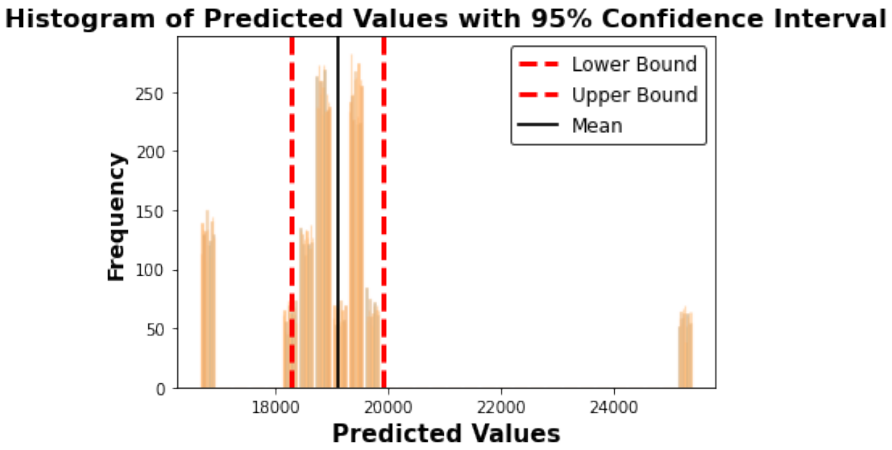


Figure 7: Histogram of Predicted Values with 95% Confidence Interval

This histogram shows the distribution of the predicted values of our model, with the x-axis representing the range of predicted values and the y-axis representing the frequency of each predicted value. The predicted values roughly follow a normal distribution, centered around 18000 to 20000. The red dashed lines indicate the lower and upper bounds of the confidence interval, which are 18266.48 and 19940.83, respectively, and indicate that we can have 95% confidence that the true predicted value

will fall within this range in this sample. The black solid line represents the sample mean of 19103.656, which is the average value of all predicted values and can serve as the central point of the sample.

The number of reported results varies from day to day, and according to our model, the number of reported results increases with the contest number, but the rate of increase gradually slows down after contest number 150 and tends to be stable. This change is caused by the **spread of the game in real life** and the **daily difficulty**.

3.2.3 Prediction of reported results distribution

We have used three different heuristic algorithms, LSTM, MLP, and XGBoost, to predict the future distribution of reported results using the word attribute feature data of Model II, and obtained the prediction results of different algorithms. For example, the situation of 2 tries in future reported results is shown in **Figure 8** (the figures of other situations are detailed in the appendix section, **Figure 14**).

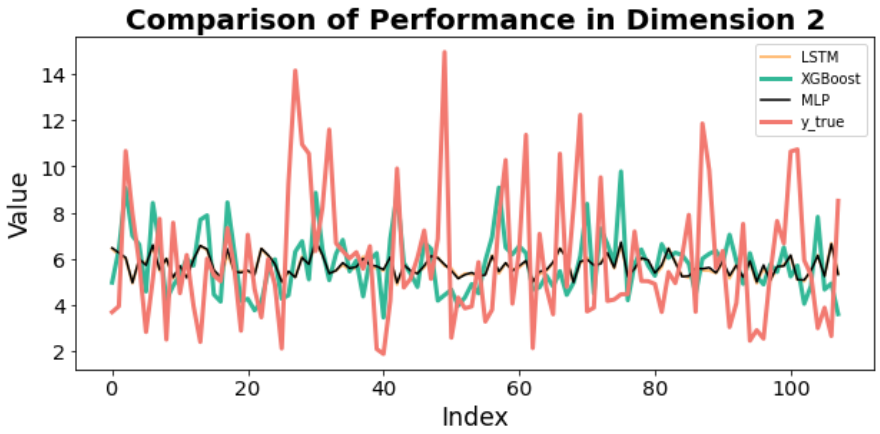


Figure 8: Comparison of Performance in Dimension 2

We compiled the distribution predictions for the word "EERIE" on March 1, 2023, using the word attribute features data from Model II, generated by three different heuristics algorithms: LSTM, MLP, and XGBoost. The results are shown in Table 3.

Table 3: Performances Of MLP,LSTM

METHOD	ATTEMP1			ATTEMP2			ATTEMP3			...	ATTEMP7		
	MSE	RMSE	MAE	MSE	RMSE	MAE	MSE	RMSE	MAE	...	MSE	RMSE	MAE
MLP	0.55	0.74	0.45	8.11	2.85	2.21	31.86	5.64	4.63	...	3.90	1.98	1.47
LSTM	0.55	0.74	0.49	8.16	2.86	2.22	31.96	5.65	4.64	...	**3.84**	**1.96**	1.45
XGBOOST	**0.55**	**0.74**	**0.45**	**7.34**	**2.71**	**2.05**	**29.16**	**5.40**	**4.45**	...	4.38	2.09	**1.45**

Based on the above figures, it can be observed that the prediction values of XGBoost algorithm are the closest to the true values on the test set. Therefore, the prediction values of XGBoost algorithm will be used in this task : $1\text{try} = 1.8873565$, $2\text{tries} = 7.1258245$, $3\text{tries} = 25.00998$, $4\text{tries} = 31.124655$, $5\text{tries} = 24.092382$, $6\text{tries} = 14.709336$, $\mathbf{X}\text{tries} = 5.2613564$.

4 Model II: Extraction and Weight Evaluation of Word Feature Attributes

4.1 Feature Extraction

In the process of word guessing games, the words that come to a player's mind are often related to whether the word is frequently used and whether its structure is simple, i.e., simple-structured high-frequency vocabulary is more likely to be chosen by players as guessed words. Therefore, we need to extract the attributes of commonly used words in daily life in order to determine whether a word is more likely to be associated by players.

Based on the phonetics and general structure of words in linguistics, we selected seven attributes to extract, including **common consonant clusters**, **common vowel clusters**, **the number of diacritical marks in phonetic notation**, **consonant-vowel-consonant (CVC) structure**, **number of repeated letters**, **types of repeated letters**, and **the number of similar words with the same structure**[5].

- Consonant clusters and vowel clusters refer to the combination of adjacent consonant letters and vowel letters, respectively;
- Diacritical marks in phonetic notation are used to indicate changes in pronunciation, intonation, and sound intensity;
- CVC structure refers to the alternating appearance of consonants and vowels in a word, which is the basic syllable structure of a word [6];
- The number and types of repeated letters can be used to analyze the spelling rules and syllable structures of words;
- Similar words with the same structure refer to words that have similar syllable structures and letter combinations. In this paper, we used the Jaro-Winkler similarity to measure their similarity [7].

With these attributes, we can obtain the percentage of the influence of each day's words on the player's score under the difficult mode.

4.2 Jaro-Winkler Similarity

For the first 7 attributes, we use the International Phonetic Alphabet as a standard and statistical methods to extract features from the given words. For the 8th attribute (number of similar words), we use the Jaro-Winkler distance algorithm to evaluate the similarity between words. The mathematical model of the algorithm is as follows:

$$\mathrm{Sim}_j = \begin{cases} 0 & \text{if m} = 0, \\ \frac{1}{3}\left(\frac{m}{|S_1|} + \frac{m}{|S_2|} + \frac{m-t}{m}\right) & \text{otherwise .} \end{cases} \tag{4}$$

In Formula (4), Sim_i is the length of string S_i; m is the number of matching characters; t is the number of character transpositions (the number of times that characters in S_1 and S_2 differ but are in the same position, divided by 2); and Sim_j represents the Jaro similarity used to measure the similarity between words.

$$\mathrm{Sim}_w = \mathrm{Sim}_j + (lp\,(1 - \mathrm{Sim}_j)) \tag{5}$$

On the basis of Jaro similarity, a new similarity measure called Jaro-Winkler similarity (Sim_w) is introduced, which takes into account the common prefix l (with a maximum value of 4) and a constant factor p. The score of pairs of strings is adjusted upward if they share a common prefix, and the value of p is limited to 0.25 to prevent the similarity score from exceeding 1. If the similarity score of S_1 and S_2 is less than or equal to $\left\lfloor \frac{\max(|s_1|, |s_2|)}{2} \right\rfloor - 1$ and they share the same label, they are considered a match.

Permutation importance [8] is a widely used feature selection method that measures the contribution of each feature to the model's performance. The basic idea is that if a feature contributes to the model's performance, then shuffling the values of that feature randomly will cause a decrease in the model's performance, since the model has already learned the importance of that feature during training. Conversely, if a feature does not contribute to the model's performance, then shuffling the values of that feature randomly will not affect the model's performance.

In this study, permutation importance was used to evaluate the weights of the attributes. First, the XGBoost model in Model 1 was used to regress on the seven types of features and the percentage of difficult selection patterns in Model 2, and the baseline score of the model was calculated when only the training set was used. In this study, the RMSE was used to calculate the baseline score, which was 0.0273.

Next, for each feature, the feature was randomly permuted, and the impact of this permutation on the model's performance score was calculated. This impact score is the feature importance score. By sorting all features according to their importance scores from highest to lowest, we can analyze

the contribution of each feature to the model's performance and select the most important features or eliminate unimportant ones. The evaluation process is as follows **Figure 9**:

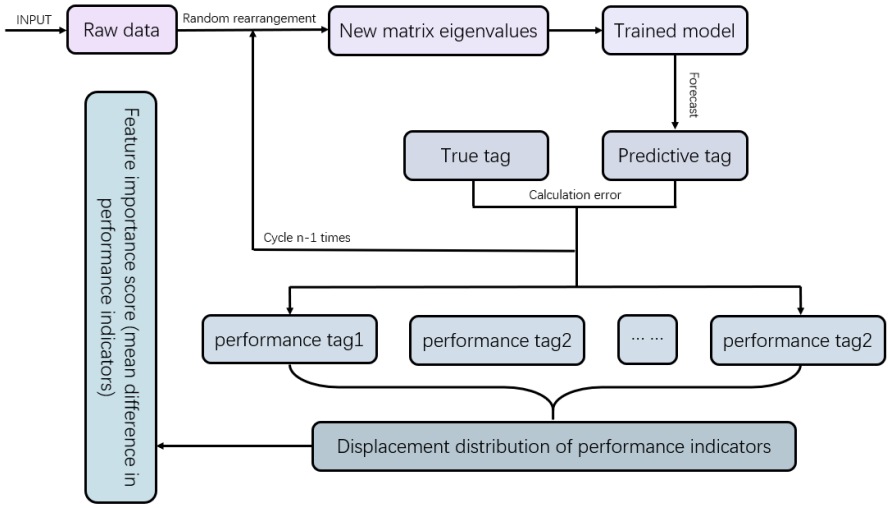


Figure 9: Permutation importance work flow

We evaluated the importance of each attribute using the Permutation Importance method, and the results are shown in Figure 10.

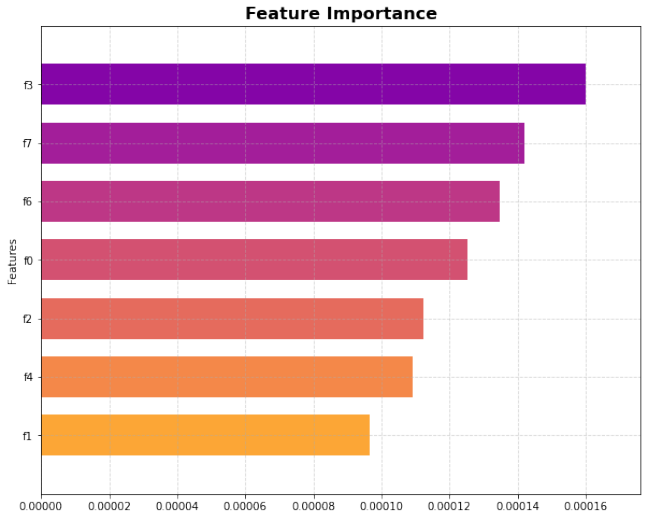


Figure 10: Feature Importance

According to the results of Permutation Importance, we can further analyze the importance of each feature for the model's predictions. When arranged from highest to lowest, the importance score of the number of similar words is the highest (0.1512), indicating that this feature has the most significant impact on predicting difficult selection patterns. The second most important features are the initial consonant, vowel, and final consonant structure (0.1103) and the presence of vowel clusters (0.0832),

which have a relatively large impact on the model's predictions. The importance scores of whether there are consonant clusters (0.0489), the number of repeated letters (0.0294), and whether there are diacritical marks (0.0269) are relatively low, indicating that these features have a smaller impact on the predictions. Finally, the importance score of the number of unique repeated letters is 0, indicating that the impact of this feature on predicting difficult selection patterns can be ignored.

These results are helpful for understanding the important features of the model's predictions and improving the model. For example, we can consider adding features related to the number of similar words, or further exploring the information of features such as initial consonant, vowel, and final consonant structure and the presence of vowel clusters to improve the accuracy of the model. At the same time, we can also remove less important features from the model to reduce its complexity and computational cost.

5 Model III: Word Difficulty Classification

5.1 Word Difficulty Evaluation

To establish a model that classifies words according to their difficulty, we first consider evaluating the difficulty of words based on the distribution of the results reported.

For word difficulty, we consider using a scoring accumulation method, multiplying the percentage p_i of each attempt i (where $i = 1, \cdots, 6, 7$) in the distribution of results corresponding to a certain word by the corresponding difficulty factor y_i. Finally, we add up these products to obtain an evaluation expression for the difficulty of the word:

$$D = \sum_{i=1}^{7} p_i * y_i \tag{6}$$

To calculate the difficulty factor y_i of a word for each attempt i, we propose three different elementary function models:

- Linear model: $y_i = i$, the linear model indicates that the difficulty factor changes at an equal interval as the number of attempts increases;
- Logarithmic model: $y_i = \ln i + 1$, the logarithmic model indicates that the incremental difficulty factor gradually decreases as the number of attempts increases;
- Exponential model: $y_i = \mathrm{e}^{i-1}$, the exponential model indicates that the difficulty factor exhibits explosive growth as the number of attempts increases.

Next, we can evaluate the difficulty of each word for each competition based on the corresponding report number.

Table 4: Word Difficulty Evaluation

	manly	**molar**	**havoc**	$\cdots$	**gorge**	**crank**	**slump**
Linear model	413.00	422.00	464.00	$\cdots$	440.00	418.18	438.38
Logarithmic model	238.11	239.31	249.28	$\cdots$	245.32	239.84	244.87
Exponential model	4011.94	5002.19	7164.27	$\cdots$	5112.67	4129.29	5096.34

5.2 Feature Attributes of Word Classes

To investigate which model more accurately classifies words, we need to observe the degree of difference between different word groups. To do so, we consider using the weighted feature attribute values λ_j of the word group and summing them up to obtain the feature attribute value C_k of the group. Then, we can evaluate the effectiveness of different models in classifying words based on the discreteness of the feature attribute values.

We divide word difficulty into five levels, "Easy, Moderate, Normal, Challenging, Hard," and classify them based on the percentile rank (from low to high) of the word's difficulty ranking.

When characterizing the feature attributes of word classes, a reasonable assumption is that the more central a word is, the greater its influence on the feature attributes of the class, while the more marginal words have less influence. Therefore, when calculating the feature attributes of a word class, we can use a Gaussian distribution to calculate the probability density value of the lower percentile rank corresponding to that percentile rank based on the difficulty level of each word in the class and its percentile rank in the class, in order to obtain the weighting factor of the feature attribute for that word.

Next, we can calculate the feature attribute values of each word class under the three models (Partial attribute values are shown in **Table 6**).

To compare the effectiveness of different models in word classification, we can calculate the discreteness of the feature attribute values of adjacent word classes under each model. Since the focus of word difficulty classification is mainly on the distinction between adjacent difficulty classes, the discreteness of the feature attribute values of adjacent word classes is defined as the sum of the squares of the differences in feature attribute values between adjacent difficulty classes:

$$VC = \sum_{k=1}^{4} (C_{k+1} - C_k)^2. \tag{7}$$

The calculated discreteness values are $VC_{\text{linear}} = 6.1798$, $VC_{\text{logarithmic}} = 6.0497$, and $VC_{\text{exponential}} = 5.4493$.

In conclusion, the exponential function model has better classification effectiveness, and it can better distinguish the differences between word classes of different difficulty levels. Therefore, when evaluating word difficulty, we choose to use the **linear model**.

5.3 Difficulty Classification of ”EERIE”

For the specific word ”EERIE,” since we do not know the distribution of the reported results with it as the answer, we need to extract its feature attributes for difficulty classification. To classify the difficulty level of ”EERIE,” we compare the feature attributes of ”EERIE” with the discreteness values of the feature attributes of each word class. According to the principle of minimum relative discreteness, ”EERIE” will be classified into the word class with the smallest discreteness value.

Table 5: The discreteness values of the feature attributes of ”EERIE” compared to each word class

	Easy	**Moderate**	**Normal**	**Challenging**	**Hard**
Attribute characteristic difference	4.4150	4.6168	4.7486	4.3413	3.9939

It can be found that according to the difficulty classification model in this paper, the difficulty of ”EERIE” is **Hard**.

6 Model IV : Dataset Feature

6.1 Distribution of the times of try

By plotting the probability density distribution of the number of times players tried each word, we found that they follow a normal distribution, as shown in **Figure 11**. It can be observed that the shapes of some normal distributions are different from each other. Therefore, we use skewness coefficient to describe whether they are skewed to the left or to the right relative to the normal distribution, and use kurtosis coefficient to describe the height of their peaks.

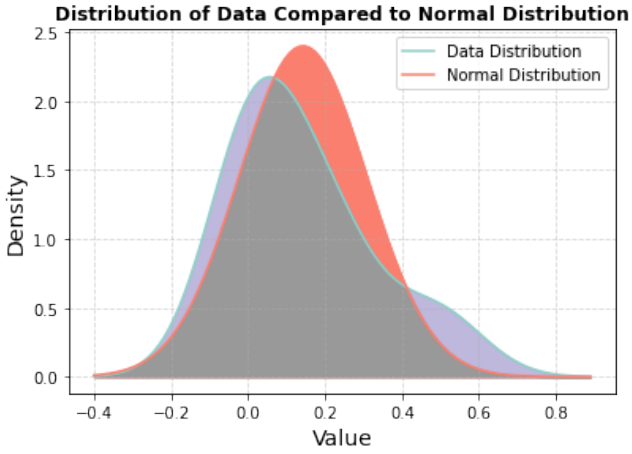


Figure 11: Distribution of Data Compared to Normal Distribution

Meanwhile, we can also learn that the higher the difficulty of the word on a given day, the more the normal distribution is skewed to the right, i.e., the greater the skewness coefficient. Conversely, the lower the difficulty of the word on a given day, the more the normal distribution is skewed to the left, i.e., the smaller the skewness coefficient. In addition, the kurtosis coefficient can reflect the concentration of player skills at the current difficulty level. Therefore, we can control the distribution of player tries by adjusting the difficulty of the words.

For a great game, we should make players feel comfortable during the gameplay, but also add some pressure to keep their motivation. Therefore, we need to ensure that the number of tries made by players follows a normal distribution while maintaining the difficulty of the words.

To expand the answer database, we can use cosine similarity to find suitable words from the vocabulary after extracting the feature vectors of the words. Specifically, the formula for calculating cosine similarity is:

$$sim(\omega_1, \omega_2) = \frac{v_1 v_2}{\|v_1\| \, \|v_2\|}. \tag{8}$$

And we can take the reciprocal of the skewness coefficient to adjust the difficulty of the words appropriately, so that the difficulty of the words over a period of time shows a relatively uniform normal distribution. This avoids the repetition of word difficulty and can effectively attract players to continue playing the game.

6.2 Changes in word features under different levels of difficulty

In the difficulty classification model, we have calculated the attribute feature values for each category, and **some of the attribute feature values** are as follows.**Table 6**.

Table 6: Partial attribute feature values of each word class under the linear function model

	count duplicate letters	**count duplicate types**	**has diacritics**	**Count Similar Words**
Easy	0.0342	0.0166	0.0461	0.1708
Moderate	0.6846	0.3328	0.9232	3.4198
Normal	0.4819	0.2409	0.9550	4.0354
Challenging	0.5031	0.2515	0.9111	3.7898
Hard	0.6526	0.3263	0.9328	3.5930

The various attributes of a word are interdependent, so it is not easy to discover strict rules when word difficulty is used as the basis for classification. However, we can identify some general trends between word attributes and difficulty from the table above, such as:

1.The number of repeated letters is roughly positively correlated with word difficulty.

2.The number of types of repeated letters is roughly positively correlated with word difficulty.

3.The number of diacritical marks is roughly positively correlated with word difficulty.

4.The more words a word has that are structurally similar to it, the easier it is to understand, but after a certain number of similar words, the difficulty increases instead.

Figure 12: Diacritical marks

The above are some examples of the patterns we discovered from the extracted attributes. These patterns are actually related to the characteristics of language. For example, words with fewer diacritical marks are easier to remember and use, so when playing the Wordle game, people usually try to fill in words with fewer diacritical marks first, making these words easier to guess. Another example is structurally similar words, which are more easily associated by people, but when there are too many structurally similar words, they can actually hinder people's guessing of the final answer. We further consulted linguistic literature and found that these patterns are in line with some basic linguistic characteristics.

6.3 Analyze the impact of the day of the week on the overall average score

By plotting a scatter plot of the change in scores over time, we can see that the number of Wordle users has stabilized after a period of fermentation, and the weekly average score has become stable. Therefore, when analyzing the overall effect of the day of the week on the mean score, we only took the data that stabilized after the 352nd game as the raw data. The relationship between score and time is shown in **Figure 4**.

Based on the selected data, we calculated the average score for each day of the week and the overall average, and plotted them on the same graph, as shown in **Figure 13**:

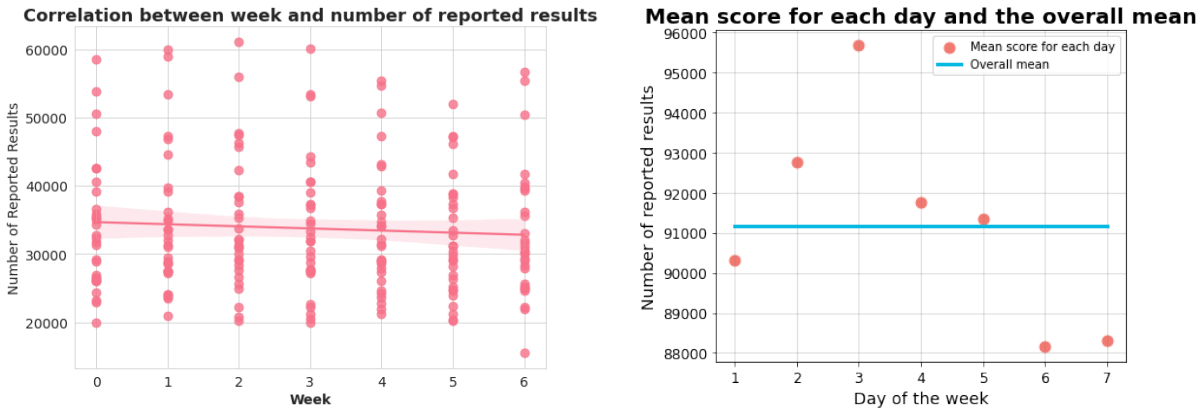


Figure 13: Average daily scores from Monday to Sunday and overall average score

We can see that the scores on Saturday and Sunday are significantly lower than the overall average, and the period with the highest average score is between Tuesday and Thursday. This clearly reflects the time periods when people participate in Wordle, and can provide useful information for game operators.

7 Model Strengths and Weaknesses

7.1 Strengths

The model we built has the following advantages:

- **Combining machine learning and linguistic data mining methods.** By comparing the predictive performance of different models, we found that the XGBoost model performed better than other models on this dataset.
- **Proposing a model for extracting word attributes based on linguistic theory.** The model includes seven features related to the structure and pronunciation of words, providing more dimensions for analyzing word difficulty.
- **Considering three different function models and using Gaussian distribution to adjust weight factors for evaluating word difficulty.** The final function model was selected based on the classification effect of different function models, and the influence of marginal vocabulary on word attributes was reduced in the calculation of word class attribute features.
- **Using the RandomSearchCV algorithm to select model parameters.** This ensured that the model parameters were set to achieve optimal performance.

7.2 Weeknesses

However, our model also has some defects and needs to be improved.

- In this article, the daily number of players participating in the Wordle game is assumed to remain stable, and the difficulty of the words is evaluated based on the percentage of attempts made by players when the game ends. However, in reality, the number of players participating gradually increased in 2022, so the sample sizes of the distributions corresponding to different words may differ greatly, which may lead to bias in the classification of word difficulty. In the next model, researchers could consider collecting and analyzing the daily number of players to adjust the data and make the model more accurate.
- In the model for extracting word attributes, the selection of some features may be subjective and limited, which may have some impact on the accuracy of word difficulty classification.
- The dataset used in this article is relatively small, which limits its generalization ability and may lead to overfitting. It may not fully learn the underlying rules and features of the data.

References

[1] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[2] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[J]. arXiv preprint arXiv:1406.1078, 2014.

[3] Chen T, Guestrin C. Xgboost: A scalable tree boosting system[C]//Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016: 785-794.

[4] Bergstra J, Bengio Y. Random search for hyper-parameter optimization[J]. Journal of machine learning research, 2012, 13(2).

[5] Lindfield K C, Wingfield A, Goodglass H. The role of prosody in the mental lexicon[J]. Brain and Language, 1999, 68(1-2): 312-317.

[6] Kearns, D.M. (2020). Does English Have Useful Syllable Division Patterns?. Reading Research Quarterly, 55(S1), S145– S160. https://doi.org/10.1002/rrq.342

[7] Li I. Analyzing difficulty of Wordle using linguistic characteristics to determine average success of Twitter players[J]. 2022.

[8] Permutaion Importance —— Rank importance. https://iceflameworm.github.io/2019/08/17/permutaion-importance/.

Appendices

A Attached figure

The group of pictures in this paper are as follows:

Group of pictures 1 Forecast the distribution of future report results

Figure 14: Forecast of remaining attempts

B A letter to the puzzle editor of the New York Times

Letter

To: The puzzle editor of the New York Times
From: Team 2311888
Date: February21st,2023
Subject:Some information and improvement suggestions from the research of Wordle

Dear Game Developers,

We are three avid fans of the Wordle game who have recently conducted a comprehensive study on the difficulty variation mechanisms of the game. We are writing to share with you our research findings and offer some valuable suggestions to enhance the popularity and entertainment value of the game.

Firstly, we found that the number of submissions for game results tends to stabilize over time, indicating that the player base has become more stable. Secondly, we discovered a correlation between the difficulty of a word and its attribute features. For example, the higher the frequency of a word, the lower its difficulty; and the sounder change structures a word has, the higher its difficulty. These findings suggest that game designers can optimize word selection and settings to enhance the game's challenge and interest. Additionally, we explored the number of attempts players make to guess a word. Our research found that players' guessing attempts roughly follow a normal distribution. Finally, we found a correlation between player scores and the day of the week, with scores showing a downward trend. This may be related to different learning and entertainment patterns of players on weekdays and weekends.

Based on these findings, we have the following suggestions:

First, we recommend optimizing word selection to make the number of attempts players make to guess a word closer to a normal distribution. Our word attribute extraction model can help game designers find the word that best matches the distribution of guessing attempts from historical data and find the closest vocabulary from the word database as the game topic. This can enhance the entertainment value of the game and allow players to enjoy the process more.

Secondly, we suggest designing different game modes or challenge modes for

weekdays and weekends, such as adding rankings and social interaction functions, so that players can compete and communicate with other players, enhancing the social and interactive nature of the game. This can increase player engagement and participation in the game and make it more enjoyable.

Furthermore, game designers can add game modes of different difficulty levels to meet the needs of players of different skill levels. For beginners, you can design simple and easy-to-understand words or add prompts and help functions to help players gradually improve their guessing ability. For advanced players, more difficult word challenges can be added to enhance the game's challenge and entertainment value, increasing player's engagement and immersion.

Finally, to attract more players, game designers can add more language support on the game page, which allowing players to freely choose different languages to play the game. Additionally, different word libraries can be provided to cater to different cultural backgrounds and habits of players from different languages and regions. By adding multi-language support, the game can expand its audience and boost its internationalization.

We hope that our suggestions can provide some valuable insights for the development of the Wordle game and help enhance its popularity and entertainment value. Thank you for taking the time to read our letter.

Sincerely,
three avid fans of the Wordle game