**Project Report: Video Game Recommender Systems**

Abstract

The world of retail is changing rapidly. Many marketers have been trying to figure out ways to pique a gamers interest, from their games purchases and their game play history, or from their interests such as action, sports, or any other types of games. Steam is one of the largest video game digital distribution services for computer games. With all their data collected from the players, we can find ways to provide game suggestions based on their likes and interests to create a better shopping experience that can potentially boost sales. A popular solution that can be used to this problem is to build a recommender system.

Problem statement

First, a recommender system is a subclass of information filtering system to predict the preference or interest in which a user gives to an item. These days, many retailers such as Amazon have been using recommender algorithms meanwhile, some other newer sites are still in need. For this project, we will be looking to build a recommender system to recommend games to users based on their preferences and their gaming habits.

The dataset that will be used in this project come from Kaggle ([Steam Video Games | Kaggle](https://www.kaggle.com/datasets/tamber/steam-video-games)). It consists of the following columns: userid, game title, user behavior and value. The behaviors consist of “purchase” and “play” and the value indicates the degree to which the behavior was performed. When the behavior is “purchase”, the value is always 1 while if the behavior is “play”, the value represents the number of hours the user played the game.

We will first use a popularity-based recommender system and followed by the Alternating Least Square (ALS) collaborative filtering to provide recommendation.

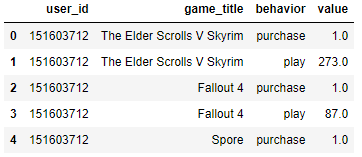
Numerical Results and Findings

1. **Popularity-based recommender system**

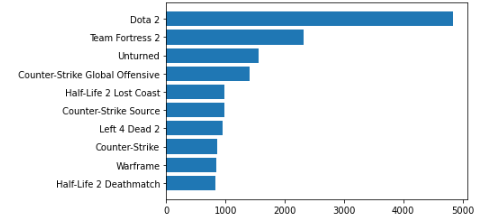
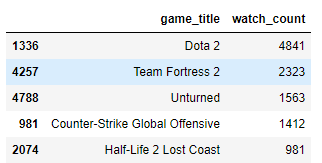
First, we explore the data, trying to understand the dataset from Steam, one of the largest video game digital distribution services for computer games.

Before going for anything too complex, we decided to start with something very simple: designing and build a popularity-based recommender system.

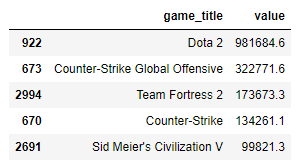
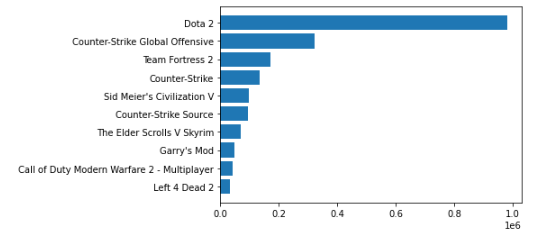
Here is how the dataset looks like:



From here, we can first filter the behavior column to see all the games purchased. With a popularity-based recommender system, we can do basic statistics to evaluate how count how many unique games and which is the most purchased games.

After that, we can also do another filter of the behavior column to see all the games played. Again, with the popularity-based recommender system, we can do the stats and evaluate again the count of the unique games and finding which games got the most hours played.

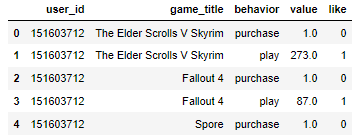


Based on the popularity-based recommender system, we can observe that the most purchased game and the most played game is Dota 2.

In the next step of the project, we will investigate using different system algorithm: ALS collaborative filtering.

1. **ALS collaborative filtering**

For this part, we first try to split the data with 2 labels, column named ‘like’. If the game is played more than 60 hours (a number I just randomly select), then it means that the user really likes the game (labelled as 1). If the game is played less than 60 hours by the user, then we label it as 0.



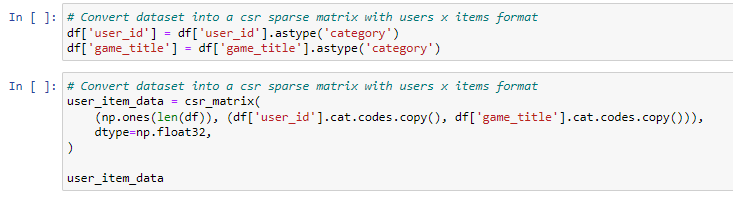
After that, we created a subgroup to identify which games are the most liked by summing up the number of likes for each game. This is the results:



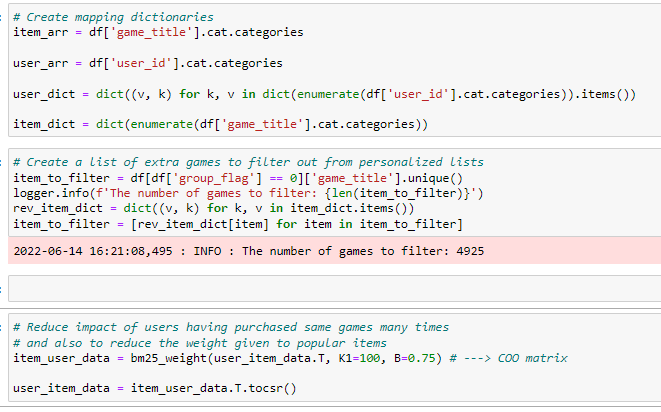
The next step is to create a group flag to attach this subgroup into the original dataset:



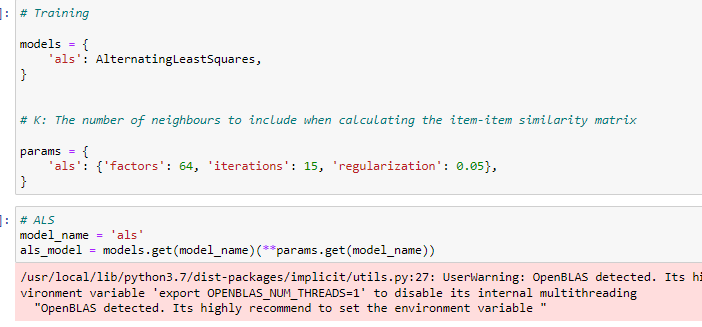
Now, we can do the sparse matrix with the user\_id and the game\_title, both as category.

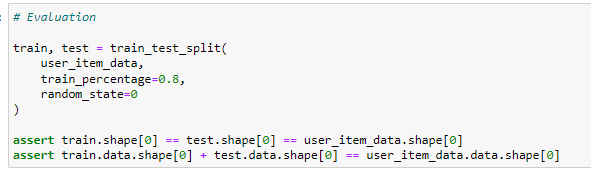


Then we can now create mapping dictionaries, list of extra games to filter from personalized list, reduce impact of users having purchased the same games many times and reduce the weight given to popular games.



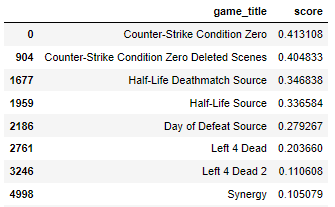
Now, we use the ALS model with the following parameters:





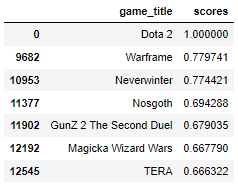
From there, we then tried to use the model to do predictions. First, from the dataset, we select

user\_id = ’5250’ to do a single user prediction. Here are the results:



As we can see, Counter-Strike games are the top recommended games for the user 5250.

We also did an item-to-item recommendation using ALS, taking ‘Dota 2’ as game and here are the results:



For the item-to-item recommendation, for users who like Dota 2, the top games are Warframe, Neverwinter and Nosgoth.

Conclusion

To conclude, we tried using popularity-based recommender system and ALS collaborative filtering. The best recommendation algorithm is the one that helps you reach your goal fastest. If ever our goal is to find which is the most popular game regardless of users, then we would go for the popularity-based model. However, if our goal is to find the best recommendations for individual users based on their purchases and time spent playing the game, then a collaborative filtering would suit better. Overall, it is an interesting exploration in understand how to use those algorithms to recommend games. If time allows, we may want to try other types of collaborative algorithm and do comparisons. As a first project on recommender systems, we are satisfied of the knowledge gained experimenting Steam’s dataset and we understand better the complexity behind that.

References

[Building a Recommender Engine Part I: Problem Statement & Collecting Data with Selenium | by Rachel Koenig | Medium](https://medium.com/@rachelmkoenig/web-scraping-recommender-systems-project-1d360fa678e4)

[Game-Recommendation-System (audreygermain.github.io)](https://audreygermain.github.io/Game-Recommendation-System/)

[Steam Video Games | Kaggle](https://www.kaggle.com/datasets/tamber/steam-video-games)