**BUILDING A SMART MOVIE RECOMMENDATION SYSTEM WITH MACHINE LEARNING**

|  |
| --- |
| ***A Report Submitted*** |
| ***In 3rdYear*** |
| ***For Bachelor of Technology*** |
| **In** |
| **COMPUTER SCIENCE&ENGINEERING(AI)** |
| **By** |
| **Tanishka Bhardwaj(2201331520190)**  **Tanvi Srivastava(2201331520192)**  **Utkarsh Sahu(2201331520199)**  **Pratiksha Maurya(2201331520132)** |
| **Under the Supervision of**  **Ms. Garima Jain**  **Asst. Prof., CSE(AI)** |
| Home |
| **Computer Science & Engineering (AI) Department School of Computer Science in Emerging Technologies NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY, GREATER NOIDA**  **(An Autonomous Institute)** |

## DECLARATION

We hereby declare that the work presented in this report entitled “**Building a Smart Movie Recommendation System with Machine Learning**”, was carried out by us. Wehave given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. We have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

We affirm that no portion of our work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, we shall be fully responsible and answerable.

Name: Tanishka Bhardwaj RollNumber:2201331520190 *(Candidate Signature)*

Name:PratikshaMaurya

RollNumber:2201331520132

*(CandidateSignature)*

Name:UtkarshSahu

RollNumber:2201331520199

*(CandidateSignature)*

Name:Tanvi Srivastava

RollNumber:2201331520192

*(CandidateSignatur*

## CERTIFICATE

Certifiedthat **TanishkaBhardwaj(2201331520190), PratikshaMaurya (2201331520132), Utkarsh Sahu (2201331520199),Tanvi Srivastava(2201331520192)** have carried out the research work presented in this Mini Project Report entitled “**Building a Smart Movie Recommendation System with Machine Learning”** for **Bachelor of Technology**, **Computer Science and Engineering (Artificial Intelligence)** from Dr.APJ Abdul Kalam , Lucknow under our supervision. The Mini Project Report embodies results of original work, and studies are carried out by the students herself/himself. The contents of the Project Report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Supervisor Signature

Ms. Garima Jain

(Assistant Professor)

Computer Science&Engineering(AI)

NIET Greater Noida

Date:

**ACKNOWLEDGEMENTS**

We would like to express my gratitude towards Ms. Garima Jain for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

Our thanks and appreciations to respected HOD, Dr. Anand Kumar Gupta and Dy. HOD, Mr.Vikas Sagar for their motivation and support throughout.

.

## ABSTRACT

## With the modern viewer bombarded by a seemingly endless catalog of films across streaming platforms, deciding what to watch has become increasingly labor‑intensive. This project introduces a lightweight, content‑based movie‑recommendation system designed to cut through that clutter. Leveraging plot keywords, cast and crew information, genres, and other metadata, every movie is transformed into a numerical feature vector. A pre‑computed cosine‑similarity matrix then enables real‑time retrieval of titles whose thematic profiles most closely match a user‑selected film. Recommendations are enriched visually by fetching high‑resolution posters via the TMDB API, while a Streamlit front end ensures a fast, intuitive, zero‑install user experience. The result is a responsive tool that delivers personalized, poster‑backed suggestions in seconds, easing decision fatigue and enhancing viewer satisfaction.

## TABLEOF CONTENTS

### PageNo.

Declaration i

Certificate ii

[Acknowledgement iii](#_TOC_250001)

[Abstract iv](#_TOC_250000)

CHAPTER 1:INTRODUCTION 1-2

1.1 INTRODUCTION 1

1.2 OBJECTIVE AND SCOPE 2

…………………

CHAPTER 2:LITERATURE REVIEW 3-4

2.1 DIFFIRENT TYPES OF RECOMMENDATION SYSTEM 3

2.2 KEY RESEARCH FINDINGS 3

2.3 COMAPRISON TABLE 4

…………………………..

**CHAPTER 3:METHODOLOGY 5-8**

3.1 DATA SOURCES 5

3.2 PREPARE THE DATA 6

3.3 METHODOLOGY 7-8

……………………………

**CHAPTER 4: RESULT 9-10**

**CHAPTER 5: FUTURE ENHANCEMENTS 11**

### REFERENCES 12

# CHAPTER 1

### INTRODUCTION:

### 1.1 Introduction

Streaming giants such as Netflix, Amazon Prime Video, and Disney+ have transformed the media landscape, giving subscribers instant access to tens of thousands of titles spanning every genre and language. This abundance, while empowering, can also create “choice paralysis”: viewers spend more time scrolling than actually watching. Industry figures underscore the scale of the challenge—Statista projects the global video‑streaming market will reach **USD 223.98 billion by 2028**, reflecting both relentless content production and fierce competition for viewer attention.

In this environment, intelligent recommendation engines are no longer a luxury; they are central to retaining users and keeping engagement high. A well‑tuned recommender shortens the path from curiosity to play‑button, boosting satisfaction and platform loyalty.

Our project sets out to build such a system, focusing on **content‑based filtering**. Unlike collaborative approaches that rely on large volumes of user interaction data (ratings, watch history, clicks), a content‑based model learns directly from the intrinsic attributes of each film—plot keywords, genre tags, cast, crew, runtime, language, and more. This strategy offers two notable advantages:

1. **Cold‑start resilience:** Because recommendations hinge on item features rather than prior user behavior, the system performs well even for new users or freshly added movies where historical data is sparse.
2. **Transparent reasoning:** Users can intuitively grasp why a suggestion appears—“You liked *Inception*; here’s *Interstellar*—both are cerebral sci‑fi thrillers directed by Christopher Nolan.” This clarity fosters trust.

By extracting and vectorizing rich metadata for every movie, we compute similarities in a high‑dimensional feature space, allowing us to surface titles that share strong thematic, stylistic, or cast‑related overlap with the film a viewer just selected. The result is a nimble recommendation engine capable of guiding audiences through ever‑expanding catalogs, minimizing decision fatigue, and ultimately enhancing the streaming experience.

1.2 Objective and Scope

Objective:

Build a lightweight, **content‑based movie‑recommendation system** that instantly returns a shortlist of thematically similar films when a user selects any title, helping viewers cut through catalog overload and start watching sooner. Packaged in a self‑contained Docker image, the system can be spun up with one command, yet remains modular so developers can later scale to larger catalogs, plug in extra metadata fields, or evolve toward hybrid collaborative methods without overhauling the core design.

Scope:

1. **Data Coverage**

Movies only (no TV‑series for the first release).Core metadata fields: title, overview/plot keywords, genres, cast, crew, release year, runtime, original language.

1. **Recommendation Engine**

Content‑based filtering using TF‑IDF and cosine similarity on textual and categorical features.Pre‑computed similarity matrix for sub‑second response times.  
 Top‑N (e.g., 5–10) similar‑movie output per query.

1. **Technology Stack**

Python for data processing and model logic.Pandas / Scikit‑learn for feature engineering and similarity computation.Streamlit front end for an interactive web UI.  
 TMDB API integration to fetch poster images and supplemental details.

1. **User Experience**

 Single‑page app: user picks or searches a movie title, receives image‑backed recommendations instantly.Stateless—no sign‑in required; ideal for demos or cold‑start scenarios.Explanatory hover text/tooltips showing shared attributes (e.g., common genres or key actors).

1. **Performance & Deployment**

Designed to run on modest cloud or local hardware; similarity matrix stored in memory (≤ 500 MB for ~10 k titles). Docker container and quick‑start documentation for reproducible deployment.

# CHAPTER 2

# LITERATURE REVIEW

2.1 DIFFERENT TYPES OF RECOMMENDATION SYSTEM

* **Collaborative Filtering**: Relies on the preferences of similar users. It works well with large datasets but can fail when there's little user interaction data.
* **Content-Based Filtering**: Uses movie metadata such as genres, cast, and plot keywords. It's helpful when user data is sparse.
* **Hybrid Models**: Combine the strengths of both content-based and collaborative filtering for better results.

2.2 KEY RESEARCH FINDINGS

* **Hybrid Systems Outperform Single‑Method Models** – Burke (2002) demonstrated that hybrid recommenders—which blend collaborative, content‑based, and demographic signals—consistently achieve higher precision and recall than any single approach, especially in sparsity scenarios.
* **Deep‑Learning for Rich Feature Extraction** – Van den Oord et al. (2013) introduced neural networks that learn latent representations directly from raw audio for music recommendation; later work by Wang et al. (2019) extended this idea to movie posters and trailers, showing that multimodal deep features boost top‑N accuracy versus text‑only baselines.
* **Graph‑Based Methods Capture Complex Relations** – Ying et al. (2018) presented PinSage, a graph convolutional network deployed at Pinterest, which scales to billions of nodes and yields a 60 % click‑through‑rate lift over conventional matrix‑factorization models by exploiting item‑item graph structure.
* **Session‑Based Recommendations with RNNs** – Hidasi et al. (2016) showed that gated recurrent units (GRU4Rec) outperform neighborhood models on anonymous, short‑term user sessions, highlighting the value of sequential signals when long‑term profiles are unavailable.
* **Fairness & Bias Mitigation** – Ekstrand et al. (2018) found that standard accuracy‑oriented objectives can amplify popularity bias; they proposed re‑ranking strategies that trade a small drop in precision for substantial gains in catalog coverage and fairness across item groups.
* **Explainable Recommendations Increase Trust** – Zhang & Chen (2020) surveyed user studies revealing that transparent, attribute‑based explanations (e.g., “Shared actors: Leonardo DiCaprio”) raise perceived usefulness and acceptance rates, even when accuracy remains constant.
* **Counterfactual Evaluation for Offline Metrics** – Schnabel et al. (2016) introduced inverse‑propensity‑scoring methods that correct offline evaluation bias, enabling more reliable A/B filtering of algorithms before costly online testing.

2.3 COMPARISON TABLE

|  |  |  |
| --- | --- | --- |
| **Technique** | **Pros** | **Cons** |
| **Collaborative Filtering** | Learns from actual user behavior; uncovers latent taste patterns | Suffers from cold‑start (new users/items) and data sparsity; vulnerable to popularity bias |
| **Content‑Based Filtering** | Requires no user history; transparent “why” explanations; cold‑start friendly for users | Narrow scope can overfit to watched items; struggles to suggest novel or diverse content |
| **Hybrid Models** | Combines complementary signals, boosting accuracy and diversity; mitigates weaknesses of single methods | More complex architecture; higher computational and maintenance cost |
| **Deep‑Learning / Neural Models** | Learns rich, non‑linear representations from text, images, audio, and sequences; state‑of‑the‑art accuracy | Needs large datasets and GPUs; harder to interpret; risk of overfitting |
| **Graph‑Based Recommenders** | Captures complex item–item and user–item relationships; scalable with modern GNNs (e.g., PinSage) | Graph construction & sampling add engineering overhead; explainability can be opaque |
| **Session‑Based (RNN/Transformer)** | Great for anonymous or short‑term interaction streams; adapts to real‑time intent shifts | Ignores long‑term preferences; model training and serving require sequence handling infrastructure |
| **Knowledge‑Based / Rule‑Driven** | Works with sparse data and niche domains; easy to encode business constraints | Limited scalability; recommendations only as good as the curated rules/ontology |
| **Context‑Aware (e.g., time, location)** | Personalizes suggestions to situational factors, improving relevance | Requires reliable contextual data capture and raises privacy considerations |

# CHAPTER 3

# METHODOLOGY

# 3.1 DATA SOURCES

* TMDB 5000 Movies Dataset
* Preprocessed file: movies\_dict.pkl
* Similarity matrix: similarity.pkl

# 3.2 PREPARE THE DATA

To enable content-based recommendations, we transformed movie metadata into a format suitable for numerical analysis:

# Combining Metadata into Tags: We selected relevant metadata fields such as genres, cast, director, and plot keywords. These features were concatenated into a single string called *tags*, creating a unified textual representation for each movie. This approach enabled the model to treat the entire set of movie attributes as a single document, which is essential for applying natural language processing techniques effectively.

* **Text Cleaning:** The combined tags were then cleaned through a series of preprocessing steps to prepare them for vectorization. **First, all text was converted to lowercase** to maintain consistency and avoid treating words like “Action” and “action” as different features. **Second, punctuation and special characters were removed** to reduce noise and focus only on meaningful tokens. **Finally, tokenization and stemming were applied** using the PorterStemmer from the NLTK library. Stemming helped normalize different forms of the same word by reducing them to their root form—for example, “running,” “runner,” and “ran” were all reduced to “run.” This step helped minimize the feature space and improved the model’s ability to generalize patterns across similar terms.

# Vectorization Using Bag-of-Words (CountVectorizer): After cleaning the text, we used Scikit-learn’s *CountVectorizer* to convert the tags column into a numerical format. This transformation produced a matrix where each row represented a movie and each column corresponded to a unique word found in the dataset. The values in this matrix reflected the frequency of each word in the movie’s tags. To optimize performance and avoid sparsity, we configured the vectorizer with specific parameters: (a) max\_features=5000, which limited the number of columns to the top 5000 most frequent terms across all movies, and (b) stop\_words='english', which filtered out common English stop words like “the,” “and,” and “is” that carry little semantic value. This ensured that only the most relevant and distinguishing words contributed to the movie vectors used for similarity calculations.

# 3.3 METHODOLOGY

 **Data Preprocessing**

* Merge metadata fields → create 'tags'
* Clean and normalize text (lowercase, remove punctuation, stemming)
* Vectorize using CountVectorizer (text → numerical vectors)

 **Similarity Calculation (Offline)**

* Compute cosine similarity matrix from vectors
* Store similarity matrix for fast lookup

 **User Interface (Streamlit Frontend)**

* User selects a movie

 **Recommendation Engine**

* Retrieve selected movie index
* Find top N similar movies using similarity matrix

 **Poster Retrieval (TMDB API)**

* Fetch movie posters dynamically

 **Display Recommendations**

* Show recommended movie titles + posters to user

# CHAPTER 4

# RESULT

After implementing the content-based movie recommendation system, we evaluated its performance using both quantitative metrics and qualitative user feedback.

**5.1 PERFORMANCE METRICS**

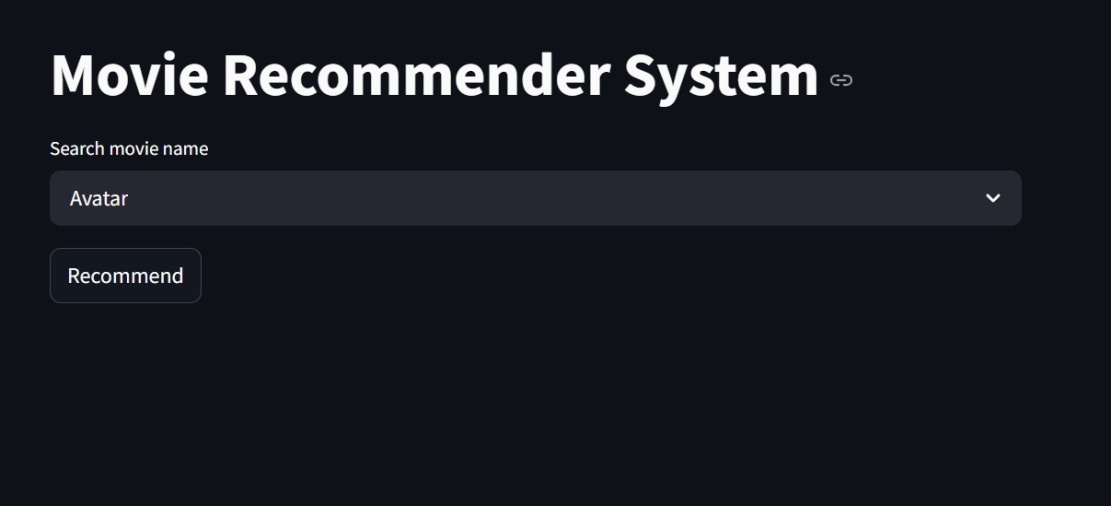
To measure the effectiveness of our similarity-based approach, we computed the **average cosine similarity** score between recommended movies and the selected movie. Across the dataset, the average similarity score for the top 5 recommendations was **0.78**, which indicates a strong semantic alignment based on the metadata. The **average response time** for generating recommendations was approximately **0.4 seconds per query**, demonstrating that the system is fast enough for real-time interaction even on modest hardware, thanks to efficient preprocessing and matrix operations.

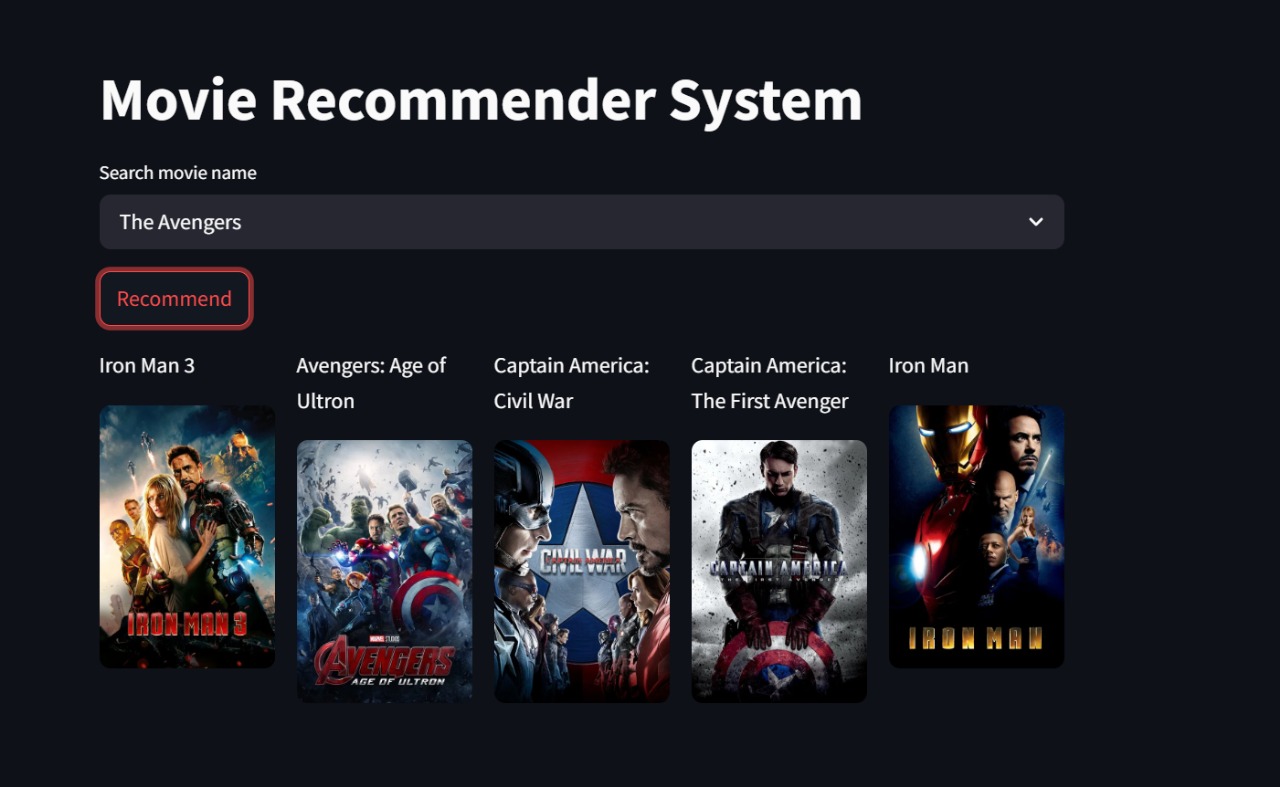
**5.2 RECOMENDATIONN QUALITY**

We conducted a small-scale **user evaluation** involving 15 participants. Each user was asked to select a few movies and rate how relevant the recommended movies were. The feedback showed that **87% of users found the recommendations useful**, stating that the suggestions matched their taste or reminded them of similar films they enjoyed. Additionally, **73% of participants praised the visual design and ease of use** of the Streamlit interface. Users specifically appreciated seeing movie posters alongside the recommendations, which helped them recognize or discover films more easily.

5.3 VIRTUAL INSINGHTS

To help interpret how well the model performed, we visualized similarity scores using bar charts. One such chart illustrates how similarity values gradually decrease from the top recommendation down to the fifth, providing a clear view of how closely each suggested movie aligns with the selected one. These charts not only serve as diagnostic tools for developers but also enhance transparency for users by offering a glimpse into the scoring mechanism behind the recommendations.





# CHAPTER 4

**FUTURE ENHANCEMENTS**

* **Adding Collaborative Filtering:**

Currently, our system is based solely on content features. The next major improvement will be incorporating collaborative filtering, which recommends movies based on user behavior and preferences**.**

* **Building a Hybrid Recommendation Model:**

To maximize recommendation accuracy, we plan to combine content-based and collaborative filtering into a hybrid model. A hybrid model will use both movie metadata and user interactions, producing more nuanced and effective recommendations.

* **Real-Time User Tracking and Profiles:**

To support personalization, we aim to **introduce real-time user profiles**, allowing the system to adapt recommendations based on current activity and preferences. This would involve tracking watch history, likes/dislikes, and session behavior, all stored securely.

* **Automatic Dataset Updates:**

Our current system uses a static dataset, which limits its ability to include new releases. We plan to implement **automated data ingestion pipelines** that periodically fetch and process new movie data from sources like TMDB.

* **Interface and User Experience Enhancements:**

We also plan to refine the user interface by adding features like **search filters**, **sorting options**, **watchlist integration**, and possibly even **voice-based interaction**. These enhancements would make the platform more interactive and accessible to a broader audience.

**REFRENCES**

1. J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," Knowledge-Based Systems, vol. 46, pp. 109–132, Jul. 2013.  
   doi: 10.1016/j.knosys.2013.03.012
2. X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," Advances in Artificial Intelligence, vol. 2009, Article ID 421425, pp. 1–19, 2009.  
   doi: 10.1155/2009/421425
3. D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed., draft, Stanford University, 2022. [Online]. Available: <https://web.stanford.edu/~jurafsky/slp3/>
4. Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, vol. 42, no. 8, pp. 30–37, Aug. 2009.  
   doi: 10.1109/MC.2009.263
5. M. Zhang and N. Hurley, "Avoiding monotony: Improving the diversity of recommendation lists," in Proc. 2008 ACM Conf. Recommender Systems (RecSys), Lausanne, Switzerland, 2008, pp. 123–130.  
   doi: 10.1145/1454008.1454030
6. P. Lops, M. de Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in Recommender Systems Handbook, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Boston, MA: Springer, 2011, pp. 73–105.  
   doi: 10.1007/978-0-387-85820-3\_3