MOVIE RECOMMENDATION SYSTEM

Submitted by:
Devanjali Patel
Chetan Bisht

BE Third Year

CSE/COE

Submitted to:

Dr. Anjula Mehto

Assistant Professor

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering

Department Thapar Institute of Engineering

and Technology, Patiala

November 2024

TABLE OF CONTENTS

Introduction

In an era where users are inundated with countless choices, helping them find the most relevant content is both a challenge and an opportunity. Recommendation systems have emerged as powerful tools to address this problem, enabling businesses to offer personalized experiences that enhance user satisfaction and engagement.

The goal of this project is to develop a **Collaborative Filtering-based Movie Recommendation System** that addresses the unique preferences of each user by predicting how they would **rate movies** they have **not yet interacted with**.

By leveraging the **patterns in user behaviour** and **movie ratings**, the system dynamically recommends films tailored to individual tastes, creating a more engaging and relevant experience.

Key Objectives:

1. **Personalized Recommendations:** Provide users with movie suggestions based on their preferences and similarities with other users.
2. **Predict User Ratings:** Estimate the ratings users would give to movies they have not rated, using collaborative filtering techniques.
3. **Optimize Prediction Accuracy:** Minimize the error between predicted and actual ratings by focusing on metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE).

Why Collaborative Filtering?

Collaborative filtering thrives on the collective knowledge of user **interactions**, enabling it to uncover **hidden** relationships between **users** and **items** without relying on explicit features of movies. This makes it a robust approach for creating personalized recommendations, even in diverse datasets.

Business Value:

A well-designed recommendation system not only **improves user satisfaction** but also drives higher **engagement**, **retention**, and **conversion rates**. For platforms in the entertainment industry, such as Netflix or Amazon Prime Video, an effective recommender engine can significantly **enhance** user experience and overall business performance.

This project serves as a stepping stone toward harnessing the potential of machine learning to solve real-world problems, providing insights into user behaviour while showcasing the power of data-driven decision-making.

Problem Statement

With millions of movies available, users often struggle to find content that matches their preferences, leading to decision **fatigue** and **reduced** engagement. This project aims to develop a **Collaborative Filtering**-based **Movie Recommendation System** that predicts user ratings for unseen movies and delivers accurate, **personalized** recommendations, optimizing metrics like **RMSE** and **MAPE** to enhance user satisfaction and platform engagement.

Overview of the Dataset used

MOVIE LENS (20M)

- **Source:** Obtained from Grouplens.
- **Link:** **https://grouplens.org/datasets/movielens/20m/**

- **Description:**
  - 5-star ratings and free-text tagging activity from the **MovieLens** platform.
  - Contains **20,000,263 ratings** and **465,564 tag applications** for **27,278 movies**.
  - Data from **138,493 users**, spanning **January 9, 1995, to March 31, 2015**.
  - Generated on **October 17, 2016**.

- **User Selection:**
  - Users were randomly selected.
  - Each user has rated at least **20 movies**.
  - No demographic data is included; users are identified only by a unique ID.

- **Files Included:**
  - genome-scores.csv
  - genome-tags.csv
  - links.csv
  - **movies.csv**
  - **ratings.csv**
  - tags.csv

- **Files Used for Objective:**
  - **ratings.csv**: Contains User ID, Movie ID, Rating, and Timestamp.
  - **movies.csv**: Contains Movie ID, Title, and Genres.

Project Workflow

1) Problem Understanding & Objective Definition:

The project begins by understanding the problem at hand: helping users discover movies tailored to their preferences by building a **Collaborative Filtering-based Movie Recommendation System**. The key objectives include:

- Predicting user ratings for unseen movies.
- Providing personalized recommendations.
- Minimizing prediction errors using metrics like **RMSE** and **MAPE**.

2) Data Collection and Understanding:

The dataset used for this project is the **MovieLens 20M Dataset**, obtained from Grouplens. It consists of **20 million** ratings for **27,278** movies by **138,493** users**.**

- The two primary files used:
    - **ratings.csv**: Contains user IDs, movie IDs, ratings, and timestamps.
    - **movies.csv**: Contains movie IDs, titles, and genres.
- Understand the data structure, scope, and limitations (e.g., no demographic details, minimum of 20 ratings per user).

3) Data Preprocessing

To ensure the dataset is clean and ready for modelling, the following steps are performed:

1. **Data Cleaning:**
    - Remove duplicates or inconsistencies (if any).
    - Drop unnecessary columns.
2. **Handling Missing Data:**
    - Check for null values and handle them appropriately (e.g., imputation or removal).
3. **Exploratory Data Analysis:**

    - **Univariate Analysis**: Analysing each feature individually.

- **Train & Test Splitting**: Splitting the data into **train(80%)** and **test(20%)** sets before proceeding towards further EDA and Feature Engineering.
- **Bi-variate Analysis:** Analysing multiple features together to discover relations, correlations and patterns:

  i) Analysing the **Distribution of Ratings.**
  ii) Analysing the **Number of Ratings with Date.**
  iii) Analysing the **Average Ratings by Date**
  iv) Analysing the **Ratings given by the User**.
  v) Analysing the **Ratings given to the Movies**.

4. <u>Feature Engineering:</u>

- Creating **Sparse** Matrices: **a) USER_ITEM b) USER-USER c) ITEM-ITEM** matrices to compute the similarities (**COSINE SIMILARITY**)
- Calculating **Average Rating** for Users(1) and Movies(0) from user-movie sparse matrix.
- Computing Item-Item Similarity Matrix to generate movies similar to a given movie.
- Computing the USER_USER SIMILARITY MATRIX to generate users similar to the given user.

5. <u>Feature Extraction:</u>

- Create a **sample** of the actual population, due to computational constraints.
- Create Sample Sparse Matrix for Train Data
- Create Sample Sparse Matrix for Test Data
- Create features for each row and save to list, using sample train data.
- Create features for each rows and save to list for test data.
- Create the **Pandas Data Frame** from the data rows extracted from the sparse matrix for train and test set.
- Transform data for training and testing. For the test data, we just require a tuple (user, item, rating).

4) <u>Model Building</u>

- We will try to build a regression model to predict the rating given by an user to a movie based on the generated features.
- We have two Error Metrics:
- **RMSE**: **Root Mean Square Error**: RMSE is the error of each point which is squared. Then mean is calculated. Finally root of that mean is taken as final value.
- **MAPE**: **Mean Absolute Percentage Error**: The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method.
  - The difference between At and Ft is divided by the actual value At again. The absolute value in this calculation is summed for every forecasted point in time and divided by the number of fitted points n. Multiplying by 100% makes it a percentage error.
  - where At is the actual value and Ft is the forecast value.
- **Train/test Splitting**: We can split the data for train/test and segregate the independent and dependent features.
- **Model Fitting**: Fitting various models and checking their accuracy:

BASELINE MODELS:

- **<u>BaselineOnly:</u>**

  1. Predicts ratings using **global averages**, **user biases**, and **item biases**.
  2. Serves as a benchmark for assessing the improvement achieved by advanced models.
  3. Evaluation focuses on **RMSE** and **MAPE** for train and test datasets.

COLLABORATIVE FILTERING MODELS:

- **<u>KNNBasic (User-User and Item-Item):</u>**

  1. User-User and Item-Item similarity-based collaborative filtering using the Surprise library.
  2. Computes similarity matrices to recommend items based on **nearest neighbours**.

3. Results show the relative performance of user and item similarities.

- **KNNBaseline:**

1. Enhances KNNBasic by incorporating baseline estimations to adjust predictions.
2. Better **handles sparsity** by combining collaborative filtering with bias-based adjustments.

MATRIX FACTORIZATION MODELS:

- **SVD (Singular Value Decomposition):**

1. Decomposes the user-item matrix into **latent** factors for users and movies.
2. Captures **underlying patterns** in sparse data and predicts ratings.

- **SVD++:**

1. Extends SVD by incorporating **implicit** feedback, such as user preferences inferred from **unrated** movies.
2. Demonstrates improved accuracy compared to SVD.

HYBRID MODELS WITH XGBOOST:

- **XGBoost Variants:**

1. Combines features derived from collaborative filtering and matrix factorization techniques with handcrafted features.

- **XGBoost_BSL, XGBoost_KNN, and XGBoost_BSL+KNN:**

1. Use predictions from baseline, KNN, or both as input features.
2. Leverages XGBoost's gradient boosting capabilities to improve predictions.

- **XGBoost_FMF (Factorization Machine Features):**

1. Incorporates latent features from matrix factorization models as input.

- **XGBoost_BSL+MF:**

1. Combines baseline model outputs with matrix factorization features to achieve optimal performance.

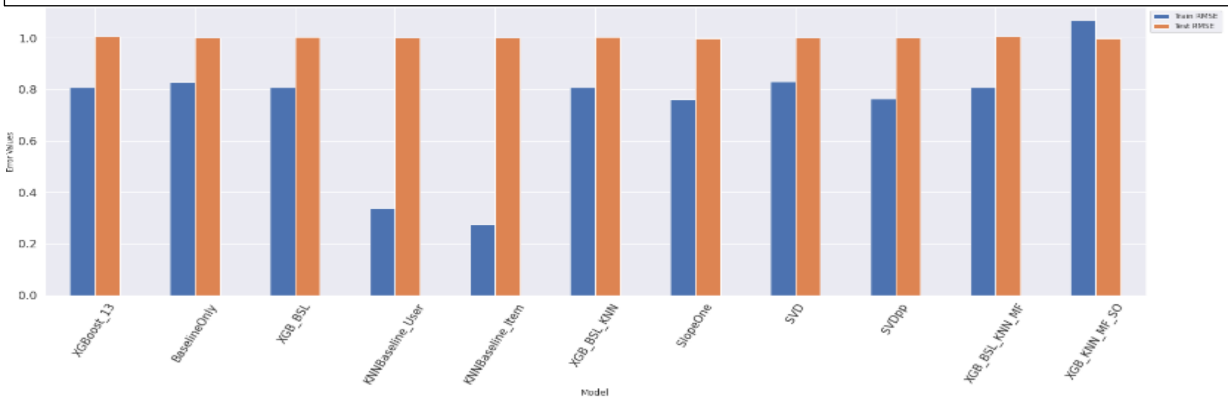
- **Generating Recommendation for Users:**

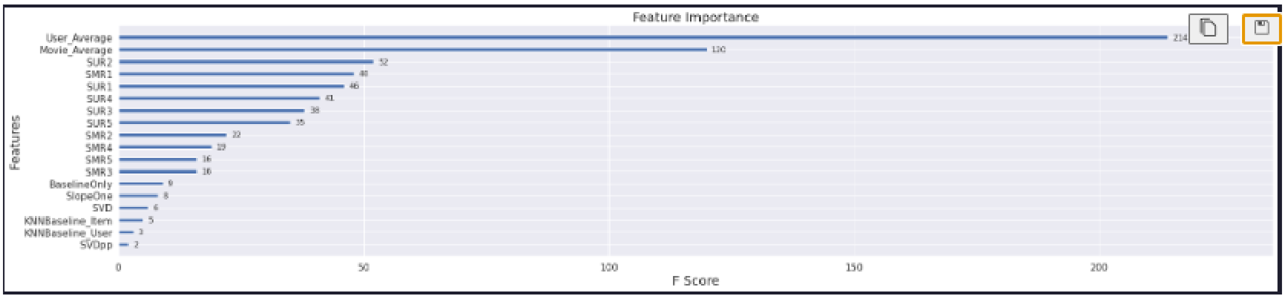
  We are using **SVDpp** to generate **atmost 10 recommendated movies** for various users, after having compared the RMSE, MAPE of all the models we used.

RESULTS

1. All the algorithms seem to do great with the differences remaining very **close** to each other.
2. We can see that by using various rating predicting algorithms together and **stacking** them up, then using final algorithms seems to result in **lowest Testing RMSE**. Eg: Surprise's BaselineOnly + KNN Baseline + SVD + SVDpp + SlopeOne together with Xgboost.
3. **SlopeOne** seems to have **lowest Testing RMSE** out of all other algorithms.
4. **SVDpp** and **SVD** are algorithms showing lower Testing RMSE among rest of the predictors except **SlopeOne**.

1) TEST AND TRAIN RMSE/ MAPE OF ALL MODELS





2) FEATURE IMPORTANCE

A COMPARISION OF THE RMSE (TRAIN, TEST) OF ALL THE MODELS USED:-

	Model	Train RMSE	Test RMSE
0	XGBoost_13	0.807228	1.004849
1	BaselineOnly	0.827371	0.999581
2	XGB_BSL	0.807346	1.003603
3	KNNBaseline_User	0.337843	0.999480
4	KNNBaseline_Item	0.274160	0.999480
5	XGB_BSL_KNN	0.807366	1.004184
6	SlopeOne	0.760572	0.999146
7	SVD	0.831105	0.999488
8	SVDpp	0.763126	0.999498
9	XGB_BSL_KNN_MF	0.807147	1.005231
10	XGB_KNN_MF_SO	1.066477	0.998114

The lowest Test RMSE is **0.998114**, achieved by **XGB_KNN_MF_SO (model index 10).** This indicates that this hybrid model likely has the best generalization performance on the test dataset.

However, **SVD++ (Singular Value Decomposition++)** combines the benefits of both **explicit feedback** (ratings) and **implicit feedback** (e.g., whether a user interacted with a movie). This makes it more effective in capturing user preferences, especially for sparse datasets.

THE FINAL OUTPUT:

```
sampled_user_id = list(top_n.keys())
```

```
# Generating recommendation using the user_Id

test_id = random.choice(sampled_user_id)
print("The user Id is : ", test_id)
Generate_Recommendated_Movies(test_id)
```

```
The user Id is :  55237
```

	Movie_Id	title	genres	Predicted_Rating
0	893	Apartment, The (1960)	Comedy|Drama|Romance	4.337311
1	2850	Lady Eve, The (1941)	Comedy|Romance	4.293874
2	213	Before the Rain (Pred dozhdot) (1994)	Drama|War	4.249252
3	2120	Shadow of a Doubt (1943)	Crime|Drama|Thriller	4.235916
4	3380	Hustler, The (1961)	Drama	4.224963
5	3711	Anatomy of a Murder (1959)	Drama|Mystery	4.219006
6	4312	Man Who Shot Liberty Valance, The (1962)	Crime|Drama|Western	4.190298
7	6902	Night of the Hunter, The (1955)	Drama|Film-Noir|Thriller	4.176537
8	4913	Witness for the Prosecution (1957)	Drama|Mystery|Thriller	4.132051
9	7045	Fog of War: Eleven Lessons from the Life of Ro...	Documentary|War	4.121651

CONCLUSION:

In this project, we explored the critical role of recommendation systems in modern applications and implemented various techniques to build an effective movie recommendation system. Through the project workflow, we delved into the following key areas:

1. **Techniques Used:**
    - Leveraged **user-user** and **movie-movie** similarities, global averages, and **collaborative filtering** techniques.
    - Integrated advanced methods like **matrix factorization** and hybrid models to enhance the system's accuracy.
2. **Prediction and Evaluation:**
    - Predicted movie ratings based on users' historical behavior and utilized error metrics such as **RMSE** and **MAPE** to evaluate model performance.
    - Demonstrated the effectiveness of combining traditional techniques with machine learning models like **XGBoost**.
3. **Application Scope:**
    - Showed the adaptability of these techniques to other domains involving user-item interactions, such as e-commerce and content platforms.
4. **Future Directions:**
    - Highlighted the potential for improvement by experimenting with additional machine learning and deep learning algorithms.
    - Opened avenues for incorporating advanced approaches like neural collaborative filtering, hybrid deep learning models, and real-time recommendation updates.

This project underscored the importance of a systematic approach to designing recommendation systems and provided a solid foundation for further exploration in the field.