Problem Chosen	2023 MCM/ICM Summary Sheet	Team Control Number
C		2318299

A Distributional Approach to Predicting Wordle Results

In 2022, Wordle took the world by storm. Having been named Google's Word of the Year, millions across the world began to share their results online on this simple word-guessing game. In this paper, we aim to predict the results for each day by analyzing data from WordleStats. WordleStats is a Twitter account that describes the results of Wordle via Twitter scraping, including the total number of reports and guess distributions.

In feature preprocessing, we seek to extract as much information from a word. First, we clean the data, finding numerous errors in word spelling, distribution reporting, and reports generated. When examining words to predict mean guess count, we use a count map of letters as predictors in our model, inspired by the **bag-of-words** approach from natural language processing. We also add features relating to the number of neighbors, consecutive letters, vowel placement, and the number of duplicated letters.

We divide the problem into 3 models. First, using time series, we build an **ARIMA**(1,1,1) model to predict the number of reported results over time, which was determined by examining ACF and PACF plots. After de-trending the data over time, we also find a significant negative correlation between the difficulty of words and the number of reported results. Second, we build a model to predict the distribution of guesses for a word using a multi-step process. Initially, we prove that the distribution of guesses is approximately normal using the **omnibus test of normality**. Next, we predict the mean and variance of the guess distribution using **LASSO** and **Ridge** regression. Afterward, we fit a normal distribution using these parameters and build our estimates using bucketing from intervals. We prove that this approach outperforms naive mean estimators by a significant margin by using a **Kullback Leibler Divergence** test, which calculates the difference between predictive and "true" distributions. Third, we build a classification model to divide words into difficulty classes: easy, medium, and hard. To do this, we first run **Principal Components Analysis** to reduce the data into 2 dimensions. Next, we apply **KMeans Clustering** with 3 cluster classes. We find among each class, there are distinct characteristics in terms of orthogonal neighbors, vowel placement, rare letters, and consecutive letters.

Finally, we use these 3 models to predict the outcome of "eerie" on March 1st, 2023. We find that the average number of generated results will be 20404 with a 70% prediction interval between 17556 and 23251. We believe that it will have a distribution of $[0.2\%, 2.4\%, 12.8\%, 30.7\%, 33.3\%, 16.4\%, 4\%]$ when predicting the guess distribution. Finally, using our clustering model, we classify it as a difficult word based on its repeated consecutive letters, vowel placement, and duplicate letters.

In our testing, we are able to identify multiple advantages for our approach. First, across various training and test sets, we show that our distributional model outperforms the mean model. We also use optimization hyperparameter tuning in our time series and classification models. Second, we use methods from natural language processing for feature preprocessing to extract as much information from a word. We also identify some disadvantages. Namely, we do not account for complex cases such as examining two-letter combinations or accounting for changes in the player base.

Keywords: Bag-of-Words, ARIMA Model, Normal Distribution Discretization, Ridge / Lasso Regularization, Kullback Leibler Divergence, Principal Components Analysis, KMeans Clustering

Contents

1 Introduction

Wordle is a game fundamentally about information theory, teaching people to guess potential answers conditioned on previous information in as few guesses as possible. As such, although it is a relatively new game, literature over the past year has primarily explored optimized computer and human strategies for solving a given Wordle problem, such as with a method proposed by [1]. However, there is little research available on the distribution of guesses by the entire population for a given Worldle problem. Exploring this research question can elucidate not only how the general population solves Wordle problems, but also how the general population reacts differently to different types of words and how that reaction shifts over time. This information is useful for Wordle game designers to predict the difficulty of their words as well as for researchers studying written language and linguistics to learn more about how people process and predict written words.

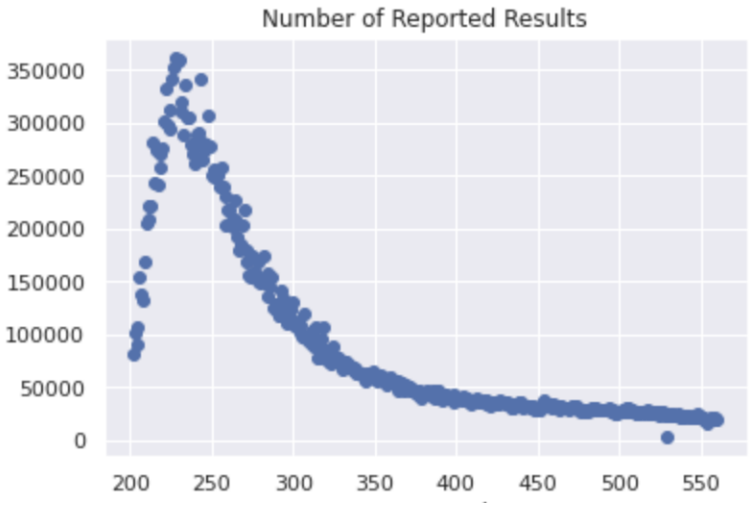
We are given a data set of Wordle words along with the distribution of guesses from January 2022 to December 2022. There are 359 observations. We are also given data such as the number of reported results on Twitter and the amount on hard mode. Within the Wordle data set, we hope to produce various models to understand the player base on Wordle a little better. We restate the tasks here for clarity and provide a general approach to each of these problems.

1. Problem 1 (Time Series): Model and predict the number of Wordle results reports on Twitter over time and understand its relationship with factors like word difficulty and percent hard mode.
2. Problem 2 (Distribution) : Model the distribution of guesses for a specified word
3. Problem 3 (Classification) : Produce a classification tool for the difficulty of a word

2 Pre-Model Exploration

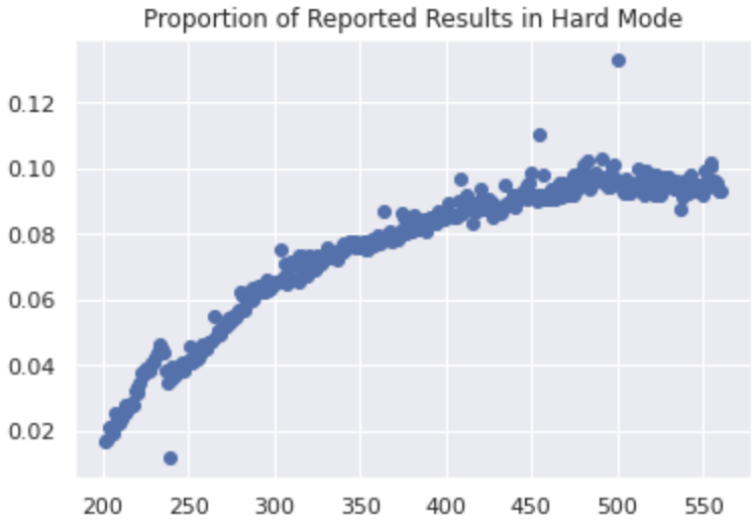
2.1 Exploratory Data Analysis

We first look at the number of reported results over time. There is a steep spike near the beginning of the data set, however, after this spike, the number of results consistently decreases.



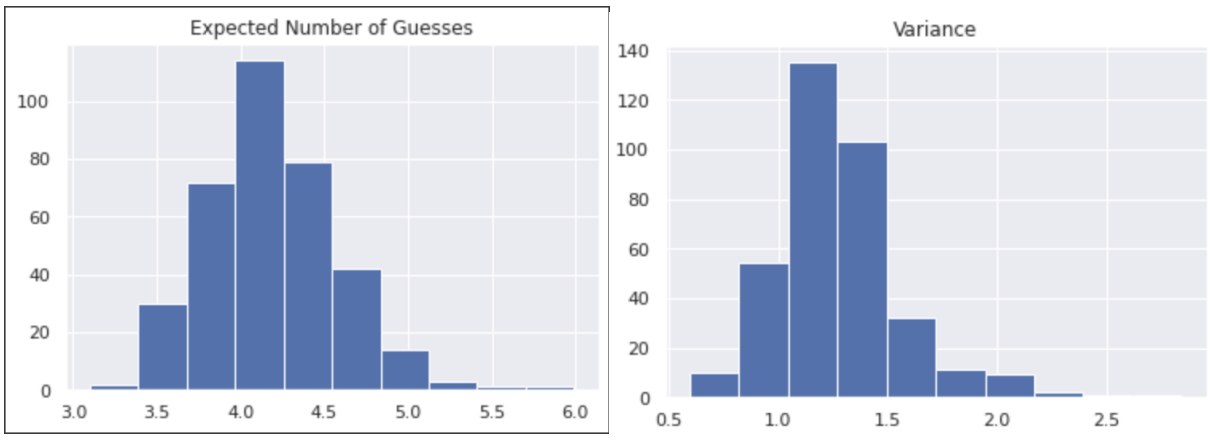
(1)

Interestingly, however, the proportion of reported results that were in hard mode increases with time. We suspect that this is because those who are better at the game continue longer, thus making up a larger proportion of the population as the total population decreases.



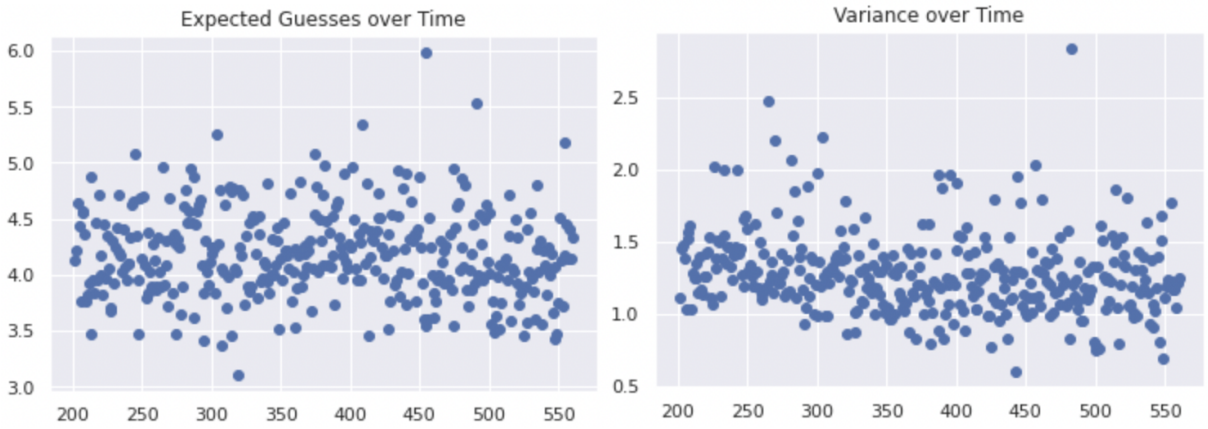
(2)

To better understand the difficulty of each word, we calculate the expected number of tries to get the word, as well as the variance in the number of tries to get the word, treating 7 or more guesses as simply 7 guesses. We observe that both are roughly normally distributed.



(3)

We also observe that neither shows a significant trend over time

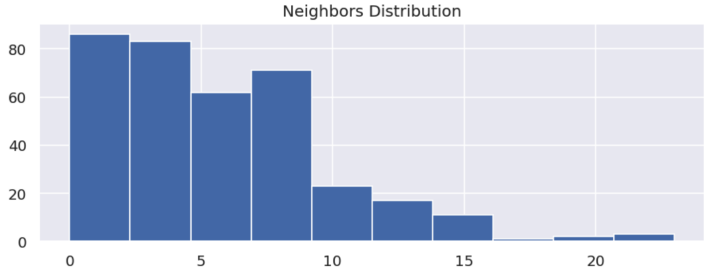


(4)

2.2 Feature Preprocessing

To ensure the viability of our data, we painstakingly check data from Twitter with the existing data set. We found some errors, including misspellings of words from ”tash” to ”marxh” to inaccurate reporting of total submitted Wordles. After cleaning this data, we scoured the literature review to examine relationships between word structure and word frequency. Namely, we sought to extract features from words to generate information about their difficulty.

[5] mentions the significance of orthogonal neighbors in word structure within guessing games like ”Scrabble” and ”Wordle”. They define orthogonal neighbors as words with the same number of letters, but a single letter changed. For example, ”hatch” and ”patch” are orthogonal neighbors. To account for orthogonal neighbors, we generate a set of 5-letter English words and count the number of neighbors within the data for each Wordle solution.



(5)

The number of orthogonal neighbors is right-skewed with the mean centered at 5.64 and a median of 5 orthogonal neighbors. Our initial belief is that having more orthogonal neighbors would increase the mean number of average guesses because there are more possible correct answers (e.g., choosing between ”hatch” and ”patch” in a Wordle guess).

Previous literature in language processing has also uncovered how human brains recognize and categorize written words concerning consonants and vowels. [6] showed that, for English words, transposed-letter non-word primes, or the swapping of two letters in the word, was only statistically significantly effective in consonant-consonant swaps and not in vowel-vowel swaps (i.e. ”ADACEMICS” vs.

"ACEDAMICS"). Thus, we examine the placement of vowels within 5-letter words. We define hard words as words where the mean number of guesses is above the 80th percentile and easy words as those with the mean number of guesses below the 20th percentile. To calculate vowel placement positioning, we calculate the distribution of I, an indicator variable with $I = 1$ if word[i] is a vowel and 0 otherwise. Overall, we find:

	count	mean	std	min	25%	50%	75%	max
easy1	72.0	0.083333	0.278325	0.0	0.0	0.0	0.0	1.0
hard1	71.0	0.126761	0.335073	0.0	0.0	0.0	0.0	1.0
easy2	72.0	0.319444	0.469533	0.0	0.0	0.0	1.0	1.0
hard2	71.0	0.718310	0.453025	0.0	0.0	1.0	1.0	1.0
easy3	72.0	0.861111	0.348257	0.0	1.0	1.0	1.0	1.0
hard3	71.0	0.295775	0.459639	0.0	0.0	0.0	1.0	1.0
easy4	72.0	0.277778	0.451046	0.0	0.0	0.0	1.0	1.0
hard4	71.0	0.323944	0.471310	0.0	0.0	0.0	1.0	1.0
easy5	72.0	0.291667	0.457719	0.0	0.0	0.0	1.0	1.0
hard5	71.0	0.267606	0.445862	0.0	0.0	0.0	1.0	1.0

(6)

The most significant difference occurred with "hard" words having more vowels in the second position and "easy" words having more vowels in the third position. To confirm our suspicions, we run an independent t-test. The t-statistic for between-group differences is defined as follows:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sigma_{\bar{X}_1,\bar{X}_2}}, \text{ where } \sigma_{\bar{X}_1,\bar{X}_2} \text{ is the pooled standard error}$$

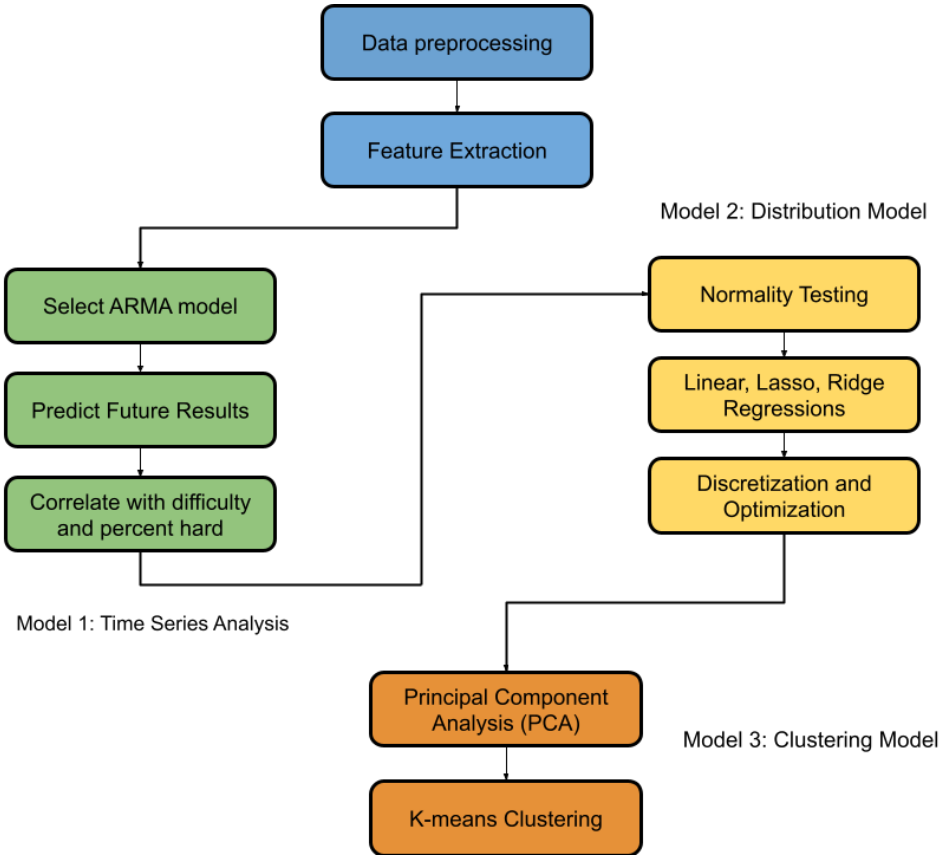
We find that under the easy2 and hard2 data, the p-value was $7.91 * 10^{-7}$. Under the hard3 and easy3 data, the p-value was $7.73 * 10^{-14}$. Thus, we encode vowel placement in the second and third positions into our models. Finally, inspired by [4], we implement a bag-of-letters approach, featurizing the words into a count map of each of the letters. This is a common feature vector in natural language processing models, described as bag of words. Through this approach, we can understand which letters increase the difficulty of guessing the Wordle word. Because we initially use a simple linear model, we add extra features such as the number of distinct letters and the number of consecutive same letters within a word. This is supported by [8] who examines information theory and the difficulty of connecting letters in Wordle guessing. Overall, we have over 31 features to examine.

Table 1: Feature List.

Feature Name	Feature Type	Description
num_letters	discrete	letter count
consec	discrete	consecutive letter count
neighbors	discrete	orthogonal neighbor count
has2	indicator	second term is vowel
has3	indicator	third term is vowel
letter	discrete	bag of letters

3 Model Setup

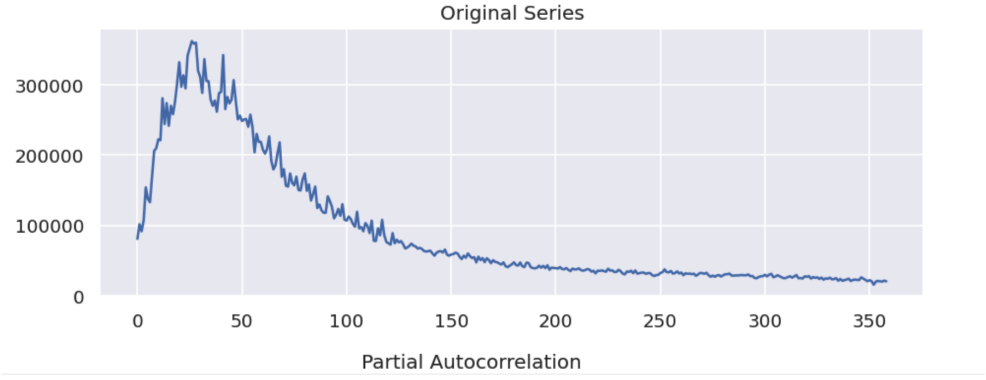
We divide our modeling into 3 distinct sections: a time series analysis to predict the number of reported scores along with the percentage of players on hard mode, a distributional model to predict distributions of guesses, and a clustering model of word difficulties to create distinct difficulty classes.



(7)

3.1 Time Series Model

The main purpose of this section is to develop a model to explain the variation of reported results daily and ultimately seek the relationship between reported results, time, and difficulty. Before establishing our model, we examine the relationship of reports over time.



(8)

There is a clear peak on February 2nd, 2022 that continuously drops over time. February marks the time when the New York Times decided to buy Wordle and add it to its platform. This is due to a drop in the fan base, as casual players leave to play other games, and Wordle's more hardcore fan base stays.

Because there is a clear pattern in the data, we decide to use the time series **ARMA** model to forecast future reported values. First, we define Y_t as times series data where t denotes the time step. The ARMA model is split into the auto-regressive and moving average portions. We can predict future values of Y. The notation $AR(p)$ is defined as

$$Y_t = \sum_{i}^{p} \beta_i Y_{t-i} + \epsilon$$

Auto-regression refers to a weighted combination of previous values to predict a current measure. Conversely, a moving average model $MA(q)$ refers to:

$$Y_t = \mu + \epsilon_t + \sum_{i}^{q} \theta_i \epsilon_{t-i}$$

Conceptually, a moving average term regresses upon white noise terms or random shocks to find future values with some mean. Therefore, combining both terms, an ARMA(p,q) model is defined as:

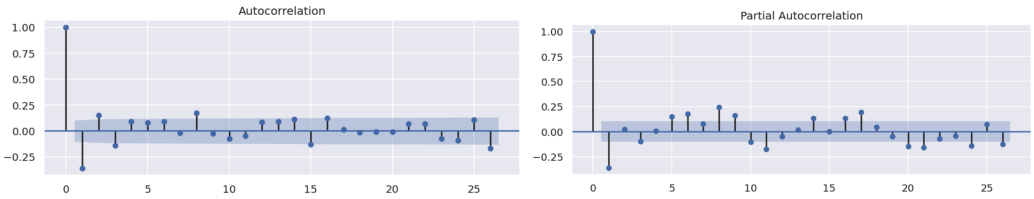
$$Y_t = \epsilon_t + \sum_{i}^{p} \beta Y_{t-1} + \sum_{i}^{q} \theta \epsilon_{t-i}$$

An important assumption of ARMA is stationarity in the times series, or $\mathbb{E}[Y|t_i] = \mathbb{E}[Y|t_j]$ where Y is time series data. Essentially, we must de-trend our data. To find stationarity, we use the Augment Dickey-Fuller Test (ADF) where the null hypothesis is the presence of a unit root in the time series.

Model	ADF Stat	P-value
After Peak	-1.71	0.42
Before Peak	-1.59	0.48

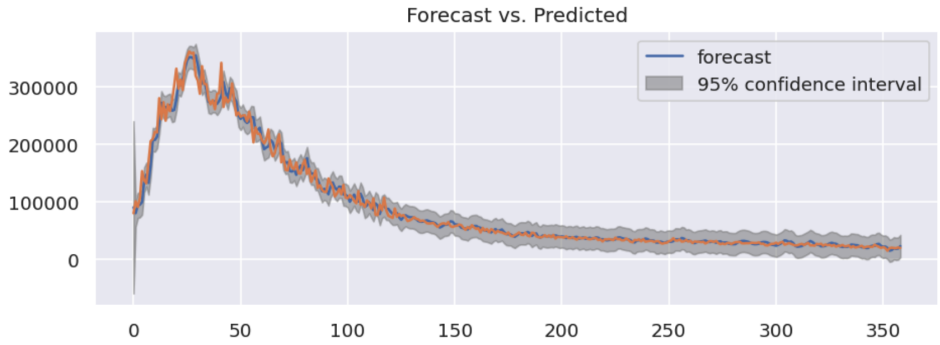
(9)

We start with different start and end points: after the peak and before the peak. We found that at both times, there was clear nonstationarity, which reduced over time. Thus, we use a differencing of 1 to make the data stationary. After this differencing, there was no longer evidence of a unit root, so our data is considered stationary over time. An **ARMA**(p, q) model takes parameters p, an auto-regressive model of order p, and q, a moving average order of q. To fit parameters, we examine an auto-correlation plot and a partial auto-correlation plot.



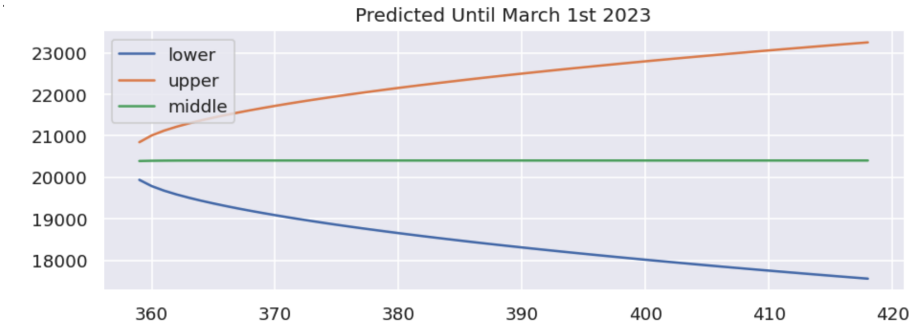
(10)

In the autocorrelation plot, we see a pattern of autocorrelation decreasing over time, we choose $q = 1$ as a significant enough threshold. In the partial autocorrelation plot, we find that lags of 1 are correlated, so we choose $p = 1$. After fitting the data based on this metric, we can compare the predicted and realized model. Therefore, we fit at **ARIMA**$(1, 1, 1)$ on the data.



(11)

We separate the data into post-peak and pre-peak, as there is a clear regime shift in the behavior of population postings. We primarily use the down-trending data in our prediction model. Using this model, we can evaluate predictions until March 1st. Match 1st lands on the 60th day of the year.

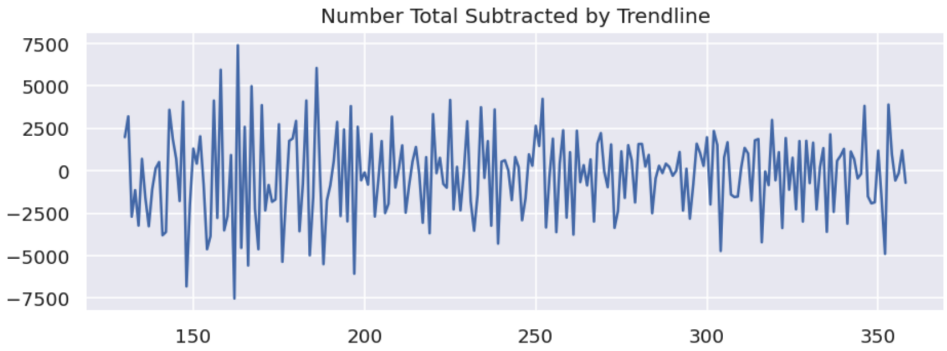


(12)

The upper and lower represent 70 percent confidence intervals. Due to the short-term nature of the model, there could be overestimates due to the recent increase. With a recent upward trend in reports, this suggests that a **prediction interval for March 1st, 2023 is** $[17556, 23251]$ **with a mean estimate of** 20404.

We want to also examine how difficult Wordle solutions impact the number of reports on Twitter. For this, we analyze two features, including *mean try* and *percent hard mode*. To do so, we must start by "de-trending" the data by removing the ARIMA estimates resulting in a zero-mean, stationary

distribution.



(13)

By de-trending the data, we can find how the average difficulty accounts for the remaining variation in the number of generated reports. Doing simple correlation tests, we find that:

Variable	Correlation
Percent Hard Mode	0.08
Mean Try	-0.309

(14)

Percent Hard Mode has a very slight positive correlation (but likely not significant) with the number of reported tests. This could make sense, as those who complete the percent hard mode are players with more interest in the game and more likely to flaunt their accomplishments. The more hardcore fan base is also likely to be active in the Wordle community, which can be correlated with hard mode. On the other hand, more casual fans of Wordle who do not play on hard mode are likely to ignore Wordle altogether if they cannot complete the Wordle. To quantify such effects, we ran a simple linear regression model. For such a model, we assume homogeneity of variance in future predictions and future relationships will reveal a similar linear relationship.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          normal_number   R-squared:                       0.107
Model:                            OLS   Adj. R-squared:                  0.101
Method:                 Least Squares   F-statistic:                     17.17
Date:                Sun, 19 Feb 2023   Prob (F-statistic):           9.04e-08
Time:                        16:43:45   Log-Likelihood:                -2921.6
No. Observations:                 289   AIC:                             5849.
Df Residuals:                     286   BIC:                             5860.
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const         1.551e+04   4212.755      3.683      0.000    7222.726    2.38e+04
percent_hard  5.355e+04   2.83e+04      1.894      0.059   -2113.025    1.09e+05
mean_try     -4933.9188    868.772     -5.679      0.000   -6643.917   -3223.920
==============================================================================
Omnibus:                       44.038   Durbin-Watson:                   2.738
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              302.842
Skew:                          -0.308   Prob(JB):                     1.73e-66
Kurtosis:                       7.977   Cond. No.                         350.
==============================================================================
```

(15)

While the R^2 of the model is certainly low (around 0.10) and AIC is large, the model reveals with statistical significance that difficult Wordle words predict a lower number of reported results. Specifically, for each 1 percent increase in the number of people completing the Wordle on hard mode, the average number of reported results will increase by 5355 when holding mean tries constant. This relates to both casual fans leaving Wordle, resulting in less reported results, and the relationship between a hardcore fanbase staying. Moreover, each increase in average try will reduce the number of reported results by 4933. This has an important implication for those attempting to optimize attendance of Wordle, showing that making fairer Wordle words could improve its decreasing trend.

Overall, we learn two major points from examining the number of reports generated and time. First, we find the number of reports generated has been on a large downhill due to decreasing popularity since its peak in February. Second, we learn that the remaining variation when controlling for the time comes from the difficulty of the word, so more difficult words result in fewer people posting in the first place. With all ARIMA models, we fail to incorporate news events and other macro factors that could shift the trend for Wordle and seek to fit a curve based on past data instead. Moreover, our linear regression model is simplistic and purely used to show a negative correlation.

3.2 Distribution Model

We seek to develop a model to predict the distribution of the reported results given a word on Wordle. For this, we extract features from the feature engineering section to inform word choices and their impact on the average guess for this distribution. Our approach is such:

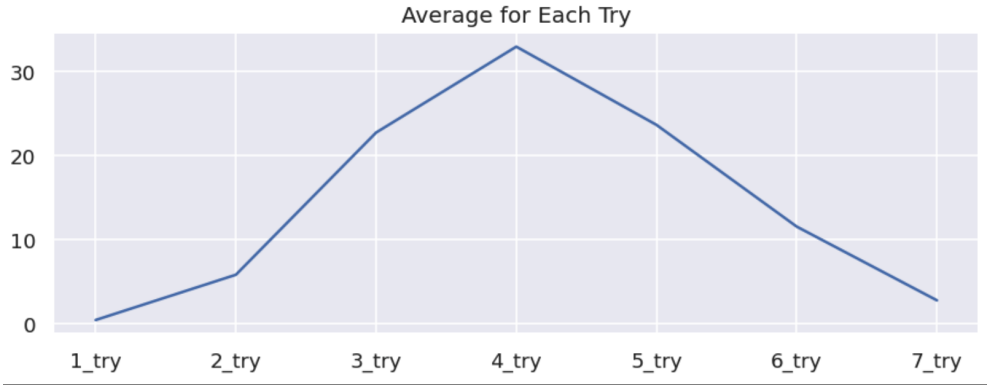
1. Assume that the distribution of reported results follows a discretized normal distribution
2. Extract features from a given word, as explained in Feature Selection
3. Predict the parameters (mean and variance) of this normal distribution using linear models (Ridge, Lasso, Least Squares)
4. Discretize the estimated normal distribution into buckets to denote the 7 choices ranging from first guess to last guess using area

We justify our assumption that the attempts per word follow a normal distribution by running an experiment on every word. Each row of data represents a density of distribution. We generate 100 data points according to this density:

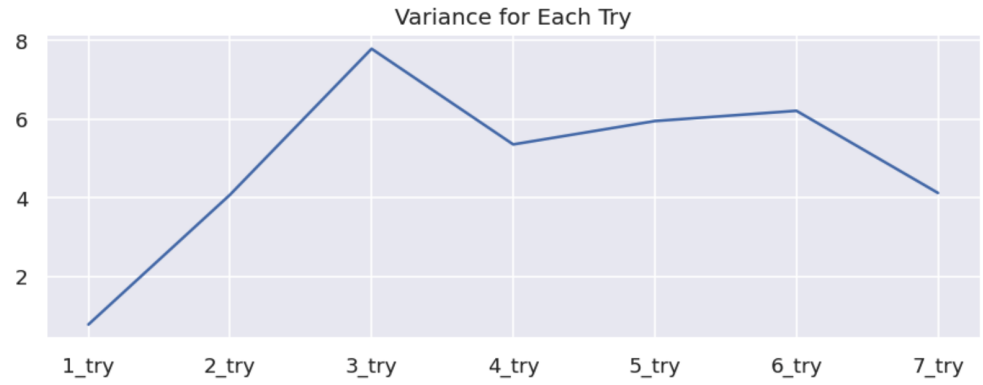
$$f_i = X + \epsilon$$

where X represents $\{1, 2, 3, 4, 5, 6, 7\}$ with the same probability as the word guess density and $\epsilon \sim N(0, 0.5^2)$ to add robust noise. For $f = [f_1, ..., f_{100}]$, we run an omnibus test for normality for each observation. Using an alpha of 0.05 for each density, we find that we rejected the null hypothesis for 5% of the words that the density followed a non-normal distribution. Therefore, we can safely assume

normality.



(16)



(17)

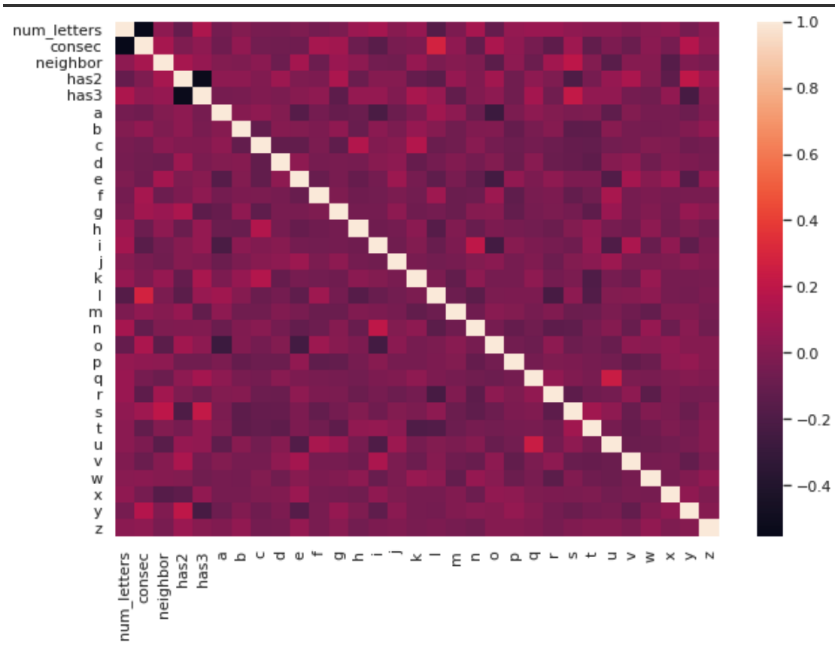
We will attempt to estimate the parameters of such normal distribution for each word using features extracted from the word via regression. The correlation between variance and mean of each try equates to 0.227, showing relatively smaller correlation. With independence, we hope to run regressions on existing variables to find their relationship in predicting both the mean and variance of this distribution. If we consider the distribution $P \sim N(\mu, \sigma^2)$ to represent the distribution of guesses before discretization. We will interpolate $\hat{\mu} = \beta_1 x_1 + ... + \beta_{31} x_{31}$ μ and $\hat{\sigma}^2 = \beta_1 x_1 + ... + \beta_{31} x_{31}$ where x represents the 31 features and β are the OLS, Ridge, and LASSO estimators for each. Thus, we will determine the distribution based on $P \sim N(\hat{\mu}, \hat{\sigma}^2)$.

Table 2: Feature List Revisited

Feature Name	**Feature Type**	**Description**
num_letters	discrete	letter count
consec	discrete	consecutive letter count
neighbors	discrete	orthogonal neighbor count
has2	indicator	second term is vowel
has3	indicator	third term is vowel
letter	discrete	bag of letters

To find the relationship among different variables, we plot a correlation heatmap to show if there are any major correlated features. There are no major correlations, but some are moderately correlated.

We must keep this in mind when modeling the results. As explained before, we hope to predict two values that parameterize the normal distribution.



(18)

Table 3: Dependent Variable List

Variable Name	Variable Type	Description
mean tries	continuous	mean of distribution
variance tries	continuous	variance of distribution

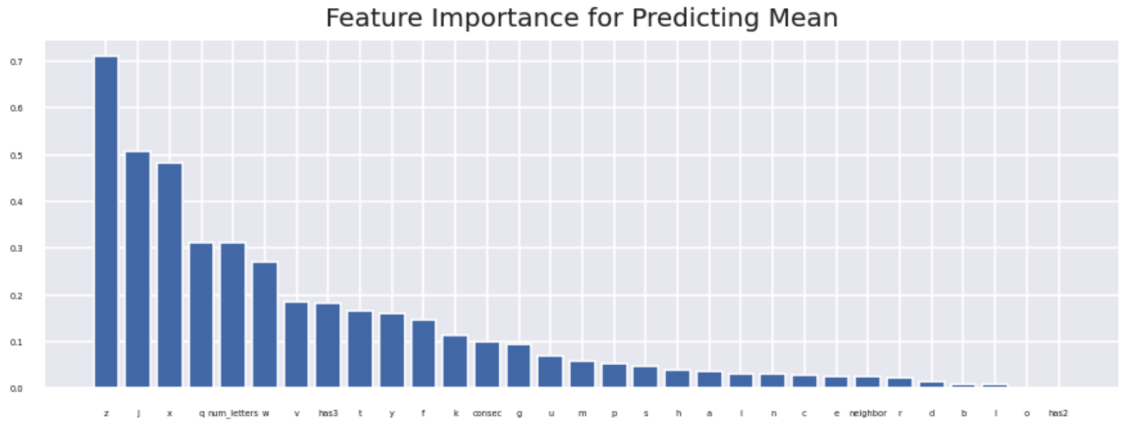
We split the data into train and test by taking a random sample of the data. We then run three models on each result and chart the results. For background, Ridge Regression utilizes L2 regularization on OLS regression. Namely, we assume a linear structure with $Y = X\beta + \epsilon$ where X is a matrix of inputs, β are the coefficients, and ϵ represents Gaussian noise. Ridge seeks to minimize $(Y - X\beta)^2 + \lambda \sum ||\beta_i||^2$ which has an important regularizing effect in both reducing the variance of the estimator and preventing issues with multicollinearity in the feature space, as described by [3]. Conversely, LASSO regression seeks to minimize $(Y - X\beta)^2 + \lambda \sum ||\beta_i||$, using L1 regularization. The main difference between Ridge and LASSO entails LASSO inducing sparsity on the coefficients, which is relevant for feature selection [3]. Therefore, we choose to add regularization here due to a variety of potentially unimportant and correlated features.

Model	Predictor	Tuning	MSE	R^2
OLS Regression	Mean	None	0.081	0.526
Lasso Regression	Mean	CV	0.082	0.526
Ridge Regression	Mean	CV	0.080	0.532

(19)

We use cross-validation of 5 folds to tune the hyper parameter λ in both the Ridge and LASSO model. Overall, we see similar results in the R^2 and MSE of each result. To find out about feature importance,

we use the features selected by LASSO as shown in the graph.



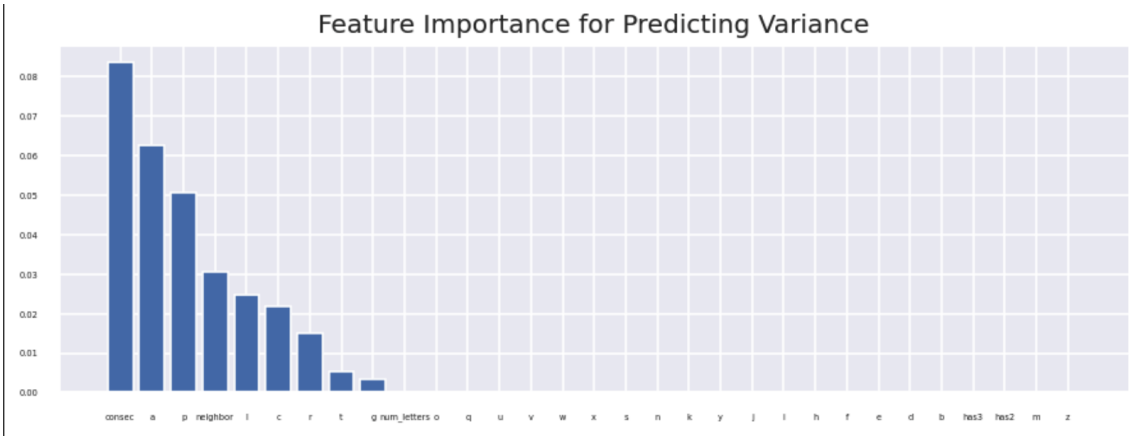
(20)

Through the results, we see that letters like ”z”, ”j”, and ”x” strongly raised the average try rate by a significant margin. Intuitively, players on Wordle choose more common letters like e, a, and i and rarely guess those rarer letters. Other features like the number of unique letters also showed a strong negative correlation with the mean. If there are duplicate letters, the strategy that many players employ of using diverse letters fails in this instance. Vowel placement also shows to have some important significance, showing that placement matters as much as letter content. Surprisingly, neighbors, with ”hatch” or ”patch” were overshadowed by others. Potentially, when choosing words, the developers of Wordle chose the most common neighbor form of that word. Now, when we attempt to predict the variance of the distribution:

Model	Predictor	Tuning	MSE	R^2
OLS Regression	Variance	None	0.079	0.271
Lasso Regression	Variance	CV	0.082	0.251
Ridge Regression	Variance	CV	0.078	0.276

(21)

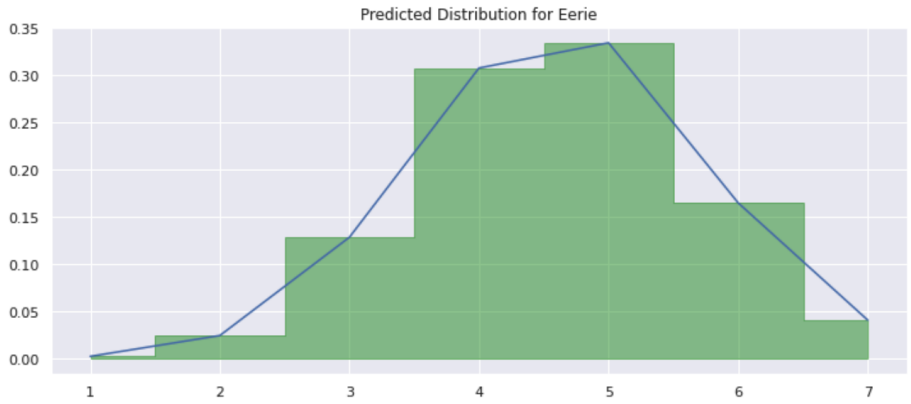
The models on average perform worse in terms of the R^2 showing that much of the variation is not being explained completely by our models. However, the mean squared error is comparable. Similarly, we also look at feature importance in the context of variance prediction.



(22)

Interestingly enough, we get an entirely different set of important parameters. Namely, we see that consecutive letters explain much of the variation among predictors. This should make sense as it splits between those who can get it within 3 tries because they guessed the letter first to those who are unable to complete it based on not picking the letter. Letters like p also exhibit some variance explanation. Finally, while orthogonal neighbors play less of a role in increasing the mean, they do increase its variance. Again, this is due to some players choosing the correct word circumstantially while others become unlucky.

Within our training set, we ultimately decide to choose the Ridge Regression model due to its flexibility and capability to deal with multicollinear features. However, there were no major differences in performances among the three linear models. In our next step of modeling, we create a feature for a given word and find it's predicted mean and variance. From there, we bucket the estimated normal distribution in order to produce the desired distribution. This process of discretizing outputted distributions has been used as an alternative to simulation with good efficacy, such as in evaluating the reliability of complex mechanical systems [7]. In our case, we used the following bucketing system: the area below 1.5 represents the proportion of 1 guess. The area between 1.5 and 2.5 represents the proportions of 2 attempts. Between 2.5 and 3.5 represents 3 attempts. Between 3.5 and 4.5 shows 4 tries. Between 4.5 and 5.5 shows 5 tries. Between 5.5 and 6.5 shows all 6 tries, and above 6.5 is the proportion of failure. Because the area under the normal curve predicted by our models will always be one, the summation of these proportions will also equal one. For example, we can featurize the word "EERIE" as such.



(23)

In other words, we believe that its distribution will have a mean of 4.6 and a variance of 1.17, showing it to be on the more difficult side. The buckets are $[0.2\%, 2.4\%, 12.8\%, 30.7\%, 33.3\%, 16.4\%, 4\%]$. There may be concerns that this approach towards model fitting a distribution can lead to biased or overfitted distributions. To prove that this model can work, we seek to run an experiment on the unseen test set. First, we define a metric to optimize over. In mathematical statistics, the Kullback-Leibler divergence or relative entropy measures how a probability distribution P is different than a reference distribution Q. For discrete distributions where χ is the sample space,

$$D_{KL}(P \mid\mid Q) = \sum_{x \in \chi} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

We define two other sample distributions which compete with our model. First, we define $T \sim N(\mu_D, {\sigma_D}^2)$ where μ_D represents the average of the mean try attempt over the training data set and

${\sigma_D}^2$ is the average variance of try attempts over the data set. A potential solution could be to guess the average mean and variance instead. Secondly, we define $R \sim \mathbb{E}(D_n)$ where we take the empirical expectation of each guess attempt as our guess for each word. Note that both distributions don't take into account information from the word and simply guess some form of average, while the third one does not assume a normal distribution. Our test is set up as such,

1. Select a word from the test dataset
2. Compute features from the word with our model and calculate R and T
3. Define our distribution P as the correct empirical distribution
4. Compute the KL divergence for each of the three distributions
5. Find and compare the average for each

Our train data set consisted of 287 data points and our test data set consisted of 72 data points. We run this test on different samples of train and test splits.

Model	Trial1	Trial2	Trial3	Trial4	Trial5
Ridge Normal	0.049	0.048	0.066	0.053	0.044
Mean Normal	0.090	0.077	0.092	0.093	0.065
Mean Empirical	0.082	0.071	0.83	0.82	0.055

(24)

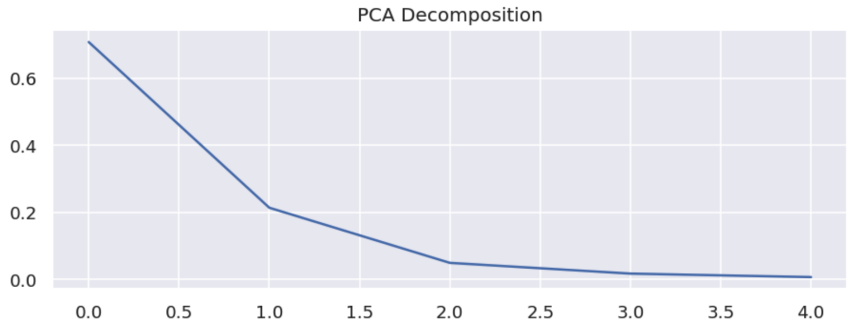
Here, using the Ridge Regression model to predict the mean and variance of a normal distribution outperformed the mean normal model and mean empirical model consistently. In terms of model performance, it seems that the mean empirical was more flexible than the mean normal in predicting similar distributions. Overall, we use Ridge Regression and 31 extracted features to predict a guess distribution in Wordle and show its superiority over naive mean average models. This makes us more confident in the prediction for "eerie". Potential uncertainties include having a small data set, assuming a linear structure, and missing important extraction of information for each word.

3.3 Classification Model

We now will develop and summarize a model to classify solution words by difficulty. Instead of arbitrarily choosing a difficulty based on the mean, we try to use a combination of dimensionality reduction and clustering to find groups to classify points in. By using an unsupervised learning method like clustering, we reduce the amount of human intervention and bias. First, our data is composed entirely of just the distribution of tries ranging from 1 to 7+. This is the most appropriate because difficulty should be based on empirical results rather than the features of the word. Therefore, we choose solely based on the empirical distribution of difficulty.

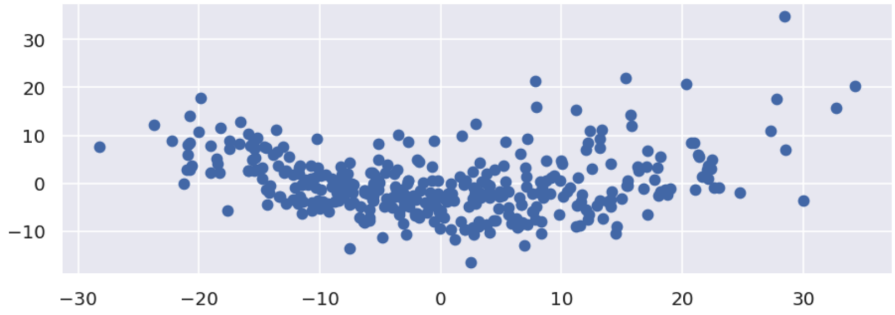
Before we apply clustering algorithms like KMeans, we decided to preprocess the data using linear dimensionality reduction, namely PCA. This is important in higher dimensions due to the curse of dimensionality, where the distance between points becomes very similar. PCA or Principal Components Analysis uses eigendecomposition to find orthogonal components which maximize variance.

Specifically, define $X^T X$ for some centered data matrix X, where $X^T X$ is proportional to the covariance matrix. By the Spectral Theorem, because $X^T X$ is symmetric, it decomposes into $U^T DU$ where D is the corresponding ordered eigenvalues in a diagonal matrix and U is an orthogonal matrix of eigenvectors. PCA concludes that directions which maximize variance are equivalent to the eigenvectors of this matrix. Using a scree plot, we can find how much each direction contributes to the variance.



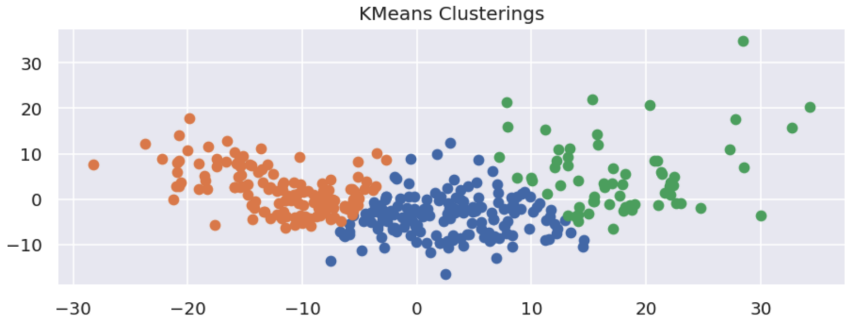
(25)

From this chart, we see that 90 percent of the variance is captured by 2 dimensions. Therefore, we can reduce the dimensionality of guesses into 2 dimensions. Conveniently, this allows us to visualize the clusters in a graph and determine a good clustering algorithm.



(26)

As there are no distinct clusters and most algorithms have would similar results, we choose KMeans Clustering into 3 classes. This is also justified by [2] who uses KMeans Clustering and PCA to understand Amazon ratings. From there, we will see if there are patterns that allow us to compare the 3 classes based on a difficulty metric. KMeans Clustering is a centroid-based clustering algorithm that switches between finding optimal centers of centroids and selecting points to fit inside of the centroid. It works particularly well when clusters are distinct, but in this case it would work well in ensuring equality in the number in each class.



(27)

We run KMeans clustering with the number of clusters set to 3. After making this chart, we see if there are any major differences in class in terms of average mean try and other characteristics. We also identify the clusters as "easy", "medium" and "hard". After finding the clusters, we calculate the means for each feature that we find interesting.

Feature	Easy	Medium	Hard
% Hard Mode	0.075	0.076	0.074
Number Letters	4.902	4.648	4.438
Consecutive	0.03	0.148	0.172
Has2	0.383	0.426	0.734
Has3	0.677	0.519	0.312
Neighbor	5.056	5.054	7.344
Mean Try	3.814	4.154	4.797
1 Try	0.797	0.284	0.266
2 Try	9.444	4.154	0.83
3 Try	30.85	20.253	12.109
4 Try	33.729	35.167	25.594
5 Try	17.752	26.296	29.191
6 Try	6.561	11.741	21.953
7 Try	1.045	2.093	8.266

(28)

Overall, because we account for the entire distribution, we choose difficulty not solely on a higher average mean try, but rather on distributional characteristics. This is important and allows us to extract even more information on the relationship between word characteristics and difficulty. For example, we see that harder words on average have more orthogonal neighbors. If there are Wordle Solutions like "water", "wager", and "wafer", people must often guess without the ability to guarantee a solution. Other important attributes include vowel placement in the second and third positions. As stated before, vowel placement shows a linear relationship in both the third and second positions. This makes sense, as third-placement vowel words oftentimes have a structure that is familiar to most players. For example, words like "shift", "there", and "prove" have common patterns.

Percent hard mode seemed independent of each difficulty. This shows that much of the variation in hard mode was seen across time. Finally, the number of letters and consecutive letters show similar patterns to the past date. Overall, using a separate scheme to model word difficulty, we see results that agree with the rationale we had when building the distribution model. When using our model in the context of "EERIE", we see that eerie is considered a hard word. There are only 3 unique letters in the word, only 1 consonant, and consecutive letters in "EE". It would be classified as "hard".

When discussing the accuracy of our model, we identify a difficulty score by looking at our training data set's distribution score. Of course, this requires us to know the distribution and rate it from there. Thus, when we do not have the distribution, we must use estimates or predictions from the previous case. This can add more estimates for errors. For example, we don't account for the word frequency in our features. This may play an important role. However, when we conducted a literature review, we saw from [8] that word frequency had little influence on ease, which we confirmed for ourselves on the data set. Still, there are various special cases that we do not account for, such as

using 2-gram features (2-letter combinations) in our data set for discussing difficulty. For example, we could individually classify words that end with an ”e” with words that use ”sh”, ”ch”, and ”tr” separately. Adding complexity could improve our accuracy model. Moreover, another concern could be losing interpretability when we apply KMeans clustering and PCA, as there is no clear-cut reason why a certain barrier was formed between classes.

4 Model Evaluation

4.1 Assumptions

In the time series model, we make a couple of assumptions. First, we assume that the model parameters are constant over time and there are no major news or macro events that cause a shift towards Wordle playing. This accounts for major level shifts or one time anomalies or seasonal trends for forecasting.

In the distribution model, we assume that for each word, the distribution of the guesses is approximately normal in a discrete fashion. We justify this through running a simulation multiple times with Gaussian noise to model the guesses as a continuous distribution and run a normality test on this for each word. Within our features, we make assumptions about the data for linear models. We assume that there are no multicollinearity between the data, there is a linear association, and the homogeneity of variance, and there is no correlation for how words for chosen. We also model the guesses in terms of a probability density distribution, assuming that players follow a similar play style.

In the classification, we apply linear dimensionality reduction, assuming that the structure is inherently linear and using Euclidean space. We assume something similar for KMeans clustering algorithm.

4.2 Advantages

1. Our models work independently of each other, using a combination of forecasting model, a linear model interpolation algorithm, and clustering to achieve the optimal results for prediction.
2. We optimize for AIC and BIC in the ARIMA time series model using grid search optimization. The projections we produce are logical for the next 60 days and continue the trend shown in the previous year
3. We show that in the distribution model, our model outperforms guessing the empirical mean or the mean normal distribution on the test data set across various tests, proving robustness. Our distribution model works across the various test and train tests we produced for the experiment.
4. We account and test for the importance of every letter and its impact on average guesses. We also account for the placement, allowing us to optimize for the amount of information extracted from a single word.

4.3 Disadvantages

1. Our models dependent on a relatively small data set of only around 300 observations. There could be some over fitting involved that could derail the predictions we make. Thus, we chose

simpler models like linear regression to account for this.

2. We do not account for more complex cases, such as using two letter combinations or dividing words into certain cases. This is also due to having a small data set.

3. We do not account for changes in behavior of the player base, but instead model our distribution based on the empirical guess results. Adding simulations, such as looking at player behavior, would be interesting in making our models also more robust.

4. Our classification model may have some intrepretability issues when classifying words, as we use methods like PCA and KMeans Clustering to choose our different difficulty classes.

5 Letter

To: Puzzle Editor of New York Times

From: MCM Team 2318299

Subject: Wordle Twitter Analysis

Date: February 20th 2023

Hello! We are writing to discuss how the community has reacted to Wordle by examining trends over time in its popularity and how to classify words based on difficulty. We use data from WordleStats, a Twitter account, which gives information about distributions of attempts and the number of reports for the year of 2022. Using this data, we develop a model to predict number of completions and distributions over time. Throughout the letter, we use "EERIE" on March 1st, 2023 as an example for our predictions.

Model Approach

To predict Wordle results, we use a set of three independent models.

1. Time Series Model: we use an ARIMA model to predict the number of participants on Wordle using posts on Twitter as a metric. Through de-trending the data, we also find significant results in difficult words predicting the number of Twitter posts.

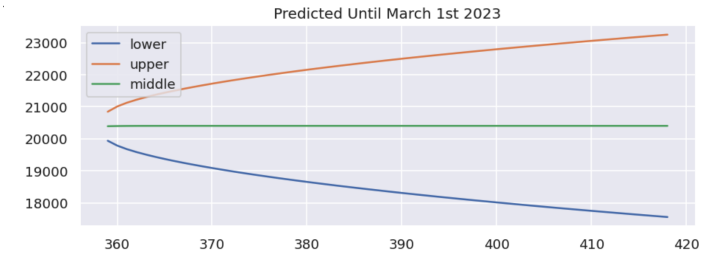
2. Distribution Model: by assuming the normality of data, we use regularized regression model Ridge to estimate the mean and variance of the data. By using Kullback Leibler Divergence metrics, we prove that this approach outperforms guessing the mean per bucket.

3. Classification Model: we run dimensionality reduction, PCA, on attempt distribution data and the clustering algorithm, KMeans Clustering, to produce distinct difficulty classes across the data set. The results agree with the intuition we built from the Distribution Model.

Model Results

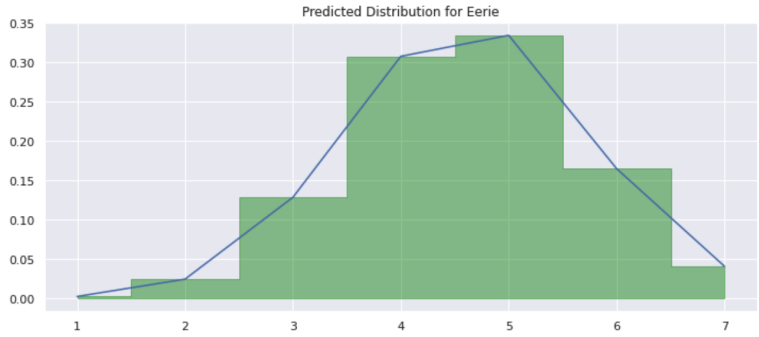
From optimizing our models over the data set, we attempt to predict the WordleStats result for "eerie", the Wordle Solution Word on March 1st, 2023. First, when examining time series, we see that the number of results will be somewhat stable around a predictable bound. Overall, we believe that "eerie" will have around 20404 reports with 9 percent of people reporting it on Hard Mode. Our 70 percent

confidence interval are numbers between 17556 and 23251.



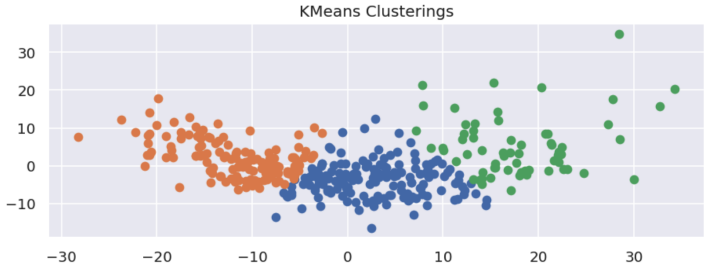
(29)

When discussing the distribution of eerie, we predict that it will have a mean of 4.6 and a variance of 1.17. This on average is a more difficult Wordle word and one with a smaller variance. For example, the average Wordle word mean try was 4.2, and the variance of the tries was 1.26. This was based on us estimating the parameters of a normal distribution and discretizing it over a set of intervals to find average guess counts.



(30)

Ultimately, through our 3-class difficulty scale, we classify "EERIE" as the hardest class of difficulty in the scale of easy, medium, and hard. We contribute this to multiple factors, including a small number of unique letters, many consecutive similar letters, and usage of mainly vowels with only 1 consonant. Each of these factors was measured within our model.



(31)

Overall, we find two main trends in the data. First, over time, the number of generated reports has dramatically decreased, indicating the popularity of Wordle decreasing. Second, we contribute Wordle difficulty to several factors: occurrence of rare letters, vowel placement, number of duplicate letters, and number of similar 5-letter words. Further discussions into the complexities and assumptions of our model can be further related to the future. If you have any other questions, please let us know!

Sincerely,
MCM Team 2318299

6 Appendix

References

[1] Benton J. Anderson and Jesse G. Meyer. Finding the optimal human strategy for wordle using maximum correct letter probabilities and reinforcement learning. *CoRR*, abs/2202.00557, 2022.

[2] Chantal Fry. Can we group similar amazon reviews: A case study with different clustering algorithms. *2016 IEEE Tenth International Conference on Semantic Computing*, 2016.

[3] Arthur Hoerl. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1), Feb 1970.

[4] Krishna Juluru, Hao-Hsin Shih, Krishna Nand Keshava Murthy, and Pierre Elnajjar. Bag-of-words technique in natural language processing: A primer for radiologists. *RadioGraphics*, 41(5):1420–1426, 2021.

[5] Ivan Li. Analyzing difficulty of wordle using linguistic characteristics to determine average success of twitter players. *Telling Stories With Data*, 2022.

[6] Stephen J. Lupker, Manuel Perea, and Colin J. Davis. Transposed-letter effects: Consonants, vowels and letter frequency. *Language and Cognitive Processes*, 23(1):93–116, 2008.

[7] Dilip Roy. The discrete normal distribution. *Communications in Statistics - Theory and Methods*, 32(10):1871–1883, 2003.

[8] David Waldron. What makes a wordle word hard? *David Waldron*, Sep 2022.