“MuseMovie”
Movie Recommendation platform

Haoxuan Huang
Xiaoyue Chen
Yining Zhao

Table of Contents

1. Introduction

1.1. Background

The cold-start problem is a widespread issue that plagues most movie platforms. This occurs when new users sign up and have minimal or no viewing history on the platform, making it challenging to provide relevant movie recommendations based on their preferences and behavior. While some movie platforms ask users for their preferred genres and some users would respond to that, the recommendations they offer are typically very general and not tailored to individuals. As a result, these recommendations are not customized to the user's tastes and may not be helpful in suggesting movies they will enjoy. Accordingly, we try to come up with a new methodology to figure out these problems so that the platform can recommend the satisfied movies to new users with little information.

1.2. Project Goal and Objectives

We aim to build a movie platform that can deliver tailored movie suggestions to users based on their interactions and the existing users' interactions with the website, i.e. their viewing history and their ratings to the movies, which can be easily accessed. In this way, the platform can automatically recommend the customized movies to users and also keep updating even though the platform does not collect users' specific preferences. Additionally, the platform can timely detect the users' preference changes according to their view history and ratings, and make appropriate adjustments.

To achieve this, we propose using a machine learning model that aggregates the users' preference level to different movie tags and each movie ratings given by all users. The users' preference level is calculated by movie ratings and movie tag relevance from users. Finally, we can evaluate the model's performance by testing its recommendations on a sample of customer ratings. With our platform, new customers can expect a seamless, enjoyable movie-watching experience, complete with top three movie recommendations that perfectly align with interactions with the website.

2. Data

2.1. Data description

The dataset, in general, involves the user's rating, tags, and tag relevance of each movie. It contains 25 million ratings and 1 million tag applications across 62 thousand movies.

There are four major data file:

1) Movies: it contains the name and dataset ID of a movie, as well as its genres.
2) Rating: it contains the user ID, movie ID, and the rate (from 0.5 to 5.0) this user gives to the movie.
3) Tags: it contains the user ID, movie ID, and the tags this user gives to the movie.
4) Genome_score: it contains movie ID, tag ID, and the tag's relevance of the movie.

Additionally, there are two sub-datasets which are used to connect datasets:

1) Links: it contains movie ID IMDB movie ID, and TMDB ID.
2) Genome-tags: it contains tags ID and the name of the tags, which are written by customers with typos and punctuation characters.

In this project, we mainly use two data files, i.e. Rating and Genome_score. In the Rating file, it lists about 160,000 different users, about 60,000 movies, and about 25M movie ratings. In the Genome_score file, it lists about 13,000 movies, and each movie has 1128 tags relevance score. However, due to computing resource constraints, we created a small dataset with 1000 movies and 100 users (about 5000 records) for our project.

2.2. Data preprocessing

In general, there are two kinds of recommending algorithms: 1) directly calculate the preference of a certain user to a certain movie; 2) find a group of users who are similar to a certain user, and use the rating this group of users give to a certain movie as a reference. In this project, we mainly focus on the first one.

For calculating the preference of a user to a movie, we need three kinds of information: the tags of the movie, the quality of the movie, and the tags the user likes. It is easy to get the first two: for the tags of movies, we use the genome_score dataset to get the tag relevance; for the quality of the movie, we use the movie's average rating, calculated using ratings dataset, as a

reference. For the third one, we calculate the user's tag preference using the rating the user has given and the corresponding movies' tags.

Our assumption is: if the user likes the movie (give a high rating), then it is very likely that the user likes the tag of the movie, too; if the user dislikes a movie, then he or she might dislike the tags this movie contains. In calculation, we first scale the ratings from [0.5,5] to [-1,1]. Then, for each movie the user has rated, scaled rating * movie's tag relevance will be added to this user's preference score. For example, let's say there are only 5 tags. If a user watches a movie whose tag relevance are (0.5 0.5 1 1 0),and the rate he gives is 4 out of 5 (0.6 after scaling) then 0.6*(0.5 0.5 1 1 0) will be added to the user's preference score. An advantage is that the calculation mentioned above can be done very easily using matrix multiplication. For example:

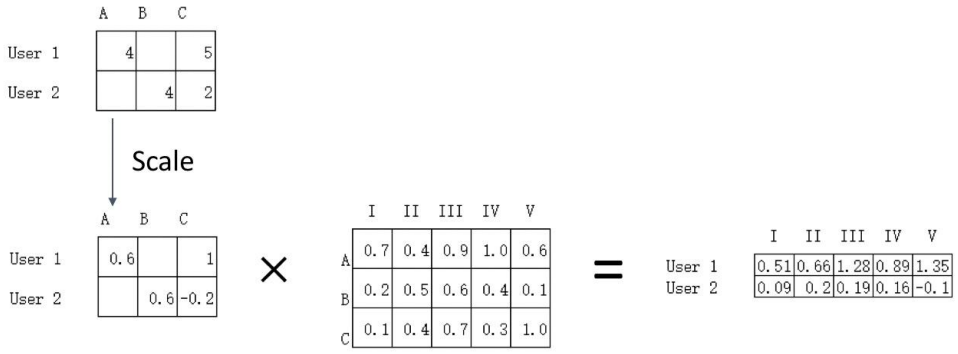

Figure 1: Matrix calculation of user preference

In this example, there are 2 users, 3 movies (A, B, and C), and 5 tags (I, II, III, IV, V). The matrix at left is the users' ratings to movies, the matrix at the middle is the movies' tags relevance, and the matrix at right is the users' preference matrix. We can say user 1 likes tag III, since its preference is 1.28; user 2 may hate tag V, since its score is negative.

To split the training and testing set, we first split all users into groups: 70% training and 30% testing. For training users, we use all the movies the users have rated to calculate the users' preference, and we use it to train the model. For testing users, we randomly masked 50% of the movies the users have rated (pretend that the users have not watched the movie before). We then use unmasked movies to calculate the users' preference, and use the masked movies to test our model.

After splitting the users and masking the data, we need to generate samples. As we have mentioned, our sample inputs contain three parts: movie's tag relevance, movie's rating, user's preference. First, we concate the movie's tag relevance vector and user's preference vector. Then, we use PCA to reduce the dimension. After that, we add the movie's average rating to the vector to finish the input. We use the rating the user gives to the movie as the label to train the models.

3. Modeling

The initial method employed in our study is Principal Component Analysis (PCA), a statistical procedure widely used for dimensionality reduction. The fundamental idea behind PCA is to transform the original variables into a new set of variables, the Principal Components, which are uncorrelated and capture the underlying variance in the data in decreasing order. For instance, Originally, our dataset was defined within a 2256-dimensional space, with a structure of (3500, 2256). Through PCA, we effectively reduced the dimensions of the data, resulting in a new structure of (3500, 200).

Our second modeling approach harnesses the power of the Random Forest algorithm, an ensemble learning method that constructs multiple regression trees and uses a bagging technique. This technique could increase the overall result's robustness by reducing variance without increasing bias.

The third modeling technique we utilize is XGBoost, short for Extreme Gradient Boosting. This is a machine-learning algorithm that leverages gradient-boosting frameworks on decision trees. XGBoost is designed for speed and performance, operating under the principle of boosting where weak learners are combined to create a strong predictive model. We employ XGBoost to predict user ratings, employing its power to iteratively correct the residuals of the previous model, enhancing its predictive accuracy with each step.

Upon applying PCA, we also account for the quality of the movies themselves. We pondered whether a film's poor performance could be due to its inherent quality rather than users' dislike for movies of a certain genre. To address this, we considered incorporating overall movie ratings provided by other users into our input data. This consideration was further explored when we applied our second technique, the Random Forest algorithm.

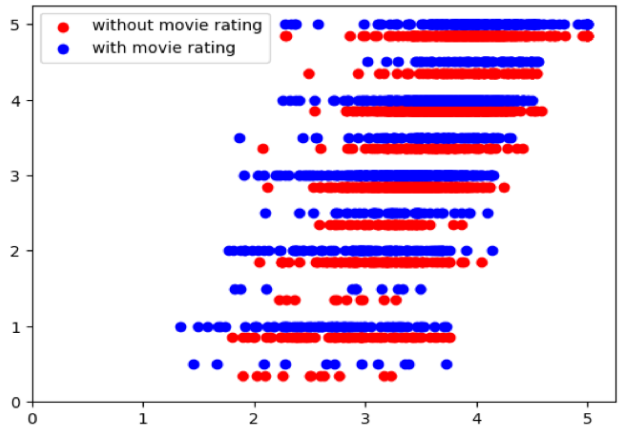

Figure 2: Result comparison of Random Forest with and without overall movie ratings

The scatter plot results indicated that the model, which used average rating information, outperformed the model that used original input data. This was evident in both low and high prediction rating regions, where the results were closer to actual ratings, and the model loss was minimized. Consequently, we decided to proceed with the input data including overall movie ratings to further compare our models.

Our third technique, XGBoost, was then put to the test and its results were compared to those of the Random Forest algorithm. According to the result in figure 3, the predictions from the XGBoost model were spread out, rendering them essentially meaningless, as each real rating corresponded to a prediction ranging from 0 to 5. This dispersion demonstrated that the model's results were less reliable than those of the Random Forest. With the Random Forest algorithm, a higher predicted rating always corresponded to a higher actual movie rating.

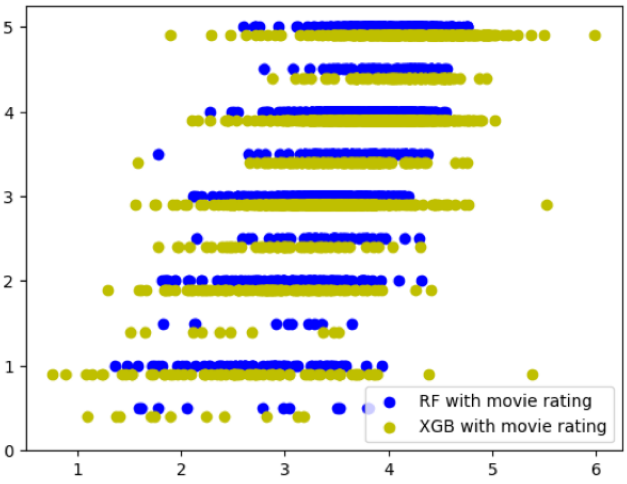

Figure 3: Result comparison of Random Forest and XGBoost

To summarize, our recommendation system operates on the principle of suggesting movies that our model predicts will have ratings higher than 4.5. This approach enables us to provide personalized recommendations to users. The thorough analysis of our models reveals that the use of the Random Forest algorithm for predicting high ratings aligns strongly with actual user ratings, ensuring the reliability and personal relevance of our recommendations.

We use Mean Square Error (MSE) as a metric of our models. The MSE values of models we mentioned above are about 0.11 in the testing set, thus we are confident that our model will produce reliable results.

4. Future Work

Looking forward, we have outlined several plans to further enhance the accuracy and reliability of our recommendation system:

1) Data Expansion: We plan to increase the size of our dataset. A larger dataset typically allows for more robust and generalizable model performance.
2) Refining Tag Relevance: We intend to fine-tune the relevance of movie tags. This could involve the use of techniques such as one-hot encoding, which could help us better differentiate and weigh larger on the higher influenced tag on a movie's rating.
3) Exploring Additional Machine Learning Models: We will explore the use of other machine learning models beyond PCA, Random Forest, and XGBoost. By testing a wider variety of models, we can potentially identify others that may yield superior predictive performance.
4) Hyperparameter Optimization: We will perform hyperparameter tuning on our models. This process involves adjusting the parameters of our machine learning algorithms to further improve their accuracy and fit the data, for example by further adjusting the forest size, tree depth, and boosting size. Hyperparameter tuning is an essential step in creating the most effective models and will help us ensure the highest possible quality of our recommendations.

5. Summary

This project addresses the cold-start problem in movie recommendation systems, which makes it difficult to provide new users with personalized movie suggestions. The proposed solution uses the viewing history and movie ratings of existing users to predict the preferences of new users. A machine learning model was developed that aggregates users' preference levels to different movie tags and ratings given by all users, using a dataset of 25 million ratings and 1 million tag applications across 62 thousand movies.

Our finalized recommendation system uses a personalized approach by prioritizing movies with predicted ratings exceeding 4.5. We first employed Principal Component Analysis (PCA) to reduce the dataset's dimensionality from 2256 to 200, mitigating issues associated with high-dimensional data. Then. We utilized the Random Forest algorithm, which utilizes bagging techniques to build multiple regression trees, leading to more robust results with minimal bias. Incorporating overall movie ratings from other users improved our model's performance, which reduced model loss and produced more accurate predictions. Although we experimented with XGBoost, the Random Forest algorithm showed more promising results, with higher predicted ratings consistently corresponding to higher actual movie ratings, demonstrating its reliability in predicting user preferences.

Furthermore, we check the Weapon of Math Destruction and model fairness. Our model is not a Weapon of Math Destruction. For example, the Opacity is not the case as the recommendation prediction process is quite clear, and it does not create self-fulfilling (or defeating) feedback loops. Even a not satisfied prediction will not have disastrous consequences on people. Besides, our machine learning model meets the legal requirement of fairness. Our company will give movie recommendations without discrimination or other limitations based on race, sex, color, religion, sex, disability, familial status, or nationality.