Predicting board game user ratings with a machine learning model

Marian Agyby

2022/11/25 (updated: 2022-12-07)

Contents

Summary

In this project we have built and evaluated a predictive regression model that can be used to predict board game user ratings for new board games. The final model performed moderately well on the unseen test data set, with an R^2 score of 0.480. Although promising, our board game rating predictor needs further analysis to be fine tuned and improved in its efficiency and accuracy.

Introduction

According to a small survey on BoardGameGeek, many board game developers report that it takes them several months to develop a board game, sometimes over a year (BoardGameGeek, n.d.b). As board game enthusiasts, we aim to answer the following question: Given certain characteristics about a new board game, can we use a machine learning model to predict how users would rate the board game? Answering this question will help board game creators understand which characteristics enhance user enjoyment and make better decisions as they develop the game, saving them time and improving the popularity of their new board game.

Methods

Data

We used a large data set containing user ratings and reviews for thousands of board games, created by BoardGameGeek and made available by tidytuesday, which can be found here (BoardGameGeek, n.d.a). The data consists of two data sets with each row representing a different board game. One data set contains the user ratings, and the other contains information about the board games, including names and descriptions, as well as several characteristics such as playing time, minimum age, number of players, etc. We first prepossessed the data by joining the two data sets together, dropping unnecessary columns and columns

with too many missing values, and splitting the data into 50% training set and 50% test set. Since the data set is sufficiently large, we also dropped rows with missing values to improve the speed of model training. The pre-processed data can be found here.

Analysis

We used the random forest (RF) algorithm to build a regression model that predicts the average user rating based on various game features. The data used to fit the model was composed of the average user rating as the target, as well as a combination of numeric, categorical and textual variables as the predictors, including description, year published, minimum players, maximum players, playing time, minimum playtime, maximum playtime, minimum age, board game category, board game mechanic, board game family, board game designer, board game artist, and board game publisher. The RF hyperparameters `max_depth`, `max_features`, `bootstrap`, `min_samples_leaf`, and `min_samples_split` were optimized using 5-fold cross-validation in a randomized search with R^2 as the scoring metric.

To perform this analysis and create this report, the Python and R programming languages (R Core Team 2022a, 2022b) were used, along with the following Python and R packages: docopt (Keleshev 2014), numpy (Harris et al. 2020), pandas (McKinney et al. 2010), altair (VanderPlas et al. 2018), scikit-learn (Pedregosa et al. 2011), tidyverse (Wickham et al. 2019), and knitr (Xie 2014). The code used to conduct this analysis and create this report can be found here.

Results & Discussion

To better understand the data, we conducted exploratory data analysis by plotting the distributions of the average rating target, the numeric features, as well as a few categorical features. Figure 1 below shows that the distribution of the average user ratings has a nearly symmetrical bell-shape and a slight left skew, with most user ratings falling between 6-7 on a 10 point scale.

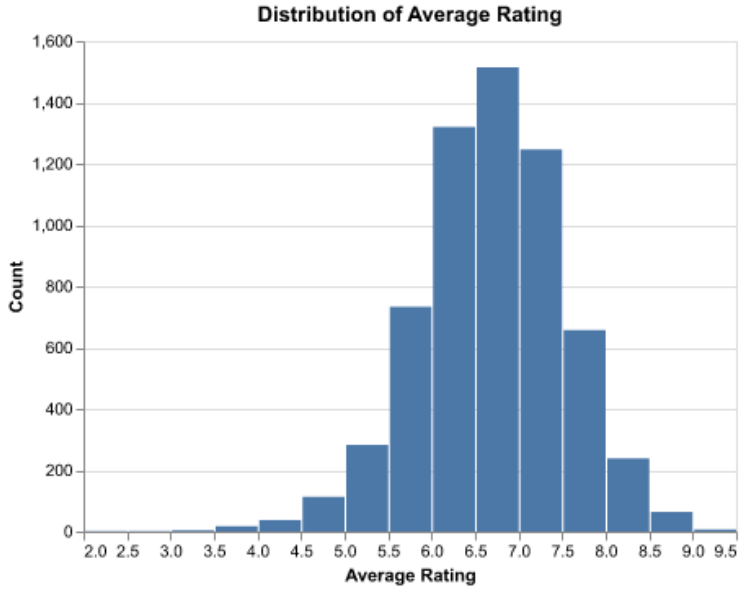


Figure 1: Figure 1. Distribution of average board game user ratings.

With the exception of minimum age, the distributions of the numeric features appear to be heavily right skewed with most values falling towards lower values (Figure 2).

To visualize the distributions of the categorical features, we augmented the training set and added a column that binarizes the average rating target column, with values larger than or equal to 7 being "high" and values

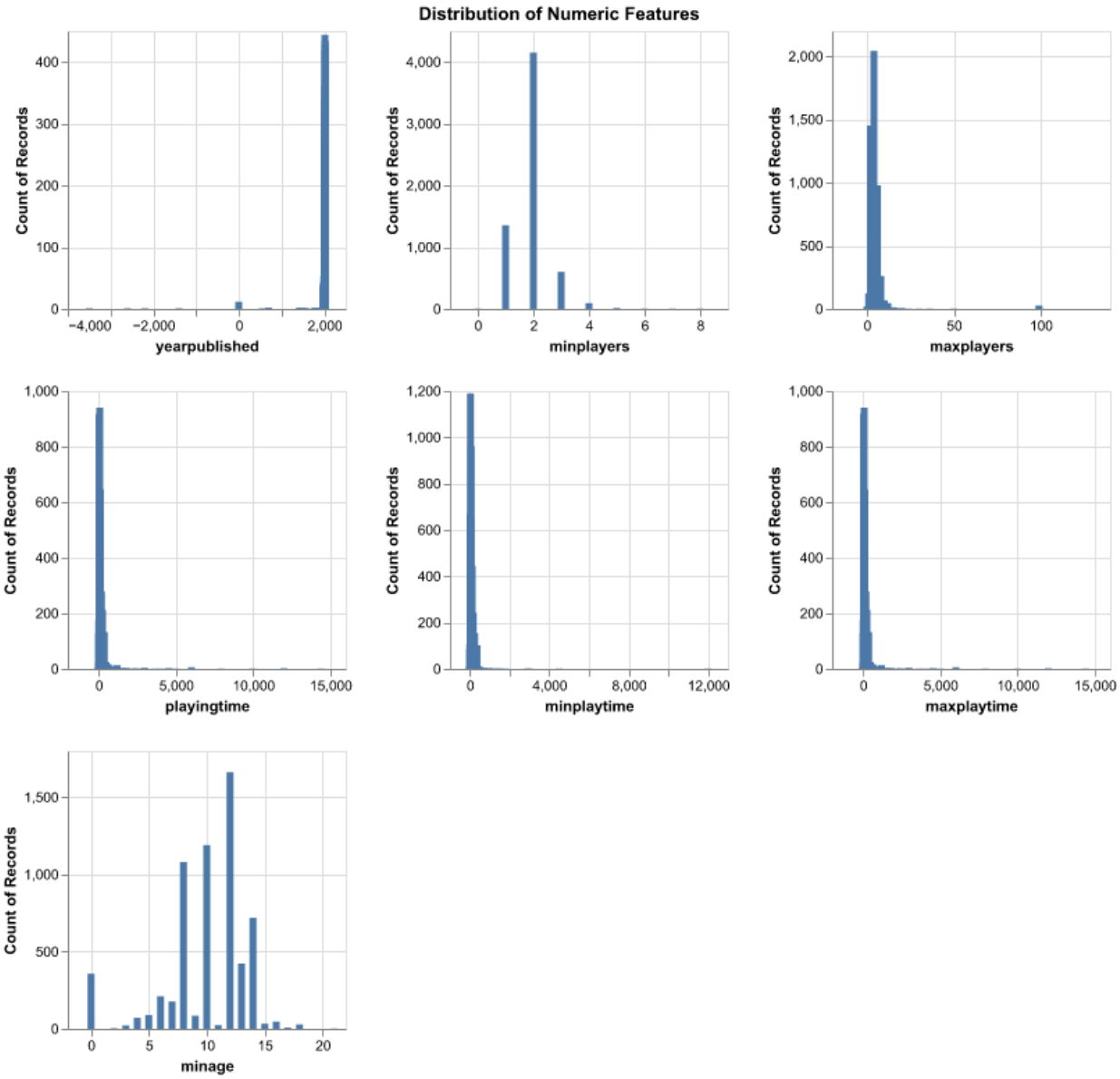


Figure 2: Figure 2. Distributions of numeric predictors.

less than 7 being "low". This allows us to observe which categories are found in board games that tend to be rated "high" versus "low", as shown in Figures 3 and 4 below. For example, we can see that many more card games are rated low than high, while war games have equal counts of high and low ratings.

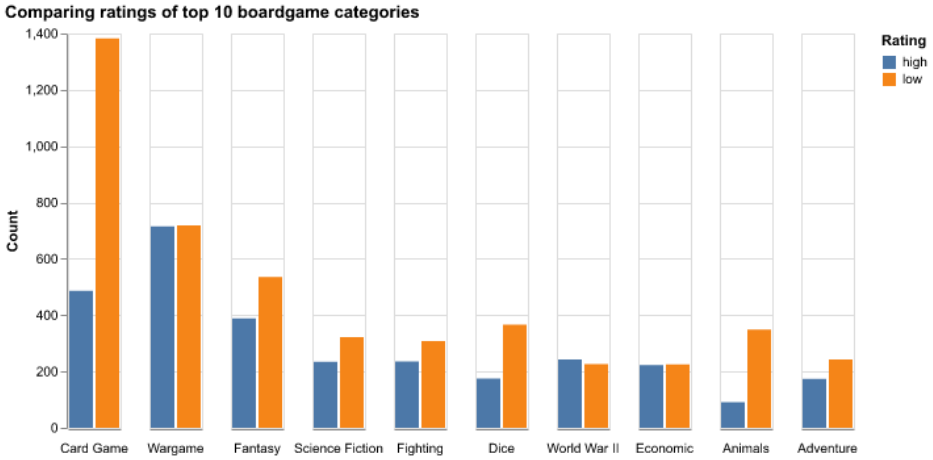


Figure 3: Figure 3. Number of board games rated high (7-10) versus low (1-6) for the 10 most common board game categories.

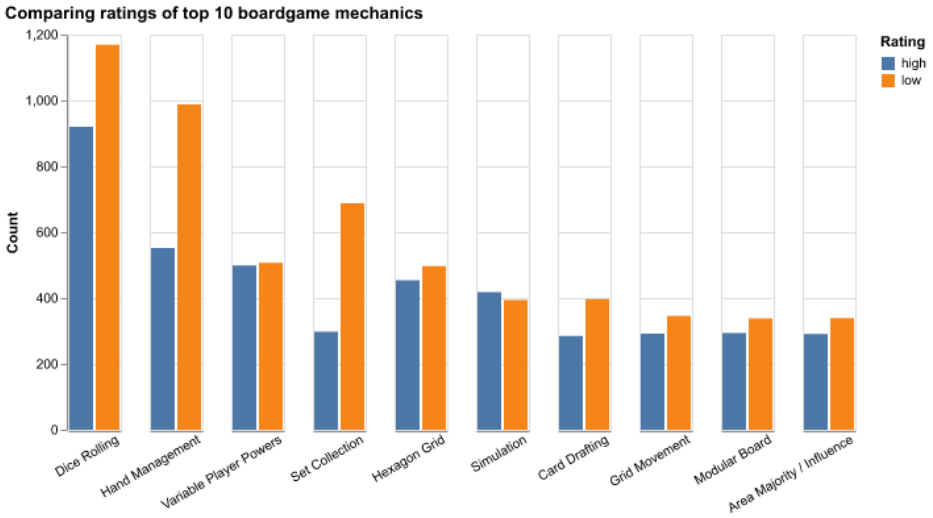


Figure 4: Figure 4. Number of board games rated high (7-10) versus low (1-6) for the 10 most common board game mechanics.

As we have a combination of categorical, numerical, and textual features, we encoded the data using a column transformer to apply the following transformations: `StandardScaler()` on the numeric features, `CountVectorizer()` on the textual features, and `MultiLabelBinarizer()` on the categorical features. `MultiLabelBinarizer()` was used rather than `OneHotEncoder()` since the categorical features contain lists of multiple values per observation.

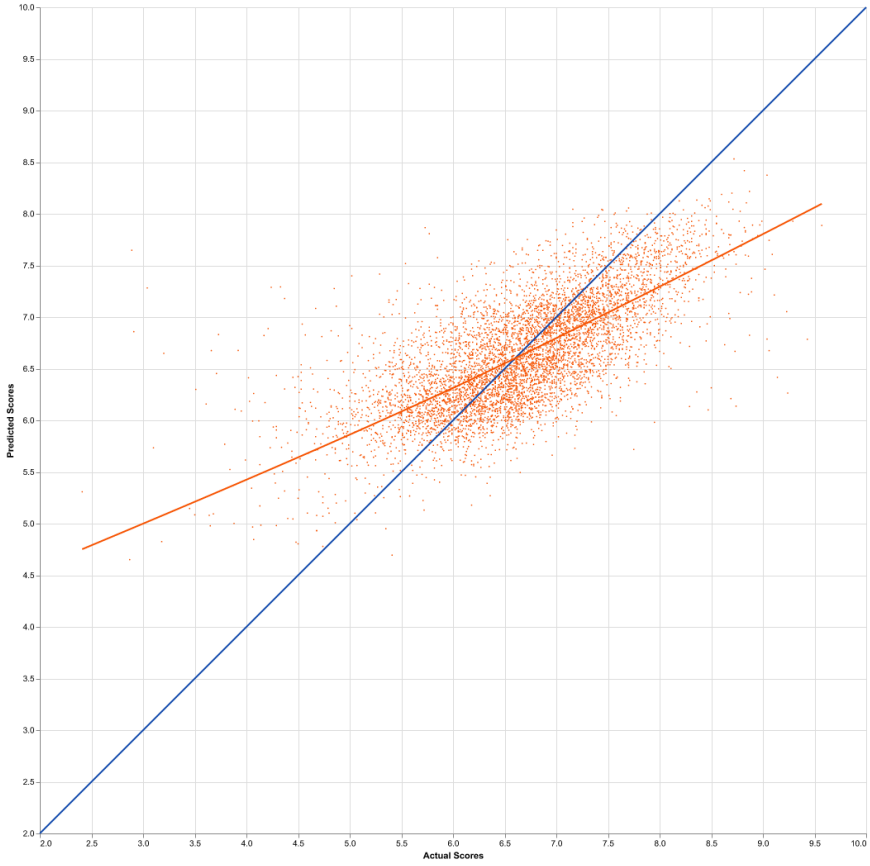
Once the data was transformed, we chose to use a dummy regressor as the baseline model and tested the performance of two regression models, the ridge linear regression model and the random forest (RF) model. For each model, we used 5-fold cross-validation in a randomized search to find their respective hyperparameter values that return the best score, using the R^2 score as the performance metric. The mean absolute percent error (MAPE) scores are also shown to display how far off predicted user ratings were from their actual

Table 1: Table 1. Model performance results using cross-validation on training data.

Metric	Dummy_Regressor	Ridge	Random_Forest
fit_time	0.000	5.452	11.754
score_time	0.000	0.327	0.097
test_r2	-0.001	0.375	0.410
train_r2	0.000	0.606	0.777
test_MAPE	-0.102	-0.078	-0.076
train_MAPE	-0.102	-0.062	-0.044

values. Table 1 below shows that both models perform better than the dummy baseline, although they have fairly low cross-validation scores with less than 50% accuracy. The RF model returns a better cross-validation R^2 score (0.410) than the ridge model (0.375), although it has a much slower fit time. The MAPE scores are also slightly better for the RF model (7.6%) than for the Ridge model (7.8%). Additionally, both models appear to be over-fitting the data, with a large gap between the train score and cross-validation score, despite hyperparameter optimization. Nonetheless, since the RF model produced better cross-validation score, we refit the data using the RF model as the final board game rating predictor.

Our final model performed moderately on the test data, with a R^2 score of 0.480 and a MAPE score of 7.3%. As such, the model performed slightly better on the test set than on the training set. Considering the large size of the data set and 50/50 train/test split, this test score can be considered a reliable indicator of our model's performance on unseen data. Figure 5 below shows how the model performed on the test data as a scatter plot of the predicted versus actual user ratings. The overall accuracy of the predictions are portrayed by the orange fit line, which appears to be askew from perfect accuracy indicated by the blue line. Additionally, there appears to be a lot of variation in the predictions, as the points appear to be spread out away from the orange fit line. Nonetheless, the model appears to be more accurate at predicting ratings around 6.5, as shown by the intersection of the blue and orange lines. This aligns with the distribution of user ratings shown in Figure 1, suggesting that our model is better at predicting commonly occurring median ratings than rarely occurring high or low ratings.



\begin{figure} \caption{Figure 5. Scatter plot of actual vs. predicted user ratings of board games in the test set using the final model. Blue line indicates where predicted ratings equal the actual ratings with 100% accuracy. Orange line is the linear fit between the actual and predicted ratings.} \end{figure}

Overall, our board game user rating predictor needs to be fine-tuned before it can be used to aid board game developers with their creative decisions. Since the test score was not very high, it would be useful to output the probability estimates of the predicted ratings so users can know how confident the model is about each prediction. The model could potentially be improved using feature selection methods to limit the number and choice of features to only those that are most important in predicting the user rating, thus eliminating "noisy" features that drown out the important patterns in the data when training the model. Additionally, the random forest model was slow and computationally intensive, so perhaps the Ridge linear regression model could be improved using polynomial feature extraction and feature selection to produce a more efficient solution.

References

BoardGameGeek, LLC. n.d.a. "BoardGameGeek User Ratings Dataset." *GitHub*. https://github.com/rfordatascience/tidytuesday/tree/master/data/2022/2022-01-25.

———. n.d.b. "On Average, How Long Do You Take to Develop a Game?: BGG." *BoardGameGeek*. https://boardgamegeek.com/thread/557292/average-how-long-do-you-take-develop-game.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. "Array Programming with NumPy." *Nature* 585 (7825): 357–62. https://doi.org/10.1038/s41586-020-2649-2.

Keleshev, Vladimir. 2014. *Docopt: Command-Line Interface Description Language*. https://github.com/docopt/docopt.

McKinney, Wes et al. 2010. "Data Structures for Statistical Computing in Python." In *Proceedings of the 9th Python in Science Conference*, 445:51–56. Austin, TX.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12 (Oct): 2825–30.

R Core Team. 2022b. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

———. 2022a. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

VanderPlas, Jacob, Brian Granger, Jeffrey Heer, Dominik Moritz, Kanit Wongsuphasawat, Arvind Satyanarayan, Eitan Lees, Ilia Timofeev, Ben Welsh, and Scott Sievert. 2018. "Altair: Interactive Statistical Visualizations for Python." *Journal of Open Source Software* 3 (32): 1057.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Xie, Yihui. 2014. "Knitr: A Comprehensive Tool for Reproducible Research in R." In *Implementing Reproducible Computational Research*, edited by Victoria Stodden, Friedrich Leisch, and Roger D. Peng. Chapman; Hall/CRC. http://www.crcpress.com/product/isbn/9781466561595.