

**Machine Learning - Project Report Document**

|  |  |
| --- | --- |
| **Student Name** | Jawahar Jovah Koppala |
| **Batch** | Ai Elite 21 |
| **Project Name** | Movie Recommendation System |
| **Project Domain** |  |
| **Type of machine learning** | Supervised machine learning |
| **Type of problem** |  |
| **Project Methodology** | CRISP-ML(Q) |
| **Stages Involved** |  |

# Stage-1 : Business Understanding

The objective is to provide our users with a seamless and enjoyable content discovery experience that not only meets their viewing preferences but also introduces them to new genres, directors, and films they may not have considered. This will contribute to the platform's overall growth and success in a saturated market.

In the competitive landscape of the entertainment industry, streaming platforms are constantly seeking ways to enhance user engagement and retention. A significant challenge faced by these platforms is the overabundance of content available, which can overwhelm subscribers and lead to decision paralysis. Users often struggle to discover new movies that align with their preferences, resulting in reduced satisfaction and increased churn rates.

**Business constraints:**

**Budget Constraints**: Limited financial resources for development, maintenance, and marketing of the recommendation system may restrict the scope of the project, including technology choices, data acquisition, and talent hiring.

**Time Constraints**: Tight timelines may limit the extent of research and development, affecting the thoroughness of algorithm testing, system deployment, and refinement based on user feedback.

**Data Availability and Quality**: The effectiveness of the recommendation system relies heavily on access to high-quality, relevant data. Constraints may arise if data is limited, outdated, or if privacy concerns restrict access to user data.

**User Experience and Acceptance**: Recommendations must not only be accurate but also align with user preferences in terms of interface design and interaction methods. Constraints may arise if users are resistant to algorithm-driven suggestions or if they have particular viewing habits.

**Ethical Considerations**: Businesses need to be mindful of the ethical implications of recommendations, ensuring that they promote diverse content and avoid reinforcing biases in user suggestions.

**Data Availability and Quality**: The effectiveness of the recommendation system relies heavily on access to high-quality, relevant data. Constraints may arise if data is limited, outdated, or if privacy concerns restrict access to user data.

**Technical Constraints**: Existing infrastructure may impose limitations on the deployment of new algorithms. Compatibility issues with current systems, scalability challenges, and resource limitations of computing power might need to be addressed.

# Stage-2 : Data Collection and Understanding

**Data Collection:**

I sourced data from various reputable online platforms and public databases that provide detailed information on movies, including ratings, reviews, and user preferences.

I carefully selected data sources that align with the goals of my project, focusing on platforms that offer rich user engagement metrics and critical reviews.

**Data Understanding:**

**Movie Identification**:

imdb\_title\_id: A unique identifier for each movie, presumably from IMDb.

title: The title of the movie.

original\_title: The original title of the movie, which might be different if the movie is known by different names in different regions or languages.

**Movie Release Information**:

year: The year the movie was released.

date\_published: The date the movie was released.

**Movie Attributes**:

genre: The type or category of the movie (e.g., action, comedy, drama).

duration: The length of the movie in minutes or hours.

country: The country where the movie was produced.

language: The primary language spoken in the movie.

**Movie Crew**:

director: The person who directed the movie.

writer: The person(s) who wrote the screenplay or script.

production\_company: The company responsible for producing the movie.

**Movie Cast**:

actors: The main actors or cast members in the movie.

Movie Description and Reception:

description: A brief summary or plot description of the movie.

avg\_vote: The average rating given to the movie by viewers or critics.

votes: The total number of votes or ratings the movie has received.

reviews\_from\_users: Reviews or comments left by users about the movie.

reviews\_from\_critics: Professional reviews of the movie from critics.

**Financial Information**:

budget: The amount of money allocated to produce the movie.

usa\_gross\_income: The total amount of money the movie earned in the United States.

worlwide\_gross\_income: The total amount of money the movie earned worldwide.

**Critical Reception**:

metascore: A score representing the critical consensus of the movie, based on reviews from professional critics.

# Stage-3 : Data Preparation

1. **Data Collection**

Source Identification: Identify and gather data from reliable sources (e.g., databases, APIs, CSV files).

Data Acquisition: Collect the data, ensuring to document the source and method of collection.

2. **Data Inspection**

Initial Exploration:

Load the dataset and check its structure using functions like head(), info(), and describe().

Identify the features (columns) and their types (e.g., numeric, categorical).

Copy

3. **Data Cleaning**

Handling Missing Values:

Identify missing values using isnull().sum() or info().

Decide on strategies for handling missing data (e.g., removal, imputation).

4. **Data Transformation**

Changing Data Types: Convert columns to appropriate data types (e.g., object to int or float).

Normalization / Scaling: Standardize or normalize numeric features if required.

Feature Engineering: Create new features that can provide additional insights or improve model performance.

5. **Data Integration**

Combining Datasets: If applicable, merge or join multiple datasets to create a comprehensive dataset.

7. **Data Splitting**

Train-Test Split: If preparing for machine learning, split the dataset into training and test sets.

Copy

# Stage-4 : Model Building

**Methodology**

**Collaborative Filtering**

**Library Used:**

The Surprise library was used for implementing collaborative filtering algorithms.

The Surprise library is a Python framework designed specifically for building and evaluating recommender systems. It is user-friendly, highly customizable, and offers a variety of tools that simplify the implementation of collaborative filtering algorithms

**Model Selection**:

Various collaborative filtering algorithms were tested, including:

SVD (Singular Value Decomposition)

KNN (K-Nearest Neighbors)

NMF (Non-negative Matrix Factorization)

**Training and Testing:**

The dataset was split into training and testing sets (80% train, 20% test).

Each algorithm was trained on the training set and evaluated using RMSE on the test set.

**Evaluation:**

The performance of each algorithm was assessed based on Root Mean Squared Error (RMSE) scores.

Best combination of the parameter that achieve max performance :

{'algo': 'KNNBaseline', 'name': 'cosine', 'user\_based': False}

with RMSE: 0.6962051487245362

**Hyperparameter Tuning with Optuna:**

Optuna was used to automatically search for the best hyperparameters for the selected algorithms.

**Content-Based Filtering:**

Technique Used: TF-IDF Vectorization and Cosine Similarity.

Data Preparation:

The item descriptions were processed using Tfidf Vectorizer to convert them into a matrix of TF-IDF features.

Similarity Calculation:

Cosine similarity scores between items were computed using the TF-IDF feature matrix.

Recommendation Generation:

A function was developed to return the top N items similar to a given item based on cosine similarity scores.

# Stage-5 : Model Training

For the recommendation system, we utilized three distinct DataFrames, each containing essential information relevant to the user-item interactions and item characteristics. The combined dataset allows us to capture richer contextual information that can enhance the model's performance

This DataFrame provides additional contextual information about the items, which can be used for content-based filtering and enriching user-item interactions.

We started by merging the User Interaction DataFrame with the Item Metadata DataFrame using the itemId column. This operation enriched the user-item interaction data with descriptive features of the items being rated.

This merged DataFrame retains all the user interactions while incorporating relevant item details.

The model is trained on these columns:

|  |  |
| --- | --- |
| COLUMNS | DType |
| 'imdb\_title\_id | object |
| 'title | object |
| 'original\_title | Object |
| year | int |
| 'date\_published | object |
| 'genre' | object |
| 'duration' | int |
| 'country' | object |
| 'language' | object |
| 'director' | object |
| 'writer' | object |
| 'production\_company | object |
| 'actors' | object |
| 'description' | object |
| 'avg\_vote | float |
| 'votes' | float |
| 'budget' | object |
| usa\_gross\_income | int |
| worlwide\_gross\_income | int |
| 'metascore' | float |
| reviews\_from\_users | float |
| reviews\_from\_critics | float |

# Stage-6 : Model Evaluating

**Root Mean Squared Error (RMSE):**

RMSE measures the average error between predicted ratings and actual ratings for user-item interactions. It provides insights into how well the model can predict user ratings.

Formula:  
 RMSE=1n∑i=1n(yi−y^i)2\text{RMSE} = \sqrt{\frac{1}{n}\sum\_{i=1}^{n}(y\_i - \hat{y}\_i)^2}RMSE=n1​i=1∑n​(yi​−y^​i​)2​

where yiy\_iyi​ is the actual rating, y^i\hat{y}\_iy^​i​ is the predicted rating, and nnn is the number of predictions.

**Mean Absolute Error (MAE):**

MAE quantifies the average absolute difference between predicted ratings and actual ratings. It is less sensitive to outliers than RMSE and provides a straightforward interpretation of the average error.

Formula:  
 MAE=1n∑i=1n∣yi−y^i∣\text{MAE} = \frac{1}{n}\sum\_{i=1}^{n}|y\_i - \hat{y}\_i|MAE=n1​i=1∑n​∣yi​−y^​i​∣

**Cross-Validation:**

Additionally, k-fold cross-validation was implemented to ensure that the evaluation results were robust and had minimal variance across different splits of the data. This process involves repeatedly splitting the data and performing training and evaluation multiple times, obtaining average values for the performance metrics.

Conclusion:

A lower RMSE and MAE indicate good predictive accuracy.

Precision and recall scores show a balanced and decent performance; however, there may still be room for improvement, especially in recall. This suggests that while many of the recommended items are relevant (high precision), some relevant items may not be included in the top recommendations (lower recall).

# Stage-7 : Model Deployment

Deploying a recommendation model using Flask involves a series of structured steps to ensure that the model is accessible as a web service. This allows users or applications to request recommendations dynamically based on their input.

Setting Up the Flask Environment

To start, you need a working environment for Flask. This can be done using a virtual environment to isolate dependencies.

Create a Virtual Environment:

python -m venv venv

venv\Scripts\activate

The deployment of the recommendation system using Flask provides a robust platform for serving personalized recommendations to users. The application is designed for scalability, security, and ease of use, ensuring that it can effectively meet user needs while being maintainable over time.

# Chanllenges Faced:

**Incomplete Data:**

Missing values can hinder the training process, leading to inaccurate recommendations. It's crucial to have robust methods for handling missing data, such as imputation or ignoring certain records.

**Inconsistent Data:**

Data collected from different sources may have inconsistencies in formatting, units, or representation, which can complicate the preprocessing steps.

**Insufficient Data:**

A lack of sufficient user interaction data can lead to poor model performance, especially for new users or items (the "cold start" problem).

**Choosing the Right Algorithm:**

Selecting an appropriate recommendation algorithm (collaborative filtering, content-based filtering, hybrid models, etc.) can be challenging, requiring experimentation and evaluation.

**Hyperparameter Tuning:**

Finding the optimal hyperparameters for the chosen algorithm can be time-consuming and may require extensive cross-validation.

**Bias in Recommendations:**

Recommendations may inadvertently favor popular items or reinforce existing biases, leading to a lack of diversity in suggestions. Ensuring fairness and addressing biases is essential for user satisfaction.

**User Preferences**:

Understanding and accurately representing user preferences can be difficult, especially if user behavior is noisy or inconsistent.

# Conclusion:

A successful recommendation system can significantly increase user satisfaction and engagement by providing personalized experiences tailored to individual preferences. By leveraging appropriate algorithms and emphasizing data quality, developers can create a system that not only meets user needs but also adapts to their evolving preferences over time.

The development and deployment of a recommendation system is both an exciting and complex undertaking. Throughout the process, various challenges such as data quality, model selection, scalability, algorithmic bias, and user engagement can arise, potentially impacting the system's performance. However, overcoming these challenges is essential for delivering a robust and effective recommendation service that enhances user experience.