


 Imacharia /  
Movie\_Recomendation




 Code  Issues  Pull requests  Actions  Projects  Wiki  Security  Insights 



☆ 0 stars    1 fork    1 watching    Activity


 Public repository

 brian had recent pushes 17 minutes ago

[Compare & pull request](#)


 main ▾



 Branches  Tags



Imacharia Merge pull request [#58](#) from Imacharia/dorinelangat-patch-1 ...

17 minutes ago  164

[View code](#)

# Movie\_Recommendation\_System



# Business Understanding

---

## 1.1) Overview

---

This project aims to develop a personalized movie recommendation system by leveraging datasets containing movie information, user ratings, and interactions. The system will utilize content-based and collaborative filtering techniques to suggest movies to users based on their preferences and past interactions.

- Modules for movie and user profiling will be developed to analyze movie attributes and user preferences, respectively. Evaluation and optimization will be conducted to enhance the accuracy
- Furthermore, leveraging movie metadata, user ratings, and collaborative filtering algorithms, the system provides personalized movie recommendations to enhance the user's movie-watching experience.

## 1.2) Problem Statement

---

- The movie industry is vast and fast evolving , with countless movies and movie sequels released each year hence can be a challenge for the users to navigate through the vast amount of content and get to know which movies align with their preferences.
- To ease this, or rather improve the users' experience, we come up withh a recommendation system that provides personalized movie recommendations based on user preferences and similarities with other users, and also aim to improve user satisfaction, increase user engagement, which ultimately drive user's retention on the platform.

## 1.3) Objectives

---

### 1.3.1) Specific Objectives

- To develop a demographic recommendation system that suggests popular movies based on user demographic attributes.
- To implement a content-based recommendation system that recommends movies based on movie overviews, cast, and keywords.
- To build a collaborative filtering recommendation system that suggests movies based on user similarities and their ratings.
- To create a hybrid recommendation system that combines the techniques from content-based and collaborative filtering approaches to provide personalized movie recommendations.

## Success Metrics

---

- **Precision:** A higher precision indicates that the recommended movies are more relevant to the user's preferences. It helps assess the system's ability to provide high-quality and targeted recommendations.
- **Recall:** A higher recall indicates that the system is able to capture a larger portion of the user's preferences and suggest a wide range of relevant movies. It helps assess the system's ability to avoid missing out on movies that the user would enjoy.
- **RMSE (Root Mean Square Error):** quantifies the difference between the predicted ratings and the actual ratings given by users. A lower RMSE indicates that the system's predictions are closer to the actual ratings, suggesting higher accuracy in estimating user preferences. It helps evaluate the system's ability to predict ratings accurately and make personalized recommendations.

## Data Understanding

---

This project utilizes two different datasets:

### TMDB\_Dataset

- TMDB is a popular database that provides comprehensive information about movies, that contains the following titles, release dates, genres, cast and crew information. Credit information is given as well about the cast and crew information whereby the cast and crew are involved in each movie. With the combination of the datasets, we gain valuable insights and perform various analyses related to the movie industry
- `id` : Unique identifier for each movie
- `title` : Title of the movie
- `cast` : List of actors/actresses in the movie
- `crew` : List of crew members involved in the movie
- `budget` : Budget of the movie
- `genres` : List of genres associated with the movie
- `homepage` : Website URL of the movie
- `keywords` : List of keywords associated with the movie
- `original_language` : Original language of the movie
- `original_title` : Original title of the movie
- `production_companies` : List of production companies involved in the movie
- `production_countries` : List of countries where the movie was produced

- `release_date` : Release date of the movie
- `revenue` : Revenue generated by the movie
- `runtime` : Duration of the movie in minutes
- `spoken_languages` : List of languages spoken in the movie
- `status` : Current status of the movie (e.g., Released, Post Production)
- `tagline` : Tagline or slogan of the movie
- `vote_average` : Average vote rating for the movie
- `vote_count` : Number of votes received by the movie
- `tags` : List of tags associated with the movie

[Ratings\\_Dataset](#) which contains:

- `userId` : A unique identifier for the user who provided the rating.
- `movieId` : A unique identifier for the movie being rated.
- `rating` : The rating given by the user to the movie, usually on a scale of 1 to 5.
- `timestamp` : The timestamp indicating when the rating was recorded or updated.

## Data Cleaning and Preparation

---

The dataset has no duplicates.

Null values are present in the `homepage` and `tagline` columns only.

Some columns within our dataset contained a list of dictionaries. We solved this by extracting the desired attribute from each element in the columns.

We combined the modified columns into one named `tags`.

## EDA

---

In this section, we visualized different features in our dataset.

The `vote_average` column is normally distributed with the majority of it being 6-8.

We also checked and dropped outliers in the `popularity`, and `vote_count` columns.

## Modeling

---

In this section, we created different recommendation systems which include:

- A demographic recommendation system based on popularity.

The model calculates a weighted rating for movies based on their vote count and average rating. It filters the movies based on a minimum vote count threshold. The calculated score is used to sort and recommend the highest-rated movies.

- A Content-based recommendation system.

The model focuses on movie overview as a basis for recommendation. It processes the overview data and applies TF-IDF (Term Frequency-Inverse Document Frequency) to calculate the importance of terms in the collection of movies. The cosine similarity is computed between movies based on the TF-

☰ README.md



- A Collaborative-based recommendation system.

This model allows for coincidental recommendations; that is, it can recommend an item to user A based on the interests of a similar user B. Furthermore, the embeddings can be learned automatically, without relying on hand-engineering of features.

- Hybrid recommendation system.

Here, we build a simple hybrid recommender that brings together techniques we have implemented in the content-based and collaborative filter based engines.

## Conclusion

In conclusion, the recommendation system serves as a valuable tool in the movie industry to address the challenge of content navigation and provide personalized movie recommendations. By understanding user preferences, leveraging similarities between users, and utilizing movie features, the system aims to enhance the user experience, increase engagement, and ultimately contribute to user retention on the platform.

## Recommendations

- Real-time Updates: Incorporate a mechanism to continuously update the movie database with the latest releases, ratings, and reviews. This will ensure that the recommendation system remains up-to-date and can provide users with the most relevant movie suggestions.
- Contextual Factors: Consider contextual factors such as time of day, location, mood, and social trends to provide personalized recommendations that align with the user's current situation and preferences.
- User Feedback and Improvement Loop: Implement a feedback mechanism that allows users to rate and provide feedback on recommended movies. Utilize this feedback to continuously improve the recommendation algorithms and enhance the accuracy and relevance of future recommendations.

## Next Steps

- **System Design and Implementation:** Design and develop a scalable and robust recommendation system architecture that can handle a large volume of users and movies. Implement the recommendation algorithms, user profiling mechanisms, and real-time updates to create a functional prototype.
- **Evaluation and Validation:** Conduct extensive testing and evaluation of the recommendation system using real-world scenarios and user feedback. Measure the system's performance in terms of recommendation accuracy, relevance, and user satisfaction to identify areas for improvement.
- **Iterative Refinement:** Continuously iterate and refine the recommendation system based on user feedback and performance evaluation results. Incorporate new features, optimize algorithms, and enhance the user interface to provide an exceptional movie recommendation experience.

## Releases

No releases published

[Create a new release](#)

## Packages

No packages published

[Publish your first package](#)

## Contributors 6



## Languages

● Jupyter Notebook 99.9% ● Python 0.1%

## Suggested Workflows

Based on your tech stack



**Actions Importer**

Automatically convert CI/CD files to YAML for GitHub Actions.

Set up



**Django**

Configure