This analysis began with the intent of scraping through the general data of several thousand Steam games. I formed a hypothesis at the beginning of this test that the score of any game on this spreadsheet could not be calculated through the data presented. Namely, I believed that the price and tags on the game itself would be enough to do so if it was possible to predict the score of a hypothetical game by these traits alone.

So, the first thing I did was parse through the price data. Any game that marked itself as free was instead given the value of 0.0 and using regular expressions, I was able to give every game with a listed price a valid number before removing any that were left blank.

Developers and Publishers would have been easy, but for some games, more than one publisher or developer was listed, so I instead made the model check how often each repeated. Giving initially a low threshold (1 each) to determine if the Dev/Publisher was Known or Unknown. These numbers increased with each iteration until their relevance tapered off in subsequent iterations.

The "categories" field was much more difficult to wrestle with. The obvious choice was to repeat what I'd done with the developer and publishers; making a one-hot encoding list of columns after parsing through the given tags. There was one issue with this, however. The tags had no dividers, so I needed to find as many relevant exceptions as I could that were statistically significant enough to be included, added them as an exception so they'd be added immediately and were checked for, and then began parsing through to add more relevant ones with each iteration.

After finishing with categories, I dropped any that left my one-hot encoded columns empty (no categories) and then trained my data using the relevant factors thus far, using each game's "all reviews" percentage as the number being trained against.

Once the dataset was cleaned, and all the data was separated, I decided to run a hyperparameter check for a Random Forest and XGBoost model, trying to find the best values for each so they could be used in subsequent iterations. I also added a Linear Regression as a point of comparison and to check if it may yeild better results. After multiple iterations, Random Forest performed modestly, achieving an R^2 score of 0.03, while XGBoost showed a slight improvement with an R^2 of 0.04. Linear Regression, however, consistently returned a negative R^2 score. To better understand the models and make alterations for each iteration, feature importance was analyzed for Random Forest and XGBoost, with the top 25 features visualized to reveal which factors influenced predictions the most.

Despite multiple iterations and many alterations to the "categories" data, the improvements in R^2 were minimal. This was likely due to several challenges. The one-hot encoding of hundreds of categories introduced high dimensionality, which diluted the contribution of individual features. Many categories were only weakly correlated with user review scores or barely showed up, limiting their predictive value. Additionally, factors influencing user reviews, like game quality or marketing, are more intangible in the first instance and not freely available in the data sheet, making it difficult for the models to achieve high accuracy. Furthermore, "High Quality" was a tag, but when used in earlier iterations, it *decreased* the accuracy of the model down to an R squared score of 0.0, which helps to prove my initial hypothesis. Many of these tags are descriptors given by the publishers or developers and a game that "self-describes" itself in the marketplace will always be less reliable than direct, critical response. If I wanted to pursue this line or try again in the future, I'd need a much more precise data sheet or one that could separate user reviews from critical ones to see if there may be a correlation there that was significant. It would also require much more time to parse through all the existing categories.