Brawl Stars Character Recommender System

Brian Stroh

UW Master of Science in Data Science

DS785: Capstone

Dr. Nathan Pollesch

May 5, 2021

Abstract

Brawl Stars is a mobile real-time mobile game enjoyed by millions of players across the globe. It consists of several game modes, and each match results in a set of winning players an a set of losing players. This data is publicly available through the app developer's API, and this paper aims to study win rates of different characters and provide recommendations to players to help maximize their likelihood of winning a match. Furthermore, this project involves the development and deployment of an online application where players can access their own recommendations on their personal devices.

The hypothesis is that certain players are better or worse than average with certain characters in the game and these insights, in addition to the overall average win rates, can inform players to make decisions regarding character selection. Players only progress in the game when they win, so maximizing a player's likelihood of winning reduces the time spent achieving in-game objectives.

Individual match history was compared to population match history using adjusted Wald confidence intervals. Alternative binomial proportion tests were considered, including Wilson score intervals, credibility theory, and others. Ultimately, adjusted Wald confidence intervals provided the most reasonable win rate estimates when compared to other methods.

Keywords: Brawl Stars, recommendation system, adjusted Wald, confidence intervals, PostGreSQL, DigitalOcean, API

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Background

Brawl Stars is a smartphone-based battle royale game that allows players around the world to battle each other in real time using different characters in different game modes in different arenas. Having earned over $1 billion in revenues in the 4 years since its inception, Brawl Stars is a game enjoyed by millions of players across the globe (Chappel, 2021). This paper focuses on the implementation and analysis of a character recommendation system to provide players the greatest probability of winning a match. The system factors in population history as well as the player's own history to recommend characters that maximize the player's probability of winning a match, given the match configuration criteria.

Certain characters have innate advantages or disadvantages in certain game types or arenas. Character selection practices historically have been based on subjective measures such as personal preference, character attachment, or perhaps even random selection (Oliver et al., 2016). This paper proposes using a data-driven approach to maximize the likelihood of winning the match. By mining historical data and identifying the characters that have the best chance at winning in the current arena and game type, a character recommendation system can save players across the globe, collectively thousands of hours on their paths to completing in-game objectives.

Project Motivation

Many hours have been wasted working through characters that are incompatible with certain game modes or arenas. Other gamers have spent time searching for video "tutorials" about best strategies to use for each character, but have not paid attention to whether or not those character are well designed for the arenas they are played in. At the time that this paper was written, there were roughly 3680 unique character-arena combinations and many more if one considers the team compositions in team-based game modes. An analytics-based approach to character and arena selection would eliminate lengthy losing streaks and would foster greater in-game success and enjoyment.

Speed-run gamers also served as an inspiration for this project. Their focus has been to identify and exploit weaknesses in the game to be able to progress at far greater rates than other gamers (Gordon, 2020). Exploiting specific character dynamics was beyond the scope of this project, but the same ideological foundation was used to find and recommend characters that outperform others on certain arenas. This allow users of the recommendation system to take advantage of the character-attribute/map-design mismatch and maximize their likelihood of winning.

Objectives

Specific objectives include:

1. Use Brawl Stars API data to provide character recommendations to players that will maximize their likelihood of winning a match, given a game type and arena.
2. Similarly, identify arenas where players can exploit weaknesses by using characters that have significantly greater win rates in that particular arena.
3. Develop a database that periodically draws updated data from the Brawl Stars API.
    a. This process must be able to identify and prevent duplicate match entries.
    b. Users of the recommendation engine must be to search for their specific account history.

            i. This would allow for better personalization of recommendations.
            ii. Depending on the amount of data available for an individual's game history, credibility of the personal dataset may vary.
    c. Host the database in the cloud to allow for dynamic storage needs.
4. Build an online interface to allow users anywhere to extract recommendations from the system in real time.

Game Terminology

Throughout this paper, various characteristics of Brawl Stars are referred to using language specific to the game. In order to best communicate the effectiveness of the recommendation system, it would help to understand core concepts native to the Brawl Stars game. At times, more general terms will be used in place of the game-specific terms. These terms and their synonyms are defined in the following sections.

SuperCell

SuperCell is the enterprise that produces the game. They host the Brawl Star API which is the primary source of data for this study, and they are directly responsible for the design and maintenance of each map, mode and brawler.

Match, Game

Each round that a player plays in Brawl Stars is called a match. Each match takes place on one map, game mode, and players must commit to using a single brawler that they will use for the length of the match. Match durations range from 30 seconds to three minutes depending on the game mode and how well the player or player's team performs.

Map, Arena

These refer to the locale in which a particular match is hosted. For every game mode, the map available to play on rotates though a cycles that refreshes every few days. Although there have been over 100 maps released, roughly 80 maps are in Brawl Star's current rotation cycle. Each map is specific to a certain game mode, with the exception of Showdown maps, which are used for both Solo Showdown and Duo Showdown game modes.

Game Mode

Presently, there are nine game modes in which players can play matches, with a tenth mode currently being entertained for reintroduction by SuperCell. These game modes include Gem Grab, Solo Showdown, Duo Showdown, Heist, Bounty, Brawl Ball, Seige, Hot Zone, Knockout and Lone Star. All game modes, except for Lone Star and the Showdown, have two teams of three players each. Lone Star and Solo Showdown are free-for-all modes with ten players each, and Duo Showdown operates with five teams of two players each.

Brawler, Character

Characters that players can choose to play with in any one match are called brawlers. There are currently 46 available brawlers in Brawl Stars. All brawlers have different mechanics from one another, and these can come in the forms of movement speed, attack speed, attack range, attack damage, attack accuracy, re-load speed, support functions, or other special functions unique to the brawler. Brawlers have to be unlocked through in-game rewards before they can be used, so not all brawlers are available to all players. It may take two years to unlock all brawlers without spending money on the game, or six months if the player elects to buy brawlers as soon as they are available.

Trophies

Matchmaking for each game instance is based on the chosen brawler's trophy count. A brawler earns between 2 and 10 trophies when they win a match, and 1 to 8 trophies are deducted from the brawler when a player loses a match. The amount of trophies won or lost depends on the brawler's current trophies, game mode and rank (if applicable).

Rank

In Showdown and Lone Star game modes, players are defeated one by one another until only one player (or team) remains. For Showdown, the player's rank for a match is equal to how many players (or teams) were left when that player (or team) was defeated. If a player (or team) is the last one alive, then they are awarded rank one and the most trophies. The second to last player or team alive is given rank two and the second- most trophies, and so on. For Lone Star, players gain points for each kill. The brawler that accumulates the most points is awarded rank one and they are given the most trophies. The player that accumulates the least number of points is assigned rank 10, and they lose the most trophies.

Win

For the modes in which players do not finish with a rank, it is clear whether a team has won or lost. For Solo Showdown and Lone Star modes, ranks one through four are considered winners by SuperCell, although rank five also gains trophies. For the purposes of this study, a "win" aligns with SuperCell's treatment of rank. This provides for easier translation of Duo Showdown ranks as well. Since there are five teams of two players, teams with ranks one and two are considered to have won. The team that ranks third does not gain or lose trophies.

Win Rate

Win rate, as referred to throughout this paper, is a function of the number of wins and the number of total matches a player had with a specific brawler on a specific game mode and map. Population win rate and individual win rate each use the same formula, but individual win rate is restricted to the match history of the player in question. Win rate is formulaically calculated as the number of wins divided by the total number of matches played.

Club

A club is an in-game collection of players that share the same group. Players join clubs for a number of reasons, but there are no tangible benefits of doing so other than being able to better coordinate team events. Each club may contain up to 100 members, and the top ranking clubs are mostly entirely full.

Chapter 2: Literature Review

Analytics in ESports

A study of the past few years has shown that competitive video-gaming, otherwise known as eSports, has grown in popularity due to accessibility to streaming services and enhanced computing technology (Roundhill, 2020). Although only a small portion of gamers compete professionally, hobbyists around the globe engaged in countless hours of gameplay, meanwhile generating a plethora of data. Whereas many developers do not make their gameplay data available to the public, some developers such as SuperCell opened the window of opportunity for analysts through application programming interfaces (APIs). This allowed players to develop models to enhance their in-game decision-making and learn from past failures and successes. Where gameplay data was made available, data-inclined fans of these games built tools that range from predicting match outcomes, factorization of components that contribute to success and even recommender systems to help improve in-game decision making (Block et al., 2018). At this point, most of the focus in eSports analytics has been on win prediction algorithms since they appeal to both experienced players and outside audiences with little game experience (Hodge et al., 2017).

Due to its extensive available data, one of the games most widely analyzed for insights was Dota 2. Similar to Brawl Stars, Dota 2 is an online team-based game where players attempt to utilize synergies of their characters to exploit weaknesses of players on the opposing team. Research by Stanford students showed that selecting characters to form win-optimized team compositions improved the probability of winning a given match significantly (Conley & Perry, 2013). Whereas less data is available through the Brawl Stars API, match summaries, when analyzed in aggregate, can also be used to identify and exploit team synergies or individual character strengths.

A research paper detailing training regimens used by Spain's semi-professional Brawl Stars club, QLASH, revealed that even the very best Brawls Stars players and coaches weren't currently using insights from data to drive character selection (Brea Castro, 2021). Brea Castro did, however, recognize in concluding statements that Big Data is critical to the future success of eSports training programs. Given the medium of gameplay, one could argue that eSports has the most potential for gameplay improvement by analyzing data from past matches.

Recommender Systems for ESports

A recommender system is defined as a "decision making strategy for users under complex information environments". They are commonly used to help guide consumer behavior in online retail environments, from Netflix to Steam to Amazon. Amazon.com, for example, uses a collaborative filtering recommendation system to group users by similar product interests (Isinkaye et al., 2015). Products that similar user groups buy are recommended to other similar users whom have not yet bought those products. Netflix uses a content-based recommender system to recommend new video content based on its similarity with other viewed content.

A recommender system centered on maximizing win probability in team-based game matches could be considered a form of a content-based recommender system, but it does not align with the traditional definition of this type of system. Whereas there are content-based recommender systems to guide consumers to *which* game they should play, there aren't many systems that guide players to *how* they should play the game. Such recommendation systems are designed to optimize win probability (Conley & Perry, 2013).

Until this point, recommender systems used in gaming were used primarily to train artificial intelligence how to play the game for the users, rather than guide user actions. Perhaps due to low profit margins or unwieldy complexity, systems focused on guiding in-game player actions are exceedingly rare compared to traditional recommender systems. Despite their lack of prominence, successfully implemented in-game recommender systems received positive sentiments from test users in the range of 80 to 90% (Sifa et al., 2018).

Proposed Models

Credibility Theory

The Bülmann-Straub model is based on expected insurance claim frequency at particular model durations based on population and individual history, and aims to predict an individual's claim frequency (Yuan-tao et al., 2018). Szymańska (2018) generalized this theory into what is called a bonus-malus system. The objective of this paper and recommendation system is not to predict frequencies of random events, so there is a fundamental difference between the goal of the Bülmann-Straub credibility model implementations. However, bonus-malus systems assume that performance will be the population mean until the individual demonstrates otherwise. Rewards are administered if the individual performs the population average, and penalties are given if the individual performs worse than the population average. Significant individual battle history data will be necessary in order to make these adjustments.

Given that the scope of this type of ensemble model blending, based on credibility theory, is limited to only the property and casualty insurance industry, alternative algorithms will be considered. This ensemble bonus-malus model may be an appropriate method to fine-tune an existing widely-researched algorithm that will be used as the basis for this recommendation system. Potential implementations are discussed in Chapter 5 of this paper.

Binomial Hypothesis Tests And Confidence Intervals

A hypothesis test is defined in the form of a null hypothesis and an accompanying alternative hypothesis, where the null hypothesis is assumed to be true until proven otherwise. In statistical practice, alternative hypotheses are the antithesis of the null hypotheses. The null hypothesis used throughout this study is: “individual win rate is equal to the population win rate”. Consequently, the alternative hypothesis is: “individual win rate is not equal to the population win rate”.

Confidence intervals are used as an extension of these hypothesis tests. If, for example, that the population win rate is equal to 50%, the individual sample win rate is 75%, and the sample size is such that the 90% confidence interval ranges from 55% as the lower bound to 95% as the upper bound. The conclusion would be that the null hypothesis is rejected at the 10% significance level, because the population win rate lies outside of the 90% confidence interval. The individual’s win rate would be statistically significantly different from the population win rate.

Confidence intervals are most commonly used with continuous test statistics, but there have been several proposed methods for binary proportion confidence intervals. Since the metric at interest in this study (win rate) is a proportion of total matches played, it is appropriate to pursue these binary proportion confidence intervals.

Wald Interval. The Wald Interval, first introduced by Laplace in 1812, is the standard procedure of proportion tests, and is widely implemented (Brown et al., 2001, p. 115). The Wald confidence interval performs poorly unless the sample size is very large, which posed some challenges with respect to the recommendation system. First, it is not meant to be used with small sample sizes. The product of the sample size and both the probability of success and probability of failure must be at least 5. This is an assumption that failed more often than not with the sparsity of individual data available in this study. Second, the Wald confidence interval has characteristics of "inconsistency, unpredictability and poor performance" at lower sample sizes (Brown et al., 2001, p. 104).

Clopper-Pearson Exact Interval. Clopper-Pearson exact confidence intervals, established in 1934, are more predictable with lower sample sizes than Wald intervals are. Clopper-Pearson exact confidence intervals have large margins, and are presumed to be most likely to contain the true value of the test statistic as a result (Böhning, 1994). These intervals are computationally intensive which is not a major restricting factor with modern technology. However, when sample sizes are small, the bounds of these confidence intervals are excessively wide (Sauro & Lewis, 2005). As noted above, sparsity of data will provide for small sample sizes in the majority of the dataset, so exact confidence intervals are not a good fit for this study.

Wilson Score Interval. Wilson score intervals have a noticeable improvement above exact and Wald intervals, particularly for smaller sample sizes. Confidence intervals are much more narrow, although this is also the primary disadvantage of this method (Agresti & Coull, 1998). By having too narrow of upper and lower limits, the Wilson score interval is likely to reject the null hypothesis even with very low sample sizes if the sample win rate proportions are 0.00 or 1.00 and if the sample size is less than five. Since sample sizes that are smaller than five are unequivocally too small to make any definitive judgments for this study, Wilson score intervals are likely not suitable for this study. They will still be explored experimentally, as discussed in further detail later in this paper.

Adjusted Wald Confidence Interval. Proposed by Agresti & Coull in 1998, this method adds two successes and two failures to the sample's proportion test statistic, and then uses the Wald method for computing confidence intervals. It was derived as an approximation of the Wilson score interval, but relies on the Wald formula which was derived long before the Wilson score interval was (Reiczigel, 2003). In the few years since it's adoption, it has become widely regarded as a top performing proportion test for both small and large sample sizes with respect to probability coverage, performance along the boundaries (proportions very close to 0.00 and 1.00), as well as simplicity in implementation (Brown et al., 2012). Intuitively, it will make it more difficult to reject population win rates that are close to 0.50 which may be problematic since match wins are equally distributed between "winners" and "losers". However, the increase in sample size will reduce the confidence interval margins which could offset the mean-reverting nature of the adjusted Wald confidence interval. This method is a preferred alternative in most cases with small sample sizes (Sauro & Lewis, 2005).

Chapter 3: Data Collection/Methodology

This chapter details the steps taken to develop and implement the Brawl Stars recommendation system from start to finish. Five main steps will be covered:

- Extract data from Brawl Stars API.
- Configure the database.
- Design the query structure.
- Develop the recommendation algorithm.
- Implement user interface in the cloud for universal access.

Data Extraction

The first step in being able to access the necessary data was to create a developer account on the Brawl Stars developer website[1]. At this point, a key was created which allows for data to be downloaded from SuperCell's servers in a secured and controlled manner, and yields consistently formatted and reliable data when requested correctly. The developer documentation was reviewed, and three unique data structures were identified as being particularly useful for this project. The *requests* package in Python was used to interface with the Brawl Stars API and perform each of the below API pulls daily.

Global Club Ranks

[1] The Brawl Stars Developer website can be found at https://developer.brawlstars.com/.

This API request returns a list of the top 200 ranking clubs by total trophy count. These may change from day to day, but this list will mostly stay the same since top clubs have low turnover. From this API request, JSON data was extracted in the form of a list of club identifiers.

Club Members

After collecting the list of the top 200 club identifiers, this list is iterated through and programmatically sends another request for the members of the club with that identifier. Each player's unique identifier is then added to a list for the next step. There are up to 100 members in each club, and most of the top clubs have exactly 100 members. At this point, nearly 20,000 unique Brawl Stars players have been selected by their unique identifiers.

Player Battle-log

The list of almost 20,000 player identifiers is then iterated through, and a request is sent to the API to download each player's recent battle history. These downloads are stored in their raw form in JSON file format, and read in at a later step using the *json* package in python. Each battle log has up to 25 of a player's most recent battles. Since 25 battles can be performed in the course of one hour, it is reasonable to assume that the players in the top clubs will have entirely new battle history every day, or even twice daily for the most active players.

Only unique battle logs are considered in this study. There can be duplicate records from less active players having the same battles in this battle history, or from top players that play against each other. A unique key was created for each battle. This key is the concatenation of the battle time (to the second) and the player's unique identifier. Since each player can only be in one battle at a time, there should only be one instance of each key in the battle database.

All battle records have the mode, map and battle time. The log for each battle appears differently depending on the battle mode. For all game modes except for Showdown and Lone Star, the battle log contains whether the current player's team won, and the team compositions. The teams are listed in random order, so a check was put in place to see whether the current player is on the first or second team, so that each of the six players in that battle can be classified as "winner" or "loser". For each of these battles, a "win" field was assigned the value of one if the player won, or zero if the player lost. The "rank" field was assigned a "None" value since it does not apply to these game modes.

For Solo Showdown and Lone Star game modes, there is no "win" classifier field, since these game modes finish with each player being given a rank. These log types have each player's information listed in sequential order from the player that finished with rank one listed first and rank 10 listed last. Duo Showdown is similarly structured, but pairs of players are listed in sequential order from rank one listed first to rank five listed last. The "win" field was assigned a "None" value since win can be determined subjectively based on rank. The "rank" field was populated based on the order of the player's appearance in the battle log record, and this field will be used to calculate whether a player has "won" or "lost" at a later stage.

Database Configuration

Depending on the game mode, there are up to 10 new records added for every battle, up to 25 new battles per API pull, and nearly 20,000 API pulls each day. This yields a maximum of 5 million new battle history records added to the database each day, with the actual count of unique records added at nearly half of that amount.

For all players that appeared in each battle record, a new record is created in a local pandas DataFrame that contains all necessary information: key, mode, map, time, player ID, brawler used, brawler trophy count, win, and rank. These temporary records are then pushed to a copy of the database. Records with multiple instances of the same key are identified once all new records have been added to the copy of the database, and duplicated entries are removed so that all records remain unique.

The database was first constructed as a pickle file, using Python's *pickle* package. This soon became unwieldy due to memory limits, and a local instance of SQL Server was used to host the database for the development phase of this project. Using Python's *pyodbc* package, SQL queries were generated and passed to this server to quickly write millions of records to the database and process duplicate entries. It was discovered during this phase that Microsoft SQL Server uses as much memory as possible in order to minimize query runtime. This has great advantages for certain business applications, but on a multi-purpose desktop machine, this means that all other processes are inhibited by the SQL Server instance (Carrig et al., 2019).

The SQL server database was later migrated to a PostGreSQL database for cloud implementation reasons. This had the added benefit of not requiring large amounts of the computer's RAM, and queries could still access all data quickly while reading and writing data back to hard disk storage. In order to interface with the PostGreSQL database, the python libraries *psycopg2* and *sqlalchemy* were used in addition to the *pandas* library.

The master database table, "records" contains a copy of the data described above in the pickle file. It was discovered later that querying on the master table returned results much slower than desired, so the master table was aggregated into three sets of paired tables, divided based on the brawler trophy count. For each set of tables, the count of the record matches is aggregated as "matches played", and win is calculated within a CASE-WHEN formula. For Solo Showdown and Lone Star game modes, win is one if rank is less than or equal to four or zero otherwise. For Duo Showdown mode, win is one if rank is less than or equal to two and zero otherwise. For all other modes, win is the one or zero populated in the win column as-is. Win is aggregated by sum across the groups and is returned as "wins". Record counts in each of the 6 aggregated tables are shown in Table 1. Additionally, "CREATE INDEX" was run on the aggregated tables in order to optimize query response time. Given that the below record counts were generates at a time when there were 69.5 million total records, this indicates that there were only an average of two matches played per player-map-mode-brawler combination within the datasets. If this was a uniform distribution, no sample sizes would be large enough to conduct statistical inference tests with, but fortunately there are many instances of only one match played per combination, and many larger groups.

Table 1
Database Aggregated Record Counts

Trophies	Individuals	Population
High	26,221,501	4,953
Medium	3,413,312	4,924
Low	4,687,001	4,941
Total	34,321,814	14,818

Query Structure

Two main querying functions were created in python to extract data from the database for analysis. Each of these queries are similar in structure, as each of them group data by mode, map and brawler. Each pulls data from one of three sets of tables, which is predetermined by the user-specified trophy range. These pre-aggregated tables are split based on the notion that more skilled players (with higher trophy counts) have different win rates with certain brawlers than players with lower trophy count brawlers. The data pulled from the pre-aggregated tables includes the mode, map, brawler, number of wins and total count of matches played.

For the population data query, the above approach is exactly the query sent to the database, with the only addition being an additional filter for the map of interest. The individual data query matches the population query with the only addition being another filter for the player ID. For the query that returns the aggregated data for all maps, there is no map or player filter, but the Boss Fight mode is filtered out because the win rates are disproportionately high for all brawlers since these matches are against a computer character instead of human players. After the queries are executed, a *pandas* DataFrame is returned and processed further as detailed in the algorithm development section below.

Algorithm Development

The recommendation system's users have the ability to choose from three different types of recommendations. These are detailed individually in the sections below.

Get Map Weaknesses

The simplest of the three recommendation algorithms is the one that does not rely on any individual player history, but is arguably the most valuable in terms of being able to maximize overall win rates in the game. This algorithm calculates win rates for each combination of mode, map and brawler with win rates calculated as "wins divided by matches played". This data is then sorted by win rate in descending order and returned to the user. This allows users to spot which maps and modes can be used to exploit very high win rates. If any of the top results match to one of the currently available game maps, the user can select the brawler that is recommended and enjoy one of the highest win rates available in the game.

Get Recommendation

This algorithm is based on a request for a specific map and mode, and is the most timely of the three recommendation algorithms in terms of being able to use it immediately. Two SQL queries are called to the database: one to collect the individual win history on that map, and one to collect the population win history on that map. Win rates are calculated for both the individual history and population history, and confidence intervals are applied to each brawler's win rates from the individual's own battle history. From the *statmodels.stats.proportion* package in python, the proportion_confint function is used with "wins" as the count and "matches_played" as the number of observations. A significance value of .1 was used so that interval margins will be smaller and allow more room for recommendations to be made on sparse data. The "agresti_coull" method was used, which is referred to as the "adjusted Wald method" in their own study and others in the Literature Review section of this paper. This method, as mentioned earlier, favors a population proportion of .5, which is conservative for the win rates being studied since each match has an equal number of "winners" and "losers". The conservatism is sufficient justification for using the greater significance level of .1 instead of .05.

For every brawler that the individual has match history for on the selected map, the confidence interval bounds are compared to the population win rate. If the population win rate is less than the lower bound of the individual's win rate confidence interval, then the individual win rate is used in place of the population win rate. Additionally, if the population win rate is greater than the upper bound of the individual’s win rate confidence interval, then the individual win rate is used in place of the population win rate. If the former is true, then the reason for the recommendation is returned as “Outperforming population win rate”. If the latter is true, then the reason for the recommendation is returned as “Underperforming population win rate”. The table of brawlers, win rates and reasons is then passed to the user of the recommendation system.

Get All Recommendations

This algorithm is similar to the “Get Recommendation” algorithm. The difference is that the above approach is applied to all map, mode, and brawler combinations. Only records where the individual’s win rate is statistically significantly different than the population win rate are returned to the user along with the reasons. This type of recommendation is helpful for users to be able to focus on their areas of weakness or exploit their strengths. If a particular brawler is underperforming population win rates for that brawler on several different maps, then it may be useful for that user to look up a tutorial video or strategy guide for that brawler specifically, and practice to improve with that brawler. Conversely, if there are several recommendations returned where a brawler is consistently outperforming population win rates on several maps, that brawler is one of the player’s strong suits and can be considered a fairly safe bet on most maps. If any map specific recommendation requests show that brawler in the top few win rates, the user can expect to perform well with that brawler on that map, even if they don’t have existing history with that combination.

Application Implementation

At the time that this paper was written, there were several options from which to choose a cloud provider to store the database and user interface. DigitalOcean was selected based on simplicity, functionality and cost. Their platform has the ability to vertically scale computing capabilities as the need grows, which is an attractive attribute for this project since the data grows significantly every day. This "elasticity" is an important feature for cloud providers that has proved to be critical to developers with similar needs (Galante et al., 2016).

Once the application and database servers were initialized, a copy of the local development PostGreSQL database was created using the pg_dump utility. Using pgAdmin to connect to the cloud database server, pg_restore was used on the SQL output from the dump procedure in order to recreate the database in the cloud. The initial database creation was slow to run, but the cloud database has been updated on an incremental basis since this with only new records being added.

Developing the application proved to be more challenging than anticipated. The idea was to have a simple interface where users could input the minimum necessary values and to press one of three buttons depending on the type of recommendation that they wanted to receive. After exploring options for deploying python web-applications, it because clear that python's *flask* package is the standard method for building simple applications. The basic *flask* concepts and code fundamentals were learned through watching Corey Schafer's video tutorial series and accompanying GitHub code which met all of the *flask* needs for this project (Schafer, 2019). The CSS and HTML portions of the application were fairly easy to code, as they are both well documented on W3Schools.com (W3Schools.com, 2021).

Once built, the recommendation system app was test locally on a desktop machine. The app was initially tested by connecting it to the local version of the match history database.

Queried recommendations returned just as they did in the development phase of the recommendation algorithms, and the results displayed in tables similar to their *pandas* python object views. The local application was then connected to the cloud-hosted version of the database, and at this point it became clear that the PostGreSQL tables needed to be optimized in some fashion. Pre-aggregated tables were creating by grouping mode, map and brawler fields and adding up the number of wins and matches played. Multiple-indexes were generated on the grouping fields, and the "ANALYZE" command was run on the local version of the database to internally optimize the PostGreSQL tables for querying (Hearth, 2019). This reduced query runtime on the local database from five seconds to 50 milliseconds, which is tremendous improvement. For the cloud database, pre-aggregated tables were created as well and multiple-index commands were run on these pre-aggregated tables. However, the "ANALYZE" command requires "super user" role privileges in PostGreSQL, and DigitalOcean does not currently allow its clients this level of access due to security concerns (DigitalOcean, 2019). Still, the new indexed data table structures reduce the query time on the cloud database to one second, which is a tremendous improvement on the 45 seconds that these queries took to run in the cloud prior to this database restructure.

Deploying the app itself was an even more challenging, and required tens of iterations of adjusting parameters, run options, and waiting for the application to build and attempt to deploy on DigitalOcean. Although there is a sample *flask* app on DigitalOcean's GitHub page, this app is minimalist in nature and many essential parameters are assumed by the package versions they used which aren't the same parameters that the recommendation system app's packages assumed. After many failed instances, DigitalOcean support was contacted and a new investigation method was explored. Several more attempts later, the app was running on the cloud. However, the app was not communicating with the cloud database server. Although DigitalOcean strongly encourages its users to only enable trusted IP addresses to access their databases, they have not enabled their own app servers to connect to their databases (DigitalOcean, 2020). The cloud database was opened up to receive connections from all public sources, and the app was able to run and query the database successfully.

Chapter 4: Results

Population Win Rates

Having over 69 million unique match history records in the PostGreSQL database allows for well-informed decision making in character selection on any given map in the Brawl Stars game. Win rates for each character are pulled on request based on the specified map and optional trophy threshold and win ranks. Table 2 demonstrates that an individual can expect to win a match on the “Split” map 65.52% of matches if they elect to play with character Belle, but they should only expect to win 14.29% of matches if they elect to play with character Piper. This is the most extreme example, but the most and least extreme differences are illustrated in Table 3. Even on the map-mode combinations where players have the least information to gain from win-rate-based character selection, they can still improve their likelihood of winning by up to 16.74% by choosing one of the better characters for that arena instead of one of the worst.

Table 2

Best and Worst "Land Ahoy" Map Brawlers

Map	Mode	Brawler	*Win Rate*
Land Ahoy	bounty	BELLE	65.52%
Land Ahoy	bounty	CARL	63.10%
Land Ahoy	bounty	SPROUT	57.86%
Land Ahoy	bounty	MR. P	57.56%
Land Ahoy	bounty	LOU	57.14%
Land Ahoy	bounty	EMZ	31.09%
Land Ahoy	bounty	BULL	27.54%
Land Ahoy	bounty	GALE	26.32%
Land Ahoy	bounty	SHELLY	21.54%
Land Ahoy	bounty	EL PRIMO	14.29%

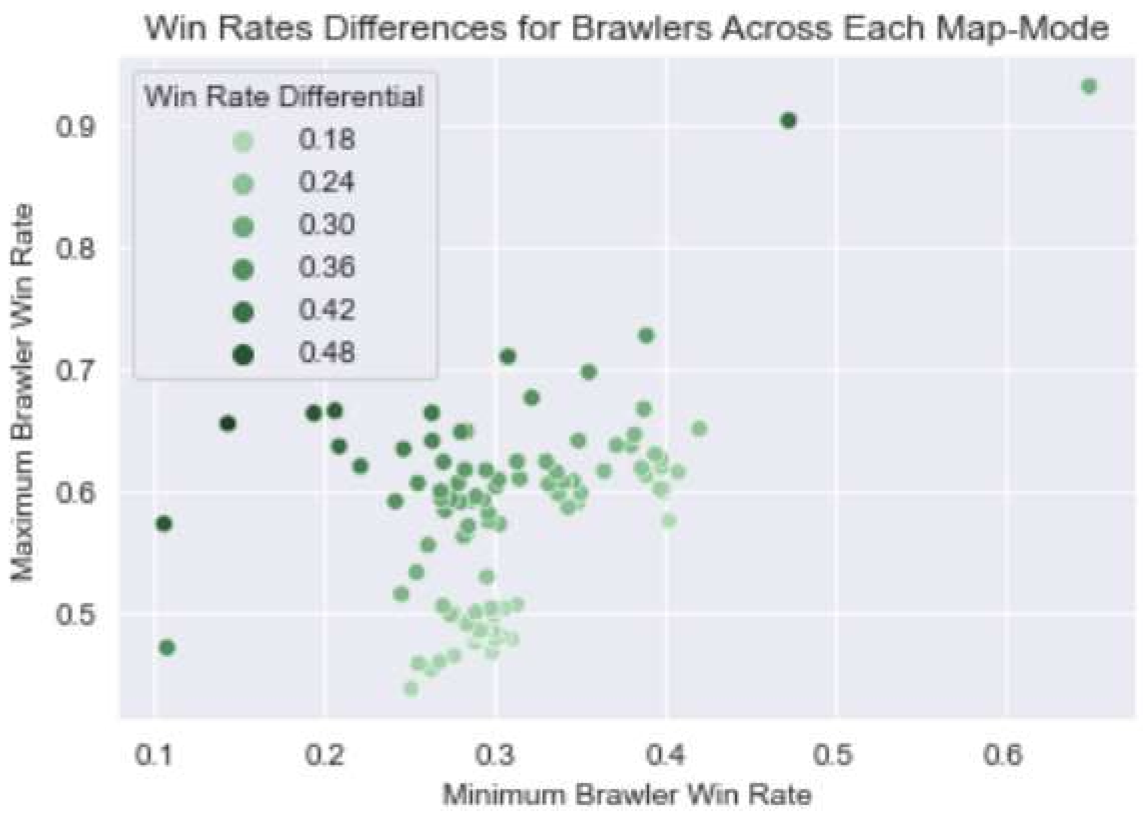
Table 3

Largest and Smallest Differences in Win Rate

Mode	Map	*Minimum Brawler Win Rate*	*Maximum Brawler Win Rate*	*Win Rate Differential*
bounty	Land Ahoy	14.29%	65.52%	51.23%
heist	Safe Zone	19.35%	66.36%	47.00%
bounty	Shooting Star	10.53%	57.28%	46.75%
heist	G.G. Mortuary	20.59%	66.55%	45.96%
bossFight	Danger Zone	47.27%	90.41%	43.14%
soloShowdown	Safety Center	30.02%	47.78%	17.77%
duoShowdown	Rockwall Brawl	30.41%	48.00%	17.59%
knockout	Snake Out	40.21%	57.52%	17.32%
duoShowdown	Cavern Churn	29.82%	46.72%	16.90%
duoShowdown	The Galaxy	31.03%	47.77%	16.74%

Figure 1 displays the distribution of minimum brawler win rate versus maximum brawler win rates for each map-mode combination. There are a few outliers on either end of the minimum brawler win rate. The map-modes with the greatest minimum brawler win rates are even above 50%, because these are on “Boss Fight” mode and players are playing against a computer player instead of each other. On the lower end, there are a few maps where certain brawlers have excessively low win rates. These brawlers should be avoided on these maps as much as possible, because the chances of winning are very low.

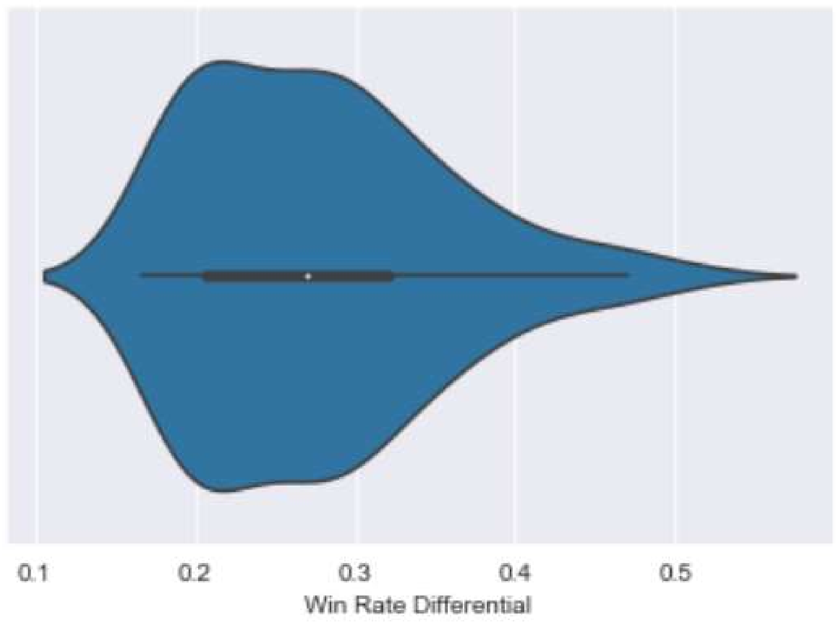
Figure 1



The relative density of win rate differences in Figure 2 further demonstrates that players can expect the best brawlers on most map-mode combinations to have win rates 20 to 35 percentage points better than the worst brawlers for those map-mode combinations. This represents the expected impact of the population based data from the recommendation system.

Figure 2

Differences Between Maximum and Minimum Win Rates for Brawlers Across Each Map-Mode



The recommendation algorithm that is based exclusively on population win rates is "Get Map Weaknesses". This recommendation identifies the best win rates for each map-mode combination and returns the top win rates. As explained further in Chaper 5, Belle has the greatest win rates in the game in most map-mode combinations. She is the newest brawler released to the game and SuperCell promotes new content this way by giving new brawlers extremely good fighting mechanics. Belle's power is expected to be reduced in the next in-game update. The results of the "Get Map Weaknesses" algorithm are shown in Table 4.

Table 4

Get Map Weakness - Top 20 Results

Mode	Map	Brawler	*Wins*	*Matches Played*	*Win Rate*
siege	Robo Highway	BELLE	727	1,024	71.00%
siege	Bot Riot	BELLE	456	654	69.72%
gemGrab	Snake Shop	BELLE	240	355	67.61%
brawlBall	Fast Fork	BELLE	2,537	3,803	66.71%

heist	G.G. Mortuary	STU	1,118	1,680	66.55%
siege	Bot Drop	BELLE	6,100	9,188	66.39%
heist	Safe Zone	BELLE	3,016	4,545	66.36%
bounty	Land Ahoy	BELLE	95	145	65.52%
brawlBall	Encirclement	BELLE	1,633	2,509	65.09%
hotZone	Triumvirate	JACKY	1,168	1,800	64.89%
siege	Olive Branch	BELLE	59	91	64.84%
brawlBall	Jumping Beans	BELLE	5,700	8,820	64.63%
heist	G.G. Mortuary	BELLE	208	322	64.60%
knockout	Ends Meet	BELLE	1,381	2,154	64.11%
hotZone	Dueling Beetles	BELLE	643	1,003	64.11%
bounty	Canal Grande	BELLE	958	1,503	63.74%
brawlBall	Triple Dribble	BELLE	9,343	14,664	63.71%
siege	Factory Rush	BELLE	3,249	5,104	63.66%
heist	Kaboom Canyon	BELLE	4,113	6,483	63.44%
bounty	Land Ahoy	CARL	53	84	63.10%

Individual-History-Informed Win Rates

For players looking to make character selection choices based on their own map-mode-brawler history, they would be presented with a panel similar to the one shown in Table 5. This player has match history for three different characters on this map that are statistically significantly different from the population win rate at the ten percent significance level. Presently, there are not widespread individual recommendations that differ from the population experience because there is not enough data yet for most map-mode-brawler combinations.

Table 5

Brawler Recommendations for Player "#2G080980", Mode "Brawl Ball", Map "Sneaky Fields" , Trophies "High"

Brawler	*Estimated Win Rate*	*Population Win Rate*	*Your Win Rate*	*Your Wins*	*Your Matches Played*	*Estimated Lower Bound*	*Estimated Upper Bound*	Reason
ROSA	74.1%	57.2%	74.1%	20	27	58.3%	85.5%	Better
CROW	71.4%	48.5%	71.4%	10	14	49.2%	86.7%	Better
BELLE	62.0%	62.0%	-	-	-	-	-	Population
LOU	60.4%	60.4%	-	-	-	-	-	Population

EMZ	58.8%	58.8%	-	-	-	-	-	Population
JACKY	56.0%	56.0%	-	-	-	-	-	Population
POCO	55.8%	55.8%	0.0%	0	1	0.0%	77.7%	Population
PAM	55.2%	55.2%	-	-	-	-	-	Population
NITA	54.8%	54.8%	-	-	-	-	-	Population
FRANK	54.2%	54.2%	28.6%	2	7	9.4%	59.7%	Population
COLT	44.4%	44.4%	0.0%	0	1	0.0%	77.7%	Population
MR. P	44.4%	44.4%	-	-	-	-	-	Population
SPROUT	44.1%	44.1%	-	-	-	-	-	Population
TICK	43.9%	43.9%	-	-	-	-	-	Population
MORTIS	43.0%	43.0%	-	-	-	-	-	Population
DYNAMIKE	40.5%	40.5%	-	-	-	-	-	Population
NANI	40.3%	40.3%	-	-	-	-	-	Population
EDGAR	39.9%	39.9%	-	-	-	-	-	Population
PIPER	39.8%	39.8%	-	-	-	-	-	Population
DARRYL	0.0%	52.5%	0.0%	0	5	0.0%	40.1%	Worse

Another option that players have is to get their individual recommendations for all map-mode-brawler combinations where their experience is statistically significantly different from the population win rate at the ten percent significance level. Table 6 shows these recommendations for the author's account history. Whereas these maps are not always available to players, being informed about individual strengths and weaknesses can influence decision making for using similar types of characters on these maps when they do become available. Players can also reference these lists and plan to use these characters on those maps once they do fall into the Brawl Stars map rotation.

Table 6

All Brawler Recommendations for Player "#8VUPQ2PP", Trophies "High"

Map	Mode	Brawler	Population Win Rate	Your Win Rate	Estimated Lower Bound	Estimated Upper Bound	Reason
Fast Fork	brawlBall	TARA	51.5%	100.0%	64.3%	100.0%	Better
Zip Zap	knockout	RICO	51.8%	100.0%	64.3%	100.0%	Better
Belle's Rock	knockout	MAX	51.1%	100.0%	59.9%	100.0%	Better
Ends Meet	knockout	SPIKE	48.0%	90.9%	66.1%	99.6%	Better

Crimewater	knockout	COLONEL RUFFS	51.1%	87.5%	57.2%	98.8%	Better
Red Herring	gemGrab	NITA	43.1%	72.7%	47.6%	88.9%	Better
Forsaken Falls	soloShowdown	DYNAMIKE	26.1%	66.7%	34.4%	88.6%	Better
Fast Fork	brawlBall	BYRON	44.1%	0.0%	0.0%	40.1%	Worse

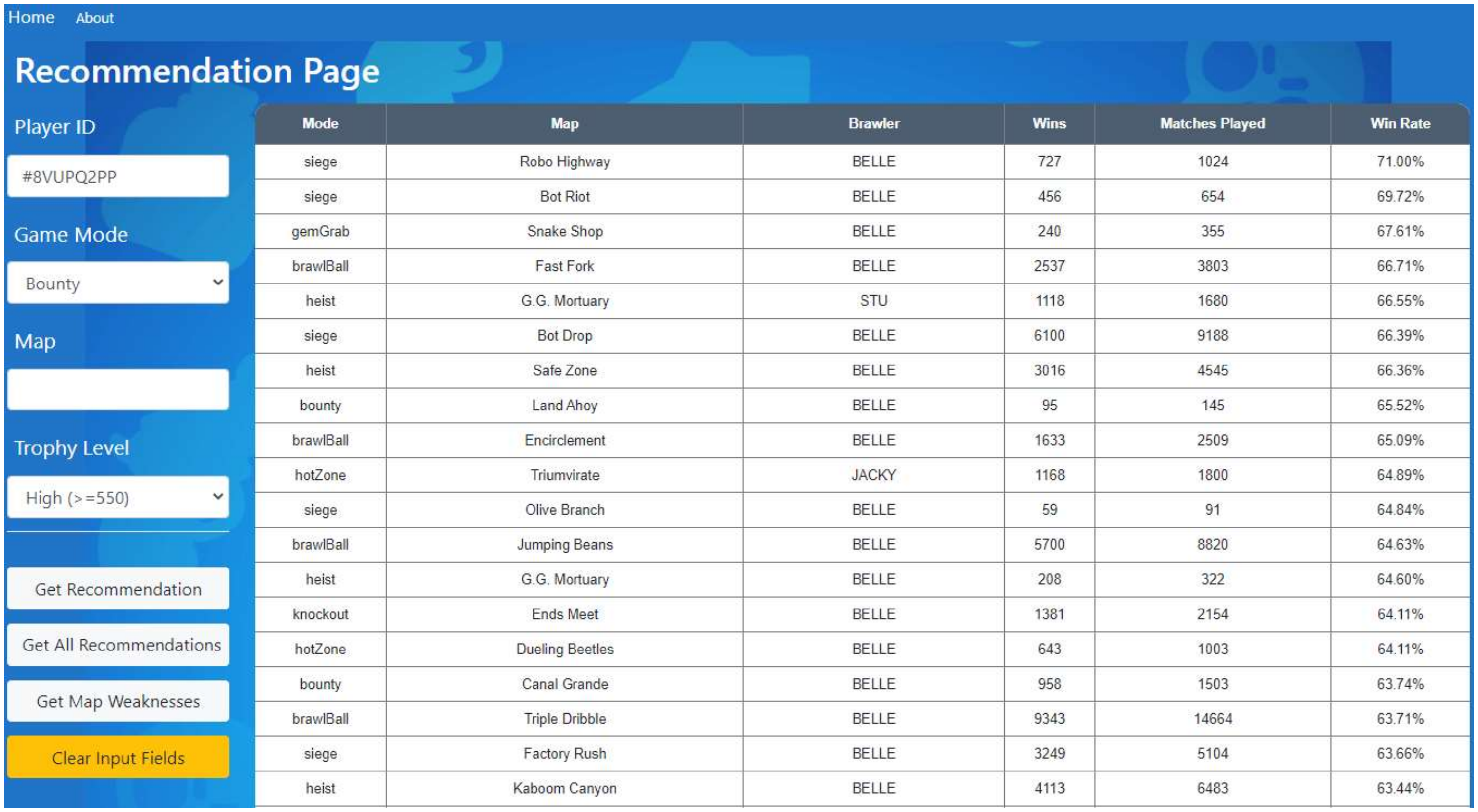
An analysis was performed to get the total portion of “Get All Recommendations” where the player’s experience was significantly different from the population average. The total proportion was 1.25%, which represents the portion of the data that help individual players to play to their strengths or to work on their weaknesses. Given the sparsity of the data and the infancy of the database, this portion is expected to grow over time.

Online Interface for Users

The values in the tables above are reflected in the tables provided in the user interface. Figure 3 shows the return from the “Get Map Weaknesses” button. Only trophy level must be specified for this query, but other inputs have default values input, which are ignored. Figure 4 shows the results from the “Get Recommendation” function. All inputs must be set for this recommendation to run as intended. Figure 5 demonstrates the result of pressing the “Get All Recommendations” button. Only player ID and trophy level must be specified for this query to run as intended.

The values in each of the figures below can be used to test the live application (see Appendix A). Player ID and Map have many valid values, so drop-down menu options were not used for those input fields. More values that can be used to test with have been included on the “About” page.

Figure 3



Player ID: #8VUPQ2PP

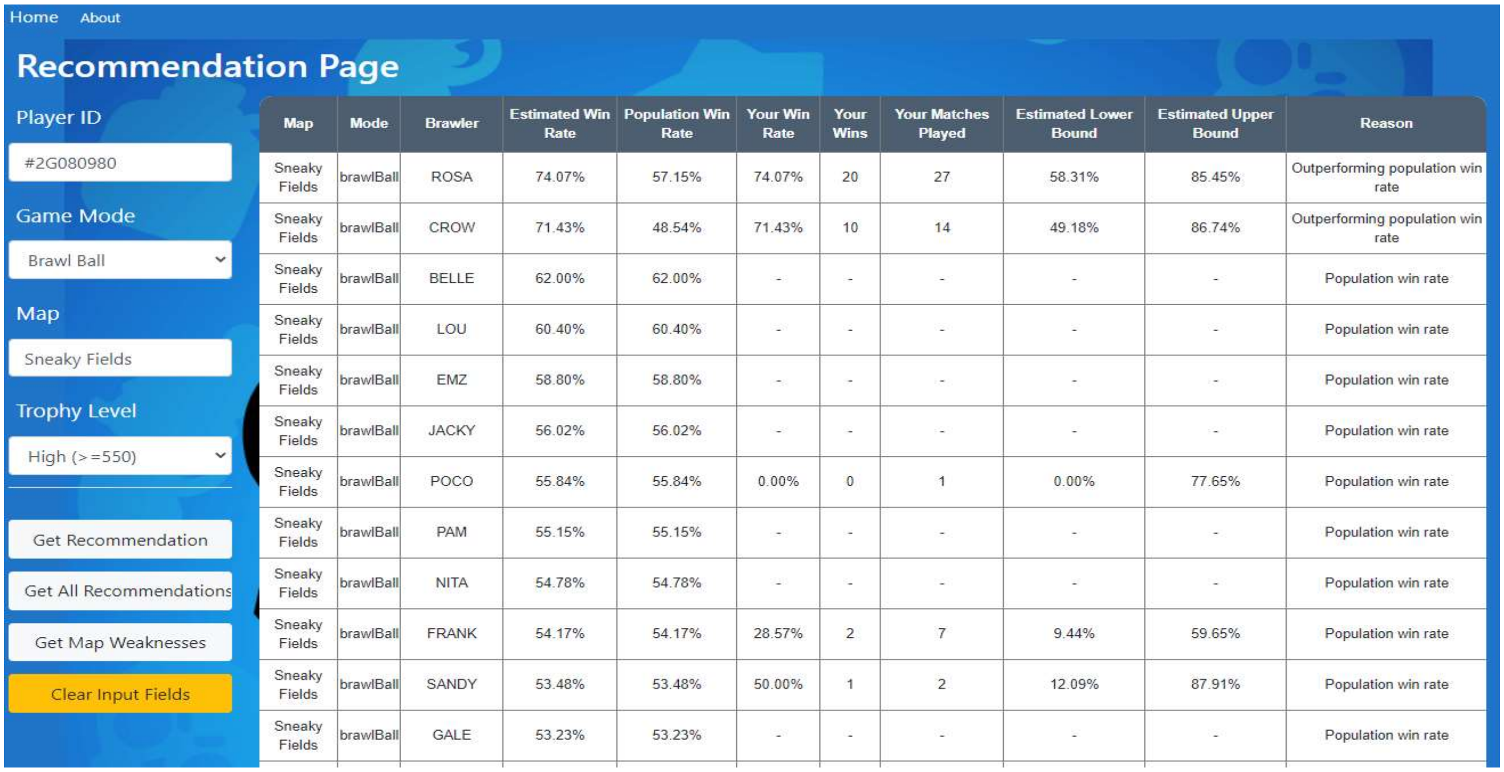
Game Mode: Bounty

Map:

Trophy Level: High (>=550)

Mode	Map	Brawler	Wins	Matches Played	Win Rate
siege	Robo Highway	BELLE	727	1024	71.00%
siege	Bot Riot	BELLE	456	654	69.72%
gemGrab	Snake Shop	BELLE	240	355	67.61%
brawlBall	Fast Fork	BELLE	2537	3803	66.71%
heist	G.G. Mortuary	STU	1118	1680	66.55%
siege	Bot Drop	BELLE	6100	9188	66.39%
heist	Safe Zone	BELLE	3016	4545	66.36%
bounty	Land Ahoy	BELLE	95	145	65.52%
brawlBall	Encirclement	BELLE	1633	2509	65.09%
hotZone	Triumvirate	JACKY	1168	1800	64.89%
siege	Olive Branch	BELLE	59	91	64.84%
brawlBall	Jumping Beans	BELLE	5700	8820	64.63%
heist	G.G. Mortuary	BELLE	208	322	64.60%
knockout	Ends Meet	BELLE	1381	2154	64.11%
hotZone	Dueling Beetles	BELLE	643	1003	64.11%
bounty	Canal Grande	BELLE	958	1503	63.74%
brawlBall	Triple Dribble	BELLE	9343	14664	63.71%
siege	Factory Rush	BELLE	3249	5104	63.66%
heist	Kaboom Canyon	BELLE	4113	6483	63.44%

Figure 4



Recommendation Page

Player ID: #2G080980

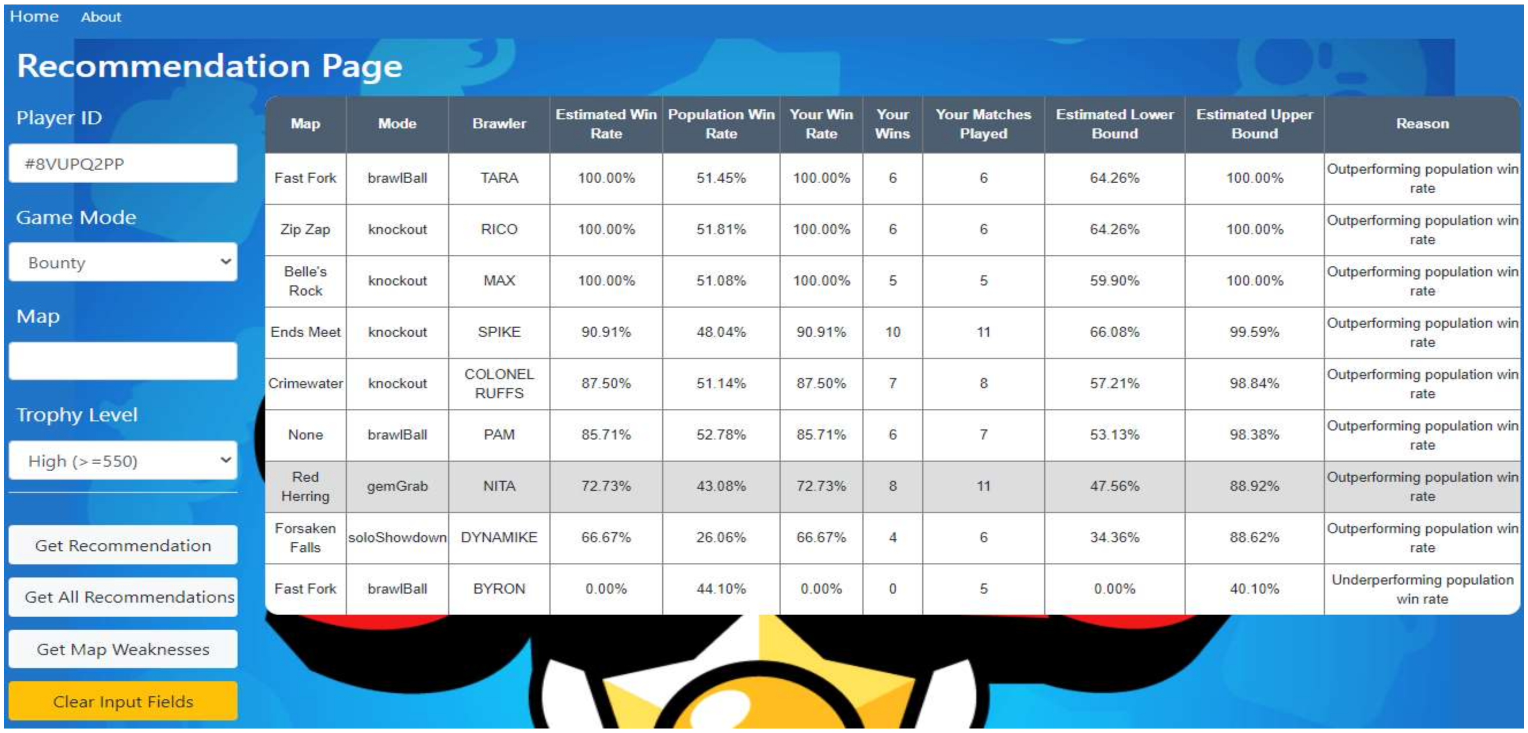
Game Mode: Brawl Ball

Map: Sneaky Fields

Trophy Level: High (>=550)

Map	Mode	Brawler	Estimated Win Rate	Population Win Rate	Your Win Rate	Your Wins	Your Matches Played	Estimated Lower Bound	Estimated Upper Bound	Reason
Sneaky Fields	brawlBall	ROSA	74.07%	57.15%	74.07%	20	27	58.31%	85.45%	Outperforming population win rate
Sneaky Fields	brawlBall	CROW	71.43%	48.54%	71.43%	10	14	49.18%	86.74%	Outperforming population win rate
Sneaky Fields	brawlBall	BELLE	62.00%	62.00%	-	-	-	-	-	Population win rate
Sneaky Fields	brawlBall	LOU	60.40%	60.40%	-	-	-	-	-	Population win rate
Sneaky Fields	brawlBall	EMZ	58.80%	58.80%	-	-	-	-	-	Population win rate
Sneaky Fields	brawlBall	JACKY	56.02%	56.02%	-	-	-	-	-	Population win rate
Sneaky Fields	brawlBall	POCO	55.84%	55.84%	0.00%	0	1	0.00%	77.65%	Population win rate
Sneaky Fields	brawlBall	PAM	55.15%	55.15%	-	-	-	-	-	Population win rate
Sneaky Fields	brawlBall	NITA	54.78%	54.78%	-	-	-	-	-	Population win rate
Sneaky Fields	brawlBall	FRANK	54.17%	54.17%	28.57%	2	7	9.44%	59.65%	Population win rate
Sneaky Fields	brawlBall	SANDY	53.48%	53.48%	50.00%	1	2	12.09%	87.91%	Population win rate
Sneaky Fields	brawlBall	GALE	53.23%	53.23%	-	-	-	-	-	Population win rate

Figure 5



Recommendation Page

Player ID: #8VUPQ2PP

Game Mode: Bounty

Map:

Trophy Level: High (>=550)

Map	Mode	Brawler	Estimated Win Rate	Population Win Rate	Your Win Rate	Your Wins	Your Matches Played	Estimated Lower Bound	Estimated Upper Bound	Reason
Fast Fork	brawlBall	TARA	100.00%	51.45%	100.00%	6	6	64.26%	100.00%	Outperforming population win rate
Zip Zap	knockout	RICO	100.00%	51.81%	100.00%	6	6	64.26%	100.00%	Outperforming population win rate
Belle's Rock	knockout	MAX	100.00%	51.08%	100.00%	5	5	59.90%	100.00%	Outperforming population win rate
Ends Meet	knockout	SPIKE	90.91%	48.04%	90.91%	10	11	66.08%	99.59%	Outperforming population win rate
Crimewater	knockout	COLONEL RUFFS	87.50%	51.14%	87.50%	7	8	57.21%	98.84%	Outperforming population win rate
None	brawlBall	PAM	85.71%	52.78%	85.71%	6	7	53.13%	98.38%	Outperforming population win rate
Red Herring	gemGrab	NITA	72.73%	43.08%	72.73%	8	11	47.56%	88.92%	Outperforming population win rate
Forsaken Falls	soloShowdown	DYNAMIKE	66.67%	26.06%	66.67%	4	6	34.36%	88.62%	Outperforming population win rate
Fast Fork	brawlBall	BYRON	0.00%	44.10%	0.00%	0	5	0.00%	40.10%	Underperforming population win rate

Chapter 5: Next Steps

App Promotion

User Experience Testing

The next logical step is promote the use of the recommendation system for user-experience testing. Avid players of Brawl Stars would know best what features they would like to have available from a recommendation app. By posting the website in a club's in-game forum, up to 100 club members could be reached as potential recommendation system testers. Any key features that are missing or could be improved upon and added iteratively until the club's testers are satisfied with the functionality of the recommendation system.

Brawl Stars Forums

The Brawl Stars Reddit forum has a large following of over 500,000 members. At least 8,500 of those members are active daily, and this would be a great way to deliver the new Brawl Stars recommendation system. Similarly, Brawl Stars has a large discord following with over 20,000 active daily members. Anyone with an account can post to these forums, and doing so would be a great way to quickly scale the recommendation system's exposure to the public. According to Forbes, social media and "contributing to the online conversation" are two of the best ways of promoting apps (Haselmayr, 2015).

Risks

The application's security is a primary concern. The cloud database can be queried by anyone, and the structural integrity of the database can be compromised by anyone who obtains the login credentials for the database. The credentials are currently posted publicly in the GitHub code for the app, and those should instead be set as parameters within the DigitalOcean interface. The database is currently set to receive connections from any IP address. DigitalOcean does not currently have a method to connect their apps to databases that are not linked on configuration, which is why the database is set up with an insecure connection (DigitalOcean, 2021). The app itself has a few security measures that were bypassed initially due to minimizing the time to deployment.

In-game updates are an ubiquitous risk. These come in the form of brawler re-balancing and new character releases. SuperCell monitors feedback from players through both explicit communication and through in-game statistics similar to the win rates utilized by this recommendation system. Every 1 to 2 months, SuperCell adjusts various characteristics on different brawlers to promote a more even playing field. This renders past win rates invalid for making future predictions. Similarly, SuperCell releases new brawlers to the game every few months. New brawlers typically have much stronger mechanics than other brawlers on average, and they enjoy high win rates until they are normalized with the other brawlers. The “Belle” brawler is a perfect example of this, since she was just released to the game 3 weeks ago. Table 3 shows that Belle has the greatest win rates on many of the maps in the game, and some by a large margin. Any time either one of these in-game updates is released, historical data should be phased out of the database as quickly as possible in order to maintain the validity of the recommendation system. A new algorithm will need to be created to address this, but the match time field in the database can be used to accomplish this.

SuperCell gave BrawlStars players the ability in late 2020 to create custom maps to submit to the community to play on. At this point it is unclear how that development will evolve into permanent content, but it possible that match history from these custom maps is feeding into the database for this project. There are many instances where the map name is unpopulated in the API data, and it is believed that these are match instances of user-submitted maps. The impact on the recommendation algorithm is negligible, but if SuperCell begins allowing these map names to flow through the API, this would expand the size of the aggregated datasets (which are grouped by mode, map and brawler) significantly. Perhaps another indicator would be included in the data from the API if this were to be the case, but this will have to be monitored closely.

Pipeline Optimization

A number of improvements can be made to the data pipeline. The current process relies on the database administrator running a number of python functions and SQL queries in sequence. This allows for checking data at intermediate steps for any unintended values, but this comes at the expense of automation. Whereas the sequential functions should still be available for further development of the recommendation algorithms, another function can be created that houses each of the sequential functions. This will automate all steps except for migrating the local database updates to the cloud-hosted database.

The cloud migration itself requires a complete overwrite of existing tables, which takes several hours to complete and the database's tables are assumed to be inaccessible during this update period. This process could be improved to include a local copy of the cloud database, which would be used to identify differences between the up-to-date local database and what is currently housed in the cloud. Instead of overwriting the entire tables, only the new data would be pushed to the aggregated tables in the cloud database. Additionally, the itemized table is included in the cloud database updates which adds significant overhead and is no longer used by the recommendation system queries. This table can be excluded from future database updates to reduce the time required to update the cloud database and associated app downtime.

When a user of the recommendation system searches for their player ID in the recommendation system and no data is found for that player, that player's ID should be added to a list that is returned to the local system that updates the database. A separate API query can be written that extracts the club ID for each of these users, and that club ID can be added to the master list that is used to make daily database updates.

User Experience Improvements

The domain name for the recommendation system app is one assigned by DigitalOcean, and is not one that is easy to remember. This recommendation app could be migrated without much difficulty to one that is easier for users to remember and to type into their browser search bar. Better yet, on top of this being a browser app, this recommendation system could be implemented into an iPhone or Android app for a better mobile user experience. That endeavor would require pretty significant effort, but it would allow the recommendation system to reach more users on the platforms of their choice.

The Get Recommendation algorithm itself has room for improvement. If the adjusted Wald confidence interval rejects the null population win rate, the estimated win rate should not be precisely the individual's win rate, but some blend of the two win rates. A more reasonable estimated win rate would be an interpolated win rate between the population win rate and the individual win rate. The weighting for the individual win rate would increase as individual observation counts increases, and only at very high individual observation counts would the individual win rate reflect the true estimated win rate for a given player. This would blend binomial proportion tests with credibility theory's bonus-malus system, as was referenced in the literature review chapter of this paper.

Lastly, additional recommendation algorithms can be incorporated within this recommendation system. For all modes except for Solo Showdown and Lone Star, each match consists of teams of 2 or 3. Creating a recommendation algorithm that identifies team compositions with the greatest win rates on each map could be even more useful than recommending individual brawlers to play with. This algorithm, if implemented, would have to be exclusively limited to population data due to the lack of match instances each individual player would have with each team composition and map. In practice, it would yield results similar to the Get Map Weaknesses algorithm, and would not involve statistical methods beyond simple averages.

These types of win rates would help players identify brawlers that have unique synergies, where their special abilities build off of one another to maximize their chance of winning a match. This proved to be out of scope for this paper, as this type of recommendation algorithm would require vastly more data to be reliable. Since count instances of each team composition in the data are far less than individual brawler counts, similar challenges with sparse data would be found in the population datasets as is found in the individual datasets for the existing recommendation algorithms.

Another refinement of the recommendation algorithm could leverage additional data mining techniques to recommend similar types of brawlers to those that have high win rates. For example, Barley, Tick, Dynamike and Sprout are all “thrower” type brawlers, meaning that they launch projectiles over walls. That have other similarities as well, and it would be reasonable to assume that on a map that has a lot of walls and Dynamike performs very well on, Barley, Tick and Sprout may also have high win rates. This is accounted for in the population win rate recommendation algorithms, but this type of association analysis could improve individual recommendations when one experiences much greater win rates than average with a certain type of brawler.

Conclusion

This paper detailed the process of implementing a Brawl Stars recommendation system from ideation to development to refinements to deployment. Many challenges were encountered along the way. Throughout the project, the underlying python code was written, tested, relied upon and then deleted and rewritten in a new form several times as the data structures used to the store the recommendation algorithms' data evolved from more primitive states to production-ready forms. Once a local app had be developed and tested, many code refinements had to be made in order to migrate the local instance of the app to a location where Brawl Stars players can access it from anywhere.

Extending beyond the current industry's definition of recommendation algorithms, this project sought to assist players in maximizing their likelihood of winning a match in Brawl Stars based on historical win rates. Whereas traditional recommendation algorithms may recommend a similar type of in-game character to use based on their previous selections and other player's preferences, the primary objective was to improve the in-game experience and success levels that the recommendation system's users have. Several types of binomial proportion tests were explored in additional to the insurance industry's concept of Credibility Theory. Adjusted Wald confidence intervals were found to be the most robust type of statistical inference test that could be applied to this study's small sample sizes.

Over 860,000 scenarios were identified where individuals can leverage their proven strengths or weaknesses to improve their likelihood of winning a match versus simply using the population average win rate. Collectively, these recommendations could save players internationally thousands of hours in their efforts to progress their Brawl Stars accounts.

Appendix A: Project Code & Application

All code created for this project can be found here: https://github.com/brianjstroh/BrawlStars/ Raw JSON API data files are not included due to their size, nor are the temporary local pickle files that are used as an intermediate step for loading data to the local PostGreSQL database for similar reasons.

The application can be accessed here: https://brawl-stars-bvsfa.ondigitalocean.app/home

As development continues, the application may be migrated to a new domain after June 2021.

Works Cited

Agresti, A., & Coull, B. (1998). Approximate Is Better than "Exact" for Interval Estimation of Binomial Proportions. The American Statistician, 52(2), 119-126. doi:10.2307/2685469

Böhning, D. (1994). Better Approximate Confidence Intervals for a Binomial Parameter. The Canadian Journal of Statistics / La Revue Canadienne De Statistique, 22(2), 207-218. doi:10.2307/3315584

Brea Castro, M. (2021). Didactic methodology in professional e-sport training. An international experience in Brawl Stars. *Retos : Nuevas Perspectivas de Educación Física, Deporte y Recreación.*, 41, 247–255.

Brown, B. M, Suesse, Thomas, & Yap, Von Bing. (2012). Wilson Confidence Intervals for the Two-Sample Log-Odds-Ratio in Stratified 2 × 2 Contingency Tables. Communications in Statistics. Theory and Methods, 41(18), 3355–3370. https://doi.org/10.1080/03610926.2011.560779

Brown, L., Cai, T., & DasGupta, A. (2001). Interval Estimation for a Binomial Proportion. Statistical Science, 16(2), 101-117. Retrieved April 24, 2021, from http://www.jstor.org/stable/2676784

Carrig, B., Ray, M., & Roth, J. (2019, October 30). In-memory database systems features and technologies - SQL Server. https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-database?view=sql-server-ver15.

Chapple, C. (2021, January 29). *Supercell Celebrates Another Unicorn as Brawl Stars Passes $1 Billion.* Sensor Tower. https://sensortower.com/blog/brawl-stars-revenue-one-billion.

Conley, K., & Perry, D. (2013). How Does He Saw Me? A Recommendation Engine for Picking Heroes in Dota 2. http://cs229.stanford.edu/proj2013/PerryConley-HowDoesHeSawMeARecommendationEngineForPickingHeroesInDota2.pdf.

DigitalOcean. (2019, June 12). *Managed Postgres Clusters: pg_restore errors for superuser privileges*. DigitalOcean. https://www.digitalocean.com/community/questions/managed-postgres-clusters-pg_restore-errors-for-superuser-privileges.

DigitalOcean. (2020, October 15). *How to add my DigitalOcean App as a trusted resource for my Managed Database*. DigitalOcean. https://www.digitalocean.com/community/questions/how-to-add-my-digitalocean-app-as-a-trusted-resource-for-my-managed-database.

DigitalOcean (2021, May 3). *Deploy Error: Health Checks Common Causes App is running slower than expected Component Issues brawlstars - failed to deploy*.

Florian Block, Victoria Hodge, Stephen Hobson, Nick Sephton, Sam Devlin, Marian F. Ursu, Anders Drachen, & Peter I. Cowling. (2018). Narrative Bytes: Data-Driven Content Production in Esports. *In Proceedings of the 2018 ACM International Conference on Interactive Experiences for TV and Online Video (TVX '18). Association for Computing Machinery*, New York, NY, USA, 29–41. DOI:https://doi.org/10.1145/3210825.3210833

Galante, Guilherme, Erpen De Bona, Luis Carlos, Mury, Antonio Roberto, Schulze, Bruno, & da Rosa Righi, Rodrigo. (2016). "An Analysis of Public Clouds Elasticity in the Execution of Scientific Applications: a Survey", *Journal of Grid Computing*, *14*(2), 193–216. https://doi.org/10.1007/s10723-016-9361-3

Gordon, K. (2020, September 24). *You Think Your Favorite Video Game Is Hard? Try Speedrunning It.* NPR. https://www.npr.org/2020/09/24/916167970/you-think-your-favorite-video-game-is-hard-try-speedrunning-it.

Haselmayr, M. (2015, November 7). *65 Simple Ways To Promote Your Mobile App*. Forbes. https://www.forbes.com/sites/allbusiness/2015/11/07/65-simple-ways-to-promote-your-mobile-app/?sh=1655b2cc48c4.

Hearth, C. (2019, September 17). *Advanced Postgres Performance Tips*. thoughtbot. https://thoughtbot.com/blog/advanced-postgres-performance-tips.

Hodge, Victoria & Devlin, Sam & Sephton, Nick & Block, Florian & Drachen, Anders & Cowling, Peter. (2017). Win Prediction in Esports: Mixed-Rank Match Prediction in Multi-player Online Battle Arena Games.

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, *16*(3), 261–273. https://doi.org/10.1016/j.eij.2015.06.005

Oliver, Mary Beth, Rogers, Ryan, Sherrick, Brett, Woolley, Julia, Bowman, Nicholas David, & Chung, Mun-Young. (2016). In control or in their shoes? How character attachment differentially influences video game enjoyment and appreciation. Journal of Gaming & Virtual Worlds., 8(1), 83–99. https://doi.org/10.1386/jgvw.8.1.83_1

Reiczigel, J. (2003). Confidence intervals for the binomial parameter: some new considerations. Statistics in Medicine., 22(4), 611–621. https://doi.org/10.1002/sim.1320

Roundhill. (2020, September 25). *Esports Viewership vs. Sports in 2020*. Roundhill Investments. https://www.roundhillinvestments.com/research/esports/esports-viewership-vs-sports#:~:text=Viewership%20has%20been%20growing%20rapidly,total%20viewership%20grew%20by%20195%25.

Sifa, R., Pawlakos, E., Zhai, K., Haran, S., Jha, R., Klabjan, D., & Drachen, A. (2018). Controlling the Crucible: A PvP Recommender Systems Framework for Destiny. *ACSW '18: Proceedings of the Australasian Computer Science Week Multiconference*, (39), 1–10. https://doi.org/10.1145/3167918.3167926

Sauro, Jeff, & Lewis, James R. (2005). Estimating Completion Rates from Small Samples Using Binomial Confidence Intervals: Comparisons and Recommendations. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 49(23), 2100–2104.

Schafer, C. M. (2019, January 24). CoreyMSchafer/code_snippets. Flask Blog. https://github.com/CoreyMSchafer/code_snippets/tree/master/Python/Flask_Blog.

Szymańska, A. E. (2018). "The Application of Buhlmann-Straub Model with Data Correction for the Estimation of Net Premium Rates in Bonus-Malus Systems of the Motor Third Liability Insurance", *Acta Universitatis Lodziensis.Folia Oeconomica,* (336), 7-21. http://dx.doi.org/10.18778/0208-6018.336.01

W3Schools. (2021). CSS Tutorial. https://www.w3schools.com/css/.

Yuan-tao Xie, Zheng-xiao Li, Rahul A. Parsa. (2018), "Extension and Application of Credibility Models in Predicting Claim Frequency", *Mathematical Problems in Engineering*, vol. 2018, Article ID 6250686, 8 pages. https://doi.org/10.1155/2018/6250686