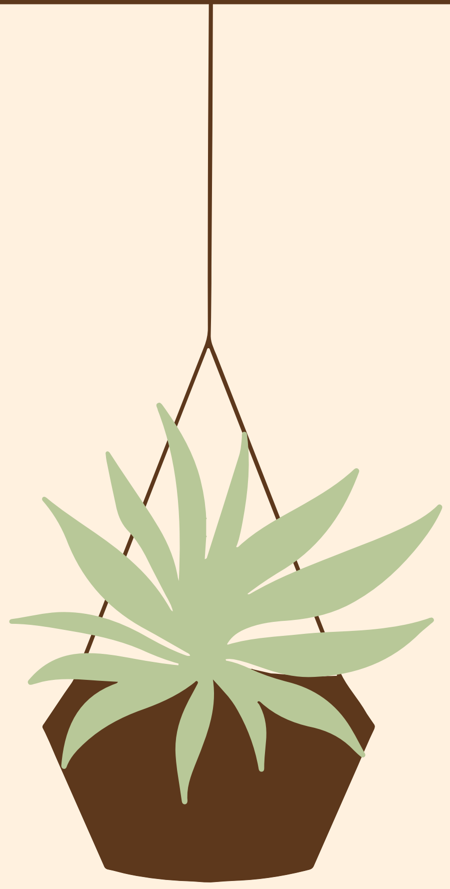
GROUP PROJECT

Yu Tang
Khizra Hanif
Yanghuijing Wang

PROJECT OVERVIEW

Movie recommendation systems leverage data mining to analyze user preferences and behavior, offering personalized suggestions from a vast and diverse collection of films. By examining user data, such systems understand preferences and generate tailored recommendations. Data mining also helps in analyzing movie content and identifying similarities to better comprehend user interests.



INTRODUCTION

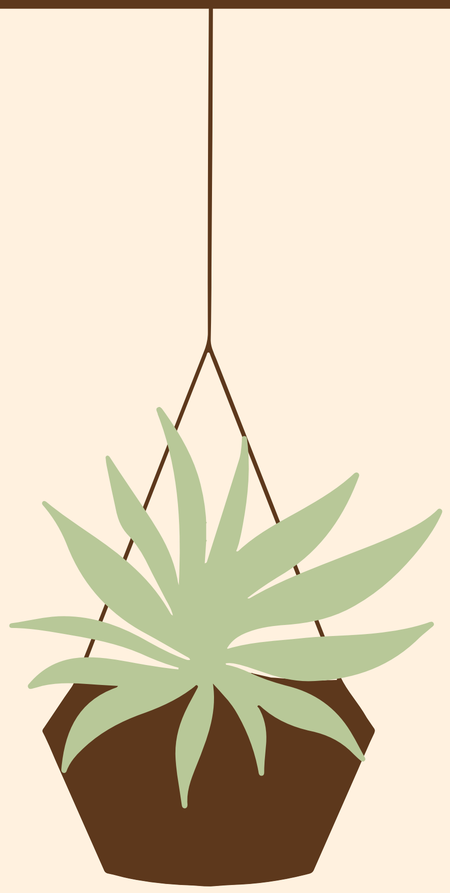
Movie recommendation systems leverage machine learning and data analysis to provide personalized movie suggestions based on user preferences and movie features. By analyzing user behavior, such as browsing history and ratings, along with movie characteristics like genres and directors, the system predicts movies that align with users' interests. These systems rely on a comprehensive dataset, recommendation algorithms, user feedback, and real-time processing to deliver accurate and scalable recommendations, enhancing the movie-watching experience for users.

RELATED WORK

Researchers have proposed various methods in movie recommendation systems to provide personalized recommendations based on user preferences and behaviors. These methods include collaborative filtering (user-based and item-based), content-based filtering, and deep learning approaches. Collaborative filtering analyzes user behaviors and similarities, while content-based filtering focuses on movie attributes and user preferences. Deep learning techniques, such as deep neural networks, recurrent neural networks, and convolutional neural networks, capture complex patterns for improved recommendation accuracy. It's important to note that algorithms and methods vary across studies as the field continues to evolve.

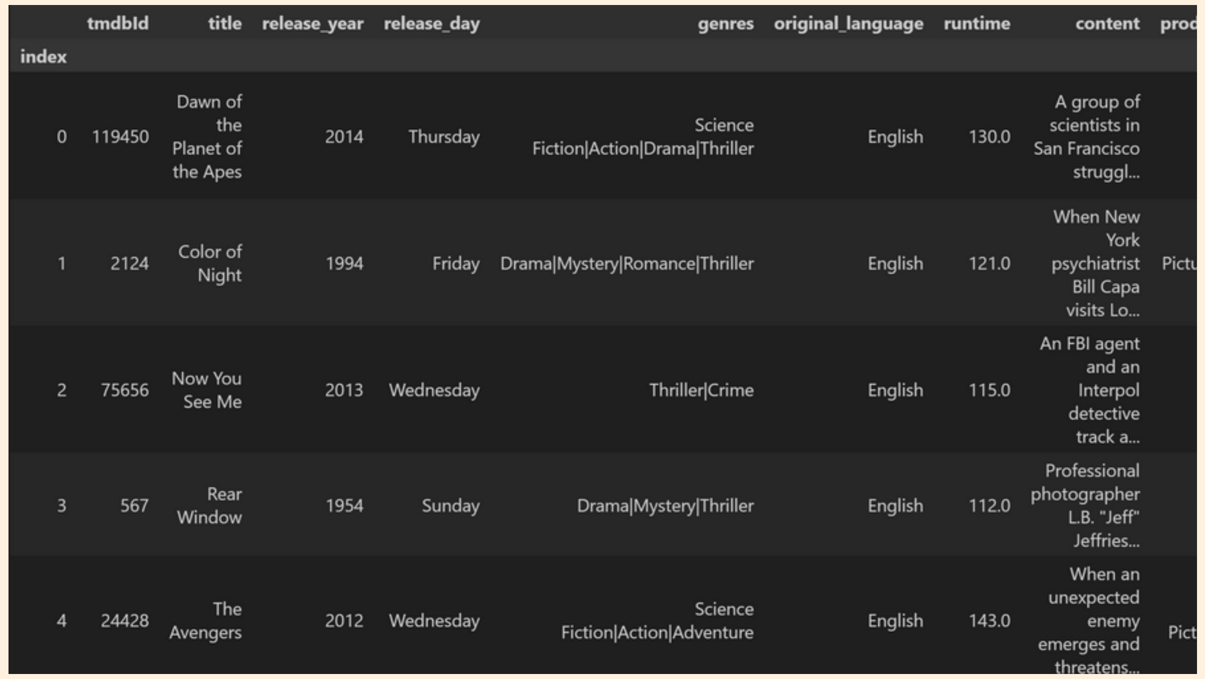
Data Preprocessing

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.



DATA COLLECTION AND ANALYSIS

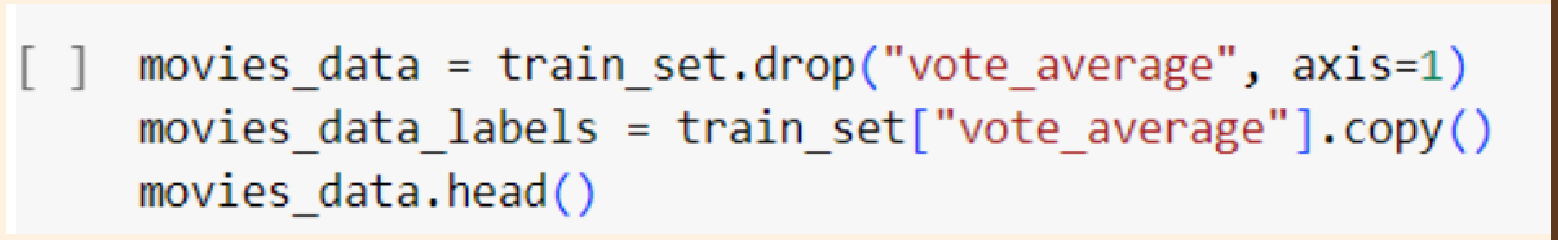
The dataset used in this study is specifically designed for movie recommendation systems and includes three datasets: Movies data, Popular Movies, and Revised Ratings. The main dataset we will be using contains movie titles, genres, and average ratings. Our project aims to recommend similar movies based on user-provided movie titles or genres.



index	tmdbId	title	release_year	release_day	genres	original_language	runtime	content	prod
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...	
1	2124	Color of Night	1994	Friday	Drama|Mystery|Romance|Thriller	English	121.0	When New York psychiatrist Bill Capa visits Lo...	Pictu
2	75656	Now You See Me	2013	Wednesday	Thriller|Crime	English	115.0	An FBI agent and an Interpol detective track a...	
3	567	Rear Window	1954	Sunday	Drama|Mystery|Thriller	English	112.0	Professional photographer L.B. "Jeff" Jeffries...	
4	24428	The Avengers	2012	Wednesday	Science Fiction|Action|Adventure	English	143.0	When an unexpected enemy emerges and threatens...	Pict

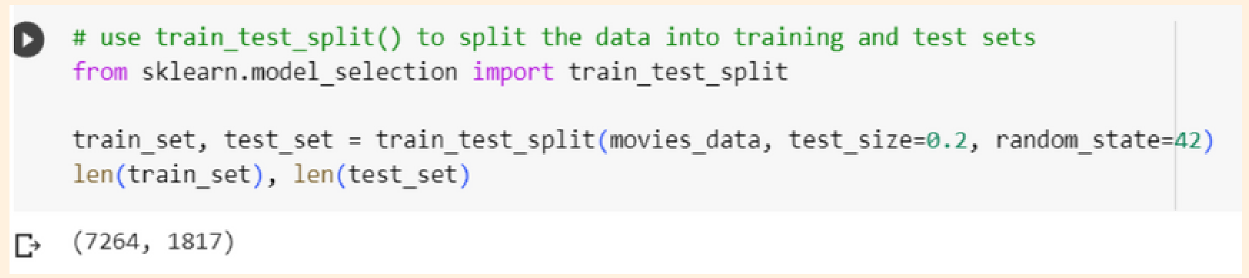
DATA CLEANING

Data cleaning involves processing and transforming datasets to eliminate errors, inconsistencies, and duplications, improving data reliability. It is a crucial step in data preprocessing, enhancing the accuracy and credibility of data analysis. In our project, we clean the imported data and select the desired information, focusing on the "vote_average" as our primary target variable.



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

DATA PARTITIONING

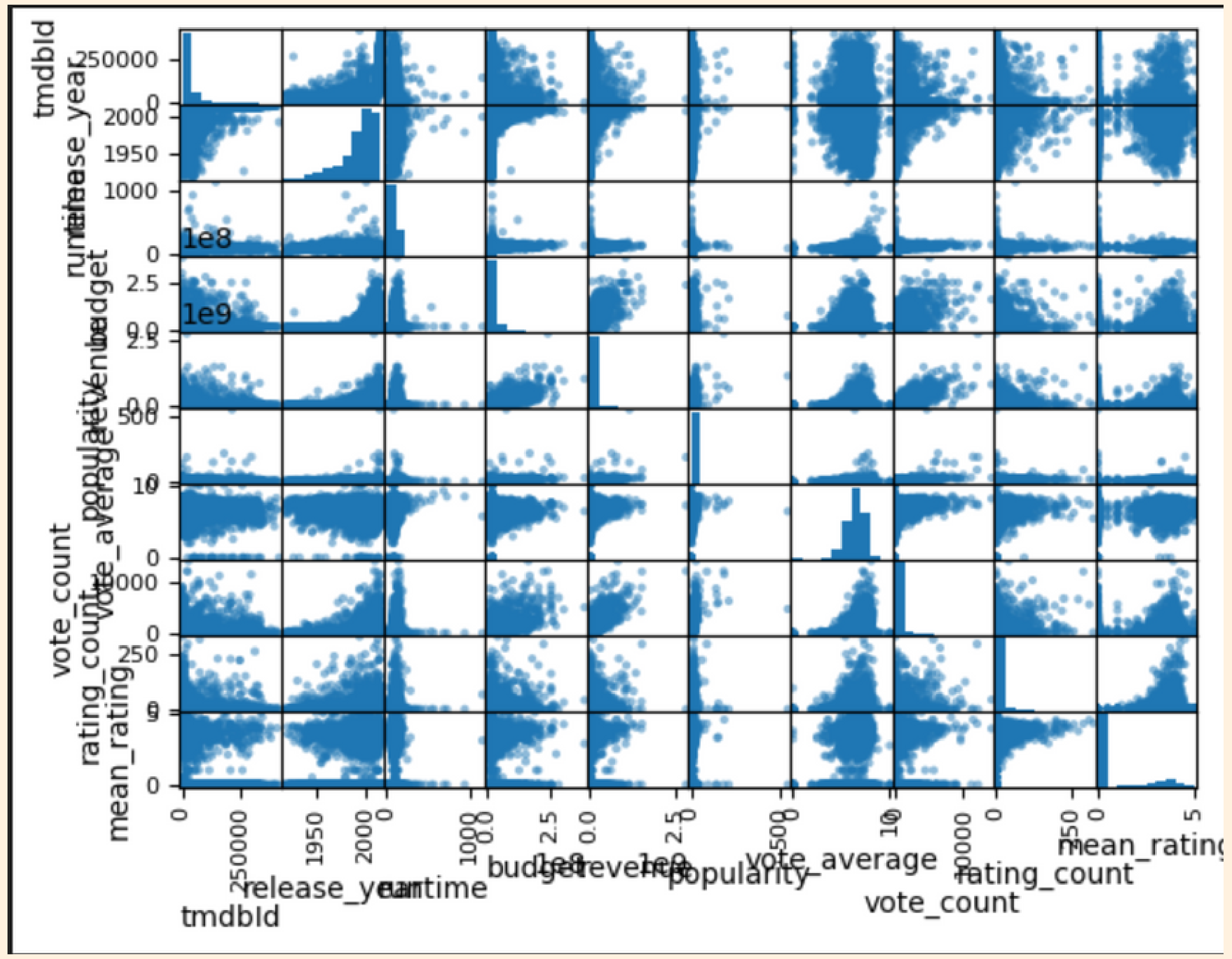


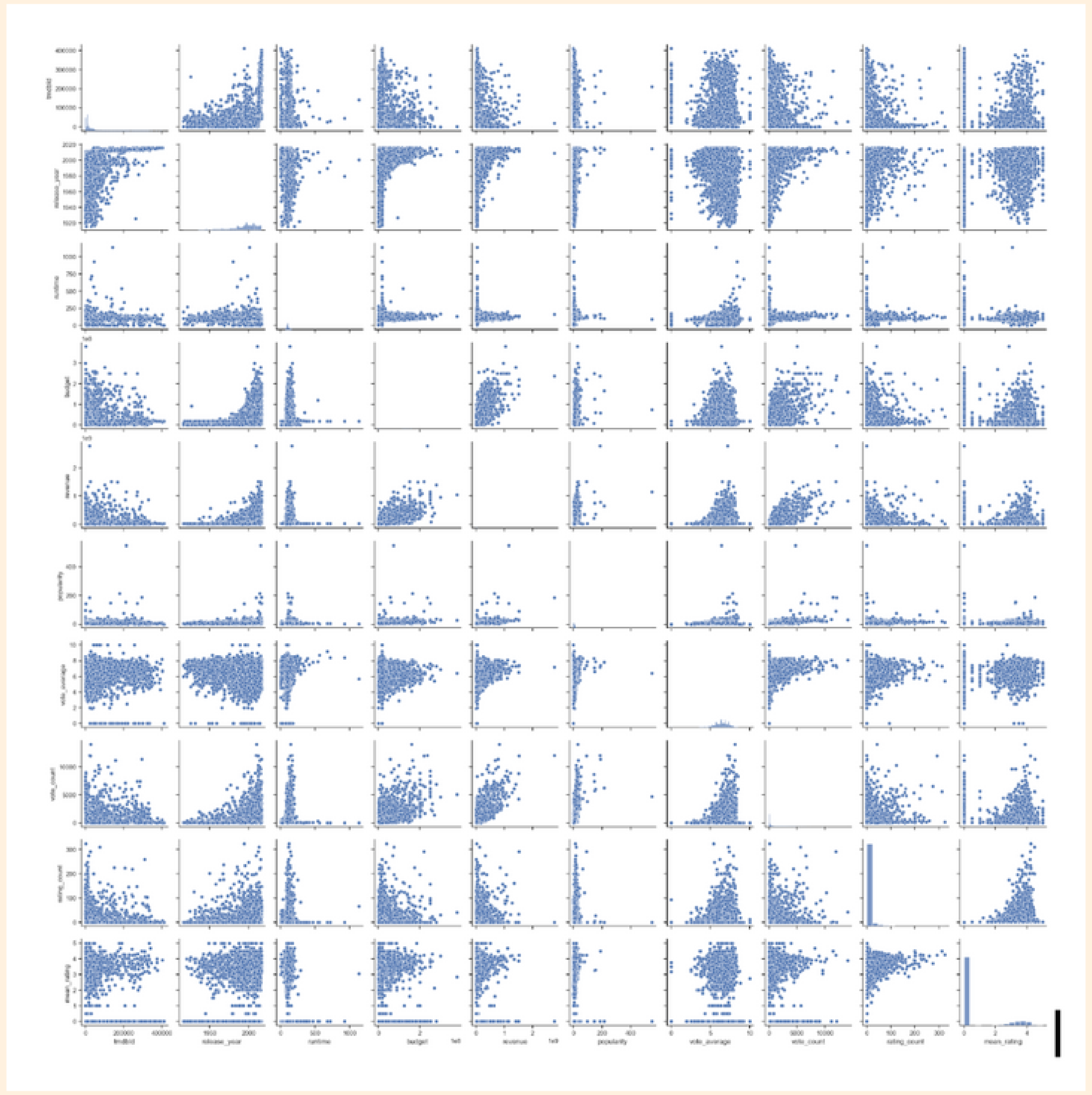
```
# use train_test_split() to split the data into training and test sets
from sklearn.model_selection import train_test_split

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

```
(7264, 1817)
```

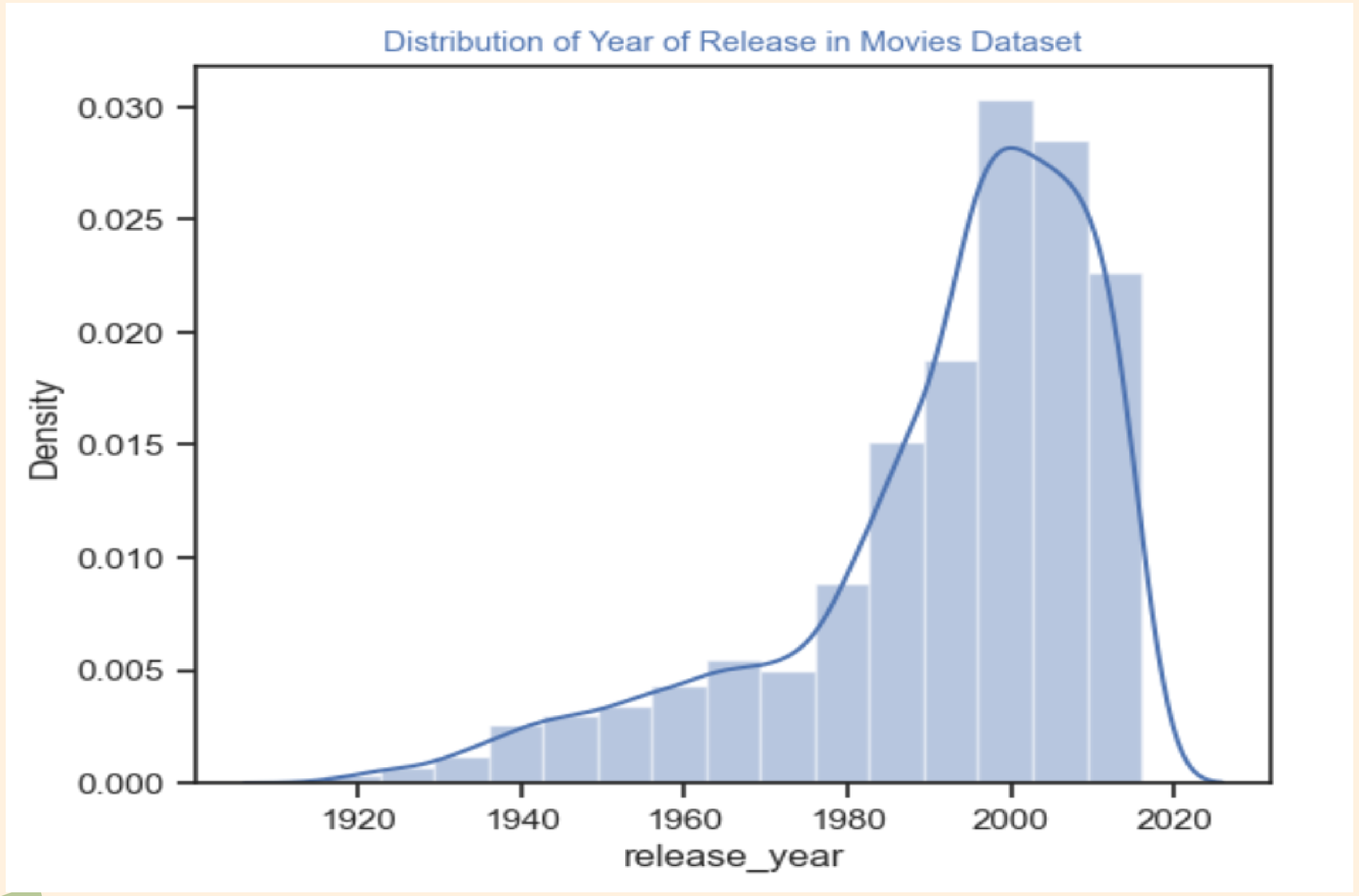
Data partitioning is the process of dividing a dataset into different portions or subsets. It can contribute to the effectiveness and efficiency of data processing, analysis, and modeling. The following diagram illustrates the process of partitioning our dataset into different portions or subsets.

DATA VISUALIZATION





DATA VISUALIZATION



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.

DATA VISUALIZATION

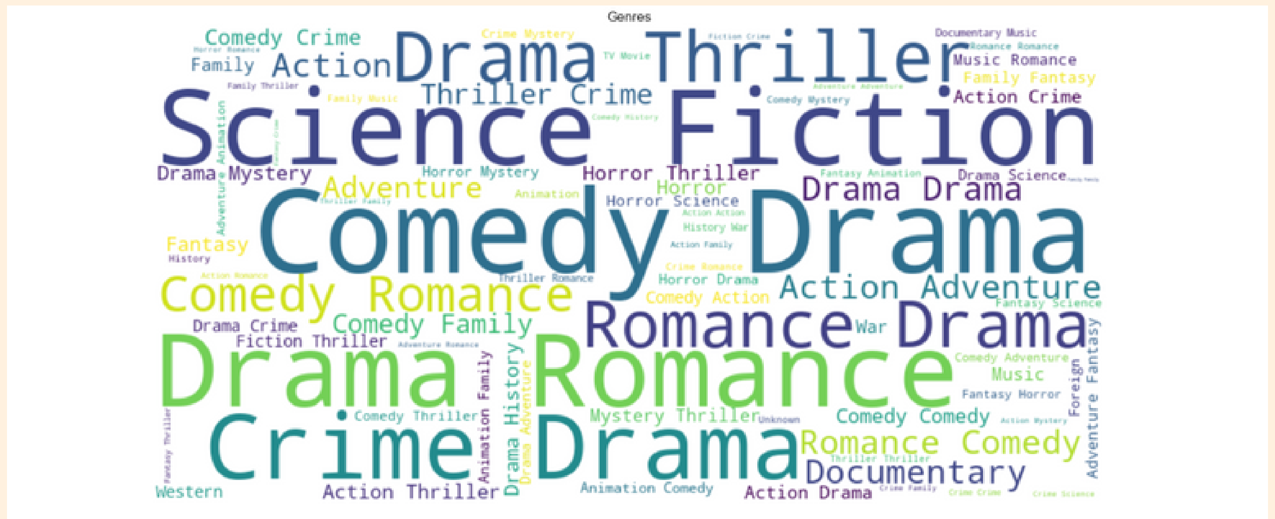
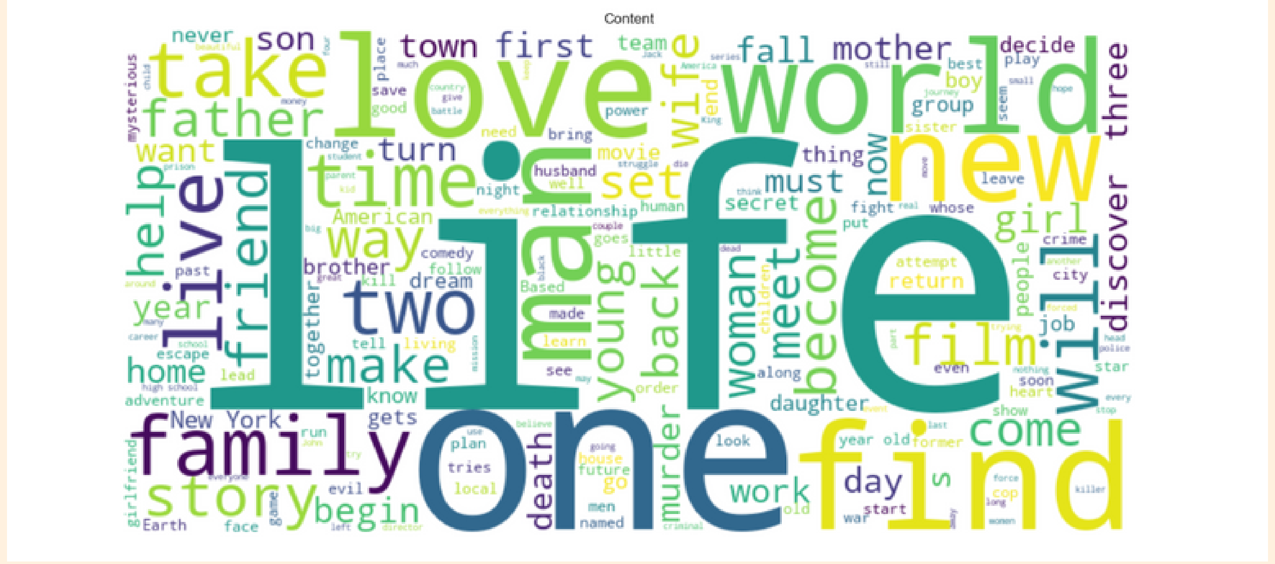


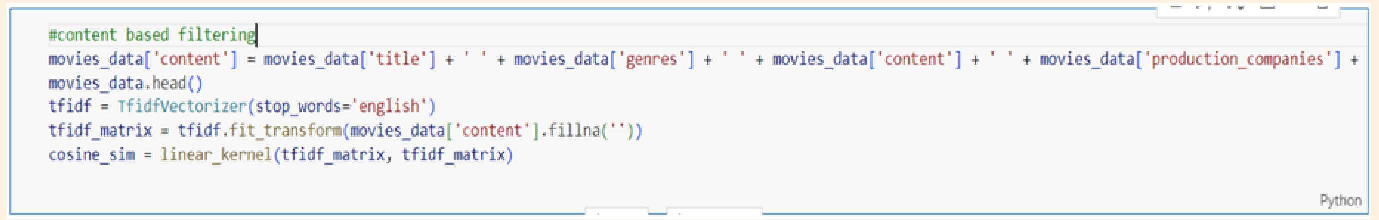
Fig: Distribution bsed on Genre



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.

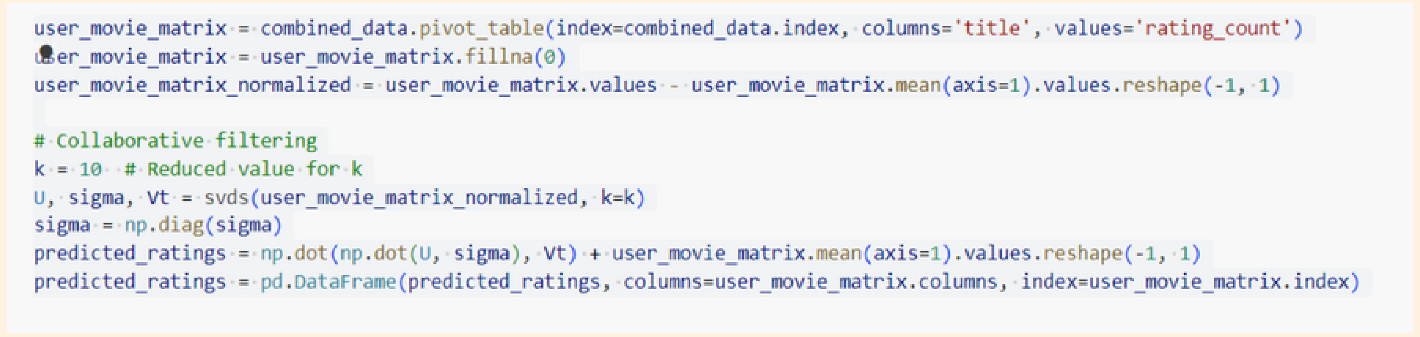
RECOMMENDATION ALGORITHMS

Our movie recommendation system combines content-based filtering and collaborative filtering techniques. It analyzes movie attributes and user ratings to provide personalized recommendations based on movie content and user behavior.

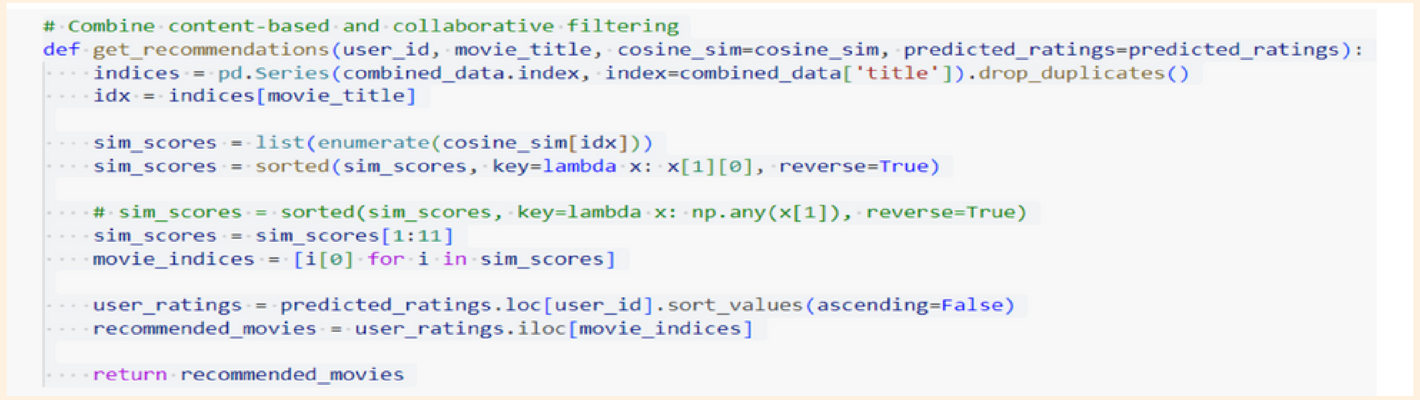


```python
#content based filtering
movies_data['content'] = movies_data['title'] + ' ' + movies_data['genres'] + ' ' + movies_data['content'] + ' ' + movies_data['production_companies'] +
movies_data.head()
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(movies_data['content'].fillna(''))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

Python



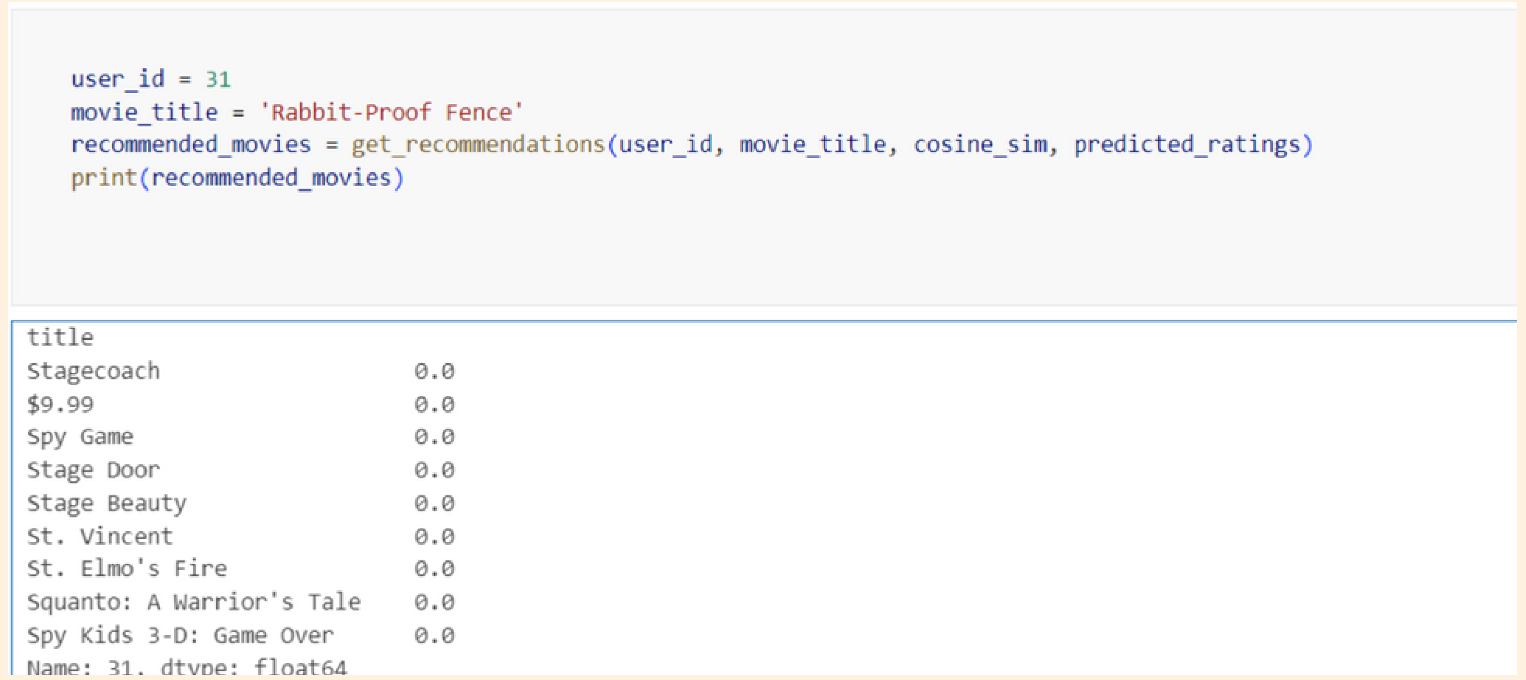
```python
user_movie_matrix = combined_data.pivot_table(index=combined_data.index, columns='title', values='rating_count')
user_movie_matrix = user_movie_matrix.fillna(0)
user_movie_matrix_normalized = user_movie_matrix.values - user_movie_matrix.mean(axis=1).values.reshape(-1, 1)

# 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)
```



```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
```

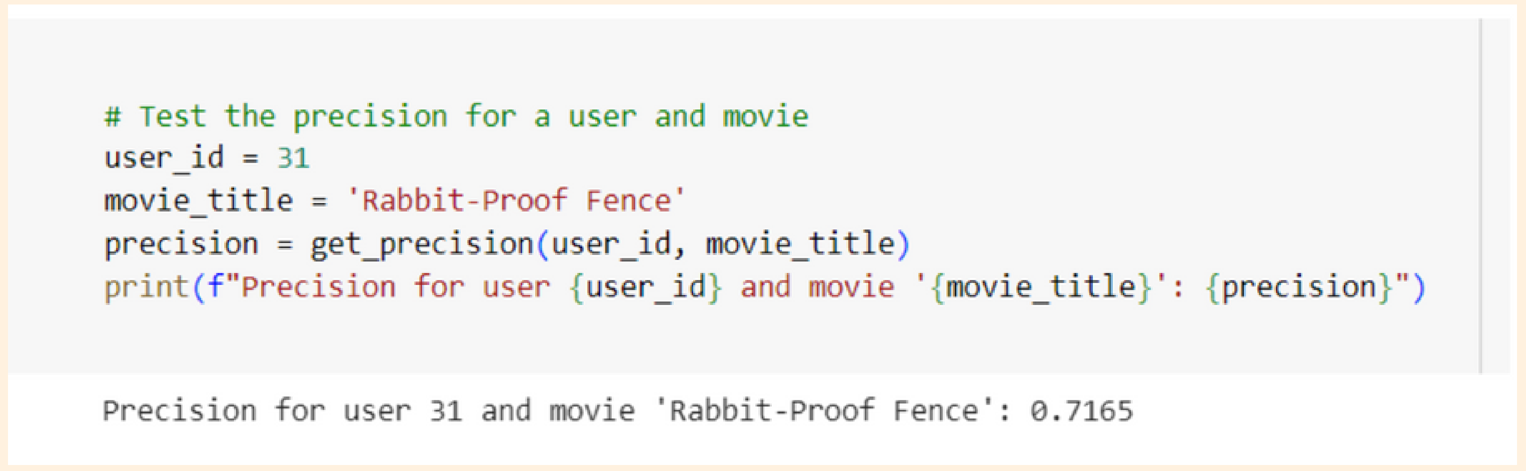
EVALUATION

The evaluation of our movie recommendation system involved selecting a user with ID 31 and the movie "Rabbit-Proof Fence" as the target. We used the get_recommendations function with the cosine similarity matrix and predicted ratings matrix. However, the results showed that all recommended movies had a predicted rating of 0.0, indicating a lack of similar movies or insufficient information for accurate predictions for this user. We further tested the system's precision using user ID and movie title



```
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
```



```
# 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
```

CONCLUSION

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

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

Hey there, who's got a burning question to ask?

A WARM THANK YOU TO ALL OF YOU!