BT4014 Project Final Report

Group 9

Ng Hong Yao (A0235009M)

Toh Hui Shan Alicia (A0204411B)

Vinessa Christabella (A0240431X)

Table of Contents

1 Introduction

1.1 Background, Motivation & Problem Statement

MovieLens is an online community where users can rate movies, write movie reviews and receive movie recommendations. However, the community is confronted with the issue of under-contribution, where a significant number of users are not active on the platform (Ling et al., 2005; Chen et al., 2010). Given that the primary feature of MovieLens is its movie recommendation, this issue might be attributed to inaccurate movie recommendation, which decreases users satisfaction and their willingness to utilise the platform.

Our team aims to improve MovieLens' personalised movie recommendations by proposing bandit algorithms. We will explore two algorithms. Our first algorithm is an epsilon decay bandit algorithm combined with a content-based recommendation system (Figure 1). Our second algorithm is a contextual bandit algorithm which considers user features. These algorithms can increase user satisfaction by recommending high-quality movies based on their past movie preferences and personal characteristics. Through comparing these two approaches, we can determine which is more effective for MovieLens' recommendation system.

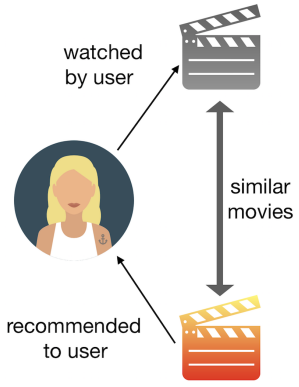

Figure 1: Content-Based Recommendation System (Movie Recommender System, 2021)

We choose bandit algorithms as they can be particularly useful for recommender systems where there is under-contribution, as they are designed to balance exploration and exploitation in order to optimise recommendations even in situations with limited data. The algorithm continuously updates its assessment of the user's interests and generates more precise recommendations when the user engages with the system and offers input on the suggested movies.

1.2 Findings and Implications

After analysing our epsilon decay and contextual bandit algorithms, we can see contextual bandit algorithms generally perform better than the epsilon decay algorithm. Specifically, the LinUCB hybrid algorithm with $\alpha = 0.25$ has the highest cumulative reward overtime at 8293.33 followed by LinUCB disjoint with $\alpha = 0.5$ at 8243.33, then Epsilon Decay with $k = 25$ at 6075.67. The cumulative reward

result is a useful indicator of the algorithm's overall performance for us to fully evaluate the algorithm effectiveness in recommending movies to our viewers.

Hence, we can conclude that LinUCB hybrid with $\alpha = 0.25$ is the best bandit algorithm to be considered for the MovieLens recommender system among our proposed algorithms.

2 Data

2.1 Dataset Description

For this project, we used the MovieLens dataset obtained from the GroupLens website (https://grouplens.org/datasets/movielens/). It is collected by the GroupLens Research Project at the University of Minnesota. The dataset comes in various sizes and we will be using the 100K variant of the data set, consisting of 100,000 ratings from 943 users on 1682 movies and each user has rated at least 20 movies.

We will be focusing on 3 main tables in the dataset:

- Users data: Includes user ID with corresponding user demographics such as their age, gender, occupation and zip code.
- Movies data: Includes information about movie details such as genres and release date. In our project, movies represent the arms while their genres and release date represent the attributes of the arms.
- Ratings data: Includes movie ratings submitted by each user.

2.2 Data Preprocessing

2.2.1 Context Features

In our experiment, the context refers to the features or demographics of the users. Since age is a continuous numeric data, we categorised it into multiple groups for further analysis. We created six age group buckets, which are under 20, 20 - 29, 30 - 39, 40 - 49, 51 - 60, and above 60. Next, we performed one-hot encoding on the categorical variables namely "age_group", "gender" and "occupation" using sklearn's OneHotEncoder. Additionally, we decided to omit "zipcode" in our analysis.

2.2.2 Arms Attributes

The attributes of the arms are represented by their "genres" and "release_date". We kept all 19 genres of movies and removed unused features such as its "IMDb URL" and "video_release_date" from the movies dataset.

2.2.3 Rewards

The rewards are the ratings that users give to a movie. In the dataset, the ratings range from 1 to 5. For the purpose of our experiment, we created a binary reward label, where ratings less than 4 were labelled 0 and 1 otherwise. We chose this threshold as 4 is close to the peak of the distribution of

ratings as shown in Figure 2. To account for limited resources, we also filtered the ratings data to include only 30 movies that have the most ratings on the website.

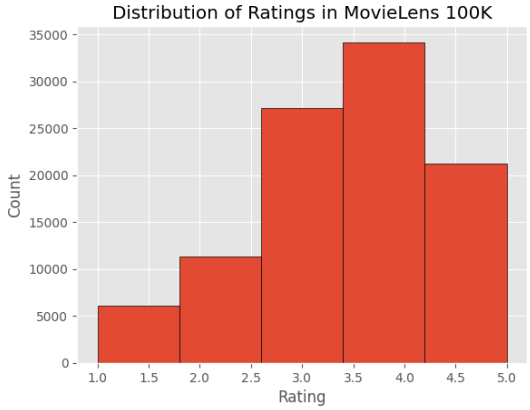

Figure 2: Distribution of Ratings

3 Empirical Analysis

We explored 2 types of algorithms, epsilon decay and contextual bandit. In order to mirror a real-world online learning scenario, we chose to conduct bootstrap resampling, with 3 bootstrap samples. In each resampling, the process begins with an empty history dataframe, which represents no knowledge of how users behave. Subsequently, it will start picking up on users' behaviour as it recommends movies that match with the recommendations present in the logged dataset. Each time there is a match in record, it will add this context to the history dataset and use it as future context for improving its recommendation policy. Over numerous iterations, the history grows larger and the algorithm will become more effective at picking the best arm.

3.1 Epsilon Decay

We modified the traditional epsilon decay algorithm to better suit our movie recommendation context and take into account individual user behaviour and preferences.

3.1.1 Decay Function

The epsilon decay algorithm gradually reduces the probability of exploration and increases the probability of exploitation over time. Unlike traditional epsilon decay, which relies on the overall timestep, our approach takes into account the unique history of each user, enabling us to tailor the algorithm to their behaviour. For instance, users who have watched a large number of movies may require less exploration and more exploitation than users with less viewing history. Hence, we define our decay function as follows:

$$epsilon = \frac{1}{(\frac{number\ of\ movies\ watched\ by\ a\ user}{number\ of\ arms} + 1)}$$

As the number of movies watched by a user increases, the epsilon value, and hence the probability to explore, decreases.

3.1.2 Exploitation

Our exploitation strategy makes use of a content-based recommendation, which suggests a movie of similar characteristics to the movies a user has previously enjoyed. In our case, we measured the similarity of movies based on genres of a movie. For example, as shown in Figure 3, if a user enjoyed a movie with crime and drama genre, as well as another movie with drama and thriller genre, we can recommend a similar movie with crime, drama and thriller genre.

	movie_id	movie_title	release_date	genres
0	1	Toy Story (1995)	01-Jan-1995	Animation|Children's|Comedy
1	7	Twelve Monkeys (1995)	01-Jan-1995	Drama|Sci-Fi
2	50	Star Wars (1977)	01-Jan-1977	Action|Adventure|Romance|Sci-Fi|War
3	56	Pulp Fiction (1994)	01-Jan-1994	Crime|Drama
4	69	Forrest Gump (1994)	01-Jan-1994	Comedy|Romance|War
5	79	Fugitive, The (1993)	01-Jan-1993	Action|Thriller
6	98	Silence of the Lambs, The (1991)	01-Jan-1991	Drama|Thriller
7	100	Fargo (1996)	14-Feb-1997	Crime|Drama|Thriller

Figure 3: Illustration on content based recommendation

We achieved this by first converting the "genres" column to a matrix of token counts. We then recommended a movie that is the most similar to the movies that a user liked (his/her movie history with reward = 1) by using the nearest neighbours algorithm with cosine distance metric (*Movielens Movie Recommendation System*, 2022).

3.1.3 Exploration

We explored the arms by narrowing down to the top k recent movies and randomly recommending one of them. This way, we could limit the exploration to more recent movies (within the 30 arms) based on the dataset's "release_date". Limiting to recent movies are more likely to be appealing and relevant to viewers, which in turn increases the likelihood that they will enjoy them. A study conducted on video recommendations also shows that the recency of the video leads to a higher level of user engagement (Zhong et al., 2018).

3.1.4 Comparison Between Different k Values

We ran 3 different k values to see which is the best in terms of cumulative reward over time. The graph below shows the comparison of cumulative rewards for the epsilon decay algorithm implemented with $k = 5$ labelled as epsilon_decay_k_5, $k = 15$ labelled as epsilon_decay_k_15, and $k = 25$ labelled as epsilon_decay_k_25.

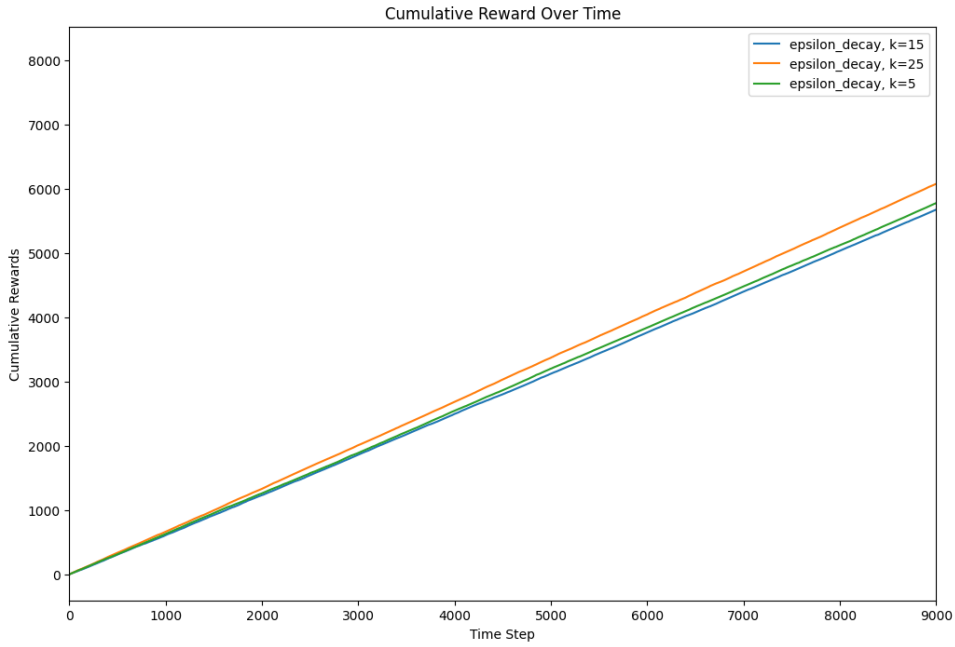

Figure 4: Graph of cumulative rewards for Epsilon Decay Algorithms

We observe that the epsilon_decay_k_25 (represented by orange line), where the algorithm chooses top 25 recent movies to explore, has the fastest increase in cumulative rewards as well as higher cumulative rewards in the long run. Therefore, among the three k values, $k = 25$ is the most promising choice. We can also observe that since $k = 5$ (represented by green line) performs better than $k = 15$ (represented by blue line), lower k does not necessarily mean lower cumulative rewards.

3.2 Contextual Bandit

In a contextual bandit approach, an additional context vector is used when learning which is the best outcome given a particular situation. We have information on the users' demographics and such contextual information may be useful for us to recommend the appropriate variant as different variants may be more effective for certain types of users.

We explored 2 types of contextual bandits namely Linear Upper Confidence Bound Disjoint (LinUCB Disjoint) and Linear Upper Confidence Bound Hybrid (LinUCB Hybrid), with 3 values of α, which is our hyperparameter. The greater the value of α, the wider the confidence bound becomes, which results in a higher emphasis placed on exploration instead of exploitation.

3.2.1 Linear Upper Confidence Bound Disjoint (LinUCB Disjoint)

The Disjoint LinUCB bandit algorithm operates under the assumption that each arm is distinct from one another and has unique features. At each time step, the algorithm will choose the arm with the greatest UCB value, which is the sum of the estimated mean and its confidence bound, and will observe the reward for the selected arm only. This allows the algorithm to both exploit arms with known high returns and explore arms with potential but uncertain returns, ultimately maximising its overall performance over time.

To implement this algorithm, we created 3 objects:

- A class object to represent a LinUCB disjoint arm
- A class object for the policy with the specified number of LinUCB disjoint arms
- Function that implements bootstrap replay using the LinUCB policy created above

The graph below shows the comparison in cumulative rewards for the LinUCB disjoint algorithms implemented with 3 alpha values.

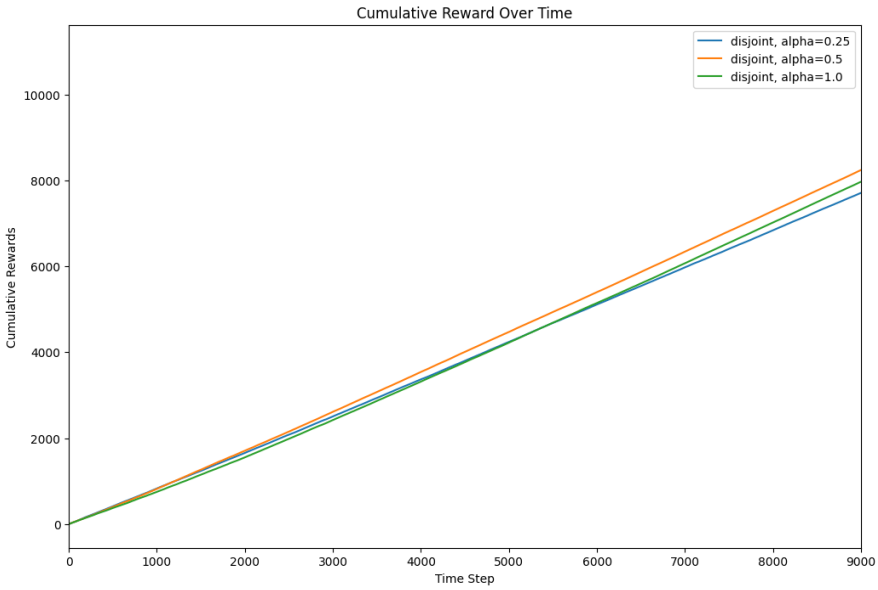

Figure 5: Graph of cumulative rewards for LinUCB Disjoint Algorithms

We observe that up to time step 5000, the algorithm with $\alpha = 1.0$, represented by the green line, has the slowest rate of increase in cumulative rewards, which is expected due to its emphasis on exploring arms with high uncertainty. However, after time step 5000, the algorithm with $\alpha = 1.0$ surpassed the cumulative rewards of the algorithm with $\alpha = 0.25$, indicating that sufficient exploration had taken place.

The algorithm with $\alpha = 0.5$ displayed consistent performance with the highest cumulative rewards at any given time step. However, as previously noted, the performance of the algorithm with $\alpha = 1.0$ improved significantly after time step 5000, suggesting the possibility of it surpassing the cumulative rewards of the algorithm with $\alpha = 0.5$ in the long run.

3.2.2 Linear Upper Confidence Bound Hybrid (LinUCB Hybrid)

In many real-world scenarios, the arms in a bandit experiment may not always be mutually exclusive in their properties. For instance, in our movie recommendation system where movies are the arms recommended to users, certain movies may have similar features, such as sharing the same genre. To address such situations, we can employ the LinUCB Hybrid model, which models the reward payoff of each arm as a linear function of both non-shared and shared components. Hence, the model can capture the similarities between arms and leverage this information to make more informed decisions about arm selection.

As previously observed in Figure 3, a movie may belong to more than one genre. In contrast to LinUCB Disjoint, where information is only updated for the observed movie at each iteration, LinUCB Hybrid allows for updating information about the reward payoff for similar movies or arms.

To implement this algorithm, we created 3 objects:

- A class object to represent a LinUCB hybrid arm
- A class object for the policy with the specified number of LinUCB hybrid arms
- A function that implements bootstrap replay using the LinUCB policy created above

The graph below shows the comparison in cumulative rewards for the LinUCB hybrid algorithms implemented with 3 alpha values.

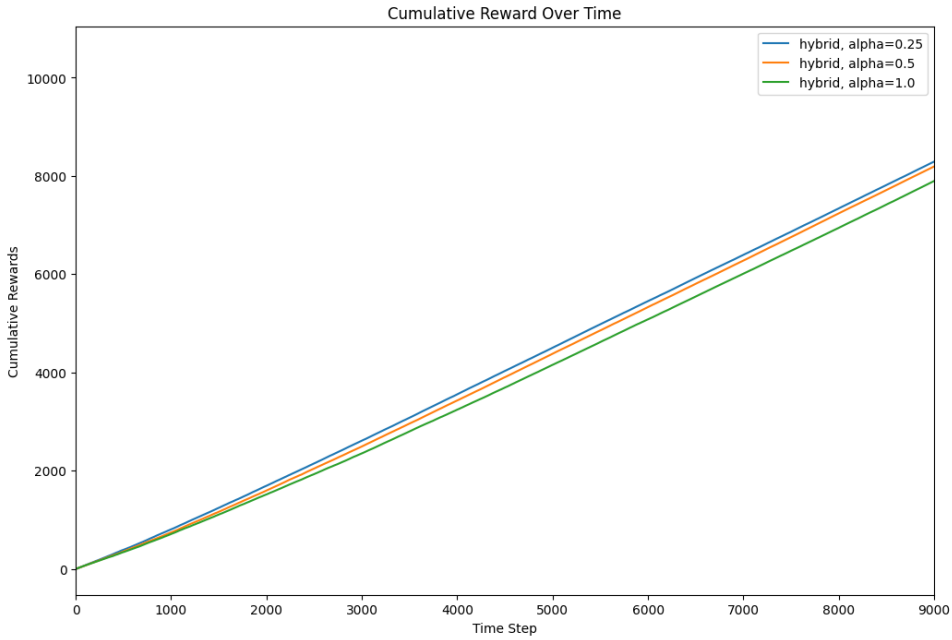

Figure 6: Graph of cumulative rewards for LinUCB Hybrid Algorithms

Due to the lower weightage of exploration, we observe that the policy with $\alpha = 0.25$ has the fastest increase in cumulative rewards and hence greatest cumulative rewards at the end of time step 9000. Policies with $\alpha = 0.25$ and $\alpha = 0.5$ had a similar performance, while policy with $\alpha = 1.0$ had a significantly slower rate in accumulating rewards. These results suggest that decreasing the weightage of exploration may lead to a faster accumulation of rewards.

3.2.3 Comparison Between Disjoint & Hybrid Models

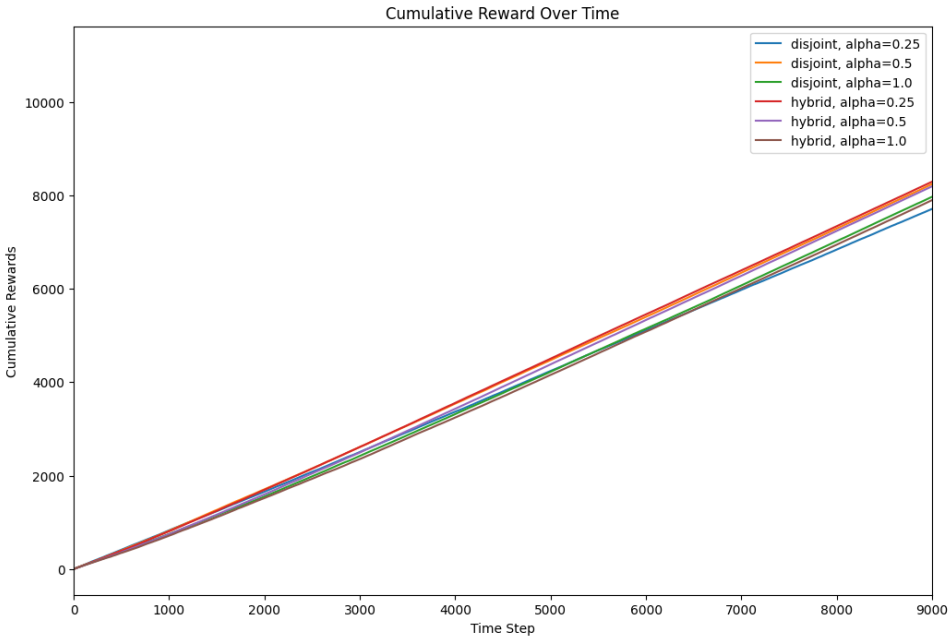

Figure 7: Graph of cumulative rewards for LinUCB Disjoint & Hybrid Algorithms

Our empirical investigation reveals that the LinUCB Hybrid model exhibits better performance compared to the LinUCB Disjoint model, as demonstrated by the higher cumulative reward achieved at the conclusion of 9000 time steps. This superiority can be attributed to the hybrid model's capacity to leverage shared and non-shared features of the arms, thereby improving the accuracy of the reward function for the arms.

3.3 Comparison Between Epsilon Decay & Contextual Bandit

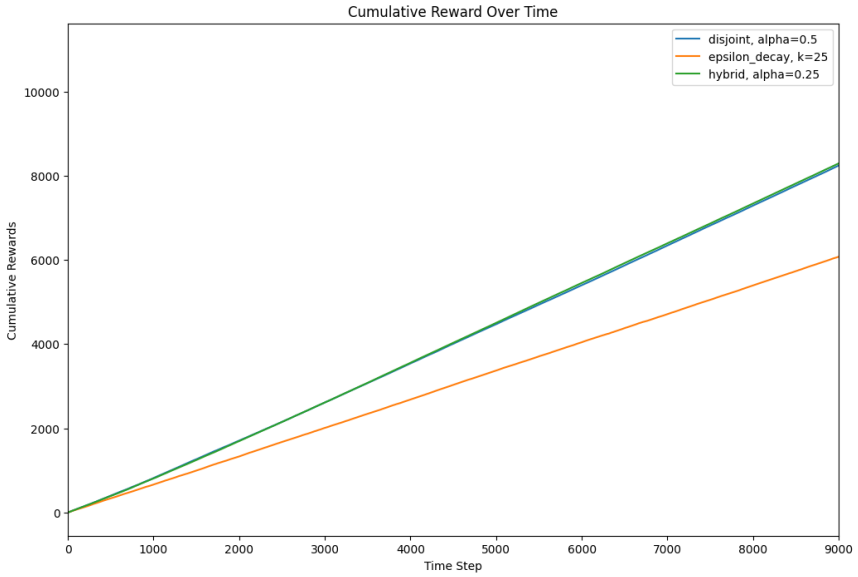

Figure 8: Graph of cumulative rewards for Epsilon-Decay, LinUCB Disjoint & Hybrid Algorithms

For the overall analysis, we plotted the cumulative rewards for the top performing model of each algorithm. From section 3.1 and 3.2, we concluded that the 3 models are epsilon decay with $k = 25$, LinUCB disjoint with $\alpha = 0.5$ and LinUCB hybrid with $\alpha = 0.25$. Finally we compared the cumulative rewards for the 3 algorithms and conclude that the best performing model is LinUCB hybrid with $\alpha = 0.25$. It achieved the greatest cumulative reward of 8293.33 at time step 9000. This value is higher than that of LinUCB disjoint with $\alpha = 0.5$ and epsilon decay with $k = 25$ which have a cumulative reward of 8243.33 and 6075.67 respectively.

4 Limitations & Future Improvements

4.1 Overall Experiment

Due to resource and time constraints, we filtered the dataset to include only a subset of the movies. While this allowed us to conduct our experiments in a more manageable and efficient manner, it also limited the diversity and representativeness of the movies in our dataset. In the future, it would be beneficial to expand the dataset and incorporate more diverse movies to obtain a more comprehensive evaluation of the algorithms' performance.

To further enhance the scope of our study and advance the field of recommender systems, future research can also consider investigating the generalizability of our findings to other datasets and platforms. Additionally, exploring the effectiveness of other types of bandit algorithms, such as Thompson sampling, can offer valuable insights into their potential to further improve recommendation quality.

4.2 Epsilon Decay

Our study employed a content-based recommender system as part of the exploitation phase. However, it should be noted that there are various types of recommender systems that can be explored for potential integration, such as collaborative filtering. Further investigation and experimentation can be conducted to determine the most suitable recommender system for a given application.

Additionally, as explained in section 3.1.3, k is a hyperparameter that determines the number of top recent movies to be explored during an exploration phase in the epsilon decay algorithm. We have only experimented with 3 values (5, 15 and 25) and therefore it should be noted that there might be other k values that might work better for the epsilon decay algorithm in the exploration phase.

4.3 Contextual Bandit

As explained in section 3.2, α is a hyperparameter that controls the balance between exploration and exploitation in our experiment. Our study has only investigated three specific values of α (0.25, 0.5 and 1.0), and it is important to note that other values might be optimal for various policies, depending on the number of arms and their similarities.

Additionally, to improve on our analysis, we can explore other possibilities in creating better contextual information such as creating an interaction term between linear covariates such as age_group and gender or even to include zipcode (a variable that we chose to exclude in the current experiment) as a covariate to improve the recommendations made by our policy.

5 Conclusion

In summary, this study aimed to improve the movie recommendations system of the MovieLens online community by proposing and evaluating two bandit algorithms: an epsilon decay bandit algorithm combined with a content-based recommendation system, and a contextual bandit algorithm which consists of LinUCB Disjoint and LinUCB Hybrid.

After evaluating the performance of these bandit algorithms, we identified that each approach had its own unique strengths and limitations. However, we ultimately chose to focus on the algorithm that demonstrated the best cumulative reward over time. This decision was based on the algorithm's ability to recommend movies that users are more likely to prefer, potentially leading to increased user ratings and engagement on the MovieLens platform.

Our results indicated that the contextual bandit algorithm outperformed the epsilon decay bandit algorithm, yielding higher cumulative rewards. By incorporating user features, the algorithm was able to better capture user preferences and improve recommendation quality, ultimately giving more accurate and personalised recommendations which can potentially enhance user satisfaction and engagement. Furthermore, among the contextual bandit algorithms that we evaluated, the LinUCB Hybrid algorithm with a hyperparameter α = 0.25 performed the best.

Overall, our study contributes to the growing body of research on bandit algorithms in recommender systems and highlights their potential to enhance the user experience in online communities like MovieLens. By improving the accuracy and relevance of movie recommendations, we hope to encourage more users to engage with the MovieLens platform and ultimately create a more vibrant and dynamic community of movie enthusiasts.

6 Github Repository

Link: https://github.com/xxalicia/BT4014_FinalProject_G09

Note: It may take over 30 hours to run the following file: LinUCB.ipynb

7 References

Chen, Y., Harper, F. M., Konstan, J., & Li, S. X. (2010). Social comparisons and contributions to online communities: A field experiment on movielens. *American Economic Review*, *100*(4), 1358–1398. https://doi.org/10.1257/aer.100.4.1358

Ling, K., Beenen, G., Ludford, P., Wang, X., Chang, K., Li, X., Cosley, D., Frankowski, D., Terveen, L., Rashid, A. M., Resnick, P., & Kraut, R. (2005). Using social psychology to motivate contributions to online communities. *Journal of Computer-Mediated Communication*, *10*(4), 00–00. https://doi.org/10.1111/j.1083-6101.2005.tb00273.x

Movie recommender system. (2021, August 13). [Github]. Rohan Sarkar. https://github.com/rohan-sarkarr/Movie-Recommender-System

Movielens movie recommendation system. (n.d.). Retrieved 21 April 2023, from https://kaggle.com/code/sachinsarkar/movielens-movie-recommendation-system

Zhong, Y., Menezes, T. L. S., Kumar, V., Zhao, Q., & Harper, F. M. (2018). A field study of related video recommendations: Newest, most similar, or most relevant? *Proceedings of the 12th ACM Conference on Recommender Systems*, 274–278. https://doi.org/10.1145/3240323.3240395