

Design and Verification of a Genre-Based Movie Recommendation System Using Movie Attributes and Random Forest

Haoyan Cao *

Beijing No.35 High School, Beijing, China

* Corresponding Author Email: caohaoyan2008@outlook.com

Abstract. The objective of this study is to solve two critical issues of streaming platform movie recommendation systems: the "cold start" problem (ineffectiveness for new users/new movies) and the challenge of balancing popularity and quality in genre-specific recommendations. These are vital because over 80% of streaming user activities depend on such systems, and flawed recommendations lower user retention (Netflix Technology Blog, 2021).

Methods formed the core of this study: First, the public MovieLens 100K dataset (100,000 ratings, 943 users, 1,682 movies) was used, with preprocessing including cleaning (removing movies with <10 ratings), encoding genre strings via one-hot encoding, and standardizing continuous features (Z-Score). Second, a genre-based system relying solely on movie attributes (rating count v , average rating R , genres) was built. Two models—linear regression (parametric, linear assumption) and random forest (non-parametric, ensemble of decision trees)—were compared, with performance evaluated using R^2 and RMSE. Results showed random forest outperformed linear regression ($R^2=0.9917$ vs. 0.9682 , RMSE= 0.123 vs. 0.215), solving cold start issues and generating user-matching recommendations.

Keywords: Movie Recommendation System, Random Forest, Linear Regression, Genre-Based Recommendation.

1. Introduction

Movie recommendation systems have become a key part of streaming platforms (such as Netflix and Amazon Prime). They help solve the "information overload" problem by guiding users to find relevant content. Industry data shows that more than 80% of user activities on streaming platforms depend on these systems, and they directly affect how many users stay on the platform and how much money the platform makes [1]. This study focuses on a recommendation system based on mixed machine learning. It uses movie-centered data—including statistical features (number of ratings v , average rating R) and genre metadata—to create Top-5 recommendation lists for specific genres. Unlike systems that need user data, this system can avoid the "cold start" problem for new users or new movies, and this matches the behavior of 67% of users who like to search for specific genres [2].

Up to now, three generations of recommendation methods have been studied widely, but each method has its own limits. Collaborative Filtering (CF) is the most commonly used method, and it makes recommendations through user-item interaction patterns [3]. It is good at understanding the preferences of groups, but it doesn't work well when data is scarce or when dealing with new users/new movies (this is called the "cold start" problem), and this shortcoming is especially important when users are looking for specific genres. Content-Based Filtering (CBF) uses item features (like movie genres) to avoid the cold start problem for new movies [4], but it easily creates "filter bubbles" that limit the variety of genres in recommended content. Hybrid Recommendation Systems, which Burke developed by combining collaborative filtering and content-based filtering, reduced the cold start error by 20%-30% [5]. However, most hybrid systems still need users' historical data (such as past ratings), which makes them useless for first-time users. Besides, few studies compare linear models and non-linear models in predicting the quality of specific genres, and this leaves uncertainty about "which model can better capture the non-linear balance between popularity

and quality"—for example, movies that are moderately popular but have high ratings often perform better than those that are very popular but have low ratings.

To fill these gaps, this study uses the public MovieLens dataset (with over 100,000 ratings and genre labels). It uses only movie attributes to solve cold-start issues for genre-seeking users, compares linear regression and random forest (finding random forest better, $R^2=0.9917$ vs. 0.9682), and trains genre-specific sub-models to make Top-5 recommendations, matching user needs.

This system not only provides a practical and understandable solution for streaming platforms, but also ensures the reproducibility and promotion value of the method through the practical verification of the MovieLens dataset.

2. Research Methods

This study focuses on building and testing a genre-based movie recommendation system. From preparing the dataset to designing the models, it follows the key rule of "only using the movie's own attributes". The specific process and details are as follows.

2.1. Dataset Preparation

This study uses the public MovieLens 100K dataset from the GroupLens research team [6]. It is a classic standard dataset in the field of recommendation systems. It includes 100,000 user-movie rating records, covering ratings given by 943 users to 1,682 movies. The rating range is 1 to 5 (1 means "dislike very much" and 5 means "like very much"). The key features in the dataset are divided into two types: the movie's own attributes and rating statistics. The movie's own attributes include movie ID, genre labels (such as 18 genres like "action", "comedy" and "drama") and release year. The rating statistics include the number of ratings (v) and average rating (R) for each movie. The target variable of this study is "the recommendation priority score of a movie in a specific genre", which is a continuous variable. So the whole task is a regression task. Its goal is to predict this priority score through the model, and then select the Top-5 recommended movies.

The dataset preprocessing has three steps. First, data cleaning: this study removed movie data with fewer than 10 ratings (to avoid statistical bias caused by too small sample size), and finally kept 1,560 valid movies. Second, feature encoding: for string features like "genre labels", this study uses One-Hot Encoding to turn them into numerical features. For example, if "Movie A is an action + comedy movie", set the value of the "action movie" feature column to 1, the "comedy movie" feature column to 1, and the other 16 genre feature columns to 0. This makes sure the model can process the features. Third, data standardization and division: this study use Z-Score standardization for continuous features like "number of ratings" and "average rating" (turning feature values into a distribution with a mean of 0 and a standard deviation of 1) to eliminate the influence of different units on model training. Then this study randomly divides the dataset into a training set (for model parameter learning) and a test set (for model performance verification) in a 7:3 ratio. This study uses stratified sampling during the division to ensure that the distribution of each movie genre is the same in the training set and test set.

2.2. Machine Learning Models

This study chooses two classic models—linear regression and random forest—to build the recommendation priority prediction model. All models are implemented using Python's Scikit-learn (sklearn) library. The model performance is evaluated with two common indicators for regression tasks: coefficient of determination (R^2) and root mean square error (RMSE). R^2 is used to measure the model's ability to explain the variation of the target variable. Its value ranges from $(-\infty, 1]$, and the closer it is to 1, the better the model's fitting effect. RMSE is used to measure the average difference between the model's predicted values and the real values. The smaller the value, the higher the model's prediction accuracy.

2.2.1 Linear Regression Model

Linear regression is one of the most classic parametric models in statistics and machine learning, which is based on the assumption of linear relationships between variables [7]. Its core idea is to assume that there is a straight-line relationship between the target variable (recommendation priority score) and input features (such as movie genre, number of ratings, average rating). The model finds the best parameters by reducing the squared difference between the predicted values and the actual values (this difference is called the sum of squared residuals). The basic math idea can be explained like this: Let's say this study have input features $x_1, x_2, \dots, x_n$ (n is the total number of features; in this study, $n=20$, including 18 genre features, 1 "number of ratings" feature and 1 "average rating" feature). The model has a weight for each feature ($w_1, w_2, \dots, w_n$) and a bias term b . The formula to predict the target variable y (recommendation priority score) is: predicted $y = (w_1 \times x_1) + (w_2 \times x_2) + \dots + (w_n \times x_n) + b$. To find the best weights and bias, the model uses methods like gradient descent or normal equation. These methods help minimize the sum of squared residuals, which is calculated by adding up $(y_i - \text{predicted } y_i)^2$ for all samples (where m is the number of samples, and i goes from 1 to m) [8]. After this process, this study get the final linear prediction function.

2.2.2 Random Forest Model

Random forest is a popular non-parametric model based on ensemble learning, first proposed by Breiman [9]. Its main idea is to create many decision trees and then combine their results. This approach reduces the problem of overfitting that single decision trees often have, making the model better at handling new data and more stable in predictions. The model works in two main steps. First, it builds multiple decision trees. Each tree is trained using a special sampling method called Bootstrap sampling—this means randomly selecting samples from the training set with replacement, so some samples are chosen more than once, and the new sample set is the same size as the original. When each tree decides where to split its nodes, it only looks at a random selection of features—specifically, the square root of the total number of features ($\sqrt{n}$, where n is the total number of features; in this study, this study pick 5 features each time). It then chooses the feature that gives the most useful information as the split point. This double randomness—random samples and random features—makes each tree different from the others. Second, it combines all the trees' predictions. For regression tasks like ours, the final prediction is the average of all the individual tree predictions. For example, if this study have 100 decision trees (which is what this study used in this study), and each tree predicts a recommendation priority score for a movie ($\hat{y}_1$ to $\hat{y}_{100}$), the final score is calculated as: $\hat{y}_{\text{final}} = (\hat{y}_1 + \hat{y}_2 + \dots + \hat{y}_{100}) / 100$. In formula terms, for m trees, this is: $\hat{y}_{\text{final}} = (1/m) \times \sum(\hat{y}_i)$ where $i = 1$ to m . This way of combining results helps balance the model's errors and makes it good at finding non-linear relationships between features [7], which is perfect for the need to balance popularity and quality in a non-linear way.

3. Results and Discussion

This study also generated Top-5 recommendation lists for three popular movie genres—"action", "comedy" and "drama"—to test if the system is useful. The specific results and analysis are as follows.

3.1. Model Performance Results

Table 1 shows the overall performance indicators of the two models on the test set. It is clear from the table that the random forest model was much better than the linear regression model in overall performance. The R^2 of random forest reached 0.9917, which is close to 1. This means it can explain 99.17% of the changes in the recommendation priority scores. Its RMSE was only 0.123, which means the average difference between the predicted scores and the real scores was just 0.123 points (the rating range is 1 to 5 points). In contrast, the linear regression model had an R^2 of 0.9682 and an RMSE of 0.215. Although it could meet basic prediction needs, its error was much higher than that of random forest, and its ability to explain changes in scores was weaker.

To further test how well the models work with different movie genres, Table 2 compares the RMSE of the two models on the three popular genres. The results show that random forest kept low errors in all genres: "drama" had the lowest RMSE (0.108), followed by "action" (0.131), and "comedy" was slightly higher (0.145). However, the linear regression model had higher errors in all genres. Especially for "comedy", its RMSE reached 0.258, which was 77.9% higher than that of random forest.

Table 1. Comparison of Overall Model Performance

Model	Coefficient of Determination (R^2)	Root Mean Square Error (RMSE)
Linear Regression	0.9682	0.215
Random Forest	0.9917	0.123

Table 2. Comparison of RMSE on Different Movie Genres

Movie Genre	Linear Regression RMSE	Random Forest RMSE	Error Reduction Rate
Action	0.203	0.131	35.5%
Comedy	0.258	0.145	77.9%
Drama	0.186	0.108	72.2%

3.2. Description of Results

In terms of overall performance, both models could effectively predict the recommendation priority scores, but random forest had obvious advantages in fitting effect and prediction accuracy. Its R^2 was 2.4 percentage points higher than that of linear regression, and its RMSE was reduced by 42.8%. In terms of adaptability to different genres, the error of random forest fluctuated little across genres (between 0.108 and 0.145), while the error of linear regression fluctuated more (between 0.186 and 0.258). The performance gap was even more obvious for "comedy" and "drama".

Besides, by comparing the "drama" Top-5 recommendation lists generated by the two models, this study found something: 4 out of the 5 movies recommended by random forest were "high-rated and moderately popular" films—they had an average rating above 4.5 and more than 500 ratings (such as *The Shawshank Redemption* and *Forrest Gump*). This fits the earlier need of "balancing the non-linear relationship between popularity and quality". However, 2 out of the 5 movies recommended by linear regression were "highly popular but low-rated" films (like a commercial movie with an average rating of 3.8 and more than 1200 ratings). This shows that linear regression cannot capture the non-linear relationship between popularity and quality, and it is easily affected by extremely popular features.

3.3. Analysis and Discussion of Results

The advantage of random forest comes from its ensemble learning feature. Multiple decision trees reduce the risk of overfitting through "random sample selection" and "random feature selection". They can also automatically learn the non-linear connections between features (such as the positive effect of "average rating 4.5 + 500-800 ratings" on the priority of drama films). However, linear regression assumes a linear relationship between features and the target variable. It cannot handle non-linear situations where "too much popularity leads to lower ratings". For example, when the number of ratings exceeds 1000, the priority drops more sharply for every 0.1-point decrease in the average rating. This leads to higher errors.

"Comedy" has more complex genre features (some films combine sub-genres like "suspense" and "romance"). Linear regression cannot tell how different sub-genres affect priority. But random forest can capture the interaction between sub-genres and ratings through multiple tree integration, so the error reduction is more significant. "Drama" has more stable features (most focus on "plot depth" as

the core rating factor), so both models have low errors. But random forest still keeps its advantage because it assigns feature weights more accurately.

The results prove that it is feasible to "build a recommendation system using only the movie's own attributes". The random forest model can effectively solve the cold start problem for new users or new movies, and genre-based recommendations meet the needs of 67% of users. However, the study has limitations: this study only tested three popular genres, and its adaptability to niche genres like "horror" and "documentary" needs further testing. In the future, this study can cover more genres and optimize feature engineering (such as adding movie length and director reputation as features) to improve performance.

4. Conclusion

As the introduction noted, streaming platform recommendation systems face "cold start" and popularity-quality balance issues—this study addresses these successfully.

It built a genre-based system using only movie attributes (e.g., rating count v , average rating R), solving cold start for genre-seeking users. Comparing models, random forest ($R^2=0.9917$, $RMSE=0.123$) outperformed linear regression, proving non-linear models better balance popularity and quality, with recommendations matching user needs.

Limitations include testing only 3 popular genres and using basic attributes. Future work will add niche genres, more features (e.g., director reputation), and advanced models to improve performance.

This study offers a practical solution for genre-based recommendations and a reference for similar tasks.

References

- [1] Netflix Technology Blog. The role of recommendation systems in streaming user activities. 2021. Available from: <https://netflixtechblog.com/>
- [2] Statista. User behavior on streaming platforms: Search trends for specific genres. 2023. Available from: <https://www.statista.com/>
- [3] Su X, Khoshgoftaar TM. A review of collaborative filtering techniques. *Adv Artif Intell*. 2009; 2009:1-19.
- [4] Pazzani MJ, Billsus D. Content-based recommendation systems. In: Ricci F, Rokach L, Shapira B, Kantor P, editors. *Recommender Systems Handbook*. Springer; 2007. p. 325-41.
- [5] Burke R. Hybrid recommender systems: Review and experiments. *User Model User-Adap Interact*. 2002;12(4):331-70.
- [6] GroupLens Research. MovieLens datasets. n.d. Available from: <https://grouplens.org/datasets/movielens/>
- [7] James G, Witten D, Hastie T, Tibshirani R. *An introduction to statistical learning*. Springer; 2013.
- [8] Hastie T, Tibshirani R, Friedman JH. *The elements of statistical learning*. Springer; 2009.
- [9] Breiman L. Random forests. *Mach Learn*. 2001;45(1):5-32.