ISSN 2278-3091
Volume 8, No.1.6, 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse0581.62019.pdf
https://doi.org/10.30534/ijatcse/2019/0581.62019

# Strategies to Map Play Time in Steam Platform Dataset into 5-Star Ratings

**Shuria Saaidin**[1], **Zolidah Kasiran**[2]
[1]Fakulti Kejuruteraan Elektrik,Universiti Teknologi MARA, Malaysia, shuria6809@uitm.edu.my
[2]Fakulti Sains Komputer dan Matematik Universiti Teknologi MARA, Malaysia, zolidah@tmsk.uitm.edu.my

## ABSTRACT

This paper presents strategies to map play time in Steam dataset into 5-Star rating. Our objective is to check whether we could use the created explicit feedback in recommender system. In this study we calculated z-score for each users' duration of playtime and using the score we applied 3 strategies to map duration of playtime into explicit feedback. These strategies are Same Size Bin (SSB), Breakpoint Based Bin Sized (BBS) and Hybrid Based Bin Size (HBS). According to RSME value and Hit Rate, we found that the best strategies to map duration of playtime into explicit feedback is using HBS strategies. We checked the feasibility of using the created explicit feedback by feeding it to KNN algorithm. We compare RSME from others researches that used the same algorithm and found the RSME value that we get are in the same range as the researches. Based on this observation, it can be concluded that it is feasible to use created explicit feedback in Recommender System.

**Key words:** Recommender System, Implicit Feedback, Explicit Feedback, Steam Platform.

## 1. INTRODUCTION

Recommender system are developed to filter information for users. Using recommender system, users can make faster and accurate choices as only data relevant to them are presented. Recommender system make suggestion to users in many domains such as movies [1], music[2], travelling route [3] and e-books[4]. Lately there are also research in suggesting online video games to users[5].

Online video games are domains that generate a lot of contents for different users' background. Although majority of online video game are played as an entertainment (players most likely will like different genres compared to others ), there are video games that created for different groups of users including preschoolers[6], patients [7] and even sedentary people that trying to be active [8]. Given this disparity of users, recommender systems are needed to provide suggestions of the right games for the right users.

Steam is one of the largest platforms for games distribution. The Steam digital distribution service was started in 2003 and is owned and operated by the Valve Corporation, a video game developer [9]. To date steam are hosting over 8 thousand games and have around 553 million active users. If a user has played about 100 games, there are more than 7000 games are available for user to choose from which quite a lot for humans to go through the entire collection. Too many choices will make user overwhelmed and sometime will lead to user refuse to choose at all[10]. This situation is called for recommender system. Recommender system are used to help user with information overloaded problems that arise as a result from large amount of data are available on the web. Recommender system would act as filtering mechanism that will suggest content that relevant to users. An active user then, will be presented by several contents based on their past behavior or past behavior of the similar user to them. Fundamentally, recommender system will predict rating for items that have not been consume by users and recommend top-N highest ratings' items to users

Recommender system are usually based on rating provided by user for certain items. This is call explicit feedback as user explicitly telling the system their preferences. When ratings are not available, there are a lot of indicator that can be used to describe how user feel about certain items. In video games, duration of playtime can be used to estimate how much user like certain games.

It can be assumed that the longer the playtime the fonder user are of the games. We based our assumption on observation made in [11], which stated that that the more time a content is displayed by a user, the more he likes it and therefore the higher he rates it. However, duration of playtime varies too much for a computer system to be able to predict how long users will play certain games contrary to star rating that have limited number of class (5 stars, 10 stars etc.).

This research aims to normalize the high variation in playtime by converting it into 5-star rating where 1 star is the rating for shortest playtime and 5-star is rating for the longest playtime. We reduced biased of the rating, since players standard of play time duration are different. For instance, user 1 would consider 5 hours play time merit to 5-star rating but user 2 must play for more than 24 hours to give 5-star rating to the video games. To reduce biased in play time duration, z-score was calculated.

To test the feasibility of the rating produced using z – score, we applied user-based collaborative filtering algorithm and item-based collaborative filtering algorithm on the modified dataset to see if we could correctly predict ratings in the dataset. The approach of using z-score in recommender system are already applied in [5] however they didn't directly convert z-score into rating but used value of the z-score instead. The Instead of truncated the value of z-score to -3 to 3 like in [5], we used all the value we get from the calculation and deal with the value outside of the normal distribution in three different ways.

## 2. RESEARCH METHOD

### 2.1 Background

Recommender System usually begin with rating prediction. A lot of algorithms are introduced, refined and used to predicted rating. Recommender System are often classify into three different categories [12] namely:

- Content-based recommendations: Rating will be predicted based on similar item rated by the user in the past.
- Collaborative recommendations: Rating will be predicted by rated items by people with similar tastes and preferences liked in the past
- Hybrid approaches: These methods combine collaborative filtering and content-based methods.

.

Recommender System also evolve to include methods such as matrix factorization [13]–[15], deep learning [13], [16], [17]and genetic algorithm [18]. Recommender System researcher focused on several issues in predicting rating for users including but not limited to data sparsity, cold start problem, scalability problem, diversity, context-aware recommender system and multi-criteria recommender. There are also interest in utilizing all the information available on how users behave to certain items. This behavior could be captured explicitly or implicitly.

### 2.2 Explicit/Implicit feedback

Recommender system always based on rating provided by user for certain items that has already been consumed. For example in Movie Lens dataset [19], user will rate movies (1 to 5) based on their preferences with 1 is least preferred and 5 is most preferred. This kind of rating are considered explicit feedback since user explicitly rated movies according to their preferences. Similarly, social networks site such as Facebook and YouTube use Like/Dislike rating to show user preference on the content publish on their site. When explicit ratings are not available, there are a lot of indicators that can be used to describe how user feel about certain items.

Implicit rating means that user preferences of certain items are captured without user being aware [11]. Indicators such as user click on certain items and time user spend in reading articles online could be used to describe user behavior towards certain items. In [11], they concluded that the more user display certain content, the higher the rating of that content will be. So, there is a direct relation between displaying time and explicit feedback. Following this observation, we could assume that the longer the games is played by a player, the more likely he will give higher rate to the games.

There are two ways of using implicit feedback in recommender system. One way is to use algorithm specifically design to deal with implicit feedback and the other way is to map implicit feedback to explicit feedback then using explicit feedback-based algorithm to recommend items[20]. In this research we use the second approach. We calculated z-score for each user playtime and mapped into 5-star rating which resemble explicit feedback.

### 2.3 Long Tail Distribution.

In statistic, long tail in a dataset means is a probability distribution that has many occurrences that are far from the central part of the data distribution. As opposed to normal distribution that resemble a bell with most data are distributed at the center, in long tail distribution most data are distributed at the end of the distribution. Figure 1 shows how data in long tail distribution are distributed.

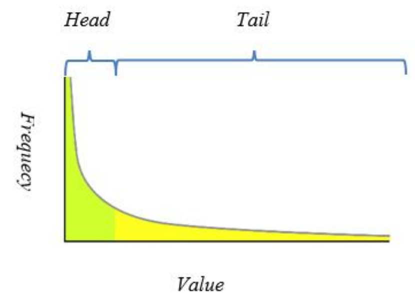

**Figure 1**: Long tail distribution

Long-tails complicate analysis because rare cases from the tail still collectively make up a significant portion of the data and so they cannot be ignored[21]. In recommender system, research on how to leverage long tail distribution in items consumption have become one of the topic discussed[22].

From recommendation system perspective, popular items will be in the head section consumed by most consumer and the less popular items will be at the tail section consumed by a smaller number of consumers. However, the popular items contribute only about 10% of the overall items making it is essential for recommender system to deal with the rest of the items that is at the tail section. In our case, head of the section are populated from smaller playtime, which means a lot of players did not spend too much time playing games, but there are also a few players that have a very long play time[9]. We need to be able to treat those players differently as players in head section have different standard in play time compared with players in the tail section.

## 2.4 Measurement Metrics

Recommender system performance could be measure using different type of metrics. Common methods used are:

### MAE

Mean Absolute Error measure how much errors occurs when predicting value for certain items. MAE in dataset could be calculated by subtracting true value with predicted value divided by number of records in a dataset.

$$\text{MAE} = \frac{|\sum(PR_i - AR_i)|}{N} \quad (1)$$

Where:

N = number of records in datasets
PR = Predicted rating
AR = Actual rating
*i* = index of record

### RSME

Root Mean Square Error (RMSE) measures **how much error there is between two data sets**. In other words, it compares a predicted value and an observed or known value. In our case we will compare error in predicting rating between train and test datasets. RSME can be calculated using (2)

$$RSME = \sqrt{\frac{\sum_{i=1}^{N}(PR_i - AR_i)^2}{N}} \quad (2)$$

Where:

N = number of records in datasets
PR = Predicted rating
AR = Actual rating
*i* = index of record

RSME determine how close are the predicted value to the actual value. The different between MAE and RSME is that in RSME, value predicted too far from the actual value will be penalized. In MAE errors are treated equally regardless how far off the prediction from the actual value.

### Hit Rate

Recommender system will provide users with a ranked list of N items they will likely be interested in, in order to encourage views and purchases. To measure how good a Top-N recommender system, hit rate are often used. For example, to evaluate top-10, we use hit rate, that is, if a user rated one of the top-10 we recommended, we consider it is a "hit". Figure 1 shows how hit rate could be calculated.

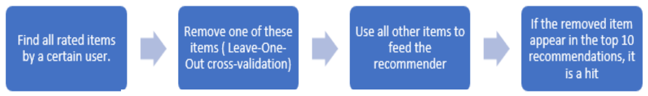
Find all rated items by a certain user.
Remove one of these items ( Leave-One-Out cross-validation)
Use all other items to feed the recommender
If the removed item appear in the top 10 recommendations, it is a hit

**Figure 2**: Steps to calculate hit rate

Using algorithms available to predict rating we could then present user with list of items with the highest predicted rating.
In this paper we will use these 3 matrices to measure recommender system that uses rating produced by using our strategies. MAE and RSME will measure the accuracy of rating prediction and hit rate will tell us whether we could suggest a relevant video to users

## 2.5 K-Nearest Neighbors Recommender System Algorithm

Recommender System has evolved to include various methods in order to predict precisely what rating users likely to give to certain items. Those methods include machine learning, deep learning, clustering and matrix-factorization. In this research we will describe and use collaborative filtering using KNN model. The reason is, this method has been tested and used in many papers hence it provides comparable result with the other research.

In content-based collaborative filtering, KNN model will calculate similarities of the target video games with all the other video games in the data set and rank them accordingly. Similarity between items could be calculated using different methods. The most common is using cosine similarity matrix. Similarity matrix between item, *i* and item, *j* is given by the equation:

$$sim(i,j) = \frac{\sum_{u\in U_{ij}} r_{ui} \cdot r_{uj}}{\sqrt{\sum_{u\in U_{ij}} {r^2}_{ui}} \cdot \sqrt{\sum_{u\in U_{ij}} {r^2}_{uj}}} \quad (4)$$

Where:

$U_{ij}$ is the set of users that have rated item *i* and *j*
$r_{ui}$ rating by user, *u* for item, *i*
$r_{uj}$ rating by user, *u* for item, *j*

KNN will return top-K video games in the rank as a recommendation. Item's, *i* predicted rating, $\hat{r}$ for active user, *u* is given by the equation:

$$\hat{r}_{ui=} \frac{\sum_{j\in N_u^k(i)} sim(i,j).\, r_{uj}}{\sum_{j\in N_u^k(j)} sim(i,j)} \quad (5)$$

Where:

$N_u^k(i)$ K-nearest neighbor items, *i* rated by user, *u*
*j* is set of $N_u^k(i)$
$sim(i,j)$ is similarity between item, *i* and item, *j*
$r_{uj}$ rating from user, *u* for item *j*

Similarly, in user-based collaborative filtering, similarity between user, *u* and user, *v* can be calculated using cosine similarity matrix using this formula:

$$sim(u,v) = \frac{\sum_{u \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{u \in U_{uv}} {r^2}_{ui}} \cdot \sqrt{\sum_{u \in U_{uv}} {r^2}_{vi}}} \tag{6}$$

Where:

$I_{uv}$ is the set of all items that rated by both user, *u* and user *v*
$r_{ui}$ rating by user, *u* for item, *i*
$r_{vi}$ rating by user, *v* for item, *i*

Predicted rating, $\hat{r}$ of item, *i* for user, *u* could be obtained using the formula:

$$\hat{r}_{ui} = \frac{\sum_{j \in N_i^k(u)} sim(u,v) . r_{vi}}{\sum_{j \in N_i^k(u)} sim(u,v)} \tag{7}$$

Where:

$N_i^k(u)$ K-nearest neighbors' user, *u* that have rated item, *i*
*j* is set of $N_i^k(u)$
$sim(u,v)$ is similarity between user, *u* and user, *v*
$R_{vi}$ rating from user, *v* for item *i*

Equation (4) to (7) are derived from surprise library for python developed by [23]. RSME and MAE will be used to measure performance in predicting the rating for each user using the explicit data created from duration of play time and hit rate will be used to measure whether we could predict relevant top N list based on the rating obtained in the above method.

## 2.6 Dataset and Experiment

Data used in this research are taken from dataset collected and analyzed in [9]. In the raw dataset there are around 109 million gamers, 7 million games and 1.1 million years of playtime. As this research are only aims to see the feasibility of converting playtime into explicit rating, the whole data dataset is not needed. Dataset was reduced to include only 10000 users that have the most playtime and only games that been played by the 10000 users are included which amounted to 1985 games.

We based our binning strategy in the assumption that the longer the play time the higher rating user will give to the game they played. Since play time varied and in a very long range: minimum play time is 61 minutes and maximum playtime is 578398 minutes with 9659 unique play time value, it is difficult to predict play time for each user.

Hence, we need mechanism to shorten the play time range and to classify each play time into fewer class. We use 5-star rating as indicator of user preferences towards games they played in order to overcome the variation of playtime value.

Figure 3 summarize how we conducted our experiment.

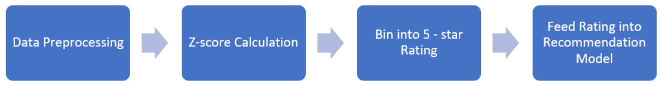

**Figure 3**: Experiment Steps

We calculate z-score to represent each playtime in the dataset to shorten the range in playtime range. It is also meant to remove bias exist between users. In statistics, z-score is the signed number of deviations from mean indicating that a datum is above the mean if positive and below the mean if negative. In theory, using z-score will eliminate user bias as z-score value will be derived from each user mean play time and standard deviation of each user play time. In our approach, we calculate the z – score for playtime, *t* of each game, *i* played by user, *u*. Z-score of a data, *z* can be calculated using this formula:

$$z = \frac{tu_m i_n - \mu}{\sigma} \tag{3}$$

Where:

*m* and *n* index for user and games respectively
$\mu$ is mean playtime of $u_m$
$\sigma$ is standard deviation of playtime for $u_m$.

Z-score is calculated and then plotted to see the distribution of the z-score across the whole dataset. We could see that z-score value for the datasets have long tail distribution. Long tail distribution in our dataset means that there are a few players that have very long playtime compared to majority of the players. This could affect the overall mapping of z-score to rating. For a z-score to be map as 5-star rating, its value must be over 5.744 which only applied to 72 /223 047(0.032 %) of the records. Most of the rating will be in the head part of the distribution which means most of the rating will indicate that user dislike video games that they have played. Due to the imbalance distribution of the rating among players, we implement 3 different strategies of binning z-score into 5-star class of rating.

To convert z-score value into ratings, we consider three strategies. The first strategy is to bin each z-score into 5 bins of the same size based on maximum and minimum value of z-score, without any special method to treat the long tail section which resembles approach in [5]. We will call this method as Same Size Bin (SSB) for the entire paper. Figure 4 visualize the mapping process.

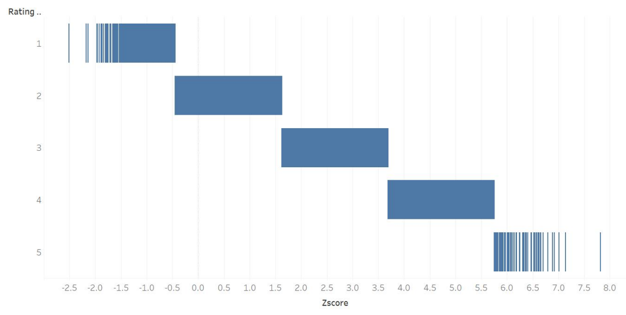

**Figure 4**: Z-score to rating using SSB

In the second strategies, we deal with long tail distribution by binning z-score into 5-star rating based on the classification method in [24]. In this strategy, head needs to be divided into breakpoints. A breakpoint is the last or first value in a class. To find the first breakpoint of the long tail data distribution the mean value of that data needs to be found. The selected mean value is then will to be the first breakpoint value. Then all values larger than that mean are selected. Out of these values the mean value of those is selected. That will be the second breakpoint. In [24] the selection of breakpoints continues in the same manner until there is only one maximum value left. Both the selection of breakpoints and the selection of number of classes are naturally developed, but in our case since we need to classify z-score into 5-star class rating, we stop calculating the mean when we found the fourth breakpoint which enable us to divide z-scores into 5 class. This method will be referred as Breakpoint Based Bin Size (BBS). Figure 5 visualize the mapping process.

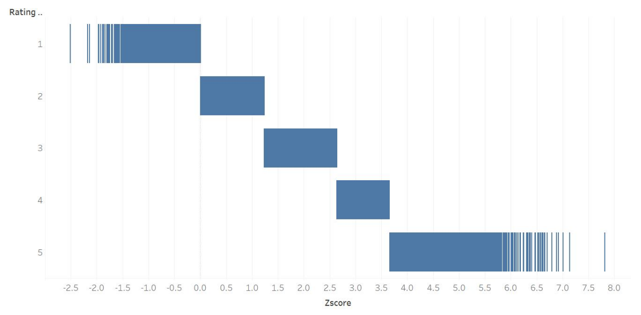

**Figure 5**: Z-score value to rating using BBS

The last method is to deal with the classification separately in the head and tail section. We use the fourth breakpoints in method 2 to separate datasets into 2 section: head and tail. In head section we divide the z-score value into 5 equal bins like in SSB where whole tail part of the dataset is mapped as 5-star rating. We called this method as Hybrid Bin Size (HBS). Figure 6 shows the mapping process.

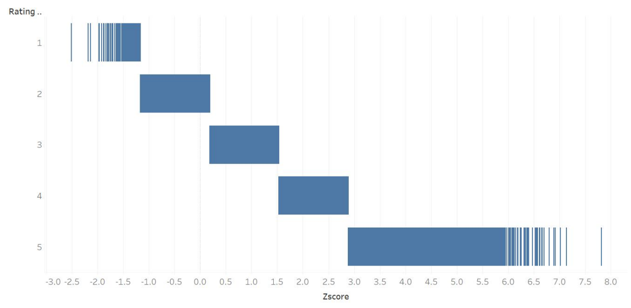

**Figure 6**: Z-score value to rating using HBS

To observe if rating obtained from z-score calculation and binned strategies are feasible to be used in recommender system, we feed the rating into item-based and user-based collaborative filtering using K-Nearest Neighbor (KNN) modelling. We used surprise library for python described in [23].

## 3. RESULT AND ANALYSIS

We calculated z- score to eliminate bias that occurs between players' playtime duration. If we compare between users, we could see different patent in their playtime. Overall, we could see that although playtime value is converted into z-score, rating still reflect that the longer the playtime, the better rating the games get. If we compare playtimes for each user, we could see that play time that is consider long enough to get 5- star rating may be differ from each user. For example, in Table 1, if we look at user 0, 7830 minutes is considered 5-star rating, but for user 3, 97255 minutes of playtime are only considered 3 rating. This is because we calculated z-score based on each user mean and standard deviation which confirm the statement of each user have different standard when rating certain items[25]. If we compare playtime and rating of each user like this, it can be concluded that we have successfully mapping playtime to their respective playtime duration pattern.

**Table 1**: Relation between playtime, z-score and rating of different users.

| User | PT | Z | Rating | User | PT | Z | Rating | User | PT | Z | Rating |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7830 | 7.81 | 5 | 1 | 11676 | 6.69 | 5 | 2 | 5775 | 5.81 | 5 |
| | 4606 | 4.36 | 4 | | 9738 | 5.51 | 4 | | 4581 | 4.47 | 4 |
| | 2653 | 2.27 | 3 | | 3136 | 1.47 | 2 | | 2482 | 2.11 | 3 |
| | 2548 | 2.16 | 3 | | 3008 | 1.39 | 2 | | 2463 | 2.09 | 3 |
| | 1701 | 1.25 | 2 | | 2854 | 1.29 | 2 | | 2133 | 1.72 | 3 |
| **User** | **PT** | **Z** | **Rating** | **User** | **PT** | **Z** | **Rating** | **User** | **PT** | **Z** | **Rating** |
| **3** | 131534 | 4.20 | 4 | 4 | 40540 | 7.13 | 5 | 5 | 1735 | 4.26 | 4 |
| | 97255 | 2.86 | 3 | | 26755 | 4.61 | 4 | | 1349 | 3.09 | 3 |
| | 93562 | 2.72 | 3 | | 4036 | 0.47 | 2 | | 987 | 1.98 | 3 |
| | 93162 | 2.70 | 3 | | 2575 | 0.20 | 2 | | 919 | 1.77 | 3 |
| | 85917 | 2.42 | 3 | | 2038 | 0.10 | 2 | | 739 | 1.23 | 2 |

*PT – Playtime in minutes*
*Z – Z-score*

### 3.1 Rating Distribution

To see the overall distribution of rating, we plotted the z-score frequency distribution of the rating across the whole dataset. Figure 7 shows the plotted distribution. Visually we could see that the distribution resemble long tail distribution with tail are at the right side of the distribution.

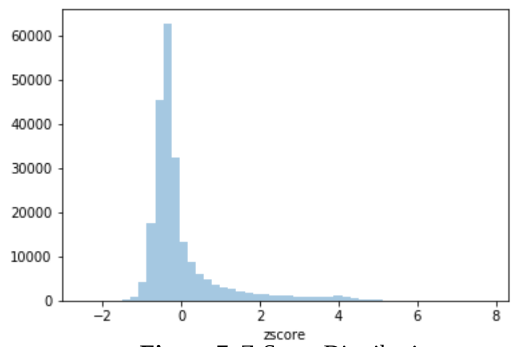

**Figure 7**: Z-Score Distribution

In our dataset context, this signifies that there are users with very extreme playtime pattern. Z-score that fall in between -3 to 3 (which indicate normal distribution) are not adequate to be mapped into 5-star ratings. We could say that using direct z-score mapping is biased towards users that played many games. User that played only several games seems not to be able to give 5-star rating to games. To removes this biased or at least to minimize the impact of long tail distribution on mapping process, we apply 3 strategies of mapping z-score into rating: SSB, BBS and HBS. Figure 8 shows the frequency of rating for each class when we are using different strategies. As a sanity check, we summarize minimum, maximum and average playtime for each strategy in Table 2.

For SSB, the distribution of rating leaning too much towards rating class 1 and 2 which reflect that majority of the players did not like the game they played. If we look at class 4 and 5 which indicated the players enjoyed the game, less than 5000 (about 2.2%) records belong to this class. As recommender system will based on the higher rating to suggest games to users, this will somehow reduce the number of games that are good enough to be suggested by the recommender system.

Using BBS strategies, although we managed to increase number of 5 ratings records, majority (74.4%) of the rating falls at rating 1 which means most of players did not like the game they played. Maximum play time of this class is 58964 minutes (Refer **Table 2**) which is equivalent to 40 days of playing. When a person spends that much time to play, it is quite wrong to conclude that he did not like the games regardless that he might played the other games longer. This situation arise because of the first class are derived from the means of z-score for the entire dataset.

Using HBS strategies, the distribution of data seems reasonable when only small fraction of records falls in rating 1 class. Maximum playtime for rating 1 class using HBS strategies is 763 minutes (Refer Table 2) which amount to 12 hours of playtime which is quite acceptable.

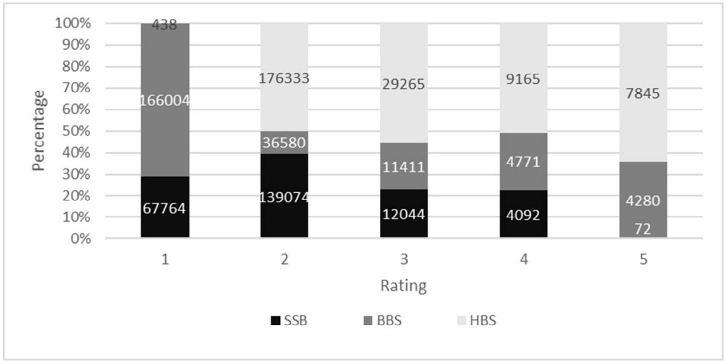

**Figure 8**: Summary of rating frequency for 3 methods

**Table 2**: Summary of Play Time for Each Class Rating Using Different Strategies

| | *Maximum Play Time (Minutes)* | | | *Minimum Play Time (Minutes)* | | | *Average Play Time (Minutes)* | | |
|---|---|---|---|---|---|---|---|---|---|
| *Rating* | **SSB** | **BBS** | **HBS** | **SSB** | **BBS** | **HBS** | **SSB** | **BBS** | **HBS** |
| *1* | 27581 | 58964 | 763 | 61 | 61 | 61 | 155 | 288 | 116 |
| *2* | 106519 | 106519 | 27581 | 61 | 101 | 61 | 678 | 1,311 | 277 |
| *3* | 371574 | 92226 | 106437 | 166 | 147 | 101 | 3,755 | 2,603 | 1,269 |
| *4* | 578398 | 371574 | 92226 | 272 | 217 | 147 | 15,676 | 4,955 | 2,548 |
| *5* | 527096 | 578398 | 578398 | 2554 | 272 | 220 | 46,602 | 15,961 | 10,166 |

We feed the rating obtain from each strategy using User K-Nearest Neighbors (KNN) and Item KNN to confirm that the rating could be used to predict user preferences. Table 3 shows the performance metric of both algorithm when using rating for each strategy. We measured accuracy of the prediction using RMSE and MAE. Lower value for RSME and MAE means that recommender algorithms can predict rating for unrated video games accurately. To check whether the algorithm can derive relevant top-10 games from rating constructed in the strategies proposed, we measure hit rate for overall dataset. Higher hit rate means a lot of left-out-items appeared in the top-N recommendation. These metrics are shows in Table 3.

**Table 3**: Performance Metric for Recommender System Model Using Created Explicit Rate

| | **RMSE** | | | **MAE** | | | **HIT RATE** | | |
|---|---|---|---|---|---|---|---|---|---|
| Algorithm | SSB | BBS | HBS | SSB | BBS | HBS | SSB | BBS | HBS |
| User KNN | 0.5496 | 0.7543 | 0.7950 | 0.3653 | 0.4914 | 0.4891 | 0.0000 | 0.0024 | 0.0090 |
| Item KNN | 0.6271 | 0.8963 | 0.8386 | 0.3974 | 0.6299 | 0.6243 | 0.0001 | 0.0001 | 0.0002 |

We could see that SSB bin strategies provide the most accurate prediction among the three strategies. If we desire an accurate rating prediction, it is best to binned z-scores in equal size bin. However, in the end of the day, the objective of a recommender system is to suggest items that hopefully will interest user enough for them buy or consumed the recommended items. Accurately predict rating is not enough, [26], [27], recommender system should be able to suggest items that interesting enough for user to consume. In that sense, hit rate are more relevant to measure the effectiveness of recommender system. Although hit rate are quite low for all
for all three strategies, we could see that HBS strategies performed a bit better compared to the others. To improve hit

rate, we could use other information available on users and games to calculate more accurate similarities between them.

To show that we got an acceptable value for RSME we compare RSME from [22] and [28] in Table 4. We choose this research because they are using the same metric and method. However, the result could be biased because they are using different dataset and different K value. We are not comparing RSME to measure performance of the recommender system but rather to show that the value that we get from using modified rating in our binning strategies are within the acceptable range.

**Table 4**: RSME comparison

| | |
|---|---|
| *Our RSME* | *0.7950* |
| *EI RSME[22]* | +-1.25 |
| *MovieLens RSME [28]* | 1.103 |

In summary, as MAE and RSME provide acceptable value for error rate, we could say that it is feasible to map playtime to explicit rating using z-score values. However, direct mapping could lead to biased for players that only played several games compared to users that play more games which lead us to consider different binning strategies. Binning strategies that deal with head and tail section of the dataset separately (HBS) provide the best result in Hit rate metric. However, if we take RMSE and MAE into account, direct mapping provides the best result.

As RMSE and MAE only measures error rate of rating prediction, it is wise to base our standard of the best strategies using hit rate value. While RSME for HBS is the highest, 0.79 error rate is quite good if we compared with the other research (refer table 3). Regardless of the algorithm and dataset used, 0.79 error rate could be considered acceptable and if we compared it with hit rate metric, HBS proof to be slightly better than the other strategies

## REFERENCES

1. S. Frémal and F. Lecron, **Weighting strategies for a recommender system using item clustering based on genres**, *Expert Syst. Appl.*, vol. 77, pp. 105–113, 2017. https://doi.org/10.1016/j.eswa.2017.01.031
2. D. Parra, A. Karatzoglou, X. Amatriain, and I. Yavuz, **Implicit feedback recommendation via implicit-to-explicit ordinal logistic regression mapping**, in *CEUR Workshop Proceedings*.
3. J. J. T, V. S. Chooralil, and J. John, **Efficient Route Recommendation System Based On Keyword Using Candidate Route Generation And Travel Route Exploration**, *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 8, no. 3, pp. 419–425, 2019. https://doi.org/10.30534/ijatcse/2019/15832019
4. N. Matsatsinis, K. Lakiotaki, and P. Delia, **A system based on multiple criteria analysis for scientific paper recommendation**, in *Proceedings of the 11th Panhellenic Conference in Informatics*.
5. B. Plunkett, B. Lin, S. Shi, and C. Painter, **The Steam Engine : A Recommendation System for Steam Users Algorithms**, pp. 1–6.
6. I. Mahazir, S. Khadijah, M. E. Ismail, and M. Nordin, **Impact of Games on Motivation , Attention and Skills in Pre-school Children**, *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 8, no. 1, pp. 157–159, 2019. https://doi.org/10.30534/ijatcse/2019/3181.32019
7. M. Romero and M. Usart, **Dementia Games: A Literature Review of Dementia-Related Serious Games**, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8101, no. January 2013, 2013, pp. 212–225.
8. J. Yim and T. C. N. Graham, **Using games to increase exercise motivation**, *Proc. 2007 Conf. Futur. Play - Futur. Play '07*, no. January 2007, p. 166, 2007. https://doi.org/10.1145/1328202.1328232
9. M. O'Neill, E. Vaziripour, J. Wu, and D. Zappala, **Condensing Steam: Distilling the Diversity of Gamer Behavior**, *Proc. 2016 ACM Internet Meas. Conf. - IMC '16*, pp. 81–95, 2016.
10. S. S. Iyengar and M. R. Lepper, **When choice is demotivating: can one desire too much of a good thing?**, *J. Pers. Soc. Psychol.*, vol. 79, no. 6, pp. 995–1006, 2000. https://doi.org/10.1037//0022-3514.79.6.995
11. E. R. Núñez-Valdéz, J. M. Cueva Lovelle, O. Sanjuán Martínez, V. García-Díaz, P. Ordoñez De Pablos, and C. E. Montenegro Marín, **Implicit feedback techniques on recommender systems applied to electronic books**, *Comput. Human Behav.*, vol. 28, no. 4, pp. 1186–1193, 2012.
12. G. Adomavicius and A. Tuzhilin, **Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions**, *IEEE Trans. Knowl. Data Eng.*, vol. 12, no. 5, p. 548, 2005.
13. H. Wu, Z. Zhang, K. Yue, B. Zhang, J. He, and L. Sun, **Dual-regularized matrix factorization with deep neural networks for recommender systems**, *Knowledge-Based Syst.*, vol. 145, pp. 46–58, 2018.
14. S. Chen and Y. Peng, **Matrix factorization for recommendation with explicit and implicit feedback**, *Knowledge-Based Syst.*, vol. 158, no. May, pp. 109–117, 2018. https://doi.org/10.1016/j.knosys.2018.05.040
15. T. V. R. Himabindu, V. Padmanabhan, and A. K. Pujari, **Conformal matrix factorization based recommender system**, *Inf. Sci. (Ny).*, vol. 467, pp. 685–707, 2018.
16. H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, **DeepFM: A Factorization-Machine based Neural Network for CTR Prediction**, pp pp. 1725–1731.
17. H.-T. Cheng *et al.*, **Wide & Deep Learning for Recommender Systems**, in *DLRS*.
18. Y. Kilani, A. F. Otoom, A. Alsarhan, and M. Almaayah, **A genetic algorithms-based hybrid recommender**

**system of matrix factorization and neighborhood-based techniques**, *J. Comput. Sci.*, vol. 28, pp. 78–93, 2018.
https://doi.org/10.1016/j.jocs.2018.08.007

19. **MovieLens Datasets**. [Online]. Available: https://grouplens.org/datasets/movielens/. [Accessed: 04-Dec-2018].
20. X. Zhao, X. Mao, L. Liu, J. Zheng, and Y. Liu, **An Improved Rating Mapping Algorithm for Music Recommender System**, *DEStech Trans. Eng. Technol. Res.*, no. apetc, pp. 1665–1670, 2018.
21. X. Zhu, D. Anguelov, and D. Ramanan, **Capturing long-tail distributions of object subcategories**, *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, no. 3, pp. 915–922, 2014.
22. Y.-J. Park and A. Tuzhilin, **The long tail of recommender systems and how to leverage it**, p. 11, 2008.
23. N. Hug, **Surprise Library for Python**. [Online]. Available: http://surpriselib.com/. [Accessed: 05-Aug-2019].
24. B. Jiang, **Head/Tail Breaks: A New Classification Scheme for Data with a Heavy-Tailed Distribution**, *Prof. Geogr.*, vol. 65, no. 3, pp. 482–494, 2013.
https://doi.org/10.1080/00330124.2012.700499
25. B. Alper and Y. Alper, **Improving accuracy of multi-criteria collaborative filtering by normalizing user ratings**, *Anadolu Univ. J. Sci. Technol. A- Appl. Sci. Eng.*, vol. 18, no. 1, pp. 225–237, 2017.
26. Y. Bai and D. Wang, **Fundamentals of Fuzzy Logic Control – Fuzzy Sets , Fuzzy Rules and Defuzzifications**, *Adv. Fuzzy Log. Technol. Ind. Appl.*, pp. 334–351, 2006.
27. P. Cremonesi, Y. Koren, and R. Turrin, **Performance of recommender algorithms on top-n recommendation tasks**, no. January 2010, p. 39, 2010.
https://doi.org/10.1145/1864708.1864721
28. A. D. Moreno Barbosa, **Privacy-enabled scalable recommender systems**, University Of Nice - Sophia Antipolis, 2015.