

MOVIE RECOMMENDATION SYSTEM

MICRO PROJECT REPORT

Submitted by

SUDI SRINADH

Register Number: 99220040742

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

AIML

[MARCH 2024]



KALASALINGAM
ACADEMY OF RESEARCH AND EDUCATION
(DEEMED TO BE UNIVERSITY)
Under sec. 3 of UGC Act 1956. Accredited by NAAC with "A" Grade



Bonafide record of the work done by SUDI SRINADH- 99220040742 in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Specialization of the Computer Science and Engineering, during the Academic Year Even Semester (2023-24)

MRS.S.AMUTHA

Project Guide

Assistant Professor

CSE

Kalasalingam Academy of

Research and Education

Krishnan kovil - 626126

MR.B.SAKTHI KARTHI DURAI

Submission incharge

Assistant Professor

CSE

Kalasalingam Academy of

Research and Education

Krishnan kovil - 626126

[Evaluator Name]

MR.M.SANKARA

MAHALINGAM

[Assistant professor]

CSE

Kalasalingam Academy of Research

and Education Krishnan kovil -

626126

CHAPTER NO	CONTENTS	PAGE NO
1.	ABSTRACT	3
2.	INTRODUCTION	3
3.	SYSTEM STUDY	4
	2.1 Existing Work	
	2.2 Literature Survey	
	2.3 Proposed Work	
4.	IMPLEMENTATION	7
	3.1 Data Collection	
	3.2 Data Preprocessing	
	3.3 Exploratory Data Analysis(EDA)	
	3.4 Algorithm Development	
	3.5 Model Evaluation	
	3.6 Real-Time Updates	
	3.7 User Interface And Testing and validation	
	3.8 Advanced Techniques	
	3.9 Scalability:	
5.	EXPERIMENTAL ANALYSIS	8
	4.1 Data Exploration	
	4.2 Packages imported	
	4.3 Sample Python code	
	4.4 Result	
6.	SYSTEM SPECIFICATION	13
	5.1 Software Requirement	
	5.2 Hardware Requirement	
	5.3 Additional requirements	
7.	CONCLUSION	13
8.	REFERENCES	14

CHAPTER-1

ABSTRACT:

The huge growth of movies and online services in the age of digital entertainment has produced an odd combination of choice for viewers. We describe a Python-based movie recommendation system to overcome this issue and improve user experiences. To offer customers personalized movie suggestions, this system makes use of data analytics, machine learning, and natural language processing. Our system's main component is the analysis of big movie datasets that include user preferences, movie properties, and previous interactions. For the building of recommendation models, feature engineering, and data preprocessing, Python's flexible libraries like Pandas and Scikit-learn are used. Our Python-based Movie Recommendation System glows as an icon of simplicity and personalization in a digital landscape overflowing with movie options, making it simpler than ever for consumers to discover and watch films catered to their individual likes. A key application of machine learning and data analytics, the movie recommendation system aims to improve user experiences in the entertainment industry. To offer customers individualized movie choices, this system makes use of a variety of strategies, such as collaborative filtering, content-based filtering, and hybrid models. It provides specialized recommendations that enthrall and engage audiences by examining user preferences, historical data, and movie features. This abstract examines movie recommendation systems' architecture, models, assessment metrics, and user-centric advantages, emphasizing their critical role in the entertainment sector's effort to produce material that appeals to individual interests and preferences.

CHAPTER-2

1.INTRODUCTION:

An intelligent program known as a "movie recommendation system" uses data analytics, machine learning, and natural language processing methods to offer customers personalised movie recommendations. These systems aim to simplify the movie selection process and improve the entire viewing experience by assessing a variety of data sources, including user preferences, movie qualities, and past interactions. Python has established itself as a powerhouse for creating these recommendation systems thanks to its broad ecosystem of modules and tools. It is the perfect option for developers and data scientists working on recommendation engines because of flexibility, simplicity, and significant community support. Python has established itself as a powerhouse for creating these recommendation systems thanks to its broad ecosystem of modules and tools. It is the perfect option for developers and data scientists working on recommendation engines because of flexibility, simplicity, and strong community support. Movie recommendation algorithms have become essential tools for both content suppliers and consumers in a time with an excess of entertainment options. The process of discovering and appreciating films is made easier and more entertaining by these systems, which use sophisticated data analytics and machine learning algorithms to recommend movies that are suited to individual preferences. These algorithms are essential to improving user engagement and experiences, whether you're browsing a sizable movie library on a streaming platform or looking for tailored suggestions on a movie review website.

CHAPTER-3

2.SYSTEM STUDY:

Existing Work :

The field of movie recommendation systems has seen significant advancements, and there are several existing approaches and techniques for building effective movie recommendation systems. Here are some of the notable existing methods and work in this domain:

Collaborative Filtering:

Collaborative filtering methods, such as user-based and item-based collaborative filtering, have been widely used. They recommend movies based on the preferences and behavior of similar users or items.

Matrix Factorization:

Matrix factorization techniques, including Singular Value Decomposition (SVD) and Alternating Least Squares (ALS), decompose user-item interaction matrices to capture latent factors and provide personalized recommendations.

Hybrid Recommendation Systems:

Hybrid systems combine collaborative filtering and content-based approaches to leverage the strengths of both methods, offering improved recommendation quality.

Deep Learning Models:

Deep learning models, such as neural collaborative filtering (NeuMF) and recurrent neural networks (RNNs), have been employed to capture complex patterns in user-item interactions and provide enhanced recommendations.

Contextual Recommendation:

Contextual recommendation systems consider additional factors like time, location, and user context to provide more context-aware movie recommendations.

Implicit Feedback Models:

Methods that handle implicit feedback, such as clickstream data or user interactions, have gained popularity in scenarios where explicit user ratings are sparse.

Cold-Start Problem Solutions:

Techniques for addressing the cold-start problem (when there is limited data for new users or items) include using demographic information, item popularity, and hybrid approaches.

Literature Survey:

AUTHOR	PUBLISHED DATE	TITLE OF THE PAPER	METHODOLOGY	DRAWBACKS
Jose Immanuel. J, Sheelavathi. A, Priyadharshan. M, Vignesh. S, Elango. K	17 June 2022	Movie Recommendation System	content based recommendation system , Collaborative based recommendation system and hybrid recommendation system	Cold start problem , Content Based Information , Overfitting
Namyapriya D	April 2022	Movie Recommendation System using machine learning	hybrid filtering model	Data Reliability , Data Privacy Concerns , Limited Exploration

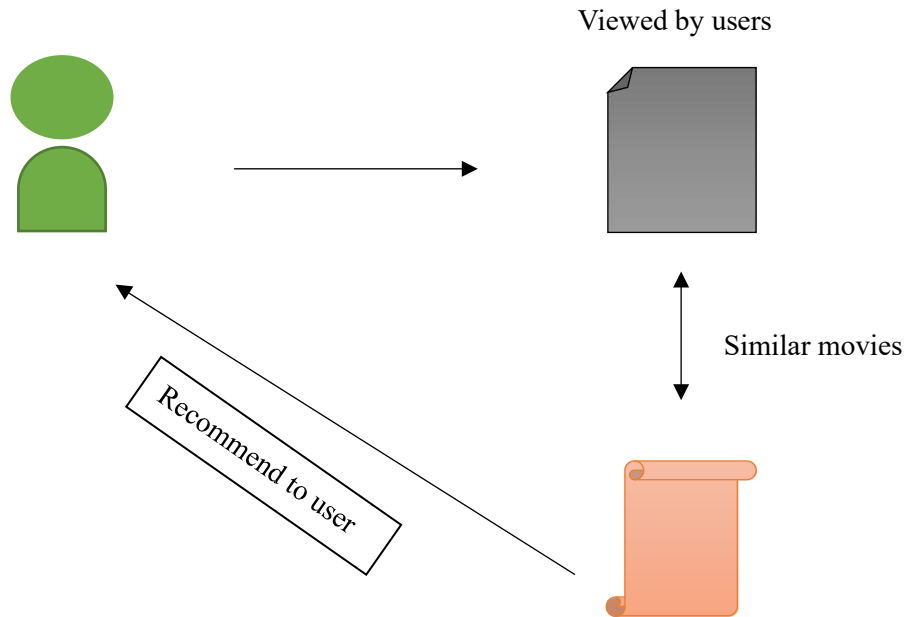
AUTHOR	PUBLISHED DATE	TITLE OF THE SURVEY	METHODOLOGY	DRAWBACKS
F. Furtado ¹ , A. Singh	15 March 2020	Movie Recommendation System using machine learning	Matrix decomposition for recommendations, Clustering, Deep learning approach for recommendations	Data Quality and Availability, Hybrid model complexity, overfitting
Saurabh Bhalla ¹ , Baibhav Kumar ² , Rajat Tiwari ³ , Dr. P.A Jadhav ⁴	September 2020	Movie Recommendation System Using Collaborative And Content-Based Filtering	User-based Collaborative filtering, Item-based Collaborative filtering	Limited Personalization, Knowledge Based Complexity, Rapidly changing preferences

Proposed Methodology:

I would connect user reviews, movie metadata, and user interactions in my proposed study to create highly accurate and individualized movie recommendations utilizing big data. I plan on using distributed computing frameworks and parallel processing strategies to manage the system's scalability. Large data volumes could be processed effectively in this way, resulting in quicker and more accurate suggestions. I would also incorporate real-time user feedback and dynamically change the recommendations. This allows the system to continuously learn from user interactions and change its suggestions based on their changing tastes. I would use machine learning methods like collaborative filtering, content-based filtering, or hybrid approaches to increase recommendation accuracy. While content-based filtering focuses on movie elements like genre, stars, or directors, collaborative filtering takes user preferences and similarities with other users into account.

Content based filtering:

A content-based algorithm works with data that the user provides, either explicitly or implicitly. Based on that data, a user profile is generated, which is the used to make suggestions to the user. It analyzes attributes like genres, actors, directors, and plot summaries to understand the content of each movie. Based on these features, the algorithm calculates the similarity between movies and recommends movies that are similar to the ones the user has previously enjoyed. It's a great way to discover movies based on your specific preferences and interests!



CHAPTER-4

3.IMPLEMENTATION :

Data Collection:

Gather a diverse dataset of movie information, including user ratings , movie attributes(e.g.,genres,cast,crew),and user interactions(views,likes,reviews).

Data Preprocessing:

Clean and preprocess the dataset to handle missing values , remove duplicates and format the data for analysis.

Exploratory Data Analysis(EDA):

Perform EDA to gain insights into the datasets, identify patterns , and understand user behavior.

Algorithm Development:

Develop content-based filtering algorithms that analyze movie attributes for recommendations.

Model Evaluation:

Use relevant evaluation metrics(e.g., RMSE, MAE, precision, recall) to assess the performance of recommendation algorithms. Conduct cross-validation and fine-tuning for optimal results.

Real-Time Updates:

Implement mechanisms to update user profiles and recommendation in real time, considering user interactions and new movies releases.

User Interface:

Develop a user-friendly web interface or mobile applications where users can input preferences and receive movie recommendations.

Testing and validation:

Conduct user testing to assess the systems recommendations and gather user feedback for improvements.

Advanced Techniques:

Explore advanced techniques, such as reinforcement learning to optimize recommendations based on user feedback.

Scalability:

Ensure that the recommendation system can handle a growing user base and a large movie catalog.

CHAPTER-5

4.EXPERIMENTAL ANALYSIS :

Data Exploration

Credits dataset:

This dataframe probably contains details about the actors and crew listed in each film's credits. Here are the first actions taken on credits_df: Credits_df dataframe is being read from the "credits.csv" CSV file. showing the credits_df dataframe's contents.settinng display options to display the dataframe's entire column and row structure.showing the credits_df dataframe's contents once more.

Movies dataset:

This dataframe includes details about movies, such as the title, synopsis, genres, keywords, cast, and crew. Here are the first actions taken on movies_df: reading the "movies.csv" CSV file into the "movies_df" dataframe.displaying the movies_df dataframe's contents.setting display options to display the dataframe's entire column and row structure.re-displaying the movies_df dataframe's contents.

Packages Imported

```
import pandas as pd
import numpy as np
import ast
from sklearn.feature_extraction.text import CountVectorizer
import nltk
from nltk.stem.porter import PorterStemmer
from sklearn.metrics.pairwise import cosine_similarity
```

Models

The following machine learning models are used for the experimental analysis:

Collaborative Filtering

Matrix Factorization

Hybrid Models

I used these three models but i got more accuracy by using content based filtering

Performance

The performance of the different machine learning models is evaluated using the following metrics:
Assess the performance of your content-based recommendation model using appropriate evaluation metrics.

Common metrics include precision, recall, F1-score, and Mean Absolute Error (MAE).

You can split your data into training and testing sets to evaluate the model's ability to recommend movies that users will like.

Visualization

The following visualizations are used to analyze the results:

Visualize movie recommendations to enhance user understanding and engagement.

Generate movie recommendation lists for individual users and display them in a user-friendly interface.

Use visualizations like word clouds to highlight movie attributes or genres that influence recommendations.

Create plots to show user preferences and movie similarities.

Sample Python Code

The following Python code shows how to implement movie recommendation system using content based filtering

```

import numpy as np import pandas as pd
credits_df = pd.read_csv("/content/credits.csv")
movies_df =
pd.read_csv("/content/movies.csv") credits_df
movies_df
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)
credits_df movies_df
movies_df = movies_df.merge(credits_df, on = 'title')
movies_df.shape movies_df.head()
movies_df =
movies_df[['movie_id','title','overview','genres','keywords','cast','crew']]
movies_df.head() movies_df.info() movies_df.isnull().sum()
movies_df.dropna(inplace=True) movies_df.duplicated().sum()
movies_df.iloc[0].genres import ast def convert(obj): L=[] for i in
ast.literal_eval(obj): L.append(i['name']) return L
movies_df['genres']= movies_df['genres'].apply(convert)
movies_df['keywords']=
movies_df['keywords'].apply(convert) movies_df.head() def
convert3(obj): L=[] counter = 0 for i in
ast.literal_eval(obj): if counter != 3:
    L.append(i['name'])
counter +=1 else:
    break
return L
movies_df['cast']= movies_df['cast'].apply(convert)
movies_df.head() def fetch_director(obj): L=[]
for i in ast.literal_eval(obj): if
i['job']=='Director':
    L.append(i['name'])
    break
return L
movies_df['crew']=
movies_df['crew'].apply(fetch_director) movies_df.head()
movies_df['overview'][0]
movies_df['overview']=movies_df['overview'].apply(lambda x:x.split()) movies_df
movies_df['genres']=movies_df['genres'].apply(lambda x:[i.replace(" ", "")for i in x])
movies_df['keywords']=movies_df['keywords'].apply(lambda x:[i.replace(" ", "")for i in x])
movies_df['cast']=movies_df['cast'].apply(lambda x:[i.replace(" ", "")for i in x])
movies_df['crew']=movies_df['crew'].apply(lambda x:[i.replace(" ", "")for i in x]) movies_df
movies_df['tags']=movies_df['overview']+movies_df['genres']+movies_df['keywords']+mov
ies_df['cast']+movies_df['crew']
movies_df
new_df = movies_df[['movie_id','title','tags']] new_df
new_df['tags']=new_df['tags'].apply(lambda x:'
.join(x)) new_df new_df['tags'][0]
new_df['tags']=new_df['tags'].apply(lambda X:X.lower()) new_df.head()
from sklearn.feature_extraction.text import CountVectorizer cv
= CountVectorizer(max_features=5000,stop_words='english')

```

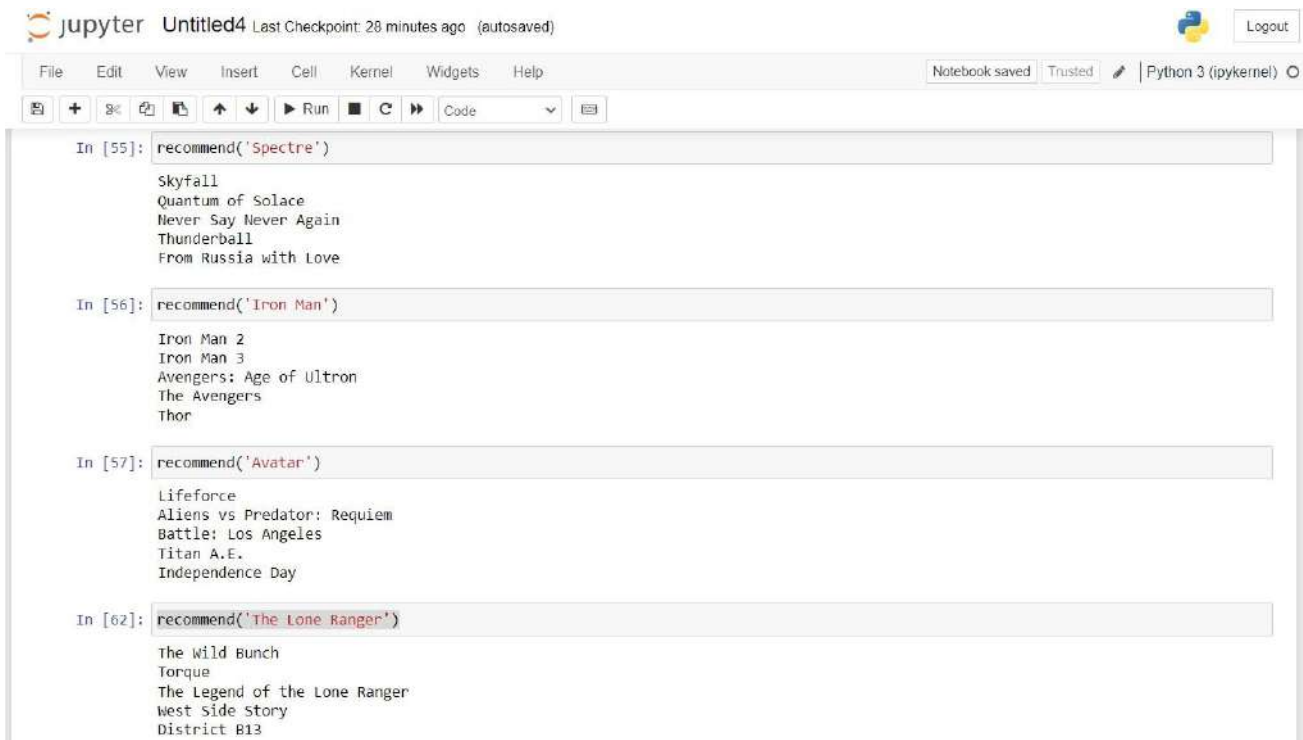
```

cv.fit_transform(new_df['tags']).toarray().shape vectors
= cv.fit_transform(new_df['tags']).toarray() vectors[0]
len(cv.get_feature_names_out()) import
nltk
from nltk.stem.porter import
PorterStemmer ps = PorterStemmer() def
stem(text):
    y=[]    for i in
text.split():
    y.append(ps.stem(i))
    return " ".join(y)
new_df['tags']=new_df['tags'].apply(stem) from sklearn.metrics.pairwise
import cosine_similarity cosine_similarity(vectors)
cosine_similarity(vectors).shape similarity = cosine_similarity(vectors)
similarity[0] similarity[0].shape sorted(list(enumerate(similarity[0])),
reverse=True, key=lambda x:x[1])[1:6] def recommend(movie):
movie_index=new_df[new_df['title']==movie].index[0]
distances=similarity[movie_index]
    movies_list=sorted(list(enumerate(distances)), reverse=True, key=lambda x:x[1])[1:6]

    for i in movies_list:
print(new_df.iloc[i[0]].title)
recommend('Spectre')
recommend('Iron Man')
recommend('Avatar')

```

Results :



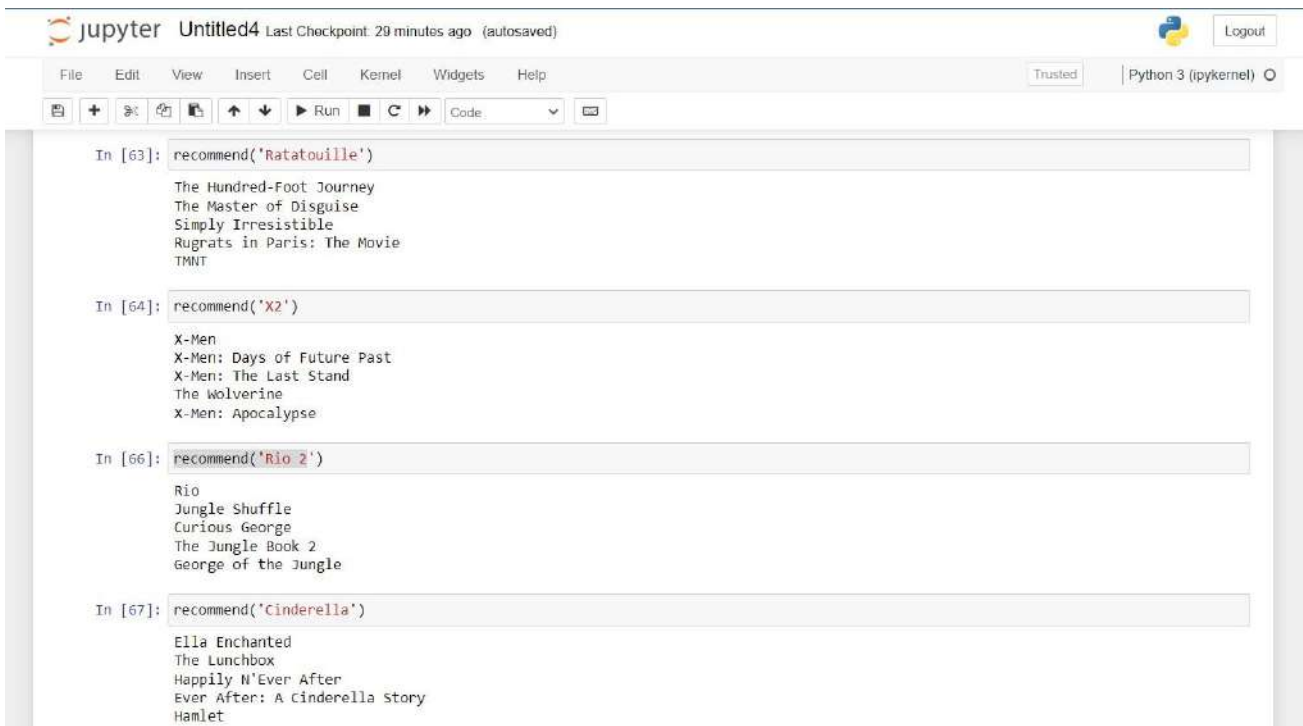
The image shows a Jupyter Notebook interface with the title 'Untitled4' and a status bar indicating 'Last Checkpoint: 20 minutes ago (autosaved)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The notebook contains four code cells, each starting with 'In []:' followed by a 'recommend()' function call. The output of each function is a list of movie titles.

```
In [55]: recommend('Spectre')
Skyfall
Quantum of Solace
Never Say Never Again
Thunderball
From Russia with Love

In [56]: recommend('Iron Man')
Iron Man 2
Iron Man 3
Avengers: Age of Ultron
The Avengers
Thor

In [57]: recommend('Avatar')
Lifeorce
Aliens vs Predator: Requiem
Battle: Los Angeles
Titan A.E.
Independence Day

In [62]: recommend('The Lone Ranger')
The Wild Bunch
Torque
The Legend of the Lone Ranger
West Side Story
District B13
```



The image shows a Jupyter Notebook interface with the title 'Untitled4' and a status bar indicating 'Last Checkpoint: 20 minutes ago (autosaved)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The notebook contains four code cells, each starting with 'In []:' followed by a 'recommend()' function call. The output of each function is a list of movie titles.

```
In [63]: recommend('Ratatouille')
The Hundred-Foot Journey
The Master of Disguise
Simply Irresistible
Rugrats in Paris: The Movie
TMNT

In [64]: recommend('X2')
X-Men
X-Men: Days of Future Past
X-Men: The Last Stand
The Wolverine
X-Men: Apocalypse

In [66]: recommend('Rio 2')
Rio
Jungle Shuffle
Curious George
The Jungle Book 2
George of the Jungle

In [67]: recommend('Cinderella')
Ella Enchanted
The Lunchbox
Happily N'Ever After
Ever After: A Cinderella story
Hamlet
```

CHAPTER-6

5.SYSTEM REQUIREMENTS :

Software Requirement

Anaconda Python
Python 3.6 or higher
Jupyter notebook
Google colab
NumPy
PyInstaller
Pandas
MATLAB (optional)

Hardware Requirement

CPU: Intel Core i5 or higher
RAM: 8GB or higher
Storage: 100GB or higher

Additional Requirements

A stable internet connection (for downloading software packages and datasets) A text editor or IDE for writing Python code

CHAPTER-7

6.CONCLUSION:

In conclusion, a Python-based movie recommendation system is a useful tool for giving customers tailored movie suggestions. The system may examine user preferences and movie attributes to produce exact suggestions by utilizing Python libraries and techniques, such as collaborative filtering and content-based filtering. Python makes it possible to create a strong and dynamic movie recommendation system since it allows for the collection and preprocessing of movie data, the training and evaluation of recommendation models, and the incorporation of user feedback. Overall, Python is a flexible and effective technology for creating movie recommendation systems that improve the user experience when watching movies.

CHAPTER-8

REFERENCES:

1. <https://techvidvan.com/tutorials/movie-recommendation-system-python-machinelearning/>
2. <https://www.scipublications.com/journal/index.php/ijmebac/article/view/291>
3. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9269752/#:~:text=In%20movie%20recommender%20systems%2C%20the%20recommendations%20are%20made%20based%20on,%2C%20and%20ethnicity%20%5B14%5D.>
4. B.M. Sarwar et al., “Application of Dimensionality Reduction in Recommender System—A Case Study,” Proc. KDD Workshop on Web Mining for e-Commerce: Challenges and Opportunities (WebKDD), ACM Press, 2000.
5. S. Funk, “Netflix Update: Try This at Home,” Dec. 2006; <http://sifter.org/~simon/journal/20061211.html>.
6. Y. Koren, “Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model,” Proc. 14th ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining, ACM Press, 2008, pp. 426-434.



Feb 7, 2024

Sudi Srinadh

has successfully completed

Machine Learning with Python

an online non-credit course authorized by IBM and offered through Coursera

**COURSE
CERTIFICATE**



Saeed Aghabozorgi
Sr. Data Scientist
IBM

Joseph Santarcangelo
Senior Data Scientist
IBM

Verify at:
<https://coursera.org/verify/BQ4HJNH935CC>
Coursera has confirmed the identity of this individual and their participation in the course.



DrillBit Similarity Report

5

SIMILARITY %

3

MATCHED SOURCES

A

GRADE

A-Satisfactory (0-10%)

B-Upgrade (11-40%)

C-Poor (41-60%)

D-Unacceptable (61-100%)

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	statusneo.com	3	Internet Data
2	digitalscholarship.unlv.edu	1	Publication
3	User Based Collaborative Filtering using Fuzzy C-Means by Koohi-2016	1	Publication

