Problem Chosen	2023 MCM/ICM Summary Sheet	Team Control Number
C		2301831

Evaluation and Prediction of Wordle Difficulty Based on the Tree Algorithm

Summary

Wordle game was widely popular in 2022. The characteristic game mechanics and the features of puzzle and leisure have attracted many netizens to play it. It is worth noting that the distribution of users' scores differs for each day of the puzzle. In this paper, our group analyzed the frequency of each letter of the five-letter word to find out the relationship between the difficulty of the game and the frequency of use.

Preliminary Preparation: We retrieved and invoked a large number of datasets on five-letter words, from which we mined and filtered out features related to difficulty. Finally, we decided to focus on the usage frequency of each letter of the five-letter words as the core of the study. Meanwhile, we cleaned the dataset and eliminated the unidentifiable and abnormal data.

For Problem 1: Considering the change in the number of people using the hard model, we used a time series model (ARIMA) for fitting and prediction. The predicted value was 20564 and the prediction interval was [17480, 24191]. Considering the relationship between word attributes and score percentages, we tried factors such as lexical category, word frequency, and the number of repeated letters, using a linear correlation model to make a judgment. Finally, we concluded that there is a strong positive relationship between the frequency of letter usage and the percentage of scores.

For Problem 2: We started from the frequency of letter usage obtained from the first question and combined the regression tree with the Boosting algorithm to obtain a scientific prediction model. We trained and evaluated the existing data set, and finally predicted the percentage of each score of EERIE as [0, 3.9, 22.55, 38.24, 25.49, 8.82, 1.06].

For Problem 3: To find out the difficulty indicator, we evaluated the difficulty level of a word by using a weighted sum of the percentages of the scores. Through observation and data testing, the difficulty index was assumed to be normally distributed among all words, and thus difficulty levels A, B, C, and D were classified from hard to easy. By using the decision tree model, the EERIE difficulty level was obtained as D after training with the dataset, and there was a large correlation between the difficulty level and the usage frequency of the letters, especially for the third and fourth letters.

For Problem 4: We encountered a number of difficulties as well as interesting phenomena in the process of exploration and modeling. We analyzed the percentage of difficult mode players over time and found that the number of players who wanted to challenge the difficult mode was increasing in percentage over time. We found that the frequency of daily use of the puzzle words and the difficulty did not have too much correlation. We also examined the influence of the distribution of vowel letters in words on word difficulty.

Finally, we summarized the advantages and disadvantages of the created models. ARIMA model, Boosting model, Regression Tree model, and Decision Tree model all can be well applied in the context of this question. However, due to certain reasons of the data, there may be some errors. In the end, we wrote a letter to the Puzzle Editor of the New York Times to present our model and results.

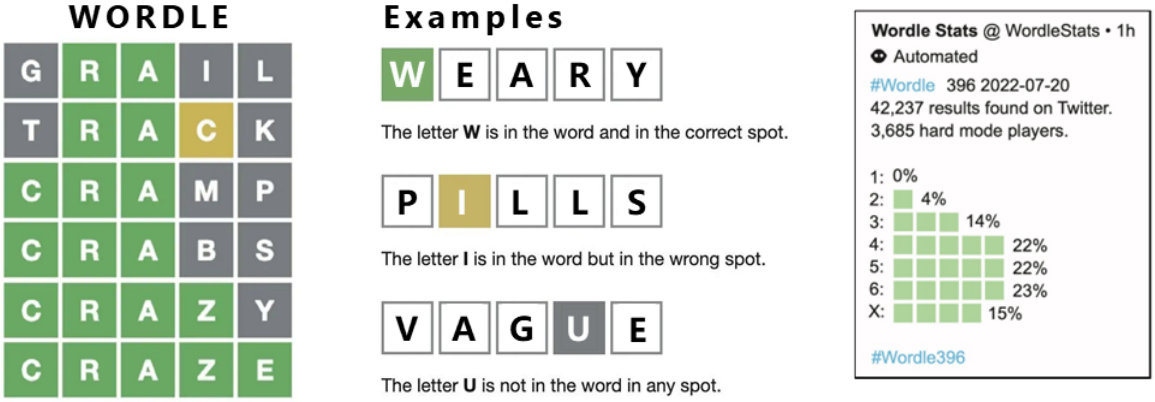
Keywords: ARIMA; Regression Tree; Boosting Model; Decision Tree; Wordle

Contents

1 Introduction

1.1 Problem Background

Wordle is a puzzle game that requires players to guess a five-letter word in six or fewer tries. Each guess must be of an actual word. If the letter is in the word and in the correct location, the tile will turn green. If the letter is in the word but not in the correct location, the tile will turn yellow. If the letter is not in the word, the tile will be gray. Players can choose either regular mode or hard mode to play. Regular mode has no additional restrictions based on the rules, while hard mode requires the player to use the correct letter (tile is yellow or green) in subsequent guesses after finding it. The example in Figure 1(a) was played in hard mode.



(a) An Example Solution of Wordle Puzzle[1] (b) Report for Jul 20,2022[2]

Figure 1: Background Brief of Wordle

To better analyze users' scores, MCM generated daily reports from January 7 to December 31, 2022, including the date, contest number, word of the day, the number of people reporting scores that day, the number of players on hard mode, and the percentage of times players spent guessing the word. The example in Figure 1(b) was the distribution of reported results for July 20, 2022.

1.2 Restatement of the Problem

- The number of reported results vary daily. Develop a model to explain this variation and create a prediction interval for the number of reported results on March 1, 2023. Analyze how certain attributes of the word affect the percentage of scores reported that were played in hard mode.
- For a given future solution word on a future date, develop a model to predict the distribution of the reported results and analyze the uncertainties of the model and predictions. Then give a specific example of the prediction for the word EERIE on March 1, 2023. Analyze the credibility of the prediction result.
- Develop a model to classify solution words by difficulty. Identify the attributes of a given word that are associated with each classification. Then classify the example word EERIE and discuss the accuracy of the classification model.
- List and describe some other interesting features of this data set.

1.3 Our Work

We totally built four models to solve the problems. They are ARIMA Model for problem 1.1, Linear Fitting Model for problem 1.2, Regression Analysis Model for problem 2, and Decision Analysis Model for problem 3. We implemented all models with the programs and analyzed the results in a comprehensive manner. Our work mainly includes the following:

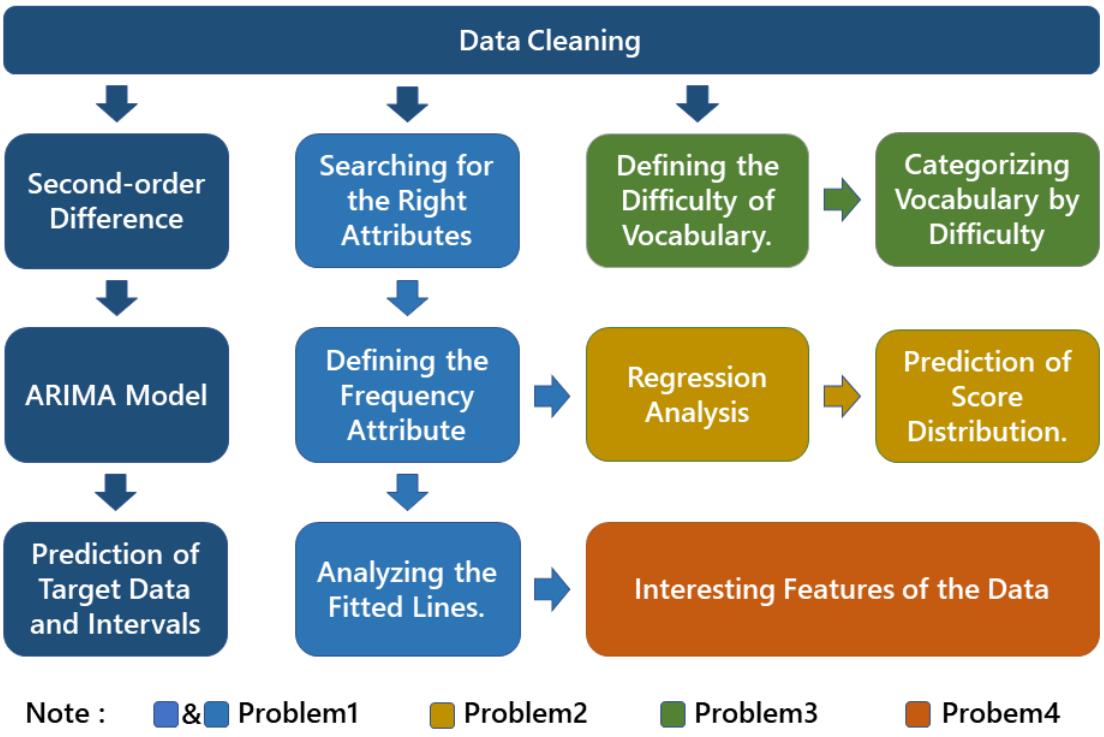


Figure 2: Model Overview

2 Assumptions and Justifications

- **Assumption 1:** Wordle always works smoothly and well.
  Justification: For the sake of authenticity, we do not exclude that Wordle has problems on some days (bad points in known data). But we assume that most of the time, especially in the future, it always works smoothly and well. Only a good and stable operation of Wordle can guarantee the reasonableness of known data and the predictability of future data.

- **Assumption 2:** Each person's word-guessing strategy remains the same.
  Justification: If the user's word-guessing strategy changes, the results will be mixed with subjective factors and not only influenced by the words. Our predictions will be unscientific. Therefore, we assume that the users' guessing strategies remain unchanged.

- **Assumption 3:** The frequency of people using five-letter words does not change with time.
  Justification: If the frequency of a five-letter word is constant, the frequency of the letters in the five positions is also constant. Then we can consider frequency as a property of a word and classify words by some criteria.

3 Problem I: Number Prediction and Attribute Analysis

3.1 Time Series Forecasting Model

3.1.1 Data Cleaning

The attached data file, like most real-world datasets, may contain some data entry errors. Since the number of reported results varies roughly according to some pattern, we use a third-order polynomial to fit the data and remove bad points by analyzing the residuals. We numbered the dates in the data from smallest to largest as the a_3 axis. Extract the number of reported results and the number in hard mode as a_1 and a_2 axes. The fitting results are shown in Figure 3.

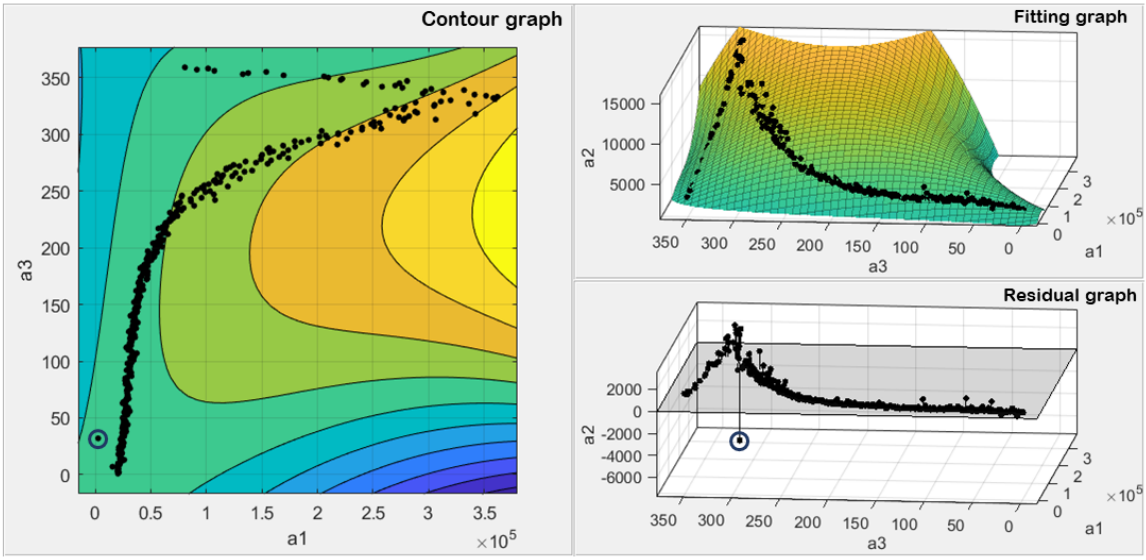


Figure 3: Polynomial Fitting and Residual Distribution

By fitting, we excluded the outlier points $\#529$ and $\#239$ (marked with circles in Figure 3). At the same time, we compared the attached words with the dictionary data[3] (by python) and excluded four-letter words and words not considered as words from the data. To ensure that the remaining amount of data is still representative, we totally cleaned out 5 bad points from 359 sets of data, accounting for 1.4%. The data we cleaned out are shown in Table 1.

Contet Number	Word	Reason
239	study	Fitting anomaly
314	clen	Four letters
473	marxh	Not a word
525	tash	Four letters
529	robin	Fitting anomaly

Table 1: The Data Cleaned Out

3.1.2 Model Building

It can be seen from Figure 1 that the total number of reported results roughly tends to increase and then decrease as the data time progresses, which is a non-stationary and trending series. Therefore, we use the method of non-stationary time series analysis to build the prediction model.

- **Second-order Difference**
  Since the number of reported results varies curvilinearly over time, we use a second-order difference model to extract the effect of curvilinear trends. Although higher-order differences theoretically work better, given the amount of data in this problem, they will also result in the loss of data points. Therefore, we use second-order difference.
  Set the reported results as $x_1, x_2, \cdots, x_n$, then the second-order difference is

$$\Delta^2 x_i = \sum_{j=0}^{2} (-1)^j C_2^j x_{i-j} \tag{1}$$

- **ARIMA Model**
  Autoregressive integrated moving average model (ARIMA model) is one of the time series forecasting analysis methods. In ARIMA (p, d, q), p, q, d each refers to the number of autoregressive terms, the number of sliding average terms, and the number of differences (orders) made to make it a smooth series. Thanks to convenient python, we can directly use the statsmodels library in python for time series prediction.
- **Confidence Interval**
  We use the arima model to obtain the predicted valuex_i' for the known data point x_i. Set the number of known data as n. Then the variance of the regression is

$$\sigma^2 = \frac{\sum (x_i - x_i')^2}{n} \tag{2}$$

  Set the number to be predicted as x_p, then the confidence interval of x_p is $x_p \pm z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$.
  Here, we can specify $\alpha = 0.05, z_{0.025} = 1.96$, which represents a 95% confidence level.

In summary, we built an ARIMA model with second-order differences. Given that the number of reported results were too large, we took logarithm of the data with a base of ten, in order to reduce the effect of large fluctuations in the data . After obtaining the predicted data and the prediction interval, the data were permuted exponentially. The model flow is shown in Figure 4.

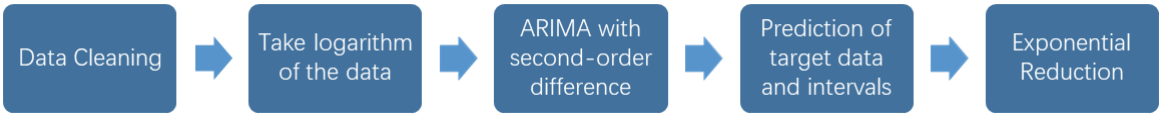
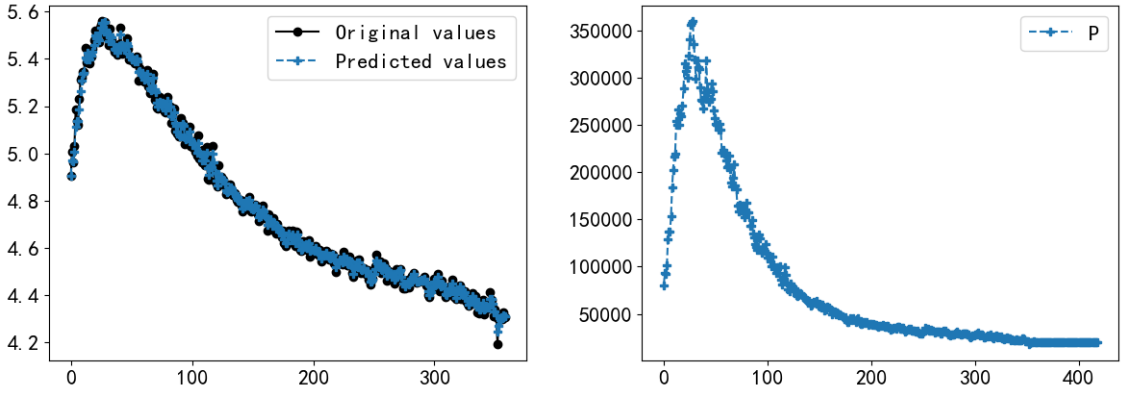


Figure 4: Time Series Analysis Flowchart

3.1.3 Results and Analysis

We implemented the model 3.1.2 programmatically (see Appendix for details). By plotting the actual values against the predicted values in the original date (as shown in Figure 5(a)), we found that our prediction works well for the known data. We also plotted all the predicted points within the original date and forecast date (as shown in Figure 5(b)), and it can be found that the number of our prediction is in accordance with the original data variation pattern.



(a) Graph of Original Versus Predicted Values (b) Graph of Predicted Data

Figure 5: Prediction Results of ARIMA Model

However , we still cannot assert that the prediction is accurate. To ensure the scientific validity of the prediction, we need to further analyze some indicators of the model (as shown in Figure 6).

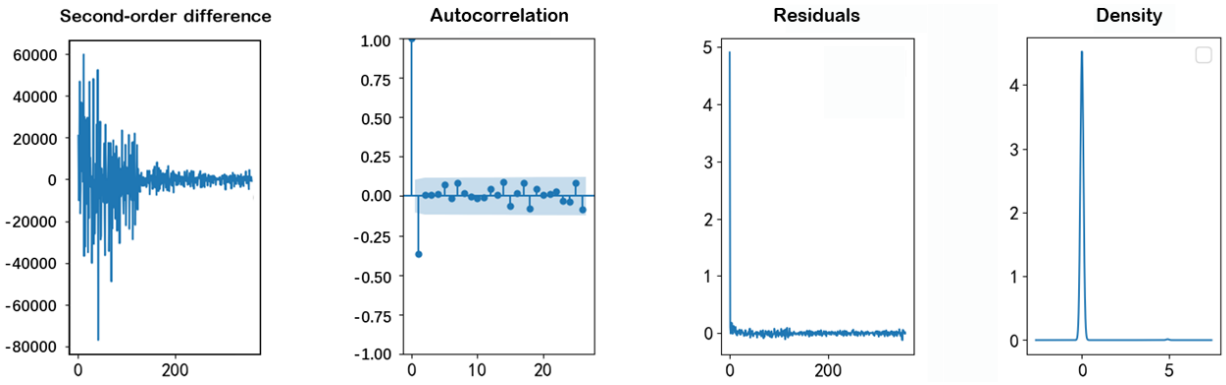


Figure 6: Indicators of ARIMA model

- **Autocorrelation:** The autocorrelations are almost all below 0.15, indicating that the existence of autocorrelation between the random error terms is weak and the model prediction is good.
- **Residuals:** The points on the residual plots are distributed around 0, random and irregular, indicating that the predictions are stable and accurate.

In addition, the images of **Second-order difference** and **Density** are both normal. Therefore, we can declare that the prediction is relatively scientific and accurate. Our prediction for March 1, 2023 is shown in Table 2. The prediction interval has a 95% confidence level. The predicted number is not in the center of the interval because we performed a logarithmic operation.

Tries	Predicted Number	Predicted Interval
March 1, 2023	20564	[17480, 24191]

Table 2: Conclusion of Problem 1-1

3.2 Linear Fit Analysis Model

3.2.1 Model Building

1. **Stats the Frequency of Word Usage**
   We retrieved data for the five-letter alphabet[3] and obtained the frequency of use of all letters in each position. Then we normalized the frequency data. Set α as the i-th of the five letters, and its frequency is denoted as $f(\alpha, i)$.

2. **Sets the Indicator of the Attributes**
   For a given five-letter word $\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5$, define its frequency attribute as

$$L(\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5) = -lg(f(\alpha_1, 1)) - lg(f(\alpha_2, 2)) - lg(f(\alpha_3, 3)) - lg(f(\alpha_4, 4)) - lg(f(\alpha_5, 5)) \tag{3}$$

   For example, the frequency attribute of 'words' is defined as

$$L(words) = -lg(f(w, 1)) - lg(f(o, 2)) - lg(f(r, 3)) - lg(f(d, 4)) - lg(f(s, 5)) \tag{4}$$

   The larger the $L(\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5)$, the less frequently the word appears.

3. **Fit the Scatter Plots**
   Plot scatter plots for different number of attempts versus frequency attributes L. Observe the distribution pattern of the images and describe the variation by fitting the data.

3.2.2 Results and Analysis

We implemented model 3.2.1 programmatically and fitted the scatter plots (see Appendix for details). The fitting plots are shown in Figure 7.

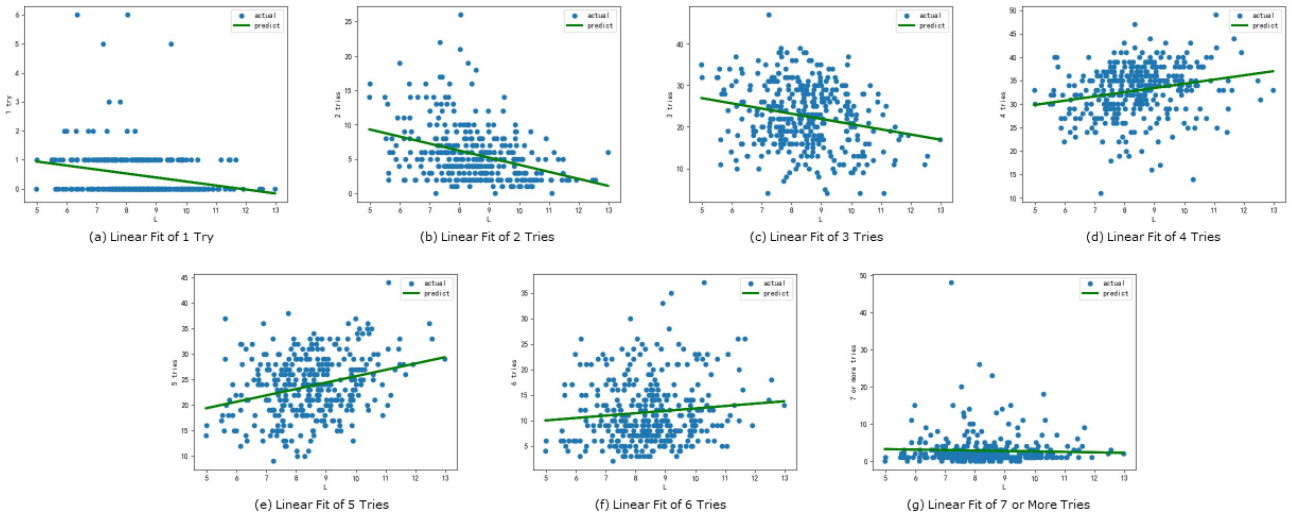


Figure 7: Scoring Percentage Fitting Graph

It is worth noting that the fit we use here is not to show how good the correlation coefficients are, but to present a more intuitive picture of the distribution pattern of the scattered data. As we can see in Figure 7, except for Figure 7(a), the rest of the scatters have a tendency to aggregate. This means that there is a correlation between the percentage distribution of the number of attempts and the frequency attribute L, and the linear fit can help us to better recognize their relationship.

Then we can analyze the slope of the fitted lines. The slopes of Fig. 7(a)(b)(c) are negative, while the slopes of Fig. 7(d)(e)(f) are positive. The slope of Figure 7(g) is also negative, but very close to 0. Thus, we can conclude that **as the frequency of word usage decreases, the frequency attribute L increases, and the number of user attempts becomes roughly more.**

In addition, we also tried other attributes of words, such as the number of vowel letters, the number of repeated letters, lexical categories of words, and so on. However, the distribution of the scatterplot is rather random and there is no clear trend. We also tried to train machine learning models to classify words by attributes, but unfortunately the test results were not good either. In the process of mining the data attributes, **we found some interesting phenomena of the data, as shown in Section 6.**

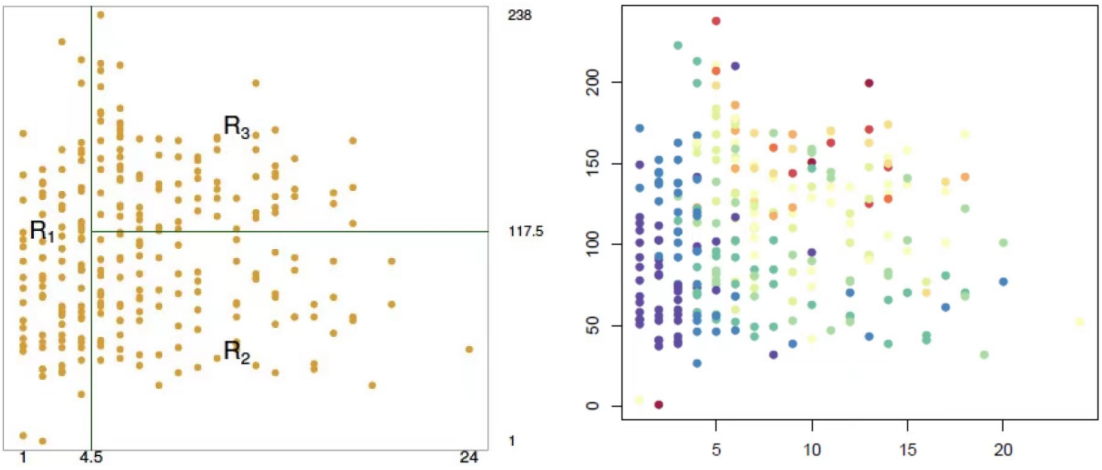
4 Problem II: Score Distribution Prediction

4.1 Model Building

From questions 1-2, we found a strong correlation between the percentage of scores and the usage frequency of each letter in a five-letter word. However, after creating the relevant graphs, we found that the percentage of scores did not correlate strongly with the usage frequency of the letter at a certain position, and the distribution was quite scattered. After comparing, we decided to use a regression tree model to portray the relationship between the usage frequency of letters and the percentage of scores.

4.1.1 Regression Tree Model

To clarify the theory of regression trees, we use Figure 8 as an example.



(a) Regression Tree Divides the Plane. (b) Regression Tree Divides a Section Again.

Figure 8: Principle of Regression Tree

In the example of the data below, the horizontal and vertical coordinates each represent a feature, and the color of the point represents the result corresponding to the feature there. The regression tree divides the plane into sections, and for samples within that section, the results are averaged over the region. As a result, we have got a simple classification prediction. For each divided section, the regression tree model continues to perform similar operations to divide the region into smaller

sections. We use the average of the sample points within that part to assign values to the data there, and then we can get a prediction model for each feature of the target data.

4.1.2 Boosting Model

The Adaboost classification model is equivalent to a forward-distributed additive model with an exponential loss function. The model f_{k+1} obtained from the $(k+1)$th round of training can be expressed as

$$f_{k+1}(x) = f_k(x) + \alpha_{k+1} g_{k+1}(x). \tag{5}$$

$g_{k+1}(x)$ is our trained weak regression function, namely the weak regression tree.

- **Optimization of Regression Tree**
  In dealing with the problem, the tree too small will lead to poor results, while the tree too large will lead to overfitting. Therefore, we try to use the idea of Boosting in integrated learning to enhance the regression tree instead of just deepening the tree.
- **Advantages of Boost Model**
  The Boosting algorithm integrates the properties of each layer of the weak regression function and evaluates the weighting to ensure that the model still performs well over different regions. Moreover, the boosting algorithm can make the error decrease gradually in the process of continuous iteration.

4.2 Results and Analysis

We implemented the regression analysis model 4.1 with the program (see Appendix for details). By training the model with the usage frequency of each letter, we finally got the prediction graphs shown in Figure 9. (The original data are represented by yellow dots and the predicted data are represented by green dots.)

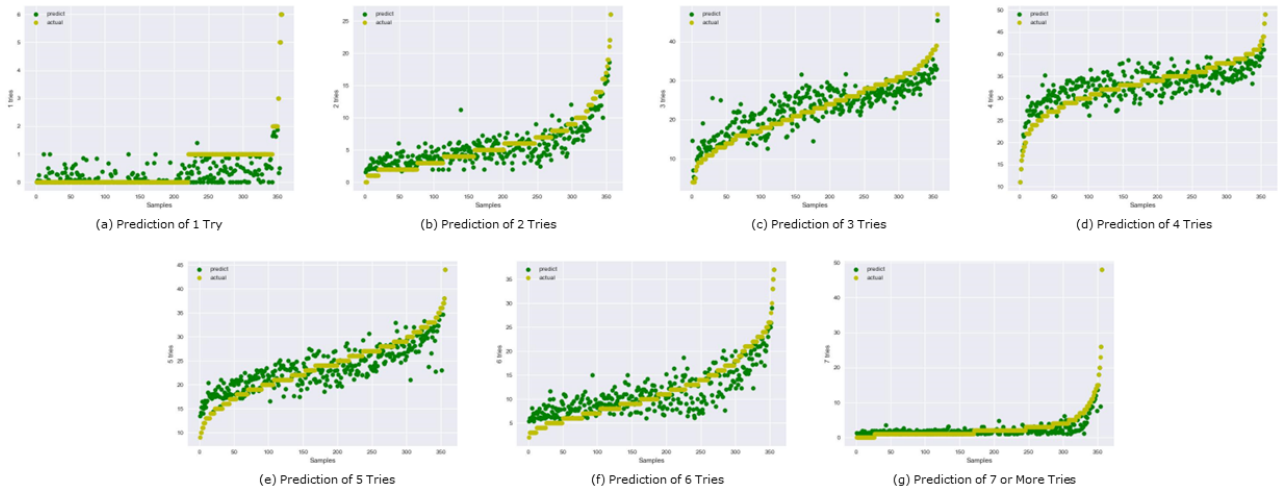


Figure 9: Predicted Scoring Percentage Distribution Graph

The images show that the predicted data points are roughly distributed around the original data, so the prediction results are relatively good. According to the letter frequency table[3], we found out the letter frequency corresponding to 'Errie' and substitute it into the model operation. The prediction results obtained are shown in Figure 10.

Number of attempts	Percentage of scores
1 try	0
2 tries	3.9
3 tries	22.55
4 tries	38.24
5 tries	25.49
6 tries	8.82
7 or more	1.08

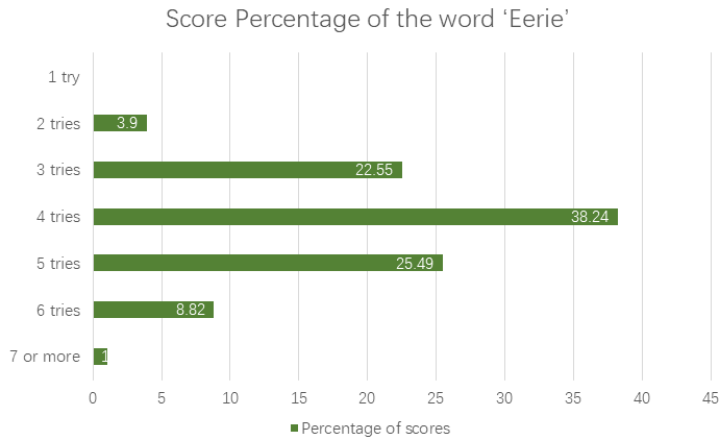


Figure 10: Prediction of the Score Percentage of Eerie

However, we still cannot ignore the gap between the predicted points and the original points. Although we tried to make improvements to the algorithm and also tried to replace the indicators for training, the prediction points never matched the original points perfectly. This may be because the indicator mining is not deep enough, the training set is too small, or the data is not representative enough. The larger the data set is, the more comprehensive the indicators we know, the more accurate the prediction will be.

5 Problem III: Vocabulary Difficulty Classification

5.1 Model Building

From Model 4.1, we can recognize the relationship between the frequency of using each letter in a word and the difficulty of word guessing. To measure the difficulty of guessing, we introduce the indicator H. The percentage of i tries for a word p is noted as y_i.

$$H(s) = 100 * (\sum_{i=1}^{6} iy_i + 8y_7) \tag{6}$$

We consider scoring the difficulty level of $H(s)$. From Figure 11, it can be assumed that the difficulty coefficients follow a normal distribution.

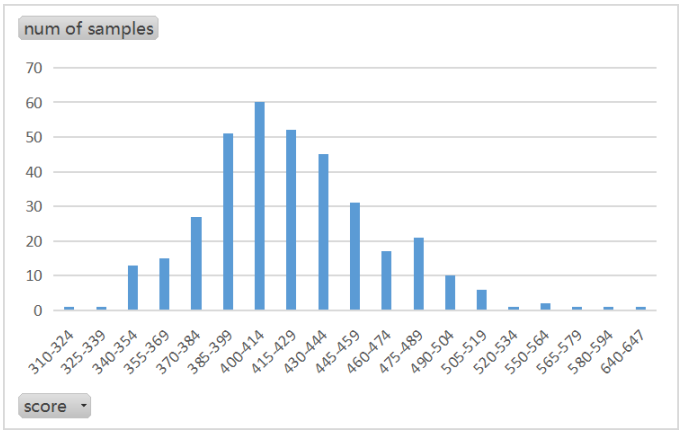


Figure 11: Plot of the Distribution of $H(s)$

Then, assume that there are as many words at different difficulty levels. That is, a five-letter word is randomly selected with equal probability of its difficulty level. From the data set, we calculated the mean and standard deviation of the difficulty levels.

$$\mu = 422.564606741573, \quad \sigma = 43.05951962648127 \tag{7}$$

Consider four levels: very hard, hard, easy, very easy.
That is, consider the node x_i, $i = 0, 1, ..., 4$, such that

$$\int_{x_i}^{x_i+1} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \, dx = \frac{1}{4}, \quad i = 0, 1, 2, 3, \quad x_0 = -\infty, x4 = \infty \tag{8}$$

Then we can get

$$A = \{s \mid H(s) > x_3\}, \quad B = \{s \mid x_3 \geq H(s) > x_2\} \tag{9}$$

$$C = \{s \mid x_2 \geq H(s) > x_1\}, \quad D = \{s \mid x_1 \geq H(s)\} \tag{10}$$

In order to classify the words better in terms of difficulty and to obtain the relevant word attributes. We consider using **decision tree algorithm.** Detailed reasons are shown below.

- The decision tree algorithm performs best on the small data sets we have (decision trees are suitable for small volumes of data).
- Decision trees are very simple and intuitive, allowing easy classification and prediction.
- Although one decision tree is prone to overfitting, we have pruning and other means to improve generalization. Decision trees are well explained in logic.

5.2 Results and Analysis

Using the decision tree model 5.1, we classified the word difficulty as shown in Figure 12.

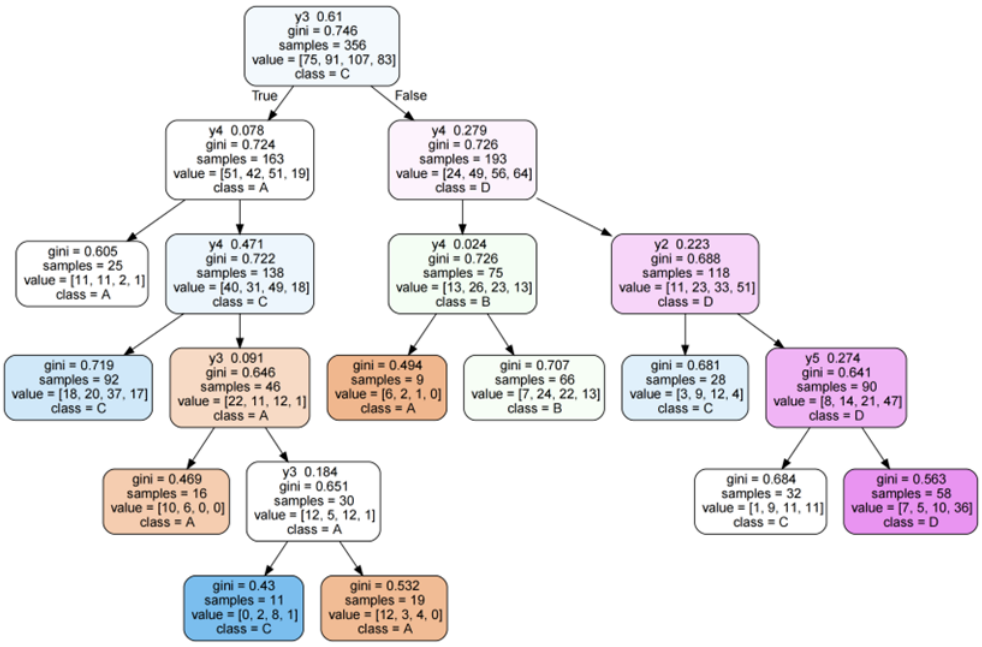


Figure 12: Diagram of Word Difficulty Classification

- In the decision tree, we can see that the less frequently the letter is used, the closer the difficulty level is to A, while the more frequently the letter appears, the closer the difficulty level is to D.
- Particularly, the usage frequency of the third letter and the fourth letter has a greater influence on the results. It can be said that the lower the usage frequency of the third and fourth letters, higher the difficulty.

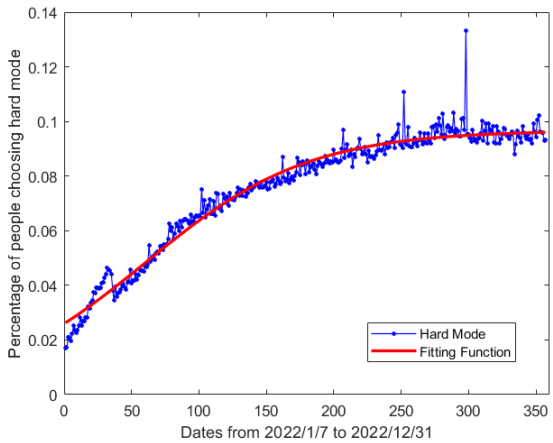
By looking up the table of letter frequencies, we got the usage frequency of letters in the corresponding positions of EERIE. We substituted the data into the model and found that **the difficulty level of EERIE is D, which is very easy** (see Appendix for details).

From the characteristics of the words, we can see that there are many e and i. These two letters are used more frequently and have a higher probability of being guessed, so the difficulty level is naturally low.

6 Problem IV: Interesting Features of the Data Set

6.1 Dare to Challenge

In real life, when we are familiar with a certain game, we tend to try a more difficult mode. Do the data in the question show such a characteristic? To figure this out, we calculated the daily selection rate of the hard mode, which is the number of people who selected the difficult mode divided by the total number of people. Then we fit it with the sigmoid function in matlab. The result is shown in Figure 13.



(a) Graph of the fitting curve

```
Fitting model:                          f(x) = a/(b+exp(-c\*x))

Coefficients  ( with 95% confidence bounds):

       a  =    0.03553               (0.03355, 0.03752)
       b  =    0.3674                (0.3479, 0.3869)
       c  =    0.01658               (0.01583, 0.01733)

Goodness of fit:

       SSE:                          0.005503
       R-square:                     0.9682
       Adjusted R-square:            0.968
       RMSE:                         0.003943
```

(b) Parameters of the Fit

Figure 13: Fitting graph of the percentage of people choosing the hard mode as the date increases

The fit is quite good. From the graph we can see that **the daily rate of choosing hard mode increases with time roughly in the form of $e^{\frac{1}{x}}$.**

6.2 Common not Always Right

We invoked the daily usage frequency lookup function WordFrequencyData from the wolfram library to sort the 356 valid data in the table from smallest to largest usage frequency. Then we were

surprised to find that the percentage of difficulty mode scores was approximately even (As shown in Figure 14). Therefore, it was concluded that **the frequency of daily use of the answer words had little effect on the distribution of scores on the day.** (Interestingly enough, the conclusion does not match our intuition. In our view, it seems that the more commonly used words are easier to guess.)

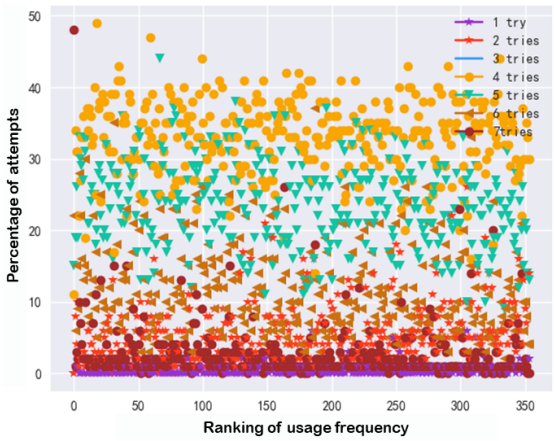


Figure 14: Percentage versus Frequency

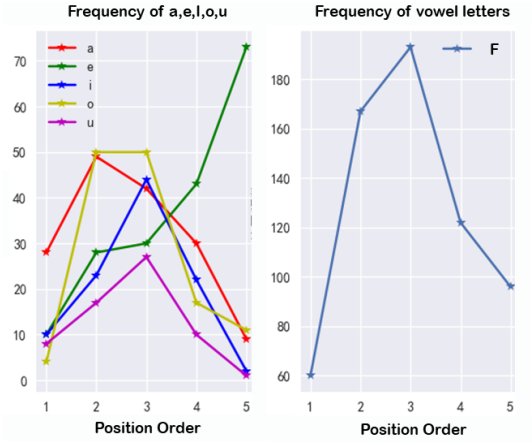


Figure 15: Distribution of Vowel Letters

6.3 Where the Vowel Letters are

We counted the distribution of vowel letters in the data words and made the following findings. (As shown in Figure 15)

- A,i,o,u are basically distributed in 1, 2, 3 positions.
- The further back the position, the more often e appears.
- Vowel letters are mostly distributed in 2,3 positions.

7 Model Evaluation and Further Discussion

7.1 Strengths

- **Time Series Analysis**
  Time series analysis is a dynamic data processing method that focuses on the interdependence of data series. The number of reports is closely related to the proximity of the day to the release date and the data from previous reports, so time series analysis is a very convincing model.
- **ARIMA Model**
  The ARIMA model is generally used when there are more than 50 observations. For non-stationary time series, the observed time series should be first differenced and transformed into a stationary time series. The reported volume of Wordle has gradually leveled off with the increase of time, so using time series analysis model is a good forecasting model.

- **Regression Tree Model**
  Through the data analysis, we found that the daily scores have a great correlation with the difficulty, so by classifying the difficulty before making predictions, we unified problem 2 and problem 3 into one model. For small data sets, we cannot fit the relationship between input and output by neural network and machine learning, and it is also difficult to classify the difficulty of words. Therefore, it is ideal to classify five-letter words into ABCD classes through a decision tree model, and the average result within each class is used as the prediction value for that class.

- **Boosting Model**
  The Boosting algorithm integrates the properties of each layer of the weak regression function and evaluates the weights, which ensure that the model still performs well on different regions. Moreover, the boosting algorithm can make the errors decrease gradually in the process of continuous iteration.

7.2 Weaknesses

- **Errors of ARIMA**
  The reported results of Wordle has a very high peak, which increases the difficulty of smoothing near the peak, and the ARIMA model will be affected to some extent.

- **Errors of Regression Tree**
  Since the dataset is not very informative, in the process of finding the attributes of words, we found that many attributes have no obvious trend in correlation with the percentage of scores itself, so the prediction bias would be quite large if the accurate function is obtained by function fitting or neural network calculation. Therefore, we selected the frequency of different letters appearing in different positions as attributes, and the difficulty division was not fine enough to make only a more rough estimate.

  At the same time, words of the type 'eerie' are relatively rare, and the number of words like this in the data set is minimal, so it may have a large chance error.

7.3 Further Discussion

To make the results more accurate, we can take the following steps.

- Use a larger data set.
- Find more word-related attribute factors.
- Use more advanced models to determine.
  For example, when the amount of data is large can be fitted by neural networks.
- Make the type of decision tree model more detailed.

8 The Letter to the Puzzle Editor of the New York Times

Dear editor,

It is our honor to be invited to analyze the results of the Wordle game report. As response to your requirement, we are here pretty glad to have the opportunity to introduce our research and suggestions to you, with the hope that it may give you some insights of the future strategies.

- **Phenomenon: The number of reported results is leveling off.**
  We used a time series model to predict the number of reported outcomes for a future period. According to our prediction, the total number of reported outcomes on March 1, 2023, is approximately 20564, which is roughly the same as the number at the end of December 2022. According to our results, we are sorry to say that the hot period of Wordle games has been over, and the total number of reported results has declined and leveled off in the region after experiencing an increase. In other words, our number of users has roughly stabilized.

- **Problem: How to retain and attract more users?**
  In order to design programs to attract more users, we analyzed the data from the results reports and found an important phenomena.

  - **The ratio of choosing hard mode is increasing.**
    By calculating the daily selection rate of the hard mode and fitting the data, we found that the daily rate of choosing hard mode increases with time roughly in the form of $e^{\frac{1}{x}}$.

  Just as in real life, when we are familiar with a certain game, we tend to try a harder mode. There are more and more Wordle players to challenge the hard mode, and we need to increase the difficulty of this game.

- **Strategy: Use lower-frequency words (especially with letter z,q,x).**
  Considering that users always place more attention on letters than on the whole words while playing this game, we gave a new definition of the frequency of words. For a word with five letters, we focused on the logarithmic sum of the frequency of each letter in its position. By plotting the scatter plot of frequency versus number of attempts, we found that the lower frequency of the word, the more attempts the users make. By making a word frequency table, we found that the letter z,q,x occurs least frequently in the words that were guessed.

- **Tool: A customized vocabulary classification and score distribution prediction tool.**
  We constructed a score percentage prediction model based on regression analysis and boosting algorithm, and a word difficulty classification model based on decision analysis. These models showed good results even with a small data set. We believe they will be of even greater value using the large amount of data from the New York Times.

We are really appreciated for this opportunity to analyze the results of the Wordle game report, and we are convinced that our proposal can be utilized in retaining and attracting more users. Please feel free to contact us for further information on the project.

Sincerely yours

MCM Team #2301831

References

[1] ”What is the Wordle that suddenly exploded abroad recently?”Accessed on February 18, 2023 at https://blog.csdn.net/sinat_33224091/article/details/122646324

[2] “Wordle Stats.” Twitter, July 20, 2022

[3] Word Frequency Data. Accessed on February 18, 2023 at https://reference.wolfram.com/language/ref/WordFrequencyData.html

[4] L. Breiman. Random forest. Machine Learning, 45:5–32, 2001.

[5] Jerome H. Friedman. Stochastic gradient boosting. Computational Statistics Data Analysis, 38(4):367–378, 2002.

[6] L. K. Hansen and P. Salamon. Neural Network Ensembles. IEEE Computer Society 1990.

[7] Lewis, David D, Schapire, Robert E, Callan, James P, Papka, and Ron. Training algorithms for linear text classifiers. pages 298–306, 1996.

[8] Dragos D. Margineantu and Thomas G. Dietterich. Pruning adaptive boosting. In Fourteenth International Conference on Machine Learning, pages 211–218, 1997.

9 Appendices

9.1 The usage frequency table of letters we made

	position1	position2	position3	position4	position5	sum
e	0.1603	0.554257	1	1	1	3.714558
t	1	0.289352	0.347469	0.283014	0.559025	2.47886
a	0.541708	0.596293	0.870765	0.275674	0.046581	2.331021
r	0.224294	0.442152	0.577404	0.457442	0.582353	2.283645
s	0.804402	0.069076	0.17977	0.407234	0.753984	2.214467
o	0.229205	1	0.767602	0.129653	0.024764	2.151224
i	0.067774	0.507917	0.983979	0.345623	0.003422	1.908715
h	0.234474	0.971041	0.123696	0.12577	0.299152	1.754134
l	0.320481	0.2687	0.208094	0.592819	0.243661	1.633755
n	0.15767	0.148106	0.288409	0.483764	0.286289	1.364238
u	0.146004	0.17773	0.641806	0.188105	0.000404	1.154049
d	0.164825	0.028522	0.207793	0.144686	0.564646	1.110472
c	0.497004	0.035916	0.084366	0.372066	0.028209	1.01756
w	0.823598	0.03615	0.061658	0.062359	0.020926	1.004691
g	0.272431	0.041412	0.267757	0.077084	0.186653	0.845338
m	0.309959	0.078547	0.188574	0.079692	0.024217	0.68099
b	0.400364	0.12269	0.082273	0.024666	0.002813	0.632806
p	0.346709	0.074656	0.091442	0.048223	3.78E-02	0.59883
f	0.383249	0.088234	0.026362	0.022342	0.022639	0.542825
y	0.083241	0.021076	0.034849	0.00405	0.260877	0.404093
v	0.083547	0.039464	0.210788	0.061888	0	0.395687
k	0.048567	0.004659	0.089582	0.076125	0.129727	0.34866
j	0.050711	0	0.017548	0.000261	5.47E-06	0.068526
x	0	0.012147	0.028021	0.000492	0.006001	0.046661
q	0.037364	0.005601	0	0	8.34E-06	0.042974
z	0.001734	0.000489	0.010158	0.006958	0.000699	0.020037

9.2 Problem 1 related codes

Problem 1-1 ARIMA model with second-order difference

```
#Question related data is stored in input3 .csv
    import pandas as pd, numpy as np
    import statsmodels.api as sm
    import matplotlib.pylab as plt
    from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

    plt.rc('axes',unicode_minus=False)
    plt.rc('font',size=16); plt.rc('font',family='SimHei')
    df=pd.read_csv('input3.csv')
    plt.subplot(1,2,1); plt.plot(df.num.diff()) #Difference
    plt.title('First-order difference')
    ax2=plt.subplot(1,2,2)
    plot_acf(df.lognum.diff().dropna(), ax=ax2,title='Self-relation')
    plt.show()

    md=sm.tsa.arima.ARIMA(df.lognum, order=(2,1,0)).fit()
    print(md.summary())

    residuals = pd.DataFrame(md.resid) #Calculate the residuals
    fig, ax = plt.subplots(1,2)
    residuals.plot(title="residuals", ax=ax[0])
    residuals.plot(kind='kde', title='Density', ax=ax[1])
    plt.legend(''); plt.ylabel('')

    zmd=md.predict(1,359)  #Comparison of raw data and predicted values
    years=[0]
    for i in range(0,359):
        years.append(i)
    plt.figure()
    plt.plot(years[1:],df.lognum,'o-k')
    plt.plot(years[1:],zmd,'P--')
    plt.legend(('Original values','Predicted values'))
    plt.show()

    print(residuals[1:])

    tmd=md.predict(1,419)

    k=[tmd[419]-1.96*0.036,tmd[419]+1.96*0.036]
    print(10**tmd[419])
    print([10**k[0],10**k[1]])
    years2=[0]
    for i in range(0,419):
        years2.append(i)

    plt.figure()
    plt.plot(years2[1:],10**tmd,'P--')
    plt.legend(('Predicted value'))
    plt.show()
```

Problem 1-2 Linear Fitting and Plotting

Figure 7 (a) as an example.

```
#Question related data is stored in Cmcm_word_tag_sp_level11.csv
```

```
    import pandas as pd
    import matplotlib.pyplot as plt
    import pylab
    from sklearn.linear_model import LinearRegression

    #Scatter plot of data distribution
    file=pd.read_csv("Cmcm_word_tag_sp_level11.csv")
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.scatter(file.iloc[:,29],file.iloc[:,22])
    plt.xlabel("L")
    plt.ylabel("7(X)try")
    pylab.show()
    print(file.iloc[:,[29,22]].corr())
    #Fitting a straight line
    model = LinearRegression() #Create linear regression objects
    x=file[['mu']]
    y=file.t7
    model.fit(x,y) #Fitting a linear regression model

    #plot
    plt.scatter(file.mu,file.t7,label = 'actual')
    plt.plot(file.mu,model.predict(x),'g-',linewidth = 3,label = 'predict')
    plt.xlabel("L")
    plt.ylabel("7 or more tries")
    plt.legend()
    plt.savefig('7_tries.jpg')
    pylab.show()
    print("intercept and coef:",model.intercept_,model.coef_)
    #View intercept and slope
```

9.3 Problem 2 related codes

Code for training.

```
#The problem-related code is stored in try7_plot.csv
    import numpy as np
    import matplotlib.pyplot as plt
    from sklearn.ensemble import AdaBoostRegressor
    from sklearn.model_selection import train_test_split
    from sklearn.tree import DecisionTreeRegressor
    from sklearn.metrics import accuracy_score
    import pandas as pd
    #Read in data

    eerie = [[0.007272044599529789, 0.2175921813446546, 0.03569171698367764,↩
                                            0.021556682611199654, 0.↩
                                           7491454429856049]]
    t = pd.read_csv("try7_plot.csv")
    t.describe()
    # Selecting the independent and dependent variables
    xs = t[['logy'+str(i+1) for i in range(5)]]
    y = t['t7']
    print(y)
    X_train, X_test, y_train, y_test = train_test_split(xs, y, test_size = 0↩
                                           .1)

```

```
    # Training data
    clf7=AdaBoostRegressor(DecisionTreeRegressor(max_depth=6), n_estimators=↩
                                                2000, learning_rate = 0.8, loss =↩
                                                 "square")
    clf7.fit(X_train,y_train)

    # Model Predictions
    y_pred = clf7.predict(X_test)
    print(y_pred)
    print(y_test)
    # accuracy

    print(str(clf7.score(X_train,y_train)))
    print(str(clf7.score(X_test, y_test)))
```

Code for prediction (1 Try as an example).

```
#Problem-related data is stored in try1_plot.csv
    x = np.linspace(1,356,356)
    t = pd.read_csv("try1_plot.csv")
    t.describe()
    # Selecting the independent and dependent variables
    xs = t[['logy'+str(i+1) for i in range(5)]]
    y = t['t1']
    y_p = clf1.predict(xm)
    print(len(y_p))
    plt.style.use('seaborn')
    plt.scatter(x,clf1.predict(xs),c='g',label = 'predict')
    plt.scatter(x,y,c='y',label = 'actual')
    plt.xlabel("Samples")
    plt.ylabel("1 tries")
    plt.legend()
    plt.savefig('1_tries_tree.jpg')
    plt.show()

    try4_eerie = clf1.predict(eerie)
    print(try4_eerie)
```

9.4 Problem 3 related codes

```
#Problem-related data is stored in Cwords2.csv
    from sklearn.feature_extraction import DictVectorizer
    import csv
    from sklearn import tree
    from sklearn import preprocessing
    import numpy as np
    import pandas as pd
    import graphviz
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import accuracy_score
    from sklearn.model_selection import train_test_split
    from sklearn.model_selection import GridSearchCV
    from sklearn.model_selection import cross_val_score

```

```
    file = open('Cwords2.csv', 'r')
    reader = csv.reader(file)
    headers = next(reader)

    featureList = []
    labelList = []

    for row in reader:
    labelList.append(row[10])
    rowDict = {}
    for i in range(11, 16):
        rowDict[headers[i]] = row[i]
    featureList.append(rowDict)
    file.close()

    for d in featureList:
        for x in d:
            d[x] = float(d[x])
    # Vetorize features
    vec = DictVectorizer()
    all_X = vec.fit_transform(featureList).toarray()
    # vectorize class labels
    all_y = [s for s in labelList]
    # Using decision tree for classification
    clf = tree.DecisionTreeClassifier(criterion='gini')
    train_X, test_X, train_y, test_y = train_test_split(all_X, all_y, ↩
                                                test_size = 0.1)
    print(test_y)
    # train the model
    clf.fit(train_X, train_y)
    # predict the test sets
    y_pred = clf.predict(test_X)

    # accuracy
    print(accuracy_score(test_y, y_pred))
    # plot
    feature_name = ['y'+str(i+1)for i in range(5)]
    dot_data = tree.export_graphviz(clf
                                    , feature_names = feature_name
                                    , class_names = ['A','B','C','D']
                                    , filled = True
                                    , rounded = True
                                )
    graph = graphviz.Source(dot_data)
    graph

    clf = tree.DecisionTreeClassifier(max_depth=6,min_samples_leaf=5,↩
                                                random_state=50)
    clf         = clf.fit(all_X, all_y)
    total_socre = clf.score(all_X,all_y)
    clf         = clf.fit(train_X, train_y)
    train_socre = clf.score(train_X,train_y)
    print("\n========Try Model============")
    print("accuracy of the whole data sets:",total_socre)
    print("accuracy of the training sets:",train_socre)

    #-------scan the best parameters--------------------------
    clf = tree.DecisionTreeClassifier()
```

```
    param_test = {
                'max_depth':range(3,15,3)
                ,'min_samples_leaf':range(5,20,3)
                ,'random_state':range(0,100,10)
    }

    gsearch= GridSearchCV(estimator=clf,
                    param_grid=param_test,
                    scoring=None,
                    n_jobs=-1,
                    cv = 5,
                    verbose=0
                    )
    gsearch.fit(train_X,train_y)
    print("\n========the result of the best parameters============")
    print("the highest score of the models:",gsearch.best_score_)
    print("the best parameters of the models:",gsearch.best_params_)

    #-------train the model with the best parameters----------------------------
    clf = tree.DecisionTreeClassifier(**gsearch.best_params_)
    clf = clf.fit(all_X, all_y)
    pruning_path = clf.cost_complexity_pruning_path(train_X, train_y)
    test_score = clf.score(test_X,test_y)

    print("\n========training results============")
    print("number of leaves:",clf.get_n_leaves())
    print("the depth of the tree:",clf.get_depth())
    print("weighs of the features:",clf.feature_importances_)
    print("\n--------accuracy of the test:----------:\n",test_score)
    print("\n------paths of CCP---------------")
    print("ccp_alphas:",pruning_path['ccp_alphas'])
    print("impurities:",pruning_path['impurities'])

    out=[]
    for ccp_alpha1 in pruning_path["ccp_alphas"]:
        i=ccp_alpha1
        clf = tree.DecisionTreeClassifier(**gsearch.best_params_,ccp_alpha=i)
        scores = cross_val_score(clf, all_X, all_y, cv=5)
        out.append(np.mean(scores))
    print("ccp_alphas:",pruning_path['ccp_alphas'])
    print("scores:",out)

    #cut the branches
    clf =tree.DecisionTreeClassifier(max_depth=9,min_samples_leaf=8,
                                     random_state=0,ccp_alpha=0.008)

    clf = clf.fit(all_X,all_y)

    test_score = clf.score(test_X,test_y)
    print("\n==============cut the branches====================:\n")
    print("accuracy of the test sets:",test_score)
    print("number of leaves",clf.get_n_leaves())

    #-------draw the tree-----------------------------
    dot_data = tree.export_graphviz(clf, out_file=None,
                        feature_names=['y'+str(i+1) for i in range(5)],
```

```
126                         class_names=['A','B','C','D'],
127                         filled=True, rounded=True,
128                         special_characters=True)
129     graph1 = graphviz.Source(dot_data)
130     print("\n==============Decision tree after cutting branches↩
                                          ====================:\n")
131     print("Decision tree after cutting branches")
132     print(clf.predict([[0.160300414,0.554257215,0.983978652,0.457442086,1]])↩
                                          )
133     graph1
```