# Predicting Purchaces in Steam

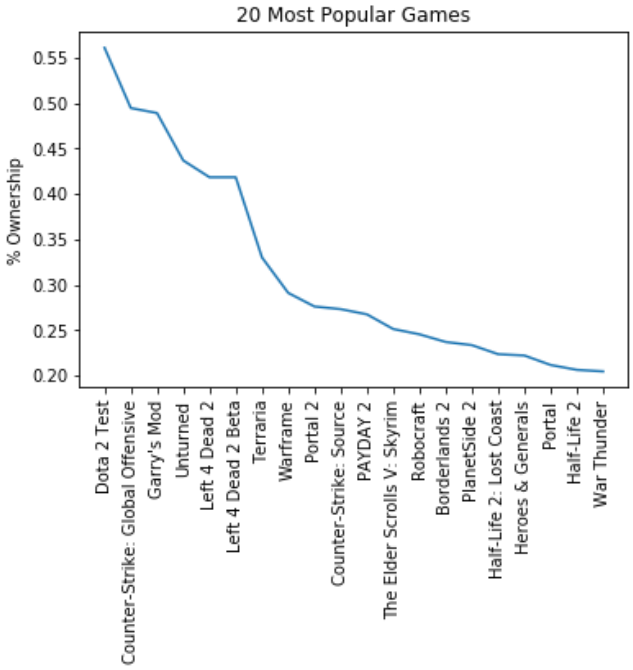
CHRIS WYNNE*, THOMAS QIN, and PETE SHEURPUKDI, UCSD Undergrads

As lifelong gamers, we decided to look into Steam, an online video-game platform for buying and playing games. Users can purchase individual games or bundles of games online and have a *library* of games to play. In this assignment, we seek to understand how individual users add games in to their libraries. In particular, the problem we want to solve is given a user and a pair, predict whether they will own the game or not. Of course, this is not particularly relevant for the present, where checking whether someone owns a particular game is really simple. However, we can use this to help recommend certain games to users as well as help developers understand which users they can market their games toward in the future.

**ACM Reference Format:**
Chris Wynne, Thomas Qin, and Pete Sheurpukdi. 2018. Predicting Purchaces in Steam. *ACM Trans. Graph.* 37, 4, Article 111 (August 2018), 4 pages. https://doi.org/10.1145/1122445.1122456

## 1 INTRODUCTION

The main goal of this Assignment was to understand how we can most effectively predict what games a particular user of Steam is likely to purchase. The dataset we used was provided by Julian McAuley and contains 5 million tuples of the following format (user, game, playtime of the game, and play time of the game in the last two weeks). It contains data from Steam users in Australia that was crawled by the authors in the first reference. There are plenty of interesting trends in the data, with a significant proportion of the users in our playerbase owning the most popular games. In particular, over half of the users we examined owner the game "Dota 2 Test" and half the users we examined owned the game "Counter Strike: Global Offensive" Not surprisingly, both of these games are free. However, one surprising insight is that over two-fifths of users own "Left 4 Dead 2", which costs $9.99. Below is a simple graph we constructed with the most popular games and the percentage of the users that own the game [1-3]. We expected an exponential dropoff in terms of popularity for games and generated a quick chart to verify our results, seen on the next page. Surprisingly enough, many games reach a massive chunk of the Steam playerbase even though the games required quite a bit of investment to play. To us, it wasn't even obvious that a majority of Steam users would have even heard of games such as "Left 4 Dead 2" or "Terraria" but both of these games cost money and over a quarter of the users in our dataset own the game. There was a huge discrepancy between




what we expected, with the most popular games being owned by at most a tenth of the community and reality, where there exist these massively popular games that a giant subset of the community own. There is no geographical spread of this data, but some basic exploratory aggregation reveals some interesting information about users of Steam. The mean number of games a user owns is 58 with a median of 26, suggesting the existence of "hardcore gamers" who individually own a huge number of games. Even so, a median of 26 games owned by a user seems high. For the games, the mean playtime was 990.5 minutes, which is almost 17 hours with a median of 34 minutes. This means for at least half the the user-game pairs, the user gets bored of the game after a short period of time and does not continue to play the game, and will possibly never bother to delete it. This paints a pretty interesting picture of an "average" Steam user, who owns many games, but only plays a small subset of them or any significant amount of time. Still, the interesting thing about these games is that lots of paid games are sitting in a user's library but not getting much playtime

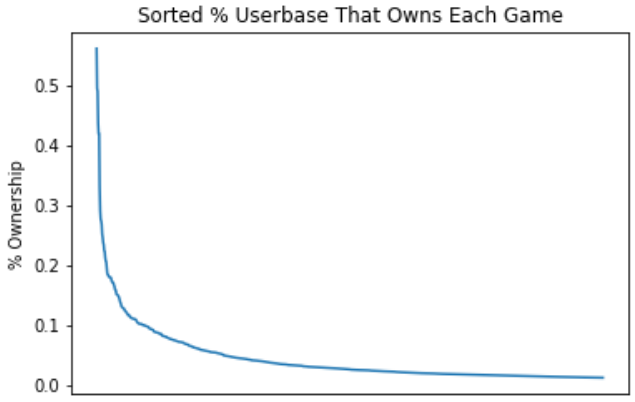
## 2 IDENTIFYING A PREDICTIVE TASK

With our picture of an average steam user in mind, we thought it would be interesting to try our own approach to a basic recommender system on Steam that was suitable for an undergraduate student. There are some highly influential works that have come before us that helped us confirm our initial exploratory analysis, but have methods that are not particularly suitable for our limited

*All three authors contributed equally to this research.

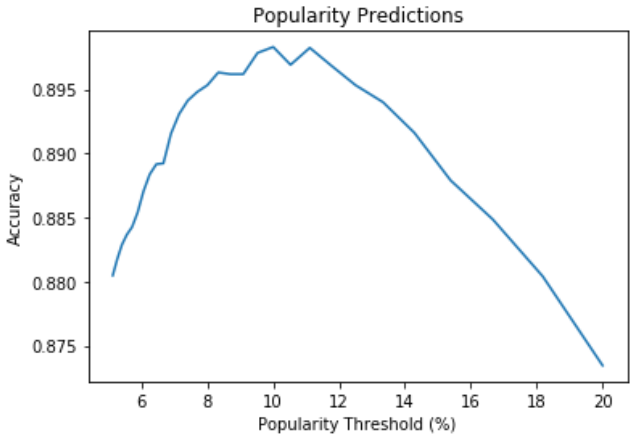
Authors' address: Chris Wynne, cwynne@ucsd.edu; Thomas Qin, tzqin@ucsd.edu; Pete Sheurpukdi, psheurpu@ucsd.edu, UCSD Undergrads, 9450 Gilman Drive, La Jolla, CA, 92092.


0730-0301/2018/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456




computing resources [4]. We decided an interesting task we would make a meaningful amount of progress on a problem similar to one we've tackled in Assignment 1, where we predict whether a user owns a particular game or not, instead of predicting whether a user has read a particular book or not. This is more interesting than the question in Assignment 1 because not all users who have read a particular book would post a review on GoodReads, which means we are not actually predicting whether a person will read a book, but rather whether a person will write a review about a book. We hope to extend our work to predicting the hours played for a user in total as well in the future, using Mean Error as a metric. The data for playtime do not follow a normal distribution so Mean Squared Error does not make sense as an error metric because for a given game, most of the users get bored and go play something else after a short time.

We wanted to establish some sort of baseline for how our model would do. The first thing we tried is to make a predictor that was comparable to the baseline that was given in Assignment 1. In particular this would look like generating a set of the most popular games and if a given game is in this set we would predict that the user owns this game. We have the baseline results in the chart on the next page. Now we just needed some sort of validation set to test this on. In order to generate this validation set, we would use a similar approach for what was instructed in Assignment 1. We took one tenth of the users in our dataset and seperated them for validation. We could have chosen the exact same approach we took in Assignment 1 where we just partition a tenth of user game pairs, but this ended up creating a significant subset of users who had no games owned and our validation set would be too imbalanced. So we used a random one tenth of our users to be part of our validation set. Of each of these users, we selected one game that they played for a positive user, game pair in our validation set. For the negative examples in our validation set, we would randomly sample a game that this user has not played and add it to our validation set. We tried our naive model parameterized by some popularity threshold p. In our naive model, we would compute an ordered list of the most popular games by the number of users and take the top p percent of that list and add them to our set of popular games. For each entry in the validation set, we would ignore the user, and look at the game. If the game was in the set, we would predict that the user would own the game. If not, then no. This turned out to perform exceedingly well on our validation set. with thresholds of p and accuracy in the figure below In order to build our model that we would use, we




would need to be able to use some more advanced techniques specific to Recommender Systems. In Assignment 1, Chris was able to beat the strong baseline by using multiple forms of Jaccard Similarity as well as exploiting the fact that the validation and test set would contain the same number of "yes" and "no" labels. We also wanted to use the additional information such as the hours played forever for more information. We did a significant amount of preprocessing of data to compute multiple forms of Jaccard similarity that will be discussed in detail in part 3.

## 3 DESIGNING OUR MODEL

We had a very competitive baseline model to beat that just predicted based on whether the game was in the most popular x percent of games or not. We also wanted a baseline to beat for a basic Jaccard similarity. We decided to use this model as a non-trivial baseline for Jaccard. Given a user-game pair, get a list of all users U who own this game. For each user in U, we compute the Jaccard similarity between our given user and the individual user in u. We take the max of these results and store this in a variable. We then need some threshold to compare this Jaccard similarity to. To generate this threshold we iterated over all games, and for each game compute the max Jaccard similarity between users who played that game. Of these Jaccard Similarities, would take the arithmetic mean of these and set it as our threshold to beat. This ended up doing quite well, scoring an accuracy of .841, which is strong but not as strong as the naive baseline but we thought it would be interesting to see if we could beat it by any significant amount. To beat this model we would have to think of a more clever way to process a given user, game pair. We decided to compute multiple forms of Jacccard similarity and represent this as a feature vector to train a Logistic Regression model on. The forms of Jaccard similarity in our model are the following.

### 3.1 owned-user-sim

This is where we would fix a game and compute the max Jaccard Similarity between users who have owned the game. This was the sole feature in the model to beat for our baseline

### 3.2 played-user-sim

This is where we would fix a game and compute the max Jaccard Similarity between people who have played the game. We thought this would be interesting because many people end up buying games to never play them.

### 3.3 sig-played-user-sim

This is where we would fix a game and compute the max Jaccard similarity between people who have played the game for more than 10 hours. 10 hours was an arbitrary but slightly educated guess on what would count as played "significantly" since a lot of the users would play games for under an hour. We hope to try multiple version of the model with different parameters but the model currently takes 8 hours train and validate and we are out of cloud computing credits. We hope to do more work in the future parameterizing this.

### 3.4 recently-played-user-sim

This is where we would fix a game and compute the max Jaccard similarity between people who have played the game recently. We thought there would be interesting temporal trends we would be able to capitalize on.

### 3.5 owned-game-sim

We fix a game and compute Jaccard similarity between users who also own the game

### 3.6 played-game-sim

We fix a game and compute Jaccard similarity between users who have played the game

### 3.7 sig-played-game-sim

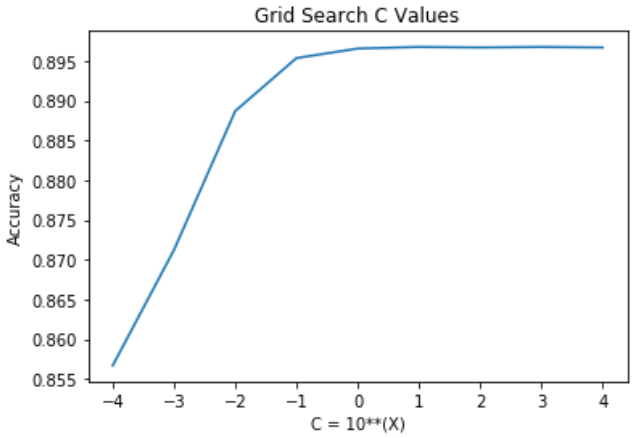
We fix a game and compute Jaccard similarity between users who have significantly played this game

### 3.8 recently-played-game-sim

We fix a game and compute Jaccard similarity between users who have recently played the game

With these features as a vector, we used Logistic Regression to train a model on our training set, with an equal number of negative options added for each of the positive user-game pairs and came with a model with an accuracy of .899, which is significantly better than our baseline Jaccard similarity of .841. Oddly enough, it was as good as the predict the most popular games classifier but we did not include that feature as part of our model on purpose. We wanted to be able to make a model that would actually have output some "yes" labels on some relatively unpopular games even if some of the only positive unpopular games we had "yes" labels on are on games of the same series or games by the same studio. We are very happy that we were able to correctly label some unpopular games.

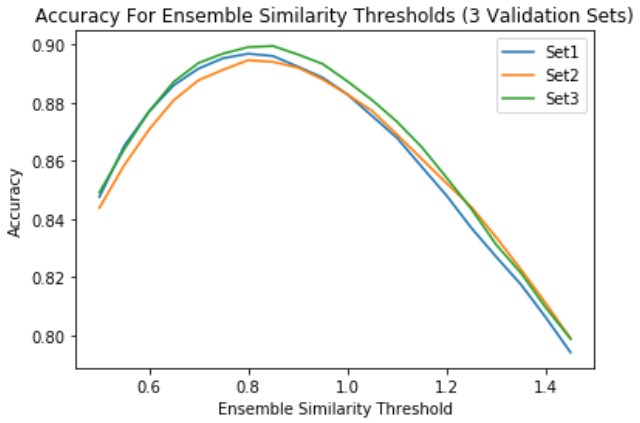
We could have added more features but we really wanted to be able to beat our naive baseline without using popularity. We are currently training a version of the model that takes popularity into account but will take some time to tune properly. It wasn't obvious to us how to use cosine similarity in any way in this model since vectorizing a game given a user or a user given a game don't really make sense unless we use some massive one-hot-encoding of every user/game which is inefficient and wouldn't tell us anything our Jaccard similarity doesn't already tell us. One more thing we chose not to try was Pearson similarity since there was no notion of a "rating" that a particular game got, just a number of hours played, which probably would not have been useful because of how often there are games that are purchsed but never played. One thing we could try in the future is an SVM since there probably are some relatively easy to classify examples where the Jaccard similarity is very very high between two users or games that potentially skew our error metric. We had a hunch it would perform better than our Logistic Regression model but it ended up quite a bit worse. There were some issues training the model since computing Jaccard similarity across our training set took quite a bit of time, as well as doing the same thing for our validation set. We would have to leave the model running overnight in order for the data to be properly represented as a feature. We also did some cross-validation with 3 different training sets (we were going to do 10-fold cross validation but couldn't because of time) and we made sure to tune our logistic regressor c values in our charts shown below




Our error across validation sets stayed relatively similar and we were quite happy with this.

A couple of key takeaways were that our naive model that just predicted based on popular games would perform quite well but doesn't really perform well on games that aren't popular. The nice thing is that it is incredibly quick to train and still has amazing accuracy.

Our Jaccard based model was quite good in terms of beating our nontrivial baseline for Jaccards similarity but took an incredibly long time to train even with the optimizations discussed in lecture. One notable thing we did was to "save" the max Jaccard similarity for one user game pair whenever we computed it for a certain one of our




features so we could just do a table lookup instead of recomputing multiple values and taking the max of them but that simply made the computation possible, not necessarily practical. We chose not to try Naive Bayes since there was the clear to all of us that we do not have the conditional independence assumption with Steam games.

Some issues we ran into with missing data included users that didn't have many games so generating a validation and the rest of the training set would be a bit tricky but just thinking about the dual version of the problem usually would be sufficient to unblock us.

## 4 RELATED LITERATURE

There were two notable papers we found, notably [1] and [4]. [1] relates to how to how to obtain the data through as well as interesting work relating to how to personalize the bundle feature that is in Steam, where users would buy multiple games at once at a discount. Another paper we found [4] shed some insight on the average user that was on Steam. It used clustering mechanisms like k-means to observe time played across various games. One surprising result from [4] was that almost two-fifths of users on Steam spread their time across multiple clusters which made us believe that something like a proper recommender system on Steam would be extremely valuable given how open a big part of the Steam community are to trying games in different clusters. It solves a similar problem to the one we are solving but with a prefix tree and association rules. We found one other paper that we found interesting but not much more insightful than [4] in showing that playtime for games across Steam can be represented in an exponential distribution.

## 5 RESULTS

Although our naive model performed quite well, we had more interesting results in our Jaccard similarity model. In particular, we got to see some insight on values of theta and see how important certain features were in terms of building our model in the first place. It turns out Jaccard similarity between fixing a game and computing the Jaccard similarity between users was the most effective. We were not surprised that the Jaccard similarity of users playing the game was also relevant in our model. We were surprised the looking at users that recently played the game was not very relevant but then after looking through the data more there were not too many positive values of recently played to begin with, especially across some of the more unpopular games. Other than that, the other features seem to add some nontrivial predictive ability to the model. In the future we hope to train our model on a very powerful VM when we get sufficient cloud compute credits. We also want to try more tuning with our model parameters, in particular trying this on a larger dataset on people with different demographics.

| | Theta |
|---|---|
| owned_user_sim | 3.881952 |
| played_user_sim | 2.047247 |
| sig_played_user_sim | 3.303169 |
| recent_played_user_sim | 2.406156 |
| owned_game_sim | 7.452165 |
| played_game_sim | 5.612925 |
| sig_played_game_sim | 3.037179 |
| recent_played_game_sim | 0.228963 |

## 6 REFERENCES

[1] Generating and personalizing bundle recommendations on Steam Apurva Pathak, Kshitiz Gupta, Julian McAuley SIGIR, 2017

[2] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In ICDM, pages 197–206. IEEE, 2018.

[3] Item recommendation on monotonic behavior chains Mengting Wan, Julian McAuley. RecSys, 2018

[4] Sifa R, Drachen A, Bauckhage C (2015) Large-scale cross-game player behavior analysis on Steam. In: Proceedings of the 11th artificial intelligence and interactive digital entertainment conference (AIIDE). AAAI

[5] F. Baumann, D. Emmert, H. Baumgartl, and R. Buettner, "Hardcore Gamer Profiling: Results from an unsupervised learning approach to playing behavior on the Steam platform," Procedia Computer Science, vol. 126, pp. 1289–1297, 2018.