23-02 DM Team Project

To recommend new games using user’s data

Group 6: 김태윤. 양성호. 정화식.

Abstract

This project focuses on the issue of recommending new games and tagging each user's characteristics that are suitable for users of the Steam Game Platform. Understanding and Analyzing users' traits like game preferences and behavior is the main motivation, through which the goal is to provide personalized game recommendations and personalized labels. The first goal is to analyze the game data owned by users and recommend games based on trends. The second goal is to suggest games that fit the user's characteristics. Our final goal is to find users' characteristics by their own games and add labels to each user.

To do this, we first crawl Steam user data and game data. A-priori and FP-growth algorithms are used to find frequency itemsets and association rules. With those results, singular value decomposition and content-based recommendation systems are being used to recommend games for each user. K-means algorithms are being used to cluster games and find each users' characteristics.

The analysis results are used to provide personalized game recommendations to users and show user specified labels. Ultimately, this research aims to explore ways to enhance user experience in using Steam Game Platform.

1. Introduction

Currently, the Steam platform offers numerous games and has millions of users around the world. This diversity offers users a wide range of choices, but at the same time makes it difficult to find the game that works best for them. Also because the Steam platform only shows user ID on the website, it is hard for users to find out other users' characteristics, which makes it hard for people to communicate with each other. The purpose of this project is to address these issues by providing Steam users with personalized game recommendations and tagging users with some traits that they have. This is an important approach that can improve the gaming experience of users and increase the utilization of the platform as a community.

The results of this study will contribute significantly to increasing the effectiveness of the Steam Game Recommendation System and also

will help improve user satisfaction with the Steam website as a community. The end goal is to make the game selection process easier and to make tag categories in detail.

2. Method

2.1. Architecture and Environment

2.1.1. Dataset Architecture

When designing this project, we thought that it would be easy to get Steam user's data by crawling the Steam website. However, when we started to code and try to get data, it took long hours. There were many errors like the 'HTTP too many requests'. With some trial and error, we finally crawled a huge amount of data, almost 30 thousand users as JSON format.

User-Games Dataset

We conducted crawling through the Steam Web API to extract the game ownership lists of actual Steam users. The crawling process involves initially extracting a list of users, then accessing the game ownership lists only if the user has set their profile to Public. The extraction of user lists begins by starting with a specific user and querying their friends list. After crawling the entire list, we store it in a stack or queue and design the process to visit and extract information from as many users as possible using BFS or DFS algorithms. With this method, we crawled almost 30 thousand candidate users.

This dataset has been preprocessed under the assumption of utilization within the Spark environment. Consequently, the dataset comprises a total of three columns: UserID, serving as a unique identification number for each user; GameID, representing the games owned by the respective users; and PlayTime, indicating the amount of time the user spent playing a particular game. In its initial state, the dataset includes data from 4,771 users without applying a 120-minute filter. After applying the filter, the dataset encompasses 4,411 users and comprises information on 20,245 distinct game titles.

```
Data columns (total 3 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   UserID    653374 non-null  object
 1   GameID    653374 non-null  int64
 2   PlayTime  653374 non-null  int64
dtypes: int64(2), object(1)
memory usage: 15.0+ MB
```

Figure 1. Structure of the User-Games Dataset

```
[ ]  df['UserID'].value_counts()

     76561198102767019    7342
     76561197968410781    7298
     76561198264362271    6566
     76561198150396857    6328
     76561198032550800    5800
                          ...
     76561198992662761       1
     76561198259075441       1
     76561198364449275       1
     76561199195911941       1
     76561198965862097       1
     Name: UserID, Length: 4411, dtype: int64
```

Figure 2. UserID Column in User-Games Dataset

```
[ ]  df['GameID'].value_counts()

     730        3342
     578080     2189
     440        2080
     550        2056
     218620     1820
                ...
     2287970       1
     2432110       1
     1608270       1
     2306740       1
     702700        1
     Name: GameID, Length: 20245, dtype: int64
```

Figure 3. GameID Column in User-Games Dataset

During the visualization process of this dataset, it was observed that both the UserID and GameID columns exhibit extreme value distributions. Both columns manifested graphs with a pronounced 'Power Laws' form, wherein a majority of the values are concentrated among the top users. Subsequent efforts in the process took into account these characteristics of the dataset.

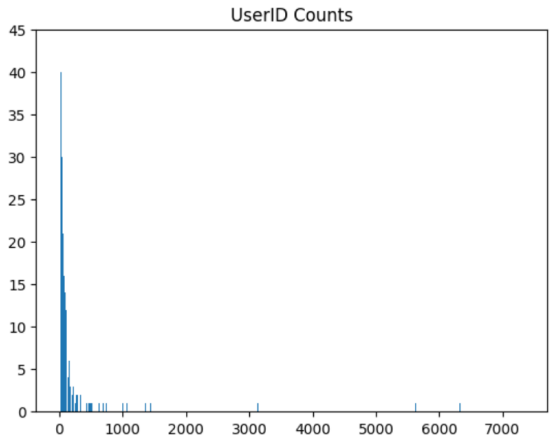

Figure 4. Graph with UserID Column Counts

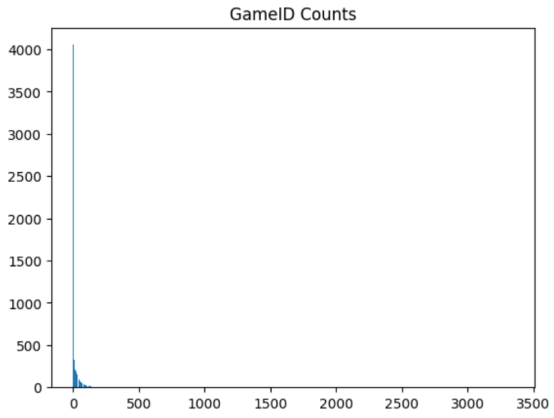

Figure 5. Graph with GameID Column Counts

Game-Feature Dataset

We could get a huge amount of data that contains a user ID and a list of game IDs that each user owns. To get all the games that users have, we extracted only games from the JSON file and used the Steam game URL to extract some features. For each game, there are various features like name, ID, required age, price, descriptions, etc. Even though there are many features, we thought that there are only three features that are important to represent the game: genre, required age, and price.

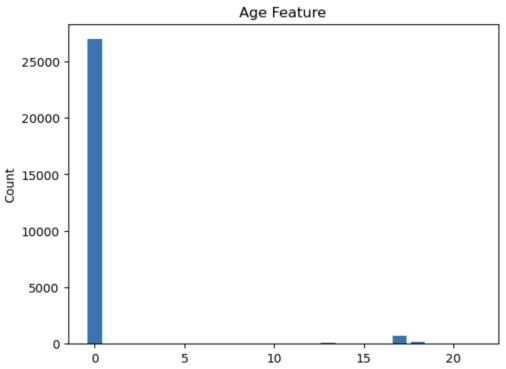

Figure 6. Graph with Age Column Counts

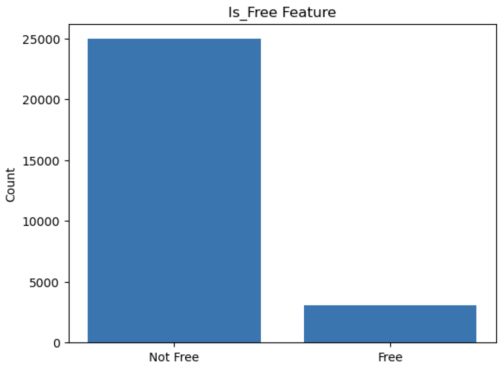

Figure 7. Graph with Is_Free Column Counts

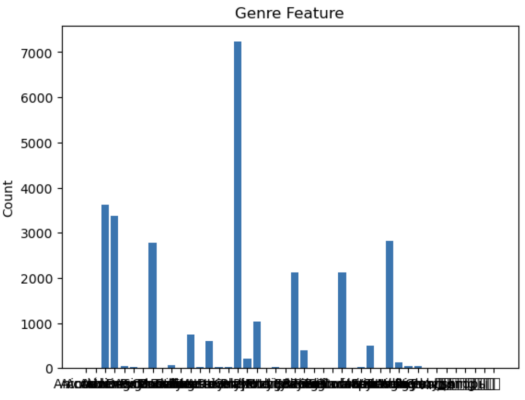

Figure 8. Graph with Genre Column Counts

Before using these features, we changed string values into integer values with different weights depending on how important the feature is. We put the highest weight value in the ‘Genre’ feature

because only this feature has relatively well distributed data.

GameID			
10	10	0	0
20	10	0	0
30	10	0	0
40	10	0	0
50	10	0	0
...	...	...	...
2432680	110	0	1
2434160	80	0	0
2435900	80	0	0
2439510	80	0	1
2439710	80	0	0

Figure 9. Structure of Game-Feature Matrix

Then we normalize these data by subtracting mean values and dividing standard deviation. From this normalized data, we made a huge matrix with an index of game IDs and features with genres, ages, and free that are all numerical values. Here is a normalized matrix and its visualized view.

GameID			
10	-1.307224	-0.200153	-0.351814
20	-1.307224	-0.200153	-0.351814
30	-1.307224	-0.200153	-0.351814
40	-1.307224	-0.200153	-0.351814
50	-1.307224	-0.200153	-0.351814
...	...	...	...
2432680	0.704861	-0.200153	2.842409
2434160	0.101235	-0.200153	-0.351814
2435900	0.101235	-0.200153	-0.351814
2439510	0.101235	-0.200153	2.842409
2439710	0.101235	-0.200153	-0.351814

Figure 10. Normalized Game-Feature Matrix

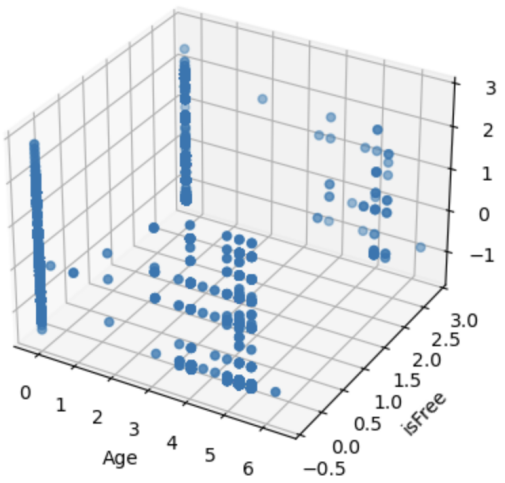

Figure 11. Visualization of the Game-Feature Matrix

Steam Games Dataset (Kaggle)

The game identifiers extracted through data crawling are in the form of integers known as AppIDs. Since it is impossible for users to immediately understand which game corresponds to each AppID, a mapping process is necessary to associate AppIDs with the actual game names. To accomplish this, we utilized the Steam Games Dataset available on Kaggle, utilizing the AppID and Name columns in the dataset to perform post-processing on the results.

This dataset contains 39 parameters and holds a total of almost 83.6k game data entries.

2.1.2. Project Environment

The code execution transpired under the default Colab configuration, employing Python 3 within the Google Compute Engine environment(CPU).

2.2. Analysis and Methods

2.2.1. Tag with Clustering

Assuming Euclidean space or distance, K-means clustering finds cluster centers that minimize the sum of squared distances from each point to its cluster center.

After we made a huge game-feature matrix, we could use K-means algorithms because all features are real numeric values and all in Euclidean space. Thus we use K-means algorithms to cluster all the games into 5 clusters and add these label information into the matrix as shown below.

GameID	Genre	Age	isFree	cluster
10	-1.307224	-0.200153	-0.351814	0
20	-1.307224	-0.200153	-0.351814	0
30	-1.307224	-0.200153	-0.351814	0
40	-1.307224	-0.200153	-0.351814	0
50	-1.307224	-0.200153	-0.351814	0
...	...	...	...	...
2432680	0.704861	-0.200153	2.842409	2
2434160	0.101235	-0.200153	-0.351814	3
2435900	0.101235	-0.200153	-0.351814	3
2439510	0.101235	-0.200153	2.842409	2
2439710	0.101235	-0.200153	-0.351814	3

Figure 12. Clustered Game-Feature Matrix

2.2.2. Frequent Itemset & Association Rules (FP-Growth)

One of the approaches for recommending games to users involves the utilization of Frequent Itemset & Association Rules. This method aims to identify specific items connected within the entire log of diverse items and discover rules that determine relationships among the identified items. We thought to apply this method to the User-Games Dataset introduced earlier, aiming to uncover meaningful relationships among items.

As for the specific method, we adopted the FP-Growth algorithm provided by PySpark, which was used in our previous HW2. Similar to the A-priori algorithm learned during lecture, FP-Growth is designed to reduce the time and cost involved in discovering Frequent Itemsets and Association Rules.

A brief comparison between the FP-Growth algorithm and the A-priori algorithm reveals that the A-priori algorithm has the advantage of a relatively straightforward implementation with its pruning operation. On the other hand, the FP-Growth algorithm leverages an FP-Tree data structure, offering superior execution performance in terms of speed.

However, there is a drawback of the A-priori algorithm as the number of items increases, leading to an exponential increase in computation, resulting in decreased speed. In contrast, the FP-Growth algorithm requires the creation of an FP-Tree, and its design and implementation are relatively complex.

2.2.3. Item-based Collaborative Filtering

The collaborative filtering method is a method of recommending to the corresponding user using the playtime of other users.

In this program, in order to utilize the collaborative filtering algorithm, the utility matrix was created using the pivot method of the pandas library.

Since the amount of rating data evaluated by each user is small compared to the total number of games, the overall utility matrix will be in a sparse form. Therefore, it was efficiently expressed by using the csr_matrix method of the Scipy library.

After that, the similarity between each game was measured using cosine similarity between items based on the playtime measured by users.

Now, given the preferred movie input from the user, the distance with each movie is obtained through the K-neighbors method of the Sklearn library, and then recommends games with the smallest distance compared to the game, that is, similar games.

2.3. Evaluation Criterion

2.3.1. Tagging a User

After clustering games, we evaluated our user specified tagging system by tagging two users who have very different properties of games. For example, user 1 has games that are mostly featured as ('Action', 'Free', 'No limited age') and user 2 has games that are mostly featured as ('Strategy', 'Free', 'Has age limits'). We could see that our system tagged them with very different numbers as shown below.

```
User1 Tag :  0.22727272727272727
User2 Tag :  1.4166666666666667
```

Figure 13. Tag of User1 and User2

The assigned tag number will be represented by a color and displayed alongside the user ID on the Steam website. This tag color will enable Steam community users to readily identify others with similar characteristics.

2.3.2. Recommendation Comparison with Real Data

We established the following criteria for assessing and evaluating the validity of recommendation results.

We selected user IDs not included in the current User-Games Dataset, and for each of these users, we chose a certain number of top-played games based on their playtime. These selected games were then input into our recommendation system to obtain results. Subsequently, we compared the results with the actual user-owned game records to gauge the degree of similarity as the evaluation metric.

3. Results

3.1. Tag with Clustering

Here is a visualization of clustering game-feature matrix.

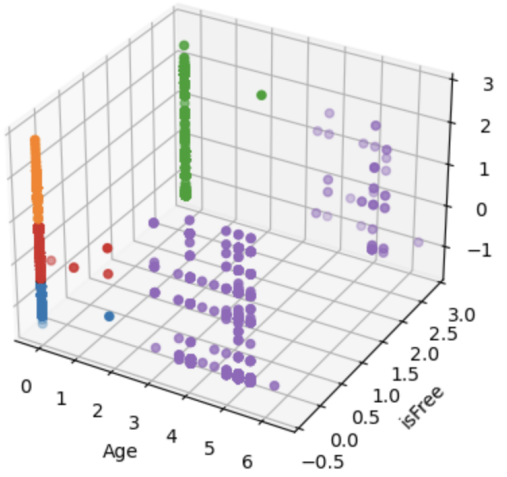

Figure 14. Visualization of Clustered Matrix

As we could see from this visualized cluster view , we could find out that the Age feature is used first to divide data into two, purple vs others. K-means algorithms also divide data into another two groups using isFree feature, green vs others. Finally, algorithms cluster 'free and no limited aged' games into three clusters with Genre feature, orange vs red vs blue.

According to these clustered results, we could tag each user by calculating the average value of cluster number. For example, if one user has games(game ID) that are 10, 20, 2439710 then cluster numbers are equal to 0, 0, 3. Then we could average those cluster numbers and get 1 as a tag number.

The assigned tag number will be represented by a color and displayed alongside the user ID on the Steam website. This tag color will enable Steam community users to readily identify others with similar characteristics.

3.2. Recommendation with FP-Growth

With conducted methods, we could make recommendation system that only require user's Steam64ID. If a user inputs his or her ID on the system, it recommends based on a higher lift score.

	index	antecedent	consequent	confidence	lift	support
1475	6077	['Portal']	['Half-Life 2']	0.520803	3.096043	0.082294
1531	6296	['Portal 2', "Garry's Mod", 'Left 4 Dead 2', '...	['Half-Life 2']	0.515065	3.061927	0.081387
1224	5048	['Portal 2', 'Left 4 Dead 2', 'Team Fortress 2...	['Half-Life 2']	0.504736	3.000527	0.084561
2120	8790	['Portal 2', "Garry's Mod", 'Team Fortress 2',...	['Half-Life 2']	0.504249	2.997633	0.080707
724	3017	['Portal 2', "Garry's Mod", 'Left 4 Dead 2']	['Half-Life 2']	0.503958	2.9959	0.086602
1884	7813	['DayZ', 'Counter-Strike: Global Offensive']	['Arma 3']	0.579027	2.966422	0.086375
1415	5828	['DayZ']	['Arma 3']	0.560709	2.872577	0.093176
2818	11634	['DARK SOULS™ III']	['Monster Hunter: World']	0.545455	2.623773	0.095217
1284	5304	['Portal', 'Left 4 Dead 2', 'Team Fortress 2']	['Portal 2']	0.860465	2.589026	0.083881
1444	5953	['Portal', 'Team Fortress 2', 'Counter-Strike:...	['Portal 2']	0.86019	2.588197	0.082294

Figure 15. Result of Recommendation with FP-Growth 1
(UserID : 76561198358526237 / Cluster ID : 0)

	index	antecedent	consequent	confidence	lift	support
480	4054	['XCOM: Enemy Unknown']	['XCOM® 2']	0.655052	4.799727	0.085241
1103	9115	["Sid Meier's Civilization® VI", 'Team Fortres...	["Sid Meier's Civilization® V"]	0.763948	3.003366	0.080707
38	332	["Sid Meier's Civilization® VI", 'PAYDAY 2']	["Sid Meier's Civilization® V"]	0.731076	2.874131	0.083201
848	7077	["Sid Meier's Civilization® VI", 'Left 4 Dead 2']	["Sid Meier's Civilization® V"]	0.729008	2.866001	0.086602
481	4059	['XCOM: Enemy Unknown']	["Sid Meier's Civilization® V"]	0.728223	2.862916	0.094763
205	1748	['FTL: Faster Than Light']	["Sid Meier's Civilization® V"]	0.72147	2.836367	0.084561
1165	9623	["Sid Meier's Civilization® V", 'PAYDAY 2']	["Sid Meier's Civilization® VI"]	0.540501	2.788478	0.083201
283	2415	["Sid Meier's Civilization® VI", 'Counter-Stri...	["Sid Meier's Civilization® V"]	0.673435	2.647523	0.090229
460	3903	["Sid Meier's Civilization® V", 'Left 4 Dead 2']	["Sid Meier's Civilization® VI"]	0.507979	2.620695	0.086602
51	431	['XCOM® 2']	["Sid Meier's Civilization® V"]	0.666113	2.618738	0.090909

Figure 16. Result of Recommendation with FP-Growth 2
(UserID : 76561198129480779 / Cluster ID : 1)

	index	antecedent	consequent	confidence	lift	support
83	11624	['DARK SOULS™ III']	['ELDEN RING']	0.538961	3.243325	0.094083
30	4379	['Starbound']	['Stardew Valley']	0.59596	2.71568	0.080254
58	8749	["Don't Starve Together", 'Terraria']	['Stardew Valley']	0.553134	2.520529	0.092043
33	4917	['Stardew Valley', 'Terraria']	["Don't Starve Together"]	0.622699	2.472302	0.092043
82	11498	['Middle-earth™: Shadow of Mordor™']	['The Witcher® 3: Wild Hunt']	0.596947	2.465479	0.088642
20	3443	['The Elder Scrolls V: Skyrim', 'Borderlands 2']	['The Witcher® 3: Wild Hunt']	0.590327	2.43814	0.094083
63	9048	['Stardew Valley', 'PAYDAY 2']	["Don't Starve Together"]	0.608547	2.416112	0.080707
31	4380	['Starbound']	["Don't Starve Together"]	0.607744	2.412925	0.081841
22	3506	['The Elder Scrolls V: Skyrim', 'Portal 2']	['The Witcher® 3: Wild Hunt']	0.580745	2.398565	0.084788
38	5365	['The Elder Scrolls V: Skyrim', "Sid Meier's C...	['The Witcher® 3: Wild Hunt']	0.579288	2.392546	0.081161

Figure 17. Result of Recommendation with FP-Growth 2
(UserID : 76561198066738500 / Cluster ID : 4)

As upper figures shown, users in different clusters can be recommended with different games. They can use this result to find their preference and new game.

However, this still has problems. Due to algorithm and approach limitations, this requires much time to recommend games. And also, this can't recommend new games or maniac games that aren't included in frequent itemsets. To resolve this, we can try other method like following content.

3.3. Recommendation with Item-based Collaborative Filtering

In the case of games recommended through the item-based collaborative filtering algorithm, it was confirmed that games of various genres were recommended regardless of the genre of the input games.

```
Recommendation criteria game: 30 Day of Defeat
1: 201330, distance: 0.4472663214425585
2: 2450, distance: 0.4877957670914522
3: 287740, distance: 0.49013143666284653
4: 58560, distance: 0.5016781410695867
5: 214610, distance: 0.5029951174700893

1. None
2. Title: "Bloody Good Time", genres: "Action"
3. Title: "The Witch's Yarn", genres: "Adventure, Casual, Indie"
4. Title: "Runaway: A Twist of Fate", genres: "Adventure"
5. Title: "Cherry Tree High Comedy Club", genres: "Action, Indie"
```

Figure 18. Result of Item-based Collaborative Filtering

4. Conclusion

By our conducted methods, we could make user tagging and

The process of extracting game data from the Steam API takes longer than we expected, and there are many aspects that require exception handling. The most regrettable part is that only three features could be used when clustering.

However, if our project's recommendation system and tagging system become more sophisticated with further research, we anticipate that the Steam website could become more active, attracting much more users.

5. References

- Martin Bustos, “Steam Games Dataset” Kaggle. [Online]. Available:https://www.kaggle.com/datasets/fronkongames/steam-games-dataset [Accessed: 28-Nov.-2023].
- Albert, Yefeng, Liang, “Steam Recommendation Systems” [Online]. Available:https://towardsdatascience.com/steam-recommendation-systems-4358917288eb [Accessed: 4-Dec.-2023].
- Scikit-Learn Developers, “sklearn.cluster.Kmeans” [Online]. Available:https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html [Accessed: 1-Dec.-2023].
- Jaeho Lee, “[이재호 강좌] 오렌지 제9강 연관 분석(Association Analysis)” Smatoos. [Online]. Available:http://www.sbr.ai/news/articleView.html?idxno=1582 [Accessed: 4-Dec.-2023].
- Zoumana, “Comparative Analysis Between Apriori and Fp Growth” Kaggle. [Online]. Available:https://www.kaggle.com/code/keitazoumana/comparative-analysis-between-apriori-and-fp-growth [Accessed: 4-Dec.-2023].
- Sergey Galyonkin, “Steamspy” [Online]. Available:https://steamspy.com/ [Accessed: 8-Dec.-2023].
- Sael Lee, (2023). Data Mining [PDF]. Available:https://eclass2.ajou.ac.kr/ultra/courses/_86684_1/cl/outline.