Final Project Report: Game Recommendation System

Instructor:
Gagan Raj Gupta

Team Gameado:
Shreyas K Shastry (11741000)
Anuj Verma (11740170)
Aatmjeet singh(11740010)
Rushabh Dhabe(11740780)
Driti Singh(11840460)

Date:20-11-2020

Introduction:

Video game industry is a multi-billion dollar industry and is expected to grow to 180.1 billion USD by 2021[1]. And we being gamers as many other young people we wanted to make a recommender system that helps players pick gamers from their previous played games.

A **recommender system**, or a **recommendation system** (sometimes replacing 'system' with a synonym such as platform or engine), is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item.They are primarily used in commercial applications.

In the project we mainly used collaborative filtering method.It is based on the principle that if two people liked the same things in the past, if one of them likes something new, the other is likely to like it too. One of the upside of using this method is that we dont require the user information, we only need user-item interaction details which are commonly not so private.

The data for these can be explicit(ex:number based ratings) and implicit(ex:Time spent on reading an article), since we only got implicit data from the steam we used the same for recommending the games to the users.

This project is implemented as our final project for the course Data analysis (DS250).

Dataset:

For this project we collected all our data from Steam[2] using their Web API. Steam is a major online video game distribution platform for PC games. The platform has around 48,000 video games in total and a user base of around 1 Billion users.

We extracted two basic datasets from the Steam API:

1. **Application Dataset:** This dataset contains information regarding each application present on the Steam platform. The dataset is broken down into two files:
    a. app_data: This file contains the general information regarding each application on Steam, such as app_id, title, genre, type etc.

[1] Global Games Market
[2] Steam Store

   b. app_reviews: This file contains the review information for each application, such as the total number of positive/negative reviews etc.

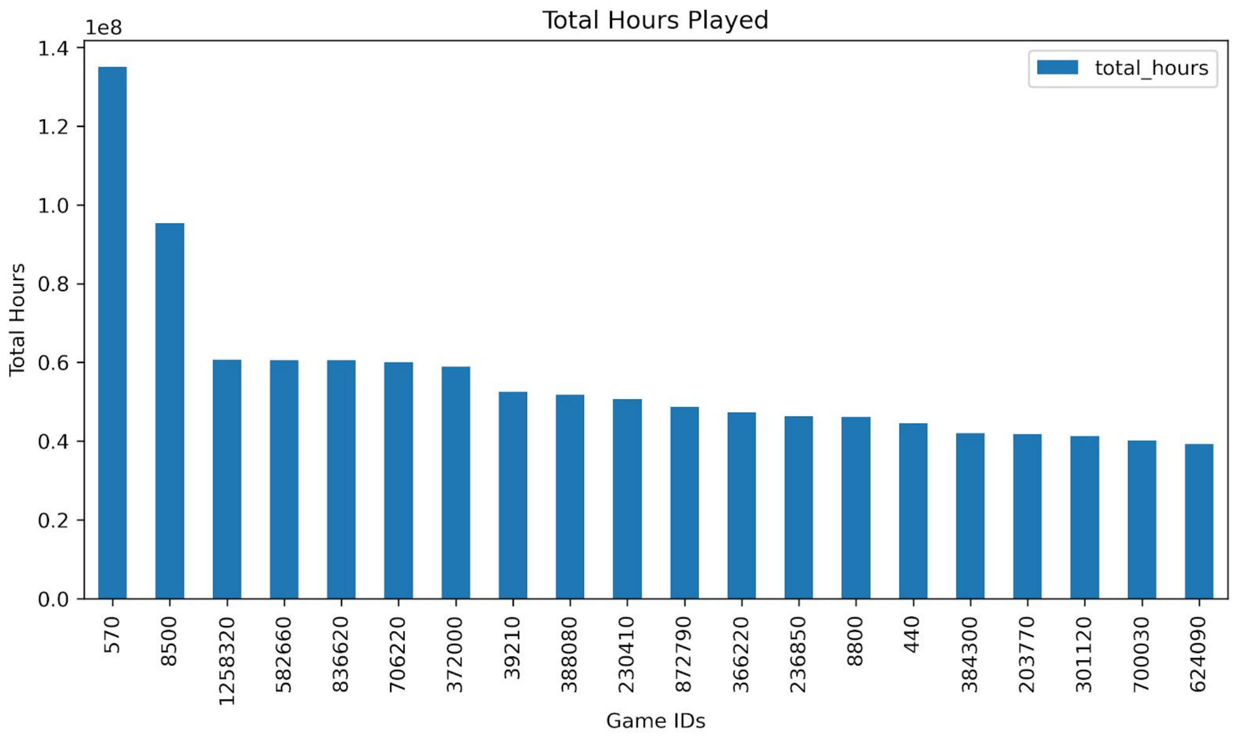
2. **User Dataset:** This dataset contains multiple user reviews for each application. We extracted at most 1000 reviews for each application. In total we got around 4 Million user reviews for all the applications.

Data Cleaning:

- We removed all the applications from both the datasets that are not of the type “game”.
- For the User dataset we removed the user reviews with 0 hours of playtime.
- We also removed the columns which do not contribute to the model.
- The notebook we used to clean the raw datasets: *link*

Data Visualization:

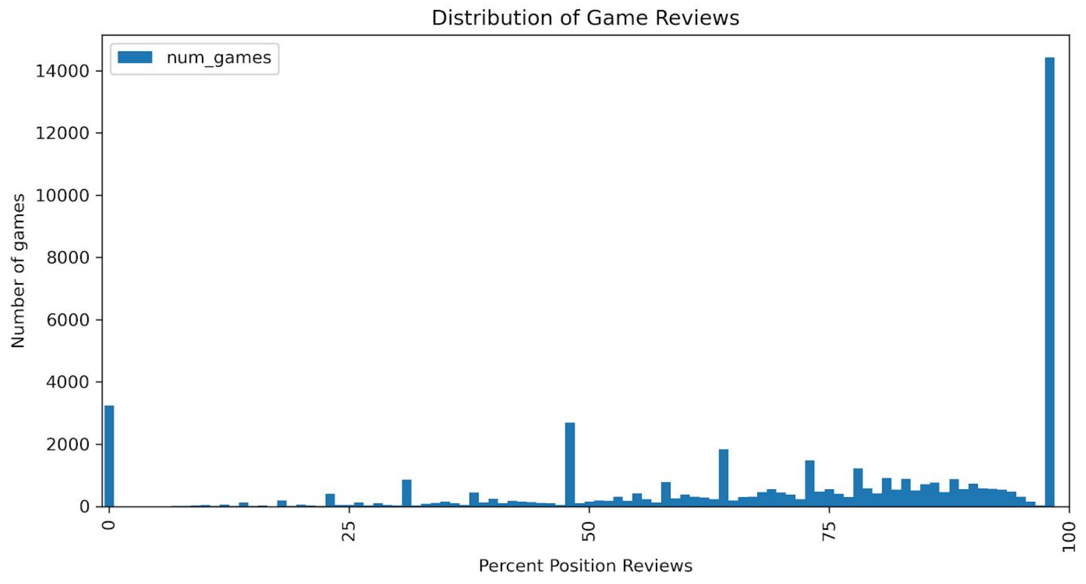
We use a histogram plot in order to better visualize the results displayed in the table above. Game titles are ordered in decreasing order based on the number of player hours. The following graph represents the total of hours played in any game (top 20) from most played to least played. It is plotted against their app_id. app_id 570 is DOTA-2 and is the most played globally.



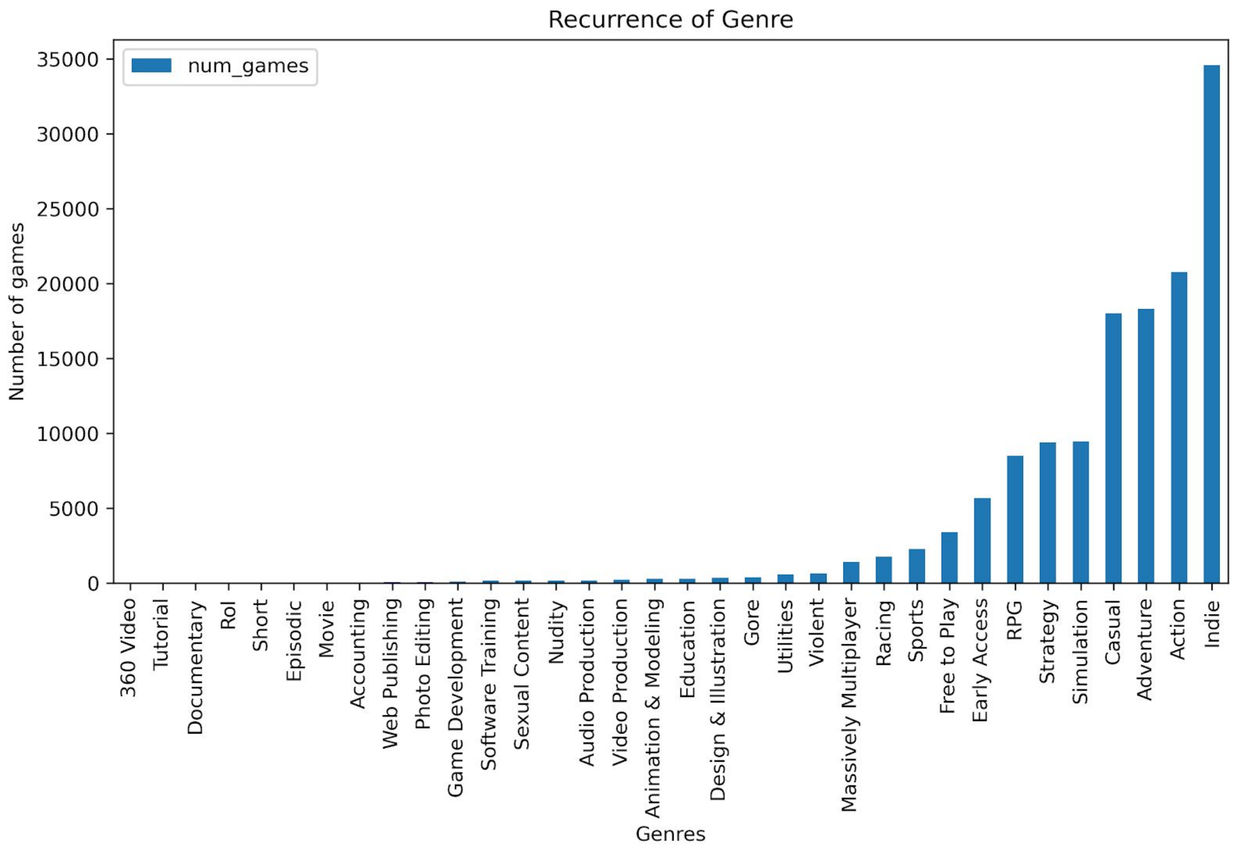
The second dataset is the app/game dataset. It contains a list of games, their steam tags, the game title, a short description,all reviews (negative/positive), popular tags (Gore, Action, Shooter, PvP…), game detail (Multi-player, Single-player, Full controller support…),genre (Action, Adventure, RPG, Strategy…). The collection is already discussed above . There are a total of 49007 games in the dataset.

To understand better how the game reviews are distributed, we plotted the amount of games with their respective percentage of positive reviews.

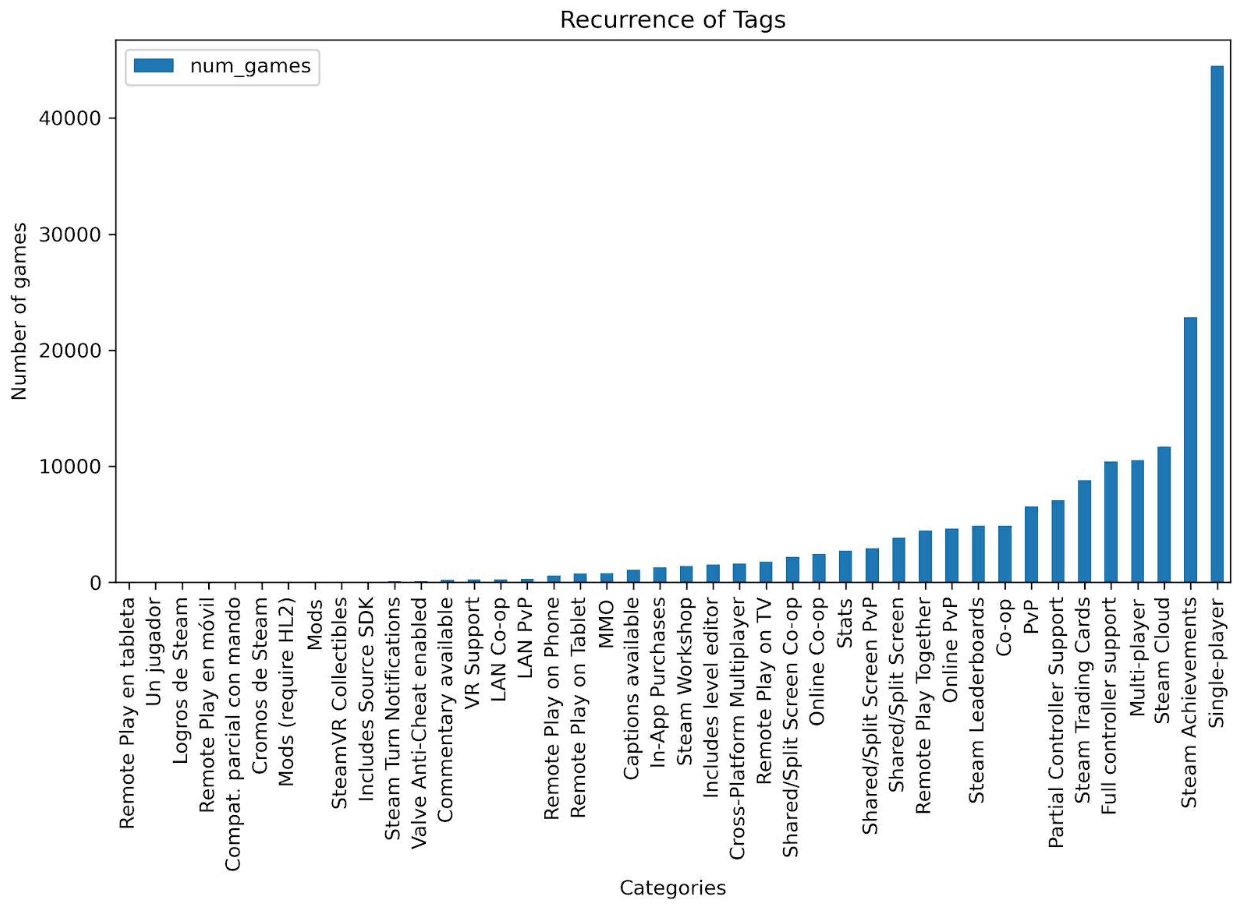
Also during the cleaning process a lot of garbage values were there and were shifted to zero. There were no games without positive reviews, so all the Zero values are those of garbage values.



The plot below lists all the game genres available in the game dataset with their respective number of games.

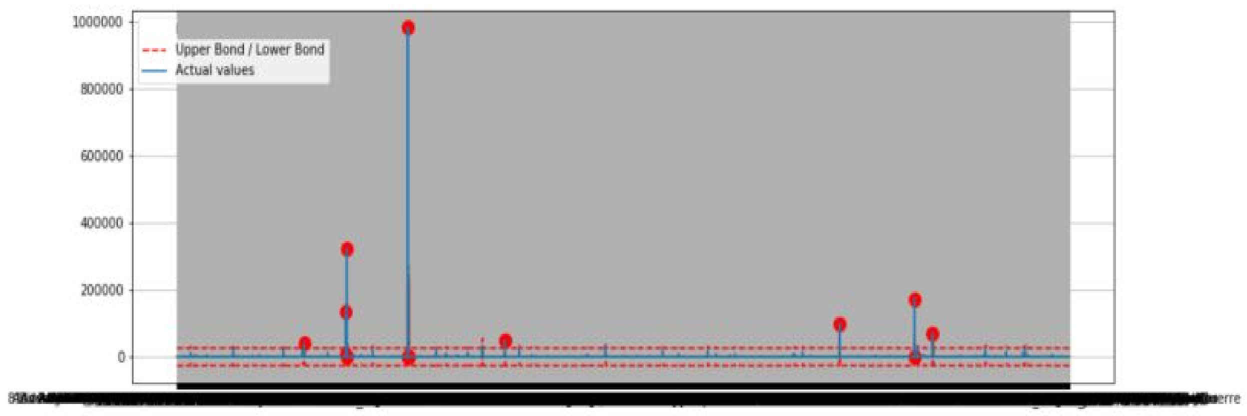


We generate a similar plot, showing the top 20 most popular game tags available in the game dataset with their respective number of games.



Anomaly detection:

After cleaning the data we applied anomaly and outlier detection detection using The Local Outlier Factor and moving average method to detect outliers of the dataset.



We can see the outliers marked in red dots in the plot.

Methodology:

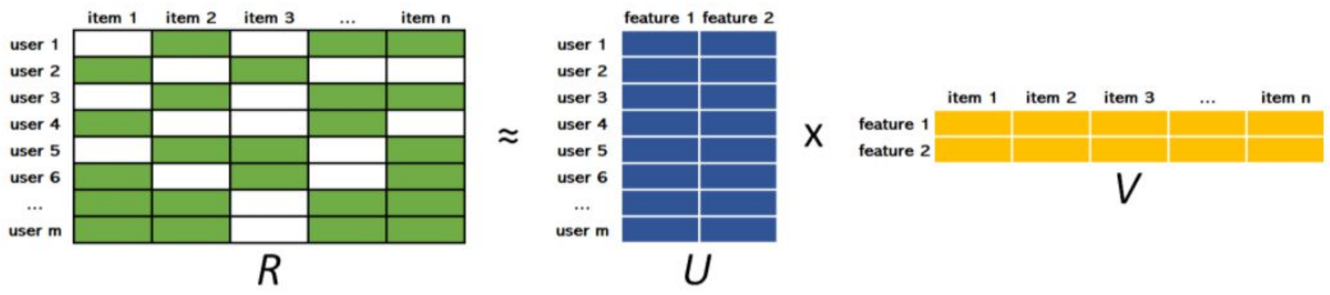
In order to produce recommendations we made two variations of collaborative filtering methods :

1. Collaborative Recommender with ALS
2. Collaborative Recommender with EM and EVD

Collaborative Recommender with ALS:
The Alternating Least square(ALS) is the model used to fit the data and also generate recommendations.

We have directly used the implementation of the ALS in the implicit library as it is implemented in Cython, making it faster and also allows parallelization of code among the threads as opposed to manual implementation.

ALS uses matrix factorization, here we break “user vs all items ” as “user vs some feature ” and “item vs some feature”.



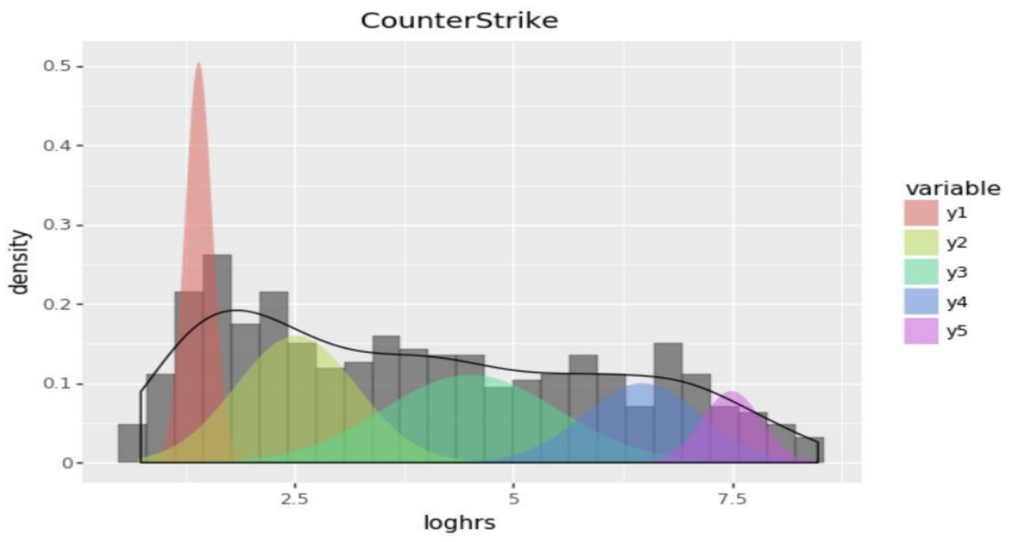
The goal here is to compute the weights of U and V such that R = UxV.
ALS alternatively optimises U and V such that the product approximately equals to R.

Collaborative Recommender with EM and EVD:

The Expectation-Maximization (EM) algorithm is an approach for maximum likelihood estimation in the presence of latent variables. It is an appropriate approach to use to estimate the parameters of a given data distribution.

In order to come up with a rating system , we decided to use the distributions of hours played for each game with the EM algorithm.

We are using the 5-star rating system.



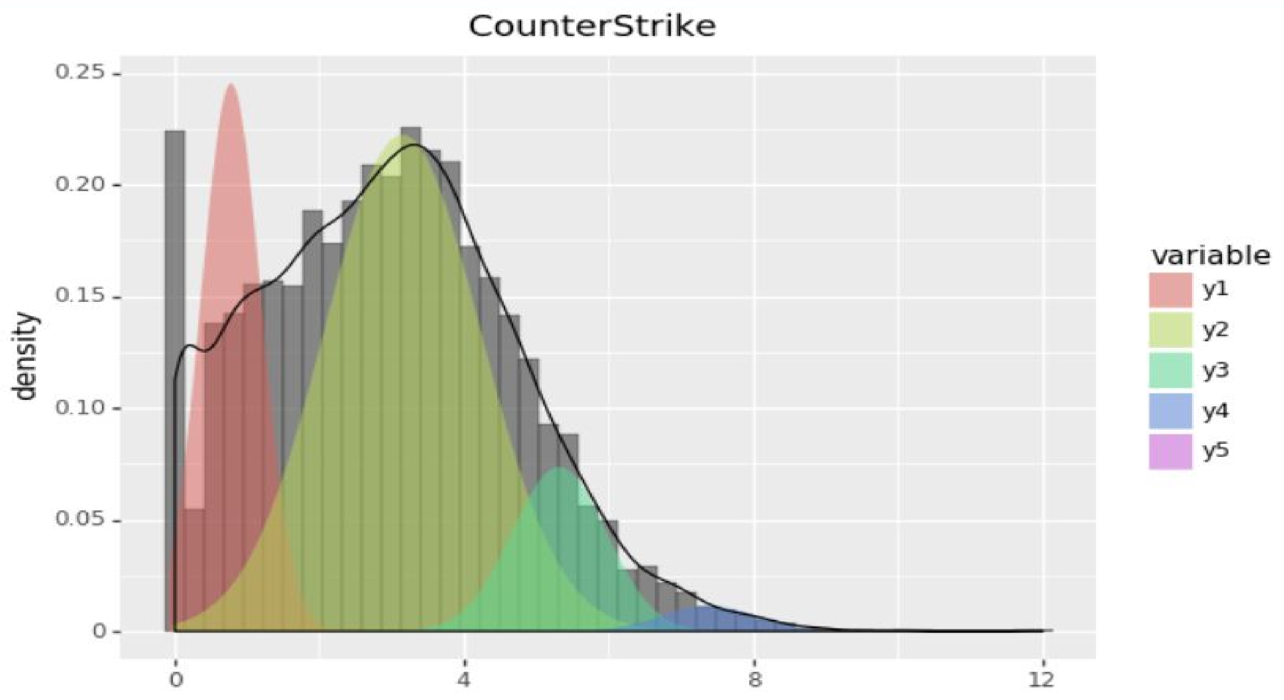
As we see EM algorithm did decent job in recognising the distributions but it could be better.

Now we apply the SVD algorithm to factorize the user-item matrix into singular vectors and singular values.

We kept the learning rate at 0.001 and number of iterations as 200 and tracked the RMSE.

Now we just apply EM-algorithm, post SVD .

Keeping the same 5-star rating which was described earlier.



Here we see the EM algorithm did a much better job than before in recognising the distributions.

Now we simply recommend top 20 similar games to the user:

```
top 20 recommended games for user 5250:
0 ) CitiesSkylines
1 ) FootballManager2015
2 ) GoatSimulator
3 ) Fallout3GameoftheYearEdition
4 ) AgeofEmpiresIIHDEdition
5 ) Terraria
6 ) TeamFortress2
7 ) MedievalIITotalWar
8 ) FootballManager2014
9 ) HalfLife2EpisodeTwo
10 ) StarTrekOnline
11 ) AmnesiaTheDarkDescent
12 ) FootballManager2012
13 ) HalfLife2
14 ) ChivalryMedievalWarfare
15 ) RedFactionGuerrillaSteamEdition
16 ) ScribblenautsUnlimited
17 ) CompanyofHeroes
18 ) Metro2033
19 ) NEOTOKYO
```

Evaluation:

Now to compare both the algorithms we calculate the ratio of the number of games in the user test dataset that are among the top 20 recommendations over the total number of games in the user test dataset.

$$ratio = \frac{Number\ of\ Games\ User\ has\ in\ Test\ Dataset\ that\ are\ among\ Recommendations}{Number\ of\ Games\ User\ has\ in\ Test\ Dataset}$$

And the results are stored as csv files under the evaluation folder.

Full project hosted on GIthub can be found at this link link click here.

Thank You..!!