Recommender System for Steam Games: A Comparative Study of Different Models

Abstract:

A recommender system filters information and makes predictions on the user preferences in order to recommend data items to the users. It is commonly used in online platforms. Steam, a popular gaming platform, utilizes the recommender system to generate and present personalized games to the users. In this essay, we would explore the applications and performances of three models: K-means model, Gaussian Mixture Models (GMM) and Item-Based Collaborative Filtering, in the Steam recommender system, by designing and building up algorithms, after which implementing tests using multiple datasets. The goal is to evaluate the strengths and limitations of each model, then conclude with the most suitable model for the Steam games, in order to provide some insights into enhancing the recommender system of Steam. The result of our research shows that, clustering methods(K-means and GMM) tend to provide overlapping recommendations, which negatively influenced the overall system performance; meanwhile, the item-based collaborative filtering model offered a distinct perspective in recommending new items, which improves the diversity and serendipity in the recommender system.

1.Introduction:

Steam, a famous gaming platform which offers access to thousands of games, is by far the largest platform for PC gaming. As well as other online platforms, a recommender system is used in Steam to recommend games to users according to their preferences. The suggestions made are based on previous user behaviors, including adding items into buying lists, purchasing items, total playing hours for each game, rating and comments made on games. The Steam’s recommender system is driven by modeling algorithms and implementing machine learning, A recommender system is a subclass of information filtering systems. It predicts the user's desired outcome, therefore produces a recommendation list for users, which allows them to discover games that match their preferences. It is vital for enhancing user engagement and satisfaction, and its development has transformed the way nowadays users interact with online platforms.

In practice, however, the performance of recommender systems is not always as expected. Popular games, such as those produced by large gaming companies, may gain more opportunities from the algorithms of appearing on the top recommendation list. By contrast, games produced by smaller companies or individual developers may draw less attention, thus the lack of user response may result in less opportunity of getting into the recommendation list. However, some users' preferences may incline more towards those unpopular games rather than big popular games, which means a failure in recommender systems.

In this paper, we are going to provide a comparative analysis of three recommender system algorithms: K-means model, Gaussian Mixture Models (GMM) and Item-Based Collaborative Filtering. The motivation behind our research is to deliver a comprehensive understanding of the most effective recommender system for Steam games, offering some suggestions on future development on gaming recommendation algorithms.

We built three recommender systems by using K-means, Item-Based Collaborative Filtering,, and Gaussian Mixture Models (GMM) respectively. The final recommendation by either clustering and filtering are based on user behaviors, including the playing hours of users for each game and the genres of games. Therefore, similar user preferences on game genres, user preferences on similar games and groups of users with similar preferences are analyzed. The user behaviors and data distributions are visualized and illustrated by graphs. The accuracy and performances of the three models are evaluated by real datasets. We designed algorithms for each model by using the Python modeling and the Pandas Library, and we utilized the Steam games dataset from the Kaggle [1].

2.Literature Review:

2.1 Recommender System of Steam

The Steam recommender system seems to be a “black box” to the public. It uses robust machine learning algorithms, analyzing data based on user behaviors and feedback(gaming history, purchasing habits, ratings, total playing times, comments, wish list, etc.), thus being able to predict user preferences. By such prediction, it produces a recommended queue available to users, which contains 12 gaming items. Moreover, it presents several recommended lists of games in the Steam store page, each list is the most popular with high ratings in a specific category where users may have shown interest before. By introducing the recommender system, Steam users are able to explore more new games that they may be interested in, with less time spent on searching and choosing. Meanwhile, the diversity and serendipity are enhanced. Those can largely improve the user experience and make discovery of a wider range of games available.

However, there are also shortcomings existing in this system. Like many recommender systems, Steam’s system tends to favor those popular games. This may be because of the larger investment made by great, developed gaming companies on game development and advertisement. Such popularity may lead to a confusion in the algorithm, which causes popular games to become more popular, while less popular games experience less recommendations despite the fact that they may have high quality content. Therefore, the diversity and accuracy in recommendations would be negatively affected. Such circumstances are called “Popular Bias”. Meanwhile, there also exists the “Cold Start Problem”, which is basically defined as the condition of lack of information on new games and new users. Without sufficient interaction data, it is hard to make predictions for accurate recommendations. Moreover, there is a concept called “Temporal Dynamics”, which indicates that user preferences are changing over time. The dynamic preferences may leave previous, old interaction data no longer meaningful, which means predictions based on such outdated data are not accurate as well. Those cases are all responsible for negatively influencing the overall performance of the whole recommendation system.

2.2 Other Works on Recommender Systems

2.2.1 Recommender system

Recommender system is a subclass of information filtering system. It predicts the user's desired outcome, therefore produces a recommendation list for users, which allows them to discover games that match their preferences. It is vital for enhancing user engagement and satisfaction, and its development has transformed the way nowadays users interact with online platforms. Basically, the recommender system consists of multiple algorithms based on statistical models. The algorithms of recommender systems can be classified into several categories: model-based, content-based, collaborative filtering, knowledge-based, hybrid, etc.

2.2.2 Model-Based Clustering Methods

In model-based algorithms for recommender systems, different clustering techniques are utilized. Clustering is a form of unsupervised machine learning technique that splits datasets into different clusters, where the same cluster has similar features. For recommendation systems, clustering methods can be used to identify groups of similar users or items. The most common approaches of clustering include K-means clustering and Gaussian Mixture Models(GMMs), which are also used in our work.

K-means clustering, a popular and basic clustering technique, essentially partitions the data set of N items into K different clusters. Such that, the sum of squared distances from the centroid of each cluster to all data points is minimized. This algorithm is extremely simple and efficient, despite having several limitations. In the recommender system, it can be utilized to identify groups of users who have similar behaviors and preferences.

Gaussian Mixture Models(GMMs), on the other hand, is a more advanced clustering technique. It is often used in probabilistic clustering, which attaches a probability of belonging to each cluster, instead of classifying an item to a single cluster. Each cluster is generated by the Gaussian distribution, and data distribution is expressed probabilistically. Unlike K-means, clusters in GMM are not assumed to be of a certain geometric shape. Thus, GMMs are considered to be capable of handling more complicated datasets with fewer limitations.

2.2.3 Collaborative Filtering

Collaborative Filtering can be classified into user-based and item-based. The former aggregate items from similar users, while the latter aggregate items from similarities in items. With regard to item-based collaborative filtering, like in our algorithm, this method basically focuses on similarities between items. From the previous user behaviors, such as user's rating and purchasing history, this algorithm identifies items similar to those that users previously interacted with and satisfied with. The similarity is determined by users' opinion. For example, if comments and ratings made on some particular items by the same users are generally the same, the two items can be considered to be similar. Therefore, recommendations are made on these similar items to the users.

3.Model and method description

3.1 K-means clustering model

3.1.1 Principles of k-means clustering

K-means clustering is one of the most common clustering algorithms, belonging to the partitioning method of clustering. The principle behind it involves initially defining k cluster centers, and then associating the samples with the nearest cluster center based on the computation of the distance between the samples and the centers. The algorithm iteratively refines these associations with the objective of minimizing the distance between the samples and their respective cluster centers.

$$arg_c \min \ J(c) = \sum_{k=1}^{K} \sum_{x^{(i)} \in \mathbb{C}_k} \| x^{(i)} - \mu^{(k)} \|_2^2$$

3-1 objective function

The optimization algorithm follows these steps:
1. Randomly select k samples as the initial cluster centers (where k is a hyperparameter representing the number of clusters, and its value can be determined based on prior knowledge or validation techniques).
2. For each sample in the dataset, compute its distance to the k cluster centers and assign it to the cluster corresponding to the nearest center.
3. For each cluster, recalculate the position of its cluster center.

4. Repeat the operations in steps 2 and 3 iteratively until a termination criterion is met (such as a specific number of iterations or no change in the positions of the cluster centers).

3.1.2 Distance metric

The k-means algorithm is based on the calculation of distance similarity to determine the nearest center point for each sample. Common distance metrics utilized in this context include Manhattan distance and Euclidean distance

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$

3-2 Manhattan distance

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

3-3 Euclidean distance

3.1.3 K value

The k-means algorithm partitions data into k clusters, and the choice of k can lead to significant differences in the performance of the algorithm. Common methods for determining the value of k include prior knowledge, the elbow method, among others

1. Priori method

The use of prior knowledge is relatively straightforward in determining the value of k, as it relies on domain-specific information. For example, in the case of the Iris flower dataset, where there are approximately three known categories, clustering validation can be performed with k set to 3.

2. Elbow method

It can be understood that as the value of k increases, leading to more partitioned clusters, the sum of the squared distances from each point to the cluster center (within-cluster sum of squares, or WSS) becomes lower. The value of k can be determined by identifying the elbow point, or the point where the curve of the reduction in WSS with respect to increasing k shows a noticeable bend.

3.2 Gaussian Mixture Models

The Gaussian Mixture Model (GMM) is a clustering algorithm that can be employed for data classification. The GMM algorithm assumes that data points are generated from one or more Gaussian distributions and estimates the parameters of the Gaussian distribution for each cluster using the method of Maximum Likelihood Estimation (MLE). In practical applications, the GMM clustering algorithm can be utilized in various domains. For example, GMM can be used to cluster facial images to more accurately recognize different faces. It can also be applied to cluster audio signals for more precise speech recognition. The following will provide a detailed explanation of the principles behind the Gaussian Mixture Model (GMM) algorithm

In GMM, it is assumed that the data is composed of several Gaussian distributions. The probability density function for a Gaussian distribution is given by:

$$f(x;\mu,\Sigma)=\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu)^{T}\Sigma^{-1}(x-\mu)\right)$$

The objective of GMM is to maximize the likelihood function of the data:

$$L=\prod_{i=1}^{N}\sum_{j=1}^{K}\pi_j f(x_i;\mu,\Sigma_j)$$

The Expectation-Maximization (EM) algorithm is used to estimate the parameters of the GMM. The process of the EM algorithm is as follows:

1. Initialize the parameters of the model.
2. Compute the responsibilities for all data points.
3. Update the parameters for each Gaussian distribution.
4. Compute the likelihood function of the model.
5. Determine whether the parameters of the model have converged. If convergence is reached, the iteration process is terminated. Otherwise, return to step 2.

Use the following formula to calculate the responsibility for each data point:

$$\gamma(z_{ij})=\frac{\pi_j f(x_i;\mu_j,\Sigma_j)}{\sum_{k=1}^{K}\pi_k f(x_i;\mu_k,\Sigma_k)}$$

Use the following formulas to update the parameters for each Gaussian distribution:

$$\mu_j = \frac{\sum_{i=1}^{N} \gamma(z_{ij}) x_i}{\sum_{i=1}^{N} \gamma(z_{ij})}$$

$$\Sigma_j = \frac{\sum_{i=1}^{N} \gamma(z_{ij}) (x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^{N} \gamma(z_{ij})}$$

$$\pi_j = \frac{\sum_{i=1}^{N} \gamma(z_{ij})}{N}$$

By continuously iterating, the GMM clustering algorithm can be used to cluster the data.

3.3 Item-Based Collaborative Filtering

Collaborative filtering is one of the earliest and most renowned recommendation algorithms. Its primary functions are prediction and recommendation. The algorithm uncovers user preferences by mining historical user behavior data, categorizing users into groups based on these preferences, and recommending products that align with similar tastes. Collaborative filtering can be divided into two categories: user-based collaborative filtering and item-based collaborative filtering. Simply put, it operates on the principle that 'like attracts like' and 'items are grouped with similar items.' In the following, we will explore the principles and implementation methods of collaborative filtering

The item-based algorithm first calculates the similarity between items, and several methods can be used to calculate this similarity:

1. Cosine-based similarity calculation, which computes the similarity between items by calculating the cosine of the angle between two vectors. The formula is as follows:

$$sim(i,j) = \cos\left(\vec{i}, \vec{j}\right) = \frac{\vec{i} \cdot \vec{j}}{\left\|\vec{i}\right\|_2 ※ \left\|\vec{j}\right\|_2}$$

2. Correlation-based similarity calculation, which computes the similarity between two vectors by determining their Pearson-r correlation coefficient. The formula is as follows:

$$sim(i,j) = \frac{\sum_{u \in U}\left(R_{u,i} - \bar{R}_i\right)\left(R_{u,j} - \bar{R}_j\right)}{\sqrt{\sum_{u \in U}\left(R_{u,i} - \bar{R}_i\right)^2}\sqrt{\sum_{u \in U}\left(R_{u,j} - \bar{R}_j\right)^2}}$$

Where represents the rating given by user u to item i represents the rating given by user u to item i,

3. Adjusted Cosine Similarity Calculation: Since cosine-based similarity calculation does not consider the rating habits of different users, (for example, some users may tend to rate higher, while others may tend to rate lower), this method eliminates the impact of different user rating habits by subtracting the average rating of the user. The formula is as follows:

$$sim(i,j) = \frac{\sum_{u \in U}\left(R_{u,i} - \bar{R}_u\right)\left(R_{u,j} - \bar{R}_u\right)}{\sqrt{\sum_{u \in U}\left(R_{u,i} - \bar{R}_i\right)^2}\sqrt{\sum_{u \in U}\left(R_{u,j} - \bar{R}_j\right)^2}}$$

Where represents the average rating given by user u represents the average rating given by user u.

(2) Prediction Calculation

Based on the previously computed similarities between items, the next step is to predict ratings for the items that the user has not yet rated. There are two prediction methods:

1. Weighted Sum.

Using the weighted sum of the ratings given by user u for the items that have been rated, the weights are the similarities between each item and item i, then taking the average of the sum of all item similarities, the calculation of user u's rating for item i is obtained. The formula is as follows:

$$P_{u,i} = \frac{\sum_{all\ similar\ items,\ N}\left(S_{i,N} \divideontimes R_{u,N}\right)}{\sum_{all\ similar\ items,N}\left(\left|S_{i,N}\right|\right)}$$

Where represents the similarity between item i and item N represents the similarity between item i and item N, and represents the rating given by user u to item Represents the rating given by user u to item N.

2. Regression.

Similar to the above weighted sum method, the regression method does not directly use the rating values of similar item N, as there is a misconception when calculating similarity using the cosine method or Pearson correlation method. Specifically, two rating vectors might be quite distant (in Euclidean distance) but still have a high similarity. This is because different users have different rating habits; some may tend to rate high, while others may tend to rate low. If two users like the same item, their Euclidean distances may be far apart due to different rating habits, but they should have a relatively high similarity. In this case, calculating using the original ratings for similar items can lead to poor prediction results. By re-estimating a new value using linear regression, and applying the same method above for prediction, the re-calculation method is as follows:

$$\acute{R}_N = \alpha \overline{R}_i + \beta + \epsilon$$

Where itemN is a similar item to item i, andα andβ are obtained through linear regression calculations on the rating vectors of itemsN andi, withϵ representing the error of the regression model.

4.Method Description

Steam is a digital distribution platform developed by Valve Corporation, which offers various games for users to buy, download, install, update, review, and share. Based on the diverse demands, the company had utilized a recommender system for providing reasonable games to each user. On the Steam homepage, divided sections suggest games that users may be interested in calculating by the algorithm.

With Steam's impressive range of offerings and its personalized recommender system, it becomes crucial to comprehend how these recommendations are formulated. For better analyzing and explaining the mechanism behind the recommender system, an accurate dataset is needed to generate an empirical model. As an open-source website, Kaggle provides public datasets for anyone interested in data science or machine learning. According to Kaggle’s terms and conditions, services of Kaggle allow users to copy or download the datasets.

The dataset used in this report is called Steam Recommender System founded on Kaggle. This dataset is a CSV file, which includes users, games, behaviors and play hours, but it does not have the types, we create it as user_id, game_title, behavior, playhours. “User_id” is based on the user id shown from the dataset, “game_title” is the name of each game, and the “playhours” is the number of hours from minimum hours played to the maximum hours played. However, the behavior is not used for follow-up research, which was not shown at the table. This table provides the basic information about this dataset.

Table 1

Type	Size	Description
user_id	12393	The number of users
game_title	5155	The number of games
playhours	298	The range of played hours

The model was generated using a computer with an Intel i9 Sixteen-Core Processor, 16GB of memory, and an NVIDIA GeForce RTX 4080 Graphics Card. For data organization, python libraries such as pandas and NumPy were employed. Additionally, all research processes were managed via the Anaconda navigator.

Table 2 Boxplot before deleting outlier

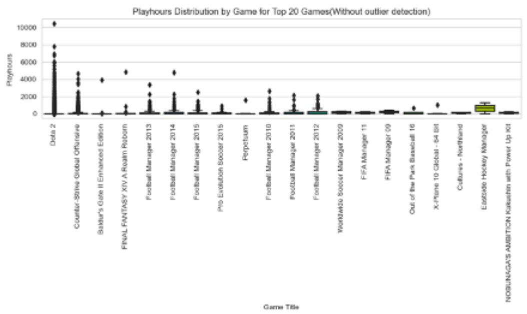
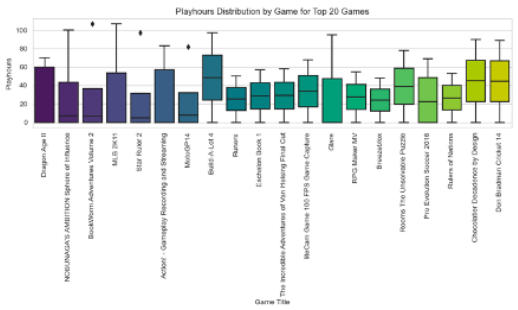


Table 3



These two box plots (Table 2 and Table 3) demonstrate varied outcomes when considering outlier detection, underscoring the significance of data cleaning for this research. Both box plots utilize the 5% and 95% quantiles as demarcation points to yield optimal results. They effectively display the top 20 games based on user playtime distribution.

Table 4

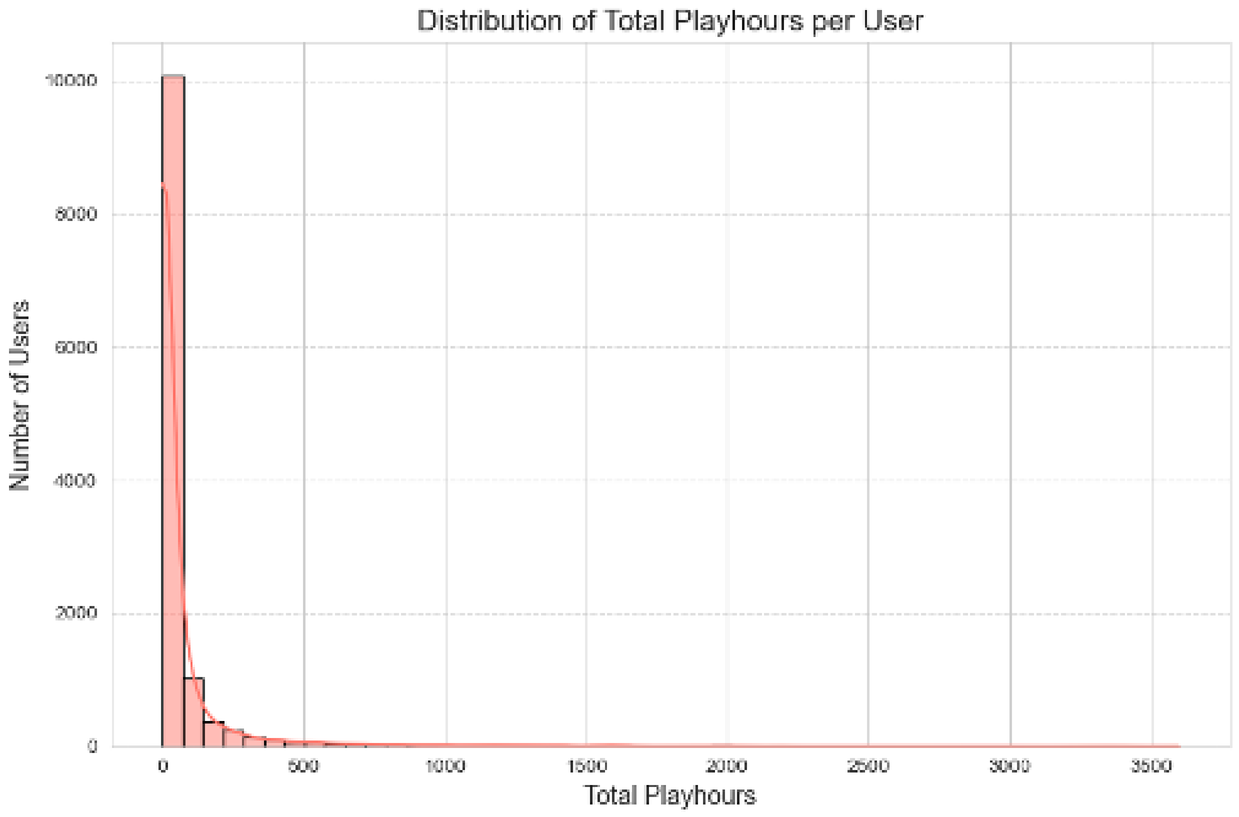
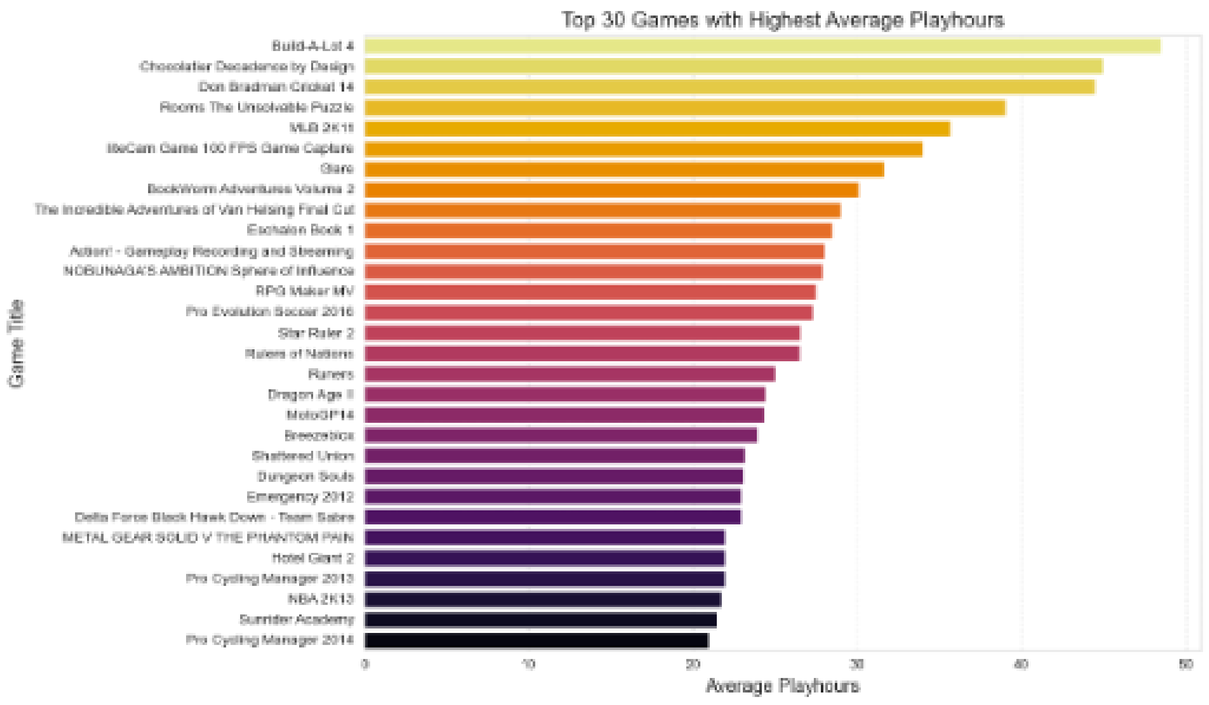
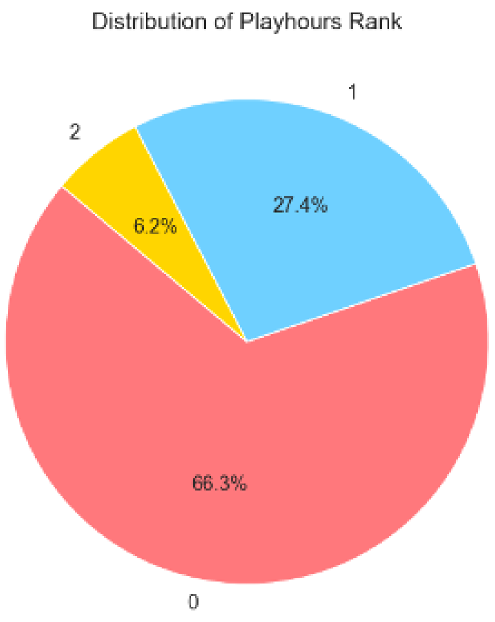


Table 5



To provide a clearer perspective on the dataset, we've illustrated the distribution of total played hours per user in Table 4, given that our research hinges on clustering based on played hours. Additionally, we've included a bar chart (Table 5) showcasing the top 30 games ranked by their average played hours to find the most popular games in this dataset.

Table 6



In examining Table 6, which presents a pie chart, we are drawn into the captivating dynamics of user engagement with games. Here, '0' embodies those enthusiasts who, for whatever reason, chose to purchase but not immerse themselves in gameplay. This segment, surprisingly, comprises a substantial 66.3%. On the other hand, '1' paints a picture of the more casual players, those who have dipped their toes into the gaming waters but haven't waded in too deep, clocking in less than 20 hours. This slice of the pie is a noteworthy 27.4%. Finally, the '2' group, although appearing to be little at 6.2%, represents die-hard gamers, those genuinely dedicated individuals who have played for more than 20 hours. This pie chart tells a story about gamers' enthusiasm for gaming, casual relationships, and squandered opportunities.

As mentioned before, this research contains three different models, which are K-Means clustering model, Gaussian Mixture Models (GMM), and Item-Based Collaborative Filtering.

First, we used K-Means clustering, a tried-and-true approach, to decipher our findings. We constructed a thorough graph and found the best K-Means number, which was a key component of our research. K-Means has limitations, particularly when dealing with different data area sizes and patterns that do not precisely match its grouping paradigm. Nevertheless, the model we develop using this clustering approach has some bias despite these drawbacks. We conducted a real-world test to demonstrate the effectiveness of our model by enabling it to curate game suggestions using a random user_id, simulating the dynamic interaction between data science and user-centric recommendations.

The Gaussian Mixture Model, often known as the normal distribution, was the next strategy we investigated in our study. This model offers a more adaptable, probabilistic approach to clustering, allowing for elliptical patterns, in contrast to the linear characteristics of K-Means clustering. The Gaussian Mixture Model was meticulously tuned before we put it to the test by feeding it a sample user profile. The conclusion was intriguing: it offered a selection of games that may be useful. Then, after careful inspection, we discovered that its recommendations were very similar to those produced by the K-Means model. Given the features of a particular dataset, this overlap highlights the subtleties of clustering algorithms and underlines how various approaches may converge to produce comparable findings.

We examined item-based collaborative filtering, a mainstay in the field of recommendation systems, for our third investigation. This strategy was broken down into two main stages: obtaining related things, and then estimating user ratings for those goods. It was unexpected to find that, even when applied to the same user profile, the item-based collaborative filtering produced different recommendation results, especially when compared to our previous models, the Gaussian Mixture Model and the K-Means clustering. The variation is significant. Due to their clustering characteristics, the Gaussian Mixture Model and K-Means may overlap, but item-based collaborative filtering bases its operation on the idea that items are similar to one another based on user interactions. The variations in the findings highlight the complex dynamics of suggestion. Even with the same input, the result might change depending on the underlying logic of the algorithm, reflecting the complex tapestry of human preferences and behaviors.

5. Conclusion

We have used a variety of algorithms and approaches throughout our study, from the linear complexities of K-Means clustering to the flexible dynamics of the Gaussian Mixture Model, as well as to the interaction between users and items in item-based collaborative filtering. Each model offered a different perspective from which to analyze user-game interactions. While the Gaussian Mixture Model and K-Means produced distinct user behavior and preference groupings, their recommendations surprisingly converged. Their clustering underpinnings can be credited for this overlap, highlighting how much results are influenced by the distribution and shape of data points.

On the other hand, item-based collaborative filtering, which was founded on the idea of item similarity based on user interactions, established this path. The differences between its suggestions and those of the other two models provide a convincing illustration of the complexity of recommendation algorithms. Collaborative filtering uses prior user-item interactions to create a dynamic portrait of user preferences in contrast to clustering approaches, which group data based on fundamental patterns.

This study emphasizes the complicated and related recommendation systems are, as well as the unique characteristics that each model offers. While some models may share insights due to their similar fundamental concepts, others, which are based on separate principles, might provide unique results. The main lesson is that no one model provides a comprehensive perspective. Instead, it fills in a unique gap in the picture by emphasizing a particular aspect of user behavior and preferences.

Combining insights from many models may be advantageous for practical applications, resulting in a thorough and well-rounded recommendation system. Such a hybrid strategy can maximize the benefits of each paradigm while minimizing its flaws. The relevance and quality of suggestions must be continuously maintained as datasets expand and change in the future. This will eventually improve user experiences and engagement on platforms like Steam.