

A Movie Recommendations: A Collaborative Filtering Approach Implemented in Python

Nor Syazana Abdul Kodit¹, Tajul Rosli Razak^{1*}, Mohammad Hafiz Ismail³,
Shakirah Hashim⁴, Tengku Zatul Hidayah Tengku Petra⁵, Nur Farraliza Mansor⁶

^{2,4,5,6}*School of Computing Sciences, College of Computing, Informatics and Mathematics, Universiti Teknologi MARA, Shah Alam, Selangor.*

^{1,3}*School of Computing Sciences, College of Computing, Informatics and Mathematics, Universiti Teknologi MARA Perlis Branch, Arau Campus, Arau, Perlis.*

ARTICLE INFO	ABSTRACT
<p><i>Article history:</i> Received: 31 January 2024 Revised: 27 February 2024 Accepted: 27 February 2024 Online first: 1 March 2024 Published 1 March 2024</p>	<p>In-home entertainment, selecting the perfect movie is a pervasive challenge, amplified by many streaming platforms like Netflix and Amazon. This study introduces a groundbreaking Movie Recommendation System with Collaborative Filtering (MRS-CF), meticulously implemented in Python. Employing Item-Based Collaborative Filtering with Cosine Similarity, the system assesses inter-movie relationships based on user-submitted titles, explicitly focusing on genre distinctions. The core contribution of MRS-CF lies in its ability to expedite the movie selection process, swiftly presenting users with a curated list of ten recommended movies strategically organised by descending similarity. Augmented with individual similarity scores, this system is crafted to optimise the user's movie-watching experience. Thirty participants were evaluated through the Perceived Ease of Use (PEOU). The PEOU results underscore the profound contribution of MRS-CF, revealing elevated user satisfaction across all dimensions. This research illuminates the potent impact of the MRS-CF, emphasising its role as a transformative tool for refining and enhancing personalised movie recommendations.</p>
<p><i>Keywords:</i> Collaborative Filtering Recommendation System Movie Selection Cosine Similarity</p> <p><i>DOI:</i> 10.24191/jcrinn.v9i1.428</p>	

1. INTRODUCTION

The World Wide Web has witnessed substantial growth, expanding both in the vastness of its knowledge space and the number of users it attracts. Consequently, navigating this expansive realm to locate pertinent information and effectively managing the abundance of data for informed decision-making has become increasingly challenging (Walek & Fojtik, 2020).

The surge in information volume has prompted a heightened focus on predictive filtering, refinement, and the coordinated delivery of pertinent content to users. Predictive methods have been employed to

^{1*} Corresponding author. *E-mail address:* tajulrosli@uitm.edu.my
<https://doi.org/10.24191/jcrinn.v9i1>

discern and gather information based on users' preferences. Moreover, in recent years, there has been growing interest in ranking and organising information according to individual user interests (Wu et al., 2018).

Even though there were plenty of movie recommendation web-based systems available, like Netflix, YouTube and so on, this movie recommendation system cannot provide movies based on the user interest in the current time as it always recommends movies based on the users' watching history (Husin et al., 2023). Also, the existing collaborative filtering used in the movie recommender system could only manage a small amount of data to provide detailed recommendations to the user (Parthasarathy & Sathiya Devi, 2023). This led to a problem where the system always recommends the same movie according to the user's watching history. The current technique cannot handle the number of movies available as the current collaborative filtering cannot cope with the existence of the sparse matrix.

This paper proposed a Python-based system that recommended movies to users according to the users' preferences, movie similarities, and ranking using the Collaborative Filtering and cosine similarity technique. This was because Collaborative Filtering was a recommendation tool that collected information on the search movie titles from related movies to determine the movies that were similar to it in terms of genre. The fundamental principle of Collaborative Filtering was that the movie title the user entered and submitted was the same as the user preferred.

2. BACKGROUND

This section briefly provides background on recommendation systems, collaborative filtering, user similarity computation, rating prediction and Python.

2.1 Recommendation System

A recommendation system was an information filtering system that sorted items based on the similarity of items and user preferences. The first work on recommendation systems came in the mid-1990s. It has become a vital study topic (Shani & Gunawardana, 2011). Over the last decade, commerce and research have collaborated to create novel ways to enhance the quality of recommender systems (Gogna & Majumdar, 2015). Due to overflowing issues, the recommendation system has become an integral feature of e-commerce and social platforms in the latest generations. In an era of big data, recommendation systems have evolved to help users find exciting items based on interests or preferences (Parthasarathy & Sathiya Devi, 2023).

Besides that, recommender systems analyse detailed data to forecast user preferences for different components (Razak et al., 2019). Following that, based on these findings, Lu et al. (2015) provided predictions and adjusted the contents of the presented webpage as much as feasible to match the user's interest and request together with the similarity of the items that the user rated or liked. It was one of the justifications for why various businesses and web applications had recently introduced systems that analysed user behaviour and interest in certain things and items to recommend the best service, solutions, or content. The goal was, of course, to enhance these corporations' revenues and profits. It was used in various fields, including online education, e-commerce, etc. YouTube, Amazon.com, Netflix, Facebook, and others were such services. Nevertheless, according to Walek and Fojtik, (2020), it was primarily concerned with broadcasting services and online digital movies. Amazon and Netflix were two of the most renowned adopters of recommender systems.

2.2 Collaborative Filtering

Another primary approach in the Recommendation System was the Collaborative Filtering algorithm. It collected and analysed enormous amounts of information on all users' behaviours and preferences and then predicted the items where users would like similar items to other similar users (Guo et al., 2017). Collaborative Filtering (CF) forecasts a user's rating for web applications by considering other users' evaluations of the products (Yao et al., 2014). International and domestic researchers and specialists have conducted an in-depth study on Collaborative Filtering suggestions; thus, Collaborative Filtering recommendation algorithms were numerous. Each Collaborative Filtering recommendation algorithm has a unique working concept; in practice, various Collaborative Filtering recommendation algorithms offer distinct advantages, while flaws such as data sparsity, cold boot, poor scalability, and others were evident (Guo et al., 2017). Data sparsity problems could occur when users rank only a few items. Most recommender systems group the ratings of similar users. However, due to a massive lack of resources or acute awareness to evaluate objects, the presented user-item matrix has vacant or undetermined ratings up to 99%, according to (Isinkaye et al., 2015). As a result, recommender systems may make irrational suggestions if no reviews or ratings are provided. The cold start problem also occurs during collaborative filtering, where the term "cold start" comes from the automotive industry. It had difficulties starting when the engine was cold but ran perfectly once it reached the ideal temperature. A similar situation applies to Recommender Systems. A Recommender System does not operate optimally when inadequate data or content is provided (Isinkaye et al., 2015).

2.3 User Similarity Computation

In Collaborative Filtering recommendation systems, user similarity was determined using the widely utilised cosine similarity technique. If a user does not review items, the user's ratings are assigned to 0 (Parthasarathy & Sathiya Devi, 2023). This method would find the similarities between the movie items and suggest the movie is like the valid movie title entered by the user regarding the movie genre and similarity based on the movie from the datasets. Fig. 1 shows how the user similarity between user and item data worked in the system's modules.

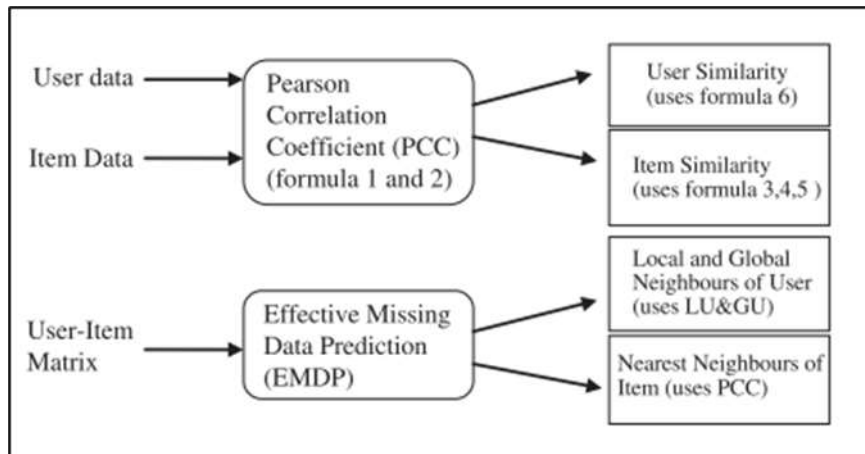


Fig. 1. Modules of the user similarity and item data

Source: Özbal et al. (2011)

2.4 Rating Prediction

The rating prediction was computed on an unrated user item obtained from the active user's neighbours K items. In rating prediction, it wanted to forecast the implicit and interacting information elements simultaneously (Schein et al., 2002). It can be measured by the degree of resemblance between users. Then, based on user similarity values, a set for the first k users near the active user will be generated (Parthasarathy & Sathiya Devi, 2023). Rating prediction was under the Memory-Based Collaborative Filtering method; users store the entire grouping of previously rated things. This data was saved as a user-item matrix (Özbal et al., 2011). This method will recommend movies depending on the dataset's movie rating information.

2.5 Python programming

Python was a powerful programming language frequently used to create websites and applications, automate operations, and analyse data (Blank et al., 2003). Python was a powerful programming language. It could develop a wide range of applications and was not specialised for any particular problem. This adaptability, along with its ease of use for beginners, has established this as one of the most widely used programming languages. Python was frequently used to create the back end of web-based application elements, which the user did not seem to see. Python's function in website development may include transmitting and receiving information from websites, analysing and dealing with datasets, URL navigation, and encryption. Python provides several libraries for website development. Django and Flask are two popular ones.

3. A MOVIE RECOMMENDATION SYSTEM – PYTHON-BASED

This section will explore the methodology and approach for designing the Python-based movie recommendation system. There are four critical steps to developing a movie recommendation system, as described in the following section.

3.1 Step 1: Data Collection – Secondary and Online Survey

This study collected data through the survey and secondary dataset related to movies. The data that was obtained from the study was recorded.

Online Survey

A survey was conducted using Google Forms involving nine questions with two parts of questions. Forty-five respondents responded to the study, with 28 females and 17 males. The survey showed that most respondents think this movie recommendation system was interesting to use to satisfy movie recommendations. Fig. 2, Fig. 3 and Fig. 4 show some of the questions available in the survey.

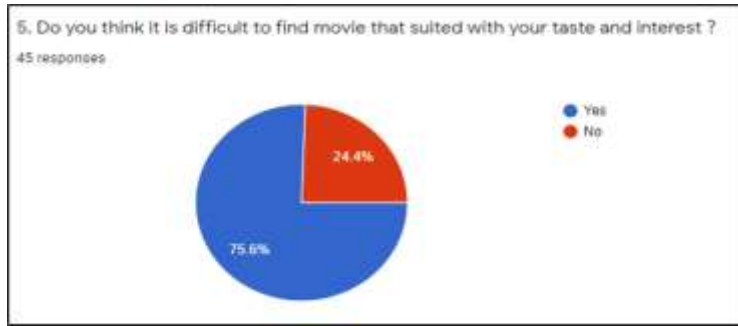


Fig. 2. Question on “Difficult in Finding Movie”

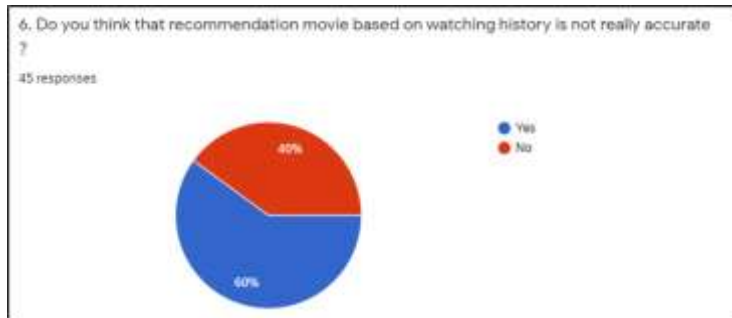


Fig. 3. Question on “Movie Recommendation Not Accurate”.

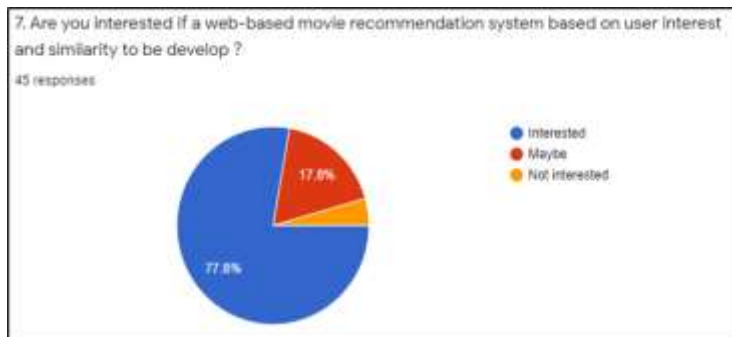


Fig. 4. Question on “Interest in Movie Recommendation System Development”

From the survey, most of the respondents agreed about developing this movie recommendation system with collaborative filtering. This movie recommendation system would be interesting to use.

Secondary Dataset

This study also used the dataset that was available online. The dataset used was known as the MovieLens dataset gathered by the GroupLens Research Project at the University of Minnesota. This data set consisted of 4802 movies with all data related to the movie: movie ID, movie title, genre, and release date. It also provided the average user rating for each movie in the dataset.

3.2 Step 2: Development of A Movie Recommender System

The approaches used in developing the recommender system in this project were Collaborative Filtering, cosine similarity and Item-based recommendation.

Collaborative Filtering

This study used a Collaborative Filtering Algorithm. The system can automatically recommend movies to users based on the movie genre analysed from the movie title that the user had entered. It is worth noting that, even with a collaborative filtering approach, similarities were not discovered using user ages or other information about individuals or products. It was determined strictly based on a user's direct or indirect preferences for an item. For example, two users might be deemed comparable despite a significant age gap if they offer the same genre to 10 movies. With this, an item-based collaborative filtering algorithm was used to ensure the movies' genre could be used to recommend movies with the highest similarity rating value.

Item-Based Recommendation

Item-based Collaborative Filtering has been an excellent method that generates precise recommendations. The primary concept behind this method was to seek things comparable to those that users already had submitted. This study used this approach to compare the movie genre from the movie title the user submitted to locate the identical movie according to the genre by applying it to the Cosine Similarity.

Cosine Similarity

Next, Cosine Similarity was used to determine the degree of similarity between the two elements. The Cosine Similarity provided a value that quantifies the similarity between two components, with -1 being the least similar and one being the most similar. The datasets must be imported into the project environment to identify the similarity. To read the data from the datasets, pandas must be installed to ensure that the data can be accessed using the functions and methods found in the pandas. For instance, the Cosine Similarity function was used to identify the similarity of the movies based on the genre. The code in Fig. 5 demonstrates how to import pandas and use the function to determine which movies were comparable to the needed film based on their genre.

```
import flask
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

Fig. 5. Import pandas and cosine similarity function

The equation of this cosine similarity may be seen below:

$$Cos = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2 \sum_1^n b_i^2}} \quad (1)$$

where a and b are two vectors.

3.3 Step 3: Python GUI Design

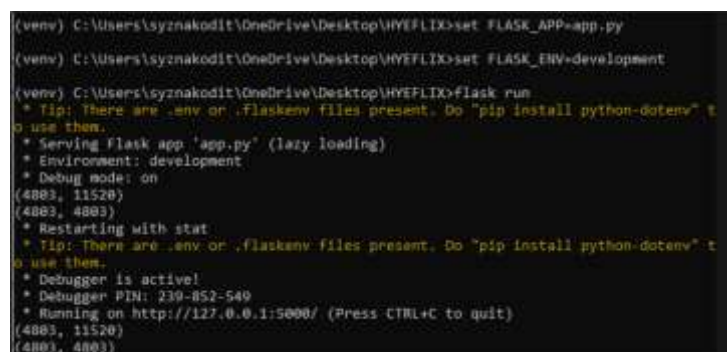
Python has a plethora of frameworks and provides several possibilities for designing a Graphical User Interface (GUI). The most used GUIs in Python were Tkinter, wxPython, and JPython. These GUIs have a toolkit and package that must be imported into the environment. In this project, the GUI was used as a Tkinter.

Tkinter Library

Tkinter is the most widely used of all the GUI techniques. It was a standard Python interface to the Python-supplied Tk GUI toolkit. Python with Tkinter was the quickest and most effortless approach to constructing Graphical User Interface (GUI) applications. Tkinter extends the Tk GUI toolkit with a robust object-oriented interface. This toolkit came with 15 types of widgets.

Flask Framework

This study was developed using the Python programming language, so it can only be viewed and run in the Python environment. The Flask framework was used in this project to enable this system to operate in a web browser. Its environment needed to be activated in the command prompt (CMD) to use the Flask. Fig. 6 shows starting this framework by specifying the project file name. Once activated, this framework provided a web host server to run the system.

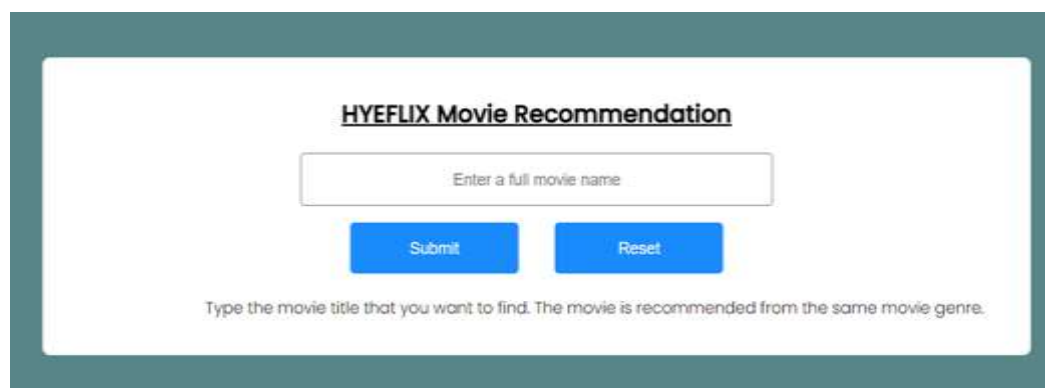


```
(venv) C:\Users\syznakodit\OneDrive\Desktop\HYEFLEX>set FLASK_APP=app.py
(venv) C:\Users\syznakodit\OneDrive\Desktop\HYEFLEX>set FLASK_ENV=development
(venv) C:\Users\syznakodit\OneDrive\Desktop\HYEFLEX>flask run
* Tip: There are .env or .flaskenv files present. Do "pip install python-dotenv" to use them.
* Serving Flask app 'app.py' (lazy loading)
* Environment: development
* Debug mode: on
(4883, 11520)
(4883, 4883)
* Restarting with stat
* Tip: There are .env or .flaskenv files present. Do "pip install python-dotenv" to use them.
* Debugger is active!
* Debugger PIN: 239-852-549
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
(4883, 11520)
(4883, 4883)
```

Fig. 6. Activate flask framework

3.4 Step 4: GUI – A Movie Recommendation System

This section discusses the interface of the main system page. Fig. 7 shows the system's main page with a text box area where users can enter the movie title. It also came with two buttons: the submitted button and the reset button. Users must click the submit button to get the recommendation once they enter the movie title.



HYEFILX Movie Recommendation

Enter a full movie name

Submit Reset

Type the movie title that you want to find. The movie is recommended from the same movie genre.

Fig. 7 Main Page

Once users click the submit button, it will redirect them to the recommendation page, listing ten similar movies to the movie title offered by the users. The list of recommended movies was in descending order from the most similar movie to the least equal movie. It would also display the movie releases, as shown in Fig. 8. On the other hand, Fig. 9 shows the recommended similarity score of the movie, where value 1 indicates the most similarity. In contrast, value -1 indicated the least similar movie. This similarity score was displayed in the running Flask framework in the CMD. However, as this system uses the dataset data, the users need to submit a valid movie title just like in the dataset. Suppose the users offered an invalid movie title. In that case, the system will display some title suggestions of the user's submitted title, as shown in Fig. 10. A go-back-to-the-homepage button was available where users could try again and present a valid movie title.



Movie Recommendations for "Shrek"

Movie Title	Released Date
Shrek 2	2004-05-19
Shrek Forever After	2010-05-16
Shrek the Third	2007-05-17
Spy Kids 3-D: Game Over	2003-07-25
The Haunted Mansion	2003-08-25
The Chronicles of Narnia: The Lion, the Witch and the Wardrobe	2005-12-07
The Chronicles of Narnia: Prince Caspian	2008-05-15
Austin Powers: The Spy Who Shagged Me	1999-06-08
Nowhere to Run	1993-01-15
Austin Powers in Goldmember	2002-07-25

Fig. 8. List of movies recommended page


```

movieId      score
(565, 0.856522622498297)
(86, 0.3850094231204073)
(106, 0.37821799012694557)
(1155, 0.18976837804524288)
(364, 0.18241315903591254)
(63, 0.17401306319570845)
(15, 0.169671088896925764)
(1358, 0.16724526162212808)
(2573, 0.1629795132842811)
(673, 0.14711643201670277)
127.0.0.1 - - [01/Feb/2022 00:15:38] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [01/Feb/2022 00:16:34] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [01/Feb/2022 00:16:52] "POST / HTTP/1.1" 200 -

```

Fig. 9. Similarity score



Fig. 10. Valid movie title suggestion

4. EVALUATION OF A MOVIE RECOMMENDATION

To validate the proposed movie recommendation system, we have conducted a Perceived Ease of Use (PEOU) test on Thirty respondents from different backgrounds. The PEOU indicates the degree to which the MRS-CF is considered relatively simple to use and understand. Please note that this testing was conducted online through Google Forms. All 30 respondents must watch the MRS-CF demonstration video. Then, they need to answer the question in the Google Form provided.

4.1 Perceived Ease of Use (PEOU)

Five questions were available as in Table 1 for the Perceived Ease of Use (PEOU) section used to measure the degree to which this MRS-CF is considered relatively easy to use and understand. Here are the results obtained from the respondents and the graph for each available question, as in Fig. 11. Note that the item in Fig. 11 refers to the question.

Table 1. Five questions for PEOU.

Questions	Scale					Mean Score
	1	2	3	4	5	
I think using the HYEFLIX: Movie Recommendation System with Collaborative Filtering is easy.			1	5	24	4.8
I think using the HYEFLIX: Movie Recommendation System with Collaborative Filtering is easy.and understandable.			1	6	23	4.7
I believe interacting with HYEFLIX: Movie Recommendation System with Collaborative Filtering will not require much mental effort to use it.				4	26	4.9
I assume I can use HYEFLIX: Movie Recommendation System with Collaborative Filtering independently.				7	23	4.8
I believe HYEFLIX: Movie Recommendation System with Collaborative Filtering will be easy to use.			1	6	2. 3	4.7
Overall score						4.8

Here are the results obtained from the respondents and the graph for each available question, as in Fig. 11. Note that the item in Fig. 11 refers to the question.



Fig. 11. Mean score for Perceived Ease of Use

The chart in Fig. 11 depicts the mean score for the Perceived Ease of Use assessment criterion. The findings indicated that most respondents thought the MRS-CF was simple. The mean score may

demonstrate it for each question, more significant than 4.0 but less than the agreed-upon scale. The respondents believe the Collaborative Filtering and Recommendation System for Movies is simple to use and explore. The mean score was 4.8.

Additionally, respondents found that the interactions between the Movie Recommendation System and Collaborative Filtering were evident and intelligible, with a mean score of 4.7. Utilising the system took less mental effort, with a mean score of 4.9. With a mean score of 4.8, the system enables them to operate it independently. Most respondents believed this method was simple, with a mean score of 4.7 indicating its applicability. Thus, the overall findings of Perceived Ease of Use suggested that most participants believed the MRS-CF was simple since it did not require much mental effort.

5. CONCLUSION

In conclusion, we introduced a Python-based movie recommendation system. Despite receiving positive feedback from the testing results, respondents provided valuable suggestions and recommendations for enhancing the system. This highlights that although the Movie Recommendation System utilising Collaborative Filtering is functional, there is room for improvement in various features for future use.

For future enhancements, we plan to leverage a larger dataset to elevate the movie recommendation system to a higher standard of excellence and perfection. By incorporating a more extensive and diverse data collection, we aim to enhance the functionality and precision of our movie recommendation algorithm, providing users with even more accurate and personalised suggestions.

6. ACKNOWLEDGEMENTS/FUNDING

The authors would like to acknowledge the support of Universiti Teknologi Mara (UiTM), Shah Alam, Selangor, Malaysia, for providing the facilities and financial support for this research. Also, the authors thank the reviewers for their thoughtful comments and efforts towards improving our manuscript.

7. CONFLICT OF INTEREST STATEMENT

The authors declared that they have no conflicts of interest to disclose.

8. REFERENCES

- Blank, D., Kumar, D., Meeden, L., & Yanco, H. (2003). Pyro: A Python-based versatile programming environment for teaching robotics. *Journal on Educational Resources in Computing (JERIC)*, 3(4), 1-es.
- Gogna, A., & Majumdar, A. (2015). A comprehensive recommender system model: Improving accuracy for warm and cold start users. *IEEE Access*, 3, 2803–2813.
- Guo, B., Xu, S., Liu, D., Niu, L., Tan, F., & Zhang, Y. (2017). Collaborative filtering recommendation model with user similarity filling. *2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC)*, 1151–1154.
- Husin, M. R. M., Razak, T. R., Malik, A. M. A., Nordin, S., & Abdul-Rahman, S. (2023). Hybrid collaborative movie recommendation system. In *2023 4th International Conference on Artificial Intelligence and Data Sciences: Discovering Technological Advancement in Artificial Intelligence and Data Science, AiDAS 2023 – Proceedings* (pp. 274–280). <https://doi.org/10.1109/AIDAS60501.2023.10284679>

- Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3), 261–273.
- Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: a survey. *Decision Support Systems*, 74, 12–32.
- Özbal, G., Karaman, H., & Alpaslan, F. N. (2011). A content-boosted collaborative filtering approach for movie recommendation based on local and global similarity and missing data prediction. *The Computer Journal*, 54(9), 1535–1546.
- Parthasarathy, G., & Sathiya Devi, S. (2023). Hybrid recommendation system based on collaborative and content-based filtering. *Cybernetics and Systems*, 54(4), 432–453.
- Razak, T. R., Halim, I. H. A., Jamaludin, M. N. F., Ismail, M. H., & Fauzi, S. S. M. (2019). an exploratory study of hierarchical fuzzy systems approach in a recommendation system. *Jurnal Intelek*, 14(2), 174–186. <https://doi.org/10.24191/JI.V14I2.233>
- Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002). DNA extraction from plant leaves with Minilys. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval-SIGIR* (Vol 2, pp. 253–260).
- Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. *Recommender Systems Handbook*, 257–297.
- Walek, B., & Fojtik, V. (2020). A hybrid recommender system for recommending relevant movies using an expert system. *Expert Systems with Applications*, 158, 113452. <https://doi.org/10.1016/J.ESWA.2020.113452>
- Wu, C.-S. M., Garg, D., & Bhandary, U. (2018). Movie recommendation system using collaborative filtering. *IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)* (pp. 11–15). IEEE Xplore. <https://10.1109/ICSESS.2018.8663822>
- Yao, L., Sheng, Q. Z., Ngu, A. H. H., Yu, J., & Segev, A. (2014). Unified collaborative and content-based web service recommendation. *IEEE Transactions on Services Computing*, 8(3), 453–466.



© 2024 by the authors. Submitted for open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).