

SAVITRIBAI PHULE PUNE UNIVERSITY



A MINI PROJECT REPORT ON
Movie Recommendation Model

Submitted by

Name: Pratik Vijay Manjarekar Roll no: 57

CLASS: TE

DIV: A

Under the Guidance of

Ms. Jyoti Rahtwan



Sinhgad Institutes

DEPARTMENT OF COMPUTER ENGINEERING
RMD SINHGAD SCHOOL OF ENGINEERING

WARJE, PUNE – 411058

2023 - 24



Sinhgad Institutes

**DEPARTMENT OF COMPUTER ENGINEERING
RMD SINHGAD SCHOOL OF ENGINEERING**

WARJE, PUNE – 411058

2023 – 24

CERTIFICATE

This is to certify that the project report entitled

“Movie Recommendation Model”

Submitted by

Name: Pratik Vijay Manjarekar

PRN No: 72262586D

is a bona fide work carried out by them under the supervision of Ms. Jyoti Raghtwan and it is submitted towards the partial fulfilment of the requirement of Savitribai Phule Pune University for Third Year.

Ms. Jyoti Raghtwan

Guide

Department of Computer Engineering

Dr. Deepali Newaskar

Head,

Department of Computer Engineering

Dr. V. V. Dixit

Principal

RMD Sinhgad School of Engineering Pune – 58

Place: PUNE

Date:

I

Certificate by Guide

This is to certify that **Mr.Pratik Vijay Manjarekar** has completed the MINI Project work under my guidance and supervision and that, I have verified the work for its originality in documentation, problem statement, implementation and results presented in the Project. Any reproduction of other necessary work is with the prior permission and has given due ownership and included in the references.

Place: PUNE

Date:

Signature of Guide

Ms. Jyoti Raghtwan

II

ACKNOWLEDGEMENT

It is my pleasure to acknowledge a sense of gratitude to all those who helped me in making this project.

I thank my Mini Project Guide **Ms. Jyoti Raghtwan** for helping me and providing all necessary information regarding this project.

I am also thankful to **Dr. Deepali Newaskar (Head - Department of Computer Engineering)** for providing me the required facilities and helping me while carrying out this project work.

Finally, I wish to thank all my teachers and friends for their constructive comments, suggestions and criticism and all those who directly or indirectly helped me in completing this project.

Pratik Vijay
Manjarekar

CONTENTS

Certificate	I
Certificate By Guide	II
Acknowledgement	III
1. Problem Statement	1
2. Introduction	1
3. Requirements	2
4. Algorithm Used	3
5. Coding	4
6. Conclusion	5
7. References	6

PROBLEM STATEMENT

Develop a movie recommendation model using the scikit-learn library in python.

Refer dataset https://github.com/rashida048/Some-NLP-Projects/blob/master/movie_dataset.csv

INTRODUCTION

In the age of digital streaming and vast movie libraries, finding the right film to watch can be a daunting task for viewers. With countless options available, users often seek personalized recommendations to enhance their viewing experience. A movie recommendation system serves as a valuable tool that helps users discover films that align with their preferences and interests.

This project implements a content-based movie recommendation system using machine learning techniques. The system analyzes various features of movies, such as keywords, cast, genres, and directors, to determine similarities between films. By leveraging these features, the system can suggest movies that are likely to resonate with the viewer based on their previous choices.

Key Components of the System:

Data Collection: The system utilizes a dataset containing detailed information about movies, including their attributes and metadata.

Feature Extraction: Relevant features are selected and combined to create a comprehensive representation of each movie.

Similarity Calculation: The system employs cosine similarity to quantify how similar different movies are to one another based on their features.

Recommendation Generation: By analyzing the similarity scores, the system generates a list of recommended movies tailored to the user's input.

This approach not only enhances user satisfaction by providing personalized suggestions but also encourages exploration of diverse films that users may not have considered otherwise. The ultimate goal is to create an engaging and user-friendly experience that simplifies the process of discovering new movies.

REQUIREMENTS

Hardware Requirements:

- Processor: Intel Core i3 or equivalent (later generations recommended)
- RAM: 4GB (8GB or more recommended for better performance)

Software Requirements:

- Operating System: Windows 10 (or macOS, Linux)
- Languages: Python (3.6.3 or later)
- Software: Anaconda, Jupyter Notebook
- Dataset: covid_vaccine_statewise.csv
- Libraries: numpy, pandas, scikit-learn

ALGORITHM

Movie Recommendation System Algorithm:

Step 1: Import Libraries

- pandas for data manipulation.
- CountVectorizer from sklearn for converting text data into feature vectors.
- cosine_similarity from sklearn for calculating similarity between movies.

Step 2: Load Data

- Read the dataset: "movie_dataset.csv".

Step 3: Inspect Dataset

- Print the columns of the dataset to understand its structure.

Step 4: Select Relevant Features

- Define a list of features that will be used for recommendations:
 1. Keywords
 2. Cast
 3. Genres
 4. director

Step 5: Handle Missing Values

- For each feature in the selected list, fill missing values with empty strings to avoid errors during processing.

Step 6: Combine Features

- Define a function combine_features(row) that concatenates the selected features into a single string for each movie.
- Apply this function to create a new column combined_features in the DataFrame.

Step 7: Convert Text Data to Feature Vectors

- Use CountVectorizer to convert the combined_features column into a matrix of token counts.

Step 8: Compute Cosine Similarity

- Calculate the cosine similarity matrix from the count matrix to determine how similar each movie is to every other movie.

Step 9: Define Recommendation Function

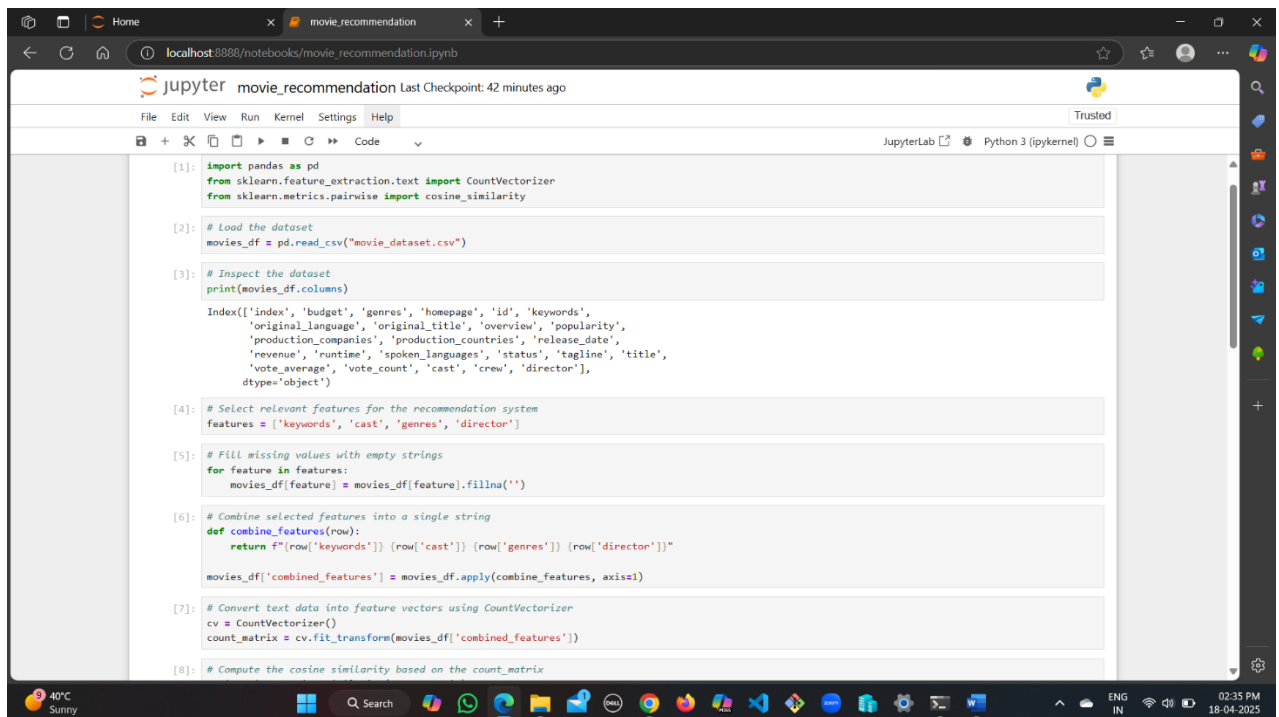
- Create a function get_recommendations(title, cosine_sim) that:
 - Takes a movie title as input.
 - Finds the index of the movie in the DataFrame.
 - Retrieves the similarity scores for that movie.
 - Sorts the movies based on similarity scores.
 - Returns the titles of the top 10 similar movies.

Step 10: Example Usage

- Call the get_recommendations() function with a specific movie title (e.g., "Avatar" or "Thor") and print the recommended movies.

This algorithm outlines the steps taken in the provided code to create a movie recommendation system based on content similarity.

CODE



```
[1]: import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity

[2]: # Load the dataset
movies_df = pd.read_csv("movie_dataset.csv")

[3]: # Inspect the dataset
print(movies_df.columns)

Index(['index', 'budget', 'genres', 'homepage', 'id', 'keywords',
      'original_language', 'original_title', 'overview', 'popularity',
      'production_companies', 'production_countries', 'release_date',
      'revenue', 'runtime', 'spoken_languages', 'status', 'tagline', 'title',
      'vote_average', 'vote_count', 'cast', 'crew', 'director'],
      dtype='object')

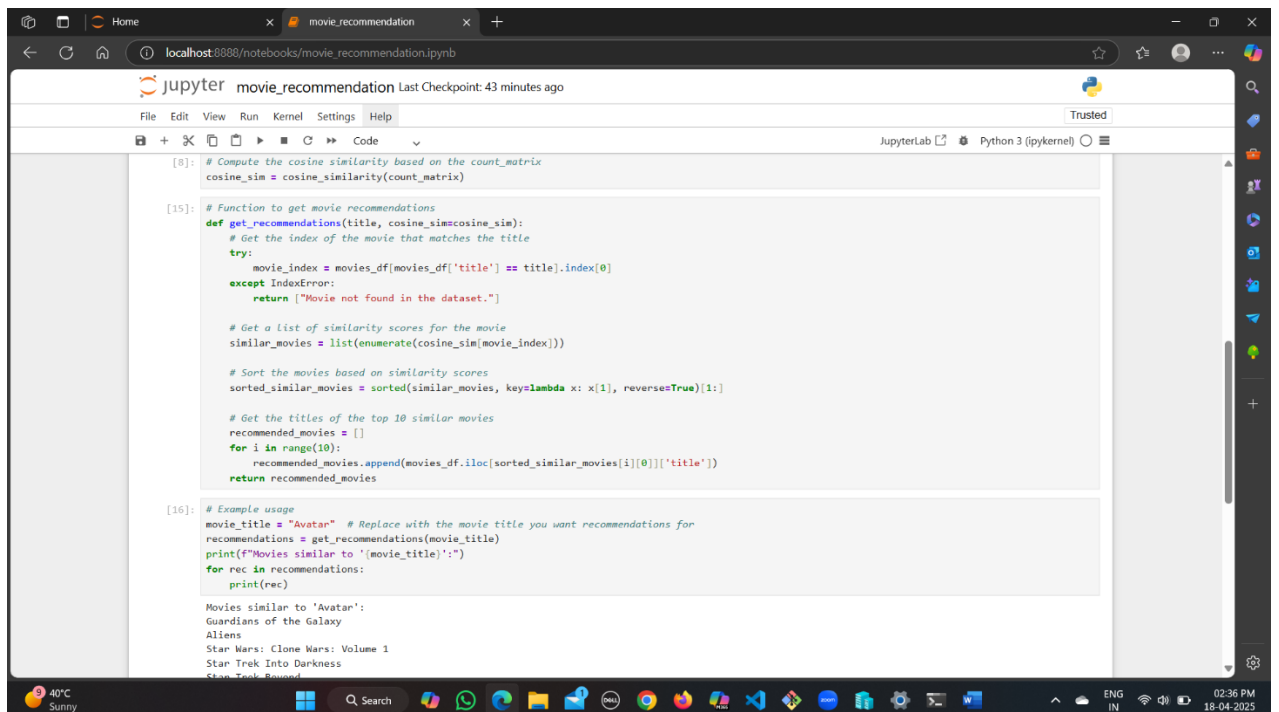
[4]: # Select relevant features for the recommendation system
features = ['keywords', 'cast', 'genres', 'director']

[5]: # Fill missing values with empty strings
for feature in features:
    movies_df[feature] = movies_df[feature].fillna('')

[6]: # Combine selected features into a single string
def combine_features(row):
    return f"{row['keywords']} {row['cast']} {row['genres']} {row['director']}"
movies_df['combined_features'] = movies_df.apply(combine_features, axis=1)

[7]: # Convert text data into feature vectors using CountVectorizer
cv = CountVectorizer()
count_matrix = cv.fit_transform(movies_df['combined_features'])

[8]: # Compute the cosine similarity based on the count_matrix
```



```
[8]: # Compute the cosine similarity based on the count_matrix
cosine_sim = cosine_similarity(count_matrix)

[15]: # Function to get movie recommendations
def get_recommendations(title, cosine_sim=cosine_sim):
    # Get the index of the movie that matches the title
    try:
        movie_index = movies_df[movies_df['title'] == title].index[0]
    except IndexError:
        return ["Movie not found in the dataset."]

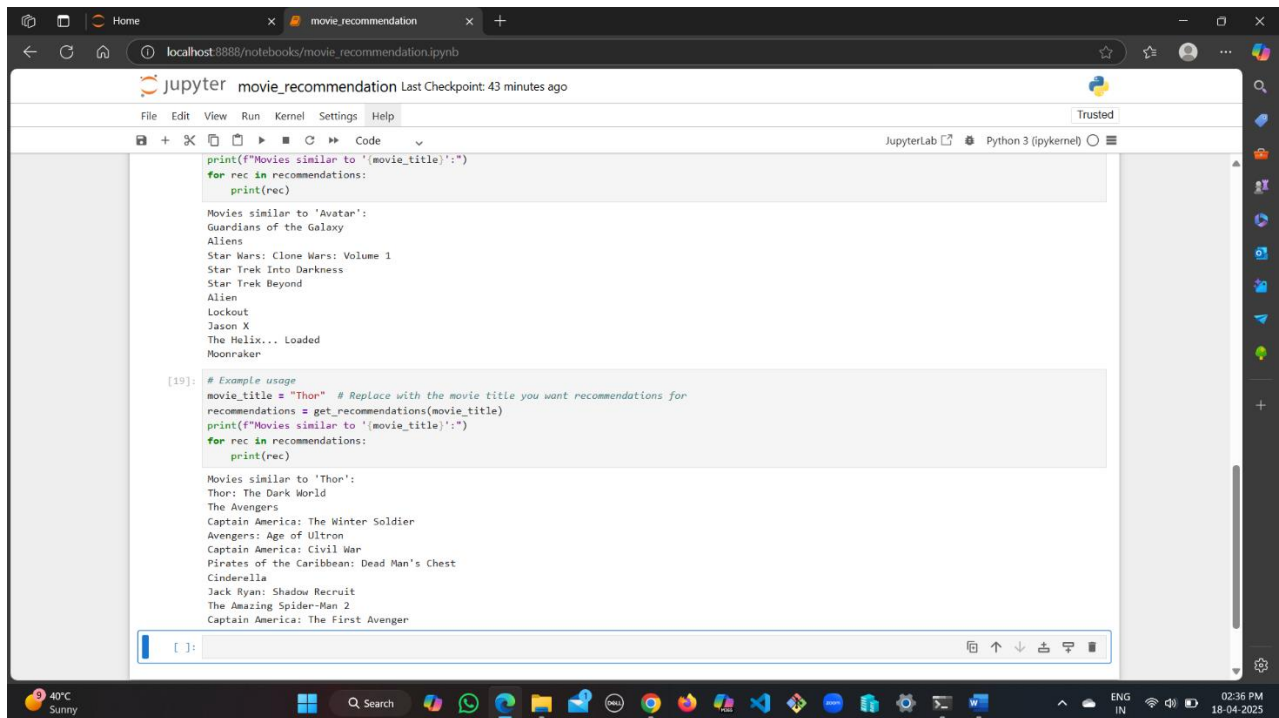
    # Get a list of similarity scores for the movie
    similar_movies = list(enumerate(cosine_sim[movie_index]))

    # Sort the movies based on similarity scores
    sorted_similar_movies = sorted(similar_movies, key=lambda x: x[1], reverse=True)[1:]

    # Get the titles of the top 10 similar movies
    recommended_movies = []
    for i in range(10):
        recommended_movies.append(movies_df.iloc[sorted_similar_movies[i][0]]['title'])
    return recommended_movies

[16]: # Example usage
movie_title = "Avatar" # Replace with the movie title you want recommendations for
recommendations = get_recommendations(movie_title)
print(f"Movies similar to '{movie_title}':")
for rec in recommendations:
    print(rec)

Movies similar to 'Avatar':
Guardians of the Galaxy
Aliens
Star Wars: Clone Wars: Volume 1
Star Trek Into Darkness
Star Trek Beyond
```



```
print(f'Movies similar to '{movie_title}':')
for rec in recommendations:
    print(rec)

Movies similar to 'Avatar':
Guardians of the Galaxy
Aliens
Star Wars: Clone Wars: Volume 1
Star Trek Into Darkness
Star Trek Beyond
Alien
Lockout
Jason X
The Helix... Loaded
Moonraker

[19]: # Example usage
movie_title = "Thor" # Replace with the movie title you want recommendations for
recommendations = get_recommendations(movie_title)
print(f'Movies similar to '{movie_title}':')
for rec in recommendations:
    print(rec)

Movies similar to 'Thor':
Thor: The Dark World
The Avengers
Captain America: The Winter Soldier
Avengers: Age of Ultron
Captain America: Civil War
Pirates of the Caribbean: Dead Man's Chest
Cinderella
Jack Ryan: Shadow Recruit
The Amazing Spider-Man 2
Captain America: The First Avenger
```

Conclusion

The development of a movie recommendation system using content-based filtering demonstrates the power of machine learning and data analysis in enhancing user experiences in the digital entertainment landscape. By leveraging various features such as keywords, cast, genres, and directors, the system effectively identifies and suggests movies that align with individual user preferences.

Key takeaways from this project include:

Personalization: The recommendation system tailors suggestions based on the unique attributes of movies, allowing users to discover films that they are likely to enjoy. This personalization fosters a more engaging viewing experience.

Simplicity and Efficiency: The use of cosine similarity to measure the relationship between movies simplifies the recommendation process, making it efficient and scalable for large datasets.

Exploration of Content: By providing recommendations, the system encourages users to explore a wider range of films, potentially introducing them to genres and titles they may not have considered otherwise.

Foundation for Future Enhancements: This basic framework can be further improved by integrating additional features, such as user ratings and reviews, or by incorporating collaborative filtering techniques to enhance the accuracy of recommendations.

REFERENCES

- Dataset Source: [movie_dataset.csv](#)
- Python Libraries Used:
Pandas, Scikit-learn