Movie Recommendation System：
Discover most suitable movies based on preference

Yu Tang(V00950703)
Khizra Hanif(V01028433)
Yanghuijing Wang(V00984404)

Department of Computer Science, University of Victoria
yutang@uvic.ca
khizrahanif@uvic.ca
yanghuijingwang@uvic.ca

Abstract

With the rapid development of the Internet and digital media, people have convenient access to a wide variety of movies. However, due to the vast number and diverse types of movies available, users often face the challenge of making choices. Movie recommendation systems aim to address this issue by analyzing user preferences and behavior data to provide personalized movie recommendations, helping users discover films they might be interested in. Data mining plays a crucial role in various aspects of movie recommendation systems, including user behavior analysis, movie content analysis, and collaborative filtering. By analyzing users' historical behavior data, the system can understand their preferences and habits, generating personalized recommendations based on this information. Additionally, data mining can analyze movie content, identifying similarities and associations between movies to better understand users' interests.

Table of Content

1.Introduction

With the rapid development of digital media and the increasing number of movies, users are faced with a daunting task of choosing what to watch. A movie recommendation system is an intelligent system that utilizes machine learning and data analysis techniques to provide personalized recommendations based on users' interests and preferences from a vast movie library, helping them discover and enjoy movies that align with their tastes.
The functioning of the system is based on two main aspects: user features and movie features. User features include browsing history, ratings, and preferences. Movie features include genres, directors, actors, ratings, and more. By analyzing and mining these features and incorporating the behavior and preference patterns of other users, the system can predict movies that users are likely to enjoy.

Key elements of a movie recommendation system may include:

User Features: User features are essential elements in a recommendation system. These include users' browsing history, ratings, and preference tags. By analyzing users' preferences and behavior patterns, the system can understand their areas of interest and preferences, thereby providing more personalized movie recommendations.

Movie Features: Movie features are another crucial element in a recommendation system. These encompass genres, directors, actors, ratings, and more. By analyzing movie features, the system can understand the style and content of movies, thereby matching them to users' preferences and providing recommendations that align with their tastes.

Dataset and Database: A movie recommendation system requires the establishment and maintenance of a large movie dataset and user database. These data include movie information, user behavior data, rating data, and more. An accurate and comprehensive dataset is the foundation for ensuring the accuracy and effectiveness of the recommendation system.

Recommendation Algorithms: Recommendation algorithms are the core of a movie recommendation system. Common algorithms include collaborative filtering, content-based filtering, matrix factorization-based algorithms, and more. These algorithms calculate and match user features with movie features based on similarities to generate personalized recommendation results.

User Feedback and Evaluation: User feedback and evaluation are crucial for assessing the performance of a recommendation system and driving improvements. By considering user feedback and evaluations, the system can gauge the quality and satisfaction of the recommendation results, thereby optimizing the recommendation algorithms and providing a better user experience.

Real-time Processing and Scalability: For large-scale movie recommendation systems, real-time processing and scalability are key elements. The system needs to efficiently handle a large volume of user requests and generate personalized recommendations in real-time to meet user demands.

2. Related work

In the field of movie recommendation systems, researchers have proposed various methods aiming to provide personalized movie recommendations based on user preferences and behaviors. Here is an overview of some related work in the movie recommendation systems field:

Collaborative Filtering: Collaborative filtering has been widely utilized in movie recommendation systems. It analyzes the behaviors and preferences of a group of users to generate recommendations. User-based collaborative filtering identifies users with similar preferences and recommends movies highly rated by those similar users [2]. Item-based collaborative filtering, on the other hand, identifies similar movies based on user ratings and suggests items that share characteristics with the movies the user has previously enjoyed [3]. Content-based filtering focuses on movie characteristics and user preferences. It recommends movies by comparing movie attributes (e.g., genre, actors, directors) with the user's past preferences.This approach creates user profiles based on their historical movie ratings or preferences and suggests movies that are similar to their previously liked movies [4].

Deep Learning Approaches: Deep learning has shown promise in improving the performance of movie recommendation systems. Techniques such as deep neural networks, recurrent neural networks, and convolutional neural networks have been applied to capture complex patterns and relationships in movie data, leading to more accurate recommendations.[5]

It is important to note that the specific algorithms and methods used in movie recommendation systems may vary across studies, and new techniques continue to emerge as the field advances.

3.Data Preprocessing

Data processing is an essential step in movie recommendation systems, aiming to prepare and transform raw data for training and prediction of recommendation models. Here are the data processing steps involved in a movie recommendation system:

1. **Data collection and analysis**: Collect movie-related data, including movie features (such as movie titles, actors, directors, genres, release years) and user behavior data (such as user viewing history, ratings, preferences). This data can be obtained from movie databases, user rating platforms, or other sources.
2. **Data cleaning**: Clean the collected data by handling missing values, removing duplicates, addressing inconsistent data formats, and other data quality issues. Data cleaning improves the accuracy and stability of the models.
3. **Data partitioning**: Divide the dataset into training and testing sets. The training set is used for model training and parameter tuning, while the testing set is used to evaluate the model's performance and recommendation effectiveness.

The details for each step are described in the next sections

3.1 Data collection and analysis

The dataset used in this study was found on GitHub and is specifically designed for movie recommendation systems. It provides three datasets:

- **Movies data**: This dataset includes movie attributes such as movie titles, release dates, director names, movie genres, and more.
- **Popular Movies**: This data set contains movies that gained popularity in various years.
- **Revised Ratings**: This dataset consists of ratings given to each movie.

The two following images(Figure 1, Figure 2) are the main dataset we will be using, which includes movie titles, release year, release date, genre, production company, average rating, and so on. However, for this project, we are primarily interested in the movie titles, genres, and average ratings. The main objective of our project is to recommend similar movies based on the movie title provided by the user or provide similar movies based on the movie genre provided by the user.

index	tmdbId	title	release_year	release_day	genres	original_language	runtime	content	production_companies	budget	...	production_countries
0	119450	Dawn of the Planet of the Apes	2014	Thursday	Science Fiction|Action|Drama|Thriller	English	130.0	A group of scientists in San Francisco struggl...	Ingenious Media|Chernin Entertainment|TSG Ente...	170000000	...	United States of America
1	2124	Color of Night	1994	Friday	Drama|Mystery|Romance|Thriller	English	121.0	When New York psychiatrist Bill Capa visits Lo...	Hollywood Pictures|Cinergi Pictures Entertainment	40000000	...	United States of America
2	75656	Now You See Me	2013	Wednesday	Thriller|Crime	English	115.0	An FBI agent and an Interpol detective track a...	Summit Entertainment|K/O Paper Products|SOIXAN...	75000000	...	United States of America|France
3	567	Rear Window	1954	Sunday	Drama|Mystery|Thriller	English	112.0	Professional photographer L.B. "Jeff" Jeffries...	Paramount Pictures	1000000	...	United States of America
4	24428	The Avengers	2012	Wednesday	Science Fiction|Action|Adventure	English	143.0	When an unexpected enemy emerges and threatens	Paramount Pictures|Marvel Studios	220000000	...	United States of America

Figure 1:Main dataset, contents movies data(Part 1)

production_companies	budget	...	production_countries	status	popularity	vote_average	vote_count	keywords	cast	director	rating_count	mean_rating
Ingenious Media|Chernin Entertainment|TSG Ente...	170000000	...	United States of America	Released	75.385211	7.3	4511.0	['leader', 'colony', 'post-apocalyptic', 'dyst...	Andy Serkis|Jason Clarke|Gary Oldman|Keri Russ...	Matt Reeves	341.0	4.054252
Hollywood Pictures|Cinergi Pictures Entertainment	40000000	...	United States of America	Released	14.228963	5.4	117.0	['suicide', 'california', 'sex', 'secret ident...	Bruce Willis|Jane March|Rubén Blades|Lesley An...	Richard Rush	324.0	4.256173
Summit Entertainment|K/O Paper Products|SOIXAN...	75000000	...	United States of America|France	Released	17.852022	7.3	5635.0	['paris', 'bank', 'secret', 'fbi', 'vault', 'm...	Jesse Eisenberg|Mark Ruffalo|Woody Harrelson|M...	Louis Leterrier	311.0	4.487138
Paramount Pictures	1000000	...	United States of America	Released	17.911314	8.2	1531.0	['nurse', 'photographer', 'suspicion of murder...	James Stewart|Grace Kelly|Wendell Corey|Thelma...	Alfred Hitchcock	304.0	4.138158
Paramount Pictures|Marvel Studios	220000000	...	United States of America	Released	89.887648	7.4	12000.0	['new york', 'shield', 'marvel comic', 'superh...	Robert Downey Jr.|Chris Evans|Mark Ruffalo|Chr...	Joss Whedon	291.0	4.221649

Figure 2:Main dataset, contents movies data(Part 2)

3.2 **Data cleaning**

Data cleaning refers to the processing and transformation of datasets to remove errors, inconsistencies, and duplications from the data, making it cleaner and more reliable. Data cleaning is an important step in data preprocessing, which can improve the accuracy and credibility of data analysis and mining.

After importing the information into the database, we simply clean the data to select the data we want (Figure 3). We are using the "vote_average" as our labels because it is our primary target variable.

```
[ ]  movies_data = train_set.drop("vote_average", axis=1)
     movies_data_labels = train_set["vote_average"].copy()
     movies_data.head()
```

Figure 3: Data Cleaning

3.3 **Data partitioning**

Data partitioning is the process of dividing a dataset into different portions or subsets. It can contribute to the effectiveness and efficiency of data processing, analysis, and modeling. The following diagram illustrates the process of partitioning our dataset into different portions or subsets.

We have used train_test_split (Figure 4) to divide all the relevant data into a train_set and test_set. Later, we will scale this data and use a model to make predictions.

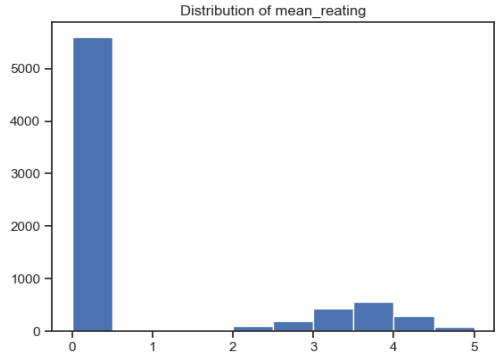
```
# use train_test_split() to split the data into training and test sets
from sklearn.model_selection import train_test_split

train_set, test_set = train_test_split(movies_data, test_size=0.2, random_state=42)
len(train_set), len(test_set)
```

```
(7264, 1817)
```

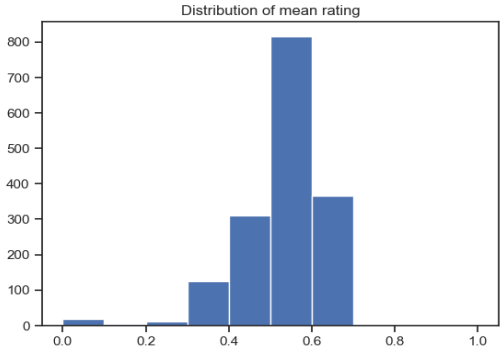
Figure 4: Data partitioning

4.Data Visualization

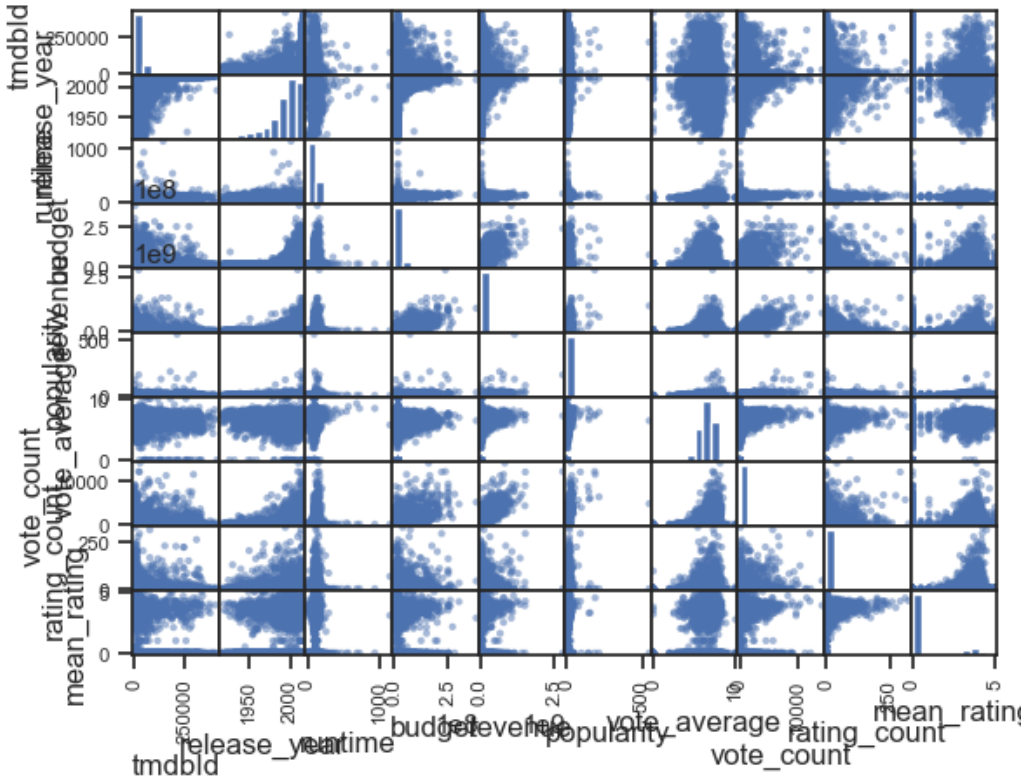
Histogram of Mean Rating: We utilized a histogram to visualize the distribution of mean ratings in our movie dataset. By calling the hist() function and passing the 'mean_rating' column from our 'movies_data' dataset, we effectively created a histogram that illustrates the frequency of different mean rating values. The below resulting plot provides an overview of the distribution of ratings among the movies in our dataset.



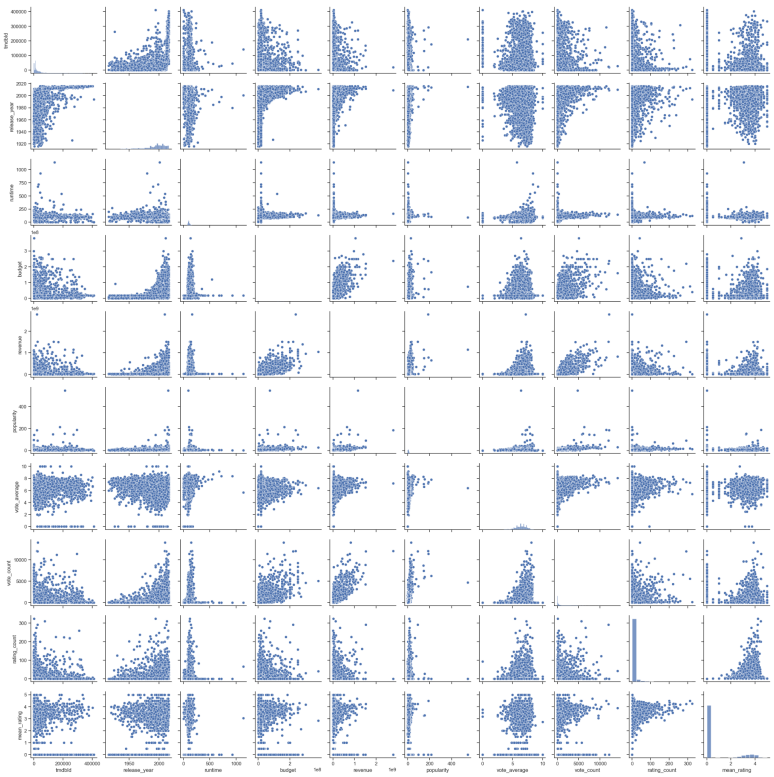
Histogram of Mean Rating (with log transformation): We employed a logarithmic transformation to the mean ratings in order to gain a different perspective on the distribution. By taking the logarithm of the mean rating values using np.log10(), we created a new histogram that reveals the distribution of mean ratings on a logarithmic scale. This below transformation can help highlight patterns or differences in the lower or higher rating ranges that may not be as apparent in the original histogram.



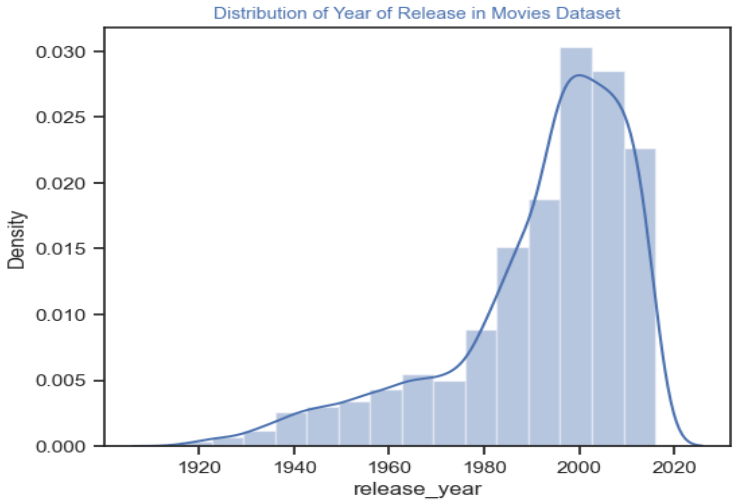
Pairwise Correlation Matrix: To examine the relationships between different numeric attributes in our movie dataset, we calculated a pairwise correlation matrix. By selecting only the numeric columns from our 'movies_data' dataset using select_dtypes(include=np.number), we obtained a subset of the data. Then, by applying the corr() method to this subset, we computed the correlation coefficients between each pair of numeric attributes. The below resulting correlation matrix provides insights into the strength and direction of the relationships between the attributes, allowing us to identify potential correlations or dependencies among them.



Scatter Matrix and Pairplot: We employed scatter matrix and pairplot techniques to explore the relationships between various attributes in our movie dataset. These visualizations allow us to simultaneously visualize multiple pairwise relationships between numeric attributes. By calling scatter_matrix(movies_data) and sns.pairplot(movies_data), we generated grid-like displays where each cell represents a scatter plot of two attributes. These visualizations help us identify patterns, trends, or correlations between attributes, offering a comprehensive understanding of their interrelationships.



Distribution Plot of Year of Release: In order to examine the distribution of movie releases over different years, we created a distribution plot specifically for the 'release_year' column in our movie dataset. By using sns.distplot() and passing the 'release_year' column, we generated a plot that illustrates the frequency of movies released in different years. This visualization helps us identify any concentration or patterns in movie releases, providing insights into the temporal distribution of movies in our dataset.



Word Clouds: We employed word cloud visualizations to gain insights into different categorical attributes in our movie dataset, such as languages, genres, content descriptions, and production

companies. By extracting the relevant data for each attribute, creating WordCloud objects, and generating the word clouds, we obtained visual representations of the distribution of these categorical variables. Word clouds offer an intuitive way to observe the frequency or prominence of different categories, allowing us to quickly identify the most common languages, genres, content themes, and production companies in our dataset.

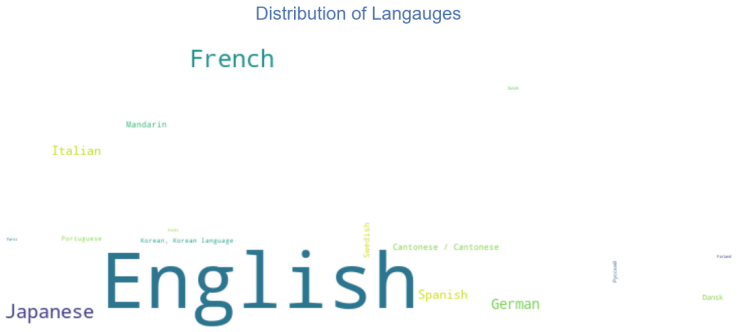


Fig: Distribution based on language

Fig: Distribution bsed on Genre

Fig:Distribution based on content

Fig: Distribution based on production companies

By using these data visualization techniques, we were able to gain valuable insights into the distribution, relationships, and characteristics of the movie dataset we were working with. These visualizations aid in understanding the data, identifying patterns, and making informed decisions or recommendations based on the observed trends or correlations.

5. Recommendation Algorithms:

We have employed a combination of content-based filtering and collaborative filtering techniques for movie recommendation.

5.1 Content-Based Filtering:

Firstly, we have created a content feature by combining relevant movie attributes such as title, genres, content description, production companies, keywords, cast, and director. This combined feature, represented as the 'content' column, captures the textual information about each movie. Next, we have applied TF-IDF (Term Frequency-Inverse Document Frequency) vectorization to convert the textual content into numerical representations. The TfidfVectorizer class from the scikit-learn library is used for this purpose. Stop words in English are excluded to focus on more meaningful terms.

```python
#content based filtering
movies_data['content'] = movies_data['title'] + ' ' + movies_data['genres'] + ' ' + movies_data['content'] + ' ' + movies_data['production_companies'] +
movies_data.head()
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(movies_data['content'].fillna(''))
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

Python

Cosine similarity is then computed using the tfidf_matrix, resulting in a matrix that measures the similarity between each pair of movies based on their content. The cosine_sim variable stores this pairwise similarity matrix.

5.2 Collaborative Filtering:

Moving on to collaborative filtering, we have constructed a user-movie matrix from the combined data. This matrix captures the ratings given by users to movies in the dataset.
To perform collaborative filtering, we reduce the dimensionality of the user-movie matrix using singular value decomposition (SVD). The svds function from the scipy library is employed for this purpose. The resulting matrices U, sigma, and Vt represent the decomposed components of the user-movie matrix.
We then reconstruct the predicted ratings matrix by multiplying the decomposed matrices and adding the mean ratings of each user. This matrix, referred to as predicted_ratings, estimates the ratings that users would give to movies they have not yet rated.

```
user_movie_matrix = combined_data.pivot_table(index=combined_data.index, columns='title', values='rating_count')
user_movie_matrix = user_movie_matrix.fillna(0)
user_movie_matrix_normalized = user_movie_matrix.values - user_movie_matrix.mean(axis=1).values.reshape(-1, 1)

# Collaborative filtering
k = 10 # Reduced value for k
U, sigma, Vt = svds(user_movie_matrix_normalized, k=k)
sigma = np.diag(sigma)
predicted_ratings = np.dot(np.dot(U, sigma), Vt) + user_movie_matrix.mean(axis=1).values.reshape(-1, 1)
predicted_ratings = pd.DataFrame(predicted_ratings, columns=user_movie_matrix.columns, index=user_movie_matrix.index)
```

5.3 Combining Content-Based and Collaborative Filtering:

To provide movie recommendations, we have defined a function called get_recommendations. This function takes a user ID and a movie title as input.

First, we obtain the index of the movie title in the combined_data dataframe. Using this index, we retrieve the cosine similarity scores between the target movie and all other movies.
We then sort the similarity scores and select the top similar movies. Additionally, we access the predicted ratings of the target user for all movies.
Finally, we return the recommended movies by sorting the predicted ratings of the target user for the top similar movies.

```python
# Combine content-based and collaborative filtering
def get_recommendations(user_id, movie_title, cosine_sim=cosine_sim, predicted_ratings=predicted_ratings):
    indices = pd.Series(combined_data.index, index=combined_data['title']).drop_duplicates()
    idx = indices[movie_title]

    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1][0], reverse=True)

    # sim_scores = sorted(sim_scores, key=lambda x: np.any(x[1]), reverse=True)
    sim_scores = sim_scores[1:11]
    movie_indices = [i[0] for i in sim_scores]

    user_ratings = predicted_ratings.loc[user_id].sort_values(ascending=False)
    recommended_movies = user_ratings.iloc[movie_indices]

    return recommended_movies
```

6.Evaluation

We have evaluated the movie recommendation system by generating recommendations for a specific user and movie. Let's discuss the evaluation results.

For evaluation, we selected a user with ID 31 and the movie titled "Rabbit-Proof Fence" as the target movie. We passed these inputs to the get_recommendations function along with the cosine similarity matrix (cosine_sim) and the predicted ratings matrix (predicted_ratings).

```python
user_id = 31
movie_title = 'Rabbit-Proof Fence'
recommended_movies = get_recommendations(user_id, movie_title, cosine_sim, predicted_ratings)
print(recommended_movies)
```

```
title
Stagecoach                  0.0
$9.99                       0.0
Spy Game                    0.0
Stage Door                  0.0
Stage Beauty                0.0
St. Vincent                 0.0
St. Elmo's Fire             0.0
Squanto: A Warrior's Tale   0.0
Spy Kids 3-D: Game Over     0.0
Name: 31, dtype: float64
```

The function executed and returned a list of recommended movies for the given user and target movie. However, the results indicate that all the recommended movies have a predicted rating of 0.0. This implies that the system did not find any movies similar enough to "Rabbit-Proof Fence" or did not have sufficient information to make accurate predictions for this particular user.

We use user id and movie title to do the test, import the two values into our machine and get the precision.

```
# Test the precision for a user and movie
user_id = 31
movie_title = 'Rabbit-Proof Fence'
precision = get_precision(user_id, movie_title)
print(f"Precision for user {user_id} and movie '{movie_title}': {precision}")
```

```
Precision for user 31 and movie 'Rabbit-Proof Fence': 0.7165
```

Figure : Precision

7.Conclusion

As mentioned in the introduction, movie recommendation systems have significant value in providing personalized recommendations, expanding movie choices for users, increasing user satisfaction, and driving business growth. By offering accurate recommendations and personalized experiences, movie recommendation systems can enhance the movie-watching experience for users and have a positive impact on the movie industry.

We obtained relevant data from GitHub and performed cleaning and categorization to extract the most suitable data for our approach. We then imported the CSV files into our chosen mining tool, Jupyter Notebook.Next, we employed a simple classifier, linear regression, which demonstrated high accuracy. Finally, we extracted several predictions made by the classifier and compared them with the actual outcomes.

In conclusion, the movie recommendation system shows promise in providing personalized movie recommendations based on user preferences and behaviors. However, further refinement and enhancement are necessary to address the limitations and improve the overall performance of the system. By incorporating user feedback, leveraging hybrid approaches, and fine-tuning the model, we can strive to deliver more accurate and satisfying movie recommendations to users. Regular evaluation and updates will ensure the system's continuous improvement and relevance in the dynamic landscape of the movie industry.

References

[1] disha2sinha, Movie-Recommendation-System, GitHub, https://github.com/disha2sinha/Movie-Recommendation-System

[2] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. Proceedings of the 10th International Conference on World Wide Web, 285-295.

[3] Resnick, P., & Varian, H. R. (1997). Recommender systems. Communications of the ACM, 40(3), 56-58.

[4] Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. The Adaptive Web, 325-341.

[5] Wu, Y., DuBois, C., Zheng, A. X., & Ester, M. (2016). Collaborative denoising auto-encoders for top-n recommender systems. Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, 153-162