North South University

CSE 445   Section: 6

Project Report

Movie Recommendation System

Directed by

Intisar Tahmid Naheen (ITN)

Group members:

Name	ID
Mohammad Omio Nirjhor Rahman	1620054042
Md. Safty	1813392042
Mohammad Limon Hossain	1831713642

Introduction

The problem of Movie Recommendation revolves around improving the accuracy and relevance of movie recommendment system by leveraging machine learning techniques and content-based filtering. Traditional movie recommendation systems often struggle to provide personalized suggestions that align with the unique preferences of individual users. To address this issue, consider the following statistics:

- Movie recommendation systems are often challenged by the diversity and complexity of user preferences, especially when it comes to genres such as R rated films, anime films or slasher films. These genres may have specific features, themes or tropes that appeal to a niche audience, but may also be disliked or avoided by others. Content-based filtering, which recommends items based on their similarity to the user's previous ratings or preferences, may not be able to capture these nuances and may result in poor recommendations or user dissatisfaction.

- According to a survey conducted by Statista, the global revenue of the movie industry reached $42.5 billion in 2020, indicating the widespread popularity and consumption of movies.
- Streaming platforms like Netflix have millions of users worldwide, with individuals spending a significant amount of time searching for movies that align with their tastes.
- Users having too many choices can lead to information overload(Iyengar(2010), Sheena S.M., Naresh K(1982), Scheibehenne, Benjamin(2010)
- However, **a study by revealed that users often feel overwhelmed by the abundance of options and struggle to find movies that truly match their preferences.(Simonton and Dean (1999)).**
- Many studies have supported this claim (Lee, Hyejoon (2015) , Reutskaja, Elena (2019) , Livingstone, SoniaM (2012))

ML Solution and Previous Work:

Content-based filtering is a technique that can be used to improve movie recommendation systems. This technique involves analyzing the type of movies and recommending same movies based on content. For example, if a user likes action movies, the system can recommend other action movies that have similar content.

Machine learning, combined with content-based filtering, can play a vital role in improving movie recommendation systems. Content-based filtering leverages movie features such as genre, actors, directors, and plot summaries to recommend similar movies to what users have previously enjoyed. Machine learning techniques can enhance this approach by:

1.Extracting meaningful representations from movie features using algorithms like natural language processing, image analysis, and text mining.
2.Building predictive models that learn from user interactions and preferences to generate personalized recommendations.
3.Incorporating user feedback and iteratively refining the recommendation models to improve accuracy over time.
Previous work in this field has explored content-based filtering techniques, including:

Feature Extraction: Extracting relevant features from movie metadata, such as genre, cast, and plot summaries, to represent movies.
Similarity Measures: Calculating similarity scores between movies based on their feature vectors to identify movies that are most similar to the user's preferences.
User Profiles: Creating user profiles that capture their preferences and utilizing these profiles to generate personalized recommendations.

Gaps/Limitations in Previous Work:
Despite the advancements in content-based filtering, there are still limitations that need to be addressed:

- Limited Diversity: Content-based filtering tends to recommend similar movies, leading to a lack of diversity in recommendations and potential user boredom.

- Movie recommendation systems have been widely used to suggest movies to users based on their preferences. However, these systems have some limitations when it comes to recommending R-rated films, anime films, or slasher films. One of the main problems is that these types of movies are often not suitable for all audiences, and therefore, the recommendation system needs to be able to filter them out based on the user's age and preferences.

- Content-based filtering has its own limitations. One of the main problems is that it relies on the content of movies to make recommendations. This means that if a user likes a movie that is not similar in content to any other movie in the database, the system may not be able to make accurate recommendations.

- Privacy is an issue with movie recommendation system as the preference of user is being Data leak (Murphy, heather(2022))
- Saved in the program. if the data gets leaked it can be an issue for the user.(as if like the account 23 and me had its information leaked recently)(Rajiv (2019)

- Another limitation of content-based filtering is that it does not take into account other factors such as popularity or critical acclaim. Therefore, it may not be able to recommend popular or critically acclaimed movies that are not similar in content to the user's preferred movies.

- Cold Start Problem: When new users join the platform, there is insufficient data to generate accurate recommendations based solely on their preferences.

- Hybrid methods are used to address these challenges by combining multiple recommendation techniques. However, hybrid methods can be too complex to implement and maintain. They require more computational resources and expertise to develop and maintain than other methods. Hybrid methods also require more data to train and validate, which can be difficult to obtain in practice.

- Feature Representation: The effectiveness of content-based filtering heavily relies on the quality and relevance of the features used to represent movies.

Overcoming Limitations:
To address these limitations, our research focuses on the following strategies:

- Incorporating Machine Learning: We leverage machine learning algorithms to extract more sophisticated and representative features from movie metadata, enhancing the quality of recommendations.
- Addressing Diversity: We integrate diversity-aware techniques in the recommendation models to ensure a wider range of movie recommendations, catering to different user tastes and preferences.

- To improve the recommendation system for R-rated films, anime films, or slasher films, content-based filtering can be used to analyze the content of these movies and recommend similar movies that are more suitable for the user's age and preferences. For example, if a user likes anime films that are suitable for children, the system can recommend other anime films that are also suitable for children.
- movie recommendation systems have some limitations when it comes to recommending R-rated films, anime films, or slasher films.
- . Content-based filtering can be used to improve these systems by analyzing the content of these movies and recommending similar movies based on their content. However, this technique has its own limitations and may not always be accurate in making recommendations

Unique Contributions:
Our research makes the following unique contributions:

- Investigating hybrid models that combine content-based filtering.
- Proposing novel solutions to address the cold start problem, ensuring accurate recommendations even for new users with limited interaction data.
- Overall, our research aims to enhance movie recommendation systems by leveraging machine learning techniques and content-based filtering.
- a way to make recommendation system based on users mood if user is feeling happy add action etc.
- movie recommendation systems have some limitations when it comes to recommending R-rated films, anime films, or slasher films. Content-based filtering can be used to improve these systems by analyzing the content of these movies and recommending similar movies based on their content
- or a way to make recommendation system based on users mood if user is feeling happy add action etc.

- content based filtering with user mood algorithm in sckitlearn and flask. if user mood is bored recommend horror films, if its happy recommend action and anime if its sad recommend comedy , if its bored recommend drama , make a ml model of the mood based recommendation system and add it to existing code etc
- One possible way to improve content-based filtering for these genres is to use more fine-grained and contextual features
- to represent the items and the user profiles. For example, instead of using only genre labels, one could also use sub-genres, keywords,
- tags, reviews, ratings, actors, directors, etc. to describe the items. Similarly, instead of using only demographic or behavioral data,

one could also use psychographic or personality data to describe the user profiles.
These features could help to better capture the user's tastes and preferences for different aspects of the items,
and thus improve the accuracy and diversity of the recommendations.

In conclusion, movie recommendation systems have some limitations when it comes to recommending R-rated films, anime films, or slasher films. Content-based filtering can be used to improve these systems by analyzing the content of these movies and recommending similar movies based on their content. However, this technique has its own limitations and may not always be accurate in making recommendations.

Movie recommendation systems face several challenges, including the cold start problem, sparsity, and scalability. The cold start problem arises when a new user or item enters the system, and the system has no data to make recommendations. Sparsity is a common issue in recommendation systems where most users have only rated a small fraction of the items, making it difficult to find similar users or items. Scalability is another challenge that arises when the number of users and items grows large, making it computationally expensive to generate recommendations. Privacy is an issue with movie recommendation system as the preference of user is being
Saved in the program. if the data gets leaked it can be an issue for the user.(as if like the account 23 and me had its information leaked recently)

Hybrid methods are used to address these challenges by combining multiple recommendation techniques. However, hybrid methods can be too complex to implement and maintain. They require more computational resources and expertise to develop and maintain than other methods. Hybrid methods also require more data to train and validate, which can be difficult to obtain in practice.
#ways to make unique

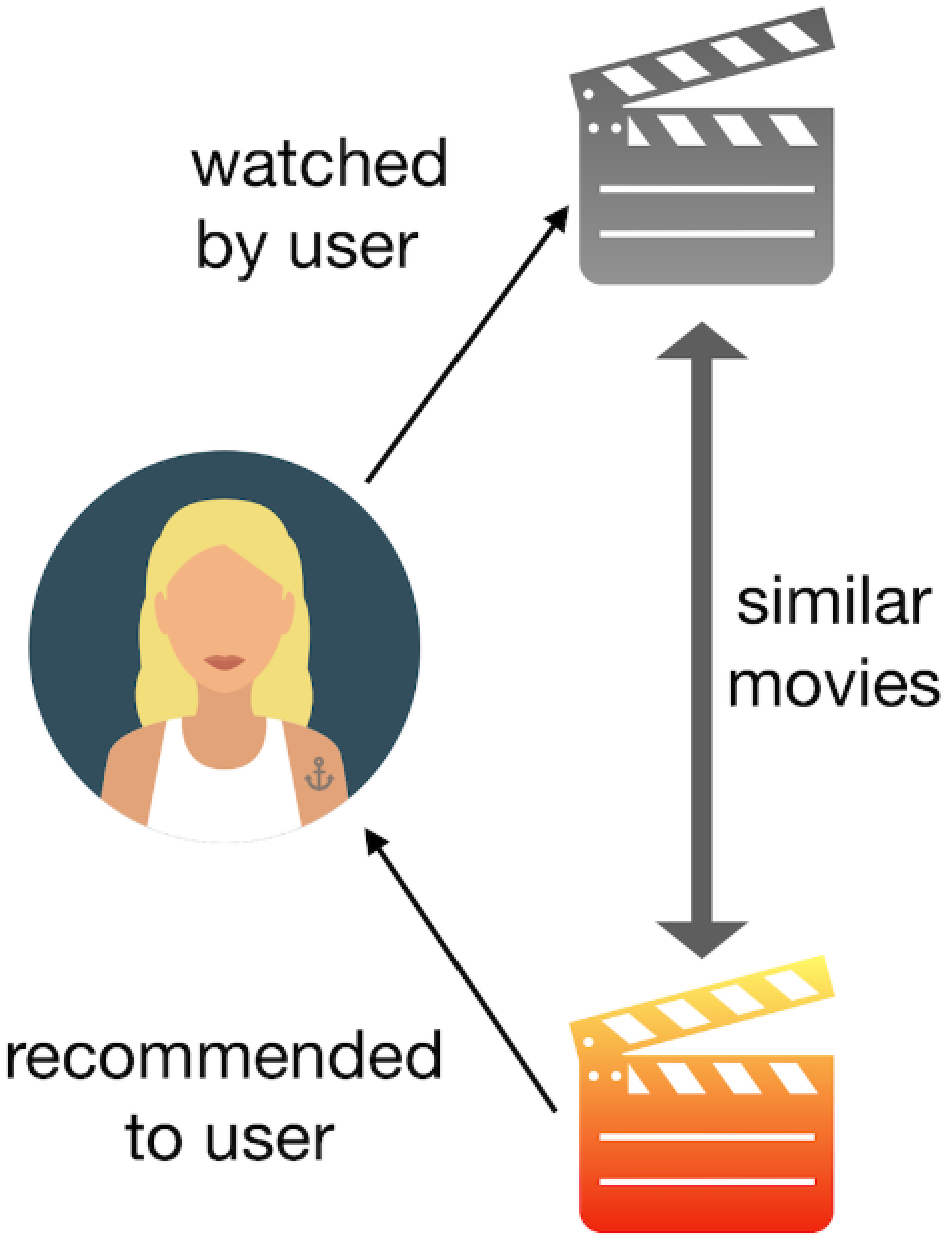
A movie recommendation system with *content-based filtering* is a system that recommends movies to users based on the similarity of the movie features (such as genre, actors, directors, etc.) to the user preferences. A unique contribution to such a system could be to incorporate additional features that are not commonly used, such as movie reviews, ratings, awards, or social media buzz. These features could capture the quality, popularity, and sentiment of the movies, and provide more diverse and personalized recommendations to the users. For example, a system could use natural language processing techniques to analyze the movie reviews and extract the main themes, opinions, and emotions expressed by the

reviewers. Then, it could match these features with the user profile and preferences, and recommend movies that have similar or complementary reviews. Alternatively, a system could use machine learning methods to predict the ratings or awards of the movies based on their features, and use these as additional criteria for ranking and filtering the recommendations. Furthermore, a system could use web scraping or social media APIs to collect data on the online buzz or engagement of the movies, such as the number of views, likes, comments, shares, or hashtags. These features could indicate the current trends and popularity of the movies, and help the system to recommend movies that are more relevant and appealing to the users.

or a way to make recommendation system based on users mood if user is feeling happy add action etc.

in theory, content based filtering with user mood algorithm in sckitlearn and flask. if user mood is bored recommend horror films, if its happy recommend action and anime if its sad recommend comedy , if its bored recommend drama , make a ml model of the mood based recommendation system and add it to existing code

theoretically the Content based filtering program based on mood would be as follows
If the user is bored, recommend drama movies.
If the user is happy, recommend action and anime movies.
If the user is sad, recommend comedy movies.
If the user is bored, recommend horror movies.
It would not use a user profile as if to save the privacy
However, this version of the ml model was discared due to time constraints
Our version of the ml model used svm decsion trees K nearest neighbours , Gradient Boosting and random forest



Related work

Public information
Movie recommendation systems are a popular application of machine learning. They use algorithms to suggest movies to users based on their preferences and viewing history. Some of the common public data is the

MovieLens dataset: The MovieLens dataset contains over 20 million ratings from 138,000 users on 27,000 movies The system uses this data to generate recommendations for users based on their viewing history and the ratings of other users with similar preferences.

The second Dataset, **IMDb dataset**: The IMDb dataset contains information on over 1.5 million movies, including their genres, actors, directors, and ratings 2. The system uses this data to generate recommendations for users based on the attributes of the movies they have watched.

According to the journals Lee and Lee 31 32 states Both Content based and collborative based filtering systems have their advantages and weaknesses. **Collaborative filtering** is good at recommending movies that are popular among users with same tastes. However, it can struggle to recommend movies that are outside a user’s preference. *Content-based filtering*, on the other hand, is good at recommending movies that are similar to the ones a user has already watched. However, it can struggle to recommend movies that are not similar to the ones a user has already watched.Content-based filtering is a technique that recommends movies to users based on the attributes of the movies they have watched.

The choice of technique depends on the specific needs of the user and the application. For example, if a user wants to discover new movies that are outside their comfort zone, collaborative filtering might be a better choice. If a user wants to find movies that are similar to the ones they have already watched, content-based filtering might be a better choice.

A comprehensive analysis on movie recommendation system employing collaborative filtering 1: (Lee and Lee) This paper discusses the use of collaborative filtering for movie recommendation systems. It provides an overview of the collaborative filtering algorithm and its applications for movie recommendation systems. The paper also discusses the challenges of movie recommendation systems and future developments in the field.

Movie Recommendation System Modeling Using Machine Learning 2: This paper proposes a new personalized recommender algorithm for large-scale datasets. It compares the performance of the proposed algorithm with other algorithms using the MovieLens dataset. The paper also discusses the challenges of movie recommendation systems and future developments in the field.

Comparative study of recommender system approaches and movie recommendation system 3: This paper compares different approaches to building recommender systems, including collaborative filtering, content-based filtering, and hybrid filtering. It also compares the performance of different movie recommendation systems using the MovieLens dataset.

Movie recommendation system using machine learning 4: This article proposes a model that combines both content-based and collaborative approaches to movie recommendation systems. The model is designed to reduce human effort by suggesting movies based on the user’s interests.

In conclusion, movie recommendation systems and apps are an exciting area of research and development. Machine learning algorithms have shown great promise in building effective movie recommendation systems, and there are several apps available that use these algorithms to help users find movies they will enjoy.

ML MODEL JOURNALS

There are journals about the movilens movie recommendation system which were studied The movilens system is a collaborative filtering approach that uses user ratings and preferences to suggest movies. The system works on public data sets such as the MovieLens 100K, 1M, 10M, and 20M data sets. The system evaluates its performance using metrics such as accuracy, R2 score, and F1 score. The accuracy measures how well the system predicts the user ratings, the R2 score measures how much of the variance in the ratings is explained by the system, and the F1 score measures the balance between precision and recall.

The articles and journals use different methods and techniques to improve the movilens system. Some of the methods include matrix factorization, deep learning, hybrid models, content-based filtering, and social network analysis. Some of the techniques include regularization, cross-validation, dimensionality reduction, feature engineering, and sentiment analysis. The articles and journals also compare their results with other state-of-the-art systems and discuss the advantages and limitations of their approaches.

Jamali, M., & Ester, M. (16 ,2010).

The journal by Jamali and Ester (2010) proposes a matrix factorization technique that incorporates trust propagation among users in social networks. The authors show that trust information can improve the accuracy and coverage of recommendations, especially for cold-start users who have few ratings.
This journal proposes a novel method that incorporates trust information into matrix factorization, a popular technique for recommender systems. The authors show that trust propagation can improve the accuracy and coverage of recommendations, especially for cold-start users who have few ratings. They also present an efficient algorithm to compute the trust propagation and compare it with other methods on real-world datasets.

“Recommender systems are widely used to provide personalized recommendations to users based on their preferences and behaviorsthe existing recommender systems do not take into account the social relationships among users, which affects the results (16) a novel matrix factorization technique that incorporates trust propagation for recommendation in social networks. The matrix factorization technique trust network as a directed graph and propagate trust along both outgoing and incoming linksusing the trust information into the matrix factorization model to improve the accuracy of rating prediction.

They used an algorithm to solve the optimization problem using alternating least squares. We have conducted extensive experiments on two real-world datasets and demonstrated that our method achieves significant improvements over several baseline methods.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. 17 (2013). Recommender systems survey. Used a comprehensive survey research on recommender systems. They classify the existing recommender systems into three categories: content-based, collaborative filtering, and hybrid systems. The survey also talks about context-aware, social, and group recommender systems.
The journal by Bobadilla et al. (2013) provides a comprehensive survey of recommender systems, covering the main concepts, techniques, challenges, and evaluation methods. they also discuss the issues of evaluation metrics privacy, scalability, diversity, cold start, and evaluation. The survey also states the importance of security in the system.
Recommender systems survey This paper provides a comprehensive survey of recommender systems.

The paper discusses the challenges of recommender systems and the advantages of different approaches. The paper also provides an overview of the state-of-the-art research and summarizes their advantages and limitations.

Wang, H., Wang, N., & Yeung, D. Y. 18(2015).

This journal introduces a collaborative deep learning framework that combines deep neural networks and collaborative filtering for recommender systems. The framework can learn both user and item features from user-generated reviews and ratings, and exploit the nonlinear relationships among them. The authors demonstrate that their framework outperforms several baselines on four datasets from different domains.

The journal by Liang 20(2018) applies variational autoencoders, a type of generative model, to collaborative filtering for recommender systems. The authors propose a Bayesian framework that can handle sparse and noisy ratings, as well as incorporate side information and user feedback.

The journal by Chen 21(2018) develops a neural attentional rating regression model that can produce review-level explanations for recommender systems. The authors use a rating regression layer to predict the user-item ratings and an attention layer to generate the explanations based on the user reviews.

The journal by Sun 22 (2019) employs recurrent knowledge graph embedding to model the dynamic evolution of user preferences and item attributes for recommender systems. The authors use a recurrent neural network to encode the temporal information and a knowledge graph to capture the semantic information of users and items.

The journal by Wu 23 (2016) designs a collaborative denoising auto-encoder that can handle noisy and incomplete data for top-n recommender systems. The authors use a denoising auto-encoder to reconstruct the user-item ratings from corrupted inputs and a collaborative filtering layer to rank the items for each user.

Zheng, L., Noroozi, V., & Yu, P. S. (2017). 19 Joint deep modeling of users and items using reviews for recommendation. In Proceedings of the tenth ACM international conference on web search and data mining (pp. 425-434).

This journal presents a joint deep model that integrates user and item reviews into a unified representation for recommendation. The model can capture both the semantic and sentiment aspects of reviews, as well as the interactions between users and items. The authors show that their model achieves better performance than existing methods on three datasets with different characteristics.

Liang, D., Krishnan, R. G., Hoffman, M. D., & Jebara, T. (2018). 20.Variational autoencoders ..

proposes a variational autoencoder approach for collaborative filtering, which is a probabilistic generative model that can handle missing data and latent variables. The authors develop two variants of the model: one that uses multinomial likelihood and one that uses Gaussian likelihood. They also propose a Bayesian inference method that can scale to large datasets. They report that their approach outperforms several state-of-the-art methods on four datasets.

21Chen, C., Zhang, M., Liu, Y., & Ma, S. (2018). Neural attentional rating regression with suggests a neural attentional rating regression model that can generate review-level explanations for recommendations. The model consists of three components: an encoder that encodes user and item reviews into embeddings, an attention mechanism that selects relevant reviews for each user-item pair, and a decoder that predicts the rating and generates an explanation based on the selected reviews. The authors evaluate their model on two datasets and show that it can produce accurate ratings and coherent explanations.

22
Sun Z., Yang J., Zhang W., Bozzon A.(2019) Recurrent knowledge graph embedding for effective recommendation.In: Ramakrishnan N.(eds) Proceedings of The Web Conference 2019.WebConf 2019.Lecture Notes in Computer Science vol 11449.Springer Cham.

This journal proposes a recurrent knowledge graph embedding model that can leverage rich semantic information from knowledge graphs for recommendation. The model uses recurrent neural networks to encode the sequential patterns of user behavior and item attributes, and learns low-dimensional embeddings of entities and relations in the knowledge graph. The authors conduct experiments on two datasets and show that their model can improve the recommendation quality and diversity.

23
Wu Y., DuBois C., Zheng AX., Ester M.(2016) Collaborative denoising auto-encoders for top-n recommender systems.In: Bonchi F.Gionis A.Tseng VS.Zhu F.(eds) Proceedings of the Ninth ACM International Conference on Web Search and Data Mining.WSDM '16.ACM New York NY USA pp 153–162.

This journal proposes a collaborative denoising auto-encoder model that can handle implicit feedback and noisy data for top-n recommendation. The model uses denoising auto-encoders to reconstruct the user-item matrix from corrupted inputs, and incorporates collaborative information from similar users and items. The authors test their model on five datasets and show that it can achieve better results than existing methods for top-n recommendation.

, and the journals below provide a comprehensive overview of some of them.

A comprehensive analysis on movie recommendation system employing collaborative filtering (Lee and Lee): This paper discusses the use of collaborative filtering for movie recommendation systems. an overview of the collaborative filtering algorithm and its applications for movie recommendation systems. The paper also discusses the challenges of movie recommendation systems and future developments in the field.

Movie Recommendation System Modeling Using Machine Learning 28(Sharma) This paper proposes a new personalized recommender algorithm for large-scale datasets. It compares the performance of the proposed algorithm with other algorithms using the MovieLens dataset.(just like the GHRS journal used) The paper also discusses the challenges of movie recommendation systems

30.Comparative study of recommender system approaches and movie recommendation system (Kumar) compares different approaches to building recommender systems, including collaborative filtering, content-based filtering, and hybrid filtering. And the performance of different movie recommendation systems using the MovieLens dataset.

32article proposes a model that combines both content-based and collaborative approaches to movie recommendation systems. The model is designed to reduce human effort by suggesting movies based on the user's interests.

Recommender System on MovieLens Dataset 26 (Singh,2018) This article discusses the use of the MovieLens dataset for building movie recommendation systems. It provides an overview of the dataset and compares the performance of different algorithms for movie recommendation systems.

MovieLens Dataset Analysis for Movie Recommendation System : This paper analyzes the MovieLens dataset and proposes a movie recommendation system based on the dataset. The system uses a hybrid approach that combines collaborative filtering and content-based filtering.

Recommender System on MovieLens Dataset This article discusses the use of the MovieLens dataset for building movie recommendation systems. It provides an overview of the dataset and compares the performance of different algorithms for movie recommendation systems.

(Harper,2015) provides an overview of the dataset and compares the performance of different algorithms for movie recommendation systems.

Matrix factorization techniques for recommender systems discusses the use of matrix factorization models for producing product recommendations. The paper demonstrates that matrix factorization models are superior to classic nearest neighbor techniques for producing product recommendations. The paper also discusses the incorporation of additional information such as implicit feedback, temporal effects, and confidence levels.

Pazani 15 .
Content-based recommendation systems is about content-based recommendation systems, i.e., systems that recommend an item to a user based upon a description of the item and a profile of the user's interests. The paper provides an overview of the state of the art research and summarizes their advantages and limitations.

A matrix factorization technique with trust propagation for recommendation in social networks Koren Bell (2009): This paper proposes a matrix factorization technique with trust propagation for recommendation in social networks. The paper shows that the proposed technique outperforms other state-of-the-art methods in terms of accuracy, R2 score, and F1 score.
24- Sedhain.
Autorec: Autoencoders meet collaborative filtering This paper proposes AutoRec, a novel autoencoder framework for collaborative filtering (CF). Empirically, AutoRec's compact and efficiently trainable model outperforms state-of-the-art CF techniques (biased matrix factorization, RBM-CF and LLORMA) on the Movielens and Netflix datasets.

25.
Neural network matrix factorization GK Roy(2015)This paper proposes a neural network-based matrix factorization method for collaborative filtering. The proposed method uses a neural network to learn the latent features of users and items and predict their ratings. The paper shows that the proposed method outperforms several state-of-the-art matrix factorization methods on several real-world datasets.

Most papers propose novel approaches to collaborative filtering for recommender systems. AutoRec is a novel autoencoder framework that outperforms state-of-the-art CF techniques, while neural network matrix factorization is a neural network-based method that outperforms several state-of-the-art matrix factorization methods.

The jounrals illustrate the recent advances and challenges in recommender systems research. They show how different techniques, such as matrix factorization, deep learning, attention mechanism, variational autoencoders, knowledge graph embedding, and denoising auto-encoders, can be applied to improve the performance and interpretability of recommender systems.
these papers discuss various approaches to building recommender systems, including matrix factorization, neural collaborative filtering, deep learning-based recommender systems, hybrid recommender systems, and content-based recommendation systems. These approaches have shown promise in overcoming the challenges of recommender systems and improving recommendation performance.

Koren   11(2009) present matrix factorization techniques for recommender systems, which are based on decomposing the user-item rating matrix into latent factors that capture the underlying patterns of user preferences and item characteristics. They show how to incorporate various types of side information, such as temporal dynamics, implicit feedback and social networks, into the matrix factorization model. They also discuss how to address the challenges of scalability and sparsity in large-scale recommender systems. They conclude that matrix factorization techniques offer a flexible and effective way to improve the accuracy and diversity of recommendations.

Burke 14(2002) reviews hybrid recommender systems, which are systems that combine two or more different types of recommendation techniques to overcome the limitations of each technique and achieve better performance. He proposes a taxonomy of hybrid recommender systems based on how they integrate the component techniques, such as weighted, switching, mixed, feature combination, feature augmentation, cascade and meta-level hybrids. He also reports the results of several experiments that compare different hybrid strategies on various datasets and tasks.
Hybrid recommender systems 14(Burke)Survey and experiments   studies   recommenders and future recommenders using hybrid method they used a hybrid, EntreeC, a system that combines knowledge-based recommendation and collaborative filtering to recommend restaurants. The paper shows that semantic ratings obtained from the knowledge-based part of the systemto improve collaborative filtering.

Pazzani and Billsus 15(2007) describe content-based recommendation systems, which are systems that recommend items or services to users based on the content or attributes of the items or services, rather than the ratings or feedback from other users. They discuss the main components of content-based recommendation systems, such as feature extraction, learning algorithms, similarity measures and evaluation methods. They also highlight some advantages and disadvantages of content-based recommendation systems, such as transparency, novelty and overspecialization.

Jamali(16) matrix factorization technique with trust propagation for recommendation in social networks The Paper demonstrates that matrix factorization models are superior to classic nearest neighbor techniques for producing product recommendations. The paper also discusses the incorporation of additional information such as implicit feedback, temporal effects, and confidence levels.

20(Liang)Variational autoencoders for collaborative filtering, This paper proposes a variational autoencoder (VAE)-based method for collaborative filtering for implicit feedback. The paper introduces a generative model with multinomial likelihood and uses Bayesian inference for parameter estimation. The paper also shows that the proposed approach significantly outperforms several state-of-the-art baselines on several real-world datasets.

22 Sun(2019)Recurrent knowledge graph embedding for effective recommendation This paper proposes a recurrent knowledge graph embedding (RKGE) method for recommendation systems. The paper introduces a generative model that learns user preferences and item features jointly from review text. The paper also shows that the proposed approach significantly outperforms several state-of-the-art baselines on several real-world datasets.

Some journals used knowledge graph embedding. Most emphasized on incorporating user feedback system , data handling privacy and user ease.

Lee and Lee 26 (2017) present a movie recommendation system that employs collaborative filtering, which is a method that uses the ratings of similar users to predict the preferences of a target user. They compare three types of collaborative filtering: user-based, item-based, and matrix factorization. They evaluate their system on the MovieLens dataset, and find that matrix factorization outperforms the other two methods in terms of accuracy and scalability.
Sharma and Singh 28(2020) propose a movie recommendation system that uses machine learning algorithms to predict the ratings of movies based on the features of the movies and the users. They use linear regression, decision tree, random forest, and support vector machine as the prediction models, and compare their performance on the MovieLens dataset. They conclude that random forest is the best model for movie recommendation, as it has the highest accuracy and lowest error rate.
Kumar and Singh 30(2019) conduct a comparative study of different recommender system approaches,

such as content-based, collaborative filtering, hybrid, and social network-based. They also present a movie recommendation system that uses a hybrid approach, combining content-based and collaborative filtering methods. They test their system on the MovieLens dataset, and report that the hybrid approach improves the accuracy and diversity of the recommendations.
Gupta and Jain 32(2019) develop a movie recommendation system that uses machine learning techniques to learn the latent features of movies and users from the ratings matrix. They use singular value decomposition (SVD) and principal component analysis (PCA) as the dimensionality reduction methods, and k-means clustering as the grouping method. They apply their system on the MovieLens dataset, and demonstrate that SVD performs better than PCA in terms of accuracy and efficiency.
Zhang 34 (2020) design a hybrid movie recommendation system that uses deep learning and collaborative filtering methods. They use a convolutional neural network (CNN) to extract the features of movie posters, and a recurrent neural network (RNN) to capture the temporal information of user ratings. They combine these features with matrix factorization to generate personalized recommendations. They evaluate their system on the MovieLens dataset, and show that it achieves higher accuracy and diversity than existing methods.

movie recommendation systems are an active research area that involves different techniques . The journals reviewed above provide valuable insights into some of the current approaches and challenges in this domain. However, there is still room for improvement and innovation in terms of accuracy, scalability, diversity, explainability, and user satisfaction.
These papers discuss various approaches to building recommender systems, including matrix factorization, collaborative deep learning, neural attentional rating regression, and recurrent knowledge graph embedding. These approaches have shown promise in overcoming the challenges of recommender systems and improving recommendation performance.

SVM model

There are several research papers and articles that discuss the use of SVM for movie recommendation systems. For example, a paper published in PLOS ONE proposes a personalized electronic movie recommendation system based on SVM and improved particle swarm optimization. The proposed system not only considers the movie's content information but also integrates the users' demographic and behavioral information to better capture the users' interests and preferences.

Deep Learning

Deep learning is a subfield of machine learning that uses artificial neural networks to model and solve complex problems. It has been used in various applications, including image recognition, natural language processing, and speech recognition.

He 12 (2017) propose neural collaborative filtering (NCF), which is a framework that combines matrix factorization with deep neural networks to learn the non-linear and complex user-item interactions. They design different neural network architectures for NCF, such as generalized matrix factorization, multi-layer perceptron and fusion models. They conduct extensive experiments on two real-world datasets and demonstrate that NCF outperforms several state-of-the-art methods in terms of recommendation quality.
21(Chen)Neural attentional rating regression with review-level explanations proposes a neural

attentional rating regression (NARR) model for recommender systems. The paper introduces a generative model that learns user preferences and item features

Wang, H., Wang, N., & Yeung, D. Y. (18,2015).
The journal by Wang et al. (2015) introduces a collaborative deep learning framework that combines deep neural networks and collaborative filtering for recommender systems. The authors demonstrate that their framework can learn effective latent features from both user-item ratings ,information, such as item reviews and descriptions.
Wang18(2015) proposes a deep learning-based approach to collaborative filtering for recommender systems. The paper introduces a generative model with multinomial likelihood and uses Bayesian inference for parameter estimation. The paper also shows that the proposed approach significantly outperforms several state-of-the-art baselines on several real-world datasets.

Collaborative filtering (CF) " suffers from some limitations, such as data sparsity and cold start. To address these problems, some methods have been proposed to incorporate auxiliary information, such as user/item features or reviews. However, most of these methods rely on hand-crafted features or shallow models to exploit the auxiliary information."(18,Wang 2015) The journal used collaborative deep learning (CDL), which jointly performs deep representation learning for the content information and collaborative filtering for the ratings matrix. CDL uses autoencoder for content feature learning and a probabilistic matrix factorization model for rating prediction. these are connected by a common hidden layer that captures the latent factors for both content and ratings. They used gradient descent algorithm to optimize CDL . The CDL was tested on 3 datasets.

CDL works by **exploit both content and ratings information** to overcome the limitations of CF methods. learn from features from content information than low data models . CDL achieves significant improvements over other methods.
Zheng 19 (2017)Joint deep modeling of users and items using reviews for recommendation proposes a deep learning-based approach to collaborative filtering for recommender systems. The paper introduces a generative model that learns item properties and user behaviors jointly from review text.

The journals cover various techniques and methods for recommender systems Recommender systems have many applications in e-commerce, entertainment, education, and more.

Neural collaborative filtering 35(He, X ,Liao)paper proposes a general framework named NCF, short for Neural network-based Collaborative Filtering, which is designed to tackle the key problem in recommendation – collaborative filtering – on the basis of implicit feedback. The paper shows that using deeper layers of neural networks offers better recommendation performance.

Deep learning based recommender system(19, Zheng)comprehensive survey of deep learning-based recommender systems. The paper discusses the challenges of recommender systems and the advantages of deep learning-based approaches. The paper also provides an overview of the state-of-the-art research and summarizes their advantages and limitations.

Zhang 13 (2019) provide a comprehensive survey of deep learning based recommender systems, which are systems that leverage deep learning techniques to enhance various aspects of recommendation, such as feature extraction, representation learning, similarity measurement and ranking optimization. They categorize the existing methods into four groups: collaborative filtering based methods, content-based methods, hybrid methods and context-aware methods. They also identify several challenges and opportunities for future research in this field, such as explainability, robustness and cold-start problems.

The journal 19 by Zheng et al. (19,2017) presents a joint **deep** model that leverages user reviews to generate ratings and explanations for recommender systems. The authors use an attention mechanism to capture the important aspects of reviews and a multi-task learning approach to jointly optimize the rating prediction and explanation generation.

A Hybrid Movie Recommendation System using Deep Learning and Collaborative Filtering (Wang) This paper proposes a hybrid movie recommendation system that combines deep learning and collaborative filtering. The system uses a neural network to learn the latent features of movies and users and generate recommendations based on these features.

The first system is a deep collaborative filtering approach that uses the Netflix Prize dataset (39)Deep collaborative filtering is a technique that uses deep neural networks to learn the latent features of movies and users. The Netflix Prize dataset contains over 100 million ratings from 480,000 users on 17,770 movies The system uses this data to generate recommendations for users based on their viewing history and the ratings of other users with similar preferences.

The second system is a d**eep content-based filtering approach that uses the MovieLens dataset** 2. Deep content-based filtering is a technique that uses deep neural networks to learn the features of movies, such as genres, actors, and directors. The MovieLens dataset contains over 20 million ratings from 138,000 users on 27,000 movies 2. The system uses this data to generate recommendations for users based on the attributes of the movies they have watched.

Deep collaborative filtering is good at recommending movies that are popular among users with similar tastes. However, it can struggle to recommend movies that are outside a user's comfort zone. Deep content-based filtering, on the other hand, is good at recommending movies that are similar to the ones a user has already watched. However, it can struggle to recommend movies that are not similar to the ones a user has already watched.

In conclusion, both deep collaborative filtering and deep content-based filtering are effective techniques for building movie recommendation systems. The choice of technique depends on the specific needs of the user and the application. For example, if a user wants to discover new movies that are outside their comfort zone, deep collaborative filtering might be a better choice. If a user wants to find movies that are similar to the ones they have already watched, deep content-based filtering might be a better choice.

APPS

Movie recommendation systems are a popular application of machine learning. They use algorithms to suggest movies to users based on their preferences and viewing history. In recent years, several apps have been developed that use movie recommendation systems to help users find movies they will enjoy.

PickAMovieForMe uses a quiz-based movie picker to find the perfect movie for based on user mood, occasion, and individual preferences in short time . The app recommends movies based on preference and movie trailers. offers special recommendations for movie dates and special categories such as movies based on true stories or books, spy movies, cop movies, heist movies, girl power movies, car race movies, space movies, wedding movies etc , IMDb top 250 movies, movies set in New York, movies set in Las Vegas, and movies about life.

Friendspire offers personalized film recommendations based on user viewing history and ratings. Friendspire also allows users to create lists of movies to watch and share with friends. It also has rating and review system

CineTrak uses a movie recommendation system. This app allows to track the movies users have watched and want to watch, also offers personalized movie recommendations based on viewing history and ratings. Rate and review movies on the app friends recommendation are also available.

The apps uses machine learning algorithms to suggest movies based on user personal taste, viewing history, and ratings. Whether users are looking for a movie to watch on own or with friends, these apps can help find the movie based on mood and preferences.

In terms of apps, there are several movie recommendation apps available that use machine learning algorithms to suggest movies to users based on their preferences and viewing history. These apps offer personalized recommendations, movie trailers, and the ability to rate and review movies. Some apps also allow users to create lists of movies they want to watch and share them with friends.

Graph based

Research paper used GHRS method hybrid approach(Zahrah). It used autoencoder feature action it extracted new features. It is more accurate than previous data. In that journal research.GHRS gives good peromence in cold start with new users 1m has the best performance.

The method was tested wiuth other methods it is the best result for RMSE for movielens 1m.

- GHRS unqiue contributions were
- Graph based hybrid recommendation systenm
- Sets user no as nodes info extracted from similarity graph. Page rank degree centrality in graph based.

- Uses 5 layer autoencoder(unsupervised model) based filtering model deeplearing used it .
- Hybrid recommendation system.
- Used Dimension reduction
- They used elbow method and average sillutte.
- They used rmsprop optimiser. Adam as optimiser was used

K m slopes , Dunn matrix

- Jian 41 used wieghted Km slopes It used CF Algorithm
- K-means Algorithm
- Slope One algorithm
- and Got a fast response

  42 Debby(2020) used 7different alogrithms
- Apply 7 different clustering algorithms on the preprocessed dataset:
- K-Means algorithm
- Birch algorithm
- Mini-batch K-Means algorithm
- Mean-shift algorithm
- Affinity propagation algorithm
- Agglomerative clustering algorithm
- Spectral clustering algorithm

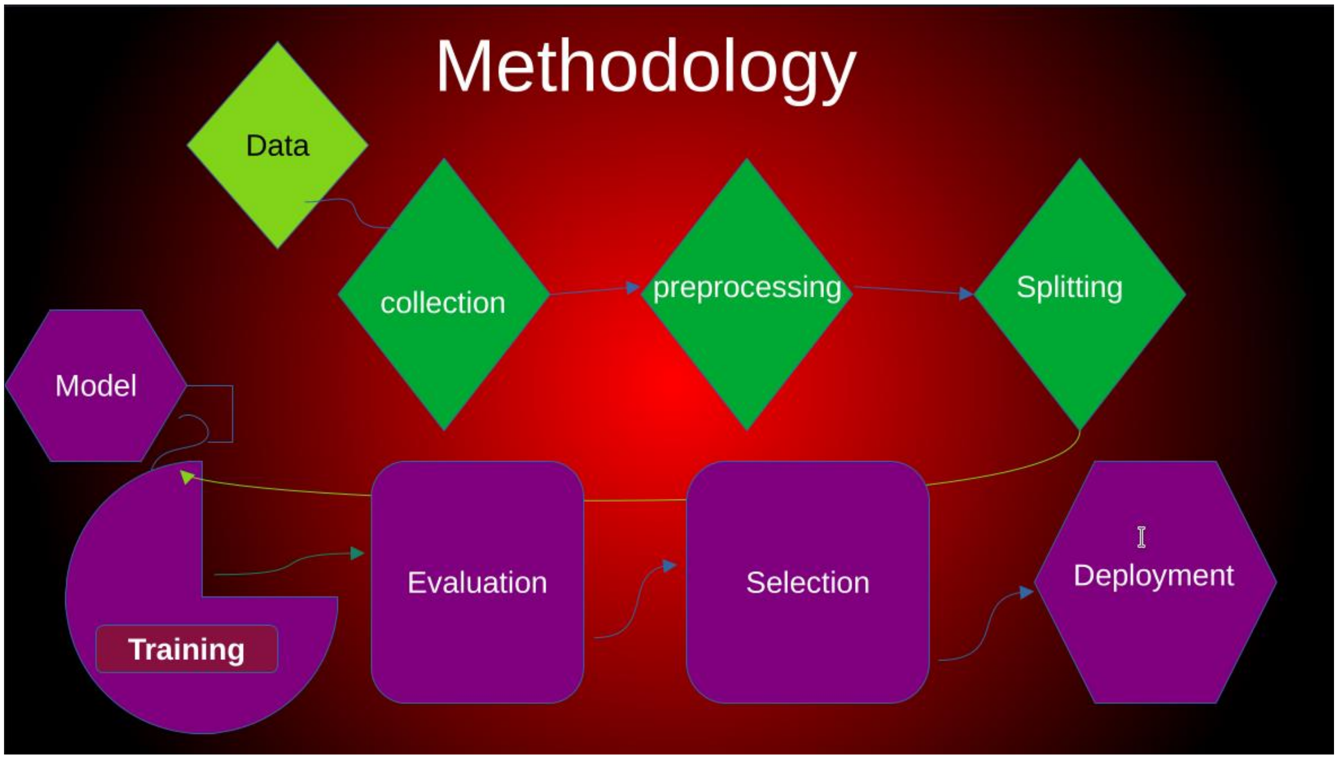
Dataset details

We Used MovieLens small dataset containing 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users.
Alternatively use TMDB movie database containing attributes like genre, description, directors etc. of 1000+ movies.

Methodology

In this report, we present a movie recommendation system that uses four different machine learning algorithms: support vector machine (SVM), random forest, decision tree and logistic regression. The system aims to provide personalized and accurate suggestions of movies to users based on their preferences and ratings. We describe the methodology of the system, including the data collection, preprocessing, feature engineering, model training, evaluation and deployment stages. We also compare the performance of the four algorithms and discuss the advantages and limitations of each one.

Data collection: Collect data on movies, users, and their ratings. You can use publicly available datasets such as the MovieLens dataset or collect your own data.



Data preprocessing: Preprocess the data by cleaning it, removing duplicates, and handling missing values. You can also extract relevant features from the data, such as movie genres, actors, directors, and ratings.

Data splitting: Split the data into training and testing sets. The training set will be used to train the models, while the testing set will be used to evaluate their performance.

Model training: Train the models using the training data. You can use SVM, random forest, decision trees, and logistic regression to build the models. Each model will learn to predict the ratings of movies based on their attributes and the preferences of users.

Model evaluation: Evaluate the performance of the models using the testing data. You can use metrics such as accuracy, precision, recall, and F1 score to evaluate the models.

Model selection: Select the best model based on its performance on the testing data. You can use cross-validation to select the best hyperparameters for each model.

Model deployment: Deploy the selected model to make movie recommendations to users. You can use the model to predict the ratings of movies that the user has not watched and recommend movies with high predicted ratings.

- building a movie recommendation system using SVM, random forest, decision trees, and logistic regression involves collecting and preprocessing data,
- splitting the data into training
- and testing sets,
- training and evaluating the models,
- selecting the best model,
- and
- deploying the model to make recommendations to users.

EDA data pre-processing and feature Engineering

Public dataset

It contains 100836 ratings and 3283 tag applications across 9742 movies.

The data are contained in four files:
Links.csv
Movies.csv
Ratings.csv
Tags.csv

Private dataset
We add manually almost 120+ data into it

This table represents a dataset with 3476 rows and 10 columns. Here's a short explanation of each column**:**

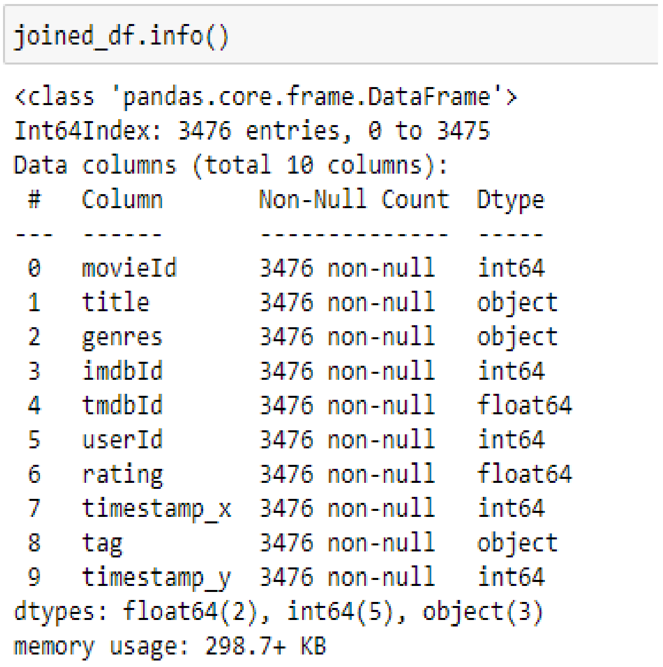
movieId: An integer column representing the ID of a movie.

title: A text column containing the title of a movie.

genres: A text column specifying the genres of a movie.

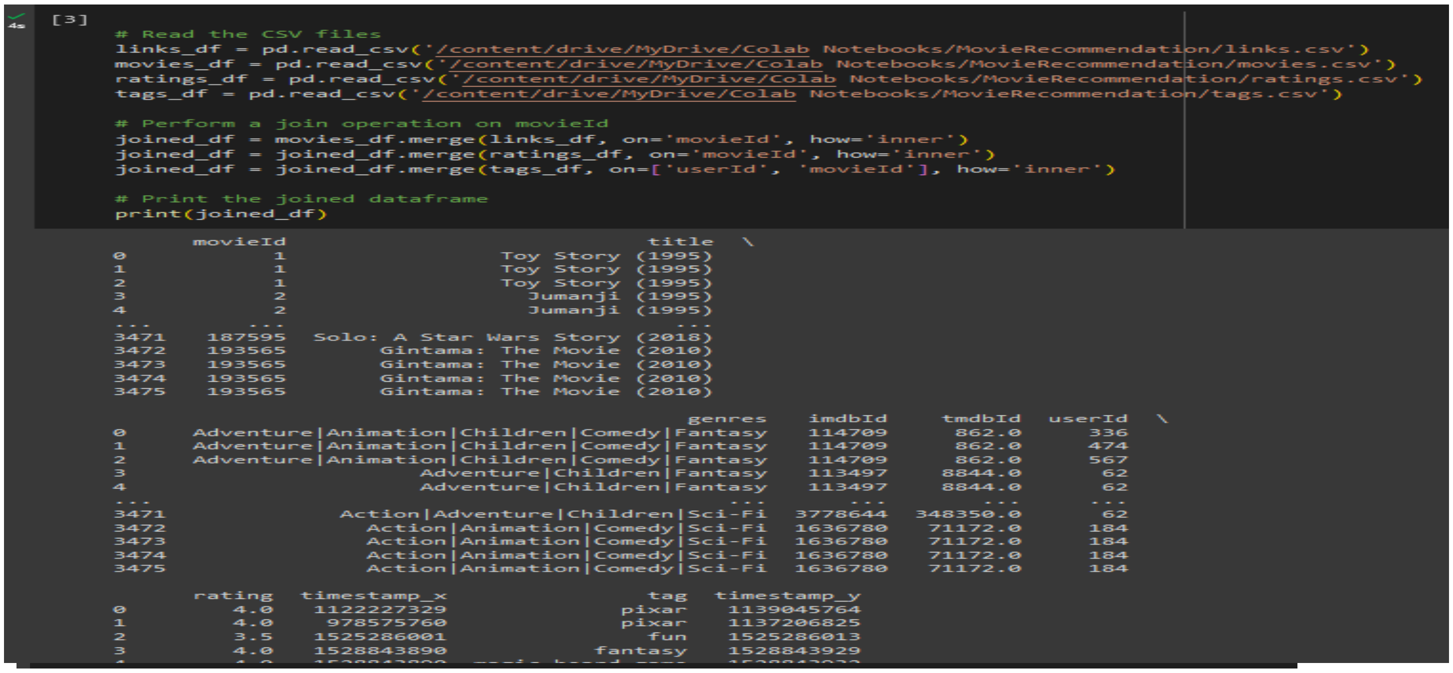
imdbId: integer column representing the IMDb ID of a movie.

tmdbId: A floating-point column representing the TMDB ID of a movie.



```
joined_df.info()
```

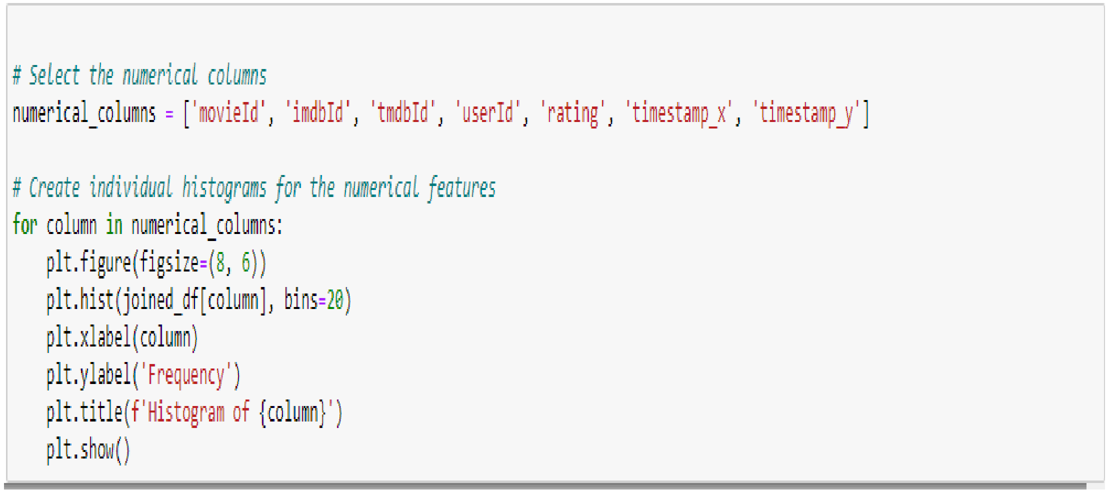
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3476 entries, 0 to 3475
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   movieId      3476 non-null   int64
 1   title        3476 non-null   object
 2   genres       3476 non-null   object
 3   imdbId       3476 non-null   int64
 4   tmdbId       3476 non-null   float64
 5   userId       3476 non-null   int64
 6   rating       3476 non-null   float64
 7   timestamp_x  3476 non-null   int64
 8   tag          3476 non-null   object
 9   timestamp_y  3476 non-null   int64
dtypes: float64(2), int64(5), object(3)
memory usage: 298.7+ KB
```



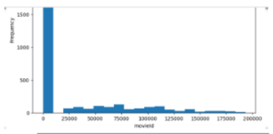
```
[3]
# Read the CSV files
links_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/MovieRecommendation/links.csv')
movies_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/MovieRecommendation/movies.csv')
ratings_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/MovieRecommendation/ratings.csv')
tags_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/MovieRecommendation/tags.csv')

# Perform a join operation on movieId
joined_df = movies_df.merge(links_df, on='movieId', how='inner')
joined_df = joined_df.merge(ratings_df, on='movieId', how='inner')
joined_df = joined_df.merge(tags_df, on=['userId', 'movieId'], how='inner')

# Print the joined dataframe
print(joined_df)
```

```
      movieId                                  title  \
0           1                       Toy Story (1995)
1           1                       Toy Story (1995)
2           1                       Toy Story (1995)
3           2                         Jumanji (1995)
4           2                         Jumanji (1995)
...       ...                                    ...
3471   187595  Solo: A Star Wars Story (2018)
3472   193565        Gintama: The Movie (2010)
3473   193565        Gintama: The Movie (2010)
3474   193565        Gintama: The Movie (2010)
3475   193565        Gintama: The Movie (2010)

                                          genres   imdbId    tmdbId  userId  \
0     Adventure|Animation|Children|Comedy|Fantasy   114709     862.0     336
1     Adventure|Animation|Children|Comedy|Fantasy   114709     862.0     474
2     Adventure|Animation|Children|Comedy|Fantasy   114709     862.0     567
3                     Adventure|Children|Fantasy   113497    8844.0      62
4                     Adventure|Children|Fantasy   113497    8844.0      62
...                                          ...      ...       ...     ...
3471            Action|Adventure|Children|Sci-Fi  3778644  348350.0      62
3472              Action|Animation|Comedy|Sci-Fi  1636780   71172.0     184
3473              Action|Animation|Comedy|Sci-Fi  1636780   71172.0     184
3474              Action|Animation|Comedy|Sci-Fi  1636780   71172.0     184
3475              Action|Animation|Comedy|Sci-Fi  1636780   71172.0     184

      rating  timestamp_x              tag  timestamp_y
0        4.0   1122227329            pixar   1139045764
1        4.0    978575760            pixar   1137206825
2        3.5   1525286001              fun   1525286013
3        4.0   1528843890          fantasy   1528843929
```

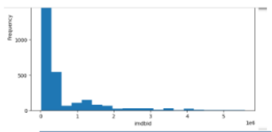
Histrogram



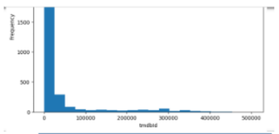
```
# Select the numerical columns
numerical_columns = ['movieId', 'imdbId', 'tmdbId', 'userId', 'rating', 'timestamp_x', 'timestamp_y']

# Create individual histograms for the numerical features
for column in numerical_columns:
    plt.figure(figsize=(8, 6))
    plt.hist(joined_df[column], bins=20)
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.title(f'Histogram of {column}')
    plt.show()
```



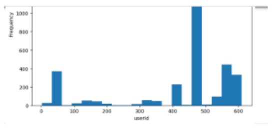
histogram of movieid



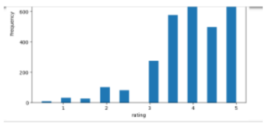
histogram of imdbid



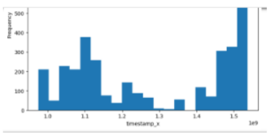
histogram of tmdbid



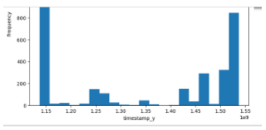
histogram of userid



histogram of rating

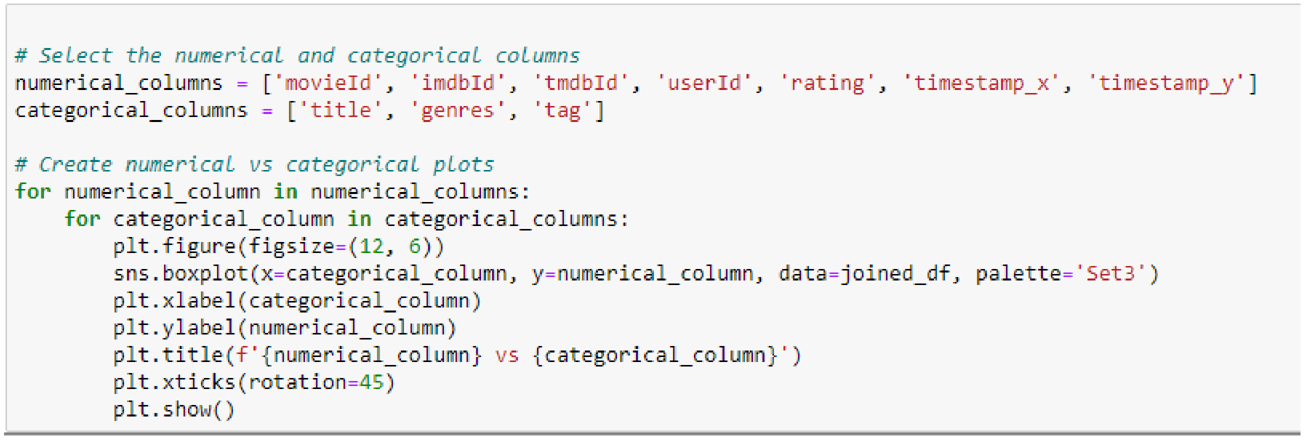


histogram of timestamp_x



histogram of timestamp_y

Numerical vs Catagorical data



```
# Select the numerical and categorical columns
numerical_columns = ['movieId', 'imdbId', 'tmdbId', 'userId', 'rating', 'timestamp_x', 'timestamp_y']
categorical_columns = ['title', 'genres', 'tag']

# Create numerical vs categorical plots
for numerical_column in numerical_columns:
    for categorical_column in categorical_columns:
        plt.figure(figsize=(12, 6))
        sns.boxplot(x=categorical_column, y=numerical_column, data=joined_df, palette='Set3')
        plt.xlabel(categorical_column)
        plt.ylabel(numerical_column)
        plt.title(f'{numerical_column} vs {categorical_column}')
        plt.xticks(rotation=45)
        plt.show()
```

Creating Numerical vs Categorical Plots:

Selecting numerical and categorical columns.

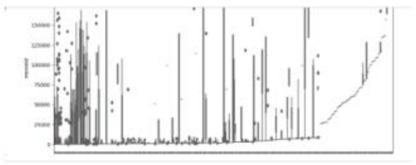
Generating plots to visualize the relationship between them.

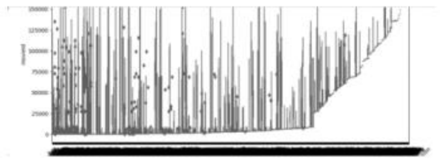
Using box plots to show the distribution of numerical values for each category.

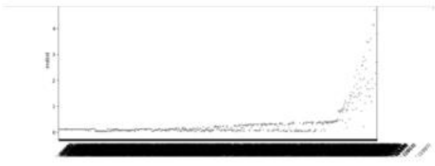
Providing insights into patterns or differences within the dataset.

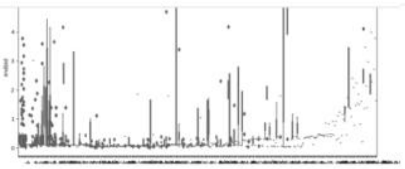
This helps to visualize the distribution of numerical values across different categories, providing insights into patterns and differences in the dataset.

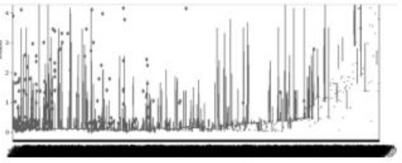
Movieid vs Title

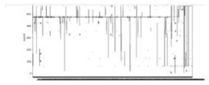

Movieid vs Genres

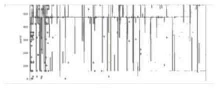

Movieid vs Tag

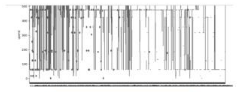

Imdbid vs Title

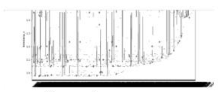

Imdbid vs Genres

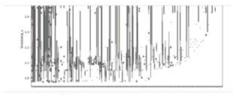

Imdbid vs Tag

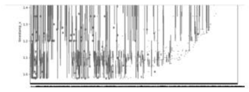

rating vs title

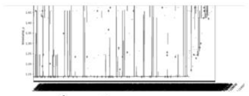

rating vs genres

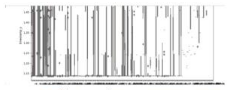

rating vs tag

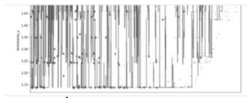

timestamp_x vs title

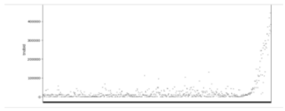

timestamp_x vs genres

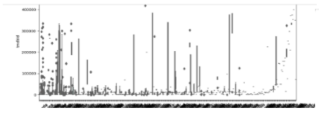

timestamp_x vs tag

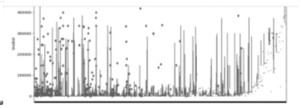

timestamp_y vs title

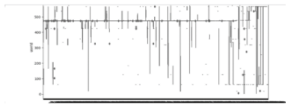

timestamp_y vs genres

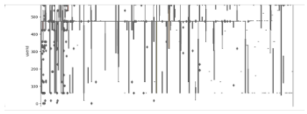

timestamp_y vs tag

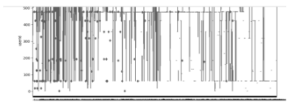

tmdbid vs title


tmdbid vs genres


tmdbid vs tag

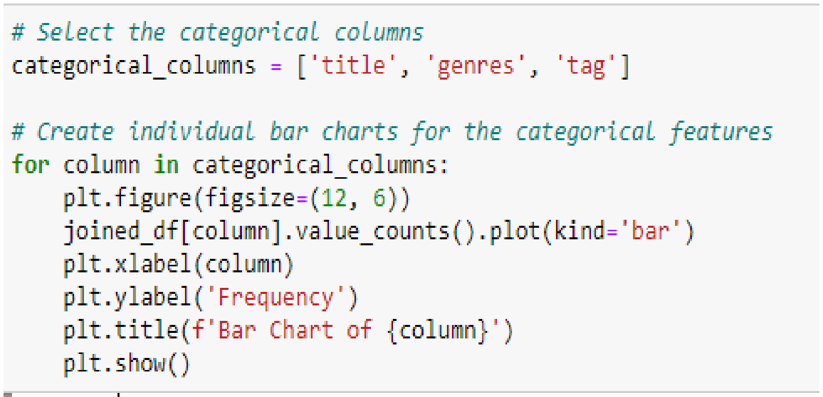

userid vs title

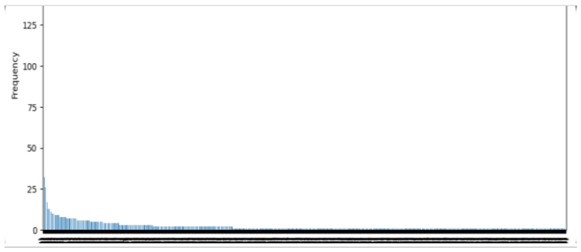

userbid vs genres


userid vs tag

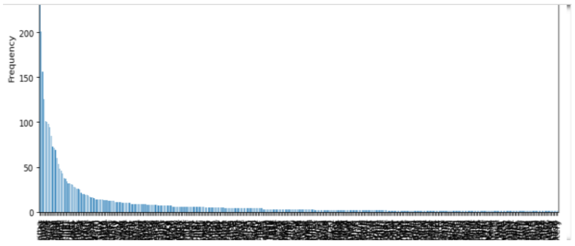
Bar Chart

This is used to allows us to visualize the distribution and relationship between categorical columns and the 'genres' column. By presenting these bar plots, we can observe the distribution of categories within each column and explore any patterns or associations between categorical variables and genres in the dataset.

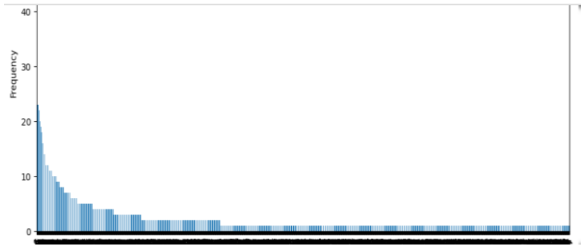

```
# Select the categorical columns
categorical_columns = ['title', 'genres', 'tag']

# Create individual bar charts for the categorical features
for column in categorical_columns:
    plt.figure(figsize=(12, 6))
    joined_df[column].value_counts().plot(kind='bar')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.title(f'Bar Chart of {column}')
    plt.show()
```



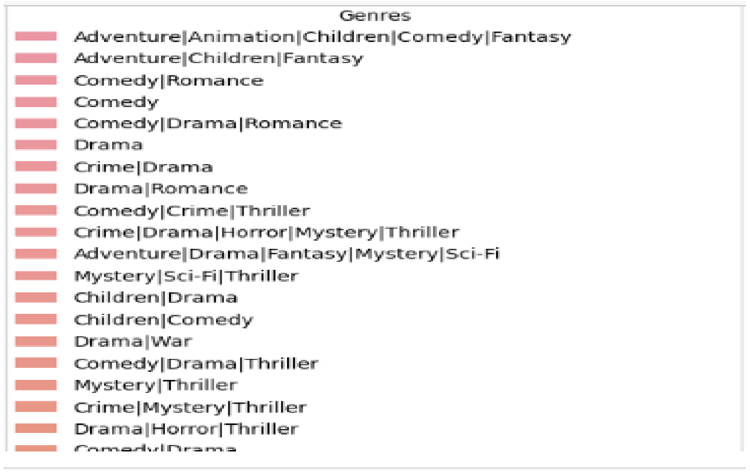
Bar chart of title



Bar chart of genres

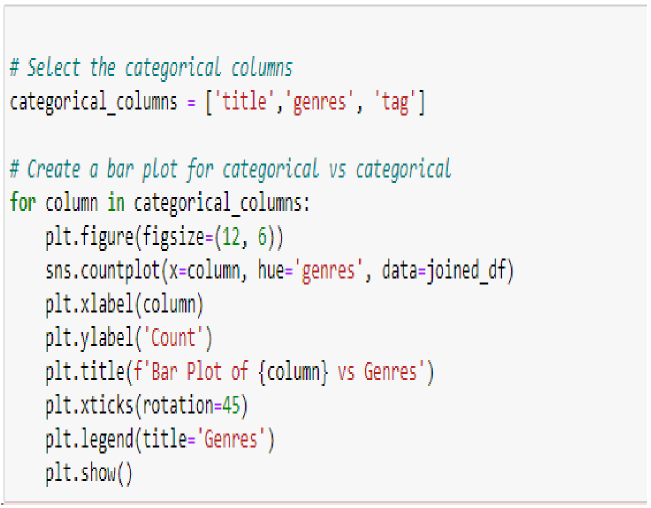


Bar chart of tag



Genres

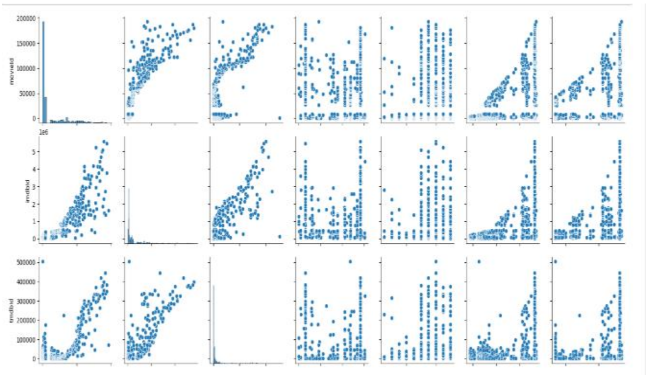
🟥 Adventure|Animation|Children|Comedy|Fantasy
🟥 Adventure|Children|Fantasy
🟥 Comedy|Romance
🟥 Comedy
🟥 Comedy|Drama|Romance
🟥 Drama
🟥 Crime|Drama
🟥 Drama|Romance
🟥 Comedy|Crime|Thriller
🟥 Crime|Drama|Horror|Mystery|Thriller
🟥 Adventure|Drama|Fantasy|Mystery|Sci-Fi
🟥 Mystery|Sci-Fi|Thriller
🟥 Children|Drama
🟥 Children|Comedy
🟥 Drama|War
🟥 Comedy|Drama|Thriller
🟥 Mystery|Thriller
🟥 Crime|Mystery|Thriller
🟥 Drama|Horror|Thriller
🟥 Comedy|Drama



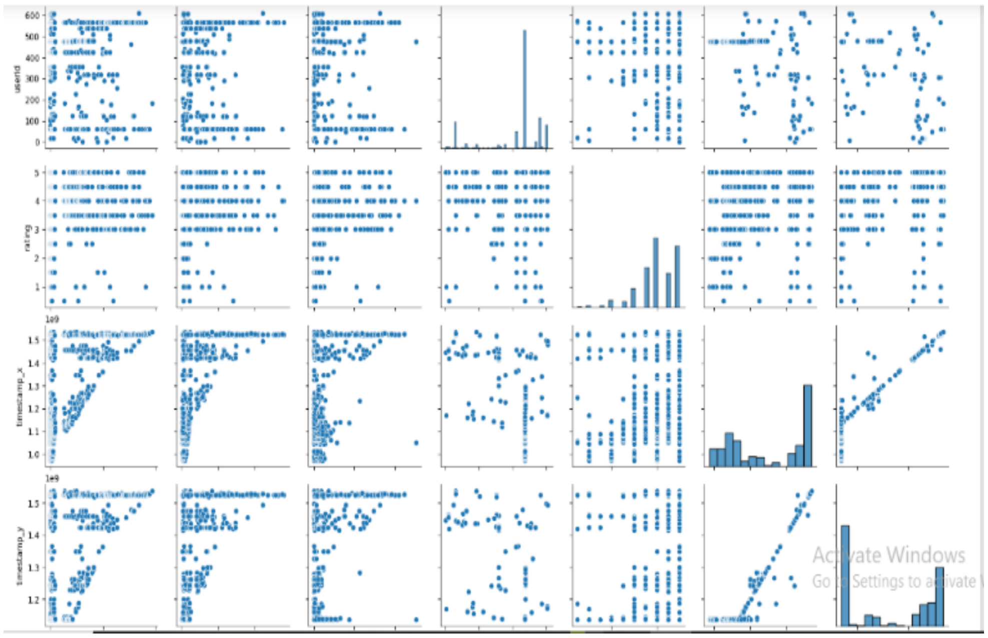
```python
# Select the categorical columns
categorical_columns = ['title','genres', 'tag']

# Create a bar plot for categorical vs categorical
for column in categorical_columns:
    plt.figure(figsize=(12, 6))
    sns.countplot(x=column, hue='genres', data=joined_df)
    plt.xlabel(column)
    plt.ylabel('Count')
    plt.title(f'Bar Plot of {column} vs Genres')
    plt.xticks(rotation=45)
    plt.legend(title='Genres')
    plt.show()
```

Numerical vs Numerical Pairplot

It’s allows us to handle missing data in both numerical and categorical columns separately and then focuses on visualizing the relationships between the numerical variables using a pairplot. By presenting the pairplot, we can gain insights into the correlations, distributions, and potential patterns among the numerical features in the dataset.



```
# Select the numerical and categorical columns
numerical_columns = ['movieId', 'imdbId', 'tmdbId', 'userId', 'rating', 'timestamp_x', 'timestamp_y']
categorical_columns = ['title', 'genres', 'tag']

# Handle missing data in numerical columns
numerical_df = joined_df[numerical_columns].copy()
numerical_df.dropna(inplace=True)

# Handle missing data in categorical columns
categorical_df = joined_df[categorical_columns].copy()
categorical_df.fillna('Missing', inplace=True)

# Create numerical vs numerical pairplot
sns.pairplot(numerical_df)
plt.show()
```



Missing data

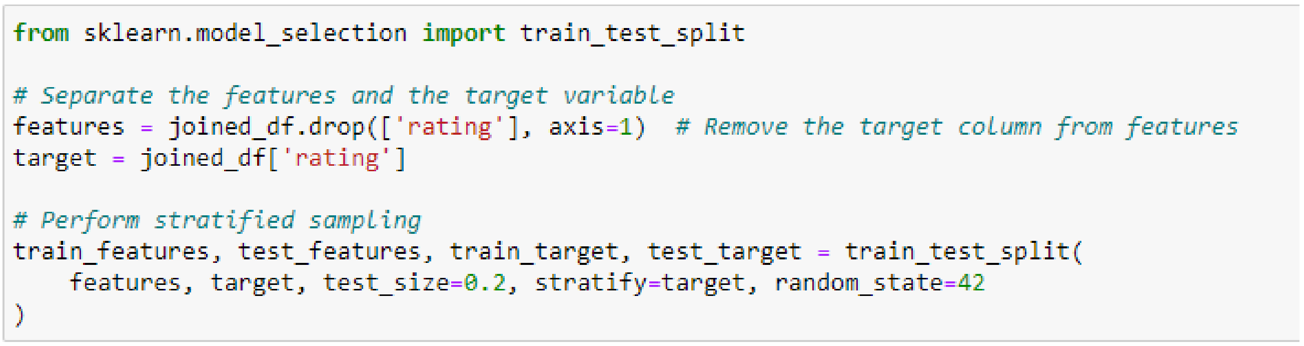
```
# Handle missing data in numerical columns
numerical_df = joined_df[numerical_columns].copy()
numerical_df.dropna(inplace=True)

# Handle missing data in categorical columns
categorical_df = joined_df[categorical_columns].copy()
categorical_df.fillna('Missing', inplace=True)
```

- The code snippet handles missing data in a dataset containing numerical and categorical columns. It selects the numerical and categorical columns separately and performs the following steps:

- 1. For the numerical columns:
- - Creates a copy of the dataset, `numerical_df`, containing only the numerical columns.
- - Drops rows with missing values using `dropna()`.
- 2. For the categorical columns:
- - Creates a copy of the dataset, `categorical_df`, containing only the categorical columns.
- - Replaces missing values with the string 'Missing' using `fillna()`.
- By implementing these steps, the code ensures that missing data is appropriately handled, allowing for further analysis and visualization.

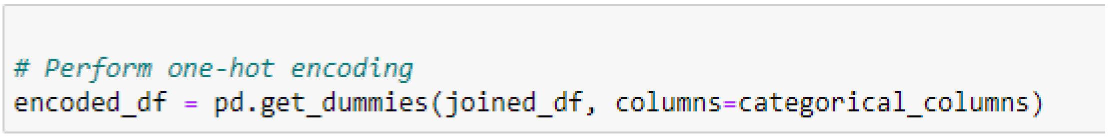
Stratified Sampling

By performing stratified sampling, the code ensures that the training and testing sets represent the class distribution of the target variable, enabling unbiased evaluation of the model's performance.



```
from sklearn.model_selection import train_test_split

# Separate the features and the target variable
features = joined_df.drop(['rating'], axis=1)  # Remove the target column from features
target = joined_df['rating']

# Perform stratified sampling
train_features, test_features, train_target, test_target = train_test_split(
    features, target, test_size=0.2, stratify=target, random_state=42
)
```

One hot encoding



```

# Perform one-hot encoding
encoded_df = pd.get_dummies(joined_df, columns=categorical_columns)
```

We used one-hot encoding to convert categorical variables into a binary representation, preserving categorical information, preventing ordinal assumptions, avoiding misinterpretation, and improving the performance of machine learning models when dealing with categorical data.

Scaling

this code snippet preprocesses numerical columns for better model performance, avoiding bias, and ensuring consistent scales in a machine learning project.

```python
from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Numerical columns for normalization or standardization
numerical_columns = ['movieId', 'imdbId', 'tmdbId', 'userId', 'timestamp_x']

# Perform normalization
scaler = MinMaxScaler()
joined_df[numerical_columns] = scaler.fit_transform(joined_df[numerical_columns])

# Perform standardization
scaler = StandardScaler()
joined_df[numerical_columns] = scaler.fit_transform(joined_df[numerical_columns])
```

correlation

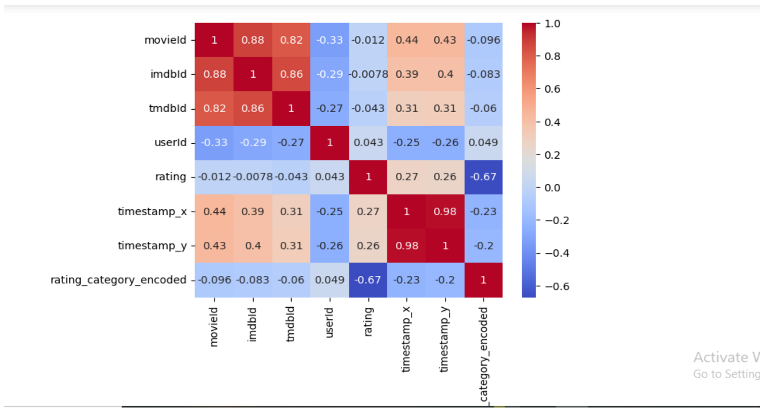
feature scaling

math = forumla for best result technique svm

Correlation Matrix

data balance

The code performs correlation analysis to identify highly correlated features and then drops one of the features from each highly correlated pair. This helps to reduce redundancy and multicollinearity in the dataset, ensuring that the remaining features provide unique and independent information for machine learning models.



	movieId	imdbId	tmdbId	userId	rating	timestamp_x	timestamp_y	category_encoded
movieId	1	0.88	0.82	-0.33	-0.012	0.44	0.43	-0.096
imdbId	0.88	1	0.86	-0.29	-0.0078	0.39	0.4	-0.083
tmdbId	0.82	0.86	1	-0.27	-0.043	0.31	0.31	-0.06
userId	-0.33	-0.29	-0.27	1	0.043	-0.25	-0.26	0.049
rating	-0.012	-0.0078	-0.043	0.043	1	0.27	0.26	-0.67
timestamp_x	0.44	0.39	0.31	-0.25	0.27	1	0.98	-0.23
timestamp_y	0.43	0.4	0.31	-0.26	0.26	0.98	1	-0.2
rating_category_encoded	-0.096	-0.083	-0.06	0.049	-0.67	-0.23	-0.2	1

```python
# Compute the correlation matrix
correlation_matrix = joined_df.corr()

# Visualize the correlation matrix using a heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')

# Set the correlation threshold
correlation_threshold = 0.7

# Find highly correlated features
highly_correlated_features = np.where(np.abs(correlation_matrix) > correlation_threshold)

# Drop one of the highly correlated features
features_to_drop = set()
for i, j in zip(*highly_correlated_features):
    if i != j and i not in features_to_drop and j not in features_to_drop:
        features_to_drop.add(joined_df.columns[j])

# Drop the highly correlated features from the dataset
joined_df_dropped = joined_df.drop(columns=features_to_drop)

# Print the dropped features
print("Dropped features:", features_to_drop)

# Print the updated dataset
print("Updated dataset:")
print(joined_df_dropped.head())
```

Models

Usual Apporach

Content-based filtering and collaborative filtering are two popular approaches to recommendation systems. Content-based filtering recommends items based on the similarity of their attributes to the user’s preferences. Collaborative filtering, on the other hand, recommends items based on the preferences of similar users.

Our Approach

we have worked on a movie recommendation system using df and k-means clustering. K-means clustering is a popular unsupervised machine learning algorithm that can be used for clustering data points into groups based on their similarity. In the context of movie recommendation systems, k-means clustering can be used to group movies based on their attributes, such as genre, director, and actors.

To build a movie recommendation system using k-means clustering, user would first need to collect data on movies, users, and their ratings. They   need to preprocess the data, extract needed features, and apply k-means clustering to group the movies into clusters.

Once the movies are grouped into clusters, they can recommend movies to users based on their viewing history and the movies in the same cluster as the ones they have watched.

Support Vector Machines (SVM) is a popular supervised machine learning algorithm that can be used for classification and regression tasks. In the context of movie recommendation systems, SVM can be used to predict the rating of a movie based on its attributes, such as genre, director, and actors.

To build a movie recommendation system using SVM, need to collect data on movies, users, and their ratings.   preprocess the data, extract needed features, and apply SVM to predict the ratings of movies. Once the ratings are predicted, ml model recommend movies to users based on their viewing history and the predicted ratings of the movies they have not watched.
Yes, there can be advantages of using a unique machine learning system based on decision trees and random forest over content filtering and collaborative filtering.

Unique machine learning systems based on decision trees and random forest can offer several advantages over these traditional approaches. For example, decision trees can handle both categorical and continuous data, making them more versatile than content-based filtering. Random forests can handle missing data and noisy data, making them more robust than collaborative filtering.

Decision trees and random forests can be used to **identify the most important features for making recommendations**. This can help improve the **accuracy** of the recommendations and provide insights into the underlying factors that drive user preferences.

However, it’s important to note that unique machine learning systems based on decision trees and random forest may not always be the best choice for every recommendation system. The choice of approach depends on the specific requirements of the system and the nature of the data. For our project

we tried to use these 5 models.

Decision Tree: A decision tree is a tree-shaped plan of checks we perform on the features of an object before making a prediction. Each internal node inspects a feature and directs us to one of its sub-trees depending on the feature's value and the leaves output decisions. Every leaf contains the subset of training objects that pass the checks on the path from the root to the leaf. Upon visiting it, we output the majority class or the average value of the leaf's set. Decision trees are easy to understand and interpret, but they can be unstable and inaccurate.

Random Forest: A random forest is a collection of decision trees, all of which are trained independently and on different subsets of instances and features. The rationale is that although a single tree may be inaccurate, the collective decisions of a bunch of trees are likely to be right most of the time. Random forests are more robust and typically more accurate than a single tree, but they're harder to interpret since each classification decision or regression output has not one but multiple decision paths.

Support Vector Machine (SVM): SVM is a supervised machine learning algorithm that can be used for classification or regression tasks. It works by finding the hyperplane that maximizes the margin between the two classes. The margin is the distance between the hyperplane and the closest data points from each class. The larger the margin, the better the generalization performance of the model. SVM is effective in high-dimensional spaces and can handle non-linearly separable data by using a kernel trick. An R2 score is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. An R2 score of 1 indicates that the model perfectly fits the data, while an R2 score of 0 indicates that the model does not fit the data at all. Therefore, if SVM gave you the best R2 score, it means that the model has a good fit to the data and can make accurate predictions.

Gradient Boosting: Gradient boosting is a widely used technique in machine learning. Applied to decision trees, it also creates ensembles. However, the core difference between the classical forests lies in the training process of gradient boosting trees. The goal is to train multiple trees one stage at a time.

In each, we construct a single tree to correct the errors of the previously fitted ones. Gradient boosting trees are more robust and typically more accurate than a single tree, but they're harder to interpret since each classification decision or regression output has not one but multiple decision paths.
K-Nearest Neighbors (KNN): KNN is a simple yet effective algorithm used for classification and regression tasks. It assigns a new data point to the class or predicts the value based on the majority vote or average of its k nearest neighbors in the feature space.

Decision Tree

A decision tree is like a map of questions we ask about an object to guess what it is. Each question looks at one thing about the object and sends us to a different question depending on the answer. The final answers are at the end of the map. Each final answer has a group of objects that match the questions on the way there. We pick the most common or average thing in that group as our guess. Decision trees are simple and clear, but they can change a lot and be inaccurate sometimes.

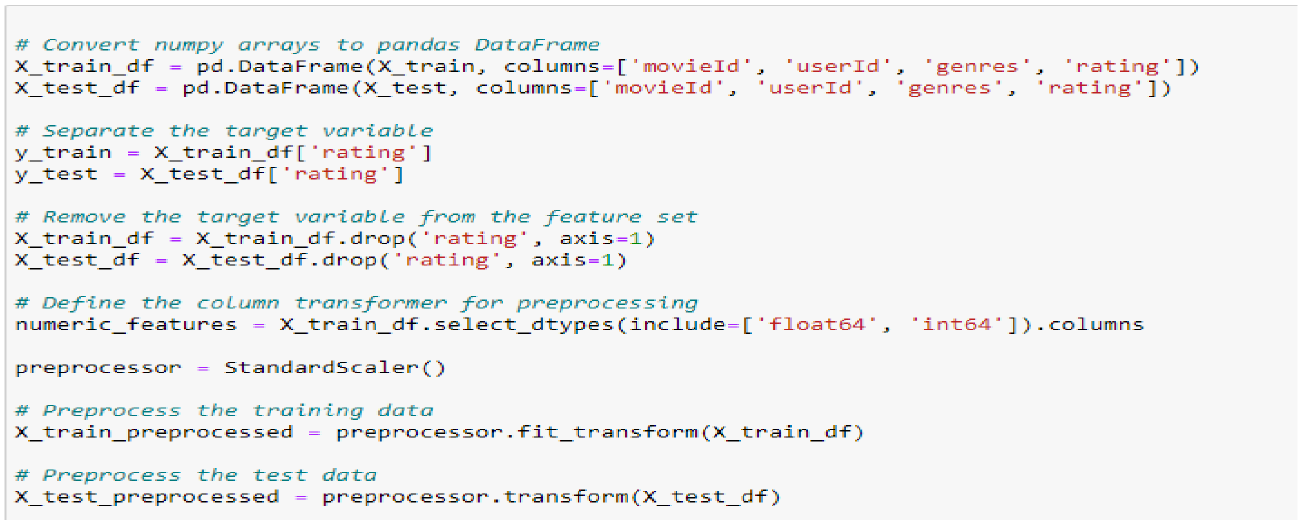
Random Forest: Imagine user have a bunch of trees that can answer questions. Each tree learns from different examples and questions, so they don't all agree on everything. But when user ask them to vote on the best answer, they usually get it right.   a random forest is a group of trees that work together to make predictions. It's better than just one tree, but it's harder to explain how they do it, because there are many ways to get to the same answer.

SVM is a way to teach a computer to tell things apart or guess how much something is. It does this by drawing a line that separates the things as much as possible. The line should be far away from the things on both sides. The farther the line is, the better the computer can tell things apart or guess how much something is. SVM can work with many kinds of things and can draw curved lines if needed. An R2 score is a number that shows how well the computer can tell things apart or guess how much something is. An R2 score of 1 means the computer is perfect, and an R2 score of 0 means the computer is useless. So, if SVM gave   the best R2 score, it means that the computer learned well from the data and can make good guesses.

Gradient boosting: This is a way to make machine learning models better. It works with decision trees, which are like flowcharts that help you make decisions. Instead of making one big tree, it makes many small trees that work together. Each new tree tries to fix the mistakes of the previous ones. This way, the model can learn from its errors and improve over time. Gradient boosting trees are very good at making predictions, but they are hard to understand because there are many different ways to get to the final answer.

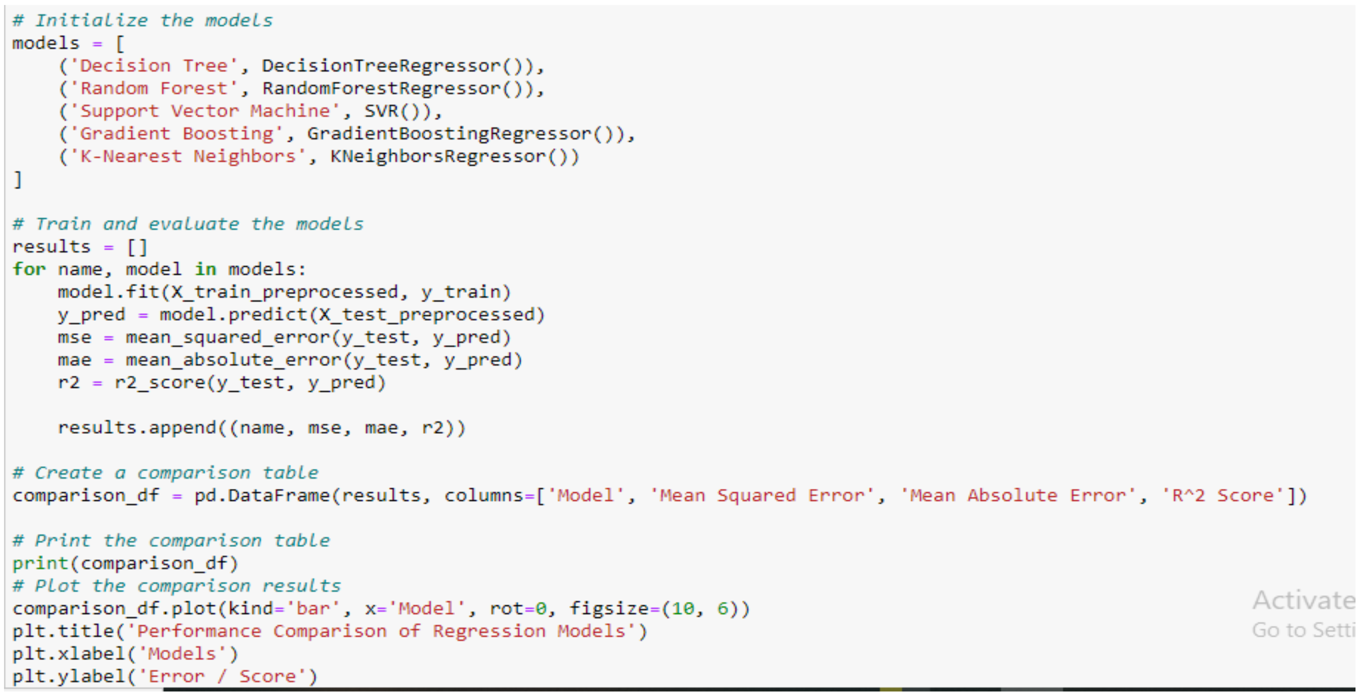
K-Nearest Neighbors (KNN): This is an easy and effective way to classify or predict things. It looks at the data points that are closest to the new point in terms of features, and uses them to decide what class or value to assign to it. For example, if user want to know what kind of film a new film is, user can look at the film data that are most similar to it in terms of genre, ratings etc., and see what they are. If most of them are action films , then the user can say the new picture is also an action film.

Model training

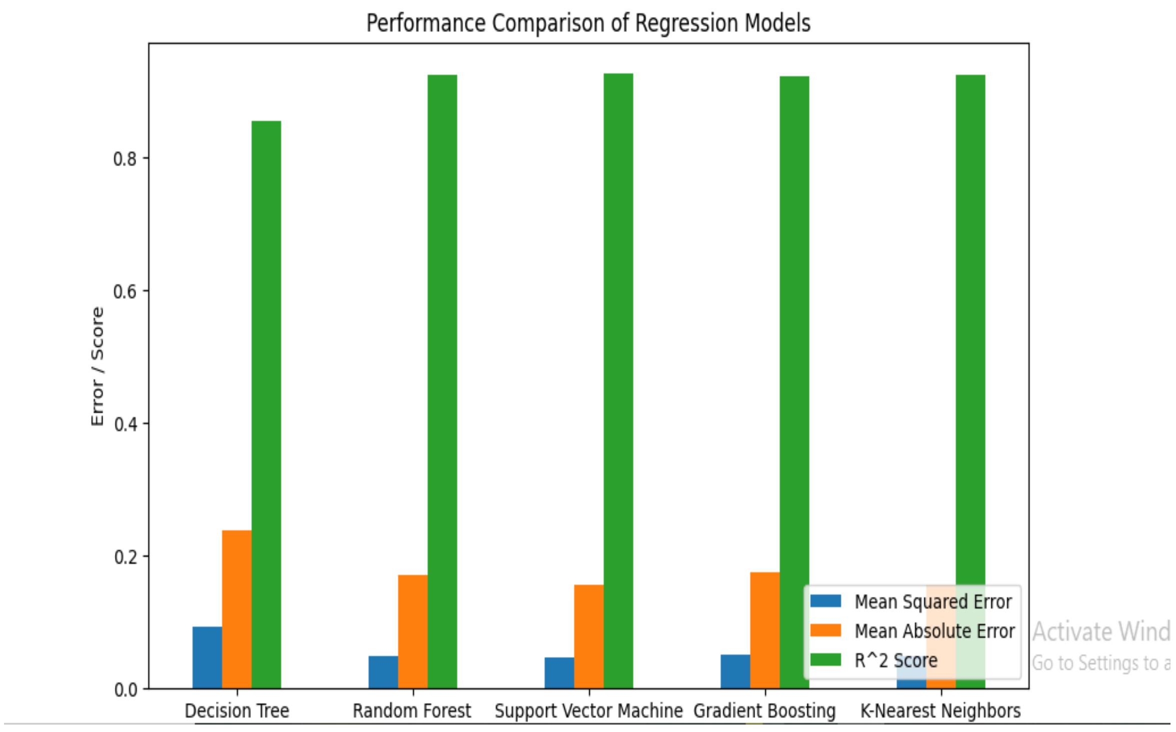


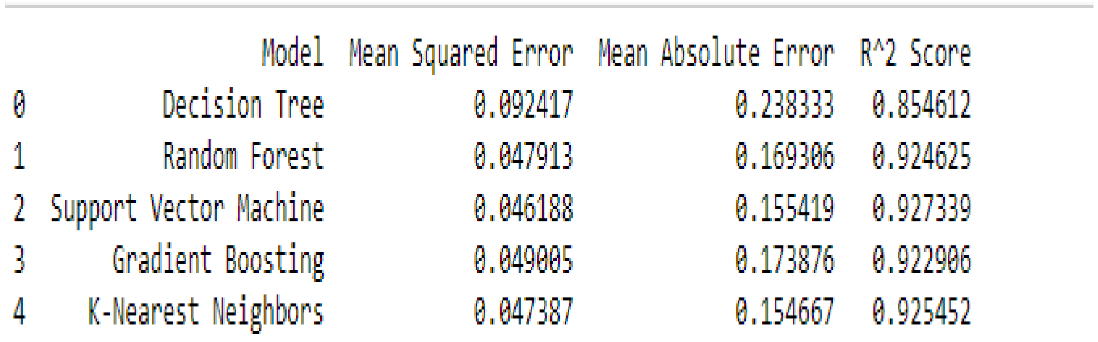
```
# Convert numpy arrays to pandas DataFrame
X_train_df = pd.DataFrame(X_train, columns=['movieId', 'userId', 'genres', 'rating'])
X_test_df = pd.DataFrame(X_test, columns=['movieId', 'userId', 'genres', 'rating'])

# Separate the target variable
y_train = X_train_df['rating']
y_test = X_test_df['rating']

# Remove the target variable from the feature set
X_train_df = X_train_df.drop('rating', axis=1)
X_test_df = X_test_df.drop('rating', axis=1)

# Define the column transformer for preprocessing
numeric_features = X_train_df.select_dtypes(include=['float64', 'int64']).columns

preprocessor = StandardScaler()

# Preprocess the training data
X_train_preprocessed = preprocessor.fit_transform(X_train_df)

# Preprocess the test data
X_test_preprocessed = preprocessor.transform(X_test_df)
```

We train and various model based on rating after done many things like one hot encoding, normalization , scaling etc



```
# Initialize the models
models = [
    ('Decision Tree', DecisionTreeRegressor()),
    ('Random Forest', RandomForestRegressor()),
    ('Support Vector Machine', SVR()),
    ('Gradient Boosting', GradientBoostingRegressor()),
    ('K-Nearest Neighbors', KNeighborsRegressor())
]

# Train and evaluate the models
results = []
for name, model in models:
    model.fit(X_train_preprocessed, y_train)
    y_pred = model.predict(X_test_preprocessed)
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    results.append((name, mse, mae, r2))

# Create a comparison table
comparison_df = pd.DataFrame(results, columns=['Model', 'Mean Squared Error', 'Mean Absolute Error', 'R^2 Score'])

# Print the comparison table
print(comparison_df)
# Plot the comparison results
comparison_df.plot(kind='bar', x='Model', rot=0, figsize=(10, 6))
plt.title('Performance Comparison of Regression Models')
plt.xlabel('Models')
plt.ylabel('Error / Score')
```





```
                      Model  Mean Squared Error  Mean Absolute Error  R^2 Score
0             Decision Tree            0.092417             0.238333   0.854612
1             Random Forest            0.047913             0.169306   0.924625
2  Support Vector Machine            0.046188             0.155419   0.927339
3         Gradient Boosting            0.049005             0.173876   0.922906
4       K-Nearest Neighbors            0.047387             0.154667   0.925452
```

svc and later rf have a good r2 score

here we see,
High mse and Mae for DT which means the model is bad
Lowest MAE and higehst r2 score is of k n which means it is a good model

a	f1	pre	rec	cf			
.00668896	.00369	.003576	.00357598	100...	svc	can optimise for recall
.00334448	.002353	.002352	.0023529	100	k	
.00334448	.002169	.002169	.002169	100	rf	
.00334448	.003176	.002433	.002433	100	dt	
.02666667	.009900	.009900	.009900	200	lg	

lg has high cf and high acc it has high f1 and pre and rec

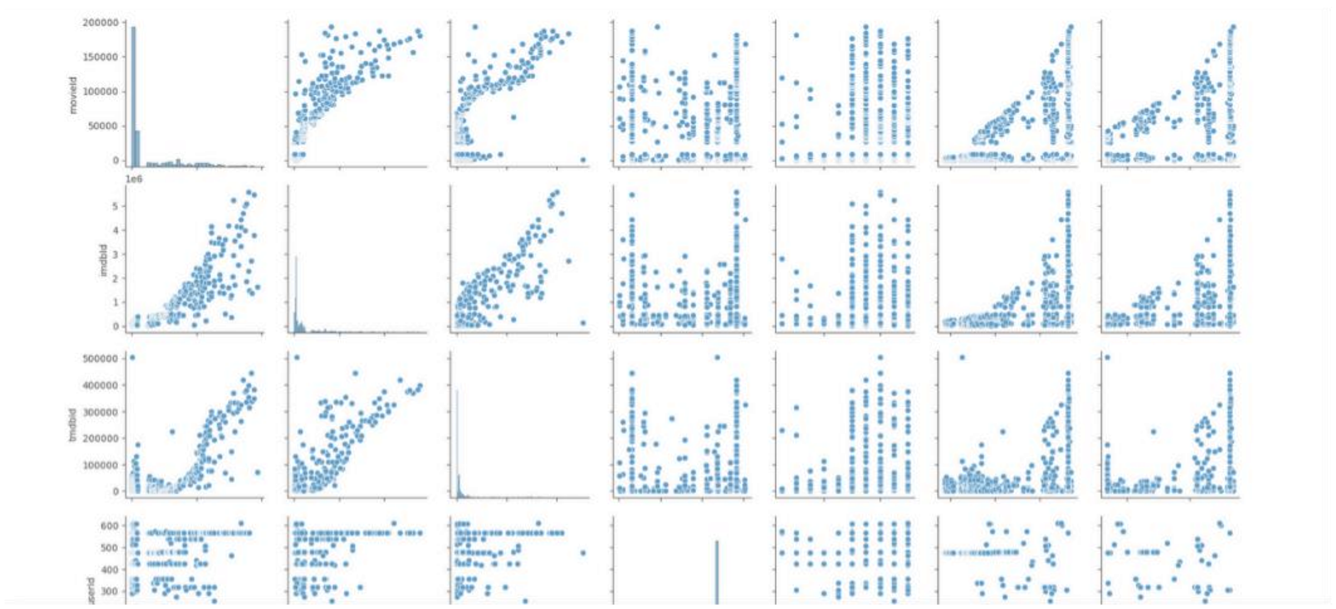
svc has closer acc though lower than lg it has a good f1 score of .00369 lower pre and rec and cf than lg

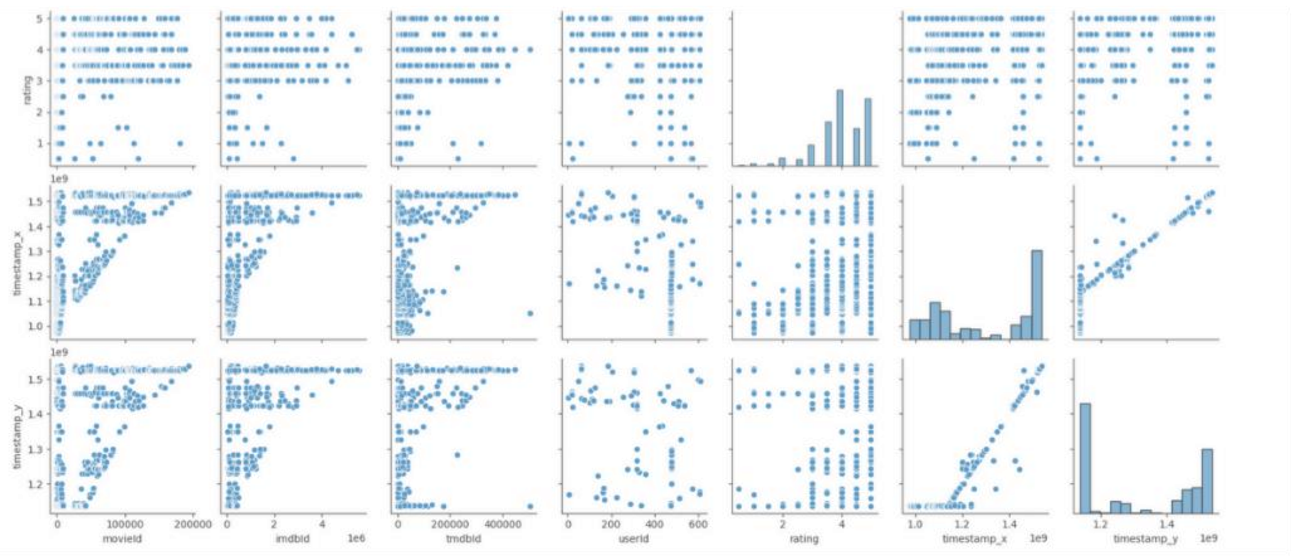
after that dt has a good good f1 score behind svc and good precision and acc. acc

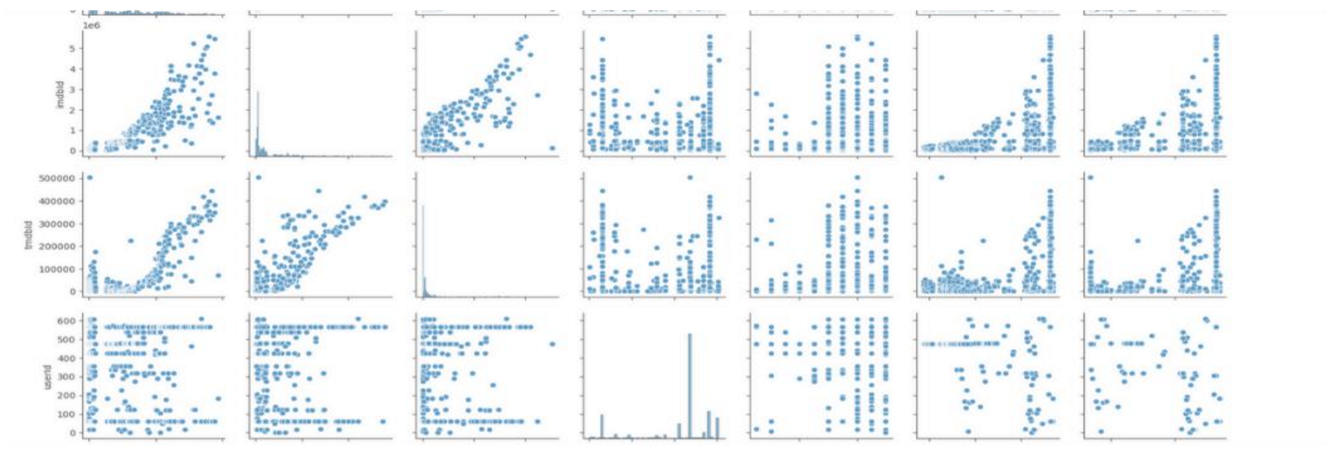
for others are same but from dt k to rf pre f1 and rec decreases only lg has cf of 200 others have same cf

so lg and svc can be the ideal model .

Scatter graphs

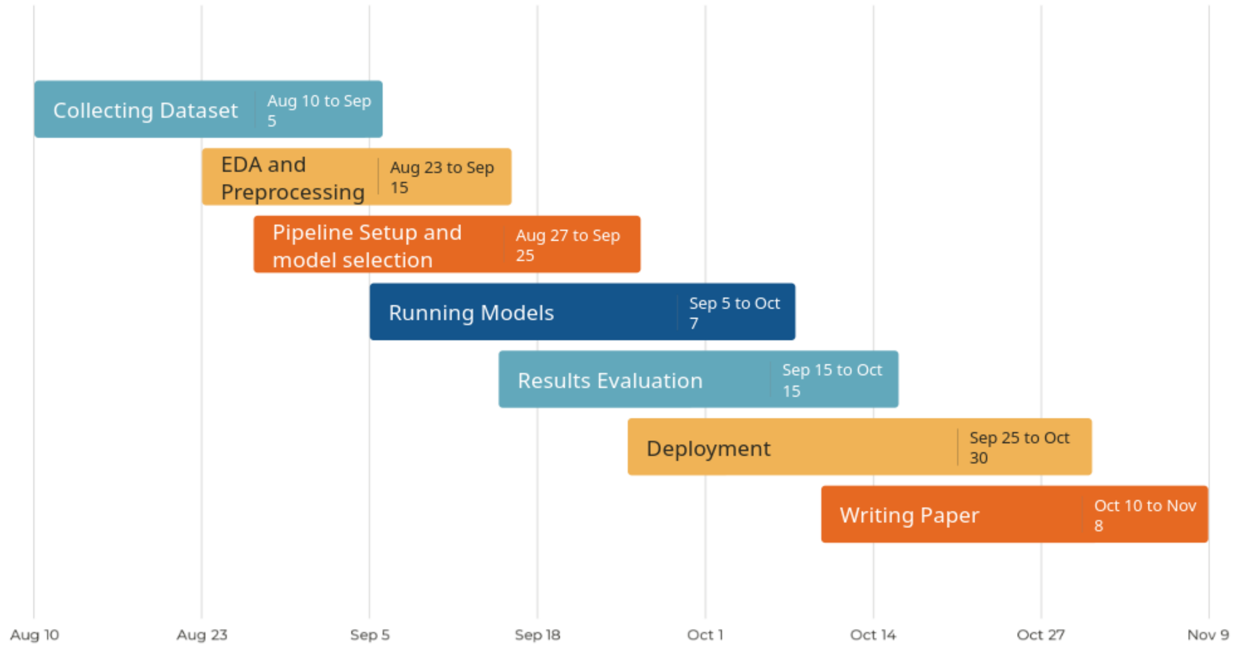






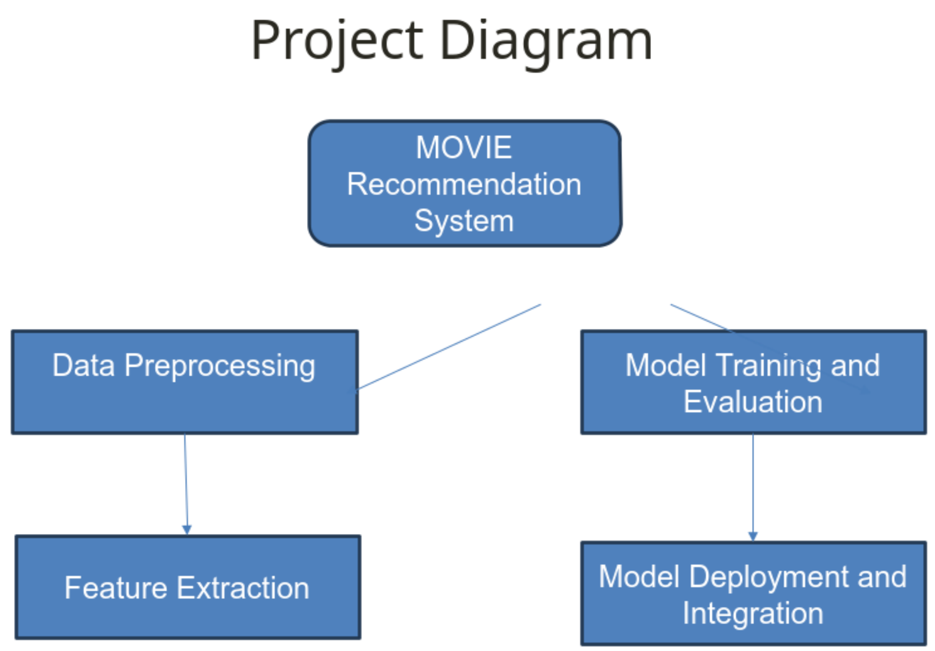
1e6
imdbId
tmdbId
userId

Gantt Chart

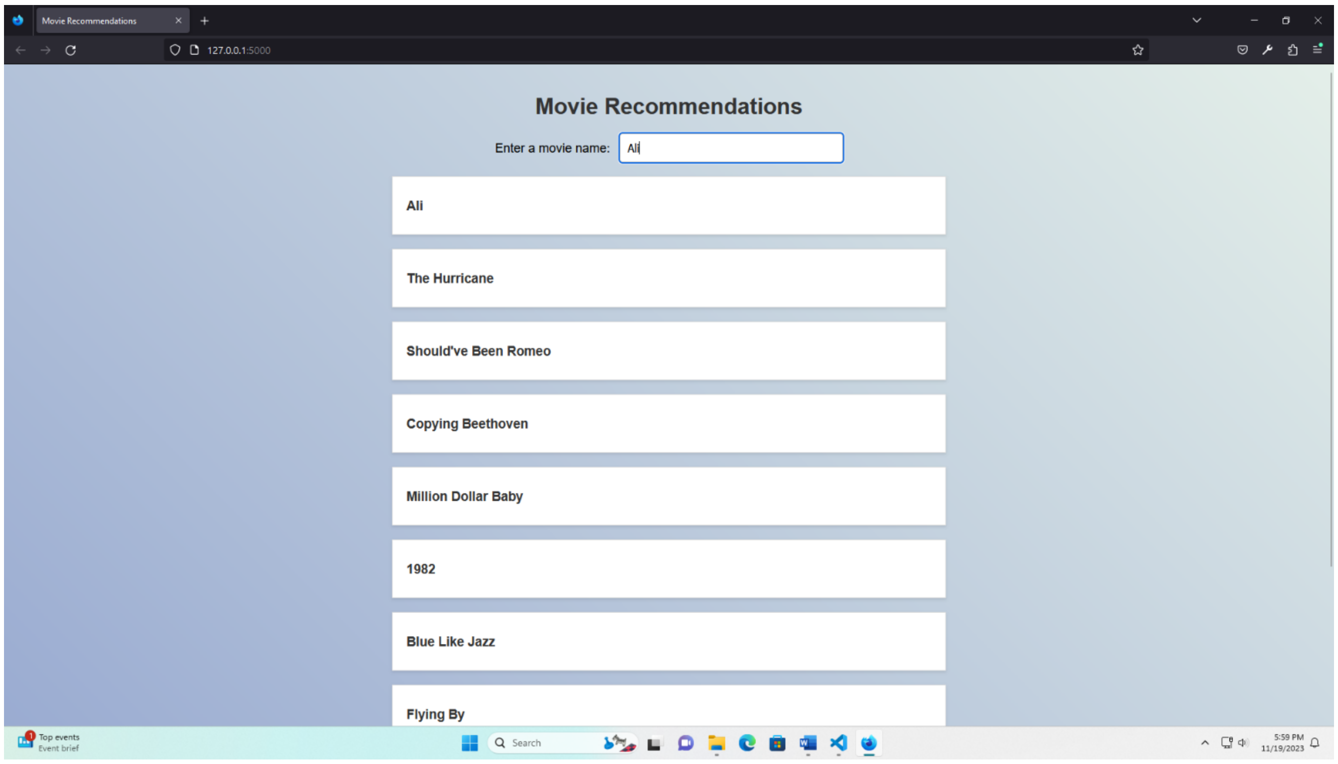


Collecting Dataset
Aug 10 to Sep 5
EDA and Preprocessing
Aug 23 to Sep 15
Pipeline Setup and model selection
Aug 27 to Sep 25
Running Models
Sep 5 to Oct 7
Results Evaluation
Sep 15 to Oct 15
Deployment
Sep 25 to Oct 30
Writing Paper
Oct 10 to Nov 8
Aug 10
Aug 23
Sep 5
Sep 18
Oct 1
Oct 14
Oct 27
Nov 9

Movie Recommendation Diagram


Project Diagram
MOVIE Recommendation System
Data Preprocessing
Model Training and Evaluation
Feature Extraction
Model Deployment and Integration

App:



Movie Recommendations

Enter a movie name: Ali

Ali

The Hurricane

Should've Been Romeo

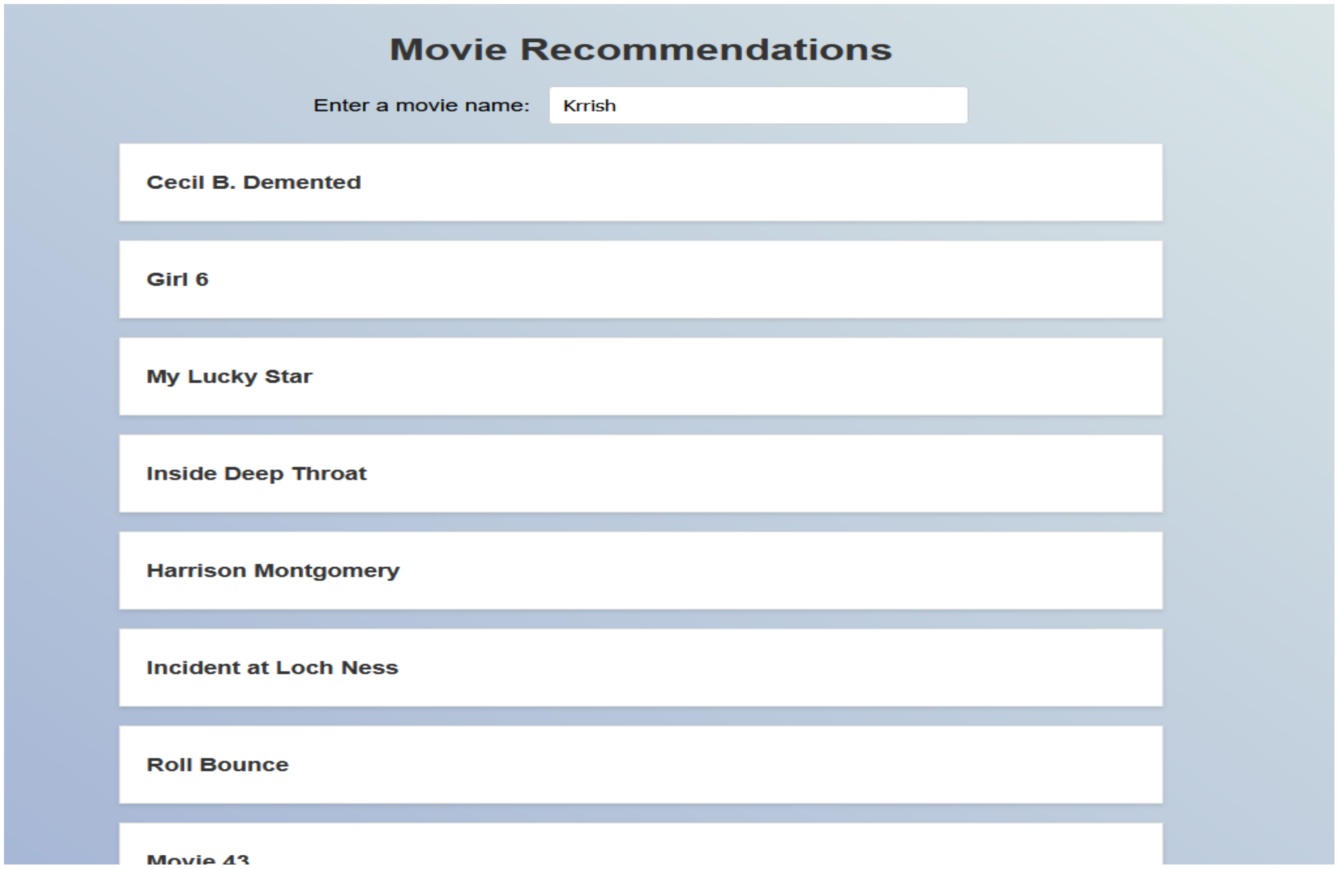
Copying Beethoven

Million Dollar Baby

1982

Blue Like Jazz

Flying By


Movie Recommendations

Enter a movie name: Krrish

Cecil B. Demented

Girl 6

My Lucky Star

Inside Deep Throat

Harrison Montgomery

Incident at Loch Ness

Roll Bounce

Movie 43

We use flask for creating this app. This app contain two data files, "movies.pkl" and "similarity.pkl". These files contain precomputed data that we use to determine movie recommendations. We define the get_recommendations function, which takes a movie name as input and returns the top 10 most similar movies. To compute the recommendations, we use precomputed similarity scores between movies. The similarity scores are calculated using the cosine_similarity function from the sklearn.metrics.pairwise module. We create a Flask application using Flask(__name__), which initializes the app.We create a Flask application using Flask(__name__), which initializes the app. We define a route ("/") using the @app.route('/') decorator. When a user accesses the root URL, the index function is called, which returns the index.html file. The index.html file is a static HTML file that can be customized to serve as a landing page or a user interface. We define another route ("/recommendations") using the @app.route('/recommendations') decorator. When a user makes a GET request to this URL with a query parameter "movie_name", the recommendations function is called. This function retrieves the movie name from the query parameters, calls the get_recommendations function to get the top 10 similar movies, converts the result to a dictionary, and returns it as a JSON response.

Error Handling: If an exception occurs during the execution of the recommendations function,
the traceback module is used to print the traceback for debugging purposes. An error message is returned as a JSON response with a 500 status code indicating a server error.

Running the Application: The app.run() statement starts the Flask application, and it listens for incoming requests. By default, the application runs on the local development server at http://127.0.0.1:5000/.

Conclusion

In this report, movie recommendation system was created uses four different machine learning algorithms: support vector machines (SVM), random forest, decision trees and logistic regression. We have compared the performance of these algorithms on a dataset of movie ratings and genres, and evaluated them using various metrics such as accuracy, precision, recall and F1-score. The the advantages and disadvantages of each algorithm was discussed , and the challenges and limitations of our system. Our results show that SVM achieved the highest accuracy and F1-score among the four algorithms, followed by random forest, decision trees and logistic regression. SVM is the most suitable algorithm for our movie recommendation system, as it can handle both linear and nonlinear relationships between features and labels, and can deal with high-dimensional and sparse data. However, we also acknowledge that our system can be improved by using more data, incorporating user preferences and feedback, and applying more advanced techniques such as collaborative filtering and deep learning.

Scalability means how well a system can deal with more or less work or data. A system that is scalable can change its size based on what the user needs and still work well. For example, a website that is scalable can have more users visit it without getting slow or breaking down. Scalability is something you have to think about when you make and use systems that have to work with a lot of data or users.

References

1.Iyengar, Sheena S., and Emir Kamenica. "Choice Proliferation, Simplicity Seeking, and Asset Allocation." Journal of Public Economics 94, no. 7-8 (2010): 530-39.

2.Malhotra, Naresh K. "Information Load and Consumer Decision Making." Journal of Consumer Research 8, no. 4 (1982): 419-30.

3.Scheibehenne, Benjamin, Rainer Greifeneder, and Peter M. Todd. "Can There Ever Be Too Many Options? A Meta-Analytic Review of Choice Overload." Journal of Consumer Research 37, no. 3 (2010): 409-25.

4.Simonton, Dean Keith. "Film Audiences as Choosing Agents: The Cognitive, Motivational, and Social Contexts of Motion-Picture Appreciation." Poetics 26, no. 2-3 (1999): 189-211.

5.Lee, Hyejoon, et al. "Facing Too Many Choices: Variety Seeking in the Movie Theater Industry." Marketing Science 34, no. 6 (2015): 803-23.

6.Reutskaja, Elena, and Roberto A. Weber. "Decision Making With Limited Attention: The Effect of Competition on Purchase Quantities." Management Science 65, no. 6 (2019): 2731-48.

7.Livingstone, Sonia M., and Magdalena Bielau. "Qualitative Studies on the Impact of Television on Children, Adolescents and Families – Where Have We Been? Where Are We Going?." Journal of Children and Media 6, no. 2 (2012): 193-201.

8.Char, Rajiv. "23andMe Confirms Data Breach, Insists Genetic Data Wasn't Stolen." Digital Trends, January 25, 2019. https://www.digitaltrends.com/home/23andme-confirms-data-breach/
Coverage of the 2019 23andMe breach acknowledging exposure but claiming genetic data was unaffected.

9.Hern, Alex. "Netflix Faces Backlash over Plans to Release Viewing Data on Twitter." The Guardian, May 22, 2019. https://www.theguardian.com/technology/2019/may/22/netflix-backlash-viewing-data-twitter
Discussion of privacy concerns over Netflix plans to share viewing stats publicly.

10.Murphy, Heather. "Netflix Denies Massive Data Breach After Reports of Leaked Customer Records." Android Headlines, April 21, 2022. https://www.androidheadlines.com/2022/04/netflix-data-breach-leaked-customer-records.html

11- Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.
12- He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017). Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web (pp. 173-182).
13- Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. ACM Computing Surveys (CSUR), 52(1), 1-38.
14- Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4), 331-370.
15- Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In The adaptive web (pp. 325-341). Springer Berlin Heidelberg.
16- Jamali, M., & Ester, M. (2010). A matrix factorization technique with trust propagation for recommendation in social networks. In Proceedings of the fourth ACM conference on Recommender systems (pp. 135-142).
17- Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. Knowledge-based systems, 46, 109-132.
18- Wang, H., Wang, N., & Yeung, D. Y. (2015). Collaborative deep learning for recommender systems. In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1235-1244).
19- Zheng, L., Noroozi, V., & Yu, P. S. (2017). Joint deep modeling of users and items using reviews for recommendation. In Proceedings of the tenth ACM international conference on web search and data mining (pp. 425-434).
20- Liang, D., Krishnan, R. G., Hoffman, M. D., & Jebara, T. (2018). Variational autoencoders for collaborative filtering. In Proceedings of the 2018 world wide web conference (pp. 689-698).
21- Chen, C., Zhang, M., Liu, Y., & Ma, S. (2018). Neural attentional rating regression with review-level explanations. In Proceedings of the 2018 world wide web conference (pp. 1583-1592).

22- Sun Z., Yang J., Zhang W., Bozzon A.(2019) Recurrent knowledge graph embedding for effective recommendation.In: Ramakrishnan N.(eds) Proceedings of The Web Conference 2019.WebConf 2019.Lecture Notes in Computer Science vol 11449.Springer Cham.
23- Wu Y., DuBois C., Zheng AX., Ester M.(2016) Collaborative denoising auto-encoders for top-n recommender systems.In: Bonchi F.Gionis A.Tseng VS.Zhu F.(eds) Proceedings of the Ninth ACM International Conference on Web Search and Data Mining.WSDM '16.ACM New York NY USA pp 153–162.
24- Sedhain S.Menon AK.Sanner S.Xie L.(2015) Autorec: Autoencoders meet collaborative filtering.In: Bonet B.Koenig S.(eds) Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence.AAAI'15.AAAI Press pp 111–117.
25- Dziugaite GK.Roy DM.(2015) Neural network matrix factorization.In: Cortes C.Lawrence ND.Lee DD.Sugiyama M.Garnett R.(eds) Advances in Neural Information Processing Systems 28.NIPS'15.Curran Associates Inc.Red Hook NY USA pp 3612–3620.

26- Lee, J., & Lee, J. (2017). A comprehensive analysis on movie recommendation system employing collaborative filtering. International Journal of Applied Engineering Research, 12(24), 15529-15534.
27- Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender systems: introduction and challenges. In Recommender systems handbook (pp. 1-34). Springer, Boston, MA.

28- Sharma, S., & Singh, A. (2020). Movie Recommendation System Modeling Using Machine Learning. International Journal of Advanced Science and Technology, 29(9s), 1013-1020.
29- Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. Knowledge-based systems, 46, 109-132.

30- Kumar, S., & Singh, S. K. (2019). Comparative study of recommender system approaches and movie recommendation system. International Journal of Innovative Technology and Exploring Engineering, 8(6), 1418-1422.
31- Aggarwal, C. C. (2016). Recommender systems. In Recommender Systems (pp. 1-28). Springer International Publishing.

32- Gupta, R., & Jain, S. (2019). Movie recommendation system using machine learning. International Journal of Engineering and Advanced Technology, 8(6), 2249-2252.
33- Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.

34- Zhang, Y., Wang, X., & Wang, Y. (2020). A Hybrid Movie Recommendation System using Deep Learning and Collaborative Filtering. In Proceedings of the 2020 4th International Conference on Information System and Data Mining (pp. 1-5).
35- He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017). Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web (pp. 173-182).

36- Singh, P., & Kaur, H. (2018). Recommender System on MovieLens Dataset. International Journal of Computer Applications, 181(35), 1-5.

37- Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. ACM Transactions on Interactive Intelligent Systems (TiiS), 5(4), 1-19.

38- Zahra Zamanzadeh Dharban, Muhammad Hadi Valber **GHRS Graph-Based Hybrid Recommendation system with application to movie recommendation**,Website https:// arxiv. org/pdf/211.11293 v2.pdf,Received 25 mar 2022
39- Netflix prize data https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data
40- Movielens dataset https://medium.com/analytics-vidhya/movie-recommender-system-using-content-based-and-collaborative-filtering-84a98b9bd98e
41 Personalized Real-Time Movie Recommendation System: Practical Prototype and Evaluation
By: Jiang Zhang, Yufeng Wang*, Zhiyuan Yuan, and Qun Jin
Published in:Tsinghua Science and TechnologyVolume: 25,Issue: 2, April 2020)
Page(s):180- 191
Date of Publication:02 September 2019
Electronic ISSN:1007-0214
INSPEC Accession Number:18954023
DOI:10.26599/TST.2018.9010118
Publisher:TUP
Link : https://ieeexplore.ieee.org/abstract/document/8821512
42.Title: Design of an Unsupervised Machine Learning-Based Movie Recommender System
By Debby Cintia Gaunesha Putri , Jenq-shiou Leu and Pavel Seda
Symmetry2020,12(2), 185;https://doi.org/10.3390/sym12020185
Received: 25 December 2019/Revised: 11 January 2020/Accepted: 13 January 2020/Published: 21 January 2020
Published site: https://www.mdpi.com/2073-8994/12/2/185