Exploring Traditional Machine Learning Techniques: A Comprehensive Study

Rakshit Singhal Parth Darshan Akarsh Katiyar
Palash Khatod Mayank Bansal Ram Prasad

Indian Institute of Technology Jodhpur
{b22ee065, b22cs040, b22cs006, b22ee095, b22cs070, b22ee054}@iitj.ac.in

Abstract

This research comprehensively explores traditional machine learning techniques for personalized movie recommendation systems within the Pattern Recognition and Machine Learning (PRML) domain. Our study focuses on applying and rigorously evaluating collaborative filtering and content-based filtering methods, emphasizing systematic performance assessments across computational efficiency, cold-start handling, and recommendation quality. Among collaborative filtering approaches, techniques like Random Forest exhibited efficient training times while generating diverse movie recommendations, though the cold-start problem persisted. For content-based filtering, Naive Bayes demonstrated superior computational efficiency and consistent performance, partially mitigating cold-start issues but lacking versatility in recommendation variety. By conducting an in-depth analysis of these traditional techniques' strengths, limitations, and real-world applicability, we aim to provide valuable insights to researchers and practitioners seeking to develop robust and personalized movie recommendation systems.

Contents

Introduction

In today's era of information and technology, world full of entertainment and works overload we need recommendation systems to enhance the entertainment experience. Movie recommendation systems, in particular, has a significant importance in assisting film enthusiasts by providing personalized suggestions tailored to their preferences. With the vast number of movies available, manually searching and identifying suitable films can be a daunting and time-consuming task. This research aims to evaluate and analyze various approaches and algorithms that can be utilized in designing an effective movie recommendation system.
The code, datasets, and additional materials for this study are available at GitHub [1]

Importance of Movie Recommendation Systems

Integrating a movie recommendation system offers a comprehensive solution to improve user satisfaction and achieve commercial success. By presenting customized suggestions based on individual tastes and preferences, users receive an enhanced experience, resulting in heightened engagement and exploration of diverse content. Employing a personalized approach fosters user loyalty and yields tangible business advantages, such as increased user retention rates and a competitive edge within the market. Ultimately, a well-designed movie recommendation system is crucial in delivering value to users while effectively accomplishing business goals.

Approaches to Movie Recommendation

Movie recommendation systems can be broadly classified into two main categories: collaborative filtering and content-based filtering.

Collaborative Filtering

Collaborative filtering utilizes user activity patterns and preferences to suggest movies. It assumes that users with similar interests will prefer similar film. By analyzing historical data on user interactions and ratings, collaborative filtering techniques can identify movies that users with comparable preferences have enjoyed, enabling personalized recommendations.

Content-based Filtering

In contrast, content-based filtering recommends movies based solely on their attributes, such as genre, director, actors, and plot summary, regardless of user activity patterns. These techniques analyze the inherent features of movies and match them with user preferences, suggesting films that align with the user's interests based on their content.

By thoroughly examining and evaluating these approaches, we aim to determine the most effective and efficient strategies for providing personalized movie recommendations. Through systematic performance assessments, method comparisons, and innovative techniques, we aim to deliver valuable insights for researchers and practitioners in recommendation systems.

Dataset

We utilised two primary tables from the dataset to develop and evaluate our movie recommendation models: ”movies” and ”ratings”. The ”movies” table encompasses comprehensive information regarding various films, including their unique identifiers, titles, and associated genres. On the other hand, the ”ratings” table captures user interactions and preferences by recording user identifiers, movie identifiers, and corresponding rating values. The integration of these two tables proved instrumental in constructing and training our recommendation systems.

Movies Data

The ”movies” table serves as a repository of movie-specific information, with each entry consisting of the following attributes:

- **Movie ID**: A unique identifier assigned to each movie, facilitating unambiguous referencing and tracking within the dataset.
- **Title**: The movie's official title provides a descriptive label for identification.
- **Genres**: A pipe-delimited (|) list of genres associated with the movie, enabling categorization and analysis based on movie types.

Ratings Data

The "ratings" table records user interactions and preferences by capturing the following information:

- **User ID**: A unique identifier assigned to each user, allowing for the tracking of individual preferences and behaviour.
- **Movie ID**: The identifier of the movie being rated establishes a link between the "movies" and "ratings" tables.
- **Rating**: A numerical value representing the user's evaluation or rating of the movie, typically ranging from 1 to 5 stars or a similar scale.

By leveraging the combined information from these two tables, we can analyze user preferences, identify patterns in movie ratings, and develop recommendation models that suggest relevant movies to users based on their past interactions and preferences. This dataset serves as the foundation for our exploration and evaluation of various traditional machine learning techniques in the context of movie recommendation systems.

Dataset Preprocessing

Optimizing parameters to minimize sparsity in the final dataset is crucial in data preprocessing for recommendation systems. The objective of this optimization is to enhance the quality of the dataset to construct recommendation models. Through the process of fine-tuning parameters such as the minimum number of user votes and minimum number of movie votes, we methodically aim to identify the combination that produces the lowest sparsity score. The filtration procedure involves eliminating individuals and items that have insufficient ratings, resulting in a more condensed dataset. The final dataset obtained, which has been reduced in sparsity, provides a more robust foundation for training recommendation models, thereby improving their accuracy and efficacy.

While our goal is to discover the minimum sparsity score, it is crucial to comprehend the consequences of increased sparsity during dataset training. While it is true that increased sparsity might enhance the efficiency of model training in specific scenarios, it is imperative to maintain a delicate equilibrium. The concept of sparsity, which refers to the proportion of missing or zero values within a dataset, presents both benefits and challenges.

Increased sparsity in datasets offers numerous benefits for training models. First and foremost, this results in decreased computing expenses due to a reduction in the number of non-zero entries in the final dataset, thus optimizing computations throughout the training process. Moreover, increased sparsity enhances the models' ability to generalize by potentially reducing noise in the data and facilitating more effective adaptation to unfamiliar situations. Furthermore, optimization techniques exhibit faster convergence when applied to sparse datasets, leading to shorter training periods.

On the other hand, datasets with excessive sparsity pose challenges such as data sparsity and the cold start problem. The issue of data sparsity arises when there is an insufficient amount of information available for the model to discern significant patterns, thereby diminishing the overall quality of recommendations. This issue is exacerbated by the cold start problem, particularly for new users or items with a limited history of interaction. To address these challenges, we employ tactics such as establishing minimum thresholds for the quantity of votes per user and each movie. This approach ensures that while increased sparsity can enhance the efficiency of model training, it is crucial to prioritize the preservation of sufficient information to ensure accurate recommendations.

Cosine Similarity Matrix

A similarity matrix quantifies the likeness or dissimilarity between pairs of objects within a dataset. In the realm of recommendation systems, it serves as a vital tool to gauge the degree of resemblance between items or users based on specific features or interactions. For instance, in collaborative filtering, a user-item similarity matrix measures the similarity between users or items by examining their historical interactions or preferences. Similarly, in content-based filtering, an item-item similarity matrix assesses the similarity between items based on their attributes or characteristics.

The entries in a similarity matrix typically range from 0 to 1, with higher values indicating greater similarity. These matrices play a pivotal role in various aspects of recommendation systems, including generating personalized recommendations, clustering similar items, and uncovering underlying patterns in the data.

One commonly used measure of similarity is the cosine similarity, which is calculated as:

$$\cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \cdot |\vec{b}|}$$

where $\vec{a}$ and $\vec{b}$ are the vector representations of the two objects being compared, and θ is the angle between them. The cosine similarity ranges from -1 to 1, with 1 indicating that the vectors are identical, 0 indicating that they are orthogonal (completely dissimilar), and -1 indicating that they are diametrically opposed.

As we delve into different modeling approaches, we will encounter similarity matrices frequently, underscoring their significance in the realm of recommendation systems.

Modelling Strategies

Collaborative and Content-based filtering are the two primary approaches for movie recommendation systems. We explore various techniques under these two categories.

Collaborative Filtering

Collaborative filtering utilizes user activity patterns and preferences to recommend movies. It assumes that users with similar interests will prefer similar movies. We investigate the following collaborative filtering techniques:

K-Nearest neighbours (KNN)

The K-Nearest Neighbours (KNN) algorithm is employed for movie recommendation, leveraging cosine similarity to measure the angle between feature vectors in a high-dimensional space. This technique facilitates the identification of similar movies based on the proximity of their feature vectors, where smaller angles indicate higher similarity.

K-Means Clustering

The implemented approach leverages K-Means clustering on movie rating data to group users with similar preferences into clusters. The optimal number of clusters is determined using the elbow method and silhouette score evaluation techniques. For a given target user, the algorithm identifies other users within the same cluster, calculates the cosine similarity between their rating vectors, and recommends highlyrated movies from the most similar users. This collaborative filtering technique enables personalized movie recommendations tailored to individual user preferences by exploiting the clustering of like-minded users and rating similarities.

Logistic Regression (LR)

Logistic Regression (LR) is employed in movie recommendation systems to model the probability of a movie being preferred by a user based on various features. It estimates the likelihood of a binary outcome, such as whether a user will like or dislike a particular movie. By learning the relationships between input features and user preferences, LR can make personalized recommendations. This technique is particularly useful for scenarios where the relationship between features and preferences is assumed to be linear.

Singular Value Decomposition (SVD)

Singular Value Decomposition reduces the dimensionality of a user-item interaction matrix by identifying latent factors representing user preferences and item characteristics. This allows for personalized recommendations based on these factors by approximating the original matrix with lower-dimensional representations.

Random Forest

Random Forest is a versatile ensemble learning method used for movie recommendation systems. It operates by constructing a multitude of decision trees during training and outputting the mode of the classes (or mean prediction) of the individual trees for classification (or regression) tasks. In the context of movie recommendations, Random Forest can capture complex relationships between various movie features and user preferences. It's robust to overfitting and can handle large datasets with high dimensionality effectively.

Content-based Filtering

Content-based filtering recommends movies based on their attributes, such as genre, director, actors, and plot summary, regardless of user activity. We explore the following content-based techniques:

Naive Bayes (NB)

This movie recommendation system utilizes Naive Bayes and TF-IDF. It preprocesses data, filters ratings, and handles sparsity. Movie genres are converted into TF-IDF vectors, and a Multinomial Naive Bayes model is trained. High probabilities indicate strong genre associations, guiding recommendations towards movies with similar genres. Low probabilities suggest weaker connections, leading to recommendations with less apparent genre similarities. This approach enables personalized recommendations by emphasizing genre similarities.

Support Vector Machines (SVM)

Support Vector Machines (SVM) are powerful classifiers used in movie recommendation systems to separate movies into different classes based on their features. SVM aims to find the hyperplane that best separates the movies belonging to different classes in the feature space.

We explore the following SVM kernel functions:

- **Linear kernel**: Constructs a linear decision boundary to separate different classes of movies. It works well when the classes are linearly separable in the feature space.
- **Polynomial kernel**: Utilizes a non-linear decision boundary to capture complex relationships between movie features. This kernel is effective for handling non-linearly separable classes.
- **Radial Basis Function (RBF) kernel**: Employs a flexible kernel suitable for high-dimensional movie feature spaces. It can capture intricate relationships between features and is effective in scenarios where the decision boundary is not easily defined.

Analysis

Within this section, we do an in-depth examination of the system's performance and efficacy. The assessment is structured into three primary domains, each providing insights into distinct aspects of the system's functionality and usefulness. Initially, a Computational Load Analysis is performed to examine the computational resources necessary for the system to function optimally. Additionally, we analyse the Cold Start Recommendations to gain insights into the system's ability to properly address the complex task of offering recommendations for new users or objects. Finally, we conduct an Evaluation of Recommendation Quality, in which we evaluate the effectiveness and appropriateness of the recommendations using qualitative observations and user input.

Section 1: Computational Load Analysis

In this section, we analyze the time taken by the system to train its model under different computational loads. We quantify the training time under varying resource allocations and workload intensities to understand the system's efficiency in model training. By examining the training time under different conditions, we aim to optimize resource allocation and enhance overall system performance while minimizing the time required for model training.

Comparative Analysis of Training Duration

Firstly, we initiate our research by a comparison assessment of the training length exhibited by each model. The present analysis involves a comprehensive assessment of the time needed to train each model utilising the dataset. The aim of this research is to examine the duration of training for each model with the purpose of identifying disparities in computational efficiency and revealing any significant patterns that may arise.

Method	min (s)	max (s)	mean (s)	median (s)	std (s)
K-Nearest Neighbours	1.540439	2.482073	1.721206	1.616034	0.288284
K-Means	13.910237	18.587061	15.410927	14.397730	1.838509
Logistic Regression	1.200041	16.597044	3.010863	1.492741	4.782015
Singular Value Decomposition	1.639110	2.662864	2.027913	1.860608	0.382153
Random Forrest	1.493579	2.101848	1.732882	1.684045	0.188758
Naive Bayes	2.484485	3.255709	2.812323	2.768102	0.243505
Linear Kernel	56.903388	78.384806	65.693817	60.204885	9.344839
Polynomial Kernel	59.129344	61.250752	59.855321	59.807297	0.633901
Radial Basis Function Kernel	59.679954	63.186314	61.000746	60.792124	1.143934

Table 1: Statistics of Methods

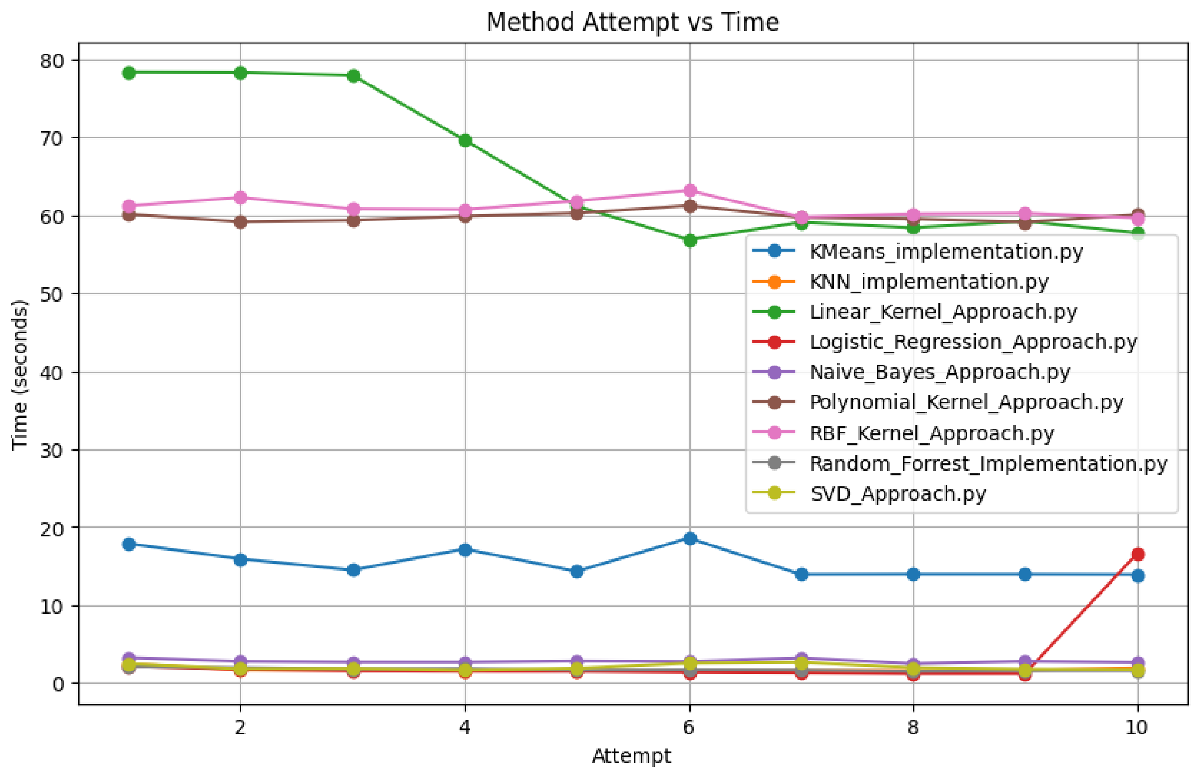

fig 1: Run Time

Key Insights

Upon analyzing the projected durations, several key insights emerge:

- **Performance Range**: The range of execution times varies significantly across different methods, indicating varying efficiency and sensitivity to different datasets.
- **Central Tendency**: Mean and median execution times provide insights into the typical performance of each method, with some methods exhibiting symmetric distributions while others show potential skewness.
- **Variability**: Standard deviation reflects the spread of data points around the mean, highlighting the consistency or variability of each method's performance.
- **Relative Efficiency**: Comparison of mean execution times reveals relative efficiency among methods, suggesting potential areas for optimization.
- **Outlier Detection**: Methods with unusually high maximum execution times may indicate the presence of outliers or extreme cases in the data.
- **Consistency vs. Complexity**: There appears to be a trade-off between consistency and complexity among the methods, with simpler models showing more consistent performance but potentially sacrificing accuracy.
- **Algorithmic Efficiency**: Some methods demonstrate algorithmic efficiency with consistently low execution times, indicating robust performance across different datasets.
- **Sensitivity to Data**: Variability in performance suggests differing sensitivities to changes in the dataset among methods.
- **Model Complexity Impact**: Execution times vary with model complexity, with more complex models generally requiring longer execution times.
- **Potential Optimization Targets**: Methods with longer execution times, higher variability, or wider ranges present opportunities for optimization to improve overall system performance.

Implications

Understanding the projected training durations of various models holds significant implications for the development of movie recommendation systems:

- **Model Selection**: The choice of model can significantly impact the development time of a movie recommendation system. Developers must weigh the trade-offs between model complexity, accuracy, and training time to select the most suitable approach.
- **Resource Allocation**: Longer training durations for certain models, such as those employing Kernel methods, may require additional computational resources or parallel processing to ensure timely model development and deployment.
- **Real-time Recommendations**: Models with shorter training durations, such as Naive Bayes or K-Nearest Neighbours, may be more suitable for real-time recommendation systems where responsiveness is crucial. However, developers must balance this with the need for accurate recommendations.
- **Scalability**: Understanding the scalability of different models is essential for accommodating growing datasets and user bases. Models with shorter training durations and lower computational complexity may scale more efficiently as the system expands.
- **User Experience**: The training duration indirectly impacts user experience, as longer development times may delay the rollout of new features or improvements to the recommendation system. Prioritizing models with shorter training durations can help enhance user satisfaction and retention.

Section 2: Analysis of Cold Start Recommendations

The primary focus is to assess the system's efficacy in offering suggestions for new users or objects, which is frequently referred to as the "cold start" challenge. The aforementioned crucial factor has a substantial impact on the system's capacity to produce precise and relevant recommendations in such circumstances.

Naive Bayes

This system mitigates the cold start problem by using movie genres for recommendations. Genre information allows suggesting movies for new items or users, offering a starting point despite limited interaction history.

Linear Kernel

The kernel partially handles cold-start problems by filtering users and movies based on their activity levels. However, it doesn't directly address cold-start issues for new users or new movies lacking genre information. Recommendations are primarily driven by genre similarity, potentially limiting effectiveness for new or unpopular items. Overall, while it offers some mitigation, it doesn't fully resolve cold-start challenges.

Polynomial Kernel

The system can handle the cold start problem to some extent by providing genre-based recommendations for new movies and general recommendations for new users, but it still requires some level of user interaction or historical data to provide more accurate and personalized recommendations.

Radial Basis Function Kernel

While this recommendation system incorporates movie genres and cosine similarity to mitigate cold-start issues, it remains challenged by the lack of data for entirely new users or movies absent from the dataset. While it can offer recommendations for users with sparse ratings based on genre similarities, it may struggle with entirely new movies not in the dataset. Therefore, while it partially addresses cold-start problems, it doesn't fully mitigate them, particularly in scenarios involving entirely new entities.

K-Means

This recommendation system might face cold-start problems. This approach relies on user-item ratings to form clusters and make recommendations. However, for new users who haven't rated any movies, the system cannot accurately assign them to a cluster or provide personalized recommendations. Thus, it doesn't effectively handle cold-start problems.

K-Nearest Neighbours

The movie recommendation model is prone to cold-start problems due to new movies lacking sufficient ratings for accurate recommendations and new users with limited data. Filtering out sparse data exacerbates this issue.

Random Forest

This approach can partially handle cold-start problems for users by predicting ratings for unrated movies based on existing user behavior. However, it doesn't address the cold-start problem for new movies, as it relies solely on existing movie ratings.

Singular Value Decomposition

Collaborative filtering cannot handle cold-start problems adequately. It relies solely on historical user-item interactions and doesn't account for new users or items with no prior ratings. For a new user or item, it cannot provide meaningful recommendations since they lack existing ratings in the dataset.

Logistic Regression

The Logistic Regression model used for movie recommendation does not effectively handle cold-start problems. It relies solely on existing user-item interactions to make recommendations. For new users who haven't rated any movies, it cannot provide personalized recommendations. Similarly, for new movies with no ratings, the model cannot offer meaningful predictions.

Section 3: Evaluation of Recommendation Quality

In this evaluation, we perform the eye-test, manually assessing and comparing the outcomes of each model.

When examining the recommendations generated by different kernel functions for movie similarity, we observe varying degrees of effectiveness in capturing thematic similarities. Linear and Radial Basis Function (RBF) kernels consistently deliver recommendations that align with the genres and tones of the input movies, such as "Memento" and "Guardians of the Galaxy," offering coherent selections within the mystery thriller and action-adventure sci-fi genres, respectively. Conversely, polynomial kernel results show a similar trend but with less diversity in recommended titles, potentially indicating a narrower focus on specific subgenres. Although Naive Bayes still generates relevant recommendations, it lacks the variety seen in other kernels, potentially restricting the exploration of different thematic elements within the recommended genres. Overall, linear and RBF kernels excel in capturing genre similarities and providing diverse yet relevant recommendations, offering users a broader range of options for cinematic exploration within their preferred genres.

Across the four recommendation models, the suggestions for User 1 vary in terms of genre diversity, time periods, and the popularity of movie titles.

- **Single Value Decomposition (SVD)**: Presents a broad array of recommendations covering action, adventure, comedy, drama, and sci-fi genres across different decades, indicating a diverse taste in movies.
- **Random Forest (RF)**: focuses predominantly on mid-1990s movies across various genres like adventure, comedy, action, and drama, reflecting a preference for both classic and popular films from that era.

- **Logistic Regression (LR)**: offers top recommendations primarily from the mid-1990s, with high `liked_probability` scores indicating a strong likelihood of user enjoyment, covering adventure, comedy, and romance genres.

Overall, while all models take into account User 1's varied movie preferences, SVD offers the most diverse and eclectic mix, spanning different genres and time periods, followed by RF and LR, which predominantly concentrate on mid-1990s movies across various genres. However, all models present recommendations tailored to User 1's tastes, providing a selection of entertaining options across different genres and themes.

Conclusion

Among collaborative filtering methods for movie recommendation, Random Forest (RF) proved efficient with the shortest processing time. However, RF, like other methods, struggled with the cold-start problem, limiting recommendations for new users. Despite this, RF offered a diverse mix of movie suggestions, showcasing its potential. Its quick processing and diverse recommendations make RF a promising choice for movie recommendation systems, warranting further investigation into addressing the cold-start challenge while leveraging its strengths.

In the domain of content-based filtering, Naive Bayes (NB) demonstrated the shortest processing time and exhibited consistent performance across evaluations. While NB, akin to other models, only partially addressed the cold-start issue, it presented a limited variety of movie recommendations. Nonetheless, due to its minimal processing time and consistent performance, NB emerged as the preferred choice among the considered models. Its efficiency and reliability underscore its suitability for content-based movie recommendation systems, warranting further investigation into enhancing its coverage and diversity of recommendations.

References

[1] `https://github.com/Bansal0527/Movie-Recomendation-System`.

Contributors

1. **Parth Darshan (B22CS040)**: Implementing K-Nearest Neighbour, K-Means, Naive Bayes Model and Analysis
2. **Mayank Bansal (B22CS070)**: Implementing Logistic Regression and Web Development
3. **Palash Khatod (B22EE095)**: Implementing Linear kernel, Video presentation and Web Development.
4. **Akarsh Katiyar (B22CS006)**: Implementing Singular Value Decomposition, Web Development and Analysis.
5. **Ram Prasad (B22EE054)**: Implementing Polynomial kernel model and Presentation
6. **Rakshit Singhal (B22EE064)**: Implementing Radial Basis Function kernel, Random Forest, Analysis and Documentation