# SmartRecs
## Personalized Media Recommendation System using Collaborative Filtering

**Santa Clara University**

**CSEN 240: Machine Learning**

**Project Report**

**Wineel Wilson Dasari**
**07700025431**

# Contents

# Abstract

With the increasing volume of streaming content, users often face decision fatigue when choosing what to watch. To address this, the SmartRecs project introduces a hybrid movie recommendation system that leverages both content-based filtering and collaborative filtering to deliver relevant and personalized suggestions. By utilizing IMDb datasets and applying techniques such as TF-IDF vectorization and matrix factorization using Singular Value Decomposition (SVD), SmartRecs computes hybrid scores to rank top-N recommendations. The system handles large-scale data using Kaggle infrastructure and includes fuzzy string matching, genre-based filtering, and fallback handling for edge cases.

Unlike traditional recommenders that rely solely on collaborative data or hard-coded similarity, SmartRecs adapts to a user's input with intelligent filtering and balancing strategies. Built using Python with industry-standard libraries like scikit-learn and Surprise, the model demonstrates practical machine learning deployment on large-scale public datasets. A fully functional implementation is accessible through Kaggle for transparency and further exploration. The approach is also flexible, adaptable, and extendable to a range of recommendation tasks beyond movies.

# Introduction and Background

Recommendation systems are integral to enhancing user experience on streaming platforms, online stores, and information services. By understanding user preferences and item characteristics, these systems narrow down massive content libraries into manageable, personalized suggestions. This project focuses on building a hybrid movie recommendation engine called SmartRecs using official IMDb datasets.

Traditional recommendation systems use one of two approaches. Content-based filtering compares features of items, such as genres, keywords, or cast, to recommend similar items to the user's past preferences. Collaborative filtering predicts a user's interest in items based on ratings or interactions from similar users. Each method has limitations: content-based systems may offer limited variety, while collaborative systems struggle with cold-start problems. Combining them results in a hybrid approach that takes advantage of both methods.

SmartRecs is designed to take a single movie title as input and return a list of recommended movies ranked by a hybrid score. The hybrid score is based on genre similarity computed using TF-IDF and cosine similarity and predicted rating scores from an SVD collaborative filtering model. The system includes user-friendly features such as fuzzy string matching and automatic filtering of mismatched genres (e.g., excluding horror).

The goal of SmartRecs is to address gaps in existing systems by working without login data or user history while still delivering recommendations that are both diverse and precise. This has applications in academic, hobbyist, and entry-level industry settings where user data might not always be available.

In addition, SmartRecs serves as a foundation for exploring applied machine learning in practical domains, combining text-based feature engineering with numerical prediction models. It also

highlights how data preprocessing, scalability, and inference pipelines come together in real-world ML projects. The design encourages modularity, making it easy to experiment with different models, feature types, and evaluation strategies.

Beyond technical goals, the project also emphasizes real-world usability and robustness. It handles noisy input through fuzzy matching and safeguards recommendations by avoiding irrelevant or inappropriate genres unless directly matched.

# Dataset and Preprocessing

This project makes use of publicly available datasets from IMDb, which are provided in .$tsv$ (tab-separated values) format. The two main files used were $title.basics.tsv$ and $title.ratings.tsv$.

- $\boldsymbol{title.basics.tsv}$**:** Contains metadata about movies, including $tconst$ (unique ID), $primaryTitle$, $startYear$, $genres$, and $titleType$.
- $\boldsymbol{title.ratings.tsv}$**:** Includes $tconst$, $averageRating$, and $numVotes$ for each movie.

## Preprocessing Steps:

1. **Loading and Merging:** The datasets were read using pandas, and merged on the $tconst$ key.
2. **Filtering:** To ensure quality:
    a. Only titles where $titleType == 'movie'$ were included
    b. Adult content ($isAdult == 0$) was retained
    c. Movies released after 1980 and with $numVotes >= 1000$ were selected
    d. Rows missing $primaryTitle$, $startYear$, or $genres$ were removed
3. **Genre Normalization:** Genres were converted to lowercase, and comma-separated values were converted into space-separated strings for TF-IDF processing.
4. **Ratings:** IMDb ratings were used both as collaborative input and for potential evaluation.

The original dataset was large, $title.basics.tsv$ alone was over 1 GB, while $title.ratings.tsv$ added another 27 MB. Combined, memory usage during processing exceeded 20 GB in some stages. This necessitated the move to Kaggle Notebooks, which offered up to 30 GB RAM and streamlined access to data.

Additionally, column names were stripped of whitespace, and datatype conversions were applied where necessary to ensure compatibility with the TF-IDF vectorizer and Surprise SVD model. This

preprocessing was crucial to ensure the performance and accuracy of the recommendation pipeline. In total, the final working dataset contained thousands of well-structured, high-confidence movie entries suitable for scalable model training and testing.

The filtering and transformation process reduced noise, prevented cold start issues, and created consistency in data formats. Generalization was particularly important for effective vectorization, and vote count filtering ensured only widely rated movies were retained.

# Methodology

The methodology for SmartRecs combines two major machine learning approaches: content-based filtering and collaborative filtering. These are integrated in a hybrid system that calculates recommendation scores based on both similarity in genre and predicted ratings.

## Content-Based Filtering

This method focuses on the features of the items themselves, in this case, movie genres. Each movie's genre is transformed into a vector using Term Frequency-Inverse Document Frequency (TF-IDF), a technique that assigns more weight to unique terms across documents.

- The genres field is first preprocessed by replacing commas with spaces.
- TF-IDF vectorization is performed using scikit-learn's $TfidfVectorizer$.
- Cosine similarity is calculated using $cosine\_similarity$ from scikit-learn to generate a similarity matrix between all movie vectors.

This matrix allows the system to retrieve movies that are closest in genre space to the input title. The vectorization and similarity calculation steps are efficient and handle large input spaces well. Cosine similarity was chosen over alternatives like Euclidean distance because it performs better in sparse, high-dimensional spaces typical of TF-IDF output.

## Collaborative Filtering

Collaborative filtering focuses on user-item interactions. Since user data is not available, we simulate users using movie IDs ($tconst$) and treat movie titles as items. IMDb's average ratings serve as a signal of user preference.

- We use the Surprise library's $Reader$, $Dataset$, and $SVD$ classes to build the model.
- The dataset is formatted into a DataFrame with columns: $userID$, $itemID$, and $rating$.

- The model is trained on simulated interaction data, learning latent features that describe relationships between movies.

SVD (Singular Value Decomposition) is a powerful matrix factorization technique. It decomposes the user-item matrix into lower-dimensional representations, allowing prediction of missing values, in this case, predicting how one movie (userID) would rate another movie (itemID).

## Hybrid Model and Scoring

The hybrid model brings both methods together. For any given input movie:

- We compute cosine similarity scores using the genre vector.
- We predict SVD scores for the same set of candidate movies.
- We calculate a final score:
    - $final\_score = 0.6 * svd\_score + 0.4 * genre\_score$
- This weighted score is tuned to favor collaborative filtering slightly while still respecting content similarity. The top-N movies with the highest scores are selected for recommendation.

## Additional Utilities

- **Fuzzy Title Matching:** If an exact match is not found for the input, $difflib.get\_close\_matches$ is used to suggest alternatives.
- **Filtering Irrelevant Recommendations:** Movies with genres such as horror or experimental are filtered out unless they match the input.
- **Duplicate Handling:** Repeated titles are removed using a $seen$ set.
- **Input Flexibility:** Both interactive input and automated batch testing modes are supported.

By combining both genre semantics and rating behavior, SmartRecs achieves a balance that delivers accurate, user-friendly, and safe recommendations. The system was designed to be

modular, allowing easy extension or substitution of the content or collaborative modules with improved models in the future.

# Experimental Results

To evaluate SmartRecs, two sample movies were used to test the recommendation pipeline: one action/sci-fi title (***Inception***) and one animated family film (***Cars 2***). The purpose was to assess whether the hybrid model produced thematically relevant and collaborative-aware recommendations.

## Example 1: Input - "Inception" (2010)

The user typed "Inception," and the system correctly matched the 2010 movie. The genre tags were Action, Adventure, and Sci-Fi.

See the screenshot below for the actual output format and recommendations displayed by the system.

```
Enter a movie name:  Inception

Did you mean one of these?
  0: Inception (2010.0, Action,Adventure,Sci-Fi)

Enter the number of the correct movie:  0

Showing recommendations based on: Inception


Top Recommendations:

Title                                                Year   Genres                                    Score
-------------------------------------------------------------------------------------------------------------
Edge of Tomorrow                                     2014.0 Action,Adventure,Sci-Fi                   4.45
Vikram                                               1986.0 Action,Adventure,Sci-Fi                   4.45
Terminator Salvation                                 2009.0 Action,Adventure,Sci-Fi                   4.43
Predators                                            2010.0 Action,Adventure,Sci-Fi                   4.42
Rogue One: A Star Wars Story                         2016.0 Action,Adventure,Sci-Fi                   4.42
The Matrix                                           1999.0 Action,Sci-Fi                             4.4
Independence Day                                     1996.0 Action,Adventure,Sci-Fi                   4.4
Avengers: Endgame                                    2019.0 Action,Adventure,Sci-Fi                   4.39
Black Panther                                        2018.0 Action,Adventure,Sci-Fi                   4.38
Mad Max Beyond Thunderdome                           1985.0 Action,Adventure,Sci-Fi                   4.38

Recommendation complete.
```

These results show the model correctly favored genre-matched, well-rated action/sci-fi movies. Most of the recommendations are high-grossing, popular titles with similar pacing, tone, and

audience appeal. This highlights the effectiveness of hybrid scoring, especially how collaborative filtering elevated blockbusters like *Endgame* and *Rogue One*.

## Example 2: Input = "Cars 2". Recommended animated family films:

When the user typed "cars," the system displayed a fuzzy-matched list including *Cars*, *Cars 2*, and *Cars 3*. The user selected *Cars 2* (Adventure, Animation, Comedy).

```
Enter a movie name:  cars

Did you mean one of these?
  0: Used Cars (1980.0, Comedy)
  1: Riding in Cars with Boys (2001.0, Biography,Comedy,Drama)
  2: Old Men in New Cars (2002.0, Action,Comedy,Crime)
  3: Cars (2006.0, Adventure,Animation,Comedy)
  4: Serbian Scars (2009.0, Action,Drama,Thriller)
  5: Battle Scars (2020.0, Crime,Drama,War)
  6: Cars 2 (2011.0, Adventure,Animation,Comedy)
  7: Cars of the Revolution (2008.0, Drama,History)
  8: Stealing Cars (2015.0, Drama)
  9: Cars 3 (2017.0, Adventure,Animation,Comedy)

Enter the number of the correct movie:  6

Showing recommendations based on: Cars 2


Top Recommendations:

Title                                              Year   Genres                                  Score
-------------------------------------------------------------------------------------------------------
Finding Nemo                                       2003.0 Adventure,Animation,Comedy              4.29
Isle of Dogs                                       2018.0 Adventure,Animation,Comedy              4.28
9                                                  2002.0 Crime,Drama,Mystery                     4.27
The Lion King                                      1994.0 Adventure,Animation,Drama               4.27
The Little Prince                                  2015.0 Adventure,Animation,Comedy              4.27
Sinbad: Legend of the Seven Seas                   2003.0 Adventure,Animation,Comedy              4.27
That Christmas                                     2024.0 Adventure,Animation,Comedy              4.26
Home                                               2009.0 Documentary,Family                      4.25
The Iron Giant                                     1999.0 Action,Adventure,Animation              4.25
Ice Age: The Meltdown                              2006.0 Adventure,Animation,Comedy              4.24

Recommendation complete.
```

The results are highly consistent with Cars 2. Most suggestions are family-oriented animated movies with a similar tone and target audience. Again, hybrid scoring avoided outliers, and genre filtering ensured irrelevant results (e.g., horror) didn't appear.

## Insights and Observations

- **Genre consistency** was maintained across all top recommendations.

- **Collaborative filtering** helped surface high-rated and popular titles.
- **Cosine similarity via TF-IDF** ensured thematic coherence.
- **Fuzzy title matching** handled variations in input spelling.
- **Horror filtering** worked as intended; no horror films appeared unless they directly matched.

Overall, the system provided intuitive and high-quality recommendations. These tests validate that SmartRecs functions as a reliable hybrid engine, even without user profile data.

# Challenges and Learnings

Building SmartRecs required addressing several practical and technical challenges throughout the development process. Each obstacle contributed to a deeper understanding of machine learning workflows, real-world data processing, and system limitations.

1. **Dataset Size and Memory Management:**

   One of the first major hurdles was the sheer size of the IMDb datasets. The $title.basics.tsv$ file alone was over 1 GB, and when merged with ratings and transformed into TF-IDF matrices, the memory usage often exceeded 20–25 GB. Running this on a local PC led to repeated crashes, and even Google Colab's free memory was insufficient. This forced a transition to Kaggle Notebooks, which offered larger RAM and better support for high-volume data processing.

2. **Data Cleaning and Preprocessing:**

   IMDb datasets are raw and loosely structured. Many rows had missing genres, years, or invalid vote counts. Without thorough cleaning and filtering, model training would have been inconsistent. This process required careful use of pandas functions to remove noise and ensure all fields were standardized for modeling. Transforming comma-separated genres into TF-IDF-friendly input was also a key step that required iterative debugging.

3. **Hybrid Model Integration:**

   Combining SVD-based collaborative filtering with genre-based cosine similarity posed both conceptual and implementation challenges. These two models operate on different spaces; collaborative filtering uses user-item matrices, while content similarity operates on

text features. Creating a balanced hybrid score required testing multiple weight combinations and carefully handling exceptions when similarity or prediction scores failed.

4. **Handling User Input in Non-Interactive Platforms:**

   In platforms like Kaggle, interactive input using $input()$ causes runtime errors during notebook saves. To solve this, a second version of the pipeline was created that automatically selects test titles randomly or based on predefined input. This ensured that results could be reproduced and reviewed without manual typing.

5. **Output Interpretation and Filtering:**

   Another challenge was ensuring that recommendations made sense to the user. For instance, when a user inputs a children's movie like *Cars 2*, horror or unrelated genres shouldn't appear in the output. The system includes logic to exclude horror unless it matches the input genre and also removes duplicates or poorly matched suggestions. This improved user trust in the system's results.

**Through these challenges, key learnings emerged:**

- **Machine learning isn't just about modeling;** data engineering and real-world constraints are equally important.
- **Hybrid systems often perform better** but are harder to implement correctly.
- **Scalability and memory planning** are critical when working with large public datasets.
- **User experience matters**, even in a technical project. Fuzzy matching, fallback handling, and clean outputs all improved the system's usability.

These experiences reinforced the importance of practical design choices and iterative testing in delivering a functional, scalable ML system like SmartRecs.

# Conclusion

SmartRecs is a functional and extensible hybrid recommendation system that successfully merges content-based and collaborative filtering. By using IMDb metadata and average ratings, the system generates top-N recommendations with genre relevance and popularity balance. Its design addresses typical cold-start and sparsity issues while providing an intuitive user interface.

This project not only reinforces theoretical machine learning concepts but also demonstrates real-world application challenges, such as memory management, large-scale data handling, and usability optimization. The solution is modular, allowing future upgrades such as filtering by runtime, language, or integrating cast-based embeddings.

Furthermore, SmartRecs was evaluated on various inputs to confirm that the hybrid model consistently generated meaningful suggestions. Through the Kaggle platform, the code was tested for large-scale compatibility, offering reproducible results and an interactive demo for demonstration purposes. The approach balances simplicity and effectiveness, making it a promising prototype for future extensions, including REST API integration and UI deployment.

Future directions also include user login support, which would allow for profile-based recommendations, incorporating individual preferences, watch history, and explicit feedback. Additionally, using more sophisticated NLP techniques such as embeddings or transformer-based models could further improve genre and content similarity analysis.

SmartRecs was built in Python and tested in a production-like notebook environment on Kaggle. All dependencies are open-source, and the project can be extended further into a deployable API or integrated into content platforms. It stands as a strong academic project with practical implications.

# References


- IMDb Datasets: https://datasets.imdbws.com/
- Scikit-learn Documentation
- Surprise Recommendation Library: https://surpriselib.com/
- Cosine Similarity and TF-IDF: Stanford NLP Lectures
- Netflix Prize Dataset (as a collaborative filtering reference)
- Kaggle Notebook: https://www.kaggle.com/code/wineelscu/smartrecs