

SCTR's Pune Institute of Computer Technology
Dhankawadi, Pune
A.Y. 2022-23

DSBDAL MINI PROJECT REPORT ON
"Movie Recommendation Model"

SUBMITTED BY

Akhilesh Ingole - 33132
Nikita Karande - 33138
Prathamesh Khude - 33141
Sanket Kittad - 33142

Under the guidance of
Prof. Abhijeet Karve



DEPARTMENT OF INFORMATION TECHNOLOGY
ACADEMIC YEAR 2022-23

SCTR's PUNE INSTITUTE OF COMPUTER TECHNOLOGY
DEPARTMENT OF INFORMATION TECHNOLOGY



CERTIFICATE

This is to certify that the project report entitled

"Movie Recommendation Model"

Submitted by

Akhilesh Ingole - 33132
Nikita Karande - 33138
Prathamesh Khude - 33141
Sanket Kittad - 33142

is a bonafide work carried out by them under the supervision of Prof. Abhijeet Karve and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University, for the DSBDA Lab in Third year Engineering of Information Technology.

Prof. Abhijeet karve
Internal Guide

Dr. A. S. Ghotkar
HOD IT

Date:
Place: Pune

Acknowledgement

We are highly obliged to the people who have given us the much-needed guidance for the mini project work. First, I would like to convey a word of gratitude to my guide, Prof. Abhijeet Karve for guiding us throughout project work and providing me excellent support by valuable guidance and by providing sufficient time for completion of my work. Without their immense help it would have been really difficult to complete this work in time.

We are also extremely grateful to Dr. Archana S. Ghotkar, Head of Information Technology Department for providing all facilities and every help for smooth progress of project work. We are really thankful to the entire staff, our friends and parents for their kind support and necessary help provided by them on time.

Akhilesh Ingole

Nikita Karande

Prathamesh Khude

Sanket Kittad

Abstract

Movie recommendation systems have become an integral part of the modern digital entertainment industry, helping users discover relevant and personalized content. This mini project aims to implement a movie recommendation model using the Scikit-learn library in Python.

The project leverages a dataset consisting of movie ratings and user preferences to build a recommendation system based on collaborative filtering. Collaborative filtering predicts users' interests by collecting preferences from a community of users with similar tastes. Scikit-learn, a popular machine learning library in Python, provides efficient tools for building and evaluating such models.

The implementation follows a step-by-step approach, starting with data pre-processing and exploratory analysis to gain insights into the dataset. Techniques such as data cleaning, feature engineering, and handling missing values are employed to ensure the quality of the input data. By implementing a movie recommendation model using Scikit-learn, this mini project provides a hands-on learning experience in building and evaluating collaborative filtering algorithms. The project's insights can be applied to real-world scenarios, contributing to the development of personalized recommendation systems in the entertainment industry.

Contents

Title Page	
Certificate	i
Abstract	iii
Contents	iv
List of Figures	v
1 Introduction	1
2 Literature Survey	2
3 Implmentation Details	4
3.1 Problem statement:	4
3.2 Resources and Data gathering / Dataset Description:	4
3.3 Pre-processing Techniques used and function used:	4
3.4 Model Selection Criteria:	4
3.5 Collaborative Filtering:	5
3.6 User Interface and Visualization techniques used:	5
4 Output	8
5 Conclusion	12
6 References	13

List of Figures

3.1	Count of viewers of genre	6
3.2	Count of ratings	6
3.3	Average rating per genre	7

1. Introduction

In today's digital age, the entertainment industry is flooded with an overwhelming amount of movies, making it increasingly difficult for users to discover content that aligns with their interests. Movie recommendation systems have emerged as a solution to this problem by leveraging machine learning techniques to provide personalized movie suggestions.

This mini project focuses on implementing a movie recommendation model using the Scikit-learn library in Python. Scikit-learn is a powerful and widely-used machine learning library that offers a comprehensive set of tools for data preprocessing, modeling, and evaluation. By leveraging its functionalities, we can construct a collaborative filtering-based recommendation system capable of suggesting movies to users based on their preferences and similarities with other users.

Movie recommendation systems have become indispensable tools in the entertainment industry, helping users navigate the vast array of available movies and discover content tailored to their preferences. This mini project focuses on implementing a movie recommendation model using the Scikit-learn library in Python. By leveraging collaborative filtering algorithms and Scikit-learn's powerful machine learning functionalities, we aim to provide users with personalized movie recommendations based on their historical preferences and similarities with other users. Through data preprocessing, algorithm implementation, and evaluation, this project equips learners with the necessary skills to build effective recommendation systems.

2. Literature Survey

1] "A survey of collaborative filtering techniques":

This survey paper provides a comprehensive overview of collaborative filtering techniques used in recommender systems. It covers both user-based and item-based approaches, discussing their advantages, limitations, and challenges. User-based collaborative filtering measures the similarity between users based on their preferences, while item-based collaborative filtering focuses on the similarity between items. The survey helps in understanding the various algorithms and their applicability in different scenarios.

2] "Collaborative filtering-based recommendation as a regression problem":

This research paper proposes treating collaborative filtering as a regression problem for generating movie recommendations. It explores the use of regression models instead of traditional ranking-based methods. The authors argue that regression-based approaches offer more flexibility and interpretability in recommendation generation. They discuss different regression models and their application in collaborative filtering.

3] "Introduction to recommender systems handbook":

This book chapter serves as a comprehensive introduction to recommender systems, covering various techniques and approaches. It discusses collaborative filtering, content-based methods, and hybrid models. The chapter also covers evaluation methods, such as precision, recall, and accuracy, for assessing the performance of recommendation models. It serves as a valuable resource for understanding the broader landscape of recommender systems.

4] "Collaborative filtering recommender systems":

This survey paper specifically focuses on collaborative filtering recommender systems. It provides an in-depth analysis of user-based, item-based, and model-based collaborative filtering approaches. The paper discusses the strengths and weaknesses of each technique, along with various.

5]”Matrix factorization techniques for recommender systems”:

This influential paper introduces matrix factorization techniques for collaborative filtering in recommender systems. Matrix factorization decomposes the user-item rating matrix into lower-dimensional matrices, representing latent factors that capture user-item interactions. This approach improves recommendation accuracy by identifying hidden patterns and relationships between users and items.

3. Implmentation Details

3.1 Problem statement:

Build a movie recommendation model using the scikit-learn library in Python. The objective is to create a system where users can input the name of a movie and receive instant recommendations for other movies they might like. The recommendations should be based on movie metadata and past user ratings.

Input: The input to the recommendation system is the name of a movie provided by the user.

Output: The output of the recommendation system is a list of recommended movies that are likely to be of interest to the user.

3.2 Resources and Data gathering / Dataset Description:

MovieLens: MovieLens provides various datasets containing movie ratings and metadata. We can choose from different sizes of datasets, ranging from 100,000 ratings to millions of ratings. We can access the MovieLens datasets at: <https://grouplens.org/datasets/movielens/> .

3.3 Pre-processing Techniques used and function used:

Cleaning movie titles using regular expressions (Regex) :This can be a useful technique in pre-processing textual data for movie recommendation systems. Regex allows you to match and manipulate patterns in strings, making it a powerful tool for data cleaning and transformation.

3.4 Model Selection Criteria:

The selected model should have a high accuracy in predicting user preferences and generating relevant recommendations. The model should be scalable to handle large datasets and a growing number of users and items. The model should address the cold start problem, which occurs when there is limited or no historical data for new users or items. The model should not only recommend popular or mainstream items but also introduce users to new and diverse options.

3.5 Collaborative Filtering:

Collaborative filtering models recommend movies based on the similarities between users' preferences or behaviors. Scikit-learn provides collaborative filtering algorithms such as Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF). These models can be effective when you have user ratings data and want to recommend movies based on similarities between users.

3.6 User Interface and Visualization techniques used:

We are using the Jupyter Notebook environment and the widgets module to create a user interface for our movie recommendation system. The on type function is an event handler that gets triggered whenever the user types in the movie name input field. It retrieves the typed title, performs a search using the search function, and then retrieves the movie ID of the top result. The Movie recommender function is the main function that initiates the movie recommendation process. It prompts the user to enter a movie title and displays the movie name input field and recommendation list using the display function.

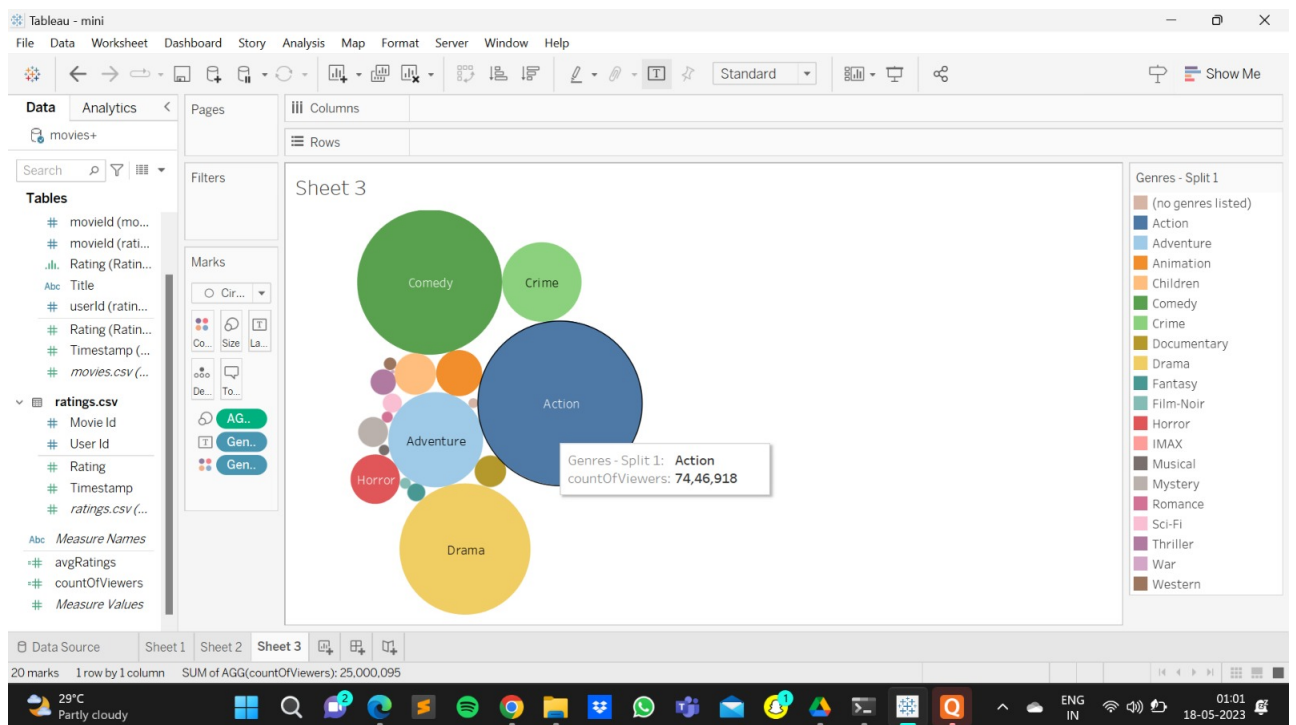


Figure 3.1: Count of viewers of genre

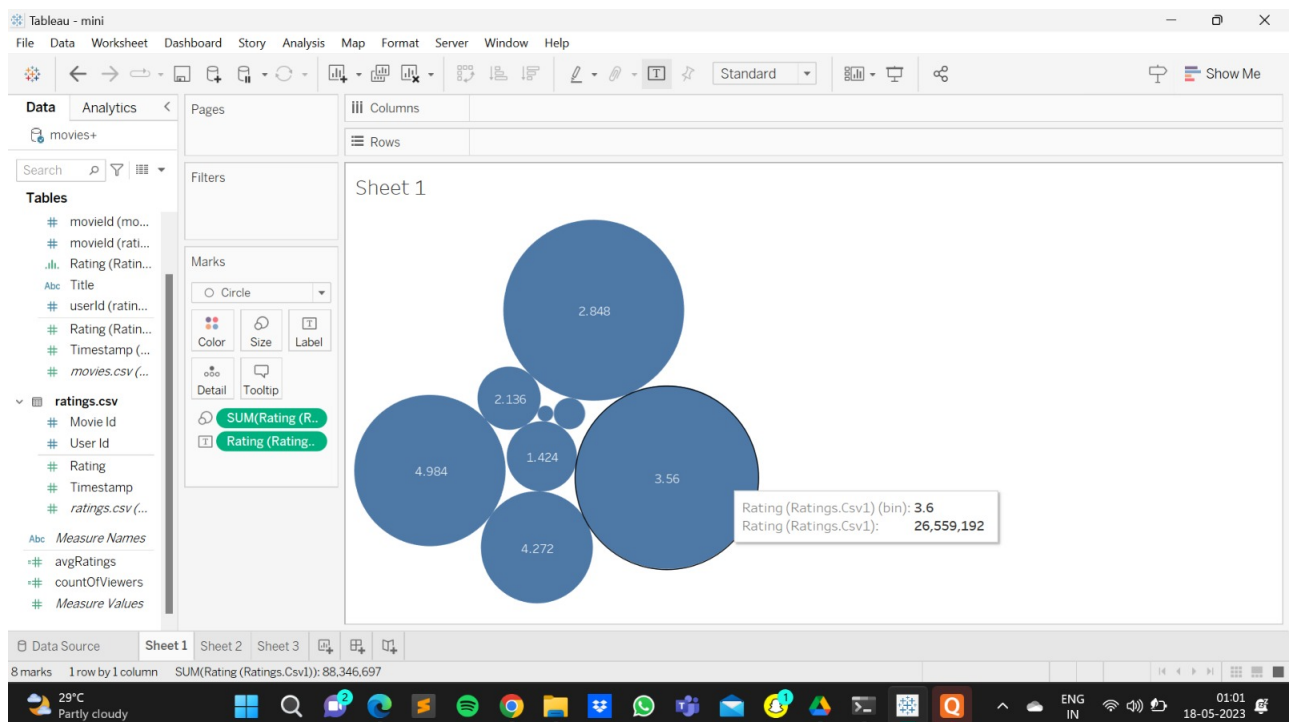


Figure 3.2: Count of ratings

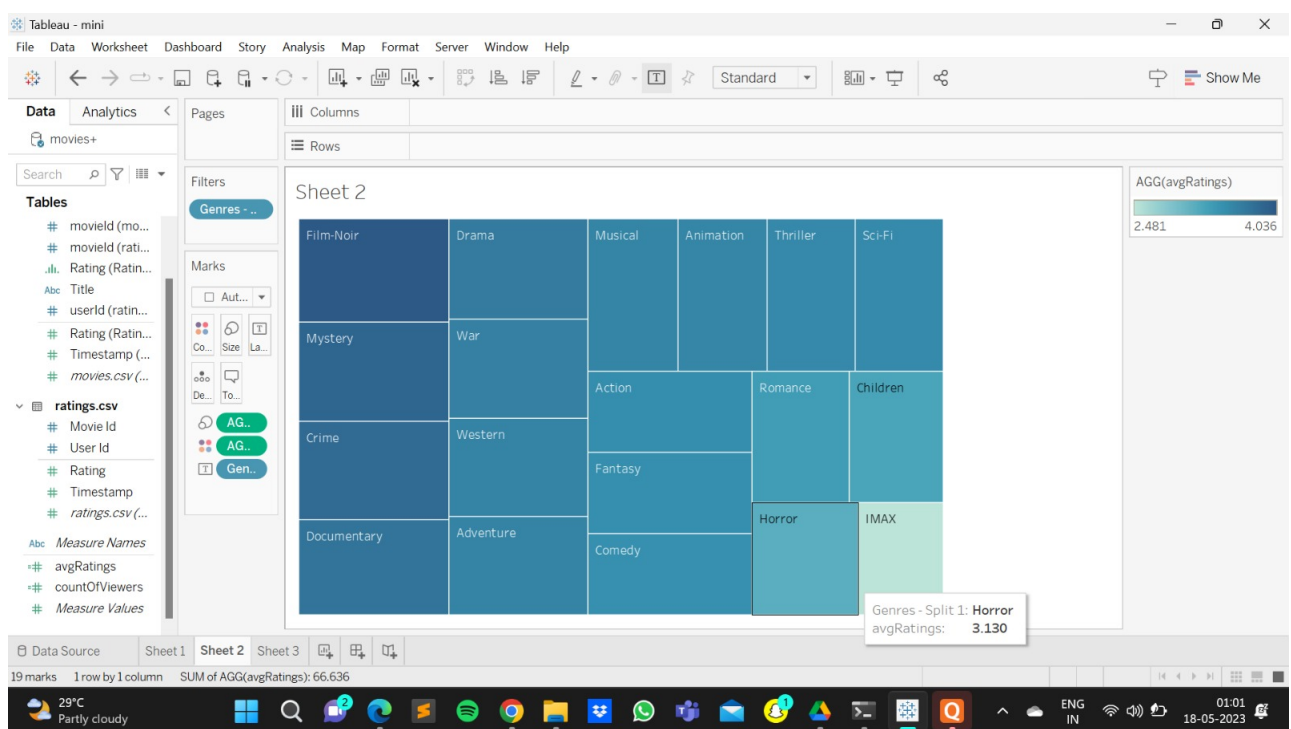
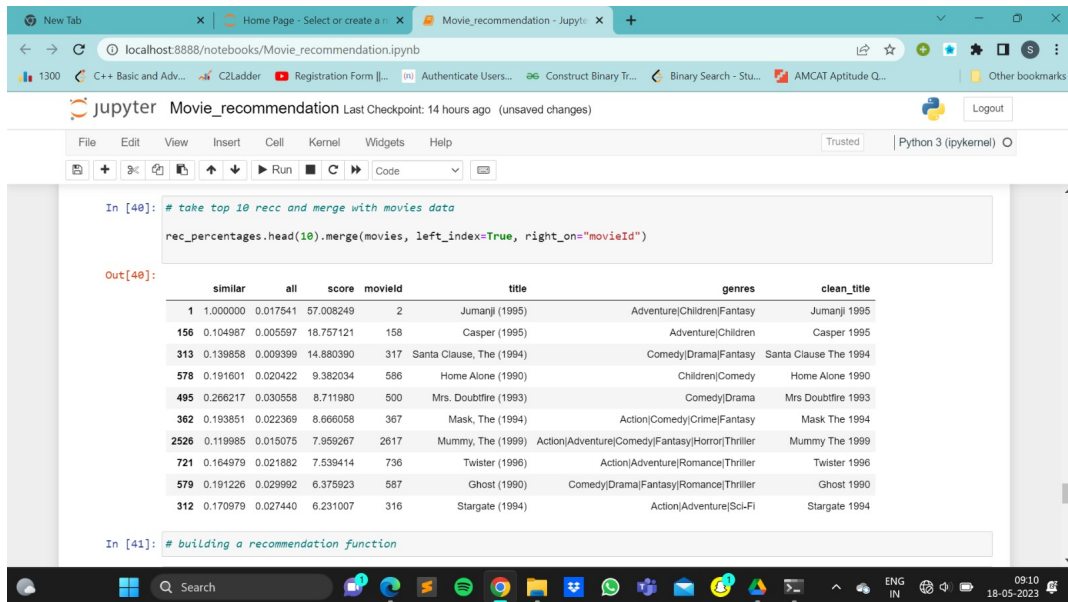


Figure 3.3: Average rating per genre

4. Output

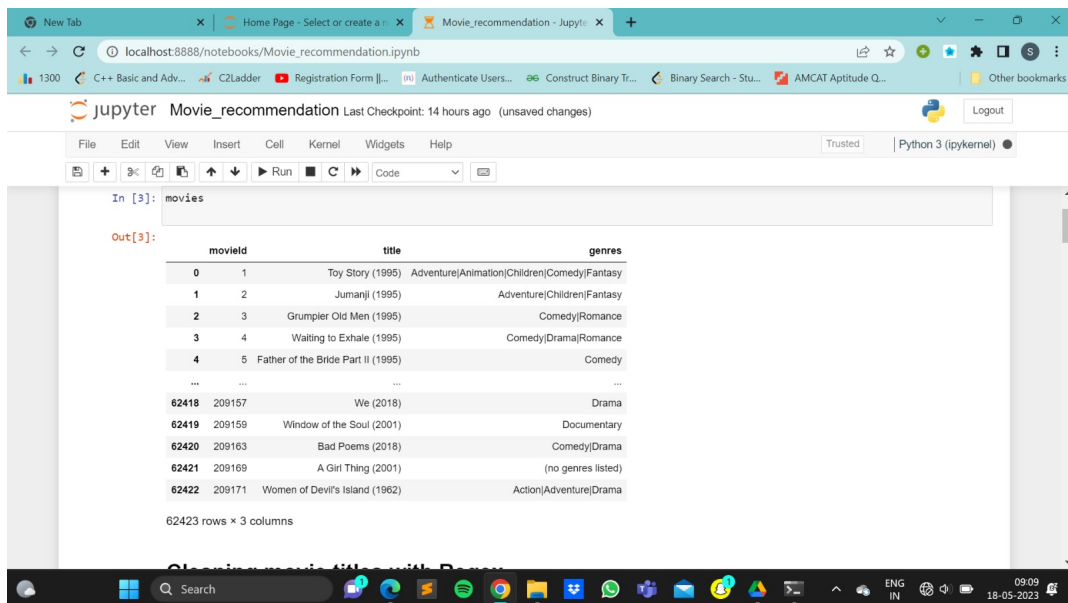


```
In [40]: # take top 10 recs and merge with movies data
rec_percentages.head(10).merge(movies, left_index=True, right_on="movieId")

Out[48]:
```

	similar	all	score	movieId	title	genres	clean_title
1	1.000000	0.017541	57.008249	2	Jumanji (1995)	Adventure Children Fantasy	Jumanji 1995
156	0.104987	0.005597	18.757121	158	Casper (1995)	Adventure Children	Casper 1995
313	0.139858	0.009399	14.880390	317	Santa Clause, The (1994)	Comedy Drama Fantasy	Santa Clause The 1994
578	0.191601	0.020422	9.382034	586	Home Alone (1990)	Children Comedy	Home Alone 1990
495	0.266217	0.030558	8.711980	500	Mrs. Doubtfire (1993)	Comedy Drama	Mrs Doubtfire 1993
362	0.193851	0.022369	8.666058	367	Mask, The (1994)	Action Comedy Crime Fantasy	Mask The 1994
2526	0.119985	0.015075	7.959267	2617	Mummy, The (1999)	Action Adventure Comedy Fantasy Horror Thriller	Mummy The 1999
721	0.164979	0.021882	7.539414	736	Twister (1996)	Action Adventure Romance Thriller	Twister 1996
579	0.191226	0.029992	6.375923	587	Ghost (1990)	Comedy Drama Fantasy Romance Thriller	Ghost 1990
312	0.170979	0.027440	6.231007	316	Stargate (1994)	Action Adventure Sci-Fi	Stargate 1994

```
In [41]: # building a recommendation function
```



```
In [3]: movies

Out[3]:
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
1	2	Jumanji (1995)	Adventure Children Fantasy
2	3	Grumpier Old Men (1995)	Comedy Romance
3	4	Waiting to Exhale (1995)	Comedy Drama Romance
4	5	Father of the Bride Part II (1995)	Comedy
...
62418	209157	We (2018)	Drama
62419	209159	Window of the Soul (2001)	Documentary
62420	209163	Bad Poems (2018)	Comedy Drama
62421	209169	A Girl Thing (2001)	(no genres listed)
62422	209171	Women of Devil's Island (1962)	Action Adventure Drama

62423 rows x 3 columns

Movie_recommendation Last Checkpoint: 14 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [12]: indices
Out[12]: array([28497, 59767, 0, 14813, 3021], dtype=int64)
```

```
In [13]: results
Out[13]:
```

movieid		title	genres	clean_title
3021	3114	Toy Story 2 (1999)	Adventure Animation Children Comedy Fantasy	Toy Story 2 1999
14813	78499	Toy Story 3 (2010)	Adventure Animation Children Comedy Fantasy IMAX	Toy Story 3 2010
0	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	Toy Story 1995
59767	201588	Toy Story 4 (2019)	Adventure Animation Children Comedy	Toy Story 4 2019
20497	106022	Toy Story of Terror (2013)	Animation Children Comedy	Toy Story of Terror 2013

Search function:

```
In [14]: def search(title):
title = clean_title(title)
query_vec = vectorizer.transform([title])
similarity = cosine_similarity(query_vec, tfidf).flatten()
indices = np.argsort(similarity)[-5][::-1]
```

Movie_recommendation Last Checkpoint: 14 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
display(search(title))
movie_input.observe(on_type, names = 'value')
```

```
In [17]: display(movie_input, movie_list)
```

Movie Title:

movieid		title	genres	clean_title
3021	3114	Toy Story 2 (1999)	Adventure Animation Children Comedy Fantasy	Toy Story 2 1999
14813	78499	Toy Story 3 (2010)	Adventure Animation Children Comedy Fantasy IMAX	Toy Story 3 2010
0	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	Toy Story 1995
59767	201588	Toy Story 4 (2019)	Adventure Animation Children Comedy	Toy Story 4 2019
20497	106022	Toy Story of Terror (2013)	Animation Children Comedy	Toy Story of Terror 2013

#comparing according to RATINGS!

```
In [18]: ratings = pd.read_csv("datasets/ratings.csv")
In [19]: ratings
```

Movie_recommendation Last Checkpoint: 14 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [19]: ratings
Out[19]:
```

	userid	movieid	rating	timestamp
0	1	296	5.0	1147880044
1	1	306	3.5	1147868817
2	1	307	5.0	1147868828
3	1	665	5.0	1147878820
4	1	899	3.5	1147868510
...
25000090	162541	50872	4.5	1240953372
25000091	162541	55768	2.5	1240951998
25000092	162541	56176	2.0	1240950697
25000093	162541	58559	4.0	1240953434
25000094	162541	63876	5.0	1240952515

25000095 rows x 4 columns

```
In [20]: ratings.dtypes
```

```
Movie_recommendation - Jupyter
localhost8888/notebooks/Movie_recommendation.ipynb

jupyter Movie_recommendation Last Checkpoint: 14 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [26]: similar_user_recs = similar_user_recs.value_counts() / len(similar_users)
similar_user_recs = similar_user_recs[similar_user_recs > .10]

In [27]: similar_user_recs
Out[27]: movieId
2 1.000000
356 0.503187
480 0.419198
318 0.416573
364 0.410574
...
1221 0.101612
134130 0.101237
99114 0.101237
231 0.100862
2081 0.100862
Name: count, Length: 162, dtype: float64

In [28]: # all the users who watched the movie that is recommended

In [29]: all_users = ratings[(ratings["movieId"].isin(similar_user_recs.index)) & (ratings["rating"] > 4)]
```

```
Movie_recommendation - Jupyter
localhost8888/notebooks/Movie_recommendation.ipynb

jupyter Movie_recommendation Last Checkpoint: 14 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [31]: #find what % of all users recommend thses movies

In [32]: all_user_recs = all_users["movieId"].value_counts() / len(all_users["userId"].unique())

In [33]: all_user_recs
Out[33]: movieId
318 0.339895
296 0.282748
2571 0.242375
356 0.233667
593 0.224374
...
292 0.018969
2 0.017541
2617 0.015075
317 0.009399
158 0.005597
Name: count, Length: 162, dtype: float64

In [34]: #creating a recommended score

In [35]: rec_percentages = pd.concat([similar_user_recs, all_user_recs], axis=1)
```

```
Movie_recommendation - Jupyter
localhost8888/notebooks/Movie_recommendation.ipynb

jupyter Movie_recommendation Last Checkpoint: 14 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [42]: def find_similar_movies(movie_id):
# finding users similar to us
similar_users = ratings[(ratings["movieId"] == movie_id) & (ratings["rating"] > 4)][["userId"].unique()
similar_user_recs = ratings[(ratings["userId"].isin(similar_users)) & (ratings["rating"] > 4)][["movieId"]

#recommending of users who recommended over 10%
similar_user_recs = similar_user_recs.value_counts() / len(similar_users)
similar_user_recs = similar_user_recs[similar_user_recs > .10]

#how common recs were
all_users = ratings[(ratings["movieId"].isin(similar_user_recs.index)) & (ratings["rating"] > 4)]
all_user_recs = all_users["movieId"].value_counts() / len(all_users["userId"].unique())

#generating and sorting score
rec_percentages = pd.concat([similar_user_recs, all_user_recs], axis=1)
rec_percentages.columns = ["similar", "all"]

rec_percentages["score"] = rec_percentages["similar"] / rec_percentages["all"]
rec_percentages = rec_percentages.sort_values("score", ascending=False)

#returning merge score
return rec_percentages.head(10).merge(movies, left_index=True, right_on="movieId")[["score", "title", "genres"]]

In [43]: #Final recommendation of movies code:
```


localhost:6888/notebooks/Movie_recommendation.ipynb

jupyter Movie_recommendation Last Checkpoint: 14 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [45]: Movie_recommender()
```

Enter the Movie title for recommending similar movies :

Movie Title:

	score	title	genres
3903	142.655934	Wall Street (1987)	Drama
4221	16.969153	Mississippi Burning (1988)	Crime Drama Thriller
4749	14.874552	Dirty Harry (1971)	Action Crime Thriller
3634	13.686092	Serpico (1973)	Crime Drama
2140	13.376980	Rounders (1998)	Drama
3291	13.343817	JFK (1991)	Drama Mystery Thriller
3266	13.107743	Hoosiers (a.k.a. Best Shot) (1986)	Drama Romance
3005	12.840212	Natural, The (1984)	Drama
2946	11.524464	Trading Places (1983)	Comedy
3898	11.166051	Planes, Trains & Automobiles (1987)	Comedy

In []:

5. Conclusion

In conclusion, implementing a movie recommendation model using the scikit-learn library in Python as a mini project offers numerous benefits. Through a literature survey, we explored various collaborative filtering techniques, matrix factorization, and hybrid models that form the foundation of recommendation systems. By leveraging the scikit-learn library, we can take advantage of its powerful machine learning algorithms and tools for building an efficient and accurate movie recommendation system.

By implementing the movie recommendation model as a mini project, we provide a practical demonstration of applying collaborative filtering algorithms and the scikit-learn library to a real-world problem. This project not only enhances our understanding of recommendation systems but also showcases our skills in data preprocessing, model training, and result visualization.

Overall, building a movie recommendation model using scikit-learn in Python as a mini project allows us to gain hands-on experience in implementing a recommender system, leveraging state-of-the-art techniques, and delivering personalized movie recommendations to users. It serves as a stepping stone for further exploration in the field of recommendation systems and provides a solid foundation for future projects and applications in the domain of personalized recommendations.

6. References

1. "Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model" by Yehuda Koren. Published in 2008 at the Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD).
2. "Matrix Factorization Techniques for Recommender Systems" by Koren, Yehuda. Published in 2009 at the IEEE Computer Society.
3. "Factorization Machines" by Rendle, Steffen. Published in 2010 at the Proceedings of the 10th IEEE International Conference on Data Mining (ICDM).
4. "SVD++: An Algorithm for Collaborative Filtering Based on Singular Value Decomposition" by Koren, Yehuda. Published in 2008 at the ACM Transactions on Intelligent Systems and Technology (TIST).
5. "Learning to Rank: From Pairwise Approach to Listwise Approach" by Xia, Fen, et al. Published in 2008 at the Proceedings of the 24th International Conference on Machine Learning (ICML).