League of Legends Pop Culture:

How The Game Is Affected By Subredditting

1. Introduction and Problem Statement

League of Legends is a popular, free online game that has been available for well over a decade. Many people who avidly play the online game League of Legends take to Reddit in the event of exciting news such as new champions (which are the playable characters in the game), new champion abilities, innovative gameplays by competitive players, and so on. These events tend to be associated with increased champion play rates in the game. For example, after the release of the Netflix original TV series "Arcane", which is a show based on the game centered around champions Jinx and Vi, League players saw increased turnout of Jinx and Vi across all types of games within League of Legends, especially ranked games.

The goal of our project is to see if we can use data to highlight a correlation between Reddit threads from r/leagueoflegends, a subreddit with 5.7 million subscribers, and Riot Games's developer data on champions' play rates in League of Legends games. Given that a correlation exists, we use subreddit and playrate data to create a model that predicts whether a subreddit thread will or will not make a significant change in champion playrate utilizing multiple modes of parameters such as text featurization, comment approval, and play rates around the time of subreddit postings.

2. Related Work:

Sentiment analysis with word2vec: Medium - Yelp Sentiment Classification Using Word2Vec
This method gave a little bit of insight as to how to get a vector for a whole comment, essentially that a simple, non-weighted average suffices in the grand scheme of multinomial prediction. Otherwise, the word2vec development was of our own research with respect to the Gensim NLP library.
League of Legends Match Crawling: Crawling Matches Using the Riot Games API

We used "Method 1: Using league-v4 in conjunction with match-v5" in the above article to gather play rate data for each champion. The article above describes a crawling method using only the Riot API to collect the data we need.

Machine Learning Methods: Feature Analysis to League of Legends Victory Prediction on the Picks and Bans Phase

The above research article explains how feature analysis was used to predict League of Legends victory based on the picks and ban phase. The models that were used in experiments were Logistic Regression, Decision Trees, Naive Bayes, k-Nearest Neighbors, Random Forest, and Support Vector Machines. Since we are also attempting classification based on feature analysis, we would also like to try the algorithms used in this research article.

3. Data Sets

Reddit Data:

Using the Reddit pushshift API (PushShift Docs), which is a Reddit API that is a collective data dump of everything that is on Reddit, we collected threads from the League of Legends subreddit from the start of June 2021 until the end of March 2022. From this, there were a total of 143,567 valid threads, where valid is defined as data entries whose links linked to actual threads, not images or videos, or a domain that was not reddit. The features pulled for each data entry include the "submission" (thread) id, the title of the thread, the url leading to the thread on Reddit, the date of creation, and the number of upvotes as in the net total of vote reactions on a thread. Table displayed in Figure 1.

	submission_id [PK] character var	title text	url text	date date	upvotes bigint
1	o00m44	I dont cs when im fed	https://www.reddit.com/r/leagueofl...	2021-06-14	2
2	o00k4w	Totally planned prediction 2021 outplay that i had prepare...	https://v.redd.it/iqthzskakb571	2021-06-14	4
3	o00jk8	UCAM Esports vs MAD Lions Madrid - LEAGUE OF LEGEN...	https://youtube.com/watch?v=wfH...	2021-06-14	1

Figure 1: `submissions` datatable retrieved from SQL

Then, using the PRAW Reddit API wrapper (https://praw.readthedocs.io/en/stable/) we collected all of the comments from each of the 140k+ threads to get 819,322 comments. These comments were then concatenated and paired with their mother submission id, and then labeled with the champion of topic within the text (up to three most frequently mentioned). The table, after grouping comments by submission, has around 88,000 entries. An example of this is shown in Figure 2.

	submission_id [PK] character var	comment_text text	topic_champion text
1	o00m44	Roam by having the wave pushed to enemy turret . I only gank if bot or top is b...	['twisted fate']
2	o00j2e	That Xerath is fucking lost LMFAO. that's not luck. you get good when you get t...	['xerath']
3	o00gbv	Your thread has a disallowed title structure, and must be resubmitted with a ne...	['aatrox', 'ahri']
4	o00c4g	I think it's just bad luck man unless they are your placement games if they are t...	['aatrox', 'ahri']

Figure 2.1: `submissions_champions_arrays` datatable retrieved from SQL

The table in Figure 2.1 is then transformed into a table that follows first normal form rules, having 148,662 entries and excluding the comment text. An example of this table is show in Figure 2.2

	submission_id [PK] character varying (6)	topic_champion [PK] text
1	o00m44	twisted fate
2	o00j2e	xerath
3	o00gbv	aatrox
4	o00gbv	ahri
5	o00c4g	aatrox

Figure 2.2: From datatable `submissions_champions` from SQL. Table with submission id and the champion of topic in first normal form (1NF). For example, rows 5 and 6 correspond to one subreddit thread where two champions are equally the most frequently mentioned champion in the thread. Row 1 suggests that the thread with id 'o00m44' had only one champion mentioned the most whose name is 'twisted fate'

After building a Word2Vec model on the entire corpus of comments, we created average word vectors for each submission and appended it to a new table with 100 columns of feature values coming from the average word vector of 100 features. See Figure 3.2 below.

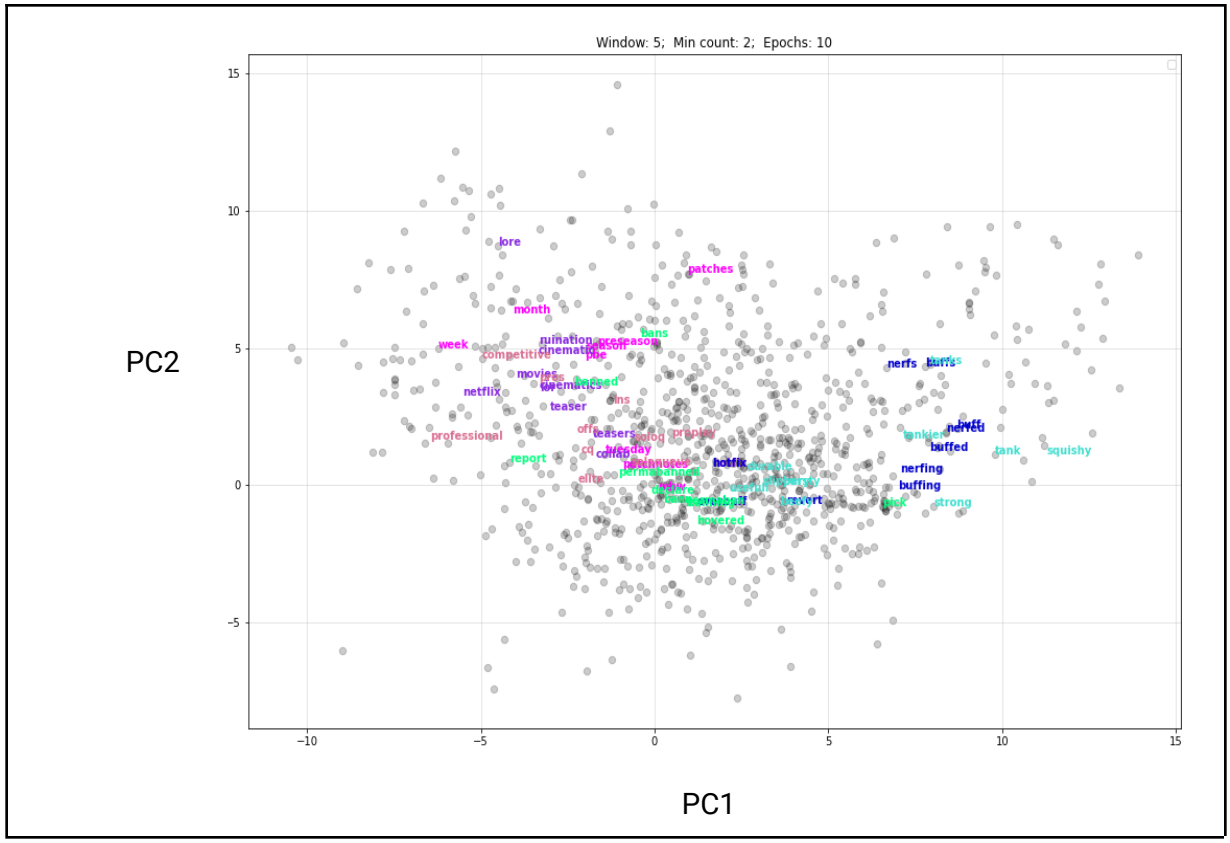


Figure 3.1: Word2Vec model cast from 100 features down to 2 dimensions via PCA (principal component analysis) for visualization purposes. Works in hot pink are related to the word "patch", purple represents that of "arcane", light green to "ban", dark blue to "nerf", cyan to "squishy", and dark pink to "ranked".

submission_id	0	1	2	3	4	5	6	7	8	9	...	90	91	92
nphcmp	0.427515	0.962019	-0.298306	-0.931229	0.148521	-0.265532	-0.225392	0.312821	0.351725	-0.826608	...	-0.871377	0.432817	-0.384758
nphcox	-0.461453	0.348936	0.292223	0.600786	-0.504491	0.381112	-0.128308	0.609844	0.154274	-0.316082	...	0.514666	-0.167961	-0.095978
nphdvg	-0.354448	-0.010296	0.290602	0.546764	-0.172587	0.946223	-0.369047	0.942072	0.419196	-0.244755	...	-0.149547	0.749959	1.011927
nphf2u	0.182016	-0.379647	-0.044724	0.525721	-0.194344	-0.459370	-0.452319	0.165828	0.381102	-0.417415	...	0.122480	0.141317	0.486795
nphg8o	-0.270704	1.065809	-0.033112	-0.029817	0.165749	0.909510	0.614613	1.142446	0.164142	-0.793884	...	0.108343	0.430246	0.228479
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
tsmbvd	-0.561517	-0.034997	0.235754	0.177085	0.188155	0.605975	0.089307	0.367986	0.160624	-0.814111	...	-0.496271	0.084638	0.732168
tsmc4e	-0.185837	0.397855	-0.265044	-0.009960	0.471385	0.437220	0.093974	-0.124487	0.610498	-0.500805	...	0.184885	-0.014865	0.220821
tsmn3d	-0.413484	1.707641	0.935986	-0.389490	0.486640	0.395572	-0.335198	1.111775	-0.034060	-0.939847	...	-0.648901	1.394263	0.860022
tsmp45	-0.698356	-0.154256	0.305001	0.835165	0.116430	0.485211	0.803038	0.181127	-0.050343	-0.687846	...	0.309111	0.285747	0.849028
tsmr13	0.793250	1.255845	0.082469	1.505851	-2.169148	-0.728317	-1.884043	-0.169799	0.108983	-0.010158	...	-1.087428	0.719117	0.255539

87728 rows × 100 columns

Figure 3.2: data from `submission_vectors.csv` showing a submission with its corresponding average vector split into columns. Table is cut off on right hand side, hiding columns 93 through 99

Riot Match Data:

(Riot Games API Documentation). The second data set that we utilized was from the Riot Games League of Legends API. The api allows us to access historical match data for every game played in League of Legends from June 2021 to the present. With information on every match played, we can then calculate the daily play rate statistics for all 159 champions in the game.

The final dataset includes the following columns:

Date: The date of the play rate data. For example, a date of "2022-01-01" indicates that the play rate data was collected for January 1st, 2022.

Champion: The champion that the play rate data was collected on. This will be used as a key to join with the topic_champion column from the Reddit data to build our model.

Play Rate: The play rate of the champion on the specific date. For example, a play rate of "0.05" indicates that the champion appeared in 5% of games on that specific day.

Datetime: A simple unix timestamp of the date. Will be used to join with the datetime column from the Reddit data in order to match play rate data with the date of the Reddit submissions.

Date	Champion	Play Rate	Datetime
12/15/2021	AurelionSol	0.001998	1.639555e+09

Figure 4: Riot Play Rate Data Example Row

An example row of our dataset is shown in Figure 4. This row indicates that on December 15th, 2021, the champion Aurelion Sol appeared in approximately 0.1% of games (a quite unpopular champion).

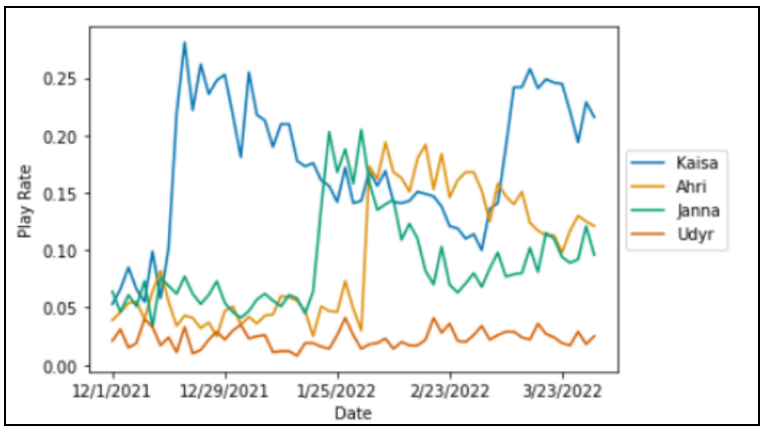


Figure 5: Champion Play Rate Over Time

Figure 5 above visualizes how champion play rate over time changes often. Four champion play rates are shown, all with different play rate trends. The champions Kaisa, Ahri, and Janna are all shown to have periodic spikes and drops in play rate over time. However, the champion Udyr has stayed unpopular over this time slot. Our project will attempt to find Reddit threads that correspond to the reasons for these spikes and drops in play rate.

4. Overall Technical Approach

Riot Match Collection:

Riot Games Developers's API does not have readily available data for us to use concerning play rates, so we collected and calculated historical match data ourselves. We used a method inspired by web crawling to collect daily play rate, starting with a random seed for a specific date, an initial match to start crawling on. Recursively, after we got our seed, we used the Riot API to collect data such as which champions were played in a specific game as well as all of the players present in that game. After saving which champions were played in that game, we then chose a random player from that match that has at least one game played on that date and repeated the crawl. We then repeated this process for each day from July 2021 until March 2022 in order to collect daily play rates data. After collecting all data, the data was stored in CSV files as well as uploaded to our PostgreSQL AWS RDS.

Reddit data collection:

We first used the PushShift Reddit API, which is a frequently updated Reddit data dump API, to collect the threads or "submission" following with the available Riot data from June 2021 to March 2022 from the League of Legends subreddit, or threads that are put in the League of Legends subdirectory on Reddit. In this collection, we collected data on the submission IDs, the date on which it was created, the title of, and the url of. This data collection, after cleaning, was stored as a CSV file. Then we used the PRAW Reddit wrapper to collect all of the comments associated with the collected thread IDs. Once a table was put together, we saved it into a CSV file. We then again used PRAW to append submission upvotes to the

submissions table, and updated the submissions CSV files. The CSV tables were then uploaded to our PostgreSQL AWS RDS.

Word2Vec:

Using the corpus of comments from the Reddit data collection. We used the Gensim NLP library to produce Word2Vec models. Using all combinations of parameters in window sizes 5 to 10, minimum word counts 2 to 4, and epoch numbers 5, 10, and 15, we produced 54 models to see which one would featurize our words the best. Our first method of evaluation was through visualization. Using a helper class edited by us, we visualized all 54 models cast to two dimensions via PCA. It turned out that the spread and location of certain groups of words did not change much relative between groups. This told us that the Word2Vec models, independent of parameter combination, produced similar results on the same corpus of text. So, the second metric of evaluation that we used was utilizing a set of "ground truth" pairs of words that we know are contextually relevant to League of Legends. Specifically, we used pairs of words that we'd expect to be fairly close together in terms of vector distance (for word vectors of 100 features) when the model is most optimal. Given that, we ran similarity tests on all of the models, picked the models that produced the shortest for each of the pairs of words, and then picked the model that appeared most frequently across all seven pairs' top models. Using the optimal model, we calculated the average word vector for all submissions, first by calculating the average word vector for a comment, then averaging the comment vectors per submission ID, and finally creating a table with submission IDs with their corresponding vector split into 100 columns.

PostgreSQL AWS RDS:

Within our cloud database, we added constraints to enforce relationships between tables as well as primary key constraints to enforce unique entries as appropriate. If a submission id is deleted from the mother datatable called submissions, it is expected that any datatable containing said deleted submission id will also delete rows containing that submission id. Additionally, some functions have been developed such as a function that takes a champion name and a date, and then calculates whether there was significant change in playrate in the two weeks after the inputted date; this function uses a simple two-sample t-test to return a boolean of significant change, though we'll be considering poisson prediction soon. Another function returns a joined table that contains all of submissions with their metadata as well as their average word vector; this function will be used for prediction model building later.

SKLearn Regression Modeling:

Scikit-learn (SKLearn) is one of the most strong machine learning libraries in Python. It uses a Python consistent way of interaction to provide a set of tools for statistical modeling and machine learning like classification, regression, clustering, and dimensionality reduction. The models we have used are logistic regression, and random forest. Logistic regression is an extremely popular AI approach that is used for classification tasks. It is used to predict the probability that an observation belongs to one of two possible classes, in our case, it predicts whether a subreddit thread will or will not make a significant change in champion playrate. We used logistic regression for its ease of use, interpretability, and scalability. There

are two forms of logistic regression, binary and multinomial. We chose binary since its target was one of two possible outcomes, no significant change or significant change. For very complex problems it may do a good job or help guide us to what a good model is for this data, and it's probably a good place to start. Features, we used 101 numeric features (100 vector values from Word2Vec + upvote count), and wanted to predict a class from 101 numeric features, easily translated into a Logistic Regression model. There are two main metrics for evaluating how well our model function, accuracy and confusion matrix which we cover later in this report. Next model random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. We used it because it performs better in class separation, randomness, and categorization despite imbalanced data. Some parameters we used are node size, the number of trees, and the number of features sampled. From there, the random forest classifier can be used to solve regression or classification problems. 2000 number of trees, minimum number of samples [2, 5, 10], maximum depth is [10, 20, 30, ….., 110], and bootstrap method of selecting samples for training each tree, bootstrap = [True, False]. Additionally, we instantiated the random search and fit it, using 3 fold cross validation, and searched across 100 different combinations, and used all available cores. We can view the best parameters from fitting the random search: the number of trees in the forest is 400, the minimum number of samples required to split an internal node is 10, the maximum depth of the tree is 70, and bootstrap is true, meaning that bootstrap samples are used when building trees. There are two main metrics for evaluating how well our model function, accuracy and confusion matrix which we cover later in this report.

5. Software

SKLearn	Logistic regression modeling, random forest modeling, cross validation, optimal model selection, analysis
API Scripting	Using a combination of Pandas, Riot Developer's API, Reddit's PushShift and PRAW APIs, we wrote scripts to collect all of the data used in this project
AWS RDS: PostgreSQL	Not only in the use of storing data in tables, but also creating constraints, indices, and complex functions to ease in the viewing of select data needed for our project's purposes
Gensim Word2Vec	Utilized for featurization of our subreddit's corpus vocabulary. This model gave us 100 features to use for each word and then each subreddit thread to be used in our predictive modeling

6. Experiments and Evaluation

Logistic Regression Modeling

Justification

Binary Classification: Our response variable is binary, representing whether or not there was a change in playrate for the specific reddit submission Features 101 numeric features (100 vector values from Word2Vec + upvote count). Want to predict a class from 101 numeric features, easily translated into a Logistic Regression model. Logistic regression is a very powerful algorithm, even for very complex problems it may do a good job and even with the right representation of the features, it can do an amazing job! From there it could help us decide whether with or without doing something to the data will or won't help us. Possibly not the best model with our data, but wanted to start with something simple.

Assumptions

1) Response variable is binary: Whether or not there was a change in play rate
2) Observations are independent: Each reddit submission independent from the previous
3) No multicollinearity in features: Most variables are not correlated. However, a few features from the Word2Vec vectors have vif scores of over 10, indicating problematic collinearity.
4) Sufficiently large sample size: Total sample size is > 30000 with least frequent outcome having ~3000 outcomes.

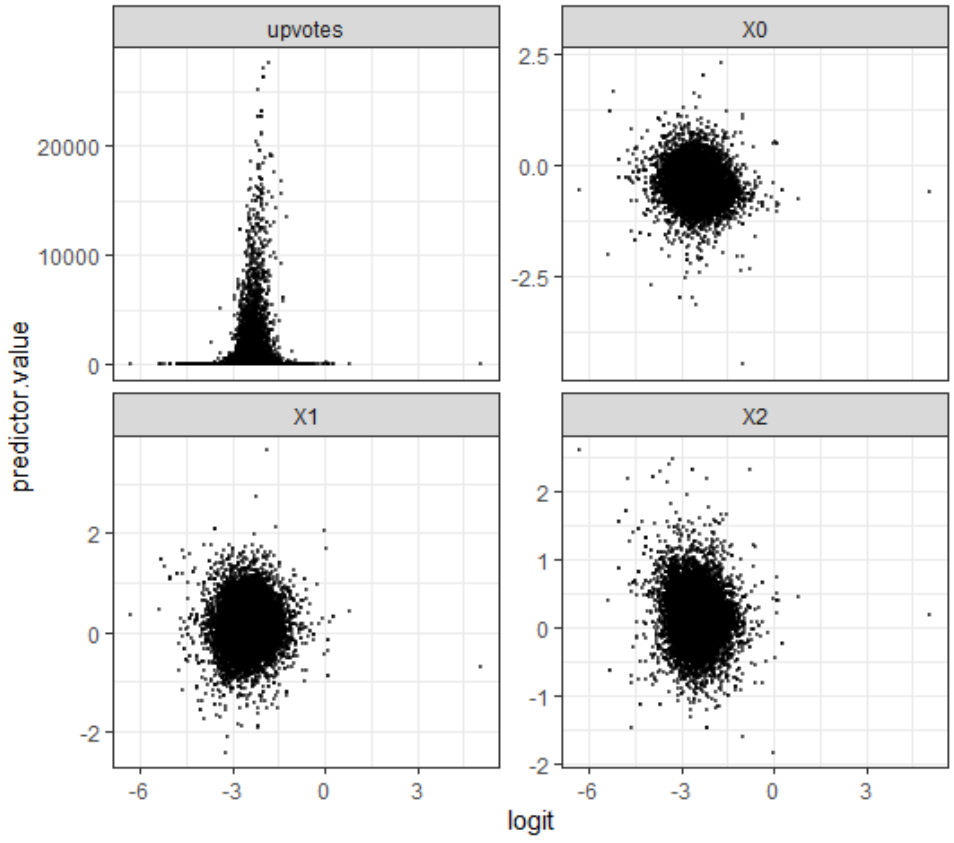


Figure 6: Predictor Values vs Logit Values of 4 Variables

5) Linearity of independent variables and log-odds: Figure 6 shows a plot of the linear relationship between continuous predictor variables and the logit of the outcome. We can see that none of the predictors here seem to have a linear relationship which violates this assumption and makes it highly improbable that logistic regression will work well.

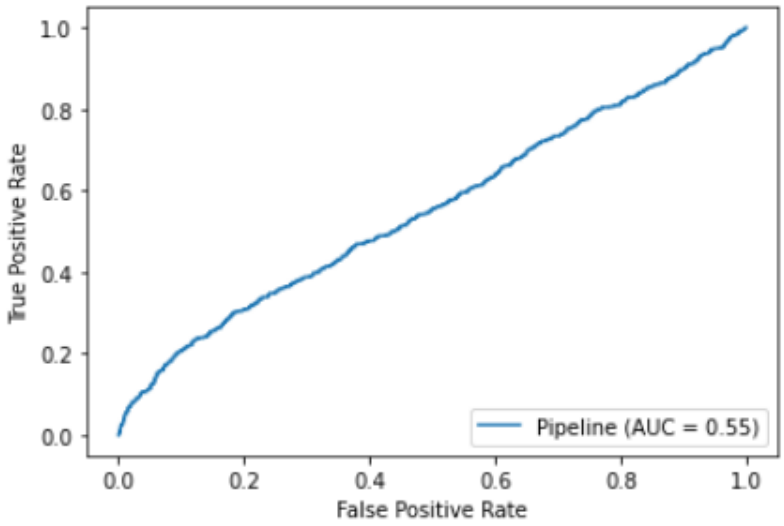


Figure 7: ROC Curve for Logistic Regression

Figure 7 shows the ROC curve for our logistic regression model. The curve comes close to the diagonal of the plot, indicating that the model does not demonstrate good performance.

Logistic regression preliminary results:

Overall, our logistic regression model was unacceptable. The combination of the classes of data being overwhelmingly unbalanced at a 11:1 ratio, the lack of colinearity in model features, and the less-than-appropriate area under the curve for the ROC curve demonstrate that logistic regression is not the appropriate method for the prediction modeling that we aim to create.

		Predicted Value	
		No Significant Change	Significant Change
True Value	No Significant Change	22589	7
	Significant Change	1955	3

Table 1: Confusion matrix for training data.

		Predicted Value	
		No Significant Change	Significant Change
True Value	No Significant Change	7533	2
	Significant Change	649	1

Table 2: Confusion matrix for testing data.

Table 1 shows the confusion matrix for the prediction classes of the training data in our logistic regression model. Where the data does actually have a significant change in playrate, the model predicts incorrectly to an extreme fault. Yet, where the data does not have a significant change in playrate, the model almost always predicts that there is not in fact a significant change in playrate. Table 2 showing the confusion matrix for the prediction classes of the testing data follows the same patterns. These errors are easily attributed to the imbalanced data. As such, our next step is in working with random forest modeling which performs better in class separation and categorization despite imbalanced data.

Random Forest Modeling

Justification

Classification: Our response variable is binary, representing most common whether or not there was a change in playrate for the specific reddit submission. Features: 2000 number of trees, minimum number of samples is how many data points a leaf of a tree must have in order to make the next decision down [2, 5, 10], maximum depth is how many levels of decisions deep the tree goes [10, 20, 30, ….., 110], and the method of selecting samples for training each tree is bootstrap. There can only be a minimum number of samples and a max level of depth. Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model, this is one of the reasons why we choose to use this model.

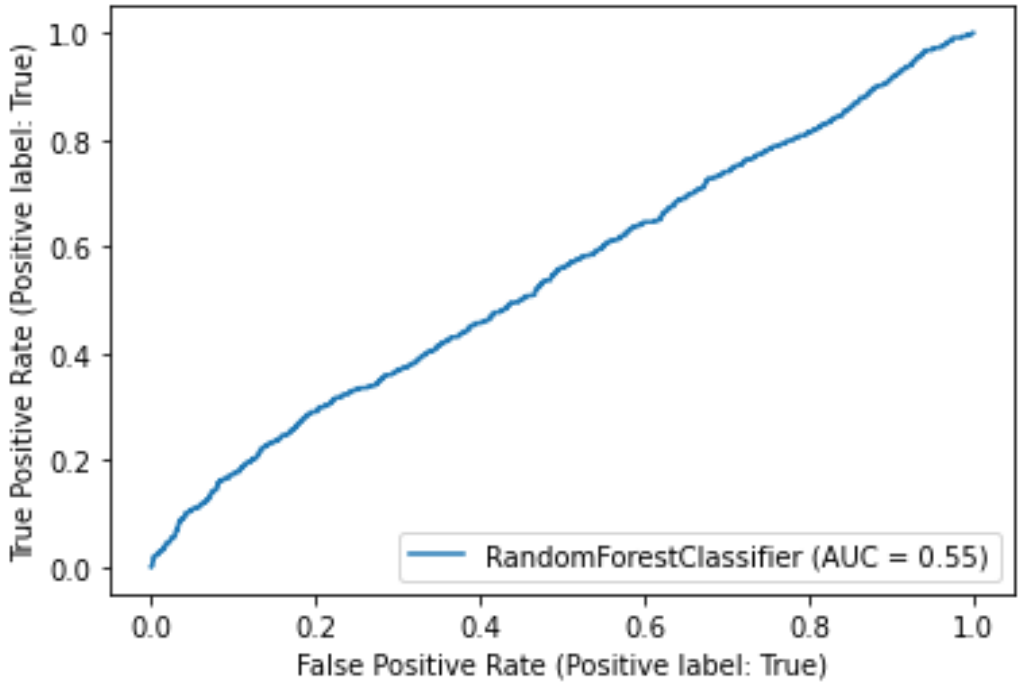
Assumptions

1) Response variable is binary: Whether or not there was a change in play rate

2) Best Split: At each step of building individual tree we find the best split of data
3) Sampling: While building a tree we use not the whole dataset, but bootstrap sample
4) Averaging: We aggregate the individual tree outputs by averaging.

		Predicted Value	
		No Significant Change	Significant Change
True Value	No Significant Change	7399	136
	Significant Change	626	24

Table 1: Confusion matrix for testing data.



7. Discussion and Conclusion

The models that we used for this project were logistic regression and a random forest. Although we primarily used logistic regression as a starting point to train a classification model, we learned the importance of assumption checking. For example, we assumed that there would be very little collinearity between our features since our only features were the number of upvotes and the Word2Vec vectors. However, we went back and checked the VIF scores after we saw the poor performance of our model, which showed that there were quite a few features that were highly correlated. Another technique that we learned was subsetting the data to account for imbalance. We knew that the imbalance of the classes would lead to poor performance, but we did not expect to see such a large improvement in performance after subsetting the data to have balanced classes. We also learned about the benefits of a random forest model over a logistic regression model. Since our predictors did not have a linear relationship with the log-odds from the logistic regression model, we had to find another classification model. We found that a random forest was a more suitable model because it is a non-linear classifier and can properly model our data using decision trees.

One lesson we learned from our project was the importance of robust method planning. We did not foresee that our champion labeling function would lead to incorrect assumptions until late into the project, which had a drastic impact on our results. For example, we did not account for the situation where our algorithm would mark a post as causing a significant change in play rate even though it was a different post that contributed to the change. Furthermore, using two different datasets and combining them in meaningful ways required more consideration and planning than we originally thought. Primarily, we did not know the most optimal way to label each posts' champion of topic. There were many situations where a champion appeared in a post the same amount of times as another champion or the comments mentioned a champion but the champion was not the primary focus of the thread. We should have spent more time in the beginning making sure our methods were robust and did not make bad assumptions.

If we had more time, we would go back and think of better methods for data collection. For example, we could have rebuilded the W2Vec model for 300 rather than 100 features, which possibly could have given us better results. Additionally, we could have re-wrangled our data with more specific constraints, i.e. requiring the thread to be about ranked or professional games rather than considering every thread on the subreddit. We would also spend more time looking at different statistical approaches for significance evaluation - something more sophisticated than a t-test of average play rates between two time blocks. Lastly, we could have also attempted forecasting play rates from historical data rather than just using a classification model.