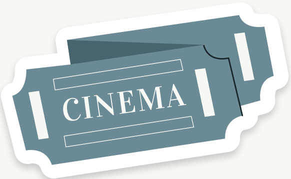


BT4014

Final Presentation

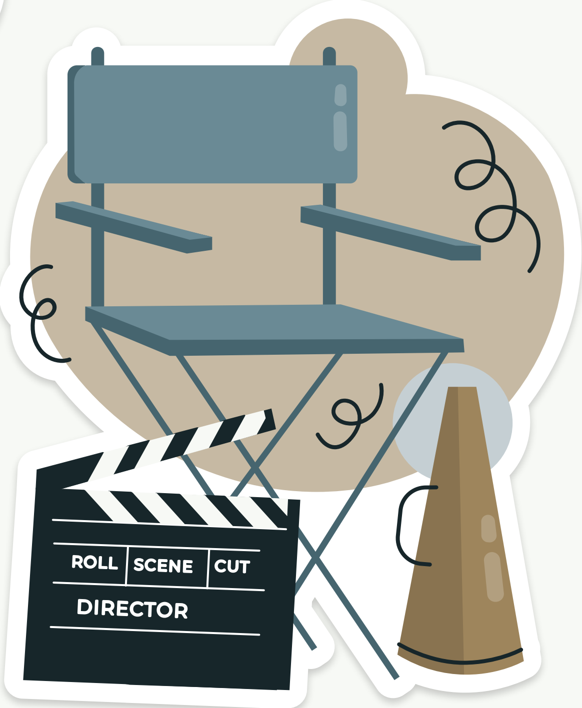
Movie Recommendation System

Ng Hong Yao (A0235009M)

Toh Hui Shan Alicia (A0204411B)

Vinessa Christabella (A0240431X)

Table of contents

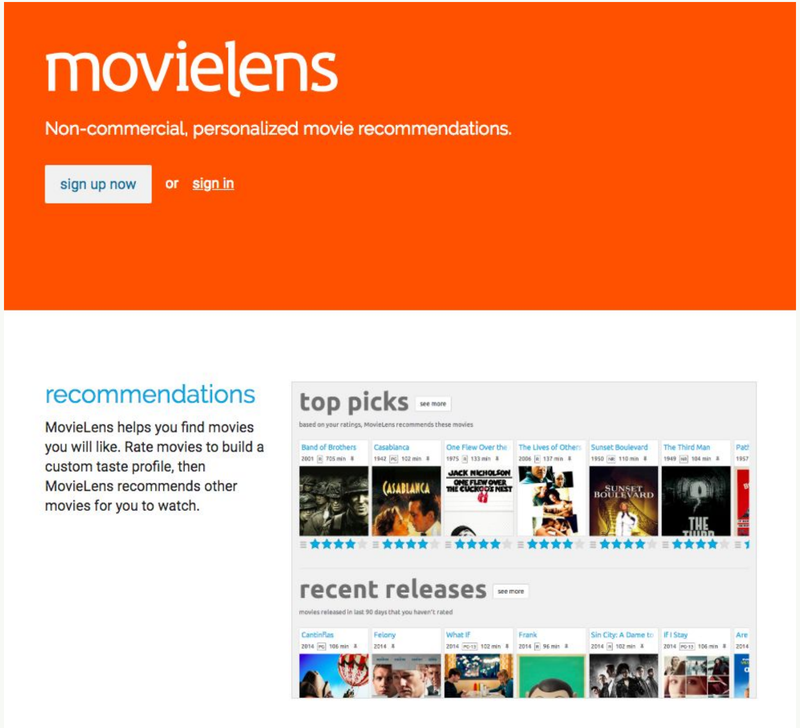


01

Introduction

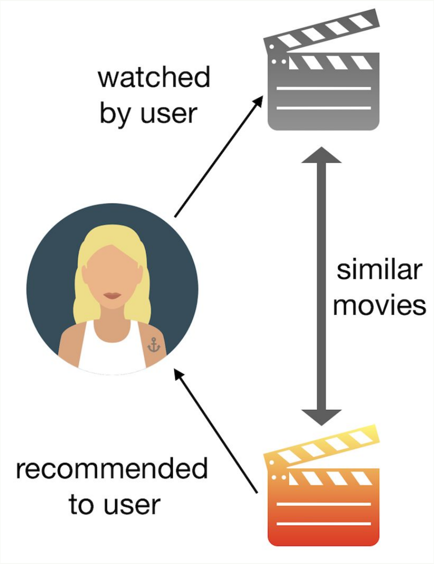
Background / Problem

- Online community where users can **rate** movies, **write** movie reviews and receive **movie recommendations**
- Problem of **under-contribution**
- **Inaccurate recommendation**, decrease in users' satisfaction & willingness to use the platform

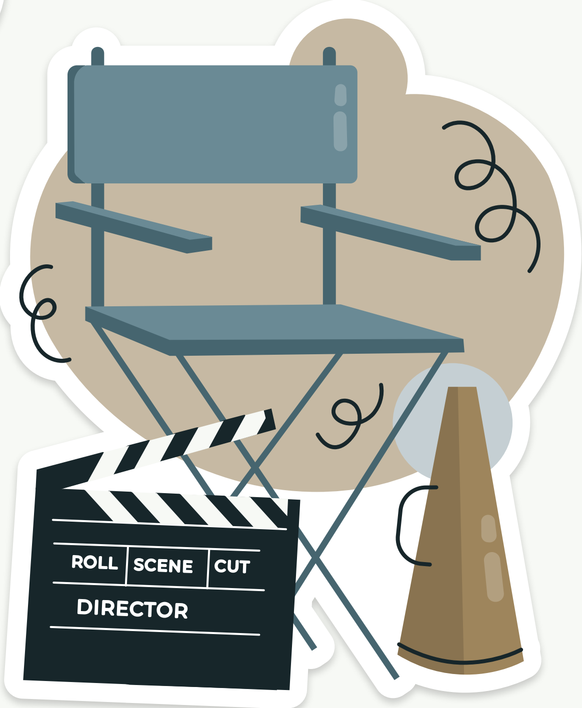


Proposed Solution

- Bandit Algorithms
  - **Epsilon decay** bandit algorithm combined with a content-based recommendation system
  - **Contextual bandit** algorithm which considers user features
- Goal: Increase user satisfaction by recommending high-quality movies



Content Based Recommendation System



02

Dataset

Dataset Description

- MovieLens dataset obtained from the grouplens website
- 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
- 3 tables of interest:
    - **Users data**: Includes user ID with corresponding user demographics
    - **Movies data**: Includes information about movies based on their genres
    - **User Movie Ratings**: Contains users’ ratings (1-5) for a movie

Dataset Preprocessing

Context Features

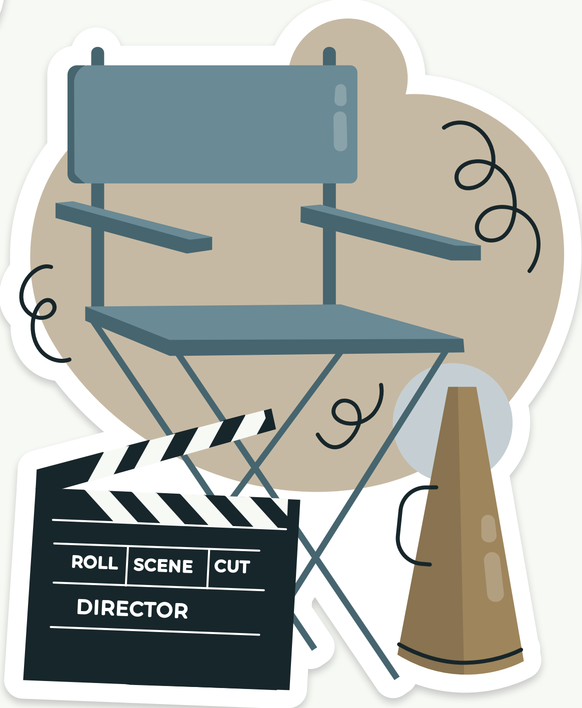
- Performing **data binning** on “age”
- 6 age group buckets: <20, 20-29, 30-39, 40-49, 51-60, >60
- Conduct **one-hot encoding** on categorical variables

Rewards

- Create **binary reward label**
- Ratings less than 4 are labeled as 0 and 1 otherwise

Filter

- Only include **30 movies** with most ratings on the website



03

Epsilon Decay

Epsilon Decay Modifications

Decay Function

Exploitation

Exploration

Decay Function

Original decay function

$$epsilon = \frac{1}{\left(\frac{\boxed{number\ of\ play}}{number\ of\ arms} + 1\right)}$$

Modified decay function

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

Decay Function

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

User A

10 movies

User B

2 movies

Lower epsilon value

Exploit more

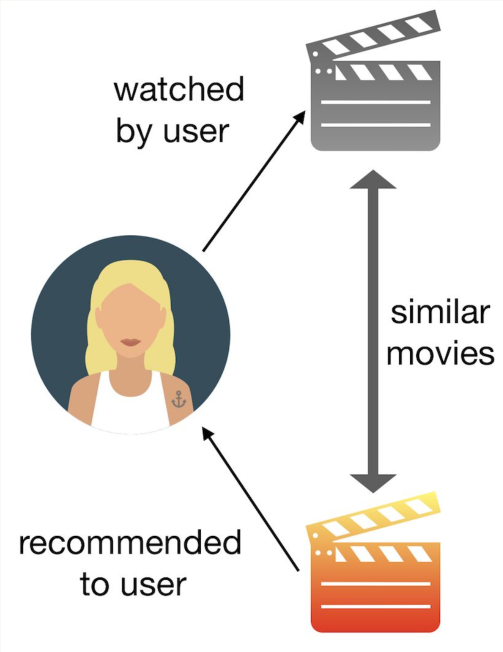
Higher epsilon value

Exploit less

Exploitation

Content Based Recommendation

suggests a movie which has similar characteristics to the movies a user has previously enjoyed



Exploitation

Content Based Recommendation

	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

User A liked these movies (gave high ratings)

Recommend this to user A

Exploitation

Content Based Recommendation

	action	adventure	animation	children	comedy	crime	drama
0	0	0	1	1	1	0	0
1	0	0	0	0	0	0	1
2	1	1	0	0	0	0	0
3	0	0	0	0	0	1	1
4	0	0	0	0	1	0	0

How we achieve this:

- Prepare data: Matrix of token counts for movie genres
- Algorithm: Nearest neighbour algorithm with cosine distance metric
    - most similar to the history of movies that the user enjoyed before
- Filter: the user has not watched that movie before

*if no result, then recommend overall best movie so far

Exploration

Recommend recent movie

Method
Pick a random movie from ‘k’ most recent movies among the arms

Rationale
Recent movies are more likely to be appealing and relevant to viewers

Hyperparameter
k $\longrightarrow$ controls the number of top recent movies selected for exploration

We explored 3 values of k (5, 15, 25)

Epsilon Decay Algorithm

Implementation:

- A class object for the content based recommendation system (called Recommender) that is used for exploitation part

```
class Recommender:
```

- For each k value, a function that contains epsilon decay algorithm

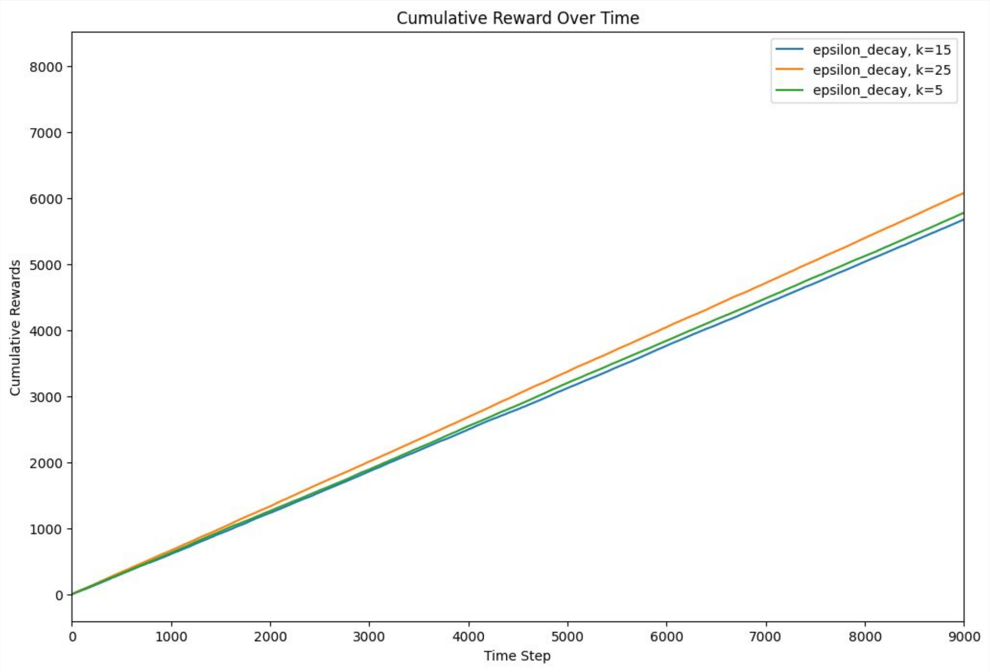
```
def epsilon_decay_k_5(user_history, history):

def epsilon_decay_k_15(user_history, history):

def epsilon_decay_k_25(user_history, history):
```

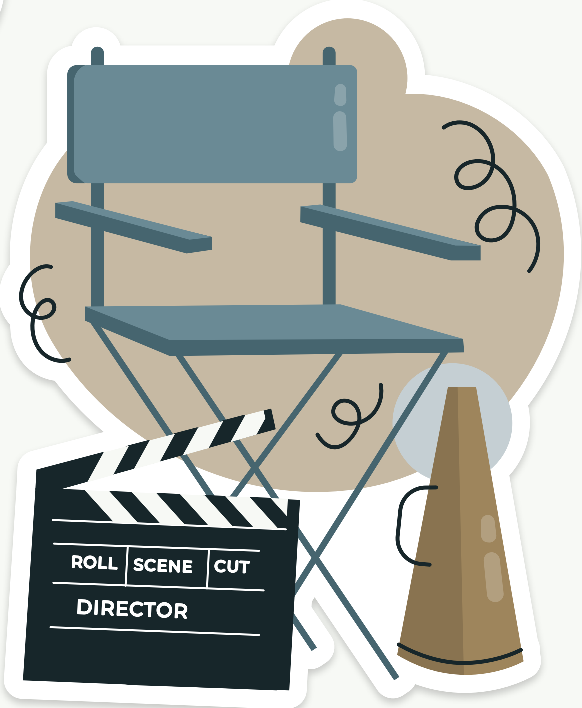
- Bootstrap resampling using 3 bootstrap samples

Epsilon Decay Algorithm



Cumulative reward over time:

- **k = 25** has **highest** cumulative rewards
- **k = 15** has the **lowest** cumulative rewards
- **k = 5** is higher than **k = 15**, lower k doesn't mean lower cumulative rewards



04

Contextual Bandit

Contextual Bandit

- Additional **context vector** is used during learning
- Information such as **user's demographics** (age, gender, occupation)
- Ability to make different recommendations to different groups of users with different tastes & preferences

Hyperparameter

- α, controls balance between exploration & exploitation
- **Higher** value of α, wider the confidence bound, **greater emphasis** on **exploration**

Contextual Bandit

Linear Upper Confidence Bound

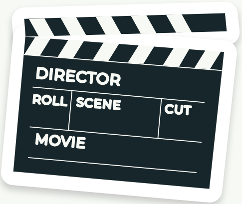
Disjoint

Linear Upper Confidence Bound

Hybrid

LinUCB Disjoint

- Arms (movies) are **distinct** from each other
- Each time step:
  - Select arm with **greatest** UCB value
  - Observe reward for selected arm only
- Strength:
  - Ability to **explore** arms with potential but uncertain returns
  - While **exploiting** arms that are known to have high returns



LinUCB Disjoint

3 Objects For Implementation:

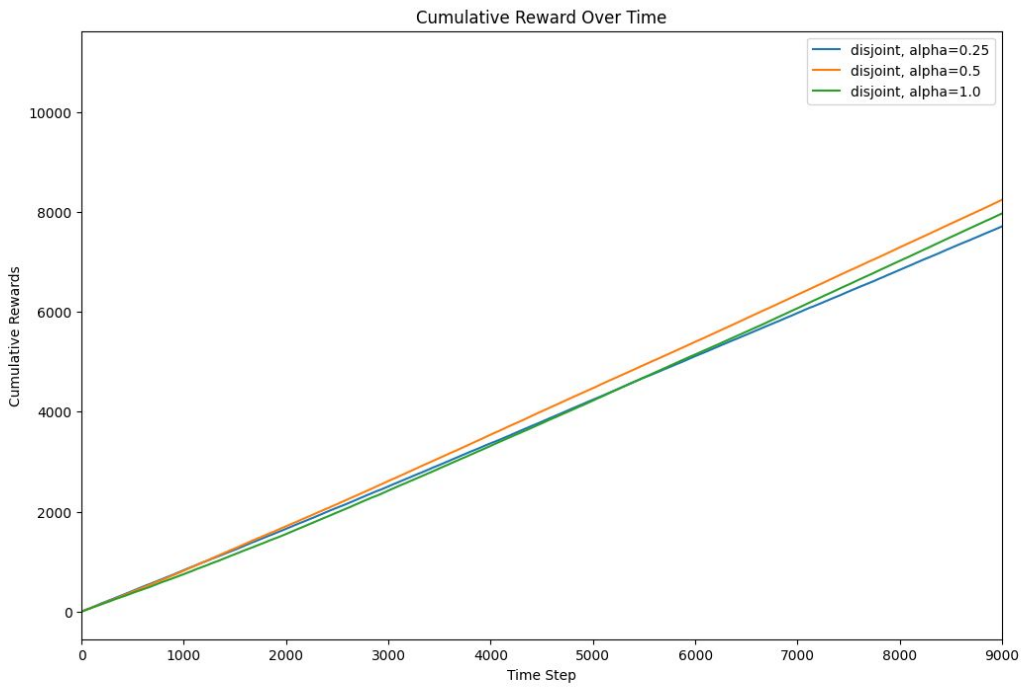
- A class object to represent a LinUCB disjoint arm
- A class object for the policy with the specified number of LinUCB disjoint arms
- A function that implements bootstrap replay using the LinUCB policy created above

```
class linucb_disjoint_arm():
```

```
class linucb_disjoint_policy():
```

```
def disjoint_bootstrap_replay(K_arms, d, alpha, top_movies_index, bootstrap_resample):
```

LinUCB Disjoint



Time Step < 5000:

- α = 1.0 has **slowest** increase in cumulative rewards due to greater emphasis on exploring arms with high uncertainty

Time Step > 5000:

- Cumulative reward of α = 1.0 surpass that of α = 0.25
- α = 0.5 has **greatest** amount of cumulative rewards

LinUCB Hybrid

- Arms (movies) are **not mutually exclusive** in properties
- Each time step:
  - Select arm with **greatest** UCB value
  - Observe reward for selected arm & **similar arms**
- Strength:
  - We account for **shared features** between arms
  - Reward payoff is a linear function of both non-shared and shared components

LinUCB Hybrid

3 Objects For Implementation:

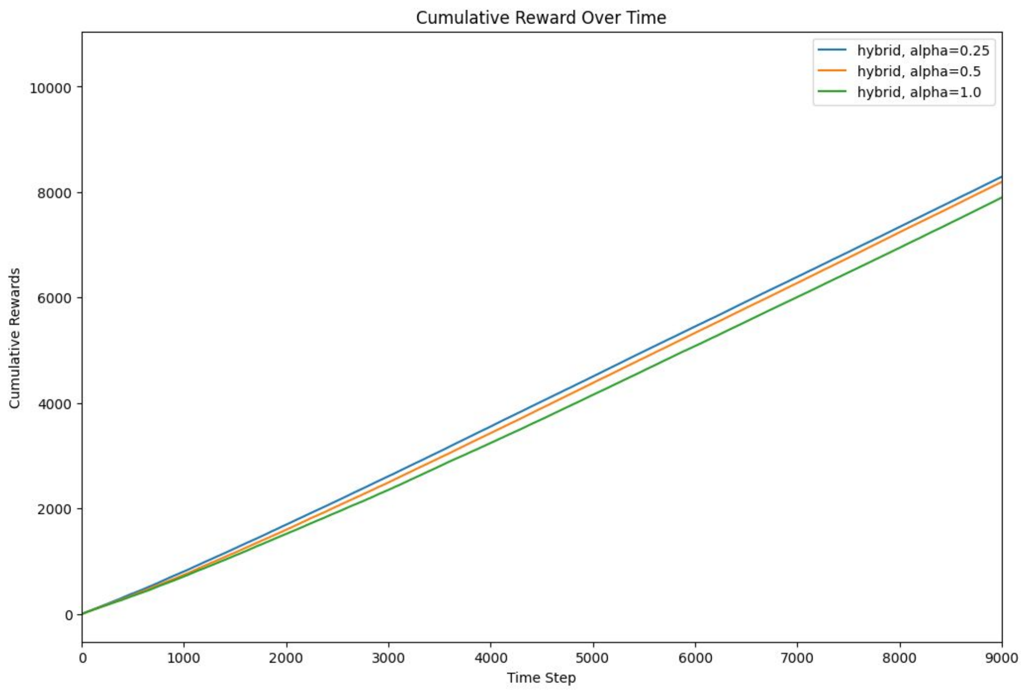
- A class object to represent a LinUCB hybrid arm
- A class object for the policy with the specified number of LinUCB hybrid arms
- A function that implements bootstrap replay using the LinUCB policy created above

```
class linucb_hybrid_arm():
```

```
class linucb_hybrid_policy():
```

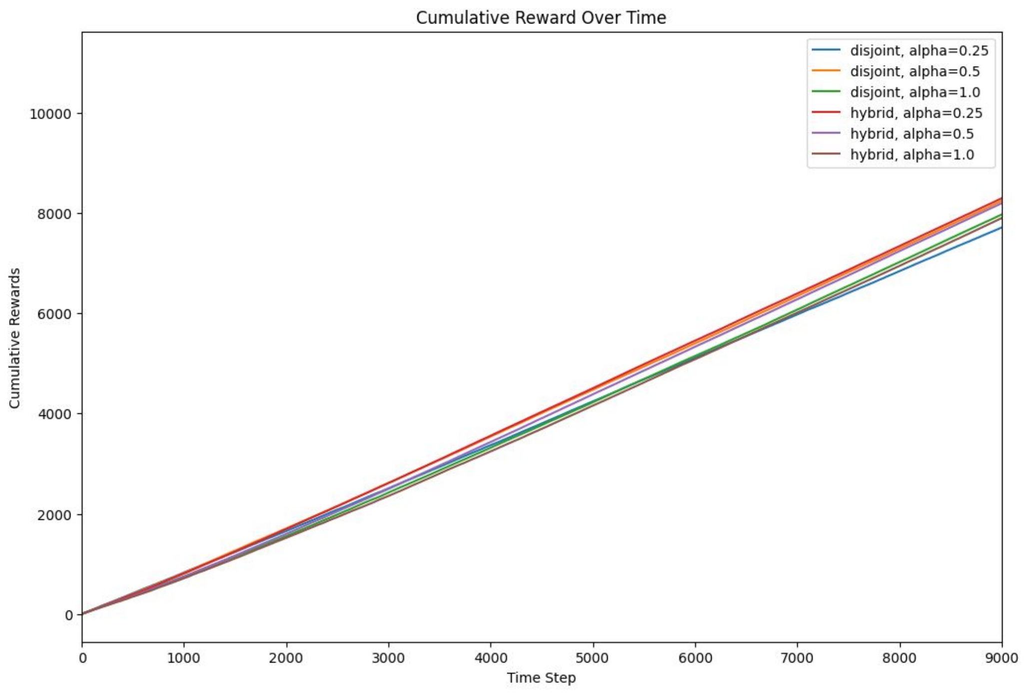
```
def hybrid_bootstrap_replay(K_arms, d, k, alpha, top_movies_index, top_movies_features, bootstrap_resample):
```

LinUCB Hybrid

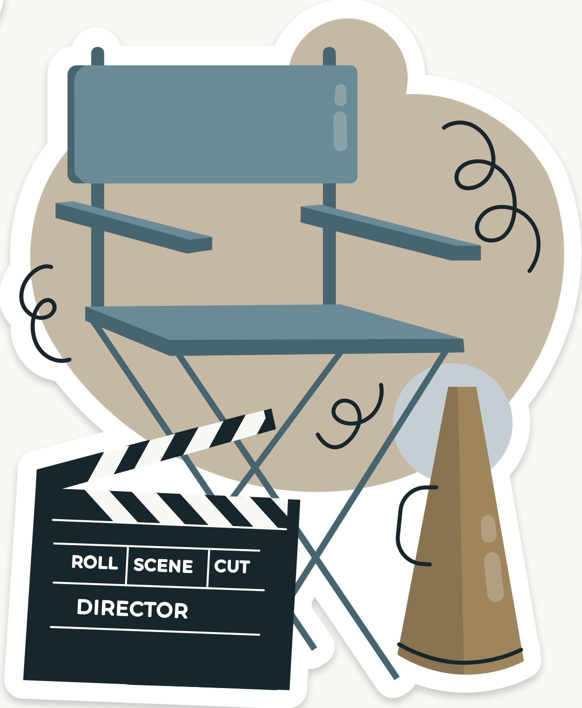


- α = 0.25 has **greatest** amount of cumulative rewards at all time step
- Lower weightage of exploration
- Significantly slower performance for α = 1.0

LinUCB Disjoint vs Hybrid



- **Hybrid** models had **greater** cumulative rewards than disjoint models
- Advantage: utilizes shared features of the arms to help model the reward function for arms with similar features



05

Limitations

Limitations

Overall project

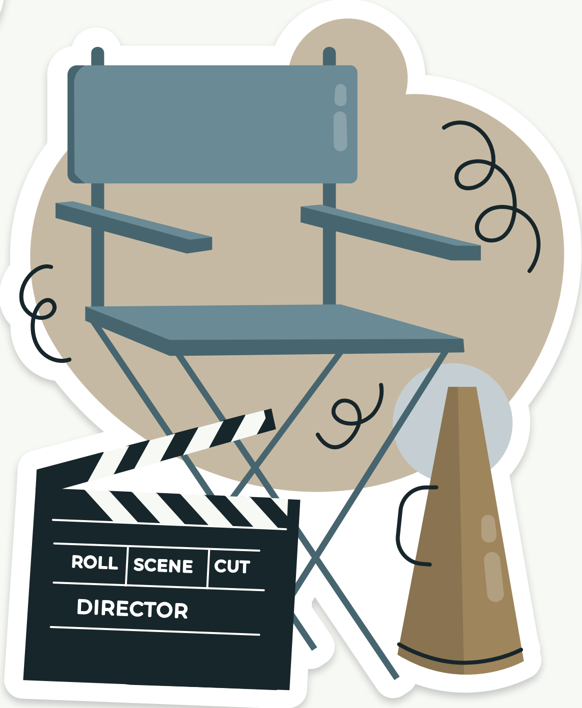
- Due to resource constraints, we **filtered** the dataset to include only **a subset** of the available movies.

ε-Decay

- We only experimented with 3 ***k* values** (5, 15, 25) but other values could work better in the exploration phase
- We employed content-based recommender system as part of exploitation phase. There could be **other recommender system to be integrated.**

Contextual Bandit

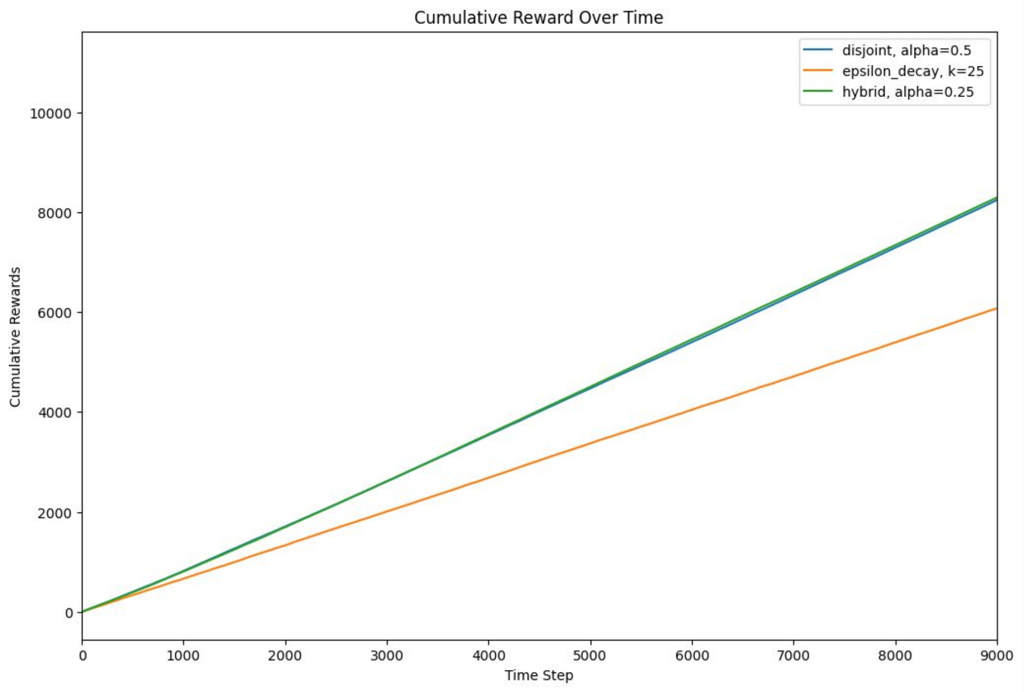
- We only experimented with 3 ***α* values** (0.25, 0.5, 1.0) but other values could work better
- We could explore more **variations of contextual information** such as creating interaction terms from the existing covariates



06

Conclusion

Conclusion



- **Best performing** model is LinUCB Hybrid, α = 0.25 with greatest cumulative reward of 8293.33
- Epsilon decay, k = 25 performed **significantly worse** than LinUCB with cumulative reward of 6075.67

Thank You