Aman Bhardwaj, Domitille Chambon, Blake DeLong, Colby Meline, Joel Nail

BDS Final Project Report

Problem Statement

In today's video game industry, the number of units a game sells is important, but the number of concurrent players is arguably more important - especially as the prevalence of Free to Play games continues to grow. Free to Play games rely on a highly active user base to continually provide revenue over time via in-game transaction. This demands high player retention in order for the developers' to achieve a sustained source of revenue. For the purposes of this report, we will refer to a game's ability to retain a high number of active users over time as *longevity*. Put simply, our problem statement asks the question "how can game developers predict their game's longevity via user-generated content?"

Hypothesis

We believe that by using the reviews of games, we can create a model that will accurately predict the longevity of a game's playerbase i.e., how well the game retains its players over time. Using Spacy's word2vec capabilities, we can transform the text of game reviews into usable data to predict whether a game will be above-average at retaining players year-after-year.

Data Source

Steam DB provides a list of the 5000 top games by player count. Steam DB provides a game's all-time peak player count, as well as the latest 24 hour peak player count. Given the game's age, we can calculate a longevity score that consists of how much player base they have lost since their peak, controlling for the length of time since the game's release. The reviews were retrieved from Steam's partner API, which allows end users to request reviews for a given App ID.

Methods

Longevity Formula: $\left(1 - \frac{All\,Time\,Peak - 24\,Hour\,Peak}{All\,Time\,Peak}\right) \times \left(\frac{\sqrt{Age}}{Max\left(\sqrt{Age}\right)}\right)$

For our implementation, we decided to predict whether or not a game's longevity would be above or below the median longevity of our data set. In order to calculate longevity, we derived the above formula. To create this formula, we first started with the depreciation of player base:

$\frac{All\,Time\,Peak - 24\,Hour\,Peak}{All\,Time\,Peak}$. This number describes the percentage of players that have left the game from its all time high player count to its current player count. As the depreciation percentage increases, we consider the longevity of the game to be worse. However, we also wanted to consider the fact that, as a game gets older, it is expected that a player base will naturally decrease. Because we are looking to predict the longevity of a player base regardless of age, controlling this variable by including it into our longevity calculation is critical. The term $\left(\frac{\sqrt{Age}}{Max\left(\sqrt{Age}\right)}\right)$ represents the age of a game (in years) that is normalized. As the age of a game increases, the depreciation percentage decreases, meaning that we consider the longevity to be better. Finally, we need to consider the edge cases. If we just consider this as our equations, $\left(\frac{All\,Time\,Peak - 24\,Hour\,Peak}{All\,Time\,Peak}\right) \times \left(\frac{\sqrt{Age}}{Max\left(\sqrt{Age}\right)}\right)$, then we will encounter issues when older games are deserted, meaning the 24 Hour Peak is 0. When this happens, the depreciation percentage is 1 and the age factor is still applied. This leads to the undesired consequence of some older deserted games still having a longevity score of above the median, contradicting the intuitive and desired result. To counter this, we subtract 1 by the depreciation percentage, ensuring that deserted games will have a longevity score of 0. With our longevity formula now in hand, we calculated it for every game and found the median value. If a game had a longevity score above the median it was assigned a 1 for being a “high longevity game” and a 0 if it was below the median and considered a “low longevity game”.

With our target binary longevity classifications established, we are now ready to pre-process the scraped review data to attempt to train our model. To do this, we tried different ways. We input our text data into Doc2Vec processor from gensim. This library is like Word2Vec but it accommodates for various writing styles and in our case that was important because we have thousands of reviewers all with different writing styles. We got good results without any tuning at 0.69 AUC and 69% accuracy using Doc2Vec. We also utilized spacy vectorizer to compare results. We used the medium sized model due to time constraints and got an AUC of 0.78 and an accuracy of 97%. We also incorporated the genres in our model design by one hot encoding the genres and tags. This increased our accuracy a lot and so we decided to go with the SpaCy vectorizer and including the genres to train our model.

Insights

During the course of our project, we discovered that the text of reviews for video games on the Steam store do have predictive power when it comes to classifying games for longevity. As such, it would be useful for game developers to closely analyze the text of the reviews that their games receive - especially early on in the game’s life.

We were able to achieve even better accuracy by performing a word frequency analysis and identifying user-defined features within the reviews. These features describe the most important aspects of the games such as their gameplay, graphics, music, and more. By identifying these aspects, we were able to increase our predictive power even moreso. This suggests that - while the overall review text is valuable - the most important information must be extracted from the reviews in order to improve predictions.

We recommend that game developers pay close attention to the specific words that customers use when reviewing their games. In order to ensure a consistently high player count of their game, developers should promote game features that are associated with high longevity e.g., playtime, price, etc. This will also require a sentiment analysis of comments to determine whether these important keywords are being referenced positively or negatively.

Reflections

This project was a great learning experience in building upon a system/dataset that was designed for other purposes. In industry, it is common to build upon other projects that originally had a different goal, so we believe that this project is a great example of utilizing a pre-built system for novel purposes. We were also excited to work on this project because it allowed us to combine multiple datasets in order to develop our model. Most assignments include a easy-to-use dataset which makes the process of developing a model much easier compared to scraping data from an online source. We found this aspect of the project to be quite interesting although it did introduce some difficulties e.g., missing data points.

This project also gave us the opportunity to practice our modeling skills, specifically with CatBoost. We were not sure what to expect when initially running our vectors through CatBoost, but we were extremely happy with the results it produced. We believe that by refining our model and preprocessing steps, we could further improve the accuracy of our predictions and produce a model that can effectively forecast the longevity of a game early in the game's life. We believe that there is great value in a model that can achieve this goal, so there is a defined business use case for this concept. Allowing developers to predict the longevity near the beginning of its life cycle would inform important business decisions such as investment in post-launch support and additional expansion packs if the outlook looks positive. Alternatively, a developer could potentially save money by pivoting away from a faltering game. All in all, this project was a great experience that provided insightful results.

Aman Bhardwaj, Domitille Chambon, Blake DeLong, Colby Meline, Joel Nail

Sources:

1. https://www.cnet.com/home/smart-home/why-most-video-games-arent-profitable/
2.