# Predicting Steam Games Rating with Regression

*Andreas S.* Teja[1*], *Muhammad Lukman I.* Hanafi[1], and *Nunung Nurul* Qomariyah[1]

[1]Computer Science Department, Faculty of Computing and Media, 11480 Bina Nusantara University, Indonesia

**Abstract.** This paper tries to find out the best regression model to predict the rating of video games. It is done by comparing multiple variables related to Metascore, such as genres and player count. In order to be able to get accurate results, we gather some data by scraping them from Steam and combining them with public data. The games in this study are from Steam since it is one of the largest computer video games distributors. In this study, we evaluate several regression models, such as Linear regression, Decision Tree, Random Forest to predict the game rating. The experiment shows that tree-based regression model, such as LightGBM and Random Forest performed better than any other regression method, with $R^2$ score above 0.9.

## 1 Introduction

For decades, video games have been growing exponentially from when they are started. From the quality aspect to how people interact with them, it is far different from the early days. It is easy to say that with each passing day the number of video games out there is increasing. With that in mind, consumers need information about video games to help them pick which game to play. In early 2001 a website was created to help with that problem, it is called Metacritic. It is a website with the goal of providing people with pieces of information and reviews of a video game [1]. Furthermore, they also provide video games with a possible rating from 0 to 100 called Metascore. How it is scored is by combining several professional reviews and weighting them to produce the final Metascore [2]. Metascore is important for the game world since it is used for benchmarking video games experience [3]. Furthermore, in a study by [4], it is suggested that video games with high scores help with the sales of the game. With that, we want to know with the vast data that is available on the internet could it predict games Metascore.

We believe that someone who is interested in video games, someone who wants to start game development, or someone who is a game developer would be interested in this research. If the research itself shows incredible results it might help game developers by showing what kind of factors could predict a game score.

In this paper, we will try to predict rating by comparing metascore with other variables. It has been proven that metascore have an impact on the success of a game [1-4]. The game that we will be taking is from the top games on Steam, as Steam is the most well-known and still growing video game distributor [5]. To test it, we used many regressions algorithm such as Linear Regression, Decision Tree, Random Forest, and so on. The detail algorithm that used in this study will be explain in Section 3.

## 2 Literature review

In this modern age, almost every product's review can be seen at glance in form of ratings and with first impressions being absolutely crucial to make or break an opportunity, various companies and data scientists alike have looked into how to predict what rating their associated product will receive.

In [6], they used Decision Forest Regression, Decision Tree Regression and Gradient Boosted Tree Regression among other methods to predict the ratings of Amazon products. As they are pulling data straight from amazon reviews that customers have written, they also needed to do text analysis in order to detect polarity. While they included a very large dataset sourced from Amazon itself, the models they used were rather limited and were mostly variations of the same model. This is in contrast to us as we included various different models to find out which may be best at predicting the ratings based on our dataset.

[7] on the other hand, not only used various models like Gradient Boosting Classifier, Extreme Gradient Boosting Classifier or AdaBoost Classifier and text analysis for polarity but they also used ensemble learning to combine the models to further improve results. They also had a large dataset to ensure that their models have ample training data and also ensured the quality of said data by ensuring that the app is at least 5 years old and have 4000 posted reviews. Although our methods are similar to a degree, they use methods that we currently do not use and will most likely use if we decide to expand on this paper. In [8] study they are

* Corresponding author: andreas.teja@binus.ac.id, muhammad.hanafi003@binus.ac.id, nunung.qomariyah@binus.ac.id

predicting the popularity of a movie based on data that are extracted from IMDB. The data they extracted include similar data type that this paper uses such as genre, and Metascore. [8] used a tool called Weka to perform the prediction using machine learning algorithms such as Logistic regression, NN, and decision tree. For some insight, Weka is a tool that has machine learning algorithms and can be used for data mining, data processing, classification, clustering, visualization, regression, and more [8]. Another study by [9] predicted the success of a movie by using IMDB data. The study also uses some of the algorithms that were used in previous study and show similar results.

A recent study in [10] shows that regression model can be used to predict movie ratings from IMDb dataset. Similar with our study, they also show that study in recommender system field is not only about using classification algorithm like collaborative filtering or clustering algorithm like content based filtering, but also by using regression method to predict number, in this case, the user rating.

A study directly related to our research was conducted by [11], where they also attempted to predict the rating of games albeit they focused on the board genre. That being said, they used a different methods and models namely Multinomial Naïve Bayes, SVC Linear and Ensemble model. They also made variants with unbalanced and balanced data.

Next, in the study by [12], they attempted to predict the rating of games on the App Store. They also used multiple models namely Random Forest, Decision Tree, Logic Regression and K-NN. However, they perhaps oversimplified their rating system as they categorized 0-3.5 as low rating and 4-5 as high rating though this is most likely due to them wanting to calculate Accuracy, Recall and Precision as well.

Lastly, in the study by [13], they attempted to improve upon a proposed method by Pacula to predict the rating of a game through hours played. The models they used are various KNN algorithms, SDV++ and Slope One.

## 3 Datasets

The datasets that are used for this research comes from multiple sources; the plan is to merge the datasets to better analyze the games. Some of the variables that are going to be taken account are the metascore, user reviews, average player, peak player count and many more.

### 3.1 Top video games 1995 - 2021 metacritic

Table 1 shows the snippet of the dataset [14]. The dataset include game name, platform, release date, summary, metascore and user score.

**Table 1.** Top video games.

| Name | Platform | release_date | Summary | meta_score | user_review |
|---|---|---|---|---|---|
| The Legend of Zelda: Ocarina of Time | Nintendo 64 | November 23, 1998 | As a young boy, Link is tricked by Ganondorf... | 99 | 9.1 |
| Tony Hawk's Pro Skater 2 | PlayStation | September 20, 2000 | As most major publishers' development efforts... | 98 | 7.4 |
| Grand Theft Auto IV | PlayStation 3 | April 29, 2008 | [Metacritic's 2008 PS3 Game of the Year; Also known as "GTA IV"] ... | 98 | 7.7 |
| SoulCalibur | Dreamcast | September 8, 1999 | This is a tale of souls and swords... | 98 | 8.4 |

### 3.2 Steam game data

The table 2 show the snippet of dataset, the dataset includes many data but the one that we are using are the response name, release date, and all of the genres [15].

**Table 2.** Steam game data.

| ... | responsename | release date | ... | genreisearlyaccess | genreisfreeoplay | genreissports | genreisracing | genreismassivelymultiplayer | ... |
|---|---|---|---|---|---|---|---|---|---|
| ... | Counter-Strike | Nov 1 2000 | ... | false | false | false | false | false | ... |
| ... | Team Fortress Classic | Apr 1 1999 | ... | false | false | false | false | false | ... |
| ... | Day of Defeat | May 1 2003 | ... | false | false | false | false | false | ... |
| ... | Deathmatch Classic | Jun 1 2001 | ... | false | false | false | false | false | ... |
| ... | Half-Life: Opposing Force | Nov 1 1999 | ... | false | false | false | false | false | ... |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

## 4 Data preparation and processing

As previously mentioned before, our dataset will come from multiple sources, so what we need to do first is to fully understand what data we want to use. Since it is going to be more focused on the games that are on Steam, we will first need to scrape more data from steam charts, because the data that is provided on Kaggle is only top 100 games. We also need to find a good range of time for testing. Furthermore, as mentioned before, due to us having multiple sources, we will also need to merge the datasets, perhaps modify some that we need which represent the same data but are named differently, and perhaps drop irrelevant data on each dataset.

### 4.1 Scraping steam chart

These are the libraries used to scrape Steam chart:

- BeautifulSoup4
- Pandas
- Requests
- Re

Since [16] has multiple layouts, the process is split into two parts, which are getting the game code and then

using the game code to get the other information about the game, such as: 730, 570, 578080, 1172470, 271590, 1446780, etc.

**Table 3.** Result steam chart scrape.

| Game Name | Month | Average Players | Gain % | Peak Players | Gain |
|---|---|---|---|---|---|
| Counter-Strike: Global Offensive | December 2021 | 546614.19 | -0.28% | 950586 | -1547.48 |
| Counter-Strike: Global Offensive | November 2021 | 548161.67 | +6.97% | 935593 | 35725.82 |
| Counter-Strike: Global Offensive | October 2021 | 512435.85 | +0.02% | 864966 | 84.93 |
| Counter-Strike: Global Offensive | September 2021 | 512350.92 | +0.05% | 942519 | 268.96 |
| Counter-Strike: Global Offensive | August 2021 | 512081.96 | +1.19% | 802544 | 6014.60 |
| ... | ... | ... | ... | ... | ... |

The first part is getting the game code, how the scraper works is by going to each page, changing its path to its distinct pathing. Each page have a label of 'p.' and then a number, where if the number is changed it will go to a certain page. Then the scraper use the keyword '/app/' for splitting the found word, the result will return the game code number.

The second part is to get the data by using the game code, first is to initialize all the data that is needed which are the name of the game, month of the data, average players, gain or loss of that month, the percentage of the gain, and the peak players of that month. Second is using the length of the game code list and the code itself to go to each page to get the data that has been mentioned before. Lastly, is to put the data into a data frame, each loop it appends the data to a premade data frame so it would not overwrite the data that is gotten and then clear the list so it does not add the same data. the result is as shown in table 3.

## 5 Model and techniques

The model we have chosen is Regression Modeling and Neural Network. We chose to abstain from Correlation Modeling because Correlation what we were trying to do is not find out if there is a relationship and instead to predict. As for Neural Network, some of our data may have non-linear dependencies so we include it just in case.

In terms of techniques, we use multiples, compare the Root-Mean-Square-Error (RMSE) and choose the one with the least RMSE. The techniques we use include but may not be limited to Linear Regression, Ridge, Lasso, KNeighborsRegressor, MLP Regressor (Neural Network), Decision Tree Regressor, Random Forest Regressor, Gradient Boosting Regressor, XGB Regressor, LGBM Regressor, and Cat Boost Regressor.

### 5.1 Preprocessing

Libraries that are required is shown below:

- LabelEncoder
- Train_test_split
- StandardScaler

First, we pick the columns we do not need or columns that may give the models an unfair or unrealistic advantage and drop them.

As we want to predict what Metascore a game might get, we remove game name since we are not using any text classifier to identify the score, platform because all of them are games that are on PC, and user review as it is similar to the Metascore, it is similar which gives the decision tree model an unfair advantage. We also remove release_date since it is a duplicate.

Next, we encode some of the data as some of them are still in the form of string like Month, Gain %, Gain and ReleaseDate. Finally, we split the data into X and y, conduct a train-test split with a train size of 70% and a random state of 1 and standardize the data by scaling train and test set data.

### 5.2 Training

Libraries that are required in the implementation are shown below:

**Table 4.** Parameters.

| Model | Parameters |
|---|---|
| LinearRegression | fit_intercept = true, copy_x = true |
| Ridge | fit_intercept = true, copy_x = true |
| Lasso | fit_intercept = true, copy_x = true |
| KNeighborsRegressor | n_neighbor = 5, weight = uniform |
| MLPRegressor | learning_rate = constant, shuffle = true |
| DecisionTreeRegressor | splitter = best, max_depth = none, min_sample_split = 2 |
| RandomForestRegressor | max_depth = none, min_sample_split = 2 |
| GradientBoostingRegressor | max_depth = 3, min_sample_split = 2 |
| XGBRegressor | max_depth = 6, eta = 0.3 |
| LGBMRegressor | max_depth = -1, learning_rate = 0.1 |

For efficiency purposes, we create a dictionary with the name of the model and the function to use said model. After that we loop through the dictionary for every model that is in said dictionary to train it. The parameters for all of the models are defaults value from the library.

## 6 Evaluation method

As mentioned before, we will be using RMSE to measure, on average, how far apart the predicted values are from the test data. In addition to that, we will also be using $R^2$ to measure the proportion of the variance in the response variable of a regression model that can be explained by the predictor variables. As RMSE measures the variance between the prediction and the real values so naturally, we were looking for the technique that has the least RMSE. For $R^2$, it tells us how well the predictor variables can explain the variation in the response variable, the closer to the value 1 it is, the better.

### 6.1 Root mean squared error (RMSE)

$$\oint RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \quad (1)$$

Predicted value
Actual value
Summation, where N is total sample size
The differences between predicted and actual value

To get the RMSE (as shown in Equation 1) of each model, we again use the same dictionary to loop through each model, keep the predicted value as the result when it tries to predict the metascore of the data and finally use the formula shown above1 to get the RMSE by comparing the actual value with the predicted value from the model.

**Table 5.** RMSE result.

| Model | RMSE |
|---|---|
| Linear Regression | 6.8347 |
| Linear Regression (L2 Regularization) | 6.8345 |
| Linear Regression (L1 Regularization) | 7.2472 |
| K-Nearest Neighbors | 3.5803 |
| Neural Network | 3.9947 |
| Decision Tree | 0.9249 |
| Random Forest | 0.5991 |
| Gradient Boosting | 2.4979 |
| XGBoost | 2.6462 |
| LightGBM | 0.5227 |

### 6.2 Normalised root mean squared error (NRMSE)

$$\oint NRMSE = \frac{RMSE}{ymax - ymin} \quad (2)$$

where:
ymax = Max rating of test dataset
ymin = Min rating of test dataset

To get the NRMSE of each model, we use the results from above, get ymax and ymin respectively and use the formula shown above (Equation 2) to get the NRMSE by dividing the RMSE with ymax and ymin.

**Table 6.** RMSE result.

| Model | NRMSE |
|---|---|
| Linear Regression | 0.1953 |
| Linear Regression (L2 Regularization) | 0.1953 |
| Linear Regression (L1 Regularization) | 0.2071 |
| K-Nearest Neighbors | 0.1023 |
| Neural Network | 0.1141 |
| Decision Tree | 0.0264 |
| Random Forest | 0.0171 |
| Gradient Boosting | 0.0714 |
| XGBoost | 0.0756 |
| LightGBM | 0.0149 |

### 6.3 Coefficient of determination $R^2$

$$\oint R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \quad (3)$$

where:

$\sum(\hat{y}_i - y_i)^2 =$ Predicted model error
$\sum(y_i - \bar{y})^2 =$ The base model error

To get the $R^2$ of each model, we use the same concept as the before but change the formula to the formula to get the $R^2$.

**Table 7.** $R^2$ result.

| Model | $R^2$ |
|---|---|
| Linear Regression | 0.22803 |
| Linear Regression (L2 Regularization) | 0.22808 |
| Linear Regression (L1 Regularization) | 0.13202 |
| K-Nearest Neighbors | 0.78816 |
| Neural Network | 0.73628 |
| Decision Tree | 0.98586 |
| Random Forest | 0.99407 |
| Gradient Boosting | 0.89689 |
| XGBoost | 0.88428 |
| LightGBM | 0.99548 |

## 7 Result and discussion

Although all of our models show great results, a few are better than others namely LGBM, Random Forest, and Decision Tree. Here we are using prediction error and residuals plot to visualize our results.

Figure 1 shows the result of LGM model. The $R^2$ score shows very high result, which is 0.995. In Figure 2, the model performance shows the $R^2$ of 0.994, which is also quite high. In Figure 3, the Decision Tree model shows the $R^2$ as high as 0.986. From the result, we can see that the model can yield a very satisfied performance when predicting the rating of the game. In this section, we only highlight the models which returned the highest $R^2$ score.

After gathering data from existing databases and scraping data ourselves, preparing said data and then training various models, we have evaluated that among the 10 models, two of them stand out namely LightGBM and Random Forest. LightGBM had the best results by a slight margin with a standard deviation of the residuals (RMSE) of 0.5227, an NRMSE of 0.0149 and a coefficient of determination ($R^2$) of 0.99548 with Random Forest following closely behind with an RMSE of 0.5991, an NRMSE of 0.0171 and an $R^2$ of 0.99407. That being said, considering that the rating scale of Metascore is 0-100, the accuracy of these models at predicting what rating a game would have is very good.

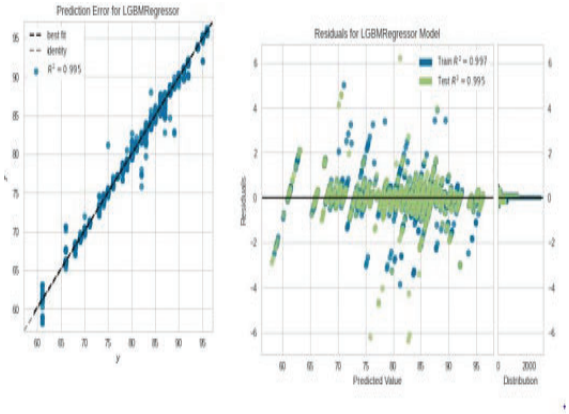
In terms of sheer accuracy, LightGBM is better than any other model we have tested as even compared to the Random Forest model, it is 14.62% more accurate but in a dataset where the data is represented in no decimal place and goes up to 100, a 14.62% difference would

mean a difference of 0.0764 which is so small that it would not even be represented in the data in any way even if rounded up. In conclusion, again, although LightGBM is technically better than Random Forest, the difference is so negligible that either would work.

Now, with that being said, how does our method compare with other related works? In section 2 we mentioned the [11] study and in said study, their rating scale was 0-10 so we had to calculate their NRMSE, their best model was the Ensemble Model with Balanced and Unbalanced SVC. However, even with their best model, they only achieved an NRMSE of 0.2496 which does make our models better.

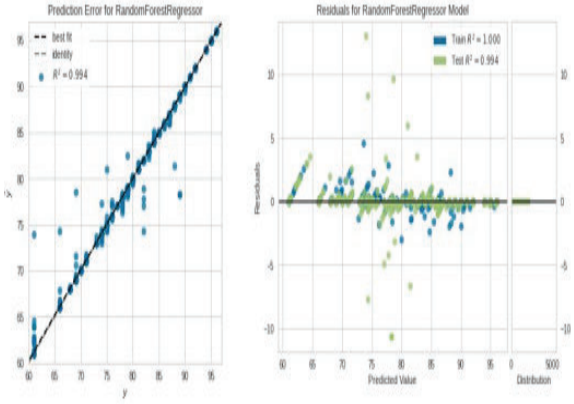
Another example is the [12] study and as their rating scale is only 0-5 and as they do not provide an NRMSE, we will again calculate it for them. With their best model, the Random Forest, they have an NRMSE of 0.0881 and which is very good but fortunately our best models are still slightly ahead.

Lastly, we have the [13] study. Here, fortunately they provided an NRMSE making the process much easier. Their best model, the SDV++, had an NRMSE of 0.2543. Again, fortunately this does mean that our models are still better.



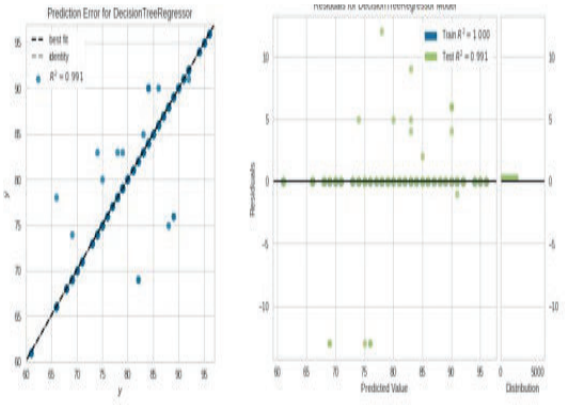

(a) Prediction Error: LGM (b) Residuals plot: LGBM

**Fig. 1.** LGBM result.




(a) Prediction Error: Random Forest
(b) Residuals plot: Random Forest

**Fig. 2.** Random forest result.




(a) Prediction Error: Decision Tree
(b) Residuals plot: Decision Tree

**Fig. 3.** Decision tree result.

## 8 Conclusion and future work

In this paper, we showed the implementation of machine learning analysis in game industry. This study explored several regression models to predict game ratings. Because we were trying to predict numerical label, the regression algorithms were used to solve this problem. Based on our findings, the tree-based regression algorithm shows the better performance when compared to the other regression methods. We also show that our model has outperformed other previous published research by other author.

To improve this research, we would like to recommend gathering more information related to the games, such as total playtime, difficulty, and art style. Furthermore, having text classification could also improve the research as it can be used to encode variable such as game name, summary and user review.

## 9 Supplementary files

The code can be accessed through Github link: https://github.com/Lynceusthepotato/Game_Prediction_DS.

## References

1. A. Greenwood-Ericksen, S. R. Poorman, P. Roy, *On the validity of metacritic in assessing game value,* Eludamos: J. Computer Game Culture **7**, 1, pp. 101–127 (2013)
2. D. Johnson, C. Watling, J. Gardner, L. E. Nacke, *The edge of glory: the relationship between metacritic scores and player experience*, in Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play, pp. 141–150 (2014)
3. M. S. Lee, C. Heeter, *What do you mean by believable characters? the effect of character rating and hostility on the perception of character believability*, J. Gaming & Virtual Worlds **4**, 1, pp. 81–97 (2012)

4. B. Bower, Valuing a video game: does score determine value? PhD thesis (2009)
5. J. Gong, Y. Ye, K. Stefanidis, *A hybrid recommender system for steam games*, in International Workshop on Information Search, Integration, and Personalization, pp. 133–144 (Springer, 2019)
6. J. Woo, M. Mishra, *Predicting the ratings of amazon products using big data*, WIREs Data Mining and Knowledge Discovery **11** (2021)
7. M. Umer, I. Ashraf, A. Mehmood, S. Ullah, G. S. Choi, *Predicting numeric ratings for google apps using text features and ensemble learning*, ETRI J. **43**, pp. 95-108 (2020)
8. M. H. Latif, H. Afzal, *Prediction of movies popularity using machine learning techniques*, International J. Computer Science and Network Security (IJCSNS) **16**, 8, pp. 127 (2016)
9. M. Jaiswal, A. Srivastava, A. Prasad, T. J. Siddiqui, *Prediction and analysis of movie success with machine learning approach* (2020)
10. N. N. Qomariyah, D. Kazakov, A. N. Fajar, *User preference modelling from movie database*, in 2020 International Conference on ICT for Smart Society (ICISS), pp. 1–5 (IEEE, 2020)
11. M. Miah, *Board game rating prediction based on reviews* https://www.kaggle.com/code/mintumiah/board-game-rating-prediction-based-on-reviews/notebook (2020)
12. G. A. Sandag, *Application rating prediction on app store using random forest algorithm* https://www.researchgate.net/publication/347712060_Prediksi_Rating_Aplikasi_App_Store_Menggunakan_Algoritma_Random_Forest (2020)
13. J. Pérez-Marcos, D. Sánchez-Moreno, V. F. L. Batista, M. D. Munoz, *Estimated rating based on hours played for video game recommendation*, in Distributed Computing and Artificial Intelligence, Special Sessions, 15th International Conference, pp. 300-307 (2019)
14. Kaggle, *Top video games 1995-2021 metacritic* https://www.kaggle.com/deepcontractor/top-video-games-19952021-metacritic (2022)
15. C. Kelly, *Steam game data* https://data.world/craigkelly/steam-game-data (2016)
16. Steamcharts, *An ongoing analysis of steam's concurrent players* https://steamcharts.com/ (n.d.)