MOVIE RECOMMENDATION SYSTEM THROUGH ADVANCED FILTERING TECHNIQUES

A Project Report Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Haneesh Malepati (2010030323)

Pydikondala Lahari (2010030372)

Mounika Kolli (2010030384)

Kamani Nandini (2010030385)

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING
K L DEEMED TO BE UNIVERSITY
AZIZNAGAR, MOINABAD , HYDERABAD-500 075

MARCH 2023

BONAFIDE CERTIFICATE

This is to certify that the project titled **Movie Recommendation System Through Advanced Filtering Techniques** is a bonafide record of the work done by

Haneesh Malepati (2010030323)

Pydikondala Lahari (2010030372)

Mounika Kolli (2010030384)

Kamani Nandini (2010030385)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING** of the **K L DEEMED TO BE UNIVERSITY, AZIZNAGAR, MOINABAD , HYDERABAD-500 075**, during the year 2023-2024.

Dr. SHADHAB SIDDIQUI
Project Guide

Dr. ARPITA GUPTA
Head of the Department

Project Viva-voce held on __________________________

Internal Examiner

External Examiner

ABSTRACT

Movie recommendation systems play a crucial role in helping users discover movies that align with their preferences, amid the ever-expanding pool of available content. This paper presents an in-depth exploration of movie recommendation systems, focusing on the utilization of advanced filtering techniques and the evaluation of their accuracy. These systems sift through an array of data attributes, including crew, descriptions, popularity, genres, and more, to provide personalized movie suggestions to users.

In this research, we delve into the mechanisms of collaborative filtering and hybrid filtering, two prominent approaches for improving recommendation accuracy. By evaluating the effectiveness of these methods, we aim to determine the most precise and reliable means of movie recommendation. As the volume of online data continues to surge, the development of robust movie recommendation systems becomes increasingly significant, offering users an enhanced and personalized viewing experience.

ACKNOWLEDGEMENT

We would like to thank the following people for their support and guidance without whom the completion of this project in fruition would not be possible.

Dr. SHADAB SIDDIQUI, our project guide, for helping us and guiding us in the course of this project .

Dr. ARPITA GUPTA, the Head of the Department, Department of DEPARTMENT NAME.

Our internal reviewers, **Dr. Pavan Kumar** , **Mr. Anantha Reddy Dasari** , **Mr. N Chiranjeevi** , **Mrs. B Uma Rani** for their insight and advice provided during the review sessions.

We would also like to thank our individual parents and friends for their constant support.

TABLE OF CONTENTS

List of Figures

Chapter 1

Introduction

1.1 Background of the Project

The project stems from the increasing need for effective movie recommendation systems in the digital age, where an ever-expanding array of movies is available across various online platforms. As users are presented with a multitude of choices, it becomes imperative to offer them tailored movie suggestions to enhance their viewing experience. Movie recommendation systems have evolved to address this challenge, leveraging diverse data attributes to provide personalized recommendations.

The foundation for this project lies in the recognition of the significance of advanced filtering techniques to improve recommendation accuracy. Traditional recommendation algorithms often have limitations, and to address these, we explore collaborative filtering and hybrid filtering methods, known for their ability to deliver more precise and personalized movie recommendations.

In a world inundated with data, these advanced techniques are essential to filter through the noise and offer users movies that align with their preferences, not only based on simple genre or popularity metrics but also considering factors like crew, descriptions, and other relevant attributes.

1.2 Problem Statement

The main goal of this project is to develop a movie recommender system using various nlp models to personalize suggestions based on user preferences by analyzing movie content for accurate matching and generating relevant recommendations.

1.3 Objectives

Design and Implement Personalized Recommendation System: Develop a personalized movie recommendation system capable of efficiently delivering movie suggestions that align with individual user preferences.

Utilize Advanced Filtering Techniques: Employ advanced filtering methods to address the difficulty of discovering relevant movies within the vast and ever-expanding content library, ensuring that users receive movie recommendations that resonate with their tastes.

Enhance Accuracy and Performance: Improve the precision and performance of conventional filtering techniques within movie recommendation systems, thereby contributing to higher user satisfaction and increased engagement with movie content.

1.4 Scope of the Project

The scope of this research project is centered on the creation of a personalized movie recommendation system, with a focus on algorithm development to efficiently cater to individual user preferences. Additionally, the integration of this system into user interfaces and existing platforms is a critical element for seamless user interaction. The project delves into advanced filtering techniques, including natural language processing, to increase recommendation accuracy. It also considers the adaptability of the system to evolving movie content and shifting user preferences, aiming to maintain relevance over time. User engagement and satisfaction metrics are evaluated, and the research findings will be thoroughly documented. Collaboration with movie industry stakeholders may be pursued to ensure the practical applicability and significance of the project's outcomes.

Chapter 2

Literature Review

2.1 Literature Survey

S.No	Authors	Title	Model Used	Challenges	Publishing Year
1.	Sushmita Roy, Mahendra Sharma, Santosh Kumar Singh	Movie Recommendation System Using Semi-Supervised Learning	Semi-Supervised Learning (clustering)	• Wanted to deploy over the cloud and it was challenging for them. • Presents more information.	October 2019
2.	Krishnaveni KS, Nimish Kapoor, Saurav Vishal	Movie Recommendation System Using NLP Tools	SVM, KNN	• ML models • For the scalability challenges they took small dataset. - high dimensions. - more training time.	2020
3.	Jain KN, Kumar V, Kumar P, Choudhury T	Movie Recommendation System: Hybrid Information Filtering System	Hybrid Filtering System(Content-based Filtering) , K-means	• The challenge was that the system wasn't working well according to the user ratings.(Description based not Popularity based)	2018

Figure 2.1: Articles

S.No	Authors	Title	Model Used	Challenges	Publishing Year
4.	V. Subramaniyaswamy, R. Logesh	A personalized movie recommendation system based on collaborative filtering.	collaborative filtering(grouping similar people)	• The challenge was to compute the movie rating accurately.	March 24, 2017
5.	Karzan Wakil, Rebwar Bakhtyar	Improving web movie recommender system based on emotions.	collaborative filtering , content based filtering	• The challenge was to capture user rating.	2015
6.	Muyeed Ahmed, Raiyan khan	Movie recommendation system using clustering & pattern recognition network.	K-means clustering	• The challenge was to separate users in order to find users with similar tastes of movies.	February 26, 2018

Figure 2.2: Articles

2.2 Overview of related works

The research papers mentioned provide an overview of diverse approaches to movie recommendation systems. They explore techniques like semi-supervised learning, NLP tools, filtering, and clustering to enhance movie recommendations. Each study tackles specific challenges, from scalability issues to capturing user ratings accurately. Notably, these projects highlight the importance of personalized recommendations. The research spans different years, demonstrating the evolving landscape of movie recommendation methodologies. Overall, these papers collectively contribute to the advance.

2.3 Advantages and Limitations of existing systems

Advantages: The existing movie recommendation systems mentioned in the papers offer several advantages, including improved recommendation accuracy and personalized recommendations. These advantages are crucial for enhancing the user experience and ensuring that users receive movie suggestions that align with their preferences.

Disadvantages: However, these systems also face certain limitations, including challenges related to data processing, scalability, and computational complexity. These limitations can impact the system's performance and may require significant computational resources to overcome.

Additionally, it's noted that some of the papers have minimal emphasis on filtering techniques, which are a fundamental aspect of recommendation systems. This could potentially affect the system's ability to filter and refine recommendations based on user preferences and content features.

Chapter 3

Proposed System

3.1 System Requirements

3.1.1 Hardware Requirments

RAM - 8.00 GB (7.87 GB usable)

Processor - Intel(R) Core (TM) i5-10300H CPU @ 2.50GHz 2.50 GHz

System-type - 64-bit operating system, x64-based processor

Version - 20H2

Edition - Windows 10 Home Single Language

3.1.2 Software Requirments

Language - Python

Operating system - Windows 10

Tools – Jupyter Notebook

3.2 Design of the System

1. Data Acquisition:

• Collect movie data: Gather information about movies, including titles, descriptions, genres, ratings, and user preferences. Data can be obtained from various sources, such as movie databases, user ratings, and user profiles.

2. Data Preprocessing:

• Data Cleaning: Handle missing values and clean the dataset for consistency.

• Text Processing: Preprocess movie descriptions by removing stop words, performing stemming or lemmatization, and converting text to numerical representations

Recommendation Algorithms:

• Content-Based Filtering: Recommend movies based on similarities between movie descriptions and user profiles.

• Collaborative Filtering: Utilize user-user or item-item collaborative filtering to suggest movies based on user behavior and preferences.

• Hybrid Models: Combine content-based and collaborative filtering to improve recommendation accuracy.

Model Training and Evaluation: • Train Models: Train the recommendation models using relevant algorithms.

• Evaluate Models: Assess model performance through evaluation metrics like RMSE (Root Mean Square Error).

3.3 Algorithms and Techniques used

3.3.1 Simple Recommender

Algorithm: This recommender uses a straightforward popularity-based approach. Movies are ranked based on their popularity, which is typically measured by factors like ratings, votes, or revenue. Techniques: Sorting and filtering the movies based on popularity and optionally by genre.

3.3.2 Content-Based Recommender (Movie Description Based Recommender)

Algorithm: Content-based recommendation relies on analyzing the content or features of the items (in this case, movies) to make recommendations. Techniques: Natural Language Processing (NLP): To analyze movie descriptions and taglines. Text vectorization (e.g., TF-IDF or Word Embeddings): To convert textual data into numerical representations. Cosine Similarity: To measure the similarity between movies based on their textual features.

3.3.3 Metadata-Based Recommender

Algorithm: This is an extension of the content-based approach, but it includes additional metadata, such as information about the cast, crew, and keywords associated with movies. Techniques: Data merging: Combining the movie dataset with crew and keyword datasets. Feature engineering: Creating features from metadata (e.g., director, actors, genres). Similarity computation: Measuring similarity between movies based on metadata features. Recommending movies with similar metadata characteristics.

3.3.4 Collaborative Filtering

Algorithm: Collaborative filtering makes recommendations based on the behavior and preferences of users. There are two main types: User-Based Collaborative Filtering: Recommends items to a user based on the preferences of users similar to them. Item-Based Collaborative Filtering: Recommends items to a user based on the similarity between items they have shown an interest in. Techniques: User-Item Matrix: Creating a matrix that represents user-item interactions (ratings, views, etc.). Similarity measures (e.g., cosine similarity or Pearson correlation): To find similar users or items. Predicting user preferences based on similar users or items.

3.3.5 Hybrid Recommender

Algorithm: A hybrid recommender combines multiple recommendation techniques to provide more accurate and personalized recommendations. It can integrate collaborative filtering, content-based filtering, and other algorithms. Techniques: Weighted hybridization: Combining the scores from different recommenders with specific weights. Switching hybridization: Using one recommender for some users and another for different users, based on certain criteria. Meta-level hybridization: Building a model that combines recommendations from various sources.

Chapter 4

Implementation

4.1 Tools and Technologies used

Jupyter Notebook: Code development

Programming Languages: Python

Data preprocessing: NLTK (Natural Language Toolkit), Scikit-Learn

Collaborative Filtering: Surprise

4.2 Modules and their descriptions

We are using a variety of Python modules in our project, including pandas, sklearn, seaborn, and others. In order to implement the various machine learning methods, the sklearn packages are employed, while seaborn is used to produce different plots and distribute data. Pandas is mostly used for data preparation and analysis.

4.2.1 Pandas (import pandas as pd)

Pandas is a powerful data manipulation library in Python. It is used for handling structured data, such as movie information, in data frames.

4.2.2 NumPy (import numpy as np)

NumPy is a fundamental library for numerical operations in Python. It is often used in conjunction with Pandas for data manipulation and mathematical computations.

4.2.3 Matplotlib (import matplotlib.pyplot as plt)

Matplotlib is a data visualization library used to create various types of plots and charts for exploring and presenting data.

4.2.4 Seaborn (import seaborn as sns)

Seaborn is a data visualization library that provides a high-level interface for creating aesthetically pleasing and informative statistical graphics.

4.2.5 Scikit-Learn

These are tools for converting text data, such as movie descriptions, into numerical representations for use in machine learning models. The linear kernel and cosine similarity functions are used for calculating the similarity between items or users in the context of collaborative filtering recommendation systems.

4.2.6 NLTK (Natural Language Toolkit)

• SnowballStemmer and WordNetLemmatizer: NLTK provides text processing libraries for natural language understanding. These modules are used for stemming and lemmatization, which are common text preprocessing techniques.
• Wordnet (from nltk.corpus import wordnet): WordNet is a lexical database that can be used to find synonyms and related words in NLP tasks.

4.2.7 Surprise (Python scikit for building and analyzing recommender systems)

Reader, Dataset, SVD, evaluate: Surprise is a Python library specifically designed for building and analyzing recommender systems. These modules are used to define a reader, create a dataset, apply matrix factorization (SVD), and evaluate the recommendation system's performance.

4.3 Flow of the System

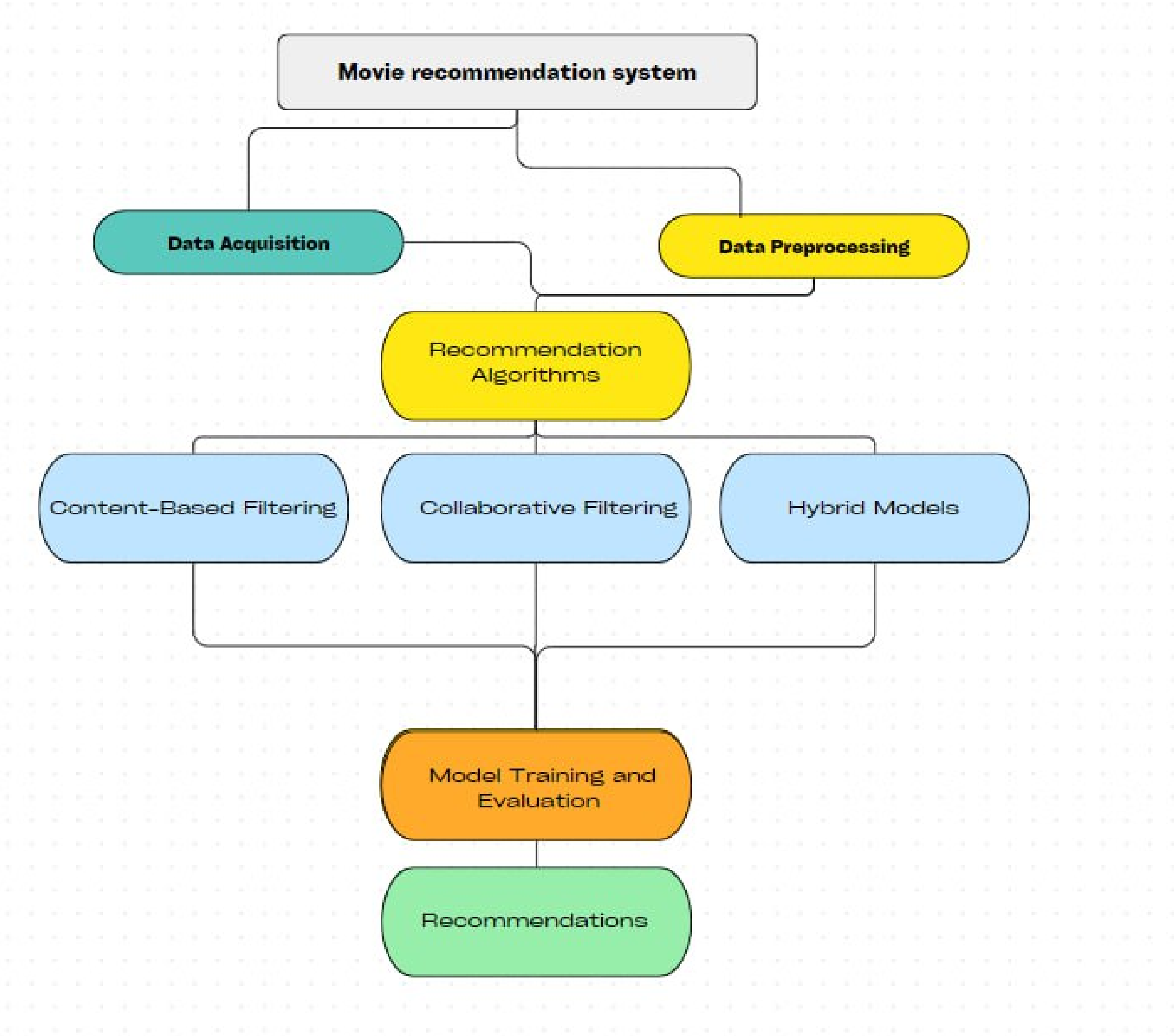


Figure 4.1: Flow chart

Chapter 5

Results and Analysis

5.1 Performance Evaluation

RMSE (Root Mean Square Error): Calculate the RMSE to measure the accuracy of the predicted ratings compared to the actual user ratings for the movies in the dataset

```
In [53]: svd = SVD()
         cross_validate(svd, data, measures=['RMSE', 'MAE'])

Out[53]: {'test_rmse': array([0.89145788, 0.89186611, 0.89796676, 0.90442265, 0.89831894]),
          'test_mae': array([0.68733393, 0.6898595 , 0.6909076 , 0.69250632, 0.69262692]),
          'fit_time': (1.0570917129516602,
           1.0568773746490479,
           1.1253776550292969,
           1.029179573059082,
           1.199488878250122),
          'test_time': (0.12394142150878906,
           0.2369089126586914,
           0.13228654861450195,
           0.10017037391662598,
           0.13175344467163086)}
```

5.2 Comparison with existing systems

In comparison to the existing projects, our project demonstrates a comprehensive and versatile approach to movie recommendation. While the existing projects have employed specific models and techniques, your project combines multiple recommendation algorithms and techniques to provide a more well-rounded and personalized recommendation system. Variety of Algorithms: Unlike some existing projects that focus on specific algorithms (e.g., collaborative filtering or SVM), our project incorporates a range of algorithms, including simple recommender, content-based recommender, metadata-based recommender, collaborative filtering, and hybrid recommendation. This diversity enables your system to adapt to different user preferences and scenarios. Challenges Addressed: Our project addresses some of the challenges faced by existing systems, such as scalability, accuracy in computing ratings, and the ability to capture user profiles effectively. By combining various techniques and models, our project aims to provide solutions to these challenges. Publication Date: Our project is well-timed in terms of relevance as it takes advantage of more recent technologies and methods. This ensures that the latest advancements in recommendation systems are considered. Holistic Approach: By combining content-based and collaborative filtering, as well as metadata-based recommendations, our project offers a holistic approach that considers both movie content and user behavior. This contributes to a more accurate and personalized recommendation system.

5.3 Limitations and future scope

While our movie recommendation system project exhibits a promising array of recommendation algorithms and techniques, it's important to recognize certain limitations and chart a path for future enhancements. Challenges include the reliance on data quality and quantity, the 'cold start' problem for new users and movies, scalability concerns as the system grows, and the need for more diverse recommendations. Addressing privacy issues related to user data is another crucial consideration. Looking forward, the project's future scope is brimming with opportunities. We can enhance personalization through deep learning methods, achieve real-time recommendations, improve explainability, and implement A/B testing for data-driven refinements. Extending the system to mobile and web platforms, exploring multi-modal recommendations, and measuring user engagement metrics are vital for a comprehensive approach. A feedback loop for user input and the incorporation of enriched metadata are also on the horizon. By navigating these limitations and embracing these future possibilities, our movie recommendation system aims to provide increasingly accurate, diverse, and personalized movie suggestions to users, ensuring its continued relevance and effectiveness in the dynamic landscape of movie recommendations.

Chapter 6

Conclusion and Recommendations

6.1 Summary of the Project

The movie recommendation system project is a comprehensive endeavor to provide users with tailored movie suggestions, taking into account various algorithms and techniques. The project embraces diverse recommendation strategies, including simple recommendation based on popularity, content-based recommendations derived from natural language processing and similarity measures, metadata-based recommendations that encompass cast, crew, and additional movie information, collaborative filtering for personalized suggestions based on user behavior, and a hybrid approach that combines multiple recommendation techniques. In summary, the design of the movie recommendation system project focuses on developing an advanced recommendation system that offers personalized movie suggestions based on user preferences. It encompasses algorithm development, advanced filtering techniques, adaptability to changing preferences, and a strong emphasis on user satisfaction and engagement. The project aims to contribute to a more tailored and enjoyable movie-watching experience for users.

6.2 Recommendations for future work

1. Advanced Machine Learning Techniques: Consider incorporating more advanced machine learning techniques, such as deep learning models (e.g., neural collaborative filtering) and reinforcement learning, to capture intricate patterns in user behavior and movie features. These techniques can further enhance the system's ability to make accurate recommendations.
2. Dynamic Real-Time Recommendations: Develop a real-time recommendation system that continuously adapts to users' changing preferences and behavior. This can involve the use of streaming data and real-time analytics to provide up-to-the-minute movie suggestions.
3. Explainability and Transparency: Enhance the explainability of recommendations by implementing techniques that provide users with insights into why a particular movie is being recommended. Transparent recommendations can foster user trust and improve their overall experience.
4. A/B Testing and Evaluation: Establish a robust A/B testing framework to systematically evaluate the performance of different recommendation algorithms and techniques. Use metrics such as click-through rates, conversion rates, and user retention to assess the effectiveness of each approach.
5. Multi-Modal Recommendations: Explore the integration of multi-modal data sources, including images, audio, and user-generated content (e.g., reviews and social media posts). Incorporating these additional data types can lead to more holistic and accurate recommendations.

Bibliography

[1] Roy, Sushmita, Mahendra Sharma, and Santosh Kumar Singh. ”Movie recommendation system using semi-supervised learning.” In 2019 Global conference for advancement in technology (GCAT), pp. 1-5. IEEE, 2019.

[2] Kapoor, Nimish, Saurav Vishal, and K. S. Krishnaveni. ”Movie recommendation system using nlp tools.” In 2020 5th International Conference on Communication and Electronics Systems (ICCES), pp. 883-888. IEEE, 2020.

[3] Jain, Kartik Narendra, Vikrant Kumar, Praveen Kumar, and Tanupriya Choudhury. ”Movie recommendation system: hybrid information filtering system.” In Intelligent Computing and Information and Communication: Proceedings of 2nd International Conference, ICICC 2017, pp. 677-686. Springer Singapore, 2018.

[4] Subramaniyaswamy, V., R. Logesh, M. Chandrashekhar, Anirudh Challa, and Varadarajan Vijayakumar. ”A personalised movie recommendation system based on collaborative filtering.” International Journal of High Performance Computing and Networking 10, no. 1-2 (2017): 54-63.

[5] Shani, G. Gunawardana, A. (2011). ”Evaluating recommendation systems”, in Recommender systems handbook, ed: Springer, 2011, pp. 257-297.

[6] Ahmed, Muyeed, et al. “TV Series Recommendation Using Fuzzy Inference System, K-Means Clustering and Adaptive Neuro Fuzzy Inference System.” 2017, pp. 1512–1519.

Appendices

Appendix A

Source code

```
pip install scikit-surprise

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from ast import literal_eval
from sklearn.feature_extraction.text import TfidfVectorizer,
    CountVectorizer
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate
from surprise.model_selection import KFold
# from surprise import GridSearch
# from sklearn.grid_search import GridSearchCV


import warnings; warnings.simplefilter('ignore')

md = pd. read_csv('movies_metadata.csv')
md.head()
md['genres'] = md['genres'].fillna('[]').apply(literal_eval).apply(
    lambda x: [i['name'] for i in x] if isinstance(x, list) else [])
vote_counts = md[md['vote_count'].notnull()]['vote_count'].astype('
    int')
vote_averages = md[md['vote_average'].notnull()]['vote_average'].
    astype('int')
C = vote_averages.mean()
C
m = vote_counts.quantile(0.95)
m
md['year'] = pd.to_datetime(md['release_date'], errors='coerce').
    apply(lambda x: str(x).split('-')[0] if x != np.nan else np.nan)
qualified = md[(md['vote_count'] >= m) & (md['vote_count'].notnull())
     & (md['vote_average'].notnull())][['title', 'year', 'vote_count',
     'vote_average', 'popularity', 'genres']]
qualified['vote_count'] = qualified['vote_count'].astype('int')
qualified['vote_average'] = qualified['vote_average'].astype('int')
```

```python
qualified.shape

def weighted_rating(x):
    v = x['vote_count']
    R = x['vote_average']
    return (v/(v+m) * R) + (m/(m+v) * C)
qualified['wr'] = qualified.apply(weighted_rating, axis=1)
qualified = qualified.sort_values('wr', ascending=False).head(250)
qualified.head(15)
s = md.apply(lambda x: pd.Series(x['genres']),axis=1).stack().reset_index(level=1, drop=True)
s.name = 'genre'
gen_md = md.drop('genres', axis=1).join(s)

def build_chart(genre, percentile=0.85):
    df = gen_md[gen_md['genre'] == genre]
    vote_counts = df[df['vote_count'].notnull()]['vote_count'].astype('int')
    vote_averages = df[df['vote_average'].notnull()]['vote_average'].astype('int')
    C = vote_averages.mean()
    m = vote_counts.quantile(percentile)
    qualified = df[(df['vote_count'] >= m) & (df['vote_count'].notnull()) & (df['vote_average'].notnull())][['title', 'year', 'vote_count', 'vote_average', 'popularity']]
    qualified['vote_count'] = qualified['vote_count'].astype('int')
    qualified['vote_average'] = qualified['vote_average'].astype('int')
    qualified['wr'] = qualified.apply(lambda x: (x['vote_count']/(x['vote_count']+m) * x['vote_average']) + (m/(m+x['vote_count']) * C), axis=1)
    qualified = qualified.sort_values('wr', ascending=False).head(250)
return qualified

links_small = pd.read_csv('links_small.csv')
links_small = links_small[links_small['tmdbId'].notnull()]['tmdbId'].astype('int')
md = md.drop([19730, 29503, 35587])
md['id'] = md['id'].astype('int')
smd = md[md['id'].isin(links_small)]
smd.shape
smd['tagline'] = smd['tagline'].fillna('')
smd['description'] = smd['overview'] + smd['tagline']
smd['description'] = smd['description'].fillna('')
tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=1, stop_words='english')
tfidf_matrix = tf.fit_transform(smd['description'])
tfidf_matrix.shape
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
cosine_sim[0]
smd = smd.reset_index()
titles = smd['title']
indices = pd.Series(smd.index, index=smd['title'])

def get_recommendations(title):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
```

```python
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:31]
    movie_indices = [i[0] for i in sim_scores]
return titles.iloc[movie_indices]

get_recommendations('The Godfather').head(10)
get_recommendations('The Dark Knight').head(10)
credits = pd.read_csv('credits.csv')
keywords = pd.read_csv('keywords.csv')
keywords['id'] = keywords['id'].astype('int')
credits['id'] = credits['id'].astype('int')
md['id'] = md['id'].astype('int')
md.shape
md = md.merge(credits, on='id')
md = md.merge(keywords, on='id')
smd = md[md['id'].isin(links_small)]
smd.shape
smd['cast'] = smd['cast'].apply(literal_eval)
smd['crew'] = smd['crew'].apply(literal_eval)
smd['keywords'] = smd['keywords'].apply(literal_eval)
smd['cast_size'] = smd['cast'].apply(lambda x: len(x))
smd['crew_size'] = smd['crew'].apply(lambda x: len(x))

def get_director(x):
    for i in x:
        if i['job'] == 'Director':
            return i['name']
    return np.nan
smd['director'] = smd['crew'].apply(get_director)
smd['cast'] = smd['cast'].apply(lambda x: [i['name'] for i in x] if isinstance(x, list) else [])
smd['cast'] = smd['cast'].apply(lambda x: x[:3] if len(x) >=3 else x)
smd['keywords'] = smd['keywords'].apply(lambda x: [i['name'] for i in x] if isinstance(x, list) else [])
smd['cast'] = smd['cast'].apply(lambda x: [str.lower(i.replace(" ", "")) for i in x])
smd['director'] = smd['director'].astype('str').apply(lambda x: str.lower(x.replace(" ", "")))
smd['director'] = smd['director'].apply(lambda x: [x,x, x])

def improved_recommendations(title):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:26]
    movie_indices = [i[0] for i in sim_scores]
    movies = smd.iloc[movie_indices][['title', 'vote_count', 'vote_average', 'year']]
    vote_counts = movies[movies['vote_count'].notnull()]['vote_count'].astype('int')
    vote_averages = movies[movies['vote_average'].notnull()]['vote_average'].astype('int')
    C = vote_averages.mean()
    m = vote_counts.quantile(0.60)
    qualified = movies[(movies['vote_count'] >= m) & (movies['vote_count'].notnull()) & (movies['vote_average'].notnull())]
    qualified['vote_count'] = qualified['vote_count'].astype('int')
    qualified['vote_average'] = qualified['vote_average'].astype('int
```

```
')
    qualified['wr'] = qualified.apply(weighted_rating, axis=1)
    qualified = qualified.sort_values('wr', ascending=False).head(10)
    return qualified
reader = Reader()
ratings = pd.read_csv('./ratings_small.csv')
ratings.head()
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
kf = KFold(n_splits=5)
kf.split(data)
svd = SVD()
cross_validate(svd, data, measures=['RMSE', 'MAE'])
trainset = data.build_full_trainset()
svd.fit(trainset)
ratings[ratings['userId'] == 1]
svd.predict(1, 302, 3)

def convert_int(x):
    try:
        return int(x)
    except:
        return np.nan
id_map = pd.read_csv('./links_small.csv')[['movieId', 'tmdbId']]
id_map['tmdbId'] = id_map['tmdbId'].apply(convert_int)
id_map.columns = ['movieId', 'id']
id_map = id_map.merge(smd[['title', 'id']], on='id').set_index('title')
indices_map = id_map.set_index('id')
def hybrid(userId, title):
    idx = indices[title]
    tmdbId = id_map.loc[title]['id']
    #print(idx)
    movie_id = id_map.loc[title]['movieId']
    sim_scores = list(enumerate(cosine_sim[int(idx)]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:26]
    movie_indices = [i[0] for i in sim_scores]
    movies = smd.iloc[movie_indices][['title', 'vote_count', 'vote_average', 'year', 'id']]
    movies['est'] = movies['id'].apply(lambda x: svd.predict(userId, indices_map.loc[x]['movieId']).est)
    movies = movies.sort_values('est', ascending=False)
    return movies.head(10)
```

Appendix B

Screen shots

B.1 Importing Data

```
md = pd. read_csv('movies_metadata.csv')
md.head()
```

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	release_dat
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	...	1995-10-3
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	tt0113497	en	Jumanji	When siblings Judy and Peter discover an encha...	...	1995-12-1
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	NaN	15602	tt0113228	en	Grumpier Old Men	A family wedding reignites the ancient feud be...	...	1995-12-2
3	False	NaN	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	31357	tt0114885	en	Waiting to Exhale	Cheated on, mistreated and stepped on, the wom...	...	1995-12-2

Figure B.1: Importing Data

B.2 Simple Recommender Results

```
qualified.head(15)
```

	title	year	vote_count	vote_average	popularity	genres	wr
15480	Inception	2010	14075	8	29.108149	[Action, Thriller, Science Fiction, Mystery, A...	7.917588
12481	The Dark Knight	2008	12269	8	123.167259	[Drama, Action, Crime, Thriller]	7.905871
22879	Interstellar	2014	11187	8	32.213481	[Adventure, Drama, Science Fiction]	7.897107
2843	Fight Club	1999	9678	8	63.869599	[Drama]	7.881753
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892	8	32.070725	[Adventure, Fantasy, Action]	7.871787
292	Pulp Fiction	1994	8670	8	140.950236	[Thriller, Crime]	7.868660
314	The Shawshank Redemption	1994	8358	8	51.645403	[Drama, Crime]	7.864000
7000	The Lord of the Rings: The Return of the King	2003	8226	8	29.324358	[Adventure, Fantasy, Action]	7.861927
351	Forrest Gump	1994	8147	8	48.307194	[Comedy, Drama, Romance]	7.860656
5814	The Lord of the Rings: The Two Towers	2002	7641	8	29.423537	[Adventure, Fantasy, Action]	7.851924
256	Star Wars	1977	6778	8	42.149697	[Adventure, Action, Science Fiction]	7.834205
1225	Back to the Future	1985	6239	8	25.778509	[Adventure, Comedy, Science Fiction, Family]	7.820813
834	The Godfather	1972	6024	8	41.109264	[Drama, Crime]	7.814847
1154	The Empire Strikes Back	1980	5998	8	19.470959	[Adventure, Action, Science Fiction]	7.814099
46	Se7en	1995	5915	8	18.45743	[Crime, Mystery, Thriller]	7.811669

```
build_chart('Romance').head(15)
```

	title	year	vote_count	vote_average	popularity	wr
10309	Dilwale Dulhania Le Jayenge	1995	661	9	34.457024	8.565285
351	Forrest Gump	1994	8147	8	48.307194	7.971357
876	Vertigo	1958	1162	8	18.20822	7.811667
40251	Your Name.	2016	1030	8	34.461252	7.789489
883	Some Like It Hot	1959	835	8	11.845107	7.745154
1132	Cinema Paradiso	1988	834	8	14.177005	7.744878
19901	Paperman	2012	734	8	7.198633	7.713951
37863	Sing Street	2016	669	8	10.672862	7.689483
882	The Apartment	1960	498	8	11.994281	7.599317
38718	The Handmaiden	2016	453	8	16.727405	7.566166
3189	City Lights	1931	444	8	10.891524	7.558867
24886	The Way He Looks	2014	262	8	5.711274	7.331363
45437	In a Heartbeat	2017	146	8	20.82178	7.003959
1639	Titanic	1997	7770	7	26.88907	6.981546
19731	Silver Linings Playbook	2012	4840	7	14.488111	6.970581

B.3 Content Based Recommender Results

```
get_recommendations('The Godfather').head(10)
```

```
973        The Godfather: Part II
8387                   The Family
3509                         Made
4196          Johnny Dangerously
29                Shanghai Triad
5667                         Fury
2412              American Movie
1582     The Godfather: Part III
4221                     8 Women
2159               Summer of Sam
Name: title, dtype: object
```

```
get_recommendations('The Dark Knight').head(10)
```

```
7931                       The Dark Knight Rises
132                               Batman Forever
1113                              Batman Returns
8227    Batman: The Dark Knight Returns, Part 2
7565                  Batman: Under the Red Hood
524                                       Batman
7901                            Batman: Year One
2579                Batman: Mask of the Phantasm
2696                                         JFK
8165    Batman: The Dark Knight Returns, Part 1
Name: title, dtype: object
```

B.4 Meta Data Based Results

```
get_recommendations('The Godfather').head(10)
```

```
973        The Godfather: Part II
8387                   The Family
3509                         Made
4196          Johnny Dangerously
29                 Shanghai Triad
5667                         Fury
2412               American Movie
1582      The Godfather: Part III
4221                      8 Women
2159                Summer of Sam
Name: title, dtype: object
```

```
get_recommendations('The Dark Knight').head(10)
```

```
7931                           The Dark Knight Rises
132                                   Batman Forever
1113                                  Batman Returns
8227       Batman: The Dark Knight Returns, Part 2
7565                      Batman: Under the Red Hood
524                                           Batman
7901                               Batman: Year One
2579                    Batman: Mask of the Phantasm
2696                                             JFK
8165       Batman: The Dark Knight Returns, Part 1
Name: title, dtype: object
```

B.5 Popularity Based Recommender Results

```
improved_recommendations('The Dark Knight')
```

	title	vote_count	vote_average	year	wr
585	Batman	2145	7	1989	6.704647
11851	How the Grinch Stole Christmas!	364	7	1966	6.045470
28657	Focus	2588	6	2015	5.891557
17428	Super 8	2496	6	2011	5.888152
19719	Kon-Tiki	248	7	2012	5.883116
13597	Il Divo	166	7	2008	5.730475
1350	Young Guns	262	6	1988	5.529145
9916	Guess Who	230	5	2005	5.160068
17758	Our Idiot Brother	369	5	2011	5.132360
150	Batman Forever	1529	5	1995	5.054144

```
improved_recommendations('Mean Girls')
```

	title	vote_count	vote_average	year	wr
1295	An American Werewolf in London	571	7	1981	6.242075
32514	The Visit	1405	6	2015	5.821797
7065	Something's Gotta Give	422	6	2003	5.617156
7067	Girl with a Pearl Earring	384	6	2003	5.599371
1665	The Horse Whisperer	296	6	1998	5.551076
10550	Good Night, and Good Luck.	274	6	2005	5.537126
13541	Frontier(s)	186	6	2007	5.471428
13107	Rachel Getting Married	165	6	2008	5.452897
14767	Carriers	288	5	2009	5.147209
5971	Darkness Falls	161	4	2003	4.908042

B.6 Hybrid Filtering Results

```
hybrid(1, 'Avatar')
```

	title	vote_count	vote_average	year	id	est
7016	Nausicaä of the Valley of the Wind	808.0	7.7	1984	81	3.132310
1180	The Good, the Bad and the Ugly	2371.0	8.1	1966	429	3.081894
16462	True Grit	1701.0	7.2	2010	44264	2.809107
6367	Duel at Diablo	22.0	6.3	1966	1403	2.789684
12822	Young People Fucking	73.0	5.9	2007	13019	2.773393
7401	Wit	31.0	7.0	2001	26976	2.754131
3457	Caddyshack	370.0	6.7	1980	11977	2.750091
15219	Steam of Life	11.0	6.9	2010	52903	2.687582
14589	The Box	610.0	5.4	2009	22825	2.680738
6603	Fire	17.0	6.0	1996	513	2.655063

```
hybrid(500, 'Avatar')
```

	title	vote_count	vote_average	year	id	est
1180	The Good, the Bad and the Ugly	2371.0	8.1	1966	429	3.611455
3529	Shanghai Noon	756.0	6.2	2000	8584	3.373114
7016	Nausicaä of the Valley of the Wind	808.0	7.7	1984	81	3.318886
15219	Steam of Life	11.0	6.9	2010	52903	3.283004
4101	Double Impact	219.0	5.3	1991	9594	3.272540
3076	Brenda Starr	7.0	5.1	1989	47070	3.222969
16462	True Grit	1701.0	7.2	2010	44264	3.189730
7401	Wit	31.0	7.0	2001	26976	3.126928
14589	The Box	610.0	5.4	2009	22825	3.119745
3658	The Patriot	1130.0	6.8	2000	2024	3.087489

Appendix C

Data sets used in the project

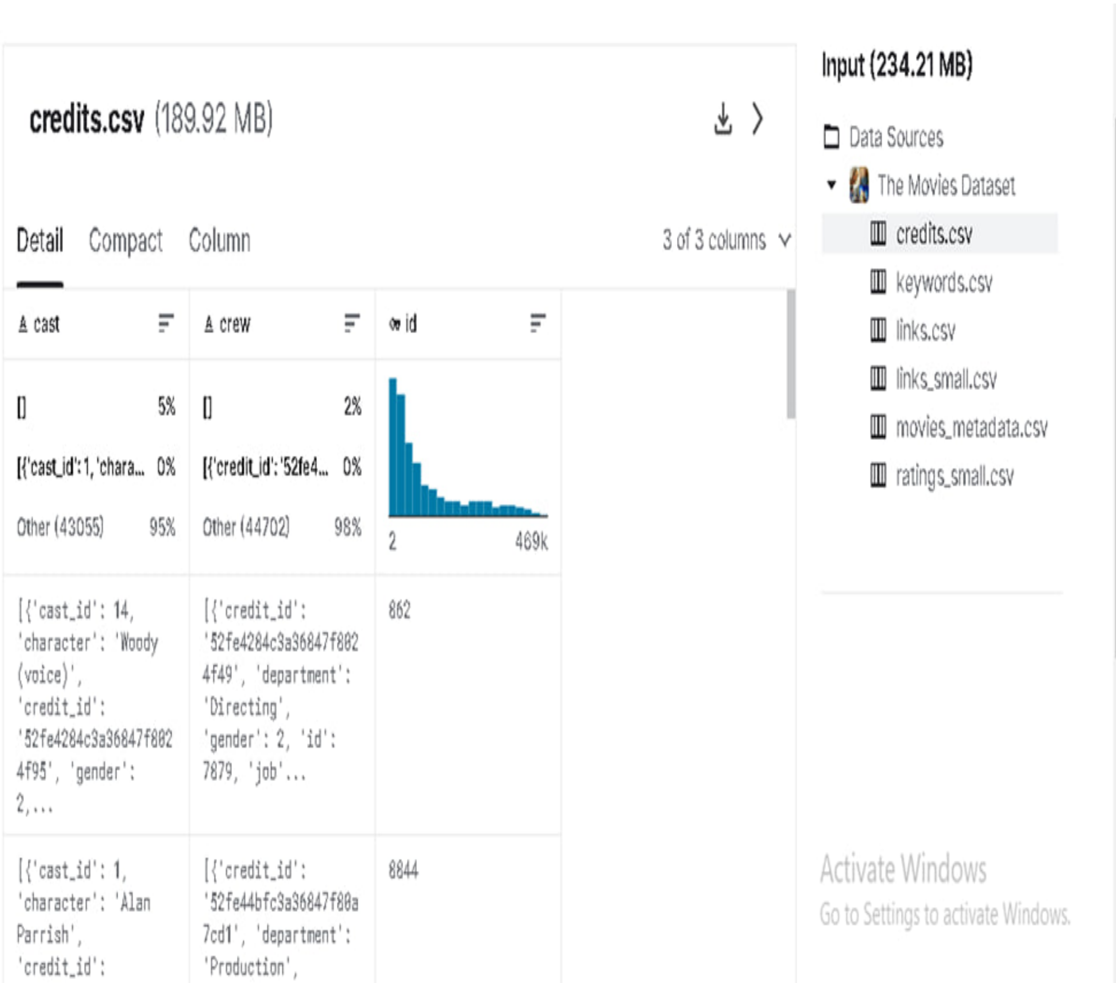
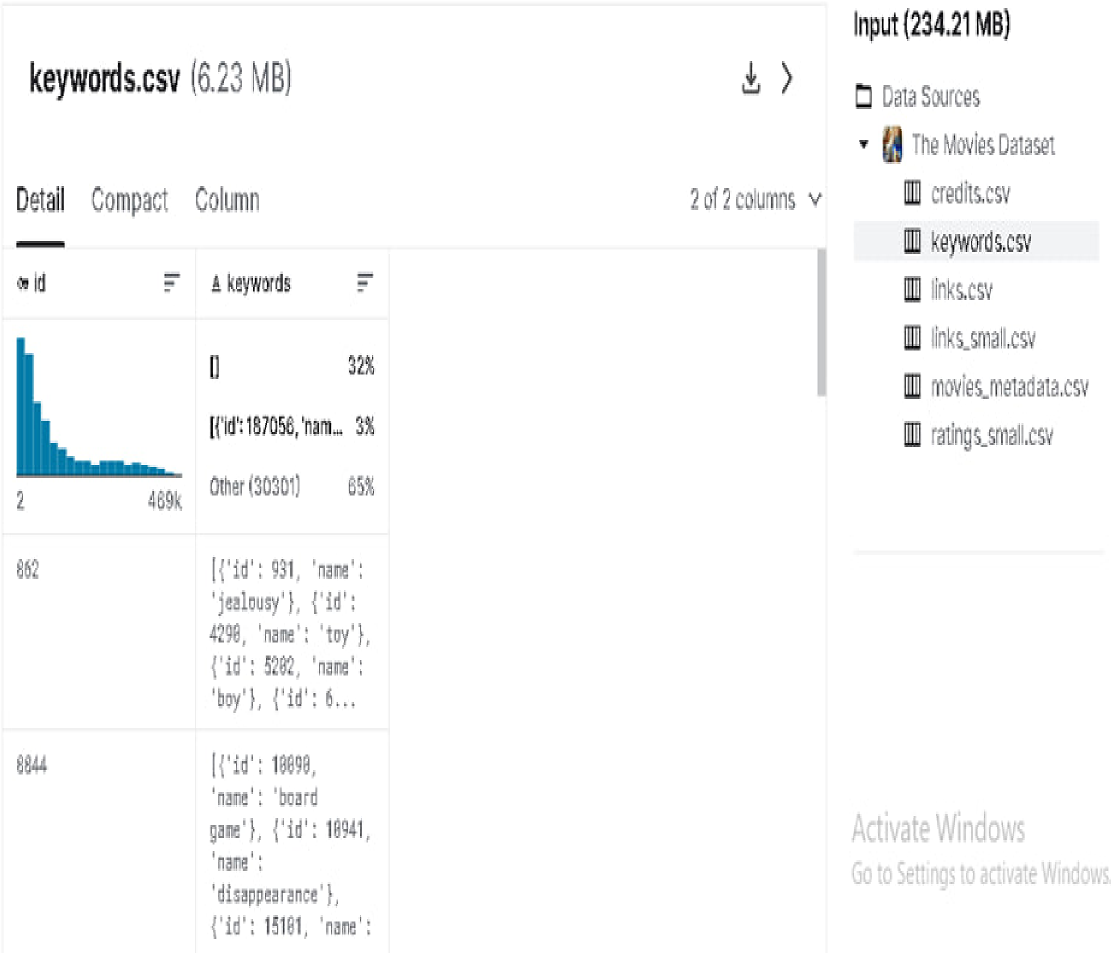
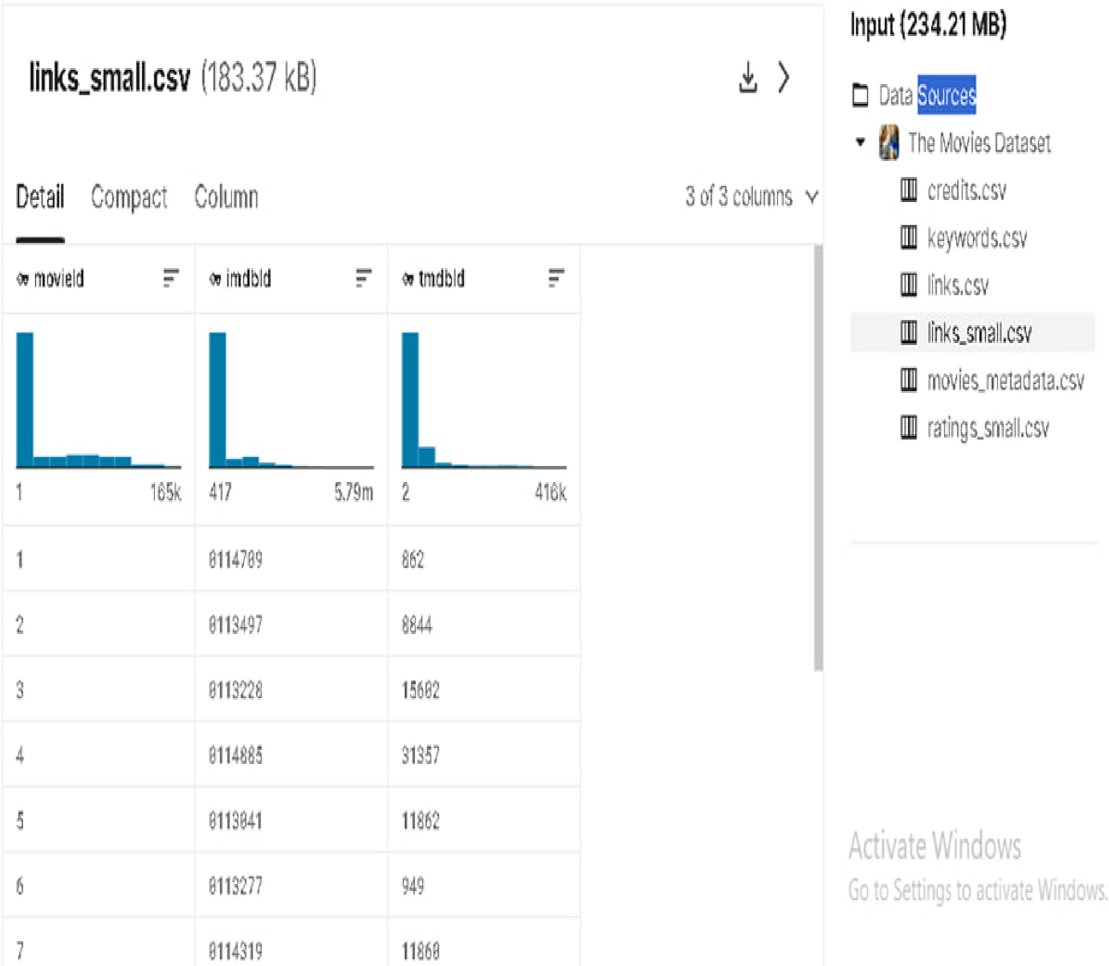
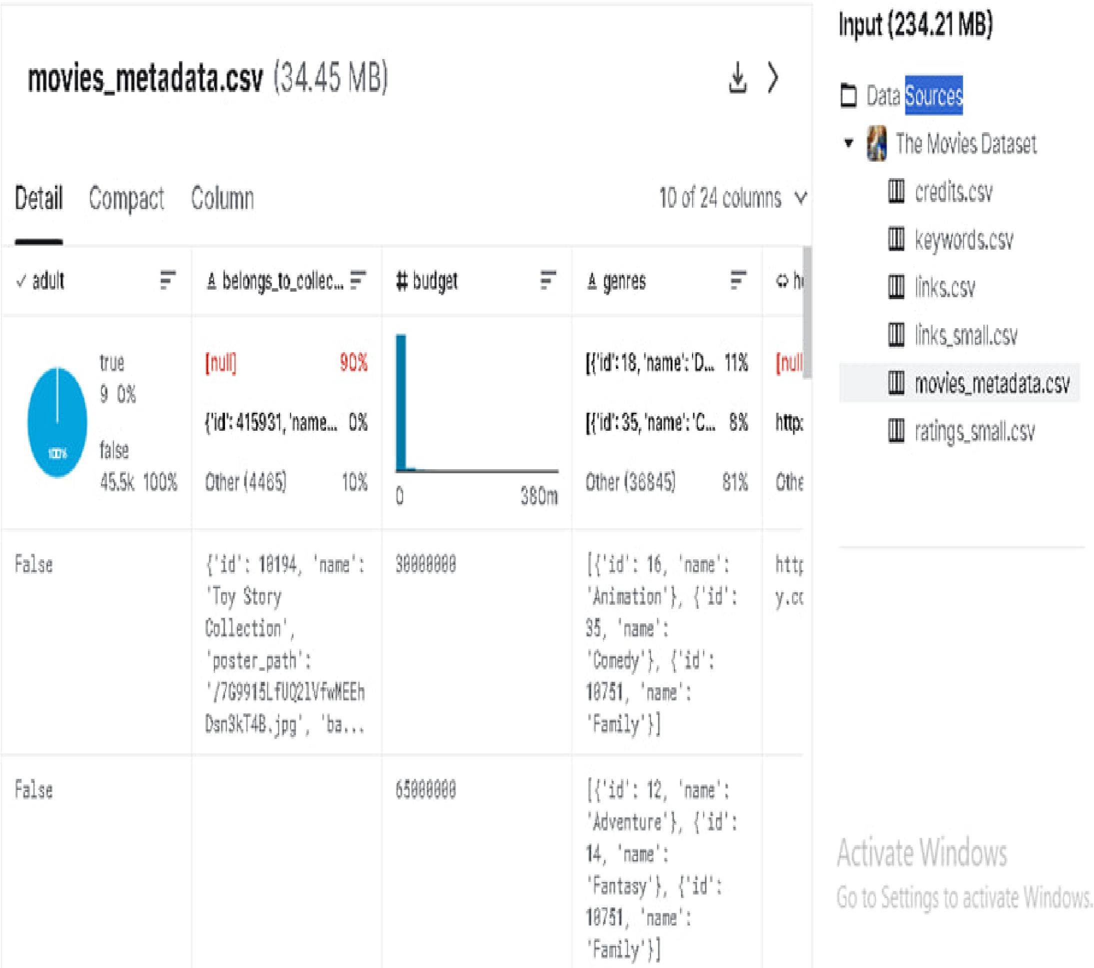


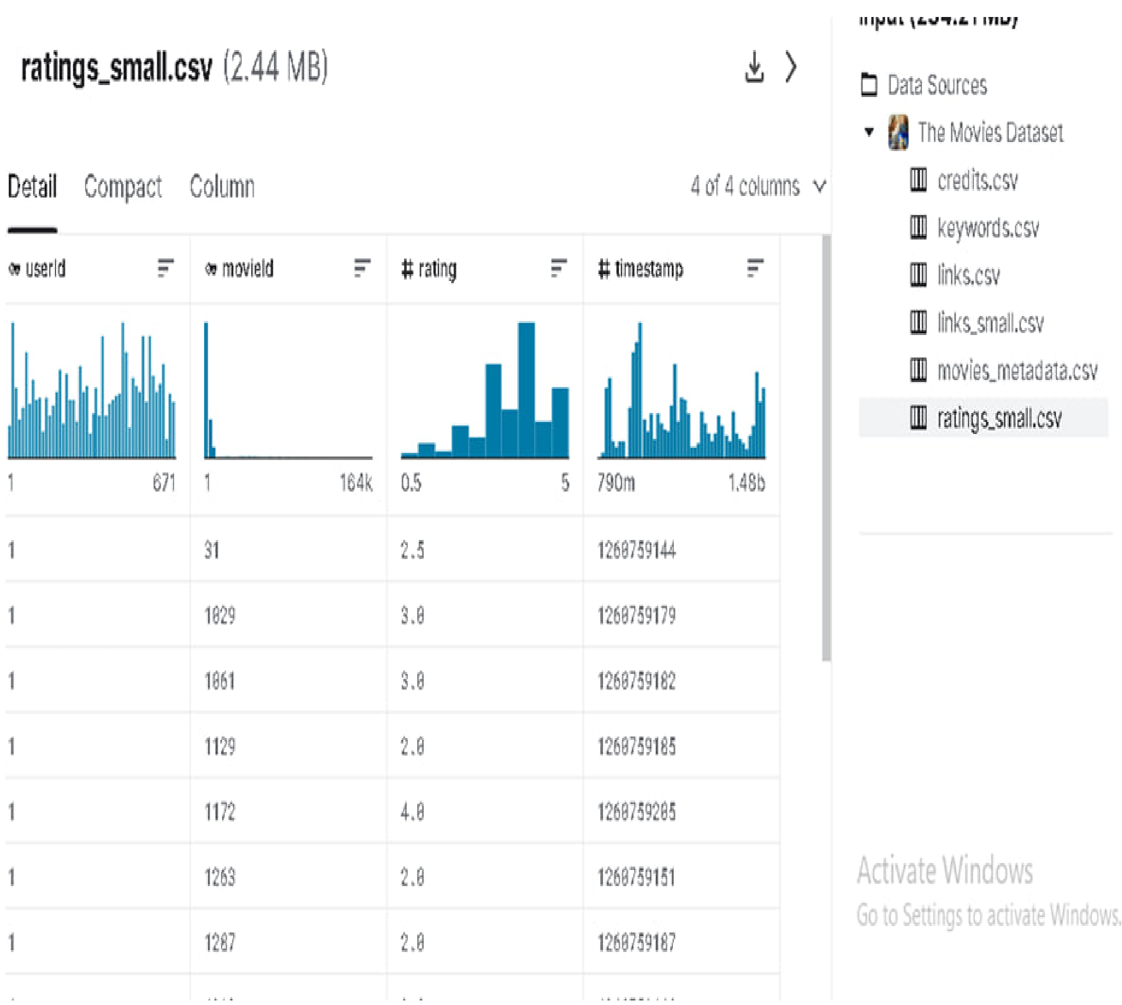
Figure C.1: Data Set


keywords.csv (6.23 MB)
Detail
Compact
Column
2 of 2 columns
id
keywords
2
469k
[]
32%
[{'id': 187056, 'nam...
3%
Other (30301)
65%
862
[{'id': 931, 'name': 'jealousy'}, {'id': 4290, 'name': 'toy'}, {'id': 5202, 'name': 'boy'}, {'id': 6...
8844
[{'id': 10090, 'name': 'board game'}, {'id': 10941, 'name': 'disappearance'}, {'id': 15101, 'name':
Input (234.21 MB)
Data Sources
The Movies Dataset
credits.csv
keywords.csv
links.csv
links_small.csv
movies_metadata.csv
ratings_small.csv
Activate Windows
Go to Settings to activate Windows.


links.csv (989.11 kB)
Detail
Compact
Column
3 of 3 columns
movieId
imdbId
tmdbId
1
176k
1
7.16m
2
469k
1
0114709
862
2
0113497
8844
3
0113228
15602
4
0114885
31357
5
0113041
11862
6
0113277
949
Input (234.21 MB)
Data Sources
The Movies Dataset
credits.csv
keywords.csv
links.csv
links_small.csv
movies_metadata.csv
ratings_small.csv
Activate Windows
Go to Settings to activate Windows.


links_small.csv (183.37 kB)
Detail
Compact
Column
3 of 3 columns
movieId
imdbId
tmdbId
1
165k
417
5.79m
2
416k
1 0114709 862
2 0113497 8844
3 0113228 15602
4 0114885 31357
5 0113041 11862
6 0113277 949
7 0114319 11860
Input (234.21 MB)
Data Sources
The Movies Dataset
credits.csv
keywords.csv
links.csv
links_small.csv
movies_metadata.csv
ratings_small.csv
Activate Windows
Go to Settings to activate Windows.


movies_metadata.csv (34.45 MB)
Detail
Compact
Column
10 of 24 columns
adult
belongs_to_collec...
budget
genres
true
9 0%
false
45.5k 100%
[null]
90%
{'id': 415931, 'name... 0%
Other (4465)
10%
0
380m
[{'id': 18, 'name': 'D... 11%
[{'id': 35, 'name': 'C... 8%
Other (36845)
81%
False
{'id': 10194, 'name': 'Toy Story Collection', 'poster_path': '/7G9915LfUQ2lVfwMEEhDsn3kT4B.jpg', 'ba...
30000000
[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id': 10751, 'name': 'Family'}]
False
65000000
[{'id': 12, 'name': 'Adventure'}, {'id': 14, 'name': 'Fantasy'}, {'id': 10751, 'name': 'Family'}]
Input (234.21 MB)
Data Sources
The Movies Dataset
credits.csv
keywords.csv
links.csv
links_small.csv
movies_metadata.csv
ratings_small.csv
Activate Windows
Go to Settings to activate Windows.


ratings_small.csv (2.44 MB)
Detail
Compact
Column
4 of 4 columns
userId
movieId
rating
timestamp
1
671
1
164k
0.5
5
790m
1.48b
1 31 2.5 1260759144
1 1029 3.0 1260759179
1 1061 3.0 1260759182
1 1129 2.0 1260759185
1 1172 4.0 1260759205
1 1263 2.0 1260759151
1 1287 2.0 1260759187
Data Sources
The Movies Dataset
credits.csv
keywords.csv
links.csv
links_small.csv
movies_metadata.csv
ratings_small.csv
Activate Windows
Go to Settings to activate Windows.