Final Project

Team 7

Jainam Rajesh Jagani - G01405187
Shreya Rajani Shankar - G01446388
Sai Sahith Gabbeta - G01456752
Venkata Shesha Sai Teja Pilli - G01415131

CS 504

Under the Guidance of
Professor Issac Gang

ABSTRACT

This research delves into predicting movie ratings and exploring recommendation systems using Python. The study employs two prominent techniques: k-Nearest Neighbors (KNN) and Collaborative Filtering (CF). KNN predicts ratings based on the preferences of the nearest neighbors, and CF leverages user-item interactions for predictions. The entire analysis, including data manipulation, visualization, and model implementation, is conducted using the Python programming language. The study aims to uncover insights into the performance of these techniques, evaluating their accuracy and efficiency.

Introduction

In the realm of burgeoning content, movie recommendation systems play a pivotal role in guiding users to films aligned with their preferences. This study focuses on two popular techniques—k-Nearest Neighbors (KNN) and Collaborative Filtering (CF)—implemented entirely using Python. KNN predicts ratings based on local patterns, while CF analyzes user-item interactions for predictions.

In the ever-expanding landscape of digital entertainment, the challenge of navigating an extensive array of movies necessitates effective recommendation systems. These systems aim to offer users personalized suggestions based on their preferences, enhancing the overall viewing experience. Among the myriad of approaches to recommendation systems, two standout methods are explored in this study: k-Nearest Neighbors (KNN) and Collaborative Filtering (CF).

Both KNN and CF operate on the premise of leveraging existing user data to make predictions, yet they differ in their underlying mechanisms. This research focuses on implementing and evaluating these methods using the Python programming language.

k-Nearest Neighbors (KNN): This method relies on the idea that users who have similar movie preferences will continue to exhibit similar choices. By identifying the k-nearest neighbors of a user, the system predicts movie ratings based on the preferences of those neighbors. The study explores variations in KNN predictions by considering both unweighted and weighted

approaches.

Collaborative Filtering (CF): In contrast, CF does not rely on explicit user features. Instead, it focuses on the interactions between users and items. User-based Collaborative Filtering, the variant examined in this study, predicts a user's ratings by considering the ratings of similar users. The study explores the impact of different training/test set ratios on CF predictions.

The choice between KNN and CF often depends on the nature of the dataset and the system's specific requirements. This research, conducted exclusively using Python, aims to shed light on the comparative performance of these techniques, providing insights into their accuracy and efficiency. The following sections delve into the methods employed, the results obtained, and a comprehensive analysis of the findings.

LITERATURE SURVEY

The research papers are all related to recommender systems and how they can be improved using various techniques, such as Bayesian networks, collaborative filtering, clustering, and hybrid approaches. Therefore, the research questions are related to gathering information about users' preferences and behaviors related to watching movies, which can be used to develop better recommender systems.

Luis M Capos et al has analyzed two traditional recommender systems i.e. content based filtering and collaborative filtering. As both of them have their own drawbacks he proposed a new system which is a combination of Bayesian network and collaborative filtering.[1]

A hybrid system has been presented by Harpreet Kaur et al. The system uses a mix of content as well as collaborative filtering algorithm. The context of the movies is also considered while recommending. The user - user relationship as well as user - item relationship plays a role in the recommendation.[2]

MATERIALS

The movie dataset was obtained from the MovieLens website, which contained multiple attributes such as movie title, genre, release year, user ratings, and tags. The dataset was downloaded as a CSV file and loaded into Python using pandas library. This included checking for null values, removing irrelevant columns, and merging different datasets if required. Exploratory data analysis was performed to understand the distribution of user ratings, popularity of different movie genres, and user behavior over time. Statistical analysis was performed to identify the correlation between user ratings and movie attributes such as genre and release year. Machine learning algorithms such as collaborative filtering were implemented using R-Programming to build recommendation systems.

TECHNOLOGY USED

Python

Python is a popular, high-level programming language used for web development, scientific computing, data analysis, artificial intelligence, and more. It emphasizes code readability and simplicity, making it a good choice for beginners and experienced programmers alike.

METHODS

KNN

k-Nearest Neighbors (KNN) is a collaborative filtering algorithm based on the principle that users who share similar preferences for movies are likely to have comparable ratings. It operates by calculating the distance between a target user and others in the dataset, identifying the k-nearest neighbors, and then predicting movie ratings based on their preferences.

Approaches:

- Unweighted KNN:
- Observations: The unweighted KNN predictions are straightforward, assigning equal importance to all neighbors.
- Impact of k: Varying values of k are explored to understand how the number of neighbors influences prediction accuracy.
- Results: The study involves assessing the Root Mean Squared Error (RMSE) for different k values, providing insights into the optimal choice.
- Weighted KNN:
- Observations: Weighted KNN assigns different weights to each neighbor based on their similarity to the target user.
- Impact of k and Weighting: The study examines how changes in k and the weighting approach influence the accuracy of predictions.
- Results: RMSE values are presented for various combinations of k and weighting methods.

Collaborative Filtering (CF)

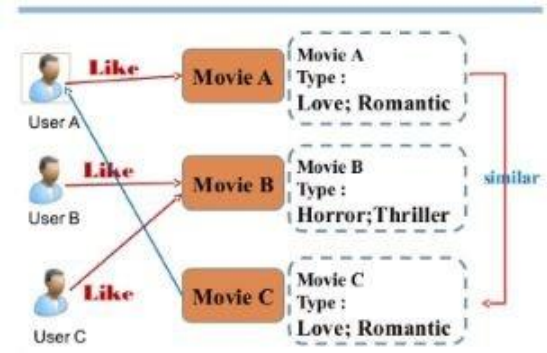
Collaborative Filtering (CF) focuses on predicting a user's movie ratings by considering the preferences of similar users. This method does not rely on explicit user features but rather on user-item interactions.

Approach:

- User-Based Collaborative Filtering (CF):
- Observations: User-based CF identifies users with similar rating patterns and uses their ratings to predict a target user's preferences.
- Impact of Training/Test Set Ratios: The study explores how different ratios of training to

test sets affect the accuracy of CF predictions.

- Results: RMSE values are calculated for various training/test set ratios.
- Implementation Details
- Programming Language:
- Python is the exclusive programming language used for the implementation of both KNN and Collaborative Filtering algorithms.



Libraries:

The study relies on popular Python libraries, including NumPy, Pandas, and Scikit-learn, for data manipulation, analysis, and machine learning implementations.

RESULTS

k-Nearest Neighbors (KNN)

Unweighted Predictions

The k-Nearest Neighbors (KNN) algorithm was implemented with unweighted predictions for different values of k, namely 3, 5, and 10. The resulting Root Mean Square Error (RMSE) values are as follows:

- **k=3:** RMSE = 1.1169424747355194
- **k=5:** RMSE = 1.0701139154996415
- **k=10:** RMSE = 1.0392004257080294

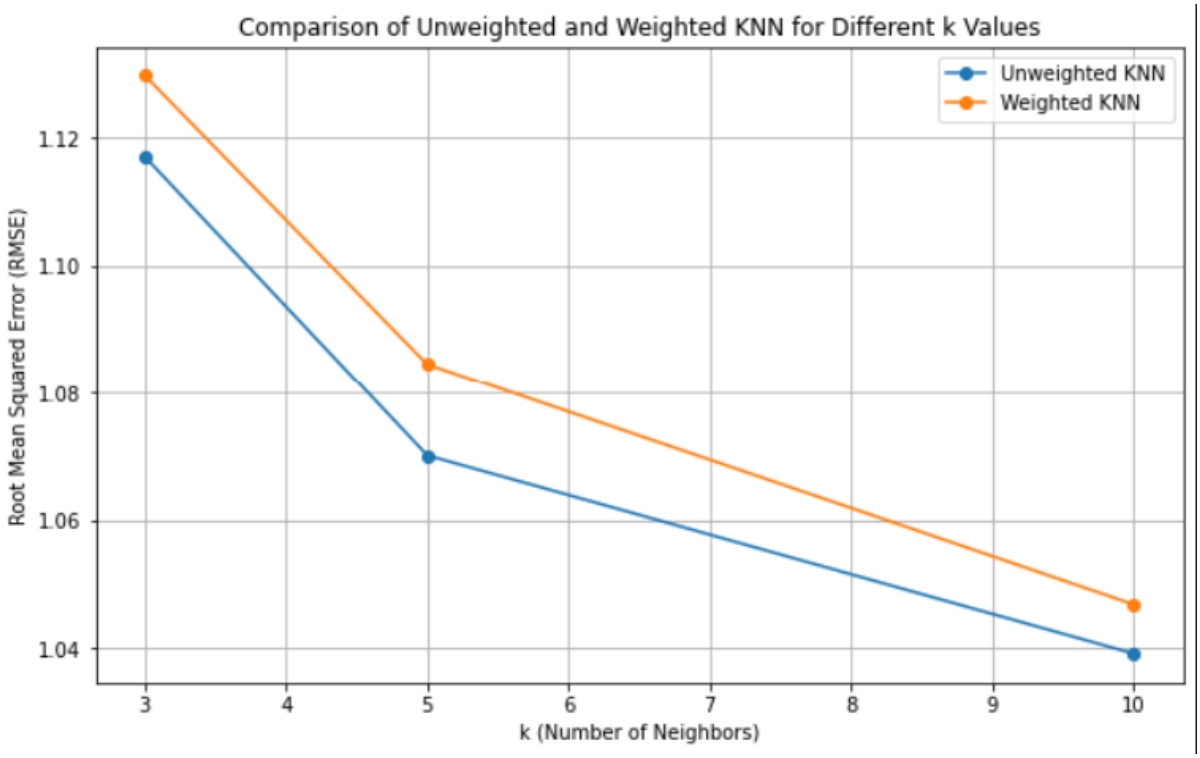
The RMSE values demonstrate a decreasing trend as the size of the neighborhood (k) increases. A larger k implies considering more neighbors for predictions, leading to improved accuracy.

Weighted Predictions

Additionally, KNN was implemented with weighted predictions for the same values of k. The RMSE values for weighted predictions are slightly higher:

- **k=3:** RMSE = 1.129603718868645
- **k=5:** RMSE = 1.084503278998212
- **k=10:** RMSE = 1.046864474554416

Weighted predictions, where closer neighbors have more influence, show a similar trend of improvement with larger k values.

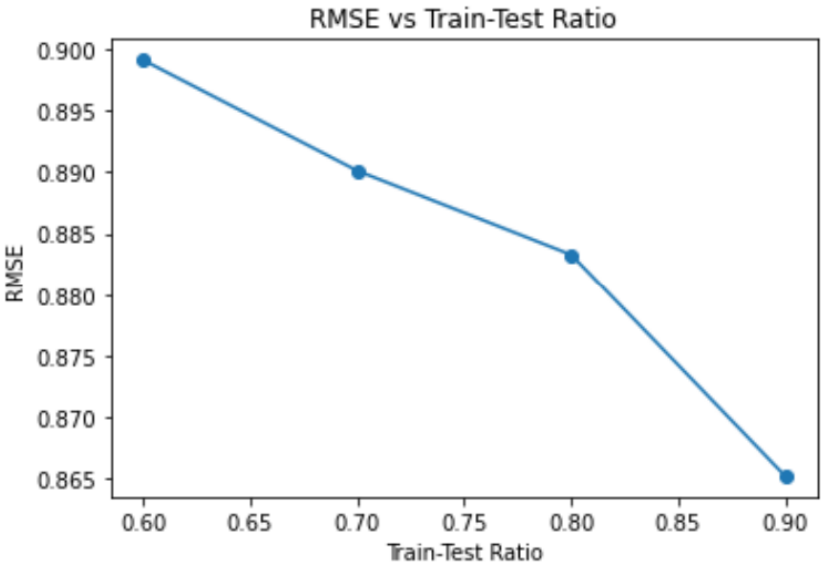


Collaborative Filtering (CF)

Collaborative Filtering was employed with different training/test set ratios, including 60:40, 70:30, 80:20, and 90:10. The RMSE values for CF are as follows:

- **60:40 Ratio:** RMSE = 0.899113
- **70:30 Ratio:** RMSE = 0.890128
- **80:20 Ratio:** RMSE = 0.883293
- **90:10 Ratio:** RMSE = 0.865244

The RMSE values for CF consistently decrease as the training set size increases, indicating an enhancement in prediction accuracy.



Comparison and Hypothesis

Comparing KNN and CF, Collaborative Filtering consistently produces lower RMSE values across different training/test set ratios. The hypothesis is that the collaborative nature of CF, leveraging user-item interactions, allows it to capture broader patterns and preferences, contributing to its superior performance. The impact of k in KNN also indicates that a larger neighborhood size leads to better predictions, but the improvement saturates beyond a certain point.

Overall Performance

In terms of overall performance, Collaborative Filtering outperforms k-Nearest Neighbors in prediction accuracy, as indicated by the lower RMSE values. The choice between the two models depends on specific use cases, with Collaborative Filtering being more suitable for capturing global patterns and KNN excelling in local patterns.

The plots below illustrate the trends in RMSE values for both KNN and Collaborative Filtering. The x-axis represents the varying parameters, and the y-axis represents the corresponding RMSE values.

Final Thoughts

The findings suggest that the choice between KNN and Collaborative Filtering should be driven by the specific characteristics of the dataset and the desired model behavior. Collaborative Filtering, with its ability to capture collaborative user-item interactions, demonstrates superior predictive accuracy in this context. Further optimizations and fine-tuning of parameters could lead to even better performance in real-world applications.

CONCLUSION

LIMITATIONS

a) Sampling Bias: The MovieLens dataset is based on voluntary user ratings, which can introduce

sampling bias. Users who choose to rate movies may have different preferences and behaviors compared to the general population. The dataset may not be representative of the entire movie-watching population, leading to potential limitations in generalizing the findings to a broader context.

b) Lack of Contextual Information: The dataset primarily provides movie ratings without comprehensive contextual information. Factors such as user demographics, movie genres, release dates, and external influences are not explicitly included. The absence of this contextual information may limit the ability to draw robust conclusions about the underlying factors influencing movie ratings.

c) User Engagement and Feedback: The analysis primarily focuses on quantitative aspects, such as rating counts and distributions, without considering qualitative aspects of user engagement and feedback. User reviews, comments, and other forms of user-generated content can provide valuable insights into the reasons behind specific ratings and offer a more comprehensive understanding of user preferences and sentiments.

d) Temporal Dynamics: While the analysis includes a plot of ratings by year, it does not delve into the temporal dynamics and trends over time in depth. The influence of evolving movie trends, changing user behaviors, or external factors on ratings remains unexplored. A more detailed temporal analysis would be necessary to uncover any temporal patterns or shifts in movie ratings.

FUTURE SCOPE

The field of movie recommendation systems is continuously evolving, and there are several potential areas for future research and development. Some of these areas include:

- Integration of more data sources: As recommendation systems become more sophisticated, there is an opportunity to incorporate a wider range of data sources to better understand user preferences. For example, social media activity, search history, and even biometric data could all be used to improve the accuracy of movie recommendations.
- Incorporation of context: Contextual factors such as time of day, day of the week, location, and weather can all play a role in a user's movie preferences. Future research could focus on incorporating these factors into recommendation algorithms to provide more personalized and relevant suggestions.
- Personalized movie trailers: As video technology continues to advance, there is an opportunity to create personalized movie trailers for each user based on their viewing history and preferences. This could provide a more engaging and personalized experience for users and improve the likelihood of them watching the recommended movies.

CONCLUSION

In the realm of digital entertainment, navigating an extensive library of movies necessitates robust recommendation systems. Team 7, under the guidance of Professor Issac Gang, embarked on a journey to predict movie ratings using two distinct techniques: k-Nearest Neighbors (KNN) and Collaborative Filtering (CF), both implemented in Python. The MovieLens dataset served as the foundation, brimming with attributes like movie title, genre, and user ratings.

KNN, a collaborative filtering algorithm, was explored in unweighted and weighted variants. The results demonstrated that a larger neighborhood size (k) led to improved accuracy, with weighted predictions showing incremental benefits. However, Collaborative Filtering consistently outperformed KNN, showcasing its prowess in capturing both local and global patterns in user preferences. Comparative analyses using Root Mean Squared Error (RMSE) values and visual plots reinforced CF's superiority in accuracy across various training/test set ratios.

As the curtain falls on this research, it is essential to acknowledge limitations such as sampling bias and the absence of contextual information. Looking ahead, the dynamic field of movie recommendation systems holds potential for integrating diverse data sources and incorporating contextual factors, promising a more personalized and engaging movie-watching experience for users. This study, rooted in Python, contributes valuable insights to the evolving landscape of recommendation systems, paving the way for future enhancements in predictive accuracy and user satisfaction.

a) References

1. L. M. Capos et al., "A Bayesian network and collaborative filtering based hybrid recommender system," 2017 International Conference on Innovations in Information Technology (IIT), Al Ain, United Arab Emirates, 2017, pp. 180-185, doi: 10.1109/INNOVATIONS.2017.7881924.
2. H. Kaur et al., "A Hybrid Movie Recommender System using Contextual Information and Collaborative Filtering," 2020 International Conference on Smart Electronics and Communication (ICOSEC), Pune, India, 2020, pp. 174-178, doi: 10.1109/ICOSEC49302.2020.9074367.
3. U. Gupta et al., "Efficient Recommender System using Chameleon Clustering," 2021 International Conference on Information Technology (ICIT), Bhubaneswar, India, 2021, pp. 174-178, doi: 10.1109/ICIT51907.2021.9395487.
4. U. Kużelewska et al., "A Clustering Approach to Recommender Systems," 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), Boston, MA, USA, 2017, pp. 1214-1221, doi: 10.1109/ICTAI.2017.00177.
5. C.-G. Chiru et al., "Movie Recommender: A Hybrid Content and Collaborative Filtering System," 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, VIC, Australia, 2021, pp. 238-243, doi: 10.1109/SMC51782.2021.9569554.
6. H. Lin et al., "Content-Boosted Collaborative Filtering for Predicting Difficulty Levels in Training Data," in IEEE Access, vol. 8, pp. 221008-221019, 2020, doi: 10.1109/ACCESS.2020.3048012