Steam game analysis

Xinfu Zhang/ Pengchong Wang / Jia He
EE
Columbia University
xz2737@columbia.edu pw2480@columbia.edu jh4001@columbia.edu

***Abstract*—One of the most important things a video game company looks is analyzing the relationship between players and games. Captivating more users mean more revenue especially with respect to the games which user will potentially like. Analyzing the relationship between games and players and making predictions about users' preferences are the keys to sustain gaming interests.**

Where does data analytics fit in here? The answer lies in data. As more and more data is generated online, data analytics can help set trends and ensure players return to the platform. This is quite very hard since data analytics should translate itself to game development if it integrates with the latter. Candy Crush, the most popular mobile video game that has more than 500 million downloads, relies on statistics and DS concepts. In fact, one study by Toby Walsh showed that this game is classified as a 'NP-hard problem' evident in computational complexity theory.

Computations present inside the game can be tweaked to improve gaming experience and as Walsh rightly mentioned in the end, "It would be interesting to see if we can profit from the time humans spend solving Candy Crush problems. Many millions of hours have been spent solving Candy Crush. "

Steam is a game platform where many game makers and producers upload and sell games on it. There are various types of games from 3A games to small self-design ones, which attracted players from all over the world. Besides its size and reputation, it offers a free open database for public. We think digging the inside potential information would be very valuable. So our job is to classify different players and make recommendation to their own interesting.

Keywords-component; steam game; relationship; classification; recommendation system

I. INTRODUCTION

Video game is taking larger and larger market in today's world, especially with the emergency of VR games. So, we focus on the online game hub "Steam" and its open database.

Game information analytics also play a very important role in keeping players hooked to the platform. Having excellent graphics and a better gameplay won't suffice. Insights from gaming data is highly appreciable. Adam Fletcher of Gyroscope Software neatly puts it this way, "It can seem odd to consider games, often characterized as artistic endeavours, as something you can measure and tune — more like a machine. Data science is no replacement to creativity and design. It is a complement to game design that can support those efforts and fill in gaps that design cannot." A specific area in DS, for example, data warehousing or data engineering, might not even be required for offline games. Furthermore, DS should also answer game-related scenarios since every game is different.

Another area which gaming companies explore is widening the install base. Also, in some instances, game may have a large user base but may not have many sessions or feature low session length when compared to the pre-launch estimates. Which means that players are no longer interested in the game. DS techniques, especially personalisation can be leveraged to increase user base and push up financial KPIs.

Steam is the world's most popular PC Gaming hub, with over 6,000 games and a community of millions of gamers. With a massive collection that includes everything from AAA masterpieces to small delicate games, so great discovery tools can be super valuable for Steam. What's more, Steam has an open database which includes various game information like the number of players, the price of the games and the rating of the games. We think the information inside the database are very useful and worth analyzing, like the actions within different user, the classification of users and the "game recommendation system" for users. We would like to analyze the information in database to show the relationship between the games and the players. In addition, we want to classify users and make recommendations to them according their preference.

II. RELATED WORKS

With over 2 billion players worldwide and an annual revenue of $20 billion in the U.S. alone, the gaming industry can no longer be overlooked as a niche sector. In America, the gaming industry is over twice as large as the movie industry ($8 billion revenue) and is expected to continue growing for the foreseeable future. This growth has resulted in a flood of player and usage data, such as:

- User play time;
- Interactions between players;
- Quitting point (when players leave a game);
- Peak server times, lag/ping rates, international connections;

...and, if a social media game, all of the user activity and profile data associated with player accounts.

The parallel growth of data analytics has resulted in a convergence between both industries, with Big Data playing an increasingly larger role in how the gaming industry collects and analyzes data. This intersection between gaming and analytics has resulted in the ability of gaming companies, such as Electronic Arts, to increase advertising

revenue, improve gameplay, and efficiently manage the user experience.

Gaming Analytics And Big Data:
Players interact with games in multiple ways, such as the device used, amount of time spent, playing style, dedication level, social media gaming usage, and in-game product purchases. The wide variety of gaming styles, coupled with the popularity of subscription & in-game revenue-earning opportunities, means that gaming companies need to customize and tailor their advertising content in order to maximize revenue.

Data analytics enables gaming companies to collect, cleanse, format, and model data in order to get a clear picture of how their users interact with their games. Over time, a holistic profile for each player emerges that conveys critical usage data, which enables companies to offer highly-specific products based on gameplay. For example, player characters who show an inclination towards armor customization will be offered in-game armor enhancement offers. Mobile device players who link their social media accounts will be offered in-game bonuses based on the number of social media friends they can recruit to the game, and so on.

Improving Gameplay Experience
Insights from gaming analytics also enable companies to improve the gameplay itself. For example, millions of player records could be analyzed to pinpoint the most likely in-game moments when players quit the game entirely; perhaps a series of quests are too boring or the challenges are too hard/easy based on character level. Identifying these gaming “bottlenecks” is critical to understanding the reasoning & timing behind a game’s churn rate. Gaming Designers and Developers can then re-examine the game’s storylines, quests, and challenges in order to refine the gaming experience and, hopefully, reduce the number of lost subscribers.

Analyzing the devices used by players also helps developers to create gaming experiences that work effectively for their user base. Exploring a dungeon via an iPhone is quite different than doing it using a widescreen attached to a laptop, so developers need to address issues such as screen size, available functionality, navigation, and character interactions. Data analytics empowers companies to address this challenge by modelling and visualizing massive amounts of heterogenous data.

Using Gaming Analytics To Improve Infrastructure
Today, games sometimes have global player bases… so the architecture supporting those users needs to be configured and implemented correctly. Online games are particularly prone to network-related metrics, such as ping and lag rates — these issues are exacerbated during peak gaming times. Again, Big Data analytics enables gaming companies to use server and network data to understand exactly when, and how, their infrastructure is being pushed to its limits. This knowledge enables companies to scale up or down according to player need; in today’s world of cloud-based PaaS/IaaS architectures (where cost is tied to usage), this information can have a dramatic impact on a company’s bottom line.

III. Algorithms and System

Part I Data Cleaning:
The first thing to do is deduplication. "Duplication" just means that you have repeated data in your dataset. This could be due to things like data entry errors or data collection methods. For example, if you're using a web scraper you may happen to scrape the same webpage more than once, or the same information from two different pages. Whatever the reason, duplication can lead you to make incorrect conclusions by leading you to believe that some observations are more common than they really are.

In this project, what we're going is that we have to find and remove duplicate records. (Removing duplicates is called "deduplication".) Here's a quick overview of what we have done:

- Visualizing duplication
- Finding & removing exact duplicates
- Finding & removing partial duplicates

Before we start, let's define what we mean by "duplication". Duplication can mean two slightly different things:
1. More than one record that is exactly the same. This is what I call "exact duplication".
2. More than one record associated with the same observation, but the values in the rows are not exactly the same. This is what I call "partial duplication", but removing these types of duplicated records is also called "record linkage".

Visualizing duplication:
visualizing is a very important part of different data cleaning tasks, and deduplication is no exception. Let's take a look at how many rows are duplicated in our dataset and where they are. There are a couple reasons we do these:

See how much duplicated data you have. If you only have a couple duplicates, or even none, you can just move on without worrying about them.

See if there are any patterns in duplication. One fairly common pattern is that you'll see big blocks of duplicates due to data being copy and pasted at the end of existing data. If that's the case, you can just remove those rows from your dataframe and call it a day.

To plot duplicates, I'm first going to create a dataframe with 1) a logical vector indicating whether or not a specific row is duplicated elsewhere in the dataset and 2) a numeric vector of the index of each row. (we are not using row numbers because if we are using the Tidyverse version of a dataframe, they get removed whenever you manipulate the

dataframe.) Then, we are going to plot that information so that each duplicated row shows up as a black line. Like so:

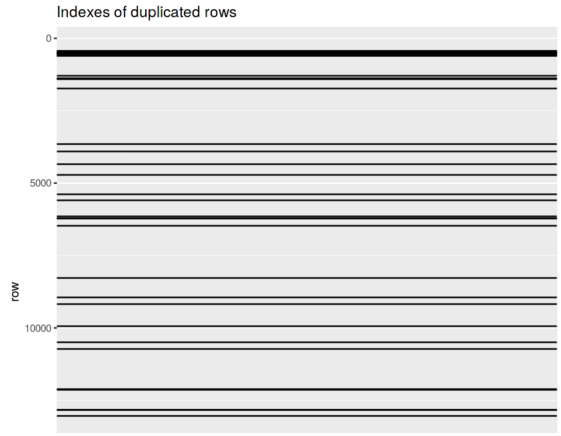

Figure 3.1. Visualization of duplications

Figure 3.1 tells us two things. First, we do have quite a bit of duplicated data, so we'll probably want to handle that somehow. Secondly, there's no clear pattern to the duplicated values, so it would be too time consuming to remove them by row index.

Removing exact and partial duplicates:

Removing exact duplicates is pretty straightforward.

Removing partial duplicates, on the other hand, is much more complex. Fortunately, there are some handy tools we can use to make the process easier. I particularly like the the dedup() function from the scrubr package by Scott Chamberlain. (If you haven't run into it before, the scrubr package can help you with a lot of different data cleaning tasks, like removing impossible latitude and longitude coordinates or standardizing dates.)

One of the hassles of removing partial duplicates is that it tends to take a lot of computation time. This is because you need to make a lot of different comparisons to determine which rows only partially match with other ones. It's also because, in this case, we're actually converting every row to a string and then comparing how similar the strings are (this can help us catch things like slightly misspelled names).

Part II classification:

K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum.

These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both k-means and Gaussian mixture modeling. They both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes. The algorithm has a loose relationship to the k-nearest neighbor classifier, a popular machine learning technique for classification that is often confused with k-means due to the name. Applying the 1-nearest neighbor classifier the cluster centers obtained by k-means classifies new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

A Bayesian Gaussian mixture model is commonly extended to fit a vector of unknown parameters (denoted in bold), or multivariate normal distributions. In a multivariate distribution (i.e. one modelling a vector {\displaystyle {\boldsymbol {x}}} {\boldsymbol {x}} with N random variables) one may model a vector of parameters (such as several observations of a signal or patches within an image) using a Gaussian mixture model prior distribution on the vector of estimates given by

Choosing value K:

The algorithm described above finds the clusters and data set labels for a particular pre-chosen *K*. To find the number of clusters in the data, we need to run the *K*-means clustering algorithm for a range of *K* values and compare the results. In general, there is no method for determining exact value of *K*, but an accurate estimate can be obtained using the following techniques.

One of the metrics that is commonly used to compare results across different values of *K* is the mean distance between data points and their cluster centroid. Since increasing the number of clusters will always reduce the distance to data points, increasing *K* will *always* decrease this metric, to the extreme of reaching zero when *K* is the same as the number of data points. Thus, this metric cannot be used as the sole target. Instead, mean distance to the centroid as a function of *K* is plotted and the "elbow point," where the rate of decrease sharply shifts, can be used to roughly determine *K*.

A number of other techniques exist for validating *K*, including cross-validation, information criteria, the information theoretic jump method, the silhouette method, and the G-means algorithm. In addition, monitoring the distribution of data points across groups provides insight into how the algorithm is splitting the data for each *K*.

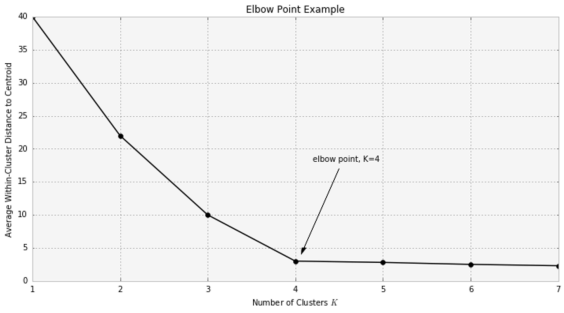

Figure 3.2. Range of K values

Part III recommendation:

A recommender system refers to a system that is capable of predicting the future preference of a set of items for a user, and recommend the top items. One key reason why we need a recommender system in modern society is that people have too much options to use from due to the prevalence of Internet. In the past, people used to shop in a physical store, in which the items available are limited. For instance, the number of video games that can be placed in a Blockbuster store depends on the size of that store. By contrast, nowadays, the Internet allows people to access abundant resources online. Steam, for example, has an enormous collection of games. Although the amount of available information increased, a new problem arose as people had a hard time selecting the items they actually want to see. This is where the recommender system comes in.

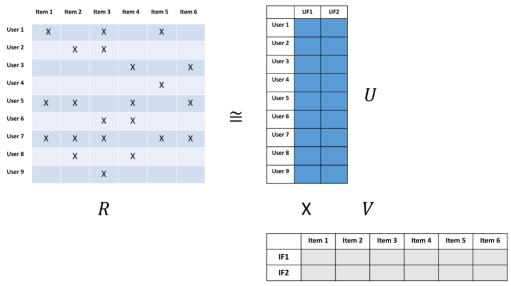

Figure 3.3. Collaborative Filtering

Collaborative Filtering does not require any information about the items or the users themselves. It recommends items based on users' past behavior. There are two categories of CF:

User-based: measure the similarity between target users and other users

Item-based: measure the similarity between the items that target users rates/ interacts with and other items

The key idea behind CF is that similar users share the same interest and that similar items are liked by a user.

Assume there are m users and n items, we use a matrix with size $m*n$ to denote the past behavior of users. Each cell in the matrix represents the associated opinion that a user holds. For instance, M_{i, j} denotes how user i likes item j. Such matrix is called utility matrix. CF is like filling the blank (cell) in the utility matrix that a user has not seen/ rated before based on the similarity between users or items. There are two types of opinions, explicit opinion and implicit opinion. The former one directly shows how a user rates that item (think of it as rating an app or a movie), while the latter one only serves as a proxy which provides us heuristics about how a user likes an item (e.g. number of likes, clicks, visits). Explicit opinion is more straight-forward than the implicit one as we do not need to guess what does that number implies. For instance, there can be a song that user likes very much, but he listens to it only once because he was busy while he was listening to it. Without explicit opinion, we cannot be sure whether the user dislikes that item or not.

The recommendation system in this use user-based way so I will explain it in detail.

User-based Collaborative Filtering

We know that we need to compute the similarity between users in user-based CF. But how do we measure the similarity? There are two options, Pearson Correlation or cosine similarity. Let u_{i, k} denotes the similarity between user i and user k and v_{i, j} denotes the rating that user i gives to item j with v_{i, j} = ? if the user has not rated that item. These two methods can be expressed as the followings:

$$u_{ik} = \frac{\sum_j (v_{ij} - v_i)(v_{kj} - v_k)}{\sqrt{\sum_j (v_{ij} - v_i)^2 \sum_j (v_{kj} - v_k)^2}} \qquad \cos(u_i, u_j) = \frac{\sum_{k=1}^{m} v_{ik} v_{jk}}{\sqrt{\sum_{k=1}^{m} v_{ik}^2 \sum_{k=1}^{m} v_{jk}^2}}$$

Both measures are commonly used. The difference is that Pearson Correlation is invariant to adding a constant to all elements.Now, we can predict the users' opinion on the unrated items with the below equation:

$$v_{ij}^* = K \sum_{v_{kj} \neq ?} u_{jk} v_{kj}$$

IV. Software Package Description

In classification system, there is one Jupiter notebook, we first use scatter plots to identify the pattern we have imagine before. Then, we use K-means and Gaussian Mixture algorithms to cluster the users.

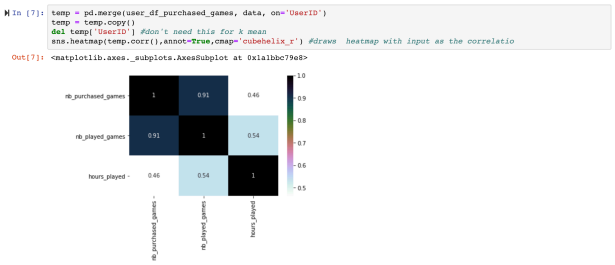

Figure 3.4. Correlation Matrix

In the recommendation system there are 2 Jupiter notebook, one is main part which uses CF to make recommendation and the other one is some codes to make some preprocessing. In the preprocessing part we reorder the games and users, giving them a new id like following:

The main recommendation system code shows answers in the last three frames, one is train and test the precision and last frame is recommendation, they will be shown in the following part.

UserID	Game	Hours_Played
5250	Team Fortress Classic	144.0
76767	Worms Armageddon	365.0
86540	XCOM Enemy Unknown	113.0
103360	Team Fortress Classic	0.0
144736	Team Fortress Classic	0.1
181212	Team Fortress Classic	1.8
229911	Worms Reloaded	165.0
298950	XCOM Enemy Within	1019.0

Original one

	0	1
0	GID	Game
1	1	007 Legends
2	2	0RBITALIS
3	3	1... 2... 3... KICK IT! (Drop That Beat Like a...
4	4	10 Second Ninja
5	5	10,000,000
6	6	100% Orange Juice

	0	1
0	1	5250
1	2	76767
2	3	86540
3	4	103360
4	5	144736
5	6	181212
6	7	229911

Figure 3.5. preprocess on data to make recommendation

IV. Experiment Results

User Classification

First, we will plot how many times each game is purchased. Also we will plot how many hours it is played to see if the most purchased games are the most played.

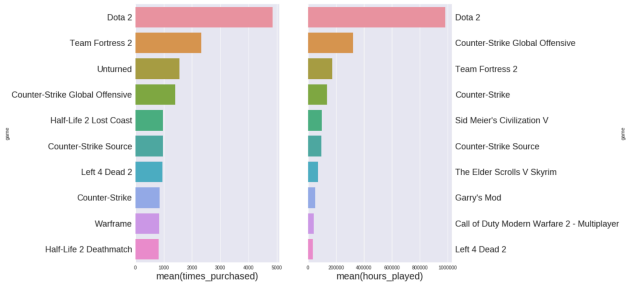

Figure 4.1. Number of games purchased

Now let's sort the most active users by hours played and have a glance at the game they play:

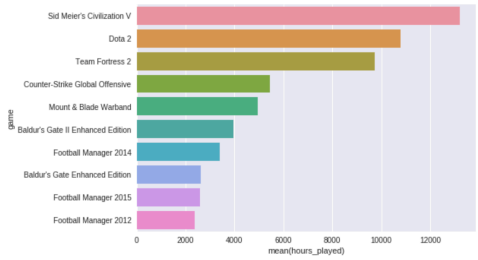

Figure 4.2. Most Active Users

By testing several value for the number of top gamers, we observe some surprising to-played games like CIvilization V being the most played within the top whereas only on the 5th position of the most-played games. Same thing for Mount & Blade Warband, not appearing in the top10 most played games for all users but present in the top with we only keep the most active players. We will now try to observe this distribution of played time for those particular games, and maybe differenciate the famous games played less than some less known games but causing hardcore gaming.

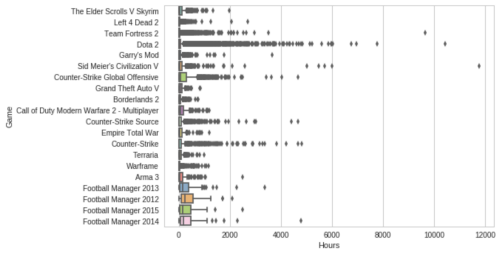

Figure 4.3. Top Games

As we can see, there are some games staying quite close to the mean played games, whereas some others like FM or Civilization have some vert extremal played times.

In this we will try to find some clusters for users, to determine profile of gamers. To do so, we will try to use :

- Number of Purchased Games vs Hours Played
- Number of Played Games vs Hours Played

1. Number of Purchased Games vs Hours Played:

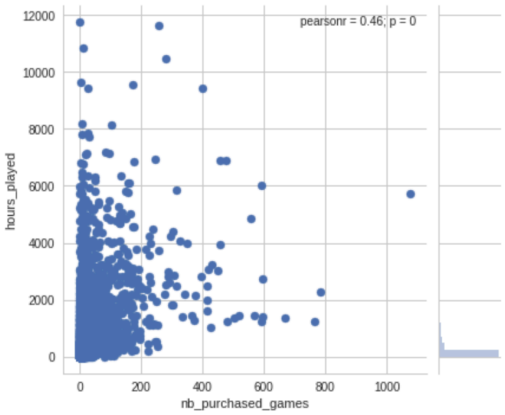

Figure 4.4 Number of Purchased Games vs Hours Played
Hard to find a pattern in this ... Let's try the another pattern: number of games played vs hours spent on games.

Then we will do the same thing on Number of Played Games vs Hours Played, with log scale on y axe:

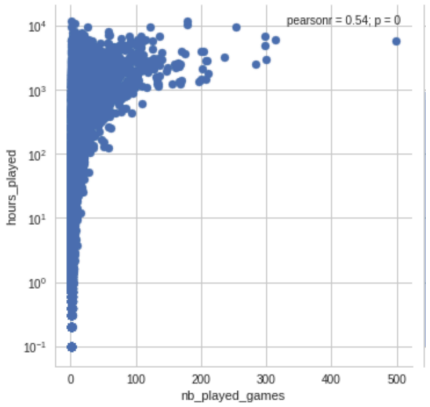

Figure 4.5 Number of Played Games vs Hours Played
To much noise in the dataset, we should try to eliminate those absurd values to focus more precisely on the core of the points:

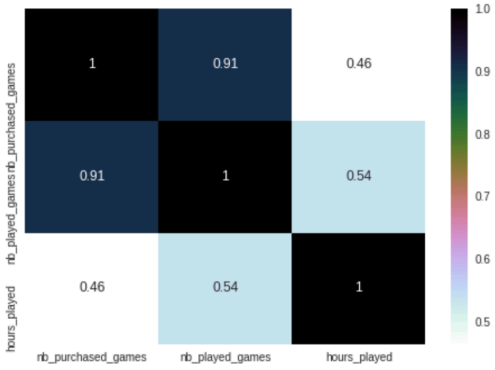

Figure 4.6 Correlation Matrix
Logically, we have a strong correlation between the number of **played games** and the number of **purchased games**
Then will be k-means algorithm and Gaussian Mixture:

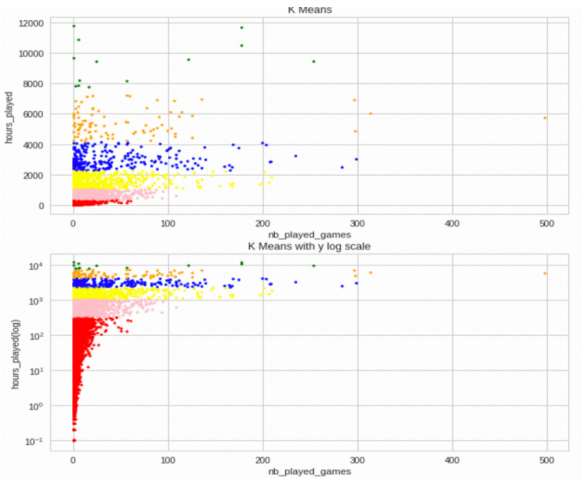

Figure 4.6. K-means Result and Log on Y Axe

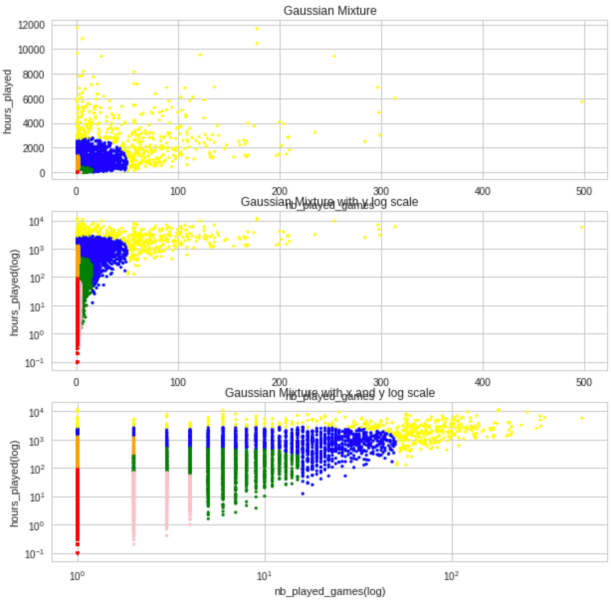

Figure 4.7. Gaussian Mixture Result and Log on Both Axe
We just tried several value of k ([0..6]) locally and kept the best one according to me. In this case, I used K-means with unlabelled data so there is no way to measure some accuracy. The choose of the k value is therefore personnal. I was just trying to separate the Try Harder (one game but playing a lot) from the multi-gamers (several games but not so many time played).

Recommendation system
There are many factors will affect the precision. The first one is whether containing the games which the user bought but did not play. At first I removed these games directly, but the precision of the outcome is not satisfying, only about 10%. This is because steam sometimes has some discounts and users will buy the game they want to play but have no time to do that, as a player, I think this does not mean they dislike it. If they indeed dislike the game, they can return the

game instead of keeping it. So I join these two columns and set the game time users buying but not playing to 1.
Second question is I have more than 5000 games while some users only bought less than 10. It is difficult to recommend games for these people and I made an experiment. As we can see, the precision becomes higher when I set the threshold that the least number of games a user have bought.

And to get a balance where most users can be recommend with a higher precision, I set the threshold to 20. So about half of the users can be recommend in the precision about 60%. If a player bought more than 60 games, the precision is satisfying which can unto 90%.

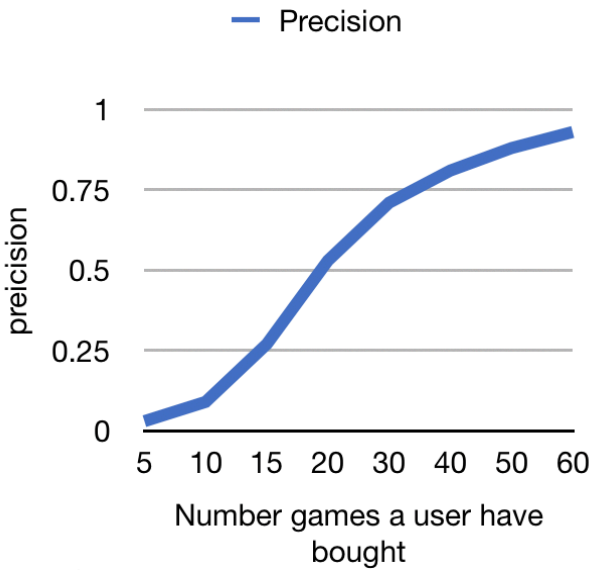

Figure 4.8. precision increase with number of games

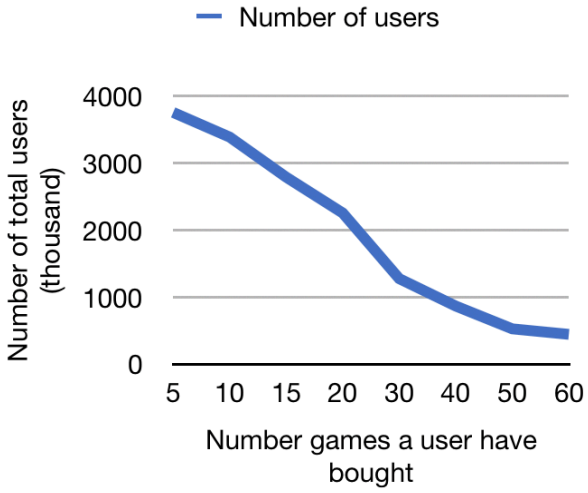

Figure 4.9. number of user decrease with number of games

The following picture shows the iteration process, after 100 times iteration the precision can be more than 0.6. I chose 80% of data to train the model and the left 20% data to test the precision. The average precision is about 60% is because the threshold is not very high, as mentioned former, it is not easy to recommend games from more than 5000 to a player only bought 20 or less.

```
            'Val Precision {:.3f}'.format(val_precision)
        )

rec = pred_pref.eval()
test_precision = top_k_precision(rec, test_matrix, k, test_users_idx)
print('\n')
print('Test Precision{:.3f}'.format(test_precision))
```

```
Iterations 0... Training Loss 1962907.25... Train Precision 0.141... Val Precision 0.231
Iterations 20... Training Loss 911040.69... Train Precision 0.297... Val Precision 0.507
Iterations 40... Training Loss 609829.00... Train Precision 0.376... Val Precision 0.537
Iterations 60... Training Loss 378611.22... Train Precision 0.418... Val Precision 0.604
Iterations 80... Training Loss 310507.56... Train Precision 0.438... Val Precision 0.649
Iterations 100... Training Loss 282468.47... Train Precision 0.459... Val Precision 0.642

Test Precision0.649
```

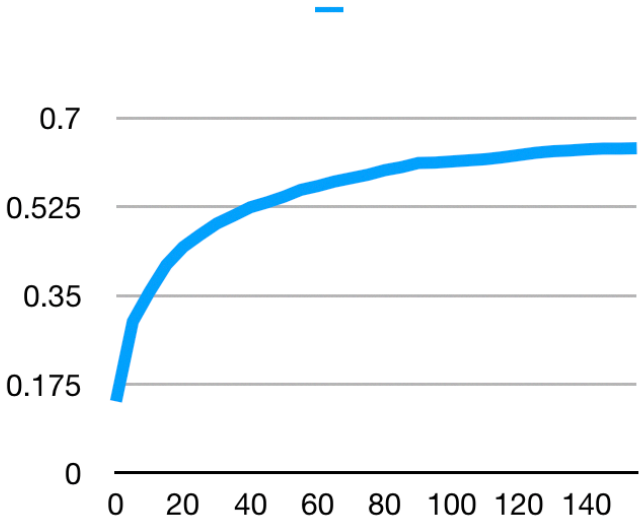

Figure 4.10. the process of iterateon

The following pictures shows a recommendation progress, it chooses a user randomly from the test group and compare the recommend game with the actually one

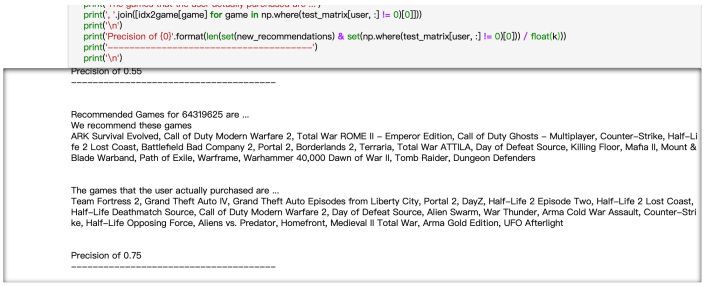

Figure 4.11. a recommendation process

V. Conclusion

In classification section, we think of two kinds of parallel relationship between users and games. The first one is if a user purchased more games, then he or she tends to spend more hours on games. The second one is if a user played more games, then he or she tends to spend more time on games. After we plot each pattern as a scatter image, we finally found that the second pattern is what we want—if a user played more games, then he or she will spend more time on games. This result lies the solid foundation for clustering.

In clustering, we tried two methods—K means and Gaussian Mixture. In K-means, we tried several K values, and we found that the value K=6 comes out the best result. From the image we can discover that the users are clustered into six gruops, and the differences between each group are the ration of hours spend on games and games played. From Guassian Mixture result, we not only get six groups, but also we can separate hard-core player (those who buy a little game but spend a lot of time on those specafic games) and play-boy player (those who played a lot of games but the total spent time is less).

With the result in clustering, we can get some insight on recommendation system. It is not easy to make recommendation for all steam users because the background of them is totally different. Some people like collecting and they will buy a lot of game. In this group, some people will try most games while some people not. On the other hand, some people only buy a few games and spend a lot of time on them. I think we should not make recommendation to them in the same way. To those buy a lot games and will play them is easy to make recommendation and the precision will decrease with the decrease of the number of games a user has.In sum, the recommendation system will work after setting some thresholds and the precision is also satisfying. So, in regard to recommending games, we need to classify them first and then make recommendation.
Acknowledgment

VI. Appendix

Contribution:
Jia He: preprocessing part
Pengchong Wang: relationship analytics and classification part
Xinfu Zhang: recommendation part

VII. References

1. Hacker Noon. (2018). *Introduction to Recommender System. Part 1 (Collaborative Filtering, Singular Value Decomposition)*. [online] Available at: https://hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75 [Accessed 18 Dec. 2018].
2. Cs224d.stanford.edu. (2018). [online] Available at: https://cs224d.stanford.edu/lectures/CS224d-Lecture7.pdf [Accessed 18 Dec. 2018].
3. En.wikipedia.org. (2018). *Collaborative filtering*. [online] Available at: https://en.wikipedia.org/wiki/Collaborative_filtering [Accessed 18 Dec. 2018].
4. K. Elissa, "Title of paper if known," unpublished.
5. Towards Data Science. (2018). *Various Implementations of Collaborative Filtering – Towards Data Science*. [online] Available at: https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0 [Accessed 18 Dec. 2018]
6. K-means clustering. (2018). Retrieved from https://en.wikipedia.org/wiki/K-means_clustering
7. Hamerly, G., & Elkan, C. (2004). Learning the k in *k*-means. Advances in neural information processing systems, 16, 281.