Movie Recommendation System

Hessa Alshaya
College of computing and information
King Saud university
443200834@student.ksu.edu.sa

Rimas Albahli
College of computing and information
King Saud university
443200631@student.ksu.edu.sa

Lama Alotibie
College of computing and information
King Saud university
443201044@student.ksu.edu.sa

Atheer alasiri
College of computing and information
King Saud university
443200626@student.ksu.edu.sa

Maymona Alotaibi
College of computing and information
King Saud university
443200582@student.ksu.edu.sa

***Abstract*— Recommender systems, widely used in e-commerce and streaming platforms, face challenges like data sparsity and cold-start problems. Decision tree algorithms, such as Gain Ratio, Gini Index, and Entropy, offer efficient solutions. This paper evaluates these decision tree methods using the MovieLens 1M dataset.**

Results show that the Gini Index provides the best balance between accuracy and processing time. while it still result in relatively low accuracy due to challenges like class imbalance and limited feature even after experimenting with different algorithmic approaches and hyperparameter optimization.

Agreeing with the insight from Literature Review and to address the challenges future work could explore Hybrid models with the concept of "Movie Swarm" a dynamic approach that leverages group-based user preferences [6] . Or traditional "collaborative filtering" , which relies on static user-item relationships [1].

***Keywords*— Recommender systems, Decision trees, Gini Index, MovieLens 1M, Accuracy, Hybrid models.**

I. Introduction

Recommender systems have become essential tools in various industries, These systems look at large amounts of data to help users navigate through information landscapes by offering personalized suggestions tailored to their preferences. As online platforms like streaming services, e-commerce websites, and social networks continue to grow, the demand for efficient recommendation algorithms has also increased.

These systems not only improve user experience but also increase user engagement and drive sales [1] . However, building effective recommendation models can be challenging due to factors like limited data availability, difficulties in making recommendations for new users with little to no history , and constantly changing user preferences [2] . Addressing these issues is crucial to creating systems that consistently deliver accurate and relevant suggestions.

Among the many algorithms explored for addressing these challenges, decision trees have gained attention due to their interpretability and efficiency [3] [4] . The following literature review examines four key studies that explore the effectiveness of decision tree-based models and related methodologies in building recommendation systems.

II. Background

The development of recommendation systems has significantly transformed user experiences across platforms like streaming services, e-commerce, and social networks. While many traditional systems rely on collaborative filtering or content-based techniques, recent research highlights the potential of decision tree algorithms to enhance the efficiency of recommendation processes. Decision trees are particularly advantageous due to their interpretability and straightforward classification capabilities [1] [2] . They have been successfully applied to personalized content recommendations, allowing systems to adapt dynamically to user preferences by identifying patterns within contextual and demographic data [2] [4] .

Moreover, decision tree models outperform in conversational recommendation scenarios where user preferences are elicited in real-time through interactions. This adaptability allows the system to quickly adjust recommendations based on user feedback, making it ideal for environments with quickly changing user behaviors [4] . By leveraging these advantages, integrating decision trees into movie recommendation systems offers a promising approach to overcoming challenges like cold-start issues and data sparsity, which have traditionally impeded the accuracy of recommendations [1] [2] .

III. Literature Review

Recommender systems have become vital in enhancing user experiences across various platforms, ranging from streaming services like Netflix to e-commerce giants like Amazon. These systems utilize user data to provide personalized suggestions, significantly increasing engagement and driving sales. However, challenges such as data sparsity, cold-start issues, and the need for scalable solutions continue to drive research into more efficient and effective algorithms. Various approaches, including collaborative filtering, content-based filtering, and hybrid models, have been explored to address these challenges.

Dutta et al. (2019) [1] , provide a comprehensive examination of **traditional recommendation techniques**, emphasizing their strengths and weaknesses. Collaborative filtering, while widely used, often faces limitations such as cold-start problems, where new users or items have insufficient data for accurate recommendations. The authors argue that integrating collaborative approaches with other models, such as decision trees, can help overcome these challenges. The study highlights the need for hybrid models that leverage multiple algorithms to optimize accuracy and scalability, especially in domains where user preferences evolve rapidly.

Building on the idea of hybrid models, Guabassi et al. (2020) [2] , explore the application of decision trees in a **ubiquitous learning environment**. Their work focuses on utilizing decision trees to personalize content recommendations based on user profiles and contextual data. Though primarily applied in the educational domain, their methodology offers insights that can be adapted to other areas, such as movie recommendations. Decision trees, known for their interpretability and computational efficiency, were shown to effectively classify users and tailor content to their needs. However, the authors mentioned that decision trees may require frequent updates to handle dynamic user data, which can be computationally intensive in large-scale applications.

Wolf et al. (2022) [3] , take the concept of decision trees further by introducing **meta-decision trees** aimed at enhancing explainability in recommendation systems. Traditional recommendation algorithms often function as "black boxes", leaving users uncertain about why certain items are suggested. By integrating explainability into the recommendation process, meta-decision trees provide transparent decision-making pathways, allowing users to understand the basis of the recommendations they receive. While this approach can boost user trust and satisfaction, the study acknowledges that adding layers of explainability increases computational complexity, potentially limiting the scalability of such systems for large datasets.

In exploring the adaptability of decision tree models, Wang et al. (2021) [4] , investigate their use in **conversational recommendation systems**. These systems dynamically adjust to user inputs through multi-turn interactions, making them ideal for real-time applications where user preferences can shift during a single session. The authors challenge the prevailing assumption that deep learning models are necessary for conversational recommendations, demonstrating that decision trees can achieve similar performance with significantly lower computational costs. By strategically utilizing decision trees to guide conversations and recommendations, the system can adapt quickly to user feedback. However, The Author point out that decision trees may struggle with highly nuanced interactions, suggesting that a hybrid approach could better handle complex conversational scenarios.

Halder et al. (2012) [5] , introduce a novel movie recommendation system designed to address challenges in both new user and item recommendations. Their approach leverages "movie swarms," a concept that clusters movies based on genres and user engagement patterns. This method enhances traditional collaborative and content-based filtering by reducing issues such as sparsity and cold-start problems. The authors demonstrate that movie swarms not only assist producers in identifying popular trends but also enable efficient recommendations through mining "interesting" and "popular" movies. Experimental validation using the MovieLens dataset confirms the system's effectiveness, although the authors note scalability concerns when handling highly diverse user preferences.

The studies reviewed show how decision tree models can be effective in improving recommendation systems. These models help solve issues like limited data, cold-start problems, and the need for clear explanations of recommendations. They also work well for real-time interactions by quickly adjusting to user feedback. However, while decision trees are efficient and easy to understand, they can face difficulties when working with very large datasets or complex user behaviors.

IV. Dataset And Attributes

This project utilizes the **MovieLens 1M Dataset** [6] , which includes over 1 million user ratings on approximately 3,900 movies, along with user demographics and movie metadata, developed by GroupLens Research Lab, is a popular dataset frequently used for building and evaluating movie recommendation systems. It consists of a substantial collection of user ratings, making it a good resource for testing the performance of various recommendation algorithms aimed at improving personalized movie suggestions.

The dataset consists of three files: **ratings.dat, users.dat, and movies.dat** .

*IV. **ratings.dat file***

The file contains user ratings.

formatted as :

UserID::MovieID::Rating::Timestamp

- UserIDs in range [1, 6040]
- MovieIDs in range [1, 3952]
- Rating are made on a 5-star cale
- Timestamp indicate when the ratings were made
- each user has rated at least 20 movies .

*IV. **users.dat file***

The file includes demographic details, formatted as :

UserID::Gender::Age::Occupation::Zip-code

With voluntarily provided information like :

- gender (M , F)
- age (in specified ranges)
- occupation (20 categories)
- zip code.

*IV. **movies.dat file***

The file contains movie metadata in the format :

MovieID::Title::Genres

Where:

- Movie titles align with IMDB entries, (including year of release).
- genres are selected from 18 categories.

Together, these files provide a comprehensive dataset for analyzing user preferences and movie recommendations.

V. DATA MINING TECHNEQUE

A **decision tree** is a data mining technique that uses a tree-like model of decisions and their possible consequences. It involves splitting data into subsets based on feature values, where each internal node represents a decision based on an attribute, and each leaf node represents an outcome or prediction. This technique is said to be well-suited for **recommendation systems** because it is **transparent, interpretable, and efficient**. The tree structure provides clear, understandable rules for recommendations, which helps users trust the system by explaining why certain suggestions are made [3] [4].

Additionally, decision trees handle both **categorical and numerical data** effectively, making them versatile for processing different types of information such as movie genres, user ratings. They are also computationally efficient, enabling real-time recommendations without requiring extensive resources. Unlike deep learning models that need large datasets, decision trees perform well with smaller datasets and can be easily updated to reflect changing user preferences. This adaptability, combined with their simplicity, makes decision trees a practical choice for building robust and scalable recommendation systems [1] [2] .

VI. ALGORITHMS

VI. Gain Ratio

ALGORITHM1: ***Gain Ratio***

```
class GainRatioDecisionTree:
def __init__(self, max_depth=None, 
random_state=None):

  self.max_depth = max_depth
 self.random_state = random_state

 def _split_gain_ratio(self, feature, target):

 total_entropy = calculate_entropy(target)
 weighted_entropy = 0
 intrinsic_value = 0

 for value in unique_values(feature):
 subset = target[feature == value]
 subset_entropy = calculate_entropy(subset)
 weighted_entropy += (len(subset) / len(target)) * 
subset_entropy

  intrinsic_value -= (len(subset) / len(target)) * 
log2(len(subset) / len(target))

 information_gain = total_entropy - 
weighted_entropy

 if intrinsic_value == 0:
 return 0

  gain_ratio = information_gain / intrinsic_value
  return gain_ratio
```

The **Gain Ratio algorithm** improves on Information Gain by addressing its bias toward attributes with many unique values. It normalizes Information Gain by dividing it by the **Intrinsic Value**, which measures the diversity of the split.

This prevents overfitting by penalizing splits that produce many small subsets. Gain Ratio ensures the selection of splits that are both informative and balanced.

VI. Gini Index

ALGORITHM1: Gini Index

```
train_and_visualize_tree('gini', X_train, y_train,
X_test, y_test, max_depth=10,
algorithm_name="Gini Decision Tree")
```

Cini Index is implemented internally within machine learning libraries like **scikit-learn,** the library automatically computes the Gini Index for evaluating splits during the tree-building process.
it is a measure of impurity or diversity used in decision trees to evaluate potential splits. It calculates the likelihood of incorrectly classifying a randomly chosen element if it were labeled according to the distribution of classes in a dataset. The formula for Gini Index is :

$$Gini = 1 - \Sigma pi2$$

where p is the proportion of instances in class iii. A value of 0 indicates pure nodes (all instances belong to one class), while higher values indicate greater impurity. The decision tree aims to minimize the Gini Index to improve classification accuracy at each split.

VI. Entropy

ALGORITHM1: entropy

```
train_and_visualize_tree('entropy', X_train, y_train,
X_test, y_test, max_depth=10,
algorithm_name="Entropy Decision Tree")
```

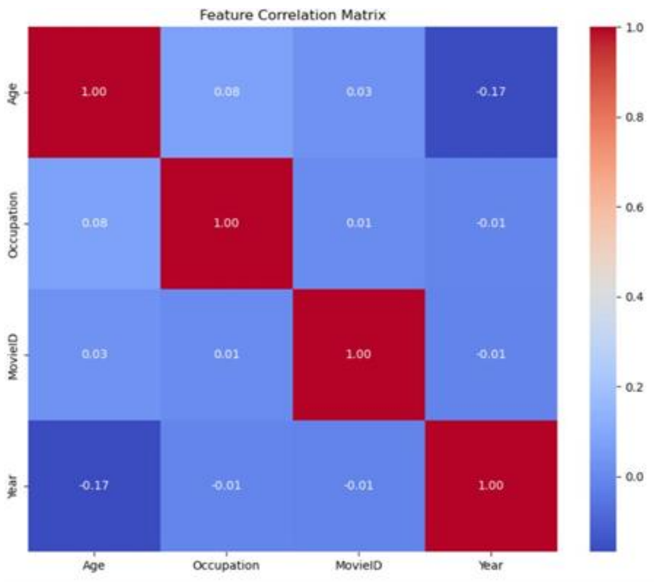
It is implemented in libraries like scikit-learn , The entropy algorithm is a measure of uncertainty or impurity used in decision tree algorithms like ID3 to evaluate the quality of splits. It calculates the level of disorder in a dataset, where lower entropy indicates more homogeneity (pure nodes) and higher entropy represents greater diversity.
The formula is :

$$H(X) = -\Sigma pi \log(pi)$$

where pi is the proportion of instances in class iii. Decision trees aim to minimize entropy by selecting features that provide the greatest **Information Gain**, which is the reduction in entropy after a split. This helps in creating more efficient and accurate trees.

Gini Index, **Entropy**, and **Gain Ratio** are key algorithms used in decision trees to evaluate the quality of splits. Each has its unique approach, **Gini Index** is faster and simpler, **Entropy** is more informative, and **Gain Ratio** ensures balanced and unbiased splits, making it suitable for more complex datasets. The choice depends on the nature of the dataset and the computational requirements.

VII. Correlation Between Attributes



Using the correlation matrix above it provides insights into the linear relationships between the different attribute in the dataset. The correlation coefficients range from -1 to 1, where a value of 1 indicates a perfect positive correlation, 0 indicates no correlation, and -1 indicates a perfect negative correlation.

VII. Highest Positive Correlation

The highest positive correlation is between **Age** and **Occupation** with a coefficient of **0.08** , which is a very weak positive relationship.

VII. Highest Negative Correlation

The highest negative correlation is between **Age** and **Year** with a coefficient of **-0.17** , which suggests slight inverse relationship.

However, all correlations are relatively close to zero, this means that the attribute are largely independent of one another. This lack of strong correlations implies that each feature provides unique information.

VIII. PREPROCESSING

Cleaning: This step usually cleans the raw data into a workable format for a classification task. This code loads three datasets, movies.dat, ratings.dat, and users.dat, each with meaningful column names. It then extracts from the Title column of movies.dat the release year using a regular expression and stores this in a new column called Year. It then cleans the Title by removing the year and trimming any excess spaces. Also, the genre column is split into lists of genres for easier manipulation and analysis.

These datasets are merged into a complete dataset based on the common columns that they share. Specifically, ratings_df is first joined to users_df on the column UserID such that user information is matched against their ratings. This resultant DataFrame is then merged with movies_df on the MovieID column, so that the movie information is joined with the ratings and user information. In this way, all data from the three datasets converge into one single DataFrame, merged_df.

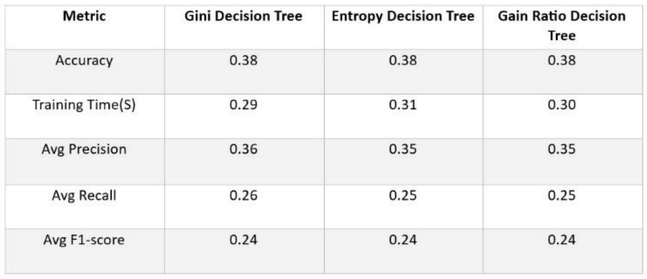
Continuing with simplifying the dataset, the columns are dropped that are not required in this analysis-namely, Timestamp, Zip-code, Title, Genres, and Gender. To reduce the size of the dataset, the program samples 40% of the data randomly and retains the same random seed for consistency. Prepare the data for classification: X features consist of Age, Occupation, MovieID, and Year, while the target variable y is the Rating of the movie. The feature values are filled with zeros so that there won't be any problem while training the model. The final step is to split the dataset into training and testing sets using an 80-20 ratio. This structured pre-processing ensures the data is clean, organized, and ready to build and evaluate a classification model.

IX. EXPERIMENTS

In our experiments, we evaluated the performance of two models using the three algorithms on a 40% sample of the MovieLens dataset. The initial model achieved a **37%** accuracy. **Parameter tuning**, such as adjusting tree depth and switching criteria, led to a slight improvement, raising accuracy to **38%**.

Class imbalance where ratings are unevenly distributed among the different classes. posed a major challenge, as less frequent classes were poorly predicted. Additionally, the **limited feature** set restricted the models' ability to capture complex relationships between users and their preferences. Resulting in lower **precision** and **recall** scores.
Despite these challenges, we explored various strategies to enhance the accuracy, such as experimenting with **different algorithmic approaches** and **hyperparameter optimization**, performance gains were minimal. These results emphasize the importance of addressing class imbalance and incorporating richer features for improved accuracy.

X. COMPARISON AND DISCUSSION



Metric	Gini Decision Tree	Entropy Decision Tree	Gain Ratio Decision Tree
Accuracy	0.38	0.38	0.38
Training Time(S)	0.29	0.31	0.30
Avg Precision	0.36	0.35	0.35
Avg Recall	0.26	0.25	0.25
Avg F1-score	0.24	0.24	0.24

With the above table that Compare the three decision tree algorithms, Using the following performance metric :

- **Accuracy**: The ratio of correct predictions (both true positives and true negatives) to the total predictions made. Higher accuracy means the model is often correct.
- **Training Time(s)**: The time that each algorithm takes it.
- **Precision**: The ratio of true positive predictions to the total predicted positives for a specific class. Higher precision means fewer false positives.
- **Recall**: The ratio of true positive predictions to the total actual positives for a specific class. Higher recall indicates fewer false negatives.
- **F1-score**: The harmonic mean of precision and recall, providing a balance between the two.

Based on the analysis of performance metrics, the three algorithms achieved the same **Accuracy** of **38%** , They can be ranked in order of effectiveness :

The **Gini Decision Tree** emerged as the most balanced approach overall, providing slightly better precision and recall rates compared to the others. The **Gain Ratio Decision Tree** and the **Entropy Decision Tree** are very similar to other. Although all three algorithms displayed comparable accuracy, their nuanced differences in precision, recall, and F1-scores influenced the final ranking.

XI. Conclusion

While the **Gini Decision Tree** Algorithm was the best of the three algorithms, The low accuracy and precision of the current movie recommendation system, with values around 40%, can be attributed to several issues. One significant factor is the high **class imbalance** present in the dataset, Also using only features like Age, Occupation, MovieID, and Year may not sufficiently **capture** the complex relationships between users and their movie preferences.

To address these limitations, future work could explore the concept of "**Movie Swarm**" a dynamic approach that leverages group-based user preferences [5] . Or traditional “**collaborative filtering”** , which relies on static user-item relationships [1] , Movie Swarm identifies and analyzes the collective behavior of groups of users, enhancing the system's ability to capture complex, evolving preferences [5] . Integrating Movie Swarm with existing techniques, such as decision trees and ensemble methods, could significantly improve prediction accuracy and model generalization, offering a promising avenue for future development in personalized movie recommendations.

We **agree** with the insights from previous work discussed in the **literature review**, especially regarding the effectiveness of hybrid models in overcoming challenges like data sparsity and cold-start issues. The integration of decision trees, as emphasized by Dutta et al. (2019) [1] , Guabassi et al. (2020) [2] , and others [3] [4] , is crucial for improving both accuracy and explainability in recommendation systems. By using a **hybrid model** that combines the strengths of different techniques, it may be possible to achieve higher accuracy and better overall performance in predicting user ratings.

References

[1] N. Sharma and M. Dutta, “Movie recommendation systems: A brief overview,” in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Jul. 2020, pp. 59–62. doi: 10.1145/3411174.3411194.

[2] I. El Guabassi, M. Al Achhab, I. JELLOULI, and B. E. EL Mohajir, “Recommender system for ubiquitous learning based on decision tree,” 2016.

[3] E. Shulman and L. Wolf, “Meta decision trees for explainable recommendation systems,” in *AIES 2020 - Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, Association for Computing Machinery, Inc, Feb. 2020, pp. 365–371. doi: 10.1145/3375627.3375876.

[4] A. S. M. A. U. Haque and H. Wang, “Rethinking Conversational Recommendations: Is Decision Tree All You Need?,” in *International Conference on Information and Knowledge Management, Proceedings*, Association for Computing Machinery, Oct. 2022, pp. 686–695. doi: 10.1145/3511808.3557433.

[5] S. Halder, A. M. J. Sarkar, and Y. K. Lee, “Movie recommendation system based on movie swarm,” in *Proceedings - 2nd International Conference on Cloud and Green Computing and 2nd International Conference on Social Computing and Its Applications, CGC/SCA 2012*, 2012, pp. 804–809. doi: 10.1109/CGC.2012.121.

[6] F. M. Harper and J. A. Konstan, “The MovieLens 1M Datasets,” *ACM Trans Interact Intell Syst*, vol. 5, no. 4, pp. 1–19, Jan. 2016, doi: 10.1145/2827872.