

Movies Recommendation System

Milestone 3 Report

Project Objective:

The primary objective of this project is to develop a movie recommendation system that suggests relevant movies to users based on their preferred movie input. The system leverages robust and extensive datasets collected from three reliable sources—IMDB, MovieLens, and TMDB—to ensure the recommendations are both accurate and reflective of real-world preferences.

A key strength of this system lies in the trustworthiness and volume of the data, which plays a crucial role in generating meaningful and practical movie suggestions. In addition to recommendation functionality, the project includes an interactive Python-based UI, enabling users to seamlessly explore, filter, and engage with the movie dataset.

Tool Type:

Recommendation system

Data Used:

- ❖ IMDB movie datasets (by genre)
(<https://www.kaggle.com/datasets/rajugc/imdb-movies-dataset-based-on-genre>)
- ❖ MovieLens Datasets (<https://grouplens.org/datasets/movielens/>)
- ❖ TMDB Dataset
(<https://www.kaggle.com/datasets/asaniczka/tmdb-movies-dataset-2023-930k-movies>)

Data Licences:

The datasets used in this project are obtained from publicly available and trustworthy sources. Each dataset adheres to the respective licensing agreements as described below:

- **IMDB Dataset (by Genre):**
Source: Kaggle / IMDB
License: This dataset is licensed under CC BY-NC-SA 4.0
- **MovieLens Dataset:**
Source: GroupLens Research
License: This dataset is licensed under CC BY 4.0, which allows reuse with

proper attribution.

- **TMDB 2023 Dataset:**

Source: Kaggle / TMDB

License: This dataset is licensed under Open Data Commons Attribution License (ODC-By) v1.0

All datasets are used solely for academic and research purposes and have not been redistributed or modified beyond the scope of analysis within this project.

Github link: <https://github.com/sanket1305/cap5771sp25-project/tree/main>

Instruction about how to run the project can be found on ReadMe page of the above repository

Tech Stack:

The following technologies and libraries were utilized in building the Movie Recommendation System:

Programming Language:

Python – the primary language used for data handling, model development, and user interface.

Data Storage:

SQLite – used for structured storage of movie-related metadata and queryable records.

Data Processing & Machine Learning:

Pandas and NumPy – for efficient data manipulation and numerical operations.

Collections – for structured handling of data using default dictionaries and counters.

scikit-learn (sklearn) – used for implementing KMeans, KNN, and other utility functions.

HDBSCAN – used for density-based clustering to uncover complex groupings in movie features.

Data Visualization & User Interface:

Matplotlib and Seaborn – for generating static visualizations during data exploration.

Streamlit – for building an interactive, web-based Python UI.

Plotly – used to create dynamic and interactive visualizations within the Streamlit app.

Project Timeline:

- ❖ Milestone 1: Data Collection, Preprocessing, and Exploratory Data Analysis (EDA) Timeline: February 5, 2025 – February 23, 2025 (2 weeks)

- Feb 5 – Feb 7: Identify and acquire datasets
- Feb 8 – Feb 10: Verify dataset accessibility, licensing, and documentation
- Feb 11 – Feb 15: Data preprocessing (handling missing data, outliers, scaling)
- Feb 16 – Feb 19: Exploratory Data Analysis (EDA) (statistical summaries, visualizations)
- Feb 20: Finalizing the EDA report and project documentation
- Feb 23: Submit Milestone 1 deliverables
- ❖ Milestone 2: Feature Engineering, Feature Selection, and Data Modeling
Timeline: February 21, 2025 – March 21, 2025 (5 weeks)
 - Feb 27 – Mar 3: Feature engineering (creating new features)
 - Mar 5 – Mar 10: Feature selection
 - Mar 11 – Mar 13: Splitting the dataset and initial model training
 - Mar 15 – Mar 18: Model tuning and hyperparameter optimization
 - Mar 20 – Mar 26: Model evaluation and comparison
 - Mar 31 – Apr 2: Preparing Milestone 2 report
 - Apr 7: Submit Milestone 2 deliverables
- ❖ Milestone 3: Evaluation, Interpretation, Tool Development, and Presentation
Timeline: March 24, 2025 – April 23, 2025 (5 weeks)
 - Apr 8 – Apr 10: Evaluate model performance on test data
 - Apr 11 – Apr 12: Interpret model results and address biases
 - Apr 13 – Apr 16: Develop an interactive dashboard for visualizations
 - Apr 17 – Apr 18: Implement the final recommendation system
 - Apr 19 – Apr 20: Tool testing and debugging
 - Apr 21: Prepare final report and GitHub repository updates
 - Apr 22: Record demo video and finalize presentation
 - Apr 23: Submit Milestone 3 deliverables

Project Milestones Report:

Milestone 1: Data Normalization and Initial Analysis

In Milestone 1, the dataset was normalized. During this process, various aspects of the data were explored. Some key observations included the large size of the datasets—300,000 records in IMDB and 1.18 million records in TMDB—as well as the similarities and mapping between the two datasets (IMDB and TMDB) provided by MovieLens.

Data analysis was performed using visualizations such as language dominance plots, pie charts for genre distribution, and scatter plots to analyze correlations between

variables. To facilitate further analysis, the processed data was stored in a database, allowing for efficient retrieval in subsequent stages of the project.

Milestone 2: Data Refinement and Feature Engineering

In Milestone 2, work continued using the data stored in the database. To reduce the number of records and retain only the most relevant data for feature extraction, common records between the TMDb and IMDb datasets were identified using the links provided in the MovieLens dataset. This refinement reduced the dataset size to approximately 50,000 records, which still provided a substantial amount of information for analysis.

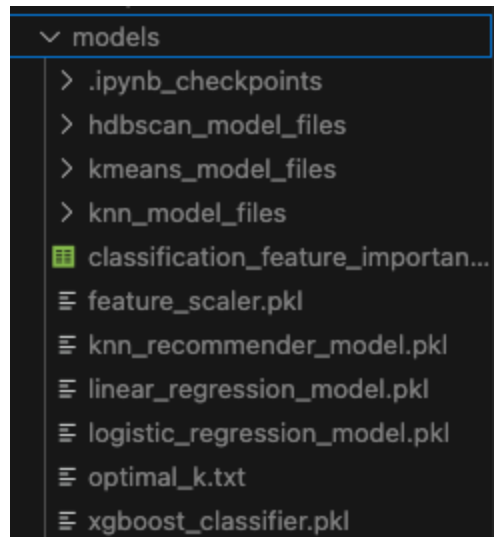
Numerical features such as rating, budget, and revenue were scaled, and feature engineering was conducted. A multi-label encoder was used for smaller categorical features such as genre, while TF-IDF vectorization was applied to high-dimensional categorical features like cast and directors. This helped capture the influence of top actors and directors in the movie recommendation system.

Several machine learning models were experimented with, including Linear Regression, Logistic Regression, K-Nearest Neighbors (KNN), and XGBoost. Although XGBoost performed well, considering the growing complexity of the project, the focus was narrowed down to building a recommendation system. Among the models evaluated, KNN yielded the best performance in line with project goals.

Milestone 3: Model Expansion and Clustering

As the project progressed into Milestone 3, the focus remained on enhancing the recommendation system. To evaluate different modeling approaches, additional models were trained, including K-Means and HDBSCAN clustering algorithms. The optimal value of k for K-Means was determined using the elbow method.

For each model, the trained models and their corresponding feature sets were saved as `.pkl` files in the `models` directory. Due to file size limitations, these models could not be pushed to GitHub. However, a link to download a ZIP archive containing the complete project setup has been provided in the README file.



After saving the models, I also created a notebook titled Milestone3.ipynb to evaluate the performance of all the models trained previously. Linear Regression and Logistic Regression were found to be among the poorer-performing models. However, this outcome was expected, as earlier analysis had shown minimal linear correlation between the features. Additionally, Logistic Regression was affected by significant class imbalance, which further impacted its performance.

```
Linear Regression validation MSE: 0.6739

Logistic Regression validation accuracy: 0.7588
Logistic Regression validation F1 score: 0.4141
Classification Report:
      precision    recall  f1-score   support

     0       0.96      0.76      0.85     5063
     1       0.28      0.76      0.41      638

 accuracy          0.76     5701
 macro avg         0.62     5701
 weighted avg      0.89     5701
```

I experimented with both K-Nearest Neighbors (KNN) and XGBoost. The goal of using KNN was to evaluate the relevance of the results in the context of building a recommendation system. In contrast, XGBoost was employed to train a classification model, with the objective of assessing the dataset's suitability for classification tasks. Among the two, XGBoost outperformed KNN in terms of overall performance.

```

KNN predicted neighbor indices and distances
[[ 7585  7586  7587  7584  7582]
 [ 9620  9619  9618  9617  9622]
 [ 8517  8514  8513  8518  8512]
 ...
 [15435 15432 15437 15434 15436]
 [  621   624   625   618   619]
 [ 7591  7590  7592  7589  7588]] [[ 9.17380715  9.50379419  9.64084972 18.43311908 56.70785764]
 [ 9.25670886 17.04688527 43.27945395 45.8781772  52.31159515]
 [ 8.31443615 27.33823457 30.84301711 32.28434809 47.80880497]
 ...
 [ 6.87765147 23.25312669 46.37535856 46.82641338 48.75762453]
 [ 9.96958591 10.79583471 11.67219258 11.91547038 12.08733908]
 [ 8.63806738 20.82508352 21.13124255 45.64741149 49.85699708]]

```

As demonstrated in the example above, the output (for KNN) includes indices and distance measurements for the recommendations generated for one of the test data points. These values indicate that the model is not overfitting. This observation was further validated manually by testing various samples through the user interface, which will be showcased in the video demonstration.

```

--- XGBoost Model Evaluation ---
Accuracy: 0.9198
Precision: 0.9119
Recall: 0.9198
F1 Score: 0.9121
ROC-AUC: 0.9128

```

The image above illustrates that XGBoost has been the most efficient model so far in terms of performance. In addition to this, I trained two clustering models: **K-Means** and **HDBSCAN**. Both models demonstrated distinguishable performance, with evaluation based on silhouette scores. The silhouette scores for each clustering model are listed below, providing insights into the clustering quality and separation achieved by each approach.

```

Silhouette Score [KMeans]: 0.8213
Silhouette Score [HDBSCAN]: 0.8599

```

For performance evaluation, I also adopted a manual approach, where genre similarity was calculated for each of the recommended results. This helped in assessing how well the recommendations aligned with the genre of the input data. The comparison results are presented below.

Final Evaluation Summary:				
	title	knn_score	kmeans_score	hdbscan_score
0	The Godfather	1.0	0.800000	0.0
1	Inception	1.0	0.700000	1.0
2	Titanic	1.0	0.933333	1.0

image: screenshot of code output based on evaluation for one sample


It is important to note that for one of the movies, the **HDBSCAN** genre similarity score was **0**. This occurred because the movie was identified as an outlier by the density-based clustering algorithm, and thus was not assigned to any cluster. Based on these observations, I decided to proceed with the **KNN** model (with cosine similarity of around 85% on avg), which consistently produced promising results. A detailed demonstration of this will be included in the video walkthrough. (**Note:** since the given query “The Godfather” had only a few genres, the genre similarity score came out to be 1, but I am considering a lot more parameters like genre, stars, actors using TF-IDF. So that KNN model does not overfit.as we have already seen distance comparison above for KNN)

As part of **Milestone 3**, I also dedicated time to developing a **user interface (UI)** for end users. The interface includes four distinct tabs, each designed to enhance user interaction and experience.

1. Insights (shows some general insights using plots for different features)



2. Movie explore (user can play with data using this tab)'



Movie Recommender System

[Insights](#)[Movies Explorer](#)[Clusters](#)[Recommendations](#)

Movie Explorer

Year Range

1906

2024

IMDb Rating

5.00

10.00

Select Genres

Choose an option

Runtime

Any

Search by Movie Title, Director, or Actor

Sort by

IMDb Rating (High to Low)

Found 21189 movies matching your criteria

Average Rating

6.2

Average Runtime

98 min

Most Common Year

2018

Showing top 100 results

Roadkill (2022)

★ 9.4 | 🕒 93 min

Action,Drama,Thriller

A thief operating along the highways of rural Australia gets caught in the crossfires of an ongoing police investigation after he mugs a serial killer...

Shelter in Solitude (2022)

★ 9.3 | 🕒 93 min

Crime,Comedy,Drama

Follows a wannabe country singer, faith-filled and her unconventional relationship...

My First Kiss and the People Involved (2016)

★ 9.2 | 🕒 81 min

Mystery,Drama

Sam's silent world confounds the fellow residents of her group home. When her only friend goes missing Sam sets out to find her, uncovering a tale of...

The Godfather (1972)

★ 9.2 | 🕒 175 min


Deck of Cards (2022)

★ 9.1 | 🕒 45 min

The Lord of the Rings: The Return of the King (2003)

★ 9.0 | 🕒 201 min

3. Clusters (this tab is from data scientist perspective, to simply view the data cluster wise)



Movie Recommender System

[Insights](#)[Movies Explorer](#)[Clusters](#)[Recommendations](#)

Explore Movies by Cluster

Select Cluster ID (KMeans):

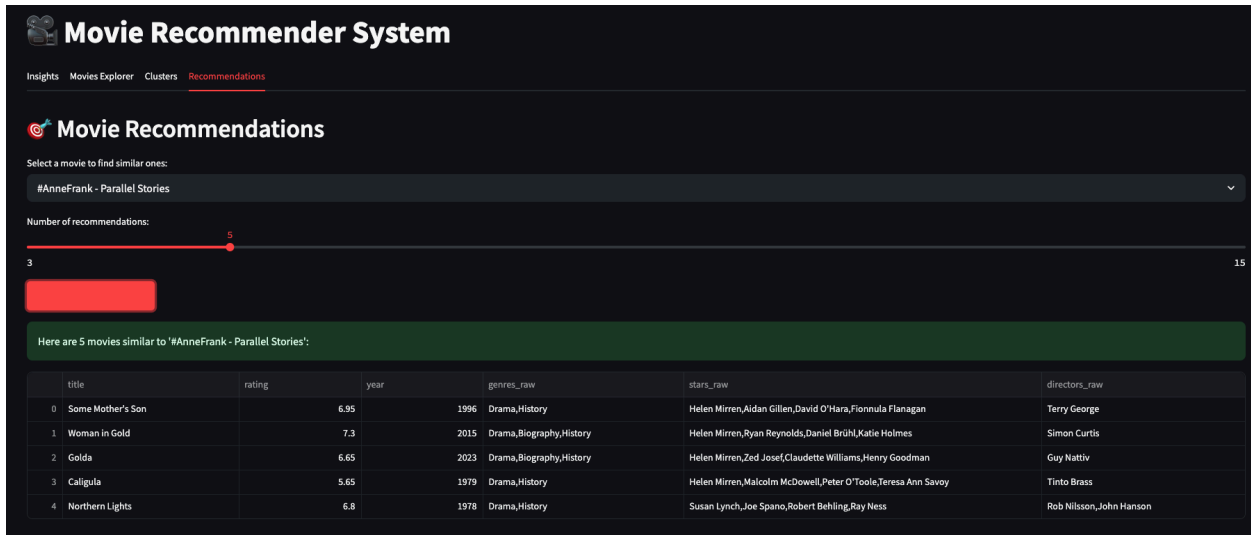
0

Found 1846 movies in cluster 0.

📄 🔍 🔄

	title	rating	year	genres_raw
0	Aviator	9.3	2016	Action,Drama,Thriller
1	Archie	9.05	2015	Drama,Thriller,Family
2	American Jihadist	8.75	2017	Drama,Thriller,History
3	The Silence of the Lambs	8.45	1991	Crime,Drama,Thriller
4	The Departed	8.35	2006	Crime,Drama,Thriller
5	Joker	8.3	2019	Crime,Drama,Thriller
6	No Country for Old Men	8.05	2007	Crime,Drama,Thriller
7	Room	8.05	2015	Drama,Thriller
8	The Imitation Game	8	2014	Drama,Thriller,Biography
9	On the Waterfront	8	1954	Crime,Drama,Thriller

4. Recommendation (User can choose a movie which they like and they will get the recommendation using KNN)



The screenshot shows a web application titled "Movie Recommender System". It has a navigation bar with "Insights", "Movies Explorer", "Clusters", and "Recommendations" (which is highlighted). Below the navigation bar, there's a section titled "Movie Recommendations" with a sub-header "Select a movie to find similar ones:". A dropdown menu shows "#AnneFrank - Parallel Stories". Below this, there's a slider for "Number of recommendations:" ranging from 3 to 15, with a red dot at 5. A red button is visible below the slider. Below the button, a green box says "Here are 5 movies similar to '#AnneFrank - Parallel Stories':". Below this, there's a table with 5 rows of movie recommendations.

	title	rating	year	genres_raw	stars_raw	directors_raw
0	Some Mother's Son	6.95	1996	Drama,History	Helen Mirren,Aidan Gillen,David O'Hara,Fionnula Flanagan	Terry George
1	Woman in Gold	7.3	2015	Drama,Biography,History	Helen Mirren,Ryan Reynolds,Daniel Brühl,Katie Holmes	Simon Curtis
2	Golda	6.65	2023	Drama,Biography,History	Helen Mirren,Zed Jozef,Claudette Williams,Henry Goodman	Guy Nattiv
3	Calligula	5.65	1979	Drama,History	Helen Mirren,Malcolm McDowell,Peter O'Toole,Teresa Ann Savoy	Tinto Brass
4	Northern Lights	6.8	1978	Drama,History	Susan Lynch,Joe Spano,Robert Behling,Ray Ness	Rob Nilsson,John Hanson

Note: How Genre, Stars are aligning to certain extent. So our recommendations do make sense too !!!

Scope and Dataset Considerations

The datasets used in this project, although extensive and sourced from trusted platforms, exhibit a noticeable imbalance—particularly in terms of language distribution. A significant majority of the entries are English-language movies, while non-English movies are underrepresented.

To ensure consistency, relevance, and better model performance, the scope of this project has been deliberately limited to English-language movies. This decision allows for more focused data processing and ensures that recommendations are generated from a sufficiently large and homogenous subset of data.

Conclusion:

Even though my goal was to build a recommendation system, I tried to explore different models. Such as classification and regression too. The UI provides a clean and simple UI for users to interact with the dataset. I also took

care of data load time by using cache in streamlit, which enhanced the performance of all queries and recommendations.