AI Enhanced Recommender System for Movies

Project Proposal
To
Prof. Rong Jin

Submitted by

Shayan Darian

02/02/2025

CPSC 597
California State University Fullerton

CALIFORNIA STATE UNIVERSITY
FULLERTON™

Department of Computer Science

CPSC 597 / 598 PROJECT / THESIS DEFINITION

To the graduate student:
1. Complete a project proposal, following the department guidelines.
2. Have this form signed by your advisor and reviewer / committee.
3. Submit it with the proposal attached, to the Department of Computer Science.

[X] Project

[ ] Thesis

Please print or type.

Student Name: Shayan Darian Student ID: 885135772

Address: 180 Tribeca (Street) Irvine (City) 92612 (Zip Code)

Home Phone: (949) 606-3352 Work Phone: (949) 606-3352

E-Mail: skdarian@csu.fullerton.edu Units: 3 Semester: Spring

Are you a Classified graduate student?

[X] Yes

[ ] No

Is this a group project?

[ ] Yes

[X] No

Proposal Date: 02/02/25

Tentative Date for Demonstration /Presentation/Oral Defense: _____

Completion Deadline: _____

Tentative Title: AI Enhanced Recommender System for Movies

We recommend that this proposal be approved:

Faculty Advisor _____
Printed name Signature Date

Faculty Reviewer _____
Printed name Signature Date

Faculty Reviewer _____
Printed name Signature Date

Table of Contents

1.	**Introduction**	5
1.1	**Problem Domain and Background**	5
1.2	**Comparison with Traditional and Current Methods**	5
1.3	**Key Issues**	6
2.	**Project Objectives**	10
3.	**Project Activities**	11
4.	**Software Requirement Specifications**	13
4.1	**Functional Requirements**	13
4.2	**Non-Functional Requirements**	14
4.3	**Constraints**	14
4.4	**Assumptions**	15
5.	**Project Environment**	15
6.	**Project Results**	16
7.	**Project Schedule**	18
8.	**References**	20

List of figures

1.2	**YouTube's Cold Start Problem**	**6**
1.3	**Collaborative-Based Filtering**	**8**
1.3	**Content-Based Filtering**	**9**
4.1	**Deep Neural Network Integration**	**14**

Introduction

Problem domain and background

As companies and businesses attempt to further increase and maximize profits, they have sought to develop a system that is able to directly recommend products and services to their customers. Systems such as these are referred to as Recommender Systems. They are built and designed with the goal of allowing the company or business with whose service the customer is using, to suggest new and additional products and items to the customer, which they will then likely be interested in purchasing or viewing, thereby improving and enhancing the user experience for the customer or user. Though the use of recommender systems has expanded beyond the use of commercial enterprise, and has practical applications for governments, universities, and other organizations. The goal of a recommender system is to solve a major problem, which is that of information overload for the user, which will then make the decision process easier and simpler for them, and will therefore enhance the user experience.

The first recommender systems came into existence a little more than 20 years ago, and were created through methods and techniques outside of the field of artificial intelligence (AI), especially in areas such as user profiling and preference discovery.

Over the last decade, we have seen a lot of progress and success made in successful AI-driven applications. Some examples of this include Deepmind's AlphaGo, the AI-driven program that was able to win the game of 'Go' when it played against a professional human player, in addition to the self-driving car, as well as others in the fields of computer vision and speech recognition. The advances that we have seen in AI, data analytics, and big data, present to us an amazing opportunity to embrace these impressive advancements in AI, and to see how we can use them to enhance recommender systems.

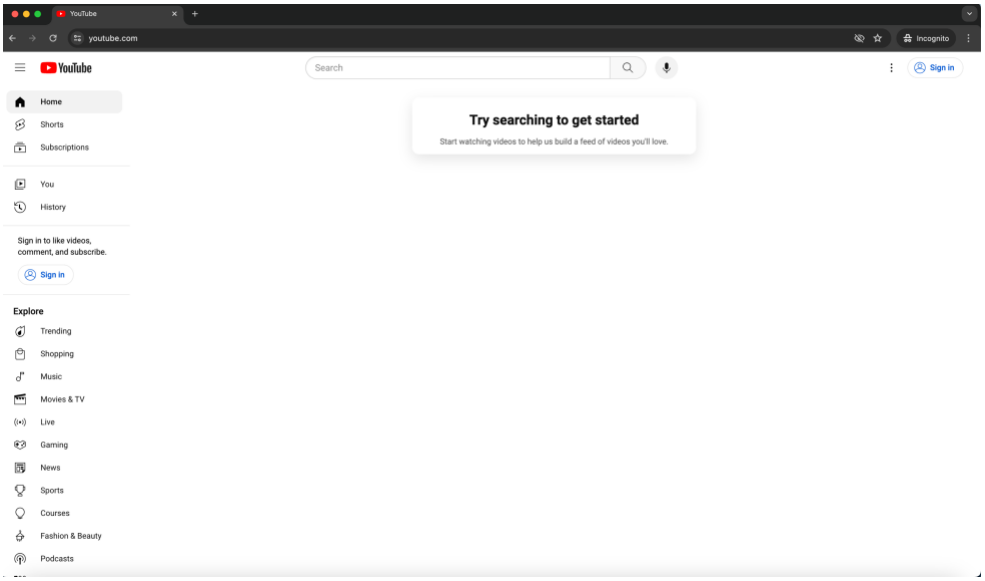
For the purposes of this project, the recommender system that I will be building will be a movie recommender system that will recommend movies to a user, and it will be an improvement over many other current recommender systems due to the integration and enhancement of various AI and machine learning techniques and methodologies.

Comparison with Traditional and Current Methods

Traditionally, the main goal of recommender systems has been to suggest new items to a user based on what they are most likely to want, through attempting to accurately recommend

different items to a user that are similar to the ones that they have interacted with. The recommender system has done this through different methods, but in more recent times with advancements and further integration of AI, this has been done by attempting to learn a user's behavior and then suggesting new items to the user based on predictions from that.

A major flaw of current recommender systems is that they are unable to give recommendations until they have collected enough data about a new user. This data is typically collected through your browsing history or through reviews that you have given. This issue is referred to as the Cold Start problem which will be discussed in greater detail later on in this paper, and ideas will be given on how to properly address and solve this issue. This issue exists for new items as well. For example, at the time of writing this paper, if you go to YouTube without any prior browsing history, instead of receiving any recommended videos to watch, you will get a message that says "Try searching to get started. Start watching videos to help us build a feed of videos you'll love". This is due to the fact that YouTube's recommender system has no data or information about you, and so it doesn't know what it should recommend to you.



youtube.com

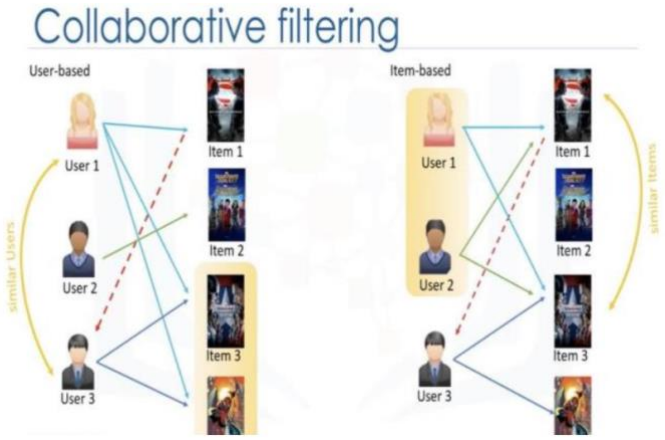
Key Issues

1. **Proper Prediction Algorithm Selection**
    - Predicting the value of items for a particular user varies according the recommendation algorithm selected.

- Knowing which recommendation algorithm to select, is crucial in the accuracy and performance of a recommender system.
- **Solution:** K-Nearest Neighbors (K-NN) Algorithm.
    i. Can be used for both Collaborative-based and Content-based filtering.

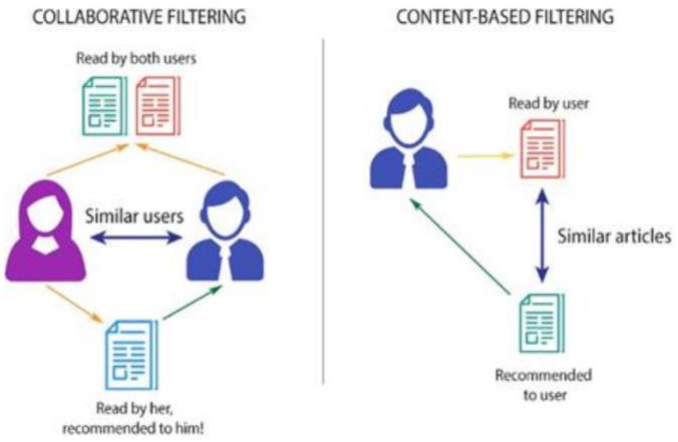
2. **Data Accuracy**
    - A database with few movies will have a higher accuracy in making correct recommendations.
    - A large database with a lot of movies will have a lower accuracy because the pool of information searched is too large.
    - Being able to maintain a high level of accuracy regardless of the size of your database is crucial for a product that wants to ensure a good user experience.
    - **Solution:** K-Nearest Neighbors Algorithm, likely using Cosine similarity measure, or a Deep Neural Network Implementation.
3. **Data Sparsity / Cold Start**
    - **For New User**
        - Not having enough data or information on a particular user, can make it difficult to provide the user with relevant and accurate recommendations.
    - **For New Item**
        - Not having enough reviews or other relevant information on a new item, can make it difficult to determine how to categorize or rank its relevance, or it can make it difficult to determine how to accurately recommend it to the correct demographic of users.
    - **Solution:** Apply Content-Based Filtering to give a new user or new item a cold start, then proceed with Collaborative-Based Filtering.
    - Shown below is a diagram detailing how Collaborative-Based Filtering works. For users, they are matched together based on their measured similarities, and this similarity measure can be determined by them for example watching or giving similar ratings to the same movies. So, for example, user 3 is determined to be similar to user 1 because they have both watched item 3 and item 4 in the list. Since user 1 has additionally watched item 1, the

recommender system can recommend item 1 to user 3, because they are a similar user to user 1. A similar phenomenon is shown for similar items on the right-hand side of the diagram.



Malik, S. (2022)

- Shown below is a diagram detailing how Content-Based Filtering works in comparison to Collaborative-Based Filtering for a new user. Content-Based Filtering will recommend a new user items (in the case of this project those items are movies) that are similar to other movies based on the content features of the movies themselves, such as genre, cast, etc. If basic demographic information about a user is known, such as their age, gender, and other similar information, then they could be movies watched by users of a similar demographic, otherwise they can also be watched by the general userbase population. This can be used to give either a new user or a new movie a cold start. From how a new user interacts with these initial recommendations, the system can start to profile the user and learn their behaviors, and thus learn to recommend them content that is more specific for them, and this can be done through Collaborative-Based Filtering.



Malik, S. (2022)

4. **Overspecialization**
    - Always recommending similar items to a user, which can cause them to become bored or unable to discover new types of items through recommendations.
    - **Solution:** Assign a low probability of sometimes recommending unrelated items to the user, allowing them to occasionally see different types of items in their recommendation list.

Project Objectives

Objectives

1. Build an enhanced recommender system that solves many of the issues that many traditional and modern recommender systems face.
2. Breakthroughs in AI and machine learning during recent years have offered much insight and opportunity into integrating these techniques and methodologies into existing technologies, and recommender systems seem to be an area which could see significant advancements and improvements through this integration.
3. Using the K-NN (K-Nearest Neighbors) algorithm with a similarity measure such as Cosine similarity, will be an effective way to determine the similarity between users, as well as movies. K-NN also yields high data accuracy, solving two key issues with one method.
4. Implementing Collaborative-Based Filtering works well for recommendations when there is already sufficient data and information that has been built for users and movies through their mutual interactions, but this filtering method has the flaw of not being able to suggest any recommendations to new users due to there being no data on the new user yet, and likewise for new movies it will not be able to accurately suggest these movies to users to the lack of data on how users like the movie. This flaw is known as the Cold Start problem. Collaborative-Based Filtering will still be implemented, but only for users and movies that have been given a cold start.
5. Users and movies will be given a cold start through Content-Based Filtering, which will be implemented for new users and new movies that enter the system for which the system has no information or data on.
6. The recommender system will also have a method for dealing with overspecialization, which will sometimes show the user unrelated or different types of movies from the ones normally in their recommendations, allowing them to explore different types of movies that they may end up enjoying, but it will not be so frequent to where it will negatively impact their user experience. It will enhance their user experience by keeping them engaged and preventing them from getting bored with the same or similar recommendations.
7. Optional: If accuracy still remains an issue, experimentation with integrating a Deep

Neural Network into the recommender system may help with further improving the accuracy.

Project Activities

How Objectives will be Achieved

1. Phase 1: Data Collection
    - The first thing that will need to be done is conducting a thorough and comprehensive search in order to find good data sets that can be used for creating a database of users and movies for the recommender system.
    - In order to deal with the cold start problem, the data sets for movies will need to have additional information such as movie genre, and possibly total box office revenue, IMDB ratings, etc.
    - The data sets for users should ideally have some additional information, such as their favorite genres, their gender, age, etc. in order to build an initial idea of what types or categories of movies that they might like.
2. Phase 2: Preprocess and Import the Data
    - Once sufficient data has been collected, it will need to be imported into the program, and then formatted properly to where it can be easily accessed by the data structures of the program itself.
3. Phase 3: Build the Skeleton for the System
    - Once the data has been properly imported, I will work on building the initial skeleton or structure for the recommender system. Key implementations such as the ones discussed below will be implemented afterwards, but at this stage, I will be working on building the structure of the recommender system's program.
4. Phase 4: Implement the Proper Prediction Algorithm, in this case K-NN
    - I will work on building an implementation for the K-NN algorithm, where for users, similar users will be classified based on their similarity measure, and for movies, similar movies will be classified based on their similarity measure. Multiple similarity groupings may be performed if it turns out that grouping users or movies together based on multiple characteristics proves to

be beneficial. This is a straight forward for the Collaborative-Based Filtering approach.

- K-NN can be used as an implementation for a Content-Based Filtering approach as well, though this will likely take longer to implement, and further research on other Content-Based Filtering options will also be explored.

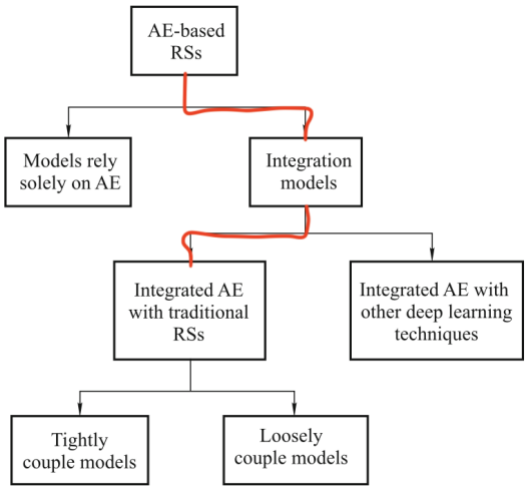
5. Phase 5: Implement Collaborative-Based Filtering
    - For users and movies that already have relationships built through mutual interactions, I will implement Collaborative-Based Filtering into the program, which will be used to actually match users and movies together, i.e. filling a user's recommendation feed with movie recommendations.
6. Phase 6: Implement Content-Based Filtering
    - For new users and new movies that have no interactions with other users and movies in the system, Content-Based Filtering will be implemented to essentially give these new users and new movies a cold start, allowing them to begin interacting and building relationships with older users and older movies in the system, until they become older users and older themselves, at which point the user or movie will no longer need Content-Based Filtering and from this point forward will instead use Collaborative-Based Filtering.
    - In order to determine the age of users and movies, a value can be assigned to each user or movie indicating this, which will gradually increase with the more interactions that they have.
7. Optional: Phase 7: Build a Deep Neural Network
    - If time allows, building a Deep Neural Network, can improve accuracy, especially if the datasets prove to be very large. While a Deep Neural Network implementation can be done for both Content-Based Filtering and Collaborative-Based Filtering, the benefits of a Deep Neural Network implementation would be greater for Collaborative-Based Filtering.
    - The exact specifications of the Neural Network, such as the type of Neural Network, the number of hidden layers, the number of nodes in each layer, and the types of hidden layers, will need to be figured out through experimentation with the datasets and with the recommender system itself.

8. Phase 8: Integrating Everything together & Performance Evaluation
    - I will make sure that every component of the program is working together as intended, and that the model has been generalized well, which would yield strong accuracy results.

Software Requirement Specifications

Functional Requirements

1. Data Collection and Preprocessing
    - The recommender system will support functionality of acquiring and preprocessing diverse data sets including, but not limited to, a list of users, along with a list of movies, and some additional information to help initially categorize the users and movies, such as movie ratings, user age, etc.
2. Algorithm Selection and Implementation
    - The recommender system will support the implementation of machine learning algorithms such as K-NN, and techniques including, but not limited to, Collaborative-Based Filtering, and Content-Based Filtering.
3. Optional: Neural Network Integration
    - If time allows or if accuracy remains to be an issue, the recommender system will include the integration of a Deep Neural Network, which performs well in achieving high levels of accuracy in many AI and machine learning applications, especially when it comes to handling large amounts of data that other systems typically struggle with.
    - Shown below is a diagram of how an Autoencoder Neural Network can be integrated into a recommender system. As shown in the red outline, I would plan to integrate a Deep Neural Network into a traditional recommender system framework, such as for a Collaborative-Based filtering implementation, rather than having the project rely solely on the Neural Network.



Zhang, G., Liu, Y. & Jin, X. (2020)

Non-Functional Requirements

1. Performance
    - The recommender system must be able to efficiently handle datasets of various sizes as input, and be able to preprocess them into a format that will be easily compatible with the data structures that the program will use. The recommender system must be able to process the data, run its program, and produce an accurate output in a reasonable time frame, and it must be able to do this while maintaining computational and memory efficiency.
2. Reliability
    - The recommender system must be able to produce results with high rates of accuracy, and the error rate of the recommender system must be minimal. For any unforeseen errors, the recommender system must be able to handle these types of scenarios, and output an error message indicating if an error has occurred.

Constraints

1. The recommender system will operate and function within the constraints of the hardware and infrastructure that it is built on.
2. The recommender system will be built using popular and well-known programming languages and development environments, ensuring that it will be compatible with

common operating systems, making it widely distributable and accessible.

Assumptions

1. The recommender system will have access to a database of users and movies, from which it will train itself with.
2. Users who test the system will be provided with clear instructions on how to use and navigate it.
3. Users who test the system will have at least a basic knowledge of computers and prior familiarity with recommender systems, such as the ones used by Amazon, YouTube, Netflix, or any major social media application.

Project Environment

Hardware Environment

1. The recommender system will be built on computer hardware that have specifications which meet the requirements needed to implement the system and carry out computational heavy tasks required for a machine learning project of this caliber.
2. Such specifications may include a high-performance computing environment, with the use of Graphic Processing Units (GPUs), which will speed up the computational time of the training and testing processes tremendously. Google Collab or Jupityr Notebook provide a limited use of such services through the use of Google servers remotely.

Software Environment

1. The main programming language that will be used for this project will be Python. Python has many libraries that are well suited for machine learning, such as NumPy, Pandas, TensorFlow, Keras, PyTorch, Scikit-Learn, and others.
2. Python will allow a machine learning project such as a recommender system, to be built and implemented in a smooth and efficient environment.
3. Git will be used to serve as the version control system, which will allow the tracking and management of the source code, and will allow for rollbacks in the code if needed.

Interactive Development Environment (IDE)

1. Google Collab and Jupityr notebook will be the main IDEs which will be used to build the recommender system, and will be especially crucial when it comes to training and testing the system.
2. Visual Studio Code may also be used for debugging purposes, as well as for managing multiple files if multiple files will be needed. If all needed libraries are able to be successfully installed on Visual Studio Code, then it may also serve as a possible alternative to Google Collab or Jupityr notebook, in case of any issues with these services.

Libraries

1. Scikit-Learn, NumPy, and Pandas will be used for data preprocessing, implementing AI algorithms, and/or for performance measurements, due to its vast collection of Python machine learning methods.
2. If a Deep Neural Network implementation is done, it will be done with PyTorch or Keras, as both libraries essentially do the same thing, but go about doing it in different ways. I would plan to use Keras, but I will experiment with both and see which works best for the purposes of this project.
3. Matplotlib will assist in data visualization, and will help in visualizing the properties of the data, which may be crucial in interpreting the outputs of the system.

Project Results

1. Built Recommender System

- Upon completion of this project, a recommender system will have been built, that will be able to give users accurate movie recommendations after having learned the user's behaviors and browsing patterns.

2. Integrated AI techniques and Methodologies

- The recommender system will have been built using machine learning models,

which will be what allows the recommender system to give personalized recommendations to a user in a smart and intelligent way, versus the traditional method of recommending similar items to every user based on popularity metrics alone.

3. **Trained Machine Learning Models**
    - The machine learning models will have been trained on the datasets given to the system as input, which will give the recommender system the ability to output new recommendations with a high level of accuracy, and a minimal level of error.

4. **Evaluation Metrics and Comparative Analysis**
    - Evaluation metrics, with exact specifications being dependent on the dataset used, will be presented.

5. **Documentation and Research Findings**
    - Detailed documentation will be presented, highlighting the research, processes, and procedures that were taken throughout the course of building this project, such as data preprocessing methodologies, algorithm selection, analysis on determining the need of further accuracy improvement through further AI integration such as deep neural networks, and other optimization procedures.
6. **Codebase and Repository**
    - The project's codebase and repository, organized through version control systems such as Git, will be shared. This will allow the examination of the details of the project's implementation, the reproducibility of its results, and the ability of future additions to the project.

7. **Presentation and Demonstration**
    - A live demonstration and presentation will be done, showing the project's functionality, key features, components, and final results of the recommender

system. This will also highlight the impact of AI integration into recommender systems, but also showcase the potential for its integration into other existing technologies.

Project Schedule

Weeks 1-2: Project Initiation and Planning (25 hours)

1. Define Project Scope and Objectives (10 hours)
    a. Clearly define the project requirements and scope of the recommender system, outlining main features and functionalities.
2. Conduct Research and Select Technologies (10 hours)
    a. Conduct any additional research needed on recommender systems and machine learning.
    b. Identify and select the Python libraries needed, and successfully download and run any dependencies needed to successfully import the needed libraries.
3. Finalize Project Proposal (5 hours)
    a. Finalize the project proposal based on the additional research done.

Weeks 3-4: Data Collection and Preprocessing (35 hours)

1. Identify and Acquire Datasets (15 hours)
    a. Conduct a search to find the necessary datasets, ensure that they meet the expectations needed for the purposes of the project.
2. Preprocess the Data (20 hours)
    a. Handle any missing values, reshaping, standardization, and normalization needed on the data in order for it to be ready to serve as input to the program.

Weeks 5-6: Algorithm Selection and Implementation (30 hours)

1. Implement skeleton / program structure (15 hours)
    a. Build the code for the basic structure of the recommender system.
2. Implement Machine Learning Algorithms (15 hours)
    a. Build a rough outline of the K-NN algorithm and any other algorithms needed, to be used for the program. This may include the code for the model

to iteratively train and improve itself, the implementation of optimization methods, and the code for accuracy and loss measurement.

Weeks 7-8: Collaborative-Based Filtering and Content-Based Filtering (30 hours)

1. Implement Collaborative-Based Filtering. (15 hours)
2. Implement Content-Based Filtering. (15 hours)

Weeks 9-12: Integrating and Combining the Program Components (40 hours)

1. Finish writing any structural code needed to ensure that all of the various features and functionalities, such as both filtering methods, are able to operate together within a single program. (20 hours)
2. Conduct multiple runs and tests of the program, and fix any major issues still lingering (20 hours)
3. Optional: If time allows, experiment with a Deep Neural Network implementation of Collaborative-Based Filtering, and see how it compares to the K-NN implementation.

Weeks 13-14: Miscellaneous Function Implementation (30 hours)

1. Implement any functionalities still needed, such as but not limited to, the solution to overspecialization. (15 hours)
2. Optimize and fine tune the rest of the system, ensuring overall high accuracy and minimal loss. (15 hours)

Weeks 15-16: Documentation, Reporting, and Finalization (30 hours)

1. Document Research Findings and Methodology (12 hours)
2. Create Final Project Report and Presentation (10 hours)
3. Prepare for Live Demonstration (8 hours)

Total Estimated Duration: 220 hours

References

1. Zhang, Q., Lu, J. & Jin, Y. Artificial Intelligence in Recommender Systems. *Complex Intell. Syst.* **7**, 439–457. **2021**

2. Pérez-López, D.; Dueñas-Lerín, J.; Ortega, F.; González-Prieto, Á. New Trends in Artificial Intelligence for Recommender Systems and Collaborative Filtering. *Appl. Sci. 13*, 8845. **2023**

3. Tegetmeier, C., Johannssen, A. & Chukhrova, N. Artificial Intelligence Algorithms for Collaborative Book Recommender Systems. *Ann. Data. Sci.* **2023**

4. Bobadilla, J.; Ortega, F.; Gutiérrez, A.; Alonso, S. Classification-based Deep Neural Network Architecture for Collaborative Filtering Recommender Systems. *International Journal of Interactive Multimedia and Artificial Intelligence. Vol. 6*, No 1. **2020**

5. Thomas, B.; John, A. K. Machine Learning Techniques for Recommender Systems – A Comparative Case Analysis, IOP Conference Series: Materials Science and Engineering 1085. **2021**

6. Varma, A., Chaduvula, A., Krishna, N. Movie Recommender System. **2021**

7. Zhang, G., Liu, Y. & Jin, X. A survey of autoencoder-based recommender systems. *Front. Comput. Sci.* **14**, 430–450. **2020**

8. Malik S. Movie Recommender System using Machine Learning. EAI Endorsed Trans Creat Tech [Internet]. **2022**

9. Cintia Ganesha Putri, D.; Leu, J.-S.; Seda, P. Design of an Unsupervised Machine Learning-Based Movie Recommender System. *Symmetry 12*, 185. **2020**

10. Jayalakshmi, S., Ganesh, N., Cˇep,R., SenthilMurugan,J. Movie Recommender Systems: Concepts, Methods, Challenges, and Future Directions. *Sensors 22*, 4904. **2022**

11. Zhang, S., Yao, L., Sun, A., Tay, Y. Deep Learning based Recommender System: A survey and new perspectives. *ACM Comput. Surv.*, 38. **2019**