Problem Chosen	2023 MCM/ICM Summary Sheet	Team Control Number
C		2300082

Predicting Wordle Results: Simple but Extraordinary

Summary

Wordle is a very popular recent guessing game. Not only is the process of guessing puzzles intriguing to many players, but the data behind it has many characteristics that have attracted people to study it.

First, by analyzing the daily variation in the number of reports, we build a model for predicting the number of entries based on a long and short-term memory neural network (LSTM). Based on this model, we predicted the number of reports on 1 March 2023 to be between approximately 16,000 and 19,000. Other models were also used as controls to confirm the accuracy of the conclusions. We then used multiple regression analysis to find a correlation between the frequency of letters in words and the proportion of scores using ordinary least squares (OLS). **Next**, we used a generalized regression neural network (GRNN) to build a model for predicting score proportions based on date sequences and letter frequencies, based on the existing correlation results. Based on this model, we predicted the correlation percentage distribution of the word "EERIE" on 1 March 2023. **Finally**, based on the above results, we considered the metrics of word classification, built a word difficulty classification model based on K-mean clustering algorithm (KNA) and deep neural network (DNN), and analyzed the difficulty level of the word "EERIE".

In addition, in the sensitivity analysis, we focus on the effect of subjective parameters on the results. **Finally**, we wrote a letter to the editor of the New York Times to describe our results in detail. Although the model developed in this paper is not very perfect, it may be useful for predicting economic indices, developing coding systems, and other related topics.

Keywords: Wordle; ISTM; GRNN; OLS; DNN; K-means; Prediction

Contents

1 Introduction

1.1 Problem Background

Wordle is a popular word guessing puzzle that is updated daily by The New York Times. The answer to each game is a real word composed of five letters. The player must guess the correct answer within six tries, then record the number of guesses. If the puzzle can't be solved within six attempts, the number of guesses is recorded as X which means failure. Each attempt gives the player a color response to the result, as shown in Figure 1, then the player can take the next step based on the feedback.

Besides, there are two modes in this game: regular mode and hard mode. In hard mode, if the player finds the right letter in a word (the feedback color is green or yellow), they have to use that letter in a later guess, which makes the game harder and more fun. Figure 2 shows an example in hard mode on February 17, 2023, where the player apparently made four attempts.

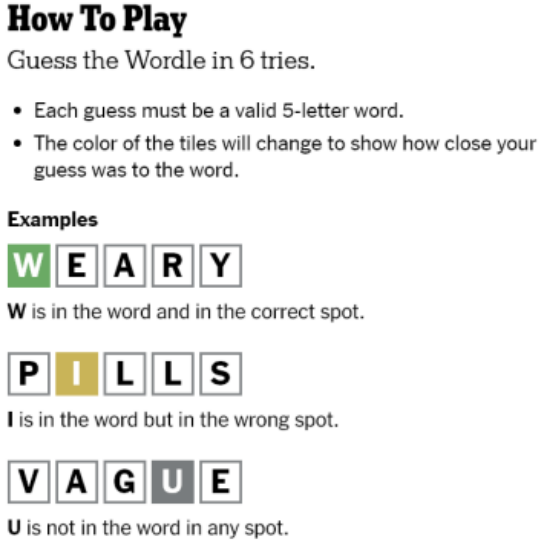


Figure 1: Some rules of the game

Figure 2: An example in hard mode

1.2 Restatement of the Problem

A Wordle score distribution report is available for the period 7 January 2022 to 31 December 2022, based on game scores from a large number of Twitter users, including date, game number, word guessed on game day, number of people reporting results in tweets, number of players, and percentage distribution of guesses.

Considering the background information and the constraints given in the problem statement, we need to solve the following problems:

- Based on the established model, provide a reasonable explanation for the daily variation in the number of results reported and make a specific prediction for the number of results reported on 1 March 2023. Then discuss the percentage effect of word attributes on the number of guesses in hard mode.
- Predict the percentage distribution of the number of guesses (0, 1, 2, 3, 4, 5, 6, X) for a word on a future date. Specifically, the percentage distribution of the number of guesses for the word EERIE on 1 March 2023 is predicted and its uncertainty is discussed.
- Develop and summarize a model, classify words according to difficulty, identify the attributes of words associated with each classification, explain the difficulty of the word EERIE using the established model, then discuss the accuracy of the classification model.
- Discover and describe other interesting features of this data set.

1.3 Problem Analysis

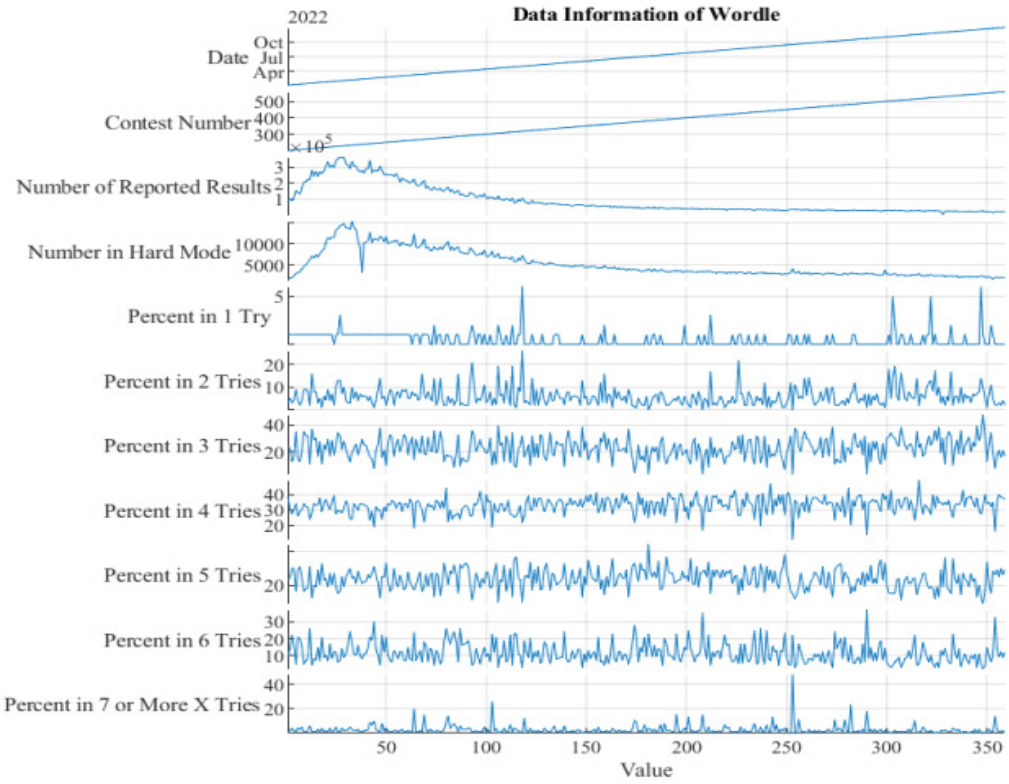


Figure 3: Image representation of data in a dataset

The dataset we have is a large set arranged in reverse chronological order, and we will

first convert it to normal order. As our data is in tabular form and there is a large amount of data, we will first transform it into a more intuitive form for presentation, see the following figure.

By observing the distribution of this data, we can see that:

a. The time interval for these data is a constant value (1 day).
b. Except for the variable race number, which changes linearly over time, all other variables show non-stationarity over time.
c. There is no significant correlation between the percentage distribution of players' guesses and the number of reports submitted and the number of people choosing the hard mode.

Based on the above, we examined the number of reports, the percentage distribution of guesses, and the number of people choosing the hard mode as time series variables.

First, in order to predict the change in the number of reported daily results, we need to adequately approximate the non-linear relationship of the time series, so we consider using a Recurrent Neural Network (RNN) for serially changing data. We also note that the time span of these data is huge (almost a year), and to avoid the problems of gradient disappearance and gradient explosion in the training process of long sequences, we build a Long Short-Term Memory Artificial Neural Network (LSTM) model. A large amount of reasonable data was extracted from the dataset to train the model, and some data was finally tested and compared. We do this repeatedly and predict 1 March 2023 to report the quantity value and record it, and finally arrive at its prediction interval.

At the same time, after building the LSTM model, we compare it with the related models of traditional machine learning to intuitively compare its advantages and disadvantages with other models.

When analyzing the influence of word attributes on the percentage distribution of guessing times, we consider the frequency and length of each letter in the word. Since each word in the dataset has the same length, we replace frequency with frequency in this case. Given the possible correlation between word attributes and the percentage distribution, as it is related to several factors, we use multiple linear regression to establish the relationship between them.

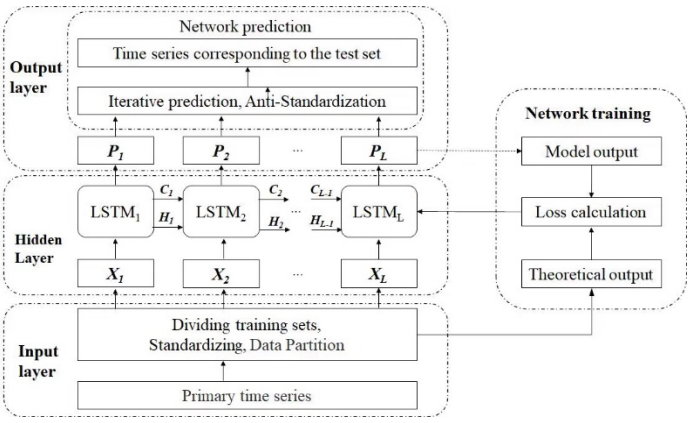
The correlations we obtained were then used to predict the percentage distribution of guesses. We chose the frequency of occurrence of each letter in the word (since the words in the dataset used were of the same length, the frequency was replaced by the frequency) and the date as independent variables, and the seven percentage distributions of the player's i trials (i = {1, 2, 3, 4, 5, 6, X}) as dependent variables, and explored the dependencies between them using a generalized regression neural network (GRNN). We also used the BP neural network algorithm for the control, where we did not use the date as an independent variable, which ultimately also shows that the date had some influence on the final results.

Finally, in order to classify the words by difficulty, we need to cluster them. To make clustering better and easier to implement, we chose the K-mean clustering algorithm (KNA), which imports data on the percentage distribution of player attempts to divide the words into three categories, and sets the three categories to a graded level. The words were coded in the order in which they appeared, so that each word had a unique code corresponding to it. Given

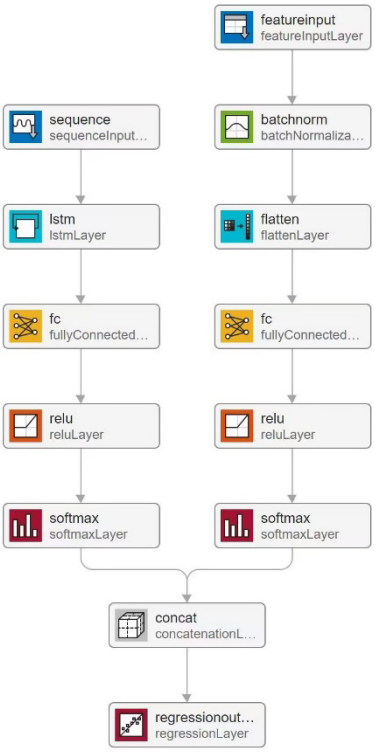
the simplicity of the task, we used the previously obtained percentage distribution of the number of guesses made by the players for the corresponding word to classify it using a deep neural network (DNN), and finally obtained the probability of the word occurring in the three binned classes.

The analysis also revealed some interesting features of the original dataset, which we will show in the form of images.

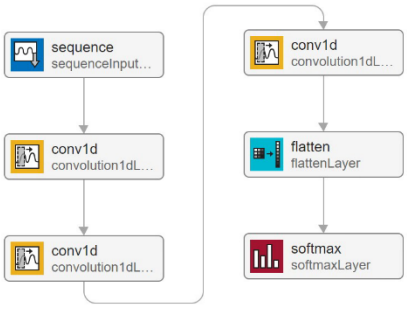
In summary, the whole modelling process is illustrated below.



Model I: Time Series Forecasting Model for the Number of Reports



Model II: A Generalized Regression Neural Network-based Model for Predicting the Percentage of Word Guesses



Model III: A deep neural network-based model for word difficulty classification

Figure 4: Model Overview

2 Assumptions and Justifications

After a thorough analysis of the problem, we make the following reasonable assumptions to simplify our model.

Assumption: The predicted value obtained by LTSM follows a normal distribution with unknown variance (i.e. satisfies the t-distribution).

Justification: The predicted value derived from LSTM can be approximated as the result of many small independent random factors.

Assumption: The 95% confidence interval obtained by Student's t test is the prediction interval.

Justification: Because of the important role of the 95% confidence interval in statistics, we chose it as the prediction interval to make the results both reliable and accurate.

Assumption: People's familiarity with a word objectively depends on how often the letters appear in that word.

Justification: If we look at the Oxford Dictionary, we see an interesting phenomenon: when we look up words using initials, there are more words beginning with "s" and fewer words beginning with "x", "z", etc. There are fewer words beginning with letters. There are fewer words that begin with letters. According to psychological theory, people tend to guess words based on what they are familiar with first, so they tend to choose the commonly used letters as their first guess based on their knowledge base. The commonly used letters are in most cases common letters, so there is a close relationship between letter frequency and people's familiarity with words. Also, people's familiarity with the words inevitably affects the final number of guesses. The current table of letter frequency distribution is shown in the figure. [1]

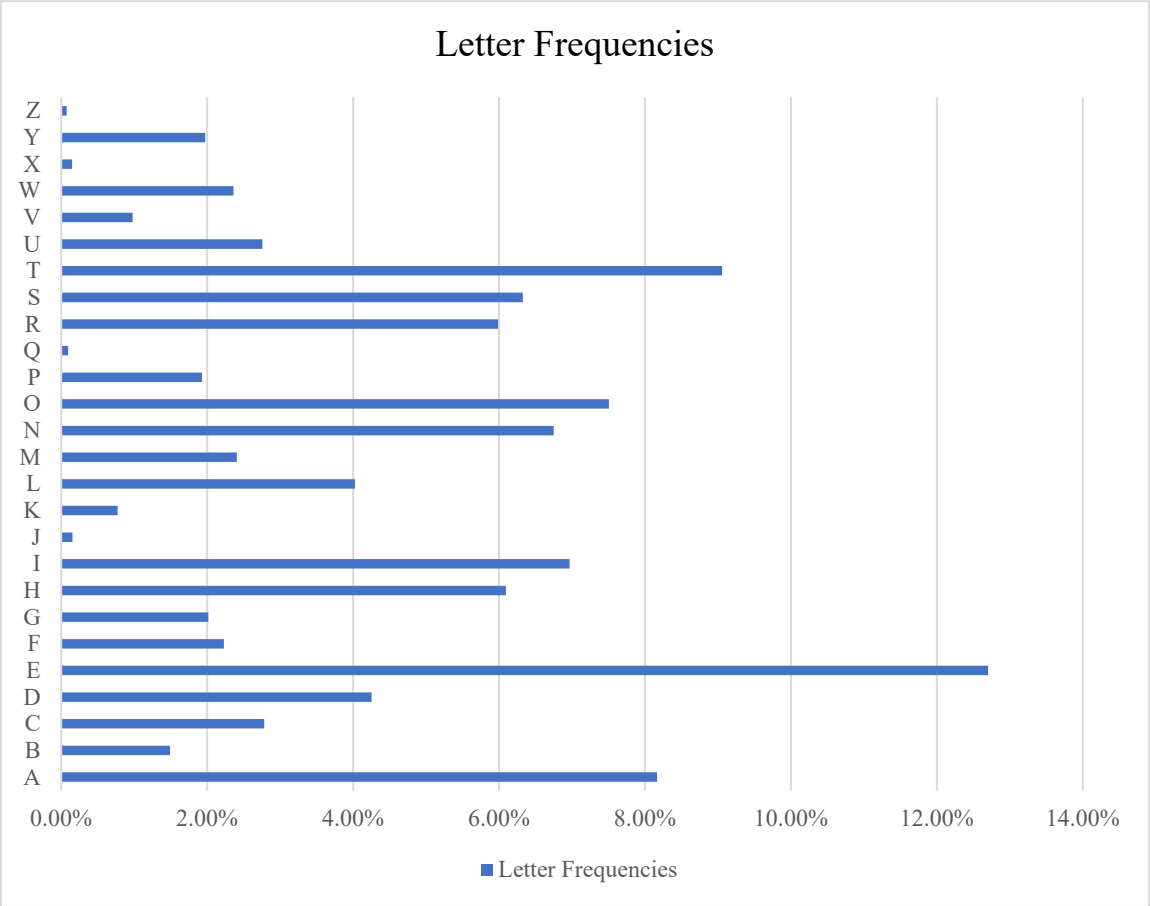


Figure 5: Letter Frequencies

In this paper, we will similarly calculate the frequency of letter occurrences in a known data set and then make predictions based on the resulting results.

3 Model Preparation

3.1 Terminology and Notations

Here are some of the terms we need to define.

- Forgetting gate: A device that causes some information to be forgotten or discarded.
- Output gate: A device that controls the content of output information.
- Input gate: A device that executes input information.
- Input layer: Artificial neural networks have several nodes that receive data. These nodes make up the input layer of the system.
- Hidden layers: The input layers process the data and pass it on to layers further down the neural network. These hidden layers process information at different levels and adjust their behavior as they receive new information.
- Output layer: The output layer consists of nodes that output data. Deep learning models that output "yes" or "no" answers have only two nodes in the output layer. Those models that output a wider range of answers have more nodes.
- Accuracy rate: The proportion of predicted correct samples to the total number of samples; the higher the accuracy rate, the better.
- F_1: The summed average of precision and recall. Precision and recall influence each other. When both need to be balanced, the F1 metric can be used.

The main mathematical notations used in this paper are listed in Table 1.

Table 1: Notations used in this paper

Symbol	Description	Unit
t	Time	day
c_t	The knowledge state of the cell at time t (long-term state)	
f_t	The forgetting factor at time t	
i_t	The inputting factor at time t	
o_t	The outputting factor at time t	
W_*(*=f,i,o,$\tilde{c}$)	The recursive weights	
$\tilde{c}_t$	The immediate state of the cell at time t (short-term memory)	
h_t	The output at time t	
$\sigma(x)$	The logistic function[1]	
$\tanh(x)$	Hyperbolic tangent function[2]	
b_* (*=$f,i,o,\tilde{c}$)	Perturbation parameters	
x,y	Random Variables	
$f(x,y)$	The joint probability density of x, y	
n	Sample size	
p	The dimension of the random variable x	
ξ	Smoothness factor	
ω	Linear relationship coefficient	

[1] The general form of the logical function is: $\sigma(x)=\frac{1}{1+e^{-x}}$ which can map data to (0, 1) interval.

[2] The general form of the hyperbolic tangent function is: $\tanh(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$. Data can be mapped to (-1, 1) interval.

3.2 The data

3.2.1 Data Collection

The data in this paper comes from Twitter, which is a known dataset, so there is no need to collect data again. We just need to extract the data that is useful to us.

3.2.2 Data Cleaning

We noticed that some data in the dataset would affect the accuracy of our model, so we need to remove or replace the corresponding data to meet our requirements.

There are two pieces of data in particular in this dataset that deserve our attention.

Table 2: Abnormal data

Date	Contest Number	Word	Number of Reported Results	Number in Hard Mode	Percent in						
					1 Try	2 Tries	3 Tries	4 Tries	5 Tries	6 Tries	7 or More Tries (X)
12/16/2022	545	rprobe	22853	2160	0	6	24	32	24	11	3
12/11/2022	540	naïve	21947	2075	1	7	24	32	24	11	1

Among them, the word "rprobe" is wrong and should be "probe". Having corrected it, we decide to keep it. Also, the word "naïve" is an English word with the same meaning as "naive", but we cannot spell the French word " ï" in the Wordle game. So, it does not meet the requirements of our model, and we will remove it.

4 Time Series Forecasting Models for the Number of Reports

For a time series variable such as the number of reports, we urgently need a model that can adequately fit its changing trend. After realizing that the general RNN model cannot meet the requirements of long time series variables, we built a suitable prediction model for the number of reports based on the LSTM model.

4.1 The Establishment of Forecasting Model

Hochreiter and Schmidhuber (1997) [2] proposed the LSTM cell. They improved the memory capacity of the standard recurrent cell by introducing a “gate” into the cell. Gers, Schmidhuber, and Cummins (2000) [3] modified the original LSTM in 2000 by introducing a forgetting gate into the cell. [4]

LSTMs are widely used for processing sequential information, such as speech recognition and machine translation, due to their ability to better detect long-term dependencies.

The diagram below shows the structure of the cell.

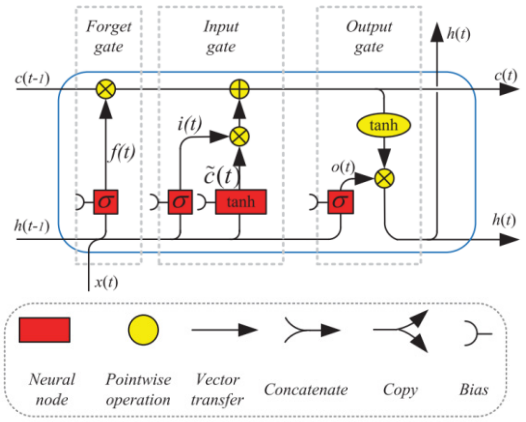


Figure 6: Structure diagram of LSTM unit

In the training process of the LSTM neural network, the data features at time *t* are first input to the input layer and the results are output by the excitation function. The output result, the output of the hidden layer at the moment *t-1* and the information stored in the cell unit at the moment *t-1* are input to the nodes of the LSTM structure, and the data are output to the next hidden layer or output layer through the processing of the input gate, output gate, forget gate and cell unit, and the results of the nodes of the LSTM structure are output to the neurons of the output layer, and the calculation of the back propagation error, and update each weight value.

Based on the intrinsic structure of the above unit, its mathematical representation is as follows.

$$\begin{aligned} f_t &= \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f) \\ i_t &= \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i) \\ \tilde{c}_t &= \tanh(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + b_{\tilde{c}}) \\ c_t &= c_{t-1} + i_t \cdot \tilde{c}_t \\ o_t &= \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o) \\ h_t &= o_t \cdot \tanh(c_t) \end{aligned} \tag{4.1}$$

We choose the Mean Absolute Percentage Error (MAPE) as the loss function.[3][4] It is also the Python-based environment with PyTorch as the deep learning framework for training and prediction. Other traditional machine learning models are also used for comparison. When it comes to fitting, many machine learning algorithms have a place in prediction, such as (multi-layer perceptron model (MLP), linear regression, decision tree model, light gradient boosting machine (LGBM) and extreme gradient boosting model (XGBoost) to name a few).

[3] The general form of MAPE is: $J=\frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i-y_i}{y_i}\right|$, where n is the number of samples in the training set, $\hat{y}_i$ is predicted values and y_i is real value.

[4] MAPE of 0% indicates a perfect model, while MAPE greater than 100% indicates a poor model.

4.2 Results

4.2.1 LSTM-based Prediction Results

After data cleansing, the distribution of the number of reports per day in the original dataset is as follows.

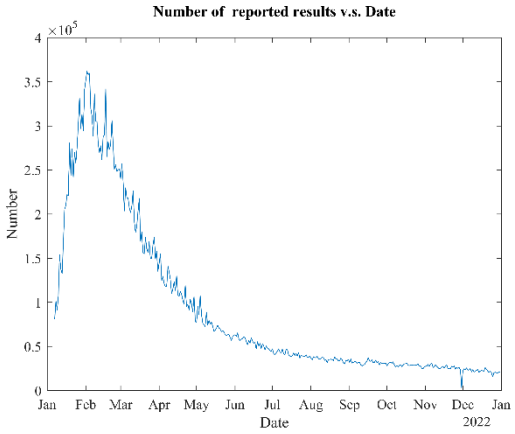


Figure 7: Number of reported results vs. Date

After data cleaning, and based on equation (4.2), the distribution of the number of reposts per day in the original dataset is shown below. We used 328 data for training and the last 30 data for testing. One of the results of the iterative process is shown in Figure 8.

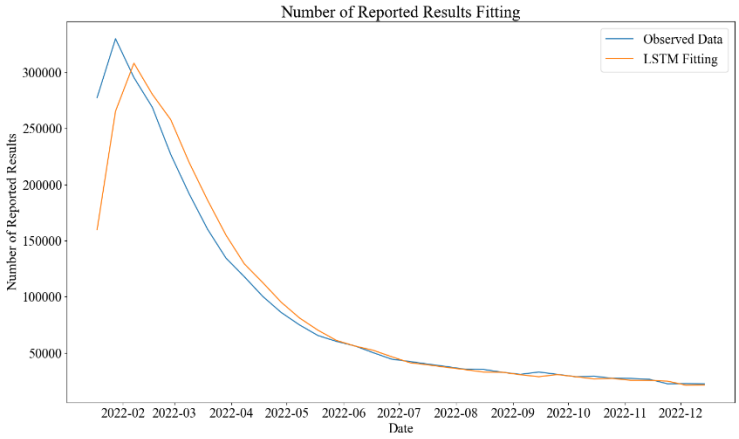


Figure 8: Number of reported results fitting

From the picture it is clear that:

a. The change in the number of reports submitted per day tends to increase and then decrease.
b. The final number of reports appears to converge on a stable value.

With this conclusion in mind, we can assume that the number of daily reports in 2023 should not vary much. We feed the last 30 historical data into the model, let the model predict the new data and then iterate to get the final predicted value of the number of reports on 1

March 2023.

When we do this several times, we find that the predicted values are distributed in this way.

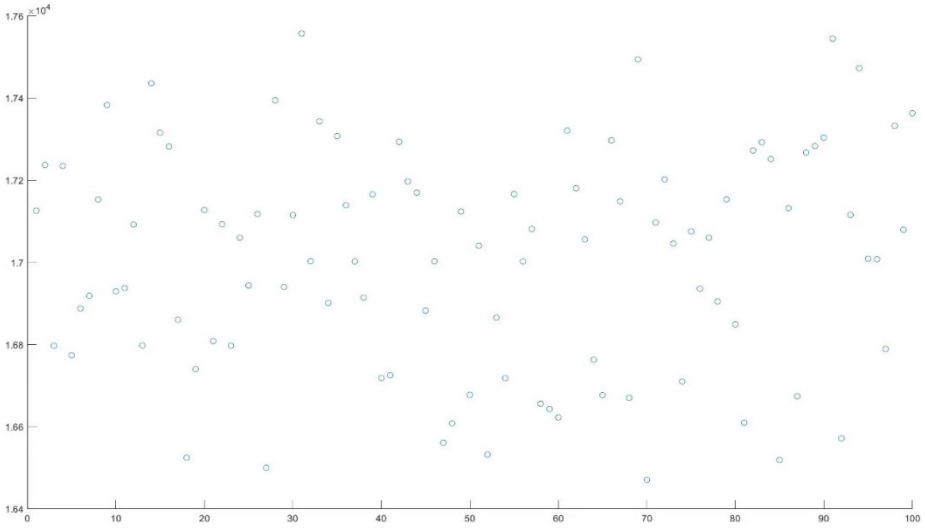


Figure 9: Distribution of predicted values

Combining this with the assumption that the predicted values follow a t-distribution, we chose the 95% confidence interval as the prediction interval, and we ended up with a prediction interval of [16000, 19000].

4.2.2 Comparison Results with Other Models

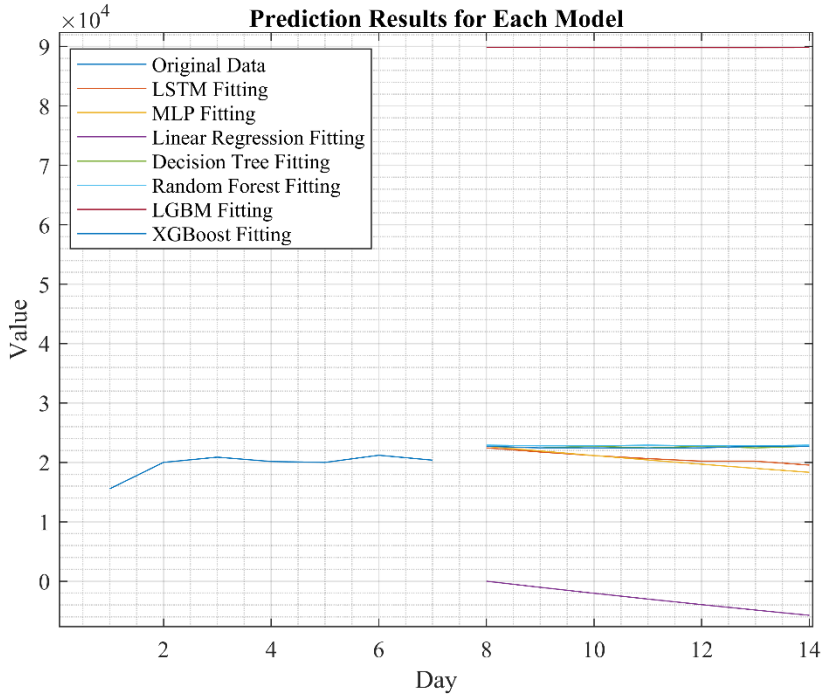


Figure 10: Prediction Results for Each Model

We used other models for training, and the performance of each model in predicting the last 7 bars of the original data is as follows.

Table 3: Prediction results for each model

Indicator	LSTM	MLP	Linear Regression	Decision Tree	Random Forest	LGBM	XGBoost
MAPE	0.54	1.42	2.09	1.45	1.44	0.76	1.45

To make it more intuitive, we compare their predictions for one week of data, as in Figure 10 and Figure 11.

As can be seen from Figure 10, the results obtained using the linear regression and LGBM models were very different from the original data, so we focused on the prediction results of the other models. As can be seen from Figure 11, the results predicted using the LSTM model generally follow the trend of the original data, while the predictions obtained using MLP converge faster, while the results obtained using Decision Tree, Random Forest and XGBoost generally converge slower.

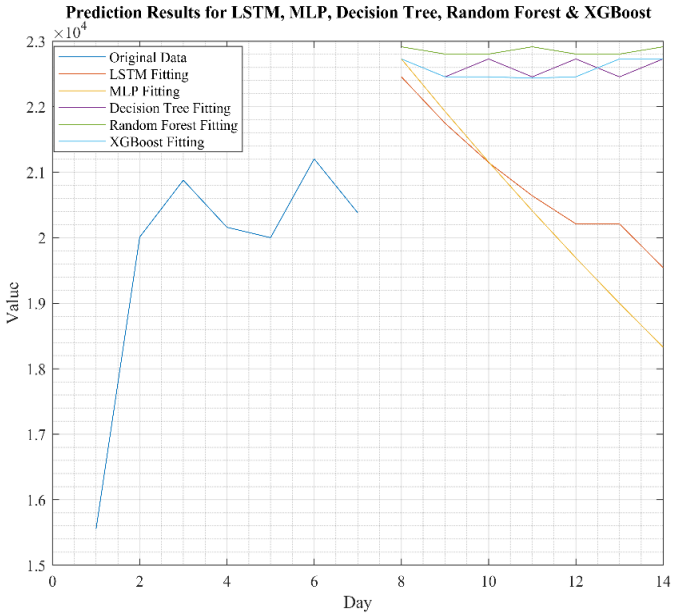


Figure 11: Prediction Results

So, in general, it is more advantageous to use the LSTM model to predict future values. The reason why the MAPE error of the LSTM is so large is that the error is larger when fitting the earlier part of the original data, as shown in Figure 8, but it performs better when fitting the later data, so it is a very good choice for predicting a longer moment later.

5 GRNN Model for Predicting the Distribution of Word Guess Counts

GRNN was proposed as a variant of the RBF ANN by Specht [5] in the 1990's. Based on non-parametric regression, GRNN performs Parzen non-parametric estimation with the sample data as the posterior condition and calculates the network output according to the maximum

probability principle. [6]

Due to its strong nonlinear mapping capability, flexible network structure, high fault tolerance and robustness, GRNN has been widely used in various fields such as signal processing, structural analysis, education, energy, food, pharmaceuticals, finance and biology. We therefore decided to use this model to solve the dependencies between word attributes, data and percentage distribution of scores.

5.1 The Establishment of GRNN Model

5.1.1 Relationship between Word Attributes and Score Percentages

In combination with the previous hypothesis, we consider firstly the attribute of a word, i.e. the frequency of each letter in the word, and secondly we believe that the length of the word may also have some influence on the guessing of the word, because if the word is longer, the probability of a correct guess will decrease geometrically according to probability theory (the multiplication principle of probability). However, as the dataset used in this paper is cleaned and all words are of length 5, the relevance of the length attribute may not be obvious. In such a case, to simplify the calculation, we can replace the frequency by the number of occurrences.

Since we need to take into account more factors (such as the number of occurrences of each letter) and to simplify our calculations as much as possible, we choose multiple linear regression to check if there is a linear dependence between them.

Let Y be the distribution matrix of scores obtained by weighted average of the percentage of guesses, α (α = {a-z}) be the frequency matrix of occurrence of letter α, W be the coefficient matrix, and L be the word length matrix. Then, for each word with a guess score y_i, we would like to have.

$$y_i = w_{ai}a_i + w_{bi}b_i + \cdots + w_{zi}z_i + w_{li}l_i + \zeta = \zeta + w_{li}l_i + \sum_{\alpha=a}^{z} w_{\alpha i}\alpha_i \tag{1}$$

To explore the relationship between them, we choose the ordinary least squares (OLS) method to calculate the matrix W and the constant ζ. That is, we require to obtain.

$$\min f(\alpha) = \sum_{i=1}^{m} L_i^{\,2}(x) = \sum_{i=1}^{m} [y_i - f(\alpha_i, w_i)]^2 \tag{2}$$

Using the OLS.fit() method in the statsmodels.api module in Python, we get the results shown in Appendix 1. We can conclude that there is indeed a relationship between the frequency of letter occurrences and the distribution of scores.

5.1.2 Model for Predicting the Distribution of Word Guess Counts

From the relationship we obtained earlier, we can fit the date sequence to the word letter frequency sequence by GRNN with the proportion of guesses. See the figure below for the structure of the GRNN we used.

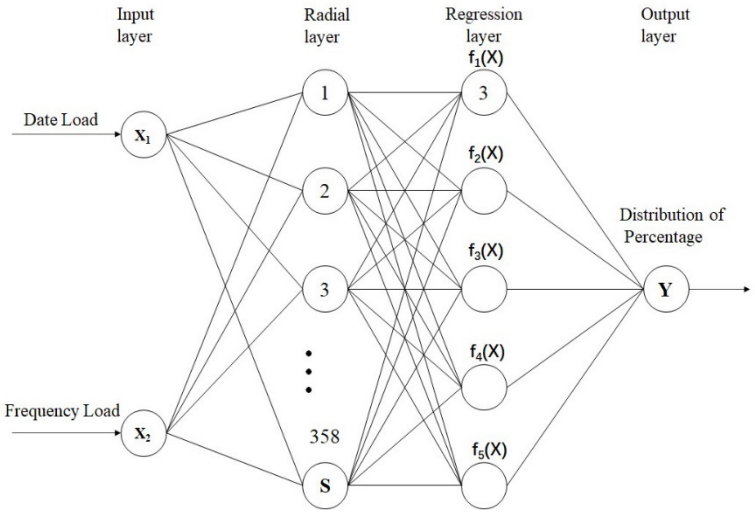


Figure 12: GRNN architecture used for the predictions study

The above network structure is based on the following mathematical principles.

The observed value of the random variable x is known to be x_0 and the regression of the random variable y with respect to x is

$$E(y \mid x_0) = y(x_0) = \frac{\int_{-\infty}^{0} y f(x_0, y)\,dy}{\int_{-\infty}^{0} f(x_0, y)\,dy} \tag{3}$$

$y(x_0)$ is the predicted output of y when the input is conditioned on x_0. Applying Parzen nonparametric estimation, the density function $f(x_0, y)$ can be estimated from the sample data set $\{x_i, y_i\}_{i=1}^{n}$ by the following equation.

$$f(x_0, y) = \frac{\sum_{i=1}^{n} e^{-d(x_0, x_i)} e^{-d(x_0, x_i)}}{n(2\pi)^{(p+1)/2} \xi^{p+1}}$$
$$d(x_0, x_i) = \sum_{j=1}^{p} [(x_{0j} - x_{ij}) / \xi]^2, d(y_0, y_i) = (y - y_i)^2 \tag{4}$$

Substituting equation (4) into equation (3) and simplifying, we get

$$y(x_0)=\frac{\sum_{i=1}^{n} y e^{-d(y_0,y_i)}}{\sum_{i=1}^{n} e^{-d(x_0,x_i)}} \tag{5}$$

We implement GRNN based on the MATLAB environment using the newgrnn function and simulate the control using a BP neural network without the date sequence as a variable. Since BP neural network has strong self-learning, adaptive and generalizing ability, it is also a good choice.

5.2 Results

5.2.1 GRNN-based Prediction Results

Our model is based on equation (5), and after the previous preparation and the corresponding data manipulation, we obtain the following results.

Table 4: Prediction results for distribution of word guess counts by GRNN

		Percent in						
Date	Word	1 Try	2 Tries	3 Tries	4 Tries	5 Tries	6 Tries	7 or More Tries (X)
3/1/2022	eerie	1%	3%	23%	39%	24%	9%	1%

5.2.2 Comparison Results with Another Model

If we use BP neural network for simulation, the results will be distributed as follows.

Table 5: Prediction results for distribution of word guess counts by BP

	Percent in						
Word	1 Try	2 Tries	3 Tries	4 Tries	5 Tries	6 Tries	7 or More Tries (X)
eerie	0.58137445%	8.08008622%	24.72970571%	31.8133658%	22.54581941%	10.16208832%	1.70634877%

Comparing Table 4 with Table 5, we find that

a. The proportions of the distributions obtained in the two ways are not very different, so they can be mutually verified as having a certain degree of confidence.
b. In addition to the minor differences between the two models, the variable of the date series has some influence on the final results.

6 K-means-DNN Model for word difficulty classification

For a hierarchical variable such as the difficulty of a word, we need a hierarchical clustering model that can adequately account for the dynamics of its distribution. Recognizing that a general hierarchical analysis model cannot satisfy the requirement that the variables are independent of each other, we built a suitable hierarchical clustering model based on the K-means model and used it to cluster the dataset given by the topic.

6.1 The Establishment of Classification Model

6.1.1 Algorithm Principle of K-means

The k-means clustering algorithm is an iterative solution clustering analysis algorithm in which the data is pre-divided into K groups, then K objects are randomly selected as initial cluster centers, then the distance between each object and each seed cluster center is calculated and each object is assigned to the cluster center closest to it. The clustering centers and the objects assigned to them form a cluster. For each assigned sample, the cluster centers are recalculated based on the existing objects in the cluster. This process is repeated until a termination condition is met. The termination conditions can be that no (or a minimum number of) objects are reassigned to different clusters, that no (or a minimum number of) cluster centers change again, and that the error sum of squares is locally minimized.

Specifically, for a given sample set, the sample set is divided into K clusters according to the size of the distance between samples. The points within the clusters should be as closely connected as possible, while the distance between the clusters should be as large as possible.

Expressed as a data expression, suppose the clusters are divided into (C_1,C_2,... ,C_k), then our goal is to minimize the squared error E.

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} \left\| x - \mu_i \right\|^2 \tag{6}$$

where μ_i is the mean vector of the cluster C_i, sometimes also called the center of mass, with the expression

$$\mu_i = \frac{1}{\left| C_i \right|} \sum_{x \in C_i} x \tag{7}$$

However, it is not easy to find the minimum of the above equation directly, which is an NP-hard problem, so only heuristic iterative methods can be used.

The pseudo-code for the above description is as follows.

Algorithm
Introduce: The Process of K-means Algorithm
(1) arbitrarily choose k objects from D as the initial cluster centers; (2) repeat (3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;

(4) update the cluster means, that is,
calculate the mean value of the objects for each other
(5) until no change;

6.1.2 Principle of DNN Algorithm

Deep learning algorithms break the limitations of traditional neural networks in terms of the number of layers and the number of nodes per layer, and their training methods are very different from traditional neural networks. Traditional neural networks set the initial values of parameters randomly and use BP algorithm to train the network by gradient descent method until convergence. However, the training of deep structures is very difficult, and the traditional methods effective for shallow layers are not very useful for deep structures, and the random initialization weights are very easy to make the objective function converge to local minima, and the residual forward propagation will be severely lost due to the large number of layers, leading to gradient spreading, so the deep learning process uses a greedy unsupervised layer-by-layer training method.

The structure of a deep neural network is shown in Fig. Deep learning networks have hundreds of hidden layers that can be used to analyze problems from many different perspectives. The computing principle is as follows.

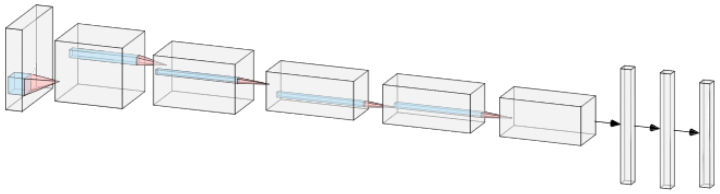


Figure 13: DNN Architecture

Assuming that there are m neurons in layer l-1, then for the output of the j-th neuron in layer l, we have.

$$a_j^{\,l} = \sigma\left(\sum_{k=1}^{m} w_{jk}^{\,l} a_k^{\,l-1} + b_j^{\,l}\right) \tag{8}$$

6.2 Results

6.2.1 K-means Clustering Results

The k-means algorithm is simple, fast converging, scalable and efficient, but it suffers from shortcomings such as difficulty in determining the number of clusters and inaccurate selection of initial clustering centers, leading to clustering results that tend to fall into local optimal solutions. Therefore, here we first perform Principal Component Analysis (PCA) on the

distribution probabilities of the number of attempts to spell words in the original dataset.

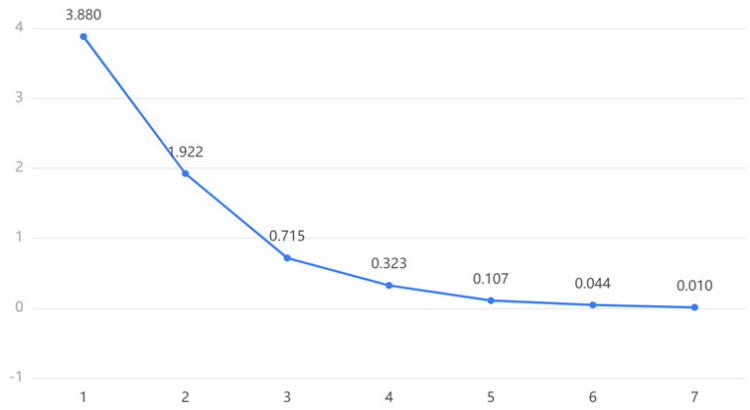


Figure 14: PCA Results

Under the assumption that the number of principal components is 1, the distribution probability of the number of spelling word attempts (1,2,3,4,5,6,X) was analyzed by principal component analysis, and according to the gravel plot of the analysis results, it can be observed that the first three components account for more than 80% of the total distribution, so there are three principal components and the original hypothesis is rejected. Therefore, we set the difficulty level of spelling words in the original dataset can be classified into three levels (1,2,3).

At this point, we have classified the difficulty level of spelling words for the dataset given in the question, and the following figure shows some of the classification results.

Table 6: Some of the classification results

	Word	Number of reported results	hard level	hard pecent
247	zesty	88974	1	0.070975791
150	youth	38381	2	0.086683515
230	yield	67115	2	0.073947702
335	wrung	294687	2	0.039105899
130	woven	33549	1	0.087424364
26	woken	23153	1	0.095020084
343	wince	241489	2	0.028365681
117	whoop	32733	2	0.090734122
339	whack	302348	2	0.033613584

6.2.2 DNN Predicted Difficulty Level

Based on the results of the K-means algorithm to classify the difficulty of the words in the dataset, we train the deep neural network with the encoding of the words as the input and the difficulty level corresponding to the input words as the output, and make a reasonable prediction of the difficulty level of "EERIE".

To solve this problem, we build the following network.

The deep learning network consists of input layer, convolutional layer, flattening layer, dense layer and classification layer. The weight parameters of the neurons are continuously updated between the different layers by feedforward computation and error backpropagation.

The network architecture for mapping this network to Keras is shown in Appendix 2.

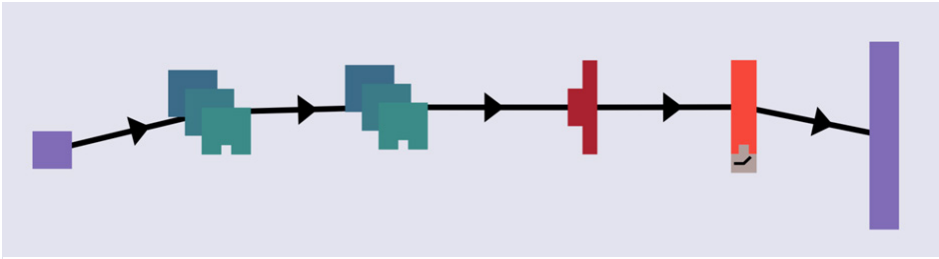


Figure 15: DNN architecture used for the predictions study

We use the frequency of different letters in the words as independent variables, the difficulty of the words as predictor variables, and a 7:3 ratio to divide the training and test sets. We also set the initialized training parameters where the training period is 300, the learning rate is 0.001, and the maximum cross-variance probability is 0.9.

We propose the following metrics to evaluate the training results of the model.

- loss: 0.08224
- accuracy: 0.8690
- val_loss: 0.07634
- val_accuracy: 0.8500
- F1: 1.0000e-01

The results show that the loss is much less than 0.5 and the accuracy is much greater than 0.5, so the learning ability and generalization ability of the deep learning network are both good, and the model can be used. The probability of the difficulty level of the given word EERIE in three types (type 1, type 2 and type 3) is 0.07548992, 0.74496204 and 0.17954797 respectively, which is the second difficulty level.

7 Sensitivity Analysis

7.1 Subjective Effects Produced by Some Parameters

Due to data limitations, the training parameters of some neural networks are determined empirically. In order to analyze the influence of these subjective factors, we have selected some important parameters in the model for slight variations and observed the resulting consequences.

- Changing the training cycle
  Based on experience, the parameter was set to 300 and the changes in the results of each model were observed when set to 270, 290, 310 and 330 respectively. The results showed no significant changes. This indicates that the stability of each model is strong.

8 Model Evaluation and Further Discussion

8.1 Strengths

The details of the algorithm are realistic and the implementation is straightforward.

8.2 Weaknesses

Our model contains many parameters whose values are more difficult to determine in most cases. Accurate determination takes more time.

8.3 Further Discussion

Our model is combined with relevant knowledge to satisfy general mathematical principles and natural laws. Perhaps our quantitative prediction model can solve the problem of economic index prediction, and the word difficulty classification model can provide a cryptographic coding system.

9 Interesting Things

9.1 Some interesting Features of the dataset

9.1.1 Normal Distributions

For the full test set, players guessing 3 attempts, 4 attempts and 5 attempts were all essentially normally distributed in the time series.

3 tries:

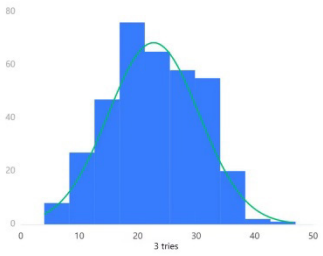


Figure 16: The normal distribution graph for 3 tries

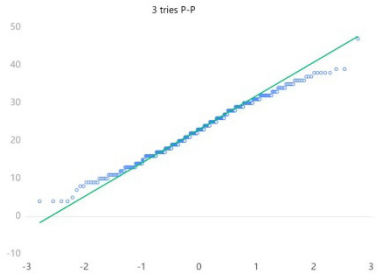


Figure 17: P-P plot of 3 tries

4 tries:

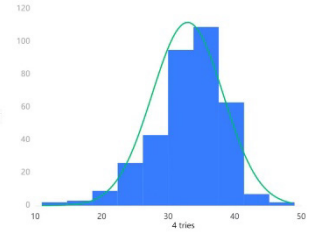


Figure 18: The normal distribution graph for 4 tries

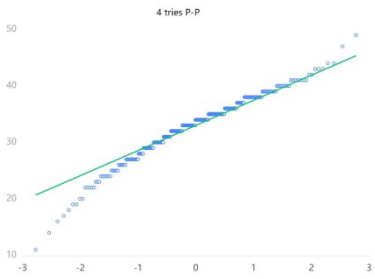


Figure 19: P-P plot of 4 tries

5 tries:

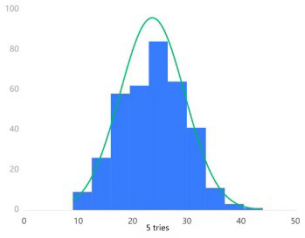


Figure 20: The normal distribution graph for 5 tries

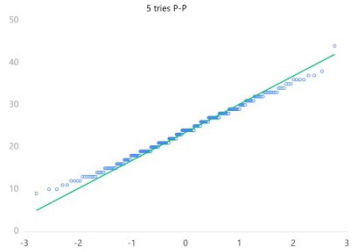


Figure 21: P-P plot of 5 tries

9.1.2 Relationship between the number of reports and the proportion of guesses

Figure 22: Heat map of the ratio of number of reports to scores

As can be seen from the graph, the correlation is not too strong, which also confirms our conclusion from Figure 3.

9.2 A Letter to the Puzzle Editor of the New York Times

Dear Editor.

How's everything going?

Thank you for creating the Wordle and sharing a large amount of real game score data collected from users with us. After our modelling and analysis, we have to say that it is a really fun puzzle game, and there is indeed something interesting behind the score distribution report. Let's take you through our team's findings.

The first thing we do is set up an appropriate model, given the characteristics of the data we have and the purpose we want to achieve, we set up a Long Short-Term Memory model.

Using the LSTM model, we made the following predictions:

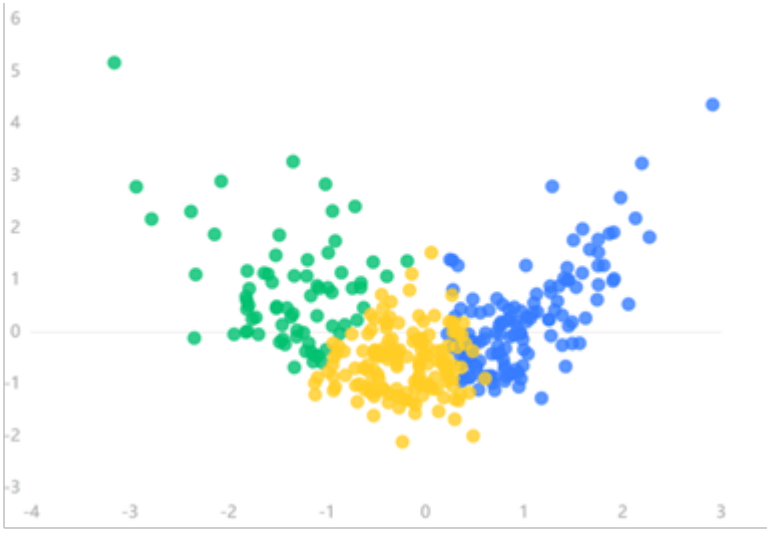
First, we predicted the number of reports on 1 March 2023 and obtained a prediction interval of [16000, 19000]. We also noticed that the frequency of the letters in the word affects the proportional distribution of the number of guesses.

Then we predicted the distribution of the number of EERIE word guesses, the results are

as follows.

Date	Word	Percent in						
		1 Try	2 Tries	3 Tries	4 Tries	5 Tries	6 Tries	7 or More Tries (X)
3/1/2022	eerie	1%	3%	23%	39%	24%	9%	1%

We also predicted the distribution of the number of times the word ERRIE was guessed and we plotted the distribution as follows.



We divide the difficulty of the words into three categories, marked with different colours, and the probability that the word ERRIE belongs to the three categories of blue, green and yellow are 0.07548992, 0.74496204, 0.17954797.

Last but not least, we discover other interesting features of this data set:

a. The 3 trials, 4 trials and 5 trials are basically normal.

b. Results and tries are not strongly correlated.

We hope our findings will help you and make a difference in the future of game design.

We all wish you a good day. If you want more details, please feel free to write to us. We look forward to your more interesting puzzle game in the future.

Team
#2300082

References

[1] Lewand, Robert Edward. Cryptological mathematics. Vol. 16. American Mathematical Soc., 2000.

[2] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[3] F. A. Gers, J. Schmidhuber and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," in Neural Computation, vol. 12, no. 10, pp. 2451-2471, 1 Oct. 2000, doi: 10.1162/089976600300015015.

[4] Yong Yu, Xiaosheng Si, Changhua Hu, Jianxun Zhang; A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. Neural Comput 2019; 31 (7): 1235–1270. doi: https://doi.org/10.1162/neco_a_01199

[5] Specht, Donald F. "A general regression neural network." IEEE transactions on neural networks 2.6 (1991): 568-576.

[6] Yifeng Chen, Liguo Shen, Renjie Li, Xianchao Xu, Huachang Hong, Hongjun Lin, Jianrong Chen, Quantification of interfacial energies associated with membrane fouling in a membrane bioreactor by using BP and GRNN artificial neural networks, Journal of Colloid and Interface Science, Volume 565, 2020, Pages 1-10, https://doi.org/10.1016/j.jcis.2020.01.003.

Appendices

Appendix 1

Introduce: OLS Regression Results of Percentage Distribution of the Number of Guesses

OLS Regression Results

Dep. Variable:	hard pecent	R-squared:	0.070
Model:	OLS	Adj. R-squared:	-0.006
Method:	Least Squares	F-statistic:	0.9177
Date:	Fri, 17 Feb 2023	Prob (F-statistic):	0.587
Time:	09:21:41	Log-Likelihood:	575.34
No. Observations:	358	AIC:	-1095.
Df Residuals:	331	BIC:	-985.9
Df Model:	27		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0336	0.150	0.223	0.823	-0.262	0.330
a	0.0560	0.061	0.917	0.360	-0.064	0.176
b	0.0566	0.061	0.930	0.353	-0.063	0.176
c	0.0574	0.061	0.943	0.347	-0.062	0.177
d	0.0750	0.061	1.232	0.219	-0.045	0.195
e	0.0575	0.061	0.950	0.343	-0.062	0.177
f	0.0600	0.062	0.961	0.337	-0.063	0.183
g	0.0566	0.061	0.929	0.354	-0.063	0.177
h	0.0510	0.061	0.835	0.405	-0.069	0.171
i	0.0577	0.061	0.952	0.342	-0.062	0.177
j	0.0668	0.066	1.008	0.314	-0.064	0.197
k	0.0564	0.061	0.920	0.358	-0.064	0.177
l	0.0551	0.061	0.909	0.364	-0.064	0.174

m	0.0567	0.061	0.929	0.353	-0.063	0.177
n	0.0536	0.061	0.876	0.382	-0.067	0.174
o	0.0523	0.061	0.861	0.390	-0.067	0.172
p	0.0559	0.061	0.919	0.359	-0.064	0.176
q	0.0413	0.066	0.631	0.529	-0.088	0.170
r	0.0536	0.061	0.882	0.378	-0.066	0.173
s	0.0629	0.061	1.029	0.304	-0.057	0.183
t	0.0648	0.061	1.065	0.288	-0.055	0.184
u	0.0698	0.061	1.150	0.251	-0.050	0.189
v	0.0650	0.063	1.024	0.306	-0.060	0.190
w	0.0555	0.061	0.907	0.365	-0.065	0.176
x	0.0630	0.064	0.990	0.323	-0.062	0.188
y	0.0769	0.061	1.262	0.208	-0.043	0.197
z	0.0706	0.065	1.083	0.280	-0.058	0.199
len	-0.0497	0.053	-0.940	0.348	-0.154	0.054

Omnibus:	695.455	Durbin-Watson:	1.660
Prob(Omnibus):	0.000	Jarque-Bera (JB):	615272.976
Skew:	12.30	Prob(JB):	0.00
Kurtosis:	204.313	Cond. No.	632.

Appendix 2

Introduce: The Architecture of DNN

Model: "sequential_8"

Layer (type)	Output Shape	Param #
conv1d_24 (Conv1D)	(None, 13, 4)	16
conv1d_25 (Conv1D)	(None, 7, 8)	200
conv1d_26 (Conv1D)	(None, 4, 16)	1168
flatten_8 (Flatten)	(None, 64)	0
dense_8 (Dense)	(None, 3)	195

Total params: 1,579
Trainable params: 1,579
Non-trainable params: 0