Steam store game recommendation system using machine learning

Zayed Humayun
Brac University
Email: zayed.humayun@g.bracu.ac.bd

***Abstract*—Steam store is one of the biggest retailers of online video games in PC. Each game in the steam store has huge amounts of data that can find similar games and recommend them to the players, making it easier for the players to purchase games of their liking. To maximize the purchase of the games in the store, the recommendation system needs to accurately predict the game of the player's liking. The models used in this paper can help us differentiate which model is better for recommending a game to the player.**

1. Introduction

Recommendation systems are widely used in many e-commerce platforms, where they aid in recommending similar products to the user. Steam released its 2021 annual report on 09 March 2022, the report consists of steam platform growth, data on the store, steam works and steam deck. The report gives insight on the active players on the platform, there are up to 69 million active users every day, 132 million active users each month and 27.4 million active users at the same time. The data also provides information on how well steam store performed better than 2020, users spent 27% more on purchasing games on steam in 2021 [1]. One of the main factors behind the increase in purchasing games is because of the ways steam enables users to find games from the search catalogue. Without a recommendation system, the users can be overwhelmed by the number of games which are available, and this can lead to an increase in the churn rate—as players start to find it hard to pick a game or users dropping the game midway and asking for a refund. Recommendation systems can help prevent this problem by ensuring the users are engaged in the platform with the correct recommendation. The recommendation system is driven by machine learning, the machine learning algorithm learns the user's preference and recommends games to match the player's personal preference.

To achieve these goals, the recommendation system needs to be robust, so that the system can recommend correct games that fit the player's preference based on what game the user is playing, what category of games the user usually plays, what genres the user prefers, and the rating is also taken into account.

The purpose of this paper is to utilize a dataset scraped from the steam store to generate recommendations. The recommendation of the game will be based on the rating of the game. The aim of this paper is to attempt to predict the values in the test dataset using the training dataset.

2. Methodology

In this section, I describe the procedure on how we used the dataset, Steam store games by Nik Davis [2]. I also explain on how I validated and analysed the data.

2.1. Dataset Description

The dataset consists of 27075 rows and 9 columns. It contains information on the names of the games, along with the genres and category the game falls into. The data also provides information on the developer and publisher of the game. Each game in the steam store has an achievements, ratings and the platforms the game can be played in associated with the game. Furthermore, the dataset also gives us an insight in the average and median playtime of the game.

2.2. Pre-Processing Techniques applied

Owner column has continuous data, so I made the data discrete by taking the median.

We have a lot of unique values for genres, so we need to do some feature engineering to deal with it.

Each game in the dataset has multiple genres associated with them. In total, there are 1552 genres combination. I split the genres of the whole dataset and got 29 unique genres. Then I added 29 new columns of genre in the dataset. The game associated with the particular genre has a value of 1 in the row, and the rest genre for the game is 0.
Similarly, each game in the dataset has multiple platforms. In total, there are 7 platform combination. The same feature engineering is done with platforms, I add 3 columns to the dataset and the game consisting of the platforms has a value of 1.

The developers, publishers column in the dataset has string values, so each developer is assigned a unique integer value by encoding.
Categories and tags have multiple names associated with it, so they are also encoded to numbers, I did not break them down to columns as there will many columns in the dataset in doing so.

There are two rating columns, which includes the positive and negative ratings. I combined the two ratings into one column by adding both the columns and taking percentage of the positive approval rating of the game.

Column name	Size
genres	1523
platforms	7
categories	3333
steamspy_tags	6423

TABLE 1. DATASET INFORMATION

Name	Range	Category
Horrible	ratings $<$ 40 %	0
Bad	40 % $<=$ ratings $<$ 55 %	1
Median	55 % $<=$ ratings $<$ 70 %	2
Good	70 % $<=$ ratings $<$ 90 %	3
Great	ratings $>=$ 90 %	4

TABLE 2. RATING CATEGORY

I categorized the rating into five categories based on the percentage of their positive approval rating. Therefore, rating now has 5 values based on their percentage range.

A new column total rating was also added, combing the positive and negative columns.

Finally, I modified the release date column to produce the age of the game by subtracting the current day with the date it was originally released.

Columns with negative correlation to the ratings were additionally dropped.

Attributes	Correlation
english	-0.027222
Action	-0.059098
Free to Play	-0.112447
Strategy	0.063746
RPG	-0.010470
Simulation	-0.088551
Racing	-0.039832
Violent	-0.037010
Massively Multiplayer	-0.176117
Sports	-0.038141
Early Access	-0.052152
Gore	-0.040377
Utilities	-0.008005
Education	-0.001650
Software Training	-0.005869
Audio Production	-0.001167
Game Development	-0.007131
Photo Editing	-0.010641
Accounting	NaN
Documentary	NaN
Tutorial	NaN

TABLE 3. NEGATIVE CORRELATION ATTRIBUTES TO RATING

2.3. Models applied

As the data is preprocessed and contains numerical information, I opted for KNN, i.e, K Nearest Neighbour classifier. Decision tree was chosen as a model because it works well with classifying datasets. The third model I chose is Random forest, as Random forest uses bootstrapping to randomly sample the dataset, which diversifies the trees.

The training dataset contains 80 % of the data and the test dataset contains 20 % of the data.

2.3.1. KNN. As I want to predict where or not a game should be recommended based on the rating. KNN computes the differences from the given training data to each of the test data in the dataset. Then KNN choses the k nearest games and run a voting scheme (for example, if the majority of the rating falls into the Great category, then I predict that the game is a Great category game).

2.3.2. Decision Tree. The main goal of the decision tree is to make the best splits between the columns (nodes) which will optimally divide the data into correct categories. The decision tree will be built using the training dataset. The goal of the algorithm is to predict a target variable rating from a set of input variable and their attributes. The algorithm builds a tree structure through a series of splits based on yes/no, from the root node via branches passing several nodes until it hits a leaf node. Once the leaf node is reached, the prediction is made.

2.3.3. Random forest. The main idea of Random forest is to create N subsets of objects, where N is the number of models, from the training set. Then each subset is used to create a decision tree. After creating all the models, the final prediction is obtained by majority voting among the models in the case of a classification task.

3. Results

I used Confusion matrix as it highlights where the models make mistakes.

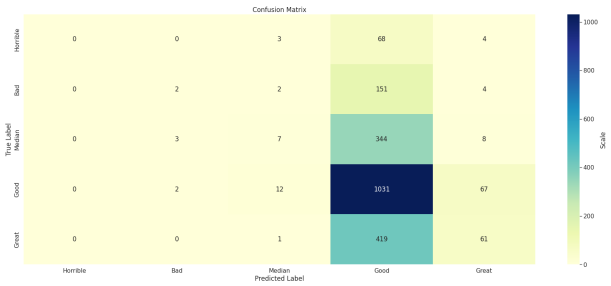

Figure 1. Confusion matrix of KNN

From the KNN confusion matrix we can see that, there are total of 2189 data and out of those only $5+1049+34 = 1088$ were classified correctly.

Accuracy score $= \frac{1088}{2189} \times 100 = 49.7\%$

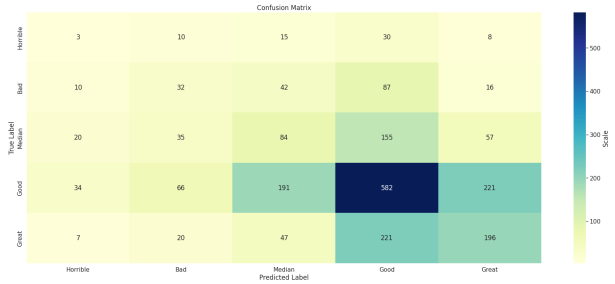

Figure 2. Confusion matrix of Decision Tree

From the Decision tree confusion matrix we can see that, there are total of 2189 data and out of those only $7 + 24 + 80 + 582 + 173 = 866$ were classified correctly.
Accuracy score $= \frac{866}{2189} \times 100 = 39.6\%$ From the Random

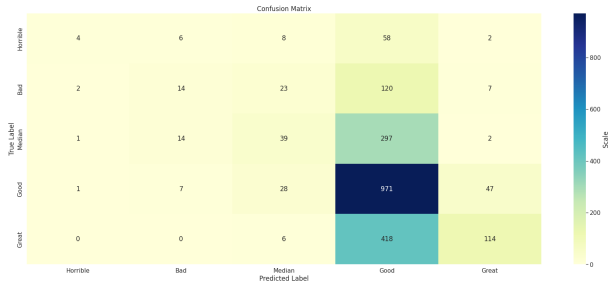

Figure 3. Confusion matrix of Random forest

forest confusion matrix we can see that, there are total of 2189 data and out of those only $1 + 11 + 24 + 992 + 116 = 1144$ were classified correctly.

Accuracy score $= \frac{1144}{2189} \times 100 = 52.3\%$

4. Conclusion

We can conclude that Random forest is giving a better result at predicting the rating of the game. It is evident from the heatmap that the decision tree is often mixing up the rating, e.g. For a good game it is giving a median rating. KNN performs on par with Random forest, but there are still some outliers.

To improve the accuracy I also did cross validation with all the three models but the overfitting issue is not solved as the accuracy does not significantly change.

References

[1] Steam store annual report. (2022, March 9). https://store.steampowered.com/news/group/4145017/view/3133946090937137590 2

[2] Davis, N. (2019, May). Steam store games. Kaggle. https://www.kaggle.com/datasets/nikdavis/steam-store-games