# Steam Store Recommendation System using Alternating Least Square (ALS)

GROUP 5, CANDIDATE NUMBERS: 44568, 52206, 49640

Additional Key Words and Phrases: Steam Game Store, Recommendation System, Alternative Least Squares, Distributed Computing.

## 1 ABSTRACT

Recommendation systems are a critical feature for businesses that deal with large amounts of data, as it assists users in finding products, services, or information tailored to their specific needs and preferences. Steam is a cross-platform online game distribution service hosting thousands of games for millions of users worldwide. This makes it an ideal platform for implementing a recommendation system. In the context of Steam, a recommendation system can help users uncover new games, enhance their engagement and satisfaction, and ultimately boost revenue growth for the business. Given the growing number of platforms and video games, several recommendation systems have been developed. However, these models may be limited by their scalability. Therefore, we report an Alternating Least Squares (ALS) algorithm that leverages latent features to predict the games users may want to play, based on game characteristics and their gameplay history. Our approach takes advantage of the ALS algorithm's ability to run parallel on multiple machines, and to capture latent factors to uncover hidden preferences and patterns for more accurate recommendations. The final ALS model with 10 latent variables and 10 iterations accurately predicted user preferences, with an average difference of 0.10 units between predicted and actual ratings. Consequently, our model may serve as a basis for future recommendation systems of video games based on distributed computing.

## 2 INTRODUCTION

Recommendations generated from machine-learning algorithms are a key component to the e-commerce and digital sales industry. Observing the patterns from how users interact with products is advantageous to both the customer and vendor. Personalisation of the preferences allows for a better experience with the store and provides better satisfaction with customers. This thus leads to increased sales due to the ability to recommend products users are more likely to purchase. Alongside an overall better customer experience, the system models the preferences of users and provides valuable insights to the vendor about certain products that may hold higher significance. It can also highlight groups of individuals that may hold similar preferences and signify a market niche.

Fundamentally, recommender systems (RS) are information processing systems requiring three main data types: the information about the items, users and transactions. Transactions refer to a relationship between a user and the item generated during the human-computer interaction. In RS, most commonly used transaction data is the item ratings provided by the user [19]. The main objective of RS is to predict that an item is worth recommending for the user. However, the recommendation approach for each RS can vary depending on the addressed domain, the knowledge used and the recommendation algorithm. Thus, Burke (2007) has classified RS techniques into six different classes [4]:

- **Content-based**: The system uses data on items the users have liked in the past to recommend similar items.
- **Collaborative filtering**: The system recommends the active user items that users with similar characteristics have liked in the past. Thus, it is referred as 'people-to-people correlation'.
- **Demographic**: The demographic profile of the user is used for recommending the item.
- **Knowledge-based**: This utilizes domain-specific knowledge to suggest items that match users' preferences and needs.
- **Community-based**: Items are suggested to users based on the preferences of their friends.
- **Hybrid recommender systems**: These RSs are created by combining the techniques of the other five RS techniques.

## 3 RELATED WORK

### 3.1 Recommendation Systems for Steam Video Games

Several papers have been published to develop recommendation algorithms for Steam video games. Notably, video game recommendation algorithms have incorporated both content-based and collaborative filtering techniques. For example, Kim et al. (2016) utilized collaborative filtering to provide recommendations based on the preferences of other users with similar tastes, as well as content-based filtering to provide recommendations based on the preferences of the individual [13].

Furthermore, implicit feedback has also been incorporated into recommendation algorithms with positive results. Using implicit ratings based on hours of video game play and social networks, Perez-Marcos et al. (2020) have developed a hybrid system for video game recommendation [18]. Moreover, Li and Zhang (2020) developed a recommendation system based on network analysis of user-generated tags [15]. As a result, it provides an alternative method for understanding the genre of video games by relying on players' comments. In addition, recommendation systems have also considered bundles of video games, since these products may be offered at a reduced price to consumers in an effort to increase revenue.

A number of complex models have been developed for recommending video games. Cheuque et al. (2019) compared three algorithms, Factorization Machines (FM), deep neural networks (DeepNN) and a mixture of both (DeepFM), to create recommendations for Steam data [5]. The results demonstrated that DeepNN was the optimal algorithm for this recommendation task. Accordingly, they found that sentiment extracted directly from game reviews is not as relevant to recommendation as would be expected.

Despite the many advantages of using a neural network as a recommendation system, there are also some reasons why an ALS

algorithm may be preferable. Due to the low computational complexity of ALS, it may be possible for ALS to manage large-scale datasets more efficiently and with fewer resources than deep neural networks. The reason for this is that ALS is capable of coping with sparse data effectively, and its iterative optimization algorithm boasts a relatively fast convergence speed. In contrast, neural networks may require more complex architectures to handle sparse data effectively, and often require more iterations and rely on computationally intensive back-propagation to update weights. Lastly, ALS generally requires less memory than deep neural networks, since it does not require the storage of intermediate activations and weight gradients during training. Further details on the ALS recommendation algorithm are provided below.

### 3.2 Alternating Least Squares

Our study focuses on collaborative filtering using Alternating Least Squares (ALS) matrix factorization. Collaborative filtering is one of the most popular and widely used techniques in RS. It clusters similar users based on their past rating history, assuming that users with similar preferences will have similar preferences in the future[2, 13]. One of the most widely used approaches for collaborative filtering is ALS matrix factorization. Which decomposes the large user-item rating matrix into two lower dimensional matrices: a user matrix and an item matrix. The latent factors in these matrices represent the underlying characteristics of users and items. The higher the number of latent factors, the better the personalization will become until the number becomes too large leading to over-fitting issue. The dot product of the corresponding rows in these matrices represents the predicted user rating (i.e. preference) for a user-item pair. Matrix factorization aids developing RS as the model learns to factorise rating matrix into user and item representations, allowing the model to predict personalised item ratings for users. It also allows less-known items (i.e. less transactions recorded) to have as rich latent representations as popular items, thus, eliminating popularity bias in the RS [14].

ALS is deemed superior compared to the nearest neighbour techniques as it allows the incorporation of additional information such as temporal effects, confidence levels and implicit feedback [16]. Implicit feedback refers to when the RS itself can infer user preferences by observing user behaviour such as browsing history or search patterns. Since it usually denotes the presence/absence of an event, it is represented by a densely filled matrix as oppose to explicit feedback or the direct rating users have given to an object which is a sparse matrix since a user is likely to rate only a handful of items [14]. However, this comes at a computational cost, which has led to development of various models that mitigate this issue. For example, ImplicitALS by Hu et al. (2008) which assign confidence uniform or popularity-based confidence weights [14], and RankALS, a ranked based ALS model by Takacs and Tikk (2012) which simplifies the objective function's derivative to reduce computational time [20]. Another approach proposed by He et al (2017) is to use an item popularity-aware weighting scheme on the missing data to tailor the matrix factorization model for learning from implicit feedback [11].

ALS is an ideal matrix factorization algorithm to run on distributed computing due to its good scalability and its parallelizing training routine. ALS aims to reduce the computational time of the convergence of the quadratic loss function through the surrogate of two convex optimization problems (i.e. parallelizing); When the user matrix are fixed, it runs a gradient descent with matrix and obtain the solutions for the items in a closed-form before holding the item matrix fixed and run the same for the user matrix [1]. ALS has good scalability as it is able to run its gradient descent in parallel across multiple partitions of the training data from a cluster of machines [11].

Several studies have investigated the effectiveness of ALS in collaborative filtering. For instance, Zhou et al. (2008) proposed ALS-WR, a weighted ALS algorithm that assigns higher weights to users and items with more ratings [23], and demonstrated its superiority over other matrix factorization methods when applied to the Netflix dataset with over 1000 hidden features [23]. Through parallelization, ALS-WR's runtime is exceptionally fast when compared to models deploying k-Nearest Neighbour algorithms for the same task. Additionally, Hu et al. (2008) suggested transforming implicit user observations into two paired magnitudes: preferences and confidence levels [12] while Wang et al (2022) proposed an interest-value based ALS model which considers both the user's degree of interest in an item and the item's inherent value, which can value depending on the context [21].

Our study is based on a research paper by Hu et al (2008) which is considered the fundamental of ALS models [12]. Due to the lack of accurate review score for each game in the dataset, we will rely on implicit feedback to guide our recommender system. We aim to apply Hu et al's ALS model on the dataset from Steam store in order to provide users game recommendations based on the latent factors such as previously purchased games' genre, developer, publisher, hours played, platforms, and categories while also incorporating weighted review scores into the model.

## 4 METHODOLOGY

### 4.1 Data

No publicly available datasets were available that included both customers and products for Steam games. Thus, our analysis combined video game characteristic data from the Steam store data and SteamSpy dataset with the Steam user data.

Data cleaning for video characteristic data was based on the code and results from Nick Davis [6–9]. First, columns with more than 50% missing data were removed from the Steam store dataset, and then missing data was examined for each column. In addition, we cleaned the Steam Spy dataset (steamspy_data.csv) for missing data. Finally, the two datasets were merged to analyse the features of the video game. The final video game characteristic data contained the following information: name, ID, required age, platform, category, genre, release date, price, English, developer, and publisher, number of positive and negative reviews, the number of owners, the median play time, the price, and the tags associated with the game. Ultimately, there were 47,296 unique video games in our video game dataset, spanning 29,902 developers, 25,807 publishers, 7 platforms, and 4,448 categories.

The Steam username was reviewed for missing data. The final Steam user dataset included the following information: User ID, game title, purchase/play and hours played. In total, there were 12,393 users and 5,155 video games played.

## 4.2 Distributed Computing

For the purpose of parsing large user-product interaction graphs along with separate sources of data, we developed a bespoke cluster setups for both exploratory data analysis and the machine learning pipeline separately. The specifications between clusters did not largely differ aside from a storage space increase on the EDA branch. This would allow for better utilization of graph data-structures as these can sometime be memory intensive with different edge and vector components being stores. Due to this functionality, the `NetworkX` and `GraphFrames` libraries would be installed to visualise relationships better between users and products more efficiently. Alongside the storage increase, the same number of worker nodes would be utilized for both clusters as to not surpass the credit limit on the project.

Figure 1 shows the workflow of the EDA process and the respective machine learning pipeline. The usage of Pyspark and Apache services here was beneficial to the project. The authors leverage the processing power in the data cleaning process to perform a multitude of transformations to the initial RDD data. This was required due to the unkempt nature of the steam api data and many irregularities within the formatting. The cleaned data was then used to provide the many visualisations we observe within the document which furthermore help us understand the nature of the users and the games. The implementation of motifs, `GraphFrames` inbuilt query system is utilized to visualise the interactions between users and games in our data analysis. This allows us to derive valuable insights from the dataset, which are presented through simple statistics and plots in the Numerical Results section.

the optimized distributed computing capabilities of Pyspark was also used to improve the scalability of the project. The user information and game dataset is a fixed set of tuples, but with 11,000 new games being added to Steam each year, we can take advantage of Pyspark's datastream objects to build an algorithm that can handle the constant influx of new data. By doing so, we can ensure that our system is scalable and can accommodate the growing volume of data without sacrificing performance.

Utilisation of Pysparks inbuilt alternating least squares functionality as a collaborative filtering algorithm also leverages the computing power of the cluster in a distributed setting. The method is described within further in the following sections, but to summarize the improvements:

- **Parallelizing:** The user-item matrix is divided into smaller blocks and distributed across the machines in the cluster. This results in quicker computation in comparison to one machine processing the data as a large matrix of users-products details.
- **Fault Tolerance:** The algorithm can finish even if one machine fails. This is done by detecting when the machine fails and assigning the same task to a new machine to process the instructions again. [1]

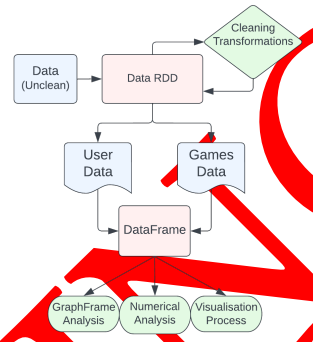


Fig. 1. EDA cluster workflow,

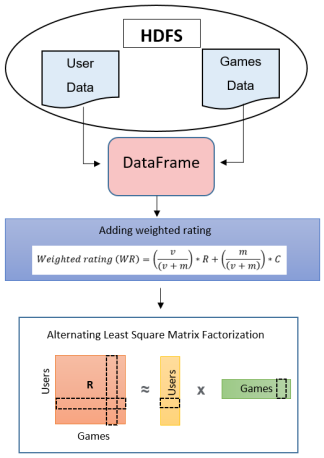


Fig. 2. Machine Learning Pipeline

## 4.3 Alternating Least Squares

In this study, we conducted our experiment on Google Cloud Dataproc cluster using 4 virtual CPUs and 15 GB of memory with Intel Broadwell E5 processor. The dataset consisted of two sets of data: a cleaned dataset containing information on all games available on the Steam store and a dataset containing information of the games owned by each Steam user, including the number of hours played on each game. The data attributes included User ID, Game ID, Game Title, Publisher, Developer, Platforms, Categories, Genre, Number of hours played, and weighted rating calculated from the number of positive and negative reviews each game received. The weighted rating was used due to the lack of actual numeric user ratings for each game. We employed IMDb's Bayes estimator to calculate and

[1] This is why the RDD/Dataframes use *lazy evaluation*, as if a machine fails, the task is assigned and the process starts from the beginning.

compare the ratings of the games [10]. The formula is as follows:

$$\text{Weighted Rating (WR)} = \frac{v}{v+m} \times R + \frac{m}{v+m} \times C \tag{1}$$

where R is the ratio of positive reviews to total reviews, v is the number of total reviews for the game, m is the minimum number of reviews required to be listed in the leaderboard which we set at 1000, and C represents the mean ratio of positive reviews to total reviews across all games.

In the ALS model, the matrix is factorized into smaller, lower-dimensional matrices which are used to represent the latent characteristics of users and items. The user factor vector ($\mathbf{u}_u$) represents the latent characteristics or preferences of the user. The elements in the vector correspond to latent features, and their values indicate the user's preference for each feature. Item factor vectors ($\mathbf{v}_i$) represent latent characteristics or features of the games. Each element of the vector represents a specific latent feature, and its value represents the extent to which that feature is present in the games. The ALS is designed to find the optimal user and item factor matrices (U and V) that minimize the cost function:

$$L = \sum_{u,i}(r_{ui} - \mathbf{u}_u^T \mathbf{v}_i)^2 + \lambda(\sum_u ||\mathbf{u}_u||^2 + \sum_i ||\mathbf{v}_i||^2) \tag{2}$$

where $r_{ui}$ is the user-item interaction matrix, $\mathbf{u}_u$ is the user factor vector for user $u$, $\mathbf{v}_i$ is the item factor vector for game $i$, and $\lambda$ is a regularization parameter.

The first term of the cost function, $\sum_{u,i}(r_{ui} - \mathbf{u}_u^T \mathbf{v}_i)^2$, calculates the squared difference between the actual and predicted weighted ratings for all interactions between users and items. ALS attempts to minimize this term by iteratively updating the items and users factors until convergence is achieved.

By penalizing large user factor and item factor vector values, the second term of the cost function, $\lambda(\sum_u ||\mathbf{u}_u||^2 + \sum_i ||\mathbf{v}_i||^2)$, prevents overfitting. $\lambda$ controls the trade-off between fitting the observed data and regularizing the complexity of the model.

To evaluate the performance of our ALS model, the dataset was split at 70:30 ratio at random with 2 partitions. We experimented with different hyperparameter settings; specifically varying MaxIter and LatentFactor between 1 to 20 while the RegParam was fixed at 0.1. We then conducted a grid search to identify the best hyperparameters for our model.

# 5 NUMERICAL RESULTS

## 5.1 Exploratory Data Analysis

*5.1.1 Data Summary.* The exploratory data analysis was, in part, based on the work of Nick Davis [6–9]. Figure 3 illustrates the video game characteristics according to whether the video game is paid or free. Notably, the exploratory analysis scatter graph includes the IMDb formula for weighted ratings, since different games have different numbers of players, therefore positive and negative comments will vary. In addition, there the number of total ratings and owners in video games were substantially skewed, and therefore logarithmic scales were applied to these variables for visualisation. Accordingly, as shown in the plot of weighted rating versus total rating, weighted ratings converge for video games with fewer total ratings, whereas weighted ratings vary more as total ratings increase. Finally, we observe that most video games were released after 2014. This is especially true for video games with fewer than 20,000 players.

We observe that as the number of owners and price increases, the log of total ratings and weighted ratings of video games increases. According to the distribution of prices and ratings across genres, the results do not appear to be confounded by genre. Based on the correlation plots, we can better understand the relationship between variables and weighted rating. There is a positive correlation between the log of owners and price with weighted ratings, with a correlation coefficient of 0.38 and 0.16, respectively. On the other hand, we observe a negative correlation between the release year and the logarithm of owners and ratings, with a correlation coefficient of -0.40.

*5.1.2 GraphFrame.* In an ideal situation, the recommendation algorithm would provide a game recommendation to users that would be able to maximize their enjoyment by matching similar games from other user groups with similar tastes, or games that have similar characteristics to games they are personally familiar with. From this understanding we can observe that certain Popular games could appear more significant if their average playtime is adjusted to account for their larger player counts, potentially overshadowing smaller franchises. If a large majority of users play a specific game for only 1 hour, the game might not be considered good due to the low playtime, but it might still be recommended based on other factors. By inspecting the most "popular" games in the methods devised further in this section, we can observe how well the recommendation algorithm works by analyzing the recommendations provided to the test set. Appendix figure 12 displays the playtime and players of the most popular games on the platform. One interesting discovery is that the games with the highest playtime do not have the highest players. One can observe that the initial price on the game dataset for many of the games with the most players appeared to be close to $0 USD. Free games are popular and a consideration in terms of the market appeal.

Inspecting two of the most popular titles on both lists of games, we can observe what the average behaviour of players would be when compared to the population within the same game. The average play-times, as shown in figure 13 is fairly high by around 40 hours for the selected popular games. The variation of these values is quite large with the "spread" of playtime following a somewhat exponential distribution. This could allow for possible recommendation by providing games with the highest likely average playtime in connection to other factors. This distribution conveys most people tend to play games for shorter periods of time in comparison to the players who have upwards of 10,000 hours. This could indicate playtime may not be a good attribute for recommending games as the majority of players purchase and play games for the minimal amount of time.

The representation of the relationship between games and users would allow for better understanding of how groups form in terms of preferences of genre, publisher, etc. By formulating a Graph network and calculating the significant nodes using a degree centrality

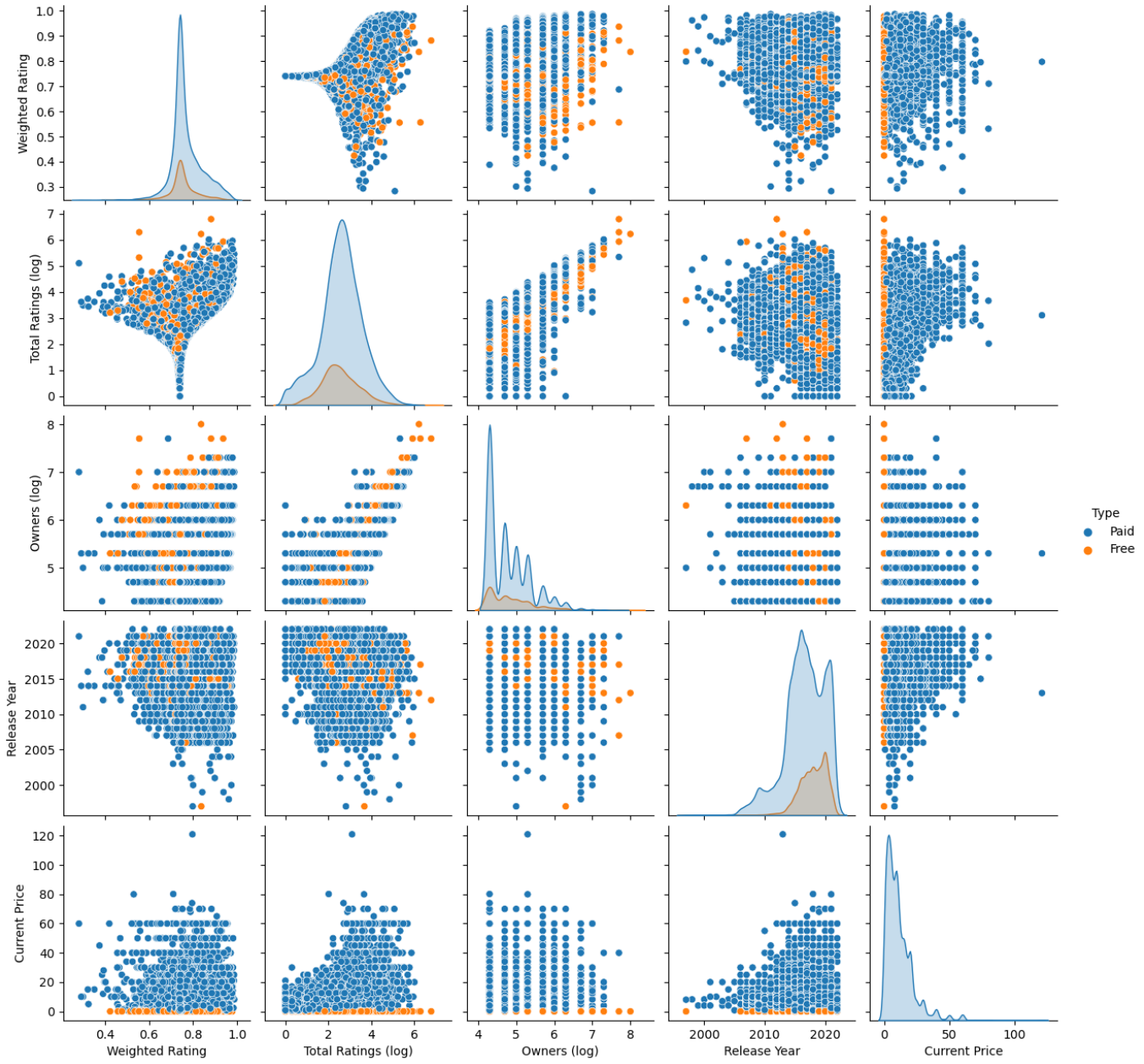


Fig. 3. Scatterplot of videogames characteristics

algorithm such as one shown in relevant literature [17], we are able to observe which nodes and specifically games are significant. These differ from the observed popular titles shown in figures 12 and 13, with Dota 2 and Left 4 Dead 2 still maintaining 1st and 2nd place respectively.

To understand the non-euclidean relationship between players and games, one needs to model and store the user relation data in an intuitive manner. Players relate to publisher Bethesda and its two most popular games in an interesting and potentially exploitable manner, as shown in appendix figure 11. Upon further inspection, one can observe that players who enjoy specific aspects of one game from a publisher also enjoy other games from the same publisher, even if the genres and game tags do not overlap. Leveraging this insight, we hypothesise that better recommendations could be provided by incorporating more attributes from the game description dataset.

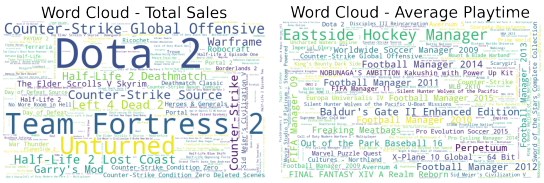


Fig. 4. Word Cloud for Games with the Highest Total Sales and Average Playtime

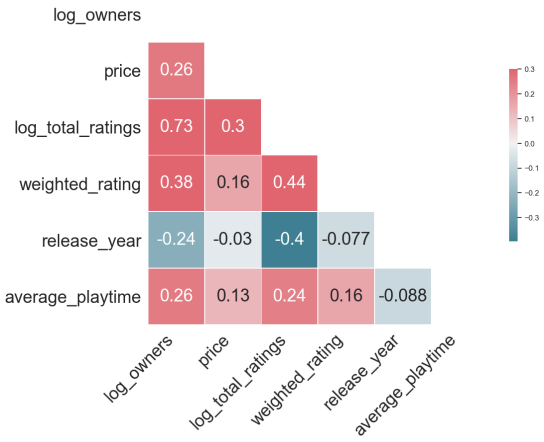


Fig. 5. Correlation Plot of Various Characteristics of Videogames

## 5.2 Alternating Least Square Model

*5.2.1 Root Mean Square Error Values vs Different MaxIter and LatentFactor.* The root mean square error (RMSE) values with MaxIter and LatentFactor, ranging from 1 to 20, are displayed in Figure 6 and Figure 7. Notably, from 9 max iterations onwards, the RMSE converged at roughly 0.1 irrespective of the number of latent factors with the exception of 1 latent factors. Increasing MaxIter value results in more iterations, leading to a more accurate prediction of ratings. Similarly, increasing the latent number factor increases the user and item features represented in the model, thus, increasing its accuracy.

*5.2.2 Computational Time.* Figure 8 and Figure 9 displayed the mean computational time for each MaxIter and LatentFactor number respectively. In Figure 8, for each MaxIteration, we made several runs with different numbers of latent factors ranging between 1 to 20 and took the average computational time. As expected, the higher the number of MaxIter, the longer is the mean computational time. This can be simply explained by the function of the MaxIter hyperparameter. MaxIter controls the number of times the ALS algorithm iterates through the process. Thus, if the MaxIter value is increased, the algorithm performs more iterations increasing computational time.

Similarly, in Figure 9, there is a general positive relationship between the latent number factor and the mean computation time. The latent number factor hyperparameter controls the number of latent

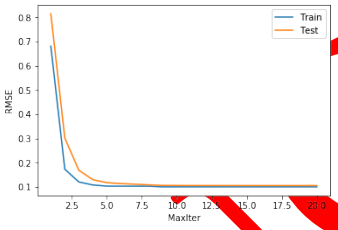


Fig. 6. RMSE vs MaxIter

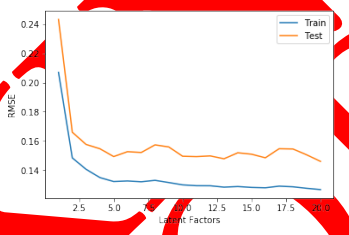


Fig. 7. RMSE vs Latent Factor Number

factors used to represent the user and item features. By increasing the latent number factor, the user and item matrices' size will increase. Bigger matrix will lead to an increase computational time as a trade-off for higher accuracy of the ALS algorithm as previously noted by Takacs and Tikk (2012) [20].

*5.2.3 Grid Search: The Best Hyperparameters.* We conducted a grid search to identify the optimal hyperparameters for the ALS algorithm based on RMSE. We performed grid search over the MaxIter and LatentNumber of 5,6,10,15 and regParam over 0.001, 0.01, and 0.1. The grid search revealed that the best ALS model specification should have both MaxIter and LatentNumber set at 10 and regParam at 0.1. At 10 iterations with 10 latent factors and regParam of 0.1, the RMSE of the test data is 0.1057.

These optimal hyperparameter values suggests that increasing the number of iterations over 10 does not lead to significant improvements in accuracy. Likewise, increasing the number of latent number factors over 10 might lead to over-fitting, reducing the model's ability to generalise.

The optimal regParam value of 0.1 suggests that the regularisation strength of the model is relatively low, allowing it to capture more complex relationships in the data without over-fitting.

## 6 CONCLUSION

The described process of the recommendation algorithm in this paper encapsulates the main concepts of distributed computing. Utilising a series of interconnected machines to solve complex problems with more computational efficiency.

The pipeline and clusters in our study were designed for fault resilience during development by implementing frequent data checkpointing, ensuring reliable handling of the data. The transformation and visualisations performed allowed for improved comprehension of the data from an analytical perspective, understanding how users play games and if they have relationships between genres and publishers in their preferences. The final model was built upon the

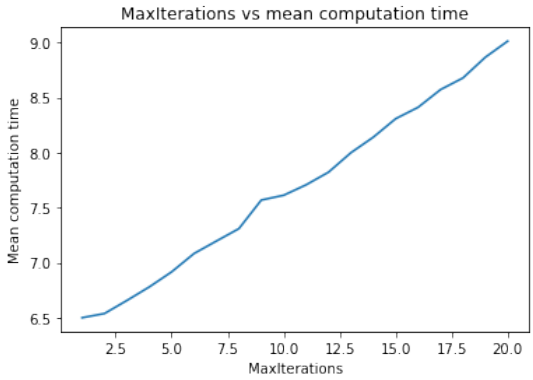


Fig. 8. Computation time of different MaxIters

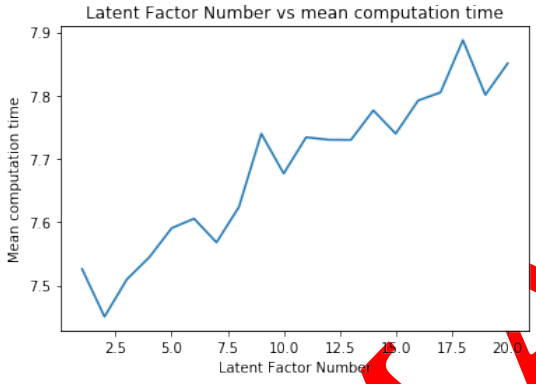


Fig. 9. Computation time of different Latent Factor Numbers

understanding that certain attributes between games had some distinction on users preferences towards them. People were more likely to play games that were similar to games they already had a liking to. The abundance of this user-product interaction data provided the perfect landscape to implement a collaborative algorithm where we used the preferences of matching users to recommend games to users with similar behaviour. The best performing model in this paper was the ALS (Alternating Least Squares) model where we implemented and benchmarked the recommendations provided. The numerical results provided show how this process was successful in providing novel and unique recommendations that were not swayed by larger more distinct groups of users with a difference in preference. The recommendations generated are intuitively related to each other, whether by publisher or by genre. The recommendation algorithm also can recommend games where the user-product interaction count is quite low, allowing for more niche and low-budget games to be provided to users. The recommendation system the authors implemented in this project is comparably better optimized and suitable for smaller data-sets where less user-product interaction data is available. The computation time and speed is faster, alongside improved interpretability as stated in the relevant literature [3]. "Alternating least squares (ALS) is notable for its speed and scalability" (Burke, 2002).

One downside to the approach taken with the ML pipe-lining was the absence of a data-stream process being implemented. Information could be directly provided from the `SteamAPI` and cleaned in real-time to provide up-to date recommendations from new games. The original static data-source is from the steam-api but only during a specific time period. The ALS algorithm is capable of providing recommendations to new users with no historical information *(cold start approach)* meaning this could be a plausible implementation for a "live" industry example where the user information is streamed in, new games are streamed in and recommendations are continuously provided. The layout of the user-product data could have been exploited in the `GraphFrame` portion of the analysis. By expanding the classes of vertices to include the category of games, the tags and other attributes, more advanced motifs could be constructed to find patterns between players not observed from the visuals in this paper. There also exists, many Graph-based recommender systems [22] for data structured similar to ours, where the user-product matrix describes an adjacency matrix. Implementing other models and comparing the results could convey the usefulness in the ALS model in the present context more successfully.

The findings of this study are important for the gaming industry as it provides a method to develop better understanding of relationships and patterns between gamers and their games, improving the quality of the store's sale strategy. Overall the project contributes to the field by demonstrating the effectiveness of the novel and interpretable recommendations provided by the ALS model, in the context of steam games and users.

## REFERENCES


[1] Andreas Bloch. An overview of collaborative filtering algorithms for implicit feedback data, 2019.

[2] J. Bobadilla, F. Serradilla, and A. Hernando. Collaborative filtering adapted to recommender systems of e-learning. *Knowledge-Based Systems*, 22(4):261–265, 2009.

[3] Robin Burke. A comparative study of collaborative filtering algorithms. *AI communications*, 15(2):99–111, 2002.

[4] Robin Burke. *Hybrid Web Recommender Systems*, pages 377–408. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[5] Germán Cheuque, José Guzmán, and Denis Parra. Recommender Systems for Online Video Game Platforms: The Case of STEAM. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, pages 763–771, New York, NY, USA, 2019. Association for Computing Machinery. event-place: San Francisco, USA.

[6] Nick Davis. Gathering Data from the Steam Store API using Python, May 2019.

[7] Nick Davis. Steam Data Cleaning in Python, June 2019.

[8] Nick Davis. Steam Data Exploration in Python, August 2019.

[9] Nick Davis. SteamSpy Data Cleaning in Python, July 2019.

[10] fxSOLVER. Bayes estimator - internet movie database (imdb).

[11] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback, 2016.

[12] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272.

[13] Soo-Cheol Kim, Kyoung-Jun Sung, Chan-Soo Park, and Sung Kwon Kim. Improvement of collaborative filtering using rating normalization. *Multimedia Tools and Applications*, 75(9):4957–4968, 2016.

[14] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[15] Xiaozhou Li and Boyang Zhang. A Preliminary Network Analysis on Steam Game Tags: Another Way of Understanding Game Genres. In *Proceedings of the 23rd International Conference on Academic Mindtrek*, AcademicMindtrek '20, pages 65–73, New York, NY, USA, 2020. Association for Computing Machinery. event-place: Tampere, Finland.

[16] Kevin Liao. Prototyping a recommender system step by step part 2: Alternating least square (als) matrix factorization in collaborative filtering, 2018.

[17] Mark Newman and Albert-László Barabási. *The Structure and Function of Complex Networks*. Princeton University Press, 2003.

[18] Javier Pérez-Marcos, Lucía Martín-Gómez, Diego M. Jiménez-Bravo, Vivian F. López, and María N. Moreno-García. Hybrid system for video game recommendation based on implicit ratings and social networks. *Journal of Ambient Intelligence and Humanized Computing*, 11(11):4525–4535, November 2020.

[19] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer-Verlag, Berlin, 2010.

[20] Gábor Takács and Domonkos Tikk. Alternating least squares for personalized ranking, 2012.

[21] Zhaoyang Wang, Jianjun Huang, and Hongyuan Sun. Improved als recommendation algorithm based on interest-value model. *J. Phys.: Conf.* 2219, 2022.

[22] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional matrix completion. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2185–2194. ACM, 2018.

[23] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize, 2008.

# A APPENDIX

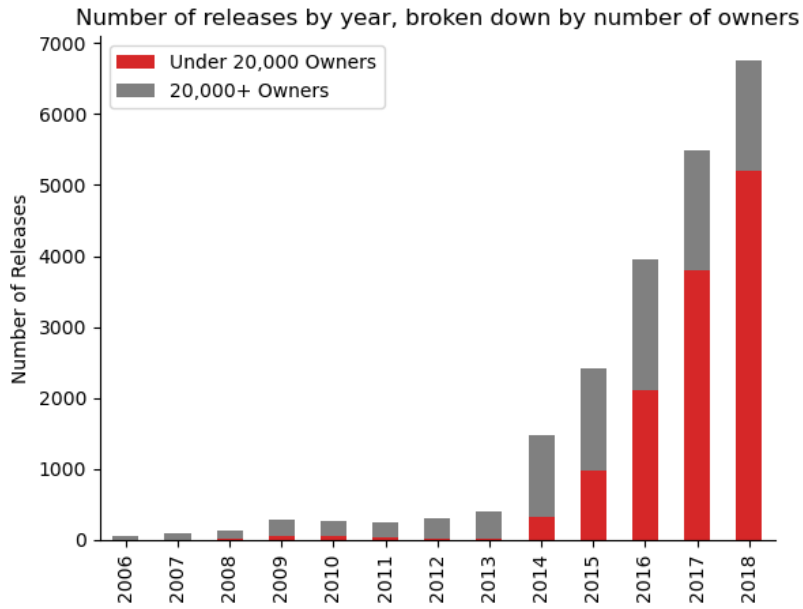


Fig. 10. Number of release by year

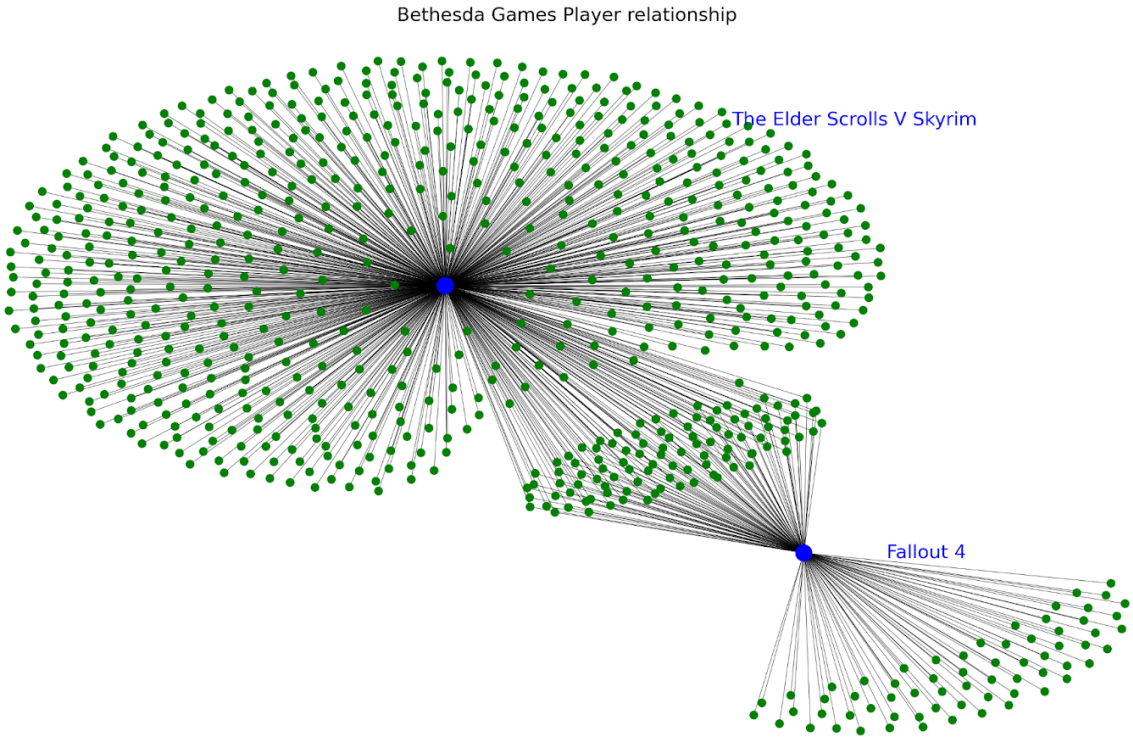


Fig. 11. Connection of players (**Green**) to Games (**Blue**)

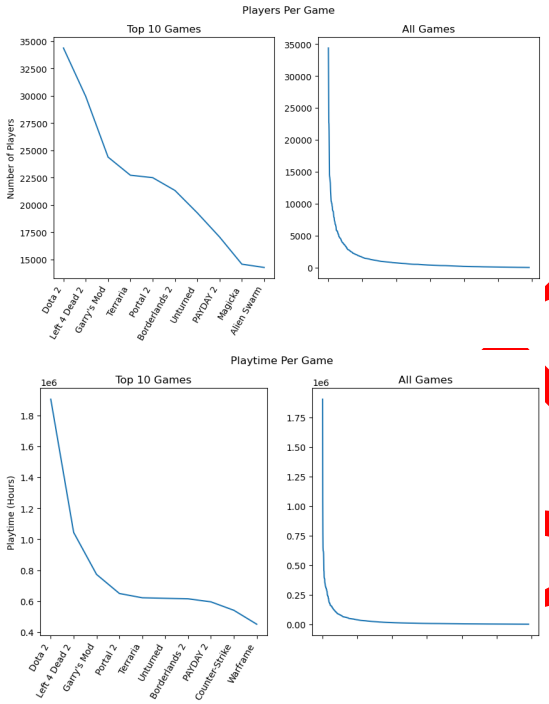


Fig. 12. Top 10 Games by Players (**Right**) and Playtime (**Left**)

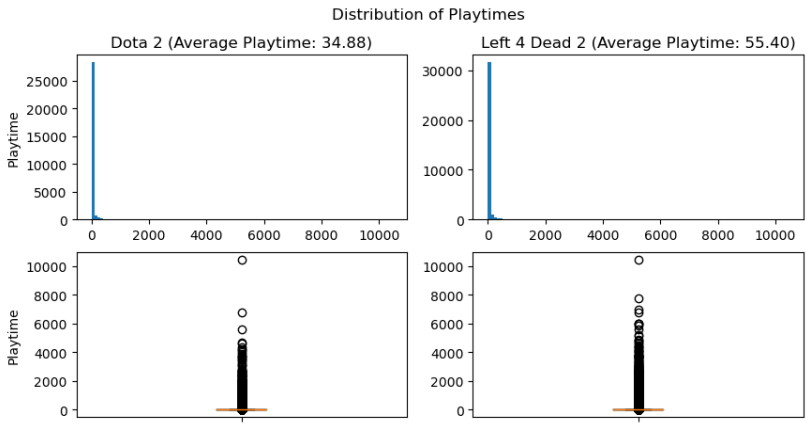


Fig. 13. The Distribution of Playtime for the Popular games **Dota 2**, **Left 4 Dead 2**

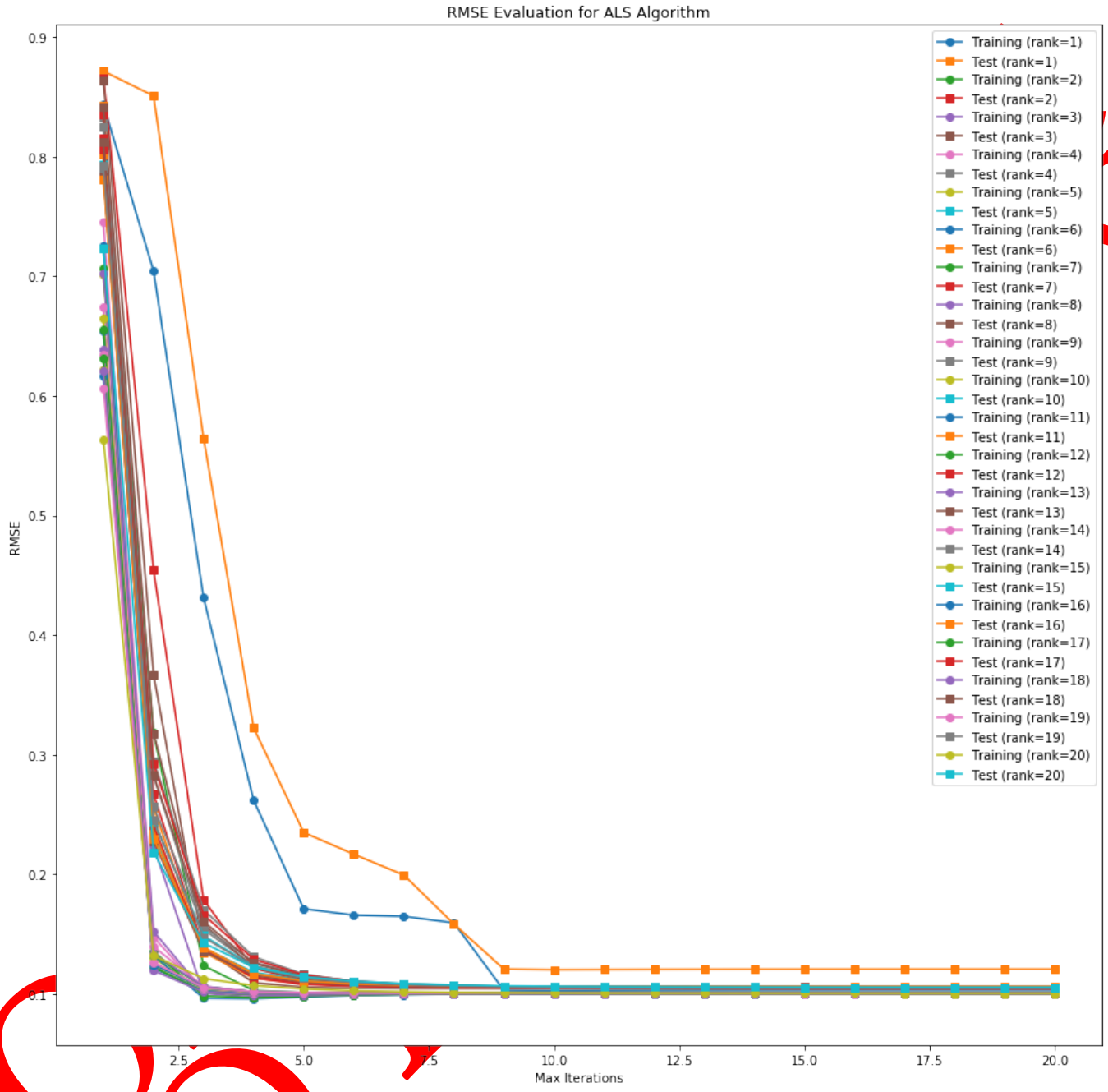


Fig. 14. RMSE on Different MaxIterations and Latent Number Factors