PREDICTING REVIEW EFFECTIVENESS OF GAMES ON STEAM

ZACHARY EVANS

Abstract. The author takes a closer look at Steams weighted vote score, a score steam created to determine how useful a review is, and determines if it can effectively be modeled by regressions. The author notes that the data associated with steam reviews is a bit chaotic and it may be hard to determine how closely associated the variables are. This is effectively shown by how poorly the data does with a linear regression. However it is then shown using a decision tree to make these evaluations is able to show a possible model that can determine Steams weighted vote score. The author concludes it is possible to model this score effectively and explores some negative ethics that may be associated with the information given in this data set and with these conclusions

1. Research question and overview of the data

Throughout the history of mankind, reviews have influenced the masses to determine what products are popular or good. With the introduction of the internet, these reviews are even more widespread and influential. Our goal is to try to figure out what correlations exist between helpful reviews and other attributes that happen at the time of the review.

We have found a dataset[1] containing information about Steam reviewers and their reviews. The dataset from Kaggle contains the features in Table 1. The primary information that we are going to be using are the reviewed game title, number of games owned, number of reviews given, whether or not the review was seen as funny, voted up, and the weighted vote score. The weighted vote score is one of the primary values we will take a look at since it is a custom value of this dataset to determine how helpful a review is. Additionally, we are given a votes up feature which determines if the review was positive or negative. We will use these features to answer the following questions.

- What features have the highest correlation with Steam's weighted vote score.
- Why do we think Steam emphasizes these features within their scores?
- What regressions can effectively identify the steam weighted vote score?

Date: 24 November 2026.

While investigating what has already been done to identify what makes an effective review, we have found that there has been some investigation based on amazon reviews done empirically[2]. The difference between that paper and ours is that we are restricting ourselves to the specific category of video games being reviewed on steam, while they focused on the more broad categories featured within amazon data.

2. Data Cleaning / Feature Engineering

When looking at the dataset, we wanted to start by analyzing missing data. To do this, we cycled through the columns and checked how many missing values are in each column. As shown by the code[3], we see that there are only 2 columns with Nan values. The first is games which contains the names of the games. Because there are only 33 games with missing values within our dataset of 498,094 losing 33 points won't be significant. Thus we are going to drop those rows. The second column with missing values is steam_china_location. Luckily this column does not answer significant questions that we want. So we will drop that entire column. The rest of the data looks pretty clean.

Now we will focus on feature engineering. Since we want to be able to look at multiple features per reviewer potentially we will keep the author steamid, and additionally we will keep all information pertaining to games to see if we can make a random forest such that we can see likelihoods of a reviewer reviewing another game. Thus, we want to keep track of both game and app id. Author number of views will be vital to some of our regression in addition to weighted vote score and games owned. Currently we don't care about the actual review so we don't need review and we don't need language. We also don't really care about the time stamps for any of our questions. So we will also drop those columns. Also since we are focusing on the weighted vote score we can drop the votes up number since it is the number of users who found the review helpful. This is not to be confused with the voted up feature which is whether the review was positive or negative. Also currently we are not looking at authors' total playtime so we will also drop those columns. Currently we do not care about steam purchase, so this feature will be dropped as well, although I do believe there are interesting questions that involve it. With these consideration in mind, we updated the columns of our data frame[3].

Due to time constraints and manpower no further feature engineering was done. Now with our data frame updated to support the features we want, we can now analyze some of the basic information we have.

3. Data Visualization and Basic Analysis

For basic data analysis we want to plot as much as possible against the vote score to see if we can find any basic correlations. We have done this in the appendix and will refer to each plot within the section. First we will analyze the plots in figure 1. In figure 1 we see that we have plotted both the Vote Score vs the Number of games owned by the reviewer and the Vote Score vs the Number of Reviews the Reviewer has made. At first glance the data is very chaotic with so many data points, however in both cases there seems to be a potential negative corelation. This would mean if true that the more games and the more reviews you own and have made the worse your review will be rated. This however is just a visual estimation and could be wrong.

Next we will analyze figure 2. Figure 2 Shows the reviewer number of hours they have played the game in the past 2 weeks and the number of hours the reviewer has played the game when they posted the review vs their review score. From the data visualization it feels hard to estimate the corelation of the number of hours within the past 2 weeks and the vote score. The data feels very spread out. However, there does seem to be a negative corelation between number of hours played at the time of the review and the vote score. This may be caused by people finding more flaws in a game the longer they play it.

In Figure 3 we analyze the time the player last played the game and the vote score. First we will notice there are many vote scores at 0 showing a variety of reviewers who may have bought the game on other consoles. We will prioritize looking at the right hand side of the graph which shows a positive correlation to reviews and the last time the game was played. Meaning the longer a game has been out there is a higher likely hood that the review is more helpful. Taking a look at how many people have voted the review funny or not we can see a positive correlation between the Vote Score and the number of votes determining if it is funny. The data is still a bit muddled though due to the volume.

The last figure 4 shows the number of comments that the review has vs the vote score This also seems to have a positive correlation but seems more ambiguous than the voted funny figure.

With the visualization done I predict a high corelation between the number of comments and if the review was voted to be funny. I also predict a strong corelation between number of hours that the reviewer has played and the review.

4. Learning Algorithms and In-depth Analysis

Now that we have a basic understanding of our analysis we are going to try to use regression to see if we can find a model that will accurately find our Vote Score. We decided it would be best to start with a linear regressor. We decided to make a train test split of 67% and 33% since this would allow us to have a large pool of data to train on and prevent overfitting. After training multiple times the linear regression seems a bit unstable. An R^2 score of -1.23×10^7 appears occasionally while other times the score is .03. After 1000 iterations of test train splits and training we have a mean of

-1.04×10^7 with a max score of .03756 and a min score of -1.23×10^7. This shows how unstable the linear regression is. This seemed to be obvious since our data does not look linear. Additionally some of our features or boolean based so this would effect the linear regression. We decided to move to decision tree regression.

To make sure we have as good of a decision tree as possible, we started by doing a parameter grid search to find the best parameters to minimize squared of predicted weighted value score. We made a parameter grid search over a max depth of 1, 3, 5, and 10 with a splitter that is either "best" or "random". The grid search also utilized a 5 section cross validation. After running our grid search we found that the best parameters were a max depth of 10 and using the "best" splitter. After running our tree regression based on our best parameters, we see that we have a mean squared error of .00166 on the test data. We do need to note that with the dataset we used we only have vote scores between .80 and 1.00, however getting a mean squared error of .00166 is indicative of model effectiveness.

Since this regression seems to work pretty well, we decided to look at what our regression thought were the most important values when determining the vote score. We did this using scikit learn regression attribute `value_importance_`, which gives a list of each of the features that are important according to our regression. Then we make a dictionary with the feature importance and its correlated feature and sort them. We then get Table 2.

When looking at table 2 we have some very interesting features that are emphasized. First, we see that the highest correlating feature is whether or not other users found the post funny or not. This makes sense since if a post is funny it means many users found the post insightful enough to vote it as funny. With a feature importance of 65% it seems very heavily weighted towards such. What I find most interesting is that the second most important feature was recommendation id. The reason why is this was Bias essentially since each recommendation should have a unique recommendation id. So bias being the next most important thing is extremely confusing to me. You will note the number of comments on a review is the next most closely correlated feature to the weighted vote score. This also makes sense since similarly to when a post was voted as funny or not, we are able to see interaction with others to determine how helpful a review is.

5. Ethical Implications and Conclusions

While it has been great having this information, steam is largely public with enough information it may be possible to narrow down who people are who left these comments and reviews based on (the full comment they gave), their location, and language which both were given in this data set. This could be a security issue and may give too much information to those who may have malicious intent. Additionally, had the experiments gone better

I wonder if it would allow LLM's be able to create reviews automatically which would hinder the review process for the video game industry.

When analyzing our model we can clearly see that the most impactful feature for the helpfulness of a review according to steams weighted vote scale is whether or not the review was voted funny. It was also emphasized that the number of comments on a review also impacts a review. This shows that other user feedback helps determine how helpful a review is. It does seem like it is possible to effectively model how helpful a review in a similar way that steam determines this. For those who would like to continue the research, feature engineering more features based on games similarity (such as genre) and seeing the impact that has on reviews could be beneficial.

6. Appendix

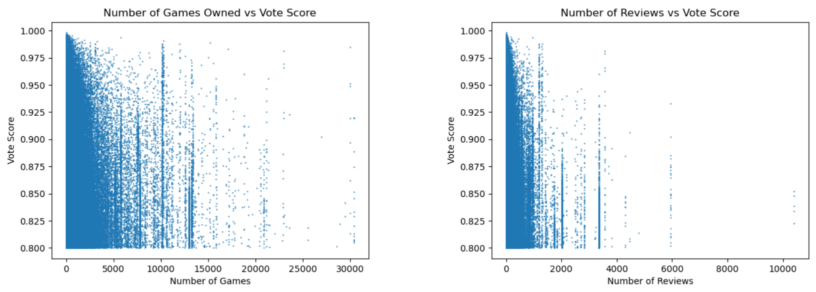

Figure 1. These plots display both the rating of the review vs the number of games the reviewer owns and the number of reviews the reviewer has written

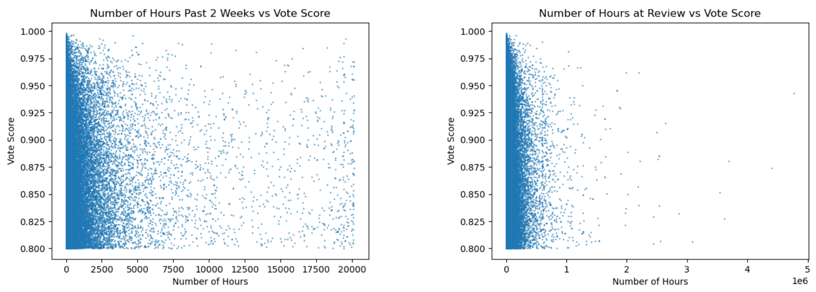

Figure 2. These plots display what the total number of hours were played in the last 2 weeks and the number of hours played at the time of the review vs the vote score.

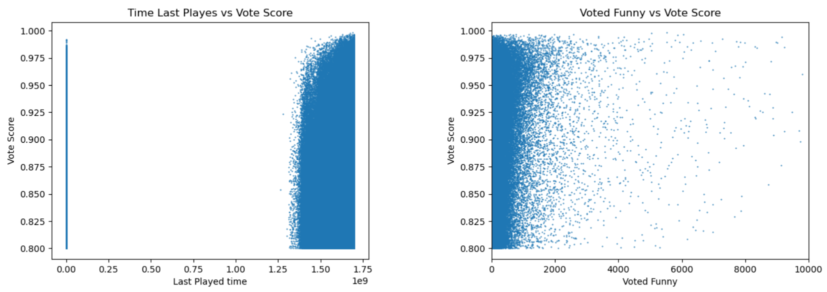

Figure 3. These plots display the last played time in unix time and if the review was voted funny vs the vote score

Attribute	**Description**
recomendationid	The unique id for the review
appid	The unique id for the game
game	The name of the game
authorsteamid	The unique id for the author
authornumgamesowned	The number of games owned by the author of the review
authornumreviews	The number of reviews made by the author
authorplaytimeforever	The number of minutes the author has played this game
authorplaytimelasttwoweeks	The number of minutes the author has played the game in the last 2 weeks
authorplaytimeatreview	The authors play time when the review was written
authorlastplayed	Unix timestamp of when the reviewer last played tha game
language	String of what language the review was in
review	The string of the review
timestampcreated	Date the review was created in unix timestamp
timestampupdates	Date the review was last updated in unix timestamp
votedup	True or False determining with true being a positive recomendation
votesup	The number of users that found the review helpful
votesfunny	The number of users who thought the review was funny
weightedvotescore	A steam generated score deciding how helpful the review is
commentcount	The number of comments posted on the review
steampurchase	True or false if the user bought the game on steam
recievedforfree	True of False if the user received the game for free
writtenduringearlyaccess	True or false if the user reviewed it during early access
hiddeninsteamchina	If the review is invisible in china
steamchinalocation	If the reviewer is in china or not

Table 1. List of all Features within the dataset and descriptions

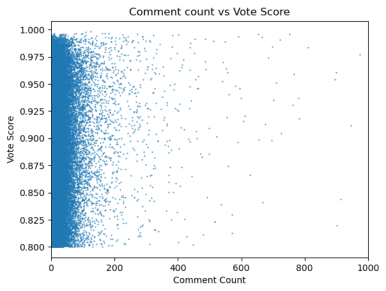

FIGURE 4. This plot is the number of comments on the review vs the vote score

Feature	**Feature importance %**
author_steamid	0.0
hidden_in_steam_china	00.0004300266
written_during_early_access	0.0418406481512941O3
received_for_free	0.07619733947244545
author_playtime_last_two_weeks	0.14451186124826354
author_num_games_owned	0.9364881882716279
author_last_played	1.1476465079212508
author_playtime_at_review	1.226864628660263
appid	2.4159055543840527
voted_up	2.5504322282344092
author_num_reviews	2.7730363029840634
comment_count	10.855022318612795
recommendationid	12.40804383820948
votes_funny	65.41971031821333

TABLE 2. List of all Features with there correlating importance

References

[1] https://www.kaggle.com/datasets/kieranpoc/steam-reviews
[2] https://www.sciencedirect.com/science/article/abs/pii/S0167923620301585?via%3Dihub
[3] https://github.com/abuzmander/Zachary_Evans_Reviewer_Analysis