Taste Muse-Your Entertainment Companion

A major project report submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

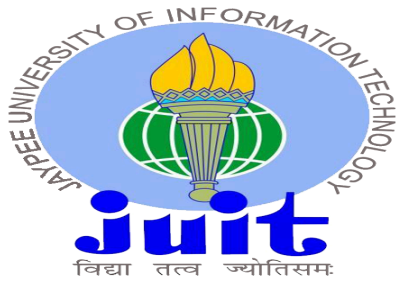
Submitted by

ISHA RAWAT(201337)

ARYAN VERMA(201335)

Under the guidance & supervision of

Dr. Kapil Rana



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

CERTIFICATE

This is to certify that the work which is being presented in the project report titled “Taste Muse-Your Entertainment Companion” in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by “Isha Rawat, 201337”, “Aryan Verma, 201138", during the period from June 2023 to December 2023 under the supervision of Dr. Kapil Rana, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Isha Rawat (201337)

Aryan Verma (201335)

The above statement made is correct to the best of my knowledge.

Dr. Kapil Rana
Assistant Professor
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat

Candidate's Declaration

I hereby declare that the work presented in this report entitled **'Taste Muse-Your Entertainment Companion'** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to December 2023 under the supervision of **Dr. Kapil Rana** (Assistant Professor(SG), Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Isha Rawat
201337
Aryan Verma
201335

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature with Date)
Dr. Kapil Rana
Associate Professor(SG)
Date:

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing to make it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor Dr. Kapil Rana, Assistant Professor SG, Department of CSE,Jaypee University of Information Technology, Waknaghat Deep Knowledge & keen interest of my supervisor in the field of "Recommendation Systems" to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to Dr. Kapil Rana, Department of CSE, for her kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

ISHA RAWAT
(201337)
ARYAN VERMA
(201335)

TABLE OF CONTENTS

ABSTRACT

In today's age of information overload, users are often overwhelmed with entertainment options, making it difficult to find content that suits their interests. To solve this problem, we propose a multi-recommendation system using machine learning algorithms and a complete, user-friendly website that offers personalized recommendations for music, movies and books.

Our system uses advanced machine learning techniques, including clustering and content filtering, to analyze user behavior, preferences and past choices. The system analyzes the patterns and connections in the content of this information and creates recommendations tailored to each user's interests and preferences. The system's end-to-end architecture provides integration and robustness, allowing it to manage a wide customer base and offer responsive and integrated solutions.

User experience. The front end is designed to use modern web technologies to provide users with a visually pleasing and intuitive interface that allows them to easily browse, search and explore content. Through machine learning algorithms and a fully integrated, user-friendly web interface, our versatile recommendations make it easy for users to browse the world, have great fun, and find content that suits their interests.

The system improves the overall experience and increases user engagement and satisfaction by providing personalized recommendations.

Chapter 1: INTRODUCTION

1.1 INTRODUCTION

Customers have an abundance of ways to entertain themselves in the digital age. Users may find it difficult to search for stuff that piques their attention because of this. To solve this challenge, we recommend TasteMuse, a best recommendation that uses advanced technology and comprehensive data to deliver personalized recommendations for books, movies, and music.

TasteMuse uses advanced algorithms to analyze user interactions, preferences and past selections. These algorithms drive engagement and content as a filtering tool, ensuring that each recommendation is tailored to each user's unique tastes.

Additionally, TasteMuse integrates with external APIs to enhance the knowledge base and improve recommendations. TasteMuse has a user-friendly interface that seamlessly delivers personalized recommendations, user data and research functionality across multiple devices. The authentication process is used to protect the user's privacy and personal information.

The success of TasteMuse will be measured by analyzing user interaction and preferences. This data will be utilized to further enhance customer service as well as improve recommending algorithms.

TasteMuse is setting a new benchmark for personalized suggestions & facilitating a more pleasurable experience for all consumers.

1.2 PROBLEM STATEMENT

Users are faced with a confusing assortment of possibilities in the ever-expanding domain of entertainment, ranging from soul-stirring songs to interesting books to compelling flicks. This enormous amount of stuff ought to be enjoyable, but because there are so many options, it frequently causes annoyance and a feeling of alienation. There is a frustrating discrepancy between what users expect & what they are recommended since they struggle to find information that speaks to their individual tastes and preferences.

Traditional methods of content discovery, such as genre-based recommendations and popularity rankings, fall short of providing tailored and satisfying suggestions. These approaches miss the subtleties of individuals as they depend solely on general groups as well as cult tests. Users are consequently frequently inundated with uninteresting or relevant ideas, which discourages them and makes them unhappy with their entertainment experiences.

Innovative solutions that can efficiently speed up the content discovery process are required in light of this gap between users and the entertainment scene. A potential solution to this problem is personalized recommendations, which provide ideas that are specifically catered to the individual likes, tastes, and emotions of each user.Personalised recommendation systems need to surpass conventional approaches and utilise advanced algorithms that can comprehend the intricacies of unique user preferences in order to overcome these obstacles.

1.3 OBJECTIVE

Consumers usually have to undertake the challenging task of surfing through an abundance of options when it comes to entertainment in order to choose the perfect book, movie, or song that fits their preferences and moods. This issue highlights the want for innovative solutions that can facilitate the discovery of content and assist consumers in discovering the entertainment they wish.

To address this critical need, we propose an advanced recommendation system that goes beyond traditional genre-based or popularity-driven suggestions. Our system delves deep into users' historical preferences, analyzing their interactions with music, movies, and books. It uncovers patterns and correlations across these domains, identifying common themes, genres, and emotional undertones that resonate with each user's unique tastes. By understanding these connections, the system can provide personalized recommendations that transcend the boundaries of individual genres, offering a tailored entertainment experience that caters to the user's overall preferences.

The foundation of our system lies in its comprehensive and diverse database of music tracks, movies, and books. Each piece of content is enriched with metadata, including genres, themes, user reviews, and critical acclaim. We collaborate with content providers to ensure the regular addition of new and relevant items, keeping the recommendations fresh, engaging, and up-to-date.

To effectively present personalized recommendations, our system features an intuitive and user-friendly interface that seamlessly integrates with various devices. Personalized playlists, mood-based channels, and thematic collections provide users with a structured and engaging way to discover new favorites. Additionally, the interface incorporates features for personalization and customization, allowing users to tailor their experience to their unique tastes.Our system's success will be evaluated through comprehensive analytics that track user engagement and preferences. This data will be utilized to refine the recommendation algorithms and continuously enhance the overall user experience.

1.4 MOTIVATION

Our motivation for developing this personalized cross-domain recommendation system stems from a deep desire to revolutionize entertainment discovery. By dismantling the barriers between conventional content categories, implementing mood detection methods, and upholding an extensive database, our system seeks to enable users to effortlessly traverse the

wide entertainment landscape, finding content that genuinely speaks to their individual preferences, moods, and patterns. We think that our system has the power to revolutionize how people find and consume entertainment, making everyone's experience more rewarding and pleasurable.

1.5 ORGANIZATION OF THE PROJECT REPORT

The project report is structured to provide a comprehensive understanding of the development and testing processes involved in creating the TasteMuse-Your Entertainment Companion. The organization is designed to present a logical flow of information, guiding readers through the various aspects of the project.

Chapter 1: Introduction

This section introduces the project TasteMuse-Your Entertainment Companion, outlining its objectives, significance, and the technological context..

Chapter 2: Literature Review

A review of relevant literature is presented, exploring existing platforms, recommendation applications, and key technologies. This chapter sets the theoretical framework for the project.

Chapter 3: System Architecture

Detailed information on the architecture of the TasteMuse-Your Entertainment Companion is provided in this chapter. It covers the technological stack, the role of smart contracts, and the overall design of the system.

Chapter 4: Testing

Chapter 4 delves into the testing phase of the project. Subsections include:

4.1 Testing Strategy: A discussion on the overall testing strategy employed, highlighting the tools used and the rationale behind their selection.

4.2 Test Cases and Outcomes: A detailed presentation of test cases for smart contracts, user interfaces, security measures, and performance aspects. Outcomes of each test case are discussed.

Chapter 5: Results and Discussion

This chapter interprets the results obtained during testing. It discusses the implications of the findings, addressing both successes and challenges encountered during the development and testing phases.

Chapter 6: Conclusion

Summarizing the project, Chapter 6 provides key takeaways, lessons learned, and the overall significance of the decentralized social media app. It also hints at potential future developments or improvements.

References

A comprehensive list of references is provided, citing sources and literature that informed the project's development and testing methodologies.

The project report aims to provide a clear and structured narrative, guiding readers through the project's inception, development, testing, and conclusions.

Chapter 2: LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

S. No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
1.	An efficient movie recommendation algorithm	13 december 2020	k-clique detection,collaborative filtering,cosine similarity measure	to have a system that is user friendly and easy to understand and use	accuracy,data privacy,subjective nature,data quality
2.	Movie recommendation using parameter tuning	26 january 2023	co-clustering,machine learning,recommendation system	create a dataset that has all relevant information about a particular movie	cold start,subjective,complexity and scalablity
3.	A Movie Recommender System: MOVREC	volume 124-no 3 , august 2021	data mining, k-mean,collaborative filtering,context based filtering	To have the most appropriate movie recommended list	relevant information,system diversifable,language and location
4.	movie recommender: concepts, method,challenges	13 may 2022	genetic algorithm,k-mean clustering,cuckoo search,principal component analysis	Make the system diversifable so that it can satisfy users from different locations	cold start,scalability,diversity,accuracy,sparsity

SNo.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
5.	survey on movie recommendation system using machine learning	volume 04/ december 2022	collaborative recommender system,hybrid and context based recommendation	The paper collects information we have intensively search free online movies data base	approximate or near-correct recommendation
6.	international journal for research movie recommendation	vol 10 (june 2022)	context based filtering,data analysis,machine learning and score point matching	Including movies in our database irrespective of their language or location so that all users from across the globe can use it	data quality and loss of time in searching
7.	College Library Personalized Recommendation System	11th CIRP Conference (2019)	Collaborative Filtering , Content-Based Filtering, Vector Space Model, , Clustering	The paper assesses the hybrid recommendation system using metrics like accuracy, coverage, and user satisfaction.	Data Sparsity, Cold Start Problem, Generalization
8.	Context-Aware Song Recommendation System Sudipta Chakrabarty	Springer (2020)	Context-Awareness, Recommendation Algorithms, Web-Based Application. User Feedback Analysis	paper covers accuracy, user satisfaction, age preferences, and user feedback's impact on system performance in its results.	Accuracy and Effectiveness, Data Availability, Ethical Considerations

S. No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
9.	Music Recommendation System Based on User's Sentiments Extracted from Social Networks	IEEE Transactions on Consumer Electronics	Sentiment Analysis Tools, Sentiment Intensity Metric (eSM), Usability Framework	The paper presents eSM and correction factor application, system performance metrics, and comparison with baseline recommendation.	Subjective Nature, Generalization, Data Privacy, Complexity, Scope
10.	Personalized Book Recommendation System using Machine Learning Algorithm	IJACSA, Volume 12 Issue 1, 2021	Machine Learning Algorithms, Cosine Similarity, Metrics	The paper evaluates the book recommendation system, favoring specificity over user-based systems, backed by ROC curves and accuracy.	Data quality, model complexity, and scalability
11.	Web-based personalized hybrid book recommendation system	IEEE	recommendation techniques, including collaborative filtering, content-based filtering, demographic filtering, and web scraping	This section reports research findings, including accuracy metrics, user satisfaction, and comparisons with other methods.	Data Quality, New User and Item Problem, Bias, Evaluation Metrics
12.	A Personalized Music Recommendation System Using Convolutional Neural Networks Approach	ICAETR	PMRS). TensorFlow, CNN, collaborative filtering	The paper evaluates PMRS using the CS metric, comparing mean CS values for music recommendations to the original MSD data.	Genre-Specific Performance, Data Enhancement, User Information,

2.2 KEY GAPS IN THE LITERATURE

Paper 1: An Efficient Movie Recommender Algorithm

- Limited exploration of various algorithms and recommendation techniques
- Lack of consideration of user preferences and contextual factors
- Inadequate evaluation of system performance on a large data set

Paper 2: Movie Recommendation Using Parameter Tuning

- Focus on a specific recommendation algorithm (co-clustering)
- Inadequate explanation of parameter tuning process
- Need for a more comprehensive evaluation to assess the generalizability of the proposed approach

Paper 3: Movie Recommendation System: MOVREC

- Limited discussion of system scalability and robustness to large datasets
- Lack of user feedback analysis to understand the effectiveness of recommendations
- Potential for bias in recommendations due to use of k-means clustering

Paper 4: Film Recommendation: Concepts, Method, Challenges

- Too much emphasis on theoretical concepts and less focus on practical aspects of implementation
- Inadequate discussion of trade-offs between different recommendation algorithms
- The need for more empirical studies to validate the proposed methods

Paper 5: Survey on Movie Recommender System Using Machine Learning

- Lack of in-depth analysis of specific recommendation algorithms and their limitations
- Insufficient discussion of the ethical implications of using machine learning for recommender systems
- More research needed to address the cold start problem in movie recommendations

Paper 6: International Journal of Film Research Recommendations

- Limited explanation of the context filtering approach and its implementation
- Lack of evaluation of system performance on a diverse data set
- Insufficient discussion of the challenges of handling large data in context-aware recommender systems

Paper 7: Personalized College Library Recommender System

- Limited discussion of the hybrid recommendation system's ability to handle different types of library resources
- Lack of user feedback analysis to understand the effectiveness of recommendations
- The need for further research on personalized recommendation systems in the context of academic libraries

Paper 8: Sudipta Chakrabarty's Contextual Song Recommender System

- Limited discussion of the ethical implications of collecting and using user data in context-sensitive recommendation systems
- Lack of evaluation of the system's performance on a diverse dataset of music genres and preferences
- Further research needed to address data availability issues in context-aware music recommendation systems

Paper 9: A Music Recommender System Based on User Sentiments Obtained from Social Networks

- Limited discussion of the effectiveness of eSM and the correction factor in accurately capturing user sentiment.
- Lack of evaluation of the system's performance on a diverse dataset of music genres and preferences.
- Inadequate consideration of the subjective nature of feelings and their impact on the accuracy of recommendations.

Paper 10: Personalized Book Recommender System Using Machine Learning Algorithm

- Overemphasis on specificity over user systems, potentially limiting the system's ability to capture different user preferences.
- Lack of exploration of different machine learning algorithms and their suitability for book recommendations.
- Insufficient discussion of the challenges of handling large book data sets and maintaining data quality.

Paper 11: A Web-Based Personalized Hybrid Book Recommender System

- Focus on the implementation of the referral system rather than comprehensive evaluation of its performance.
- Limited discussion of the effectiveness of a hybrid approach combining different recommendation techniques.
- Lack of user feedback analysis to understand the effectiveness of recommendations.

Paper 12: A Personalized Music Recommender System Using a Convolutional Neural Network Approach

- Overemphasis on Convolutional Neural Networks (CNN) approach without considering other deep learning architectures.
- Limited discussion of the problems of processing recommendations for a specific genre of music.
- Lack of exploration of techniques to improve the dataset and incorporate user information to improve recommendation accuracy.

Chapter 3: SYSTEM DEVELOPMENT

3.1 REQUIREMENTS AND ANALYSIS

3.1.1 OBJECTIVE

The objective of the Recommendation System is to not only offer personalized recommendations but also create an immersive and engaging experience for users across a wide range of entertainment mediums. By understanding and adapting to individual preferences, the ERS seeks to become a go-to platform that not only suggests content but also fosters a sense of discovery and enjoyment.

3.1.2 Functional Requirements

3.1.2.1 User Registration and Authentication

Users should have the ability to create accounts securely and unlimited.Authentication mechanisms like two-factor authentication should be implemented to ensure the integrity/security of user accounts.

3.1.2.2 Profile Creation with User Preferences

The system should allow users to create detailed profiles, including preferences related to genres, artists, authors, and other relevant parameters. Continuous updating of profiles based on user interactions ensures up-to-date and accurate recommendations.

3.1.2.3 Recommendation Engine for Movies, Books, and Music

The heart of the system lies in a robust recommendation engine. This engine employs a combination of collaborative and content-based filtering techniques, machine learning algorithms, and deep learning models to analyze user behavior and provide accurate and diverse recommendations.

3.1.2.4 Search Functionality with Filters

To enhance user autonomy, a sophisticated search functionality is implemented. Users should be able to search for specific titles, genres, or artists and apply filters to refine results based on parameters such as release date, popularity, or user ratings.

3.1.2.5 User Feedback and Rating System

An integral part of the system is the ability for users to provide feedback and ratings for the recommended content. This not only refines subsequent recommendations but also fosters a sense of user engagement and community.

3.1.2.6 Integration with External APIs for Content Information

To ensure a comprehensive and up-to-date database, the system integrates with external APIs from platforms like IMDb, OpenLibrary, and Spotify. This integration facilitates the continuous influx of new content information, ensuring that users have access to the latest entertainment options.

3.1.3 Non-functional Requirements

3.1.3.1 Response Time

The system places a high priority on responsiveness. Real-time recommendations, with response times within seconds, are crucial to providing a seamless and enjoyable user experience. Load balancing and optimization techniques are employed to maintain low latency even during peak usage.

3.1.3.2 Scalability

As the user base and content library grow, the system should seamlessly scale to accommodate increased demand. Cloud-based infrastructure and horizontal scaling practices are implemented to ensure that the system remains performant and responsive.

3.1.3.3 Security

Protecting user data is paramount. The system employs encryption protocols to secure data during transmission and storage. Regular security audits, intrusion detection systems, and adherence to industry-standard security practices safeguard user information, preventing unauthorized access and data breaches.

Filters

User Feedback and Rating System
Integration with External APIs for Content Information

3.1.3 Non-functional Requirements

Response time: The system should provide recommendations in real-time, aiming for response times within seconds.

Scalability: The system should be scalable to accommodate a growing user base and expanding content database.
Security: User data must be stored securely, and authentication mechanisms should be robust to prevent unauthorized access.

3.2 PROJECT DESIGN AND ARCHITECTURE

3.2.1 SYSTEM ARCHITECTURE

The ERS follows a microservices architecture, with components dedicated to user management, recommendation engine, content database, and external API integrations. The architecture ensures modularity and flexibility.

3.2.2 DATABASE DESIGN

A relational database is employed to store user profiles, preferences, and content information. The design includes tables for users, preferences, and a separate table for each type of entertainment (movies, books, music).

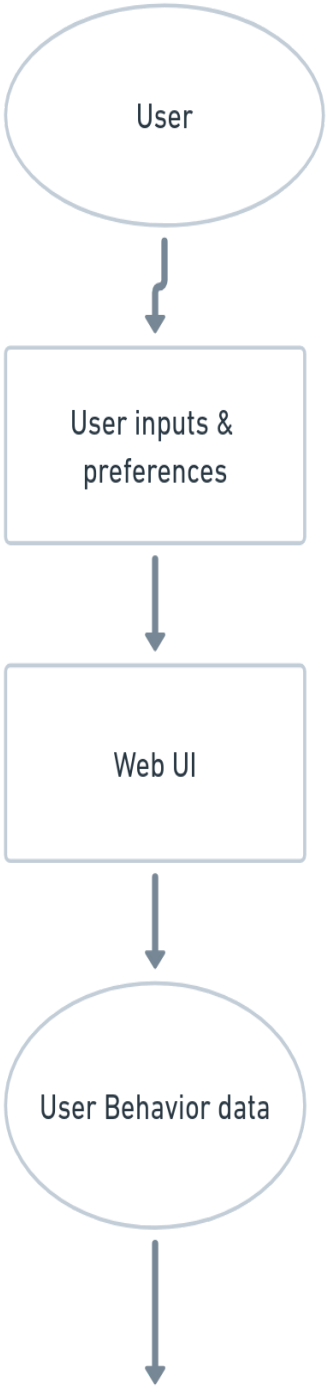


Fig-3.1:Model design

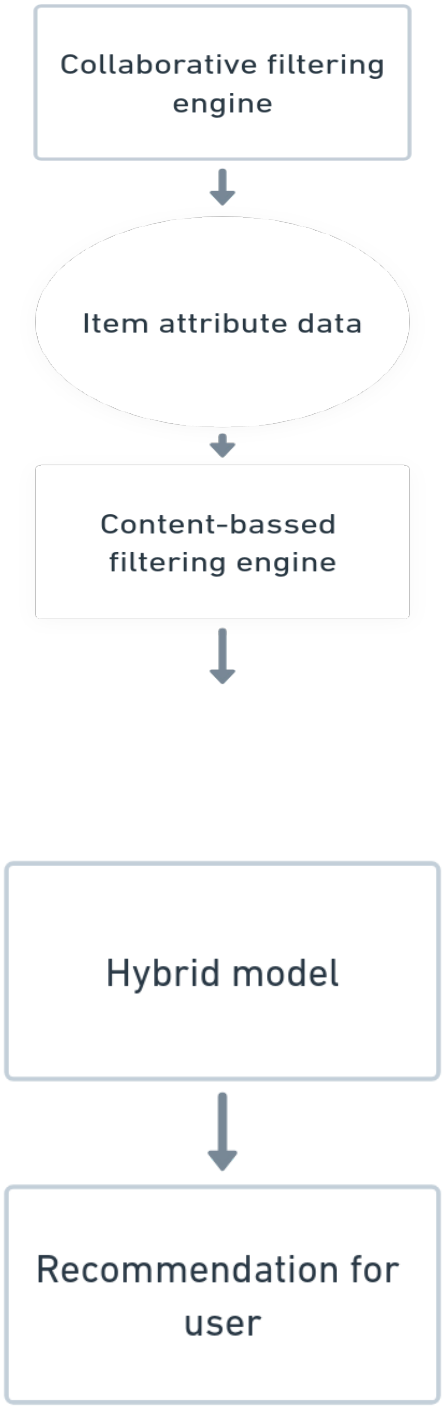


Fig-3.2:Model design continued

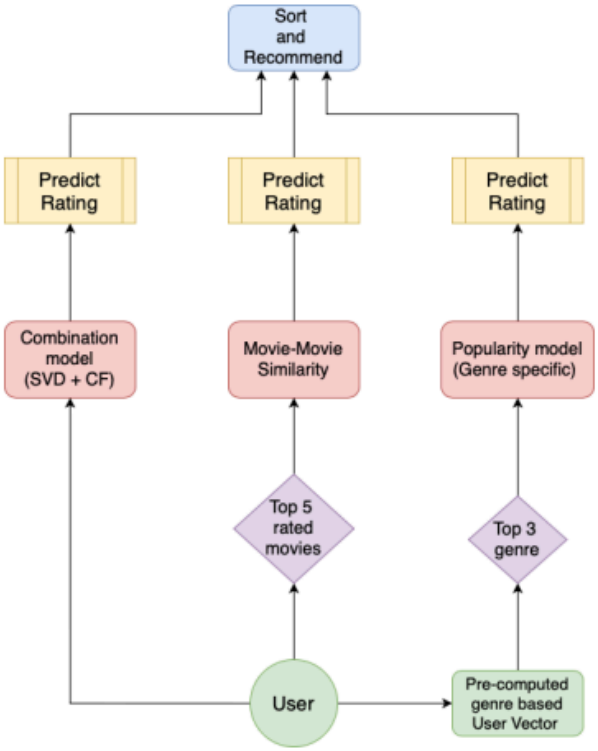


Fig-3.3: Hybrid model design

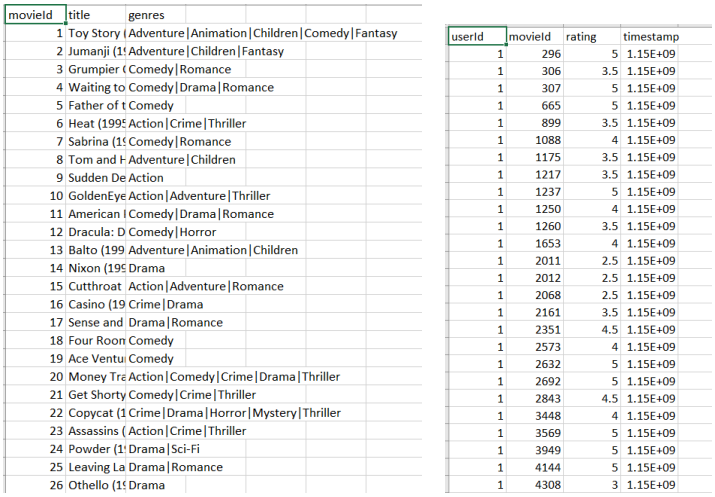
3.3 DATA PREPARATION

3.3.1 DATA COLLECTION

Content data is collected from various sources, including public APIs for movies, books, and music. User data is gathered during the registration process and supplemented with feedback and ratings.

3.31 DATA PREPARATION

Concerning the Data: The Grouplen film evaluation dataset is the primary repo for data for the study. The MovieLens suggestion service's five-star evaluation and free-text labeling activities are described in the smaller dataset (ml-latest small). 283,228 individuals have contributed 27,753,444 reviews of 58,098 distinct films to the dataset. Every rating has a number between 0 and 5. The review's timestamp, the film's genre, user-generated keywords in comments, and the associated movie's IMDB and TMDB IDs are only a few of the dataset's characteristics.



movieId	title	genres
1	Toy Story (	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1!	Adventure|Children|Fantasy
3	Grumpier (	Comedy|Romance
4	Waiting to	Comedy|Drama|Romance
5	Father of t	Comedy
6	Heat (199!	Action|Crime|Thriller
7	Sabrina (1!	Comedy|Romance
8	Tom and H	Adventure|Children
9	Sudden De	Action
10	GoldenEye	Action|Adventure|Thriller
11	American I	Comedy|Drama|Romance
12	Dracula: D	Comedy|Horror
13	Balto (199	Adventure|Animation|Children
14	Nixon (199	Drama
15	Cutthroat	Action|Adventure|Romance
16	Casino (19	Crime|Drama
17	Sense and	Drama|Romance
18	Four Roon	Comedy
19	Ace Ventu	Comedy
20	Money Tra	Action|Comedy|Crime|Drama|Thriller
21	Get Shorty	Comedy|Crime|Thriller
22	Copycat (1	Crime|Drama|Horror|Mystery|Thriller
23	Assassins (	Action|Crime|Thriller
24	Powder (1!	Drama|Sci-Fi
25	Leaving La	Drama|Romance
26	Othello (1!	Drama

userId	movieId	rating	timestamp
1	296	5	1.15E+09
1	306	3.5	1.15E+09
1	307	5	1.15E+09
1	665	5	1.15E+09
1	899	3.5	1.15E+09
1	1088	4	1.15E+09
1	1175	3.5	1.15E+09
1	1217	3.5	1.15E+09
1	1237	5	1.15E+09
1	1250	4	1.15E+09
1	1260	3.5	1.15E+09
1	1653	4	1.15E+09
1	2011	2.5	1.15E+09
1	2012	2.5	1.15E+09
1	2068	2.5	1.15E+09
1	2161	3.5	1.15E+09
1	2351	4.5	1.15E+09
1	2573	4	1.15E+09
1	2632	5	1.15E+09
1	2692	5	1.15E+09
1	2843	4.5	1.15E+09
1	3448	4	1.15E+09
1	3569	5	1.15E+09
1	3949	5	1.15E+09
1	4144	5	1.15E+09
1	4308	3	1.15E+09

Fig-3.4: Movies Dataset containing names and rating

The below mentioned code contains descriptors used to create HDF5 files
for the Million Song Database Project. What information gets in the database should be decided here.

This is part of Million Songs Dataset project from Lab ROSA (Columbia University) and The Echo Nest.

```
import os
import sys
import glob
import time
import datetime
import hdf5_utils as HDF5
import utils


def die_with_usage():
    """ HELP MENU """
    print 'create_aggregate_file.py'
    print '   by T. Bertin-Mahieux (2011) Columbia University'
    print '   tb2332@columbia.edu'
    print ''
    print 'Creates an aggregate file from all song file (h5 files)'
    print 'in a given directory.'
    print 'Aggregate files contains many songs. and none of the arrays,'
    print ''
    print 'usage:'
    print '   python create_aggregate_file.py <H5 DIR> <OUTPUT.h5>'
    print 'PARAMS'
    print '   H5 DIR     - directory contains h5 files (subdirs are checked)'
    print '   OUTPUT.h5  - filename of the aggregate file to create'
    sys.exit(0)


    MAXSTRLEN = 1024

    class SongMetaData(tables.IsDescription):
        """
        Class to hold the metadata of one song
        """
        artist_name = tables.StringCol(MAXSTRLEN)
        artist_id = tables.StringCol(32)
        artist_mbid = tables.StringCol(40)
        artist_playmeid = tables.IntCol()
        artist_7digitalid = tables.IntCol()
        analyzer_version = tables.StringCol(32)
        genre = tables.StringCol(MAXSTRLEN)
        release = tables.StringCol(MAXSTRLEN)
        release_7digitalid = tables.IntCol()
        title = tables.StringCol(MAXSTRLEN)
        artist_familiarity = tables.Float64Col()
        artist_hotttnesss = tables.Float64Col()
        song_id = tables.StringCol(32)
        song_hotttnesss = tables.Float64Col()
        artist_latitude = tables.Float64Col()
        artist_longitude = tables.Float64Col()
        artist_location = tables.StringCol(MAXSTRLEN)
        track_7digitalid = tables.IntCol()
        # ARRAY INDICES
        idx_similar_artists = tables.IntCol()
        idx_artist_terms = tables.IntCol()
```

Fig-3.5: Creation of music dataset

3.32 DATA PREPROCESSING:

• Based on User ID, we divided the data set into training and testing subsets. This way, 80 percent of every user's ratings were utilized for training the model, and the other twenty percent were utilized to confirm that our model was accurate. The train test split function in

the sci-kit-learn library is used to do the aforementioned. The feature on which the dataset is split into training and testing subsets is specified by the function's stratify attribute.

• By integrating the data set with the publicly accessible IMDB and TMDB datasets, we were able to obtain helpful data about the movie, including its release date, writer, director, and cast credits, tagline, synopsis, popularity, rating, and vote counts on their respective websites. This allowed us to make more informed recommendations about content and the significance of movie features.

Removing redundant and non-contributing features from the dataset was one step in the data-cleaning process. Genres and additional classification attributes are converted to one-hot vectors. Additionally, as screening requires a minimum of one data point, we rejected any users without fewer that 5 data points.

```
test.to_csv('melon/visual_feats.tsv', header=None, sep='\t', index=None)
print('End')

print('Time Splitting')
import datetime

years = [0, 2015, 2020]  # 0 is the minimum placeholder
for i, year in enumerate(years):
    if year != 0:
        print('Extract Data for {0}'.format(year))
        dict_tmp = dict()
        with open('melon/data_{0}.tsv'.format(year), 'w') as output_file:
            for filename in ['train', 'val', 'test']:
                with open('original_dataset/{0}.json'.format(filename)) as json_file:
                    data = json.load(json_file)
                    j = 0
                    for playlist in data:
                        date_time_obj = datetime.datetime.strptime(playlist['updt_date'], '%Y-%m-%d %H:%M:%S.%f')
                        if years[i - 1] < date_time_obj.year <= years[i]:
                            j += 1
                            if j % 1000 == 0:
                                print(j)
                            for song in playlist['songs']:
                                if playlist['id'] not in dict_tmp:
                                    dict_tmp[playlist['id']] = -1
                                relative_timestamp = dict_tmp[playlist['id']] + 1
                                output_file.write(
                                    "{0}\t{1}\t{2}\t{3}\n".format(playlist['id'], song, 1, relative_timestamp))
                                dict_tmp[playlist['id']] = relative_timestamp
        print('End Extract Data for {0}'.format(year))
```

Fig-3.6.1: Data Processing for muisc dataset

```
print("Columns: ", list(books.columns))
books.head()
```

```
Columns:  ['ISBN', 'Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher', 'Image-URL-S', 'Image-URL-M', 'Image-URL-L']
```

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher	Image-URL-S	
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.com/images/P/0195153448.0...	http://images.amazon.co
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.com/images/P/0002005018.0...	http://images.amazon.co
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.amazon.com/images/P/0060973129.0...	http://images.amazon.co
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	http://images.amazon.com/images/P/0374157065.0...	http://images.amazon.co
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	http://images.amazon.com/images/P/0393045218.0...	http://images.amazon.co

Fig-3.6.2: Data Processing for books dataset

In [ ]:
```
!pip install surprise
```

In [ ]:
```
import pickle
import os

import pandas as pd

from surprise import SVD, SVDpp
from surprise import KNNBasic, KNNBaseline, BaselineOnly
from surprise import Dataset
from surprise import Reader
from surprise import dump
from surprise.accuracy import rmse
```

In [ ]:
```
def convert_traintest_dataframe_forsurprise(training_dataframe, testing_dataframe):
    reader = Reader(rating_scale=(0, 5))
    trainset = Dataset.load_from_df(training_dataframe[['userId', 'movieId', 'rating']], reader)
    testset = Dataset.load_from_df(testing_dataframe[['userId', 'movieId', 'rating']], reader)
    trainset = trainset.construct_trainset(trainset.raw_ratings)
    testset = testset.construct_testset(testset.raw_ratings)
    return trainset, testset
```

In [ ]:
```
file_path_train = 'training_data.csv'
file_path_test = 'testing_data.csv'
traindf = pd.read_csv(file_path_train)
testdf = pd.read_csv(file_path_test)
trainset, testset = convert_traintest_dataframe_forsurprise(traindf, testdf)
```

Fig-3.6.3: Testing and Training of dataset

3.33 DATA ANALYSIS

- Genre Distribution of Movies

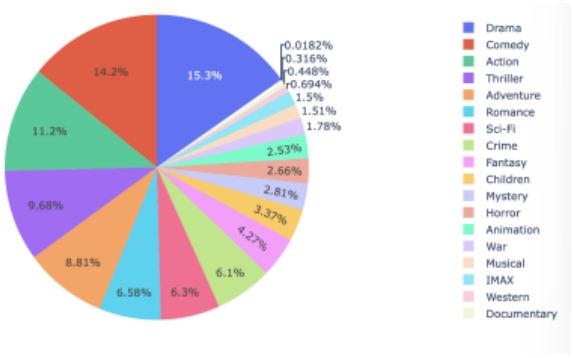


Fig-3.7: Genre distribution in training data

- Average of number of user ratings per movie = 478
- Average of number of movies rated by user = 97
- User Rating Distribution method

Item based:

There is a bottleneck in user-based CF algorithms. User-based CF algorithms suffer from serious scalability problems [23]. The problem depicts that the computations are expensive in real-world systems where there are potentially millions of users and items. This is how item-based algorithms are introduced to overcome the bottleneck by finding relationships between items rather than users. Item-based CF algorithms [4] are based on the intuition that users are interested in items similar to those previously experienced. The item-based implementation goes roughly the same process as the user-based one, namely, the four steps introduced in Section 3.1.

The step of data representation is the same. In similarity computation, the pairwise similarities are calculated based on items instead of users. For example, in the cosine similarity approach, items are represented as vectors in the N-dimentional user space and the similarity between item a and j is calculated as the cosine of the angle between

the vector of item a and the vector of item j. As a result, after this step, there will be a data structure produced to store the similarities between items rather than users. In neighbourhood formation, an item neighbourhood is formed by taking a subset of other items with the highest similarity to the active item. In making a prediction, taking the Resnisk's algorithm as an example, the prediction equation 3.3 is changed to

$$pa,j = vj + Pk\ i=1\ si,j\ (ra,i − vi)\ Pk\ i=1\ |si,j\ |$$

(3.4) where si,j is the similarity score between item i and j, vj refers to the mean rating of item j across all users, ra,i refers to the rating of user a for item i and k is the size of neighbourhood. The process of generating a top-N list of recommendation works the same way as the user-based as described in Section 3.1.4. 3.3

Matrix Factorisation There are some limitations in user-based and item-based CF algorithms. First, neighbourhoodbased algorithms heavily depend on the relationships between users and items in order to form predictions or recommendations. This indicates that a limited quantities of data cause it hard Page 20 of 40 to build the relationship model (the similarities between neighbours).

This is so called the data sparsity problem [50]. Second, they suffer from the early rater problem [51]. This issues describes that predications are hard made for items that are newly added to a system as the items are rarely rated in the system (not enough ratings to calculate predictions). It also depicts that predictions or recommendations are hard made for users who newly registered as they do not show enough their taste in the system (not neighbours).

Due to these limitations, the model-based approaches are introduced. There are many wellestablished model-based approaches, among which the matrix factorisation [5, 39] approach is widely-used. It works by making an assumption that there exist some latent features which explain user preferences. Its goal is to uncover these latent features which can afterward be used to predict items for a user more accurately [29]. It starts

with initializing the number of features K and two matrices P (|U| × K matrix) and Q (|I| × K matrix),

which represents the strength of the association association between a user (P) or an item (Q) and each of the K features. Then the multiplication of P and QT is calculated as the prediction matrix in which the predicted ratings are presented. The following graph gives a nice illustration of how the matrics are constructed.

Figure 3.1: Illustration of matrix factorisation whose goal is to find the predicted rating matrix Rˆ by learning user and item feature values With P and Q calculated, a predicted rating for user i, item j is given by the dot product of the two vectors corresponding to user i and item j, namely

$$\hat{r}_{i,j} = p_T i\, q_j = PK\ k=1\ pikqkj \backslash.$$

In order to approximate the actual user-item matrix, iterations are assigned to update P and Q and the difference, namely the loss function is given by the error between the predicted and actual ratings. There are many ways to compute the error. One example is square error loss [18]. The following equation specifies the square error over all ratings in the training set

$$T.\ E = X\ r_{i,j} \in T\ e\ 2\ i,j = X\ r_{i,j} \in T\ (r_{i,j} - X\ K\ k=1\ pikqkj\ )$$

2 (3.5) After obtaining the loss function, a decision needs to be made upon how to modify P or Q so that the error can be minimised. Many approaches have been proposed to achieve this goal, for example, the stochastic gradient descend (SGD) [40] introduced by Simon Funk1 . It works by updating the values of pik and qkj according to the direction of the gradient. For each factor, it updates the values of pik and qkj each time using update rules by looping through all each rating 1 http://sifter.org/simon/ Page 21 of 40 in the training set T until the error is minimised. Equation 3.6 gives the update rules for both pik and qkj . $p\ 0\ ik = pik + \alpha\ \delta\ \delta pik\ e\ 2\ i,j = pik + 2\alpha ei,jqkj\ q\ 0\ kj = qkj + \alpha\ \delta\ \delta qkj\ e\ 2\ i,j = qkj + 2\alpha ei,jpik$

(3.6) where α is the learning rate normally given by 0.001. With the loss function and update rules (SGD),

the difference between the prediction matrix and the actual user-item matrix is minimised. The above describes a basic process of the implementation of matrix factorisation techniques.
In literature, there are many variations to implement matrix factorisation algorithms.

For example, the approach called linear matrix factorisation(LMF) proposed by [19, 5] works by adding two bias constants and regularisation parameters to the prediction model so that the overfitting problem and latent user biases can be avoided. In this approach, the predicted rating of user i for item j is changed to

$$\hat{r}_{i,j} = p^T_i q_j = \sum_{k=1}^{K} p_{ik} q_{kj} + c_i + d_j$$

(3.7) where ci and dj are two constants representing the bias of user i and item j respectively. It is noteworthy that the two constants need to be trained together with pik and qkj . The update rules for ci and dj are specified in Equation 3.8. $c'_i = c_i + \alpha(\delta \delta c_i e^2_{i,j} - \sigma c_i) = c_i + \alpha(2e_{i,j} - \sigma c_i)$ $d'_j = d_j + \alpha(\delta \delta d_j e^2_{i,j} - \sigma d_j) = d_j + \alpha(2e_{i,j} - \sigma d_j)$ (3.8) where σ is the regularisation parameter for alleviating the overfitting problem normally given by 0.01. With the bias constants ci and dj and the

regularisation parameter added, now the error function become

$$E = \sum_{r_{i,j} \in T} e^2_{i,j} = \sum_{r_{i,j} \in T} ((r_{i,j} - \sum_{k=1}^{K} p_{ik} q_{kj} + c_i + d_j)^2 + \beta^2 (\|p_k\|^2 + \|q_k\|^2) + \sigma^2 (c^2_i + d^2_j))$$ (3.9)

where β is an additional regularisation factor used to suppress overtraining and therefore improve performance on unseen examples [5]. They are normally given by 0.01.
With the loss function and update rule, the same process as introduced earlier in the section is applied to train the predicted matrix $\hat{R}$ as closest as possible to the actual prediction model by learning the user-item matrix R.

The LMF algorithm as a variation of matrix factorisations is implemented in this project, but renamed to biased matrix factorisation(BMF).

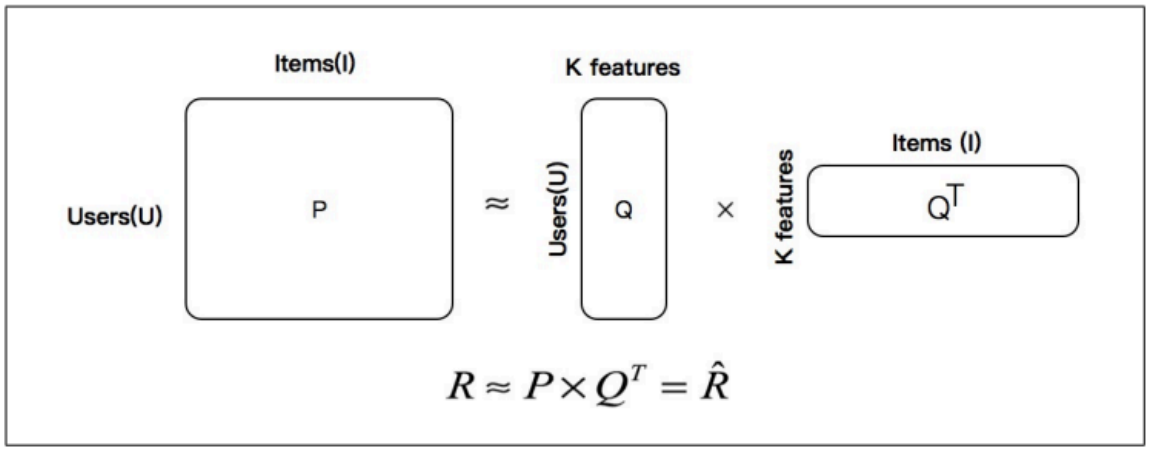


Fig-3.7.1: Illustration of matrix factorisation whose goal is to find the predicted rating matrix Rˆ by learning user and item feature values

3.4 IMPLEMENTATION

3.4.1 Technologies Used

- Programming Languages: Python (Flask for backend, React for frontend)
- Database: PostgreSQL
- Machine Learning: Collaborative Filtering Algorithms (e.g., Matrix Factorization)
- API Integrations: IMDb, OpenLibrary, Spotify

3.4.2 CODE SNIPPET (COLLABORATIVE FILTERING)

```
]: print("--------------------------RMSE----------------------")
   print("KNN Basic                ",df_knn[df_knn.Iu > 1000].sqr_err.mean()** .5)
   print("SVD                      ", df_svd[df_svd.Iu > 1000].sqr_err.mean()** .5)
   print("SVDpp                    ",  df_svdpp[df_svdpp.Iu > 1000].sqr_err.mean()** .5)
   print("KNN Baseline (item-item) ", df_knnbaseline[df_knnbaseline.Iu > 1000].sqr_err.mean()** .5)
   print("BaselineOnly             ",df_baselineonly[df_baselineonly.Iu > 1000].sqr_err.mean()** .5 )
   print("KNN Baseline (user-user) ",df_knn_user[df_knn_user.Iu > 1000].sqr_err.mean()** .5)

--------------------------RMSE----------------------
KNN Basic                0.9174613388905646
SVD                      0.8207944406250214
SVDpp                    0.8136491891525117
KNN Baseline (item-item)  0.789275629286978
BaselineOnly             0.799990922710614
KNN Baseline (user-user)  0.8198697577732832

]: iid_df = traindf.groupby(['userId'],as_index=False).movieId.count()
   iid_df.movieId.max()

]: 2158
```

Fig-3.8: Applying SVD and KNN

```
1   from surprise import SVD, BaselineOnly, SVDpp, NMF, SlopeOne, CoClustering, Reader
2   from surprise import Dataset
3   from surprise.model_selection import cross_validate
4   from surprise.prediction_algorithms import KNNBaseline, KNNBasic, KNNWithMeans, KNNWithZScore
5   from surprise import accuracy
6   from surprise.model_selection import train_test_split
7
8   import pandas as pd
9   import numpy as np
10
11
12  def convert_traintest_dataframe_forsurprise(training_dataframe, testing_dataframe):
13      reader = Reader(rating_scale=(0, 5))
14      trainset = Dataset.load_from_df(training_dataframe[['userId', 'movieId', 'rating']], reader)
15      testset = Dataset.load_from_df(testing_dataframe[['userId', 'movieId', 'rating']], reader)
16      trainset = trainset.construct_trainset(trainset.raw_ratings)
17      testset = testset.construct_testset(testset.raw_ratings)
18      return trainset, testset
19
20
21  def recommendation(algo, trainset, testset):
22      # Train the algorithm on the trainset, and predict ratings for the testset
23      algo.fit(trainset)
24
```

Fig-3.9: Predicting ratings

```
                                'baseline_rating'])
        df_svd = pd.DataFrame([[svd.uid, svd.iid, svd.r_ui, svd.est]],
                              columns=['uid', 'iid', 'og_rating', 'est_rating'])
        df_svdpp = pd.DataFrame([[svd.uid, svd.iid, svd.r_ui, svdpp.est]],
                                columns=['uid', 'iid', 'og_rating', 'est_rating'])
        df_knnb = pd.DataFrame([[svd.uid, svd.iid, svd.r_ui, knn.est]],
                               columns=['uid', 'iid', 'og_rating', 'est_rating'])
        df_slope = pd.DataFrame([[svd.uid, svd.iid, svd.r_ui, slopeone.est]],
                                columns=['uid', 'iid', 'og_rating', 'est_rating'])
        df_bonly = pd.DataFrame([[svd.uid, svd.iid, svd.r_ui, baseline.est]],
                                columns=['uid', 'iid', 'og_rating', 'est_rating'])
        test_pred_df = pd.concat([df, test_pred_df], ignore_index=True)
        test_svd_df = pd.concat([df_svd, test_svd_df], ignore_index=True)
        test_svdpp_df = pd.concat([df_svdpp, test_svdpp_df], ignore_index=True)
        test_slope_df = pd.concat([df_slope, test_slope_df], ignore_index=True)
        test_knnb_df = pd.concat([df_knnb, test_knnb_df], ignore_index=True)
        test_bonly_df = pd.concat([df_bonly, test_bonly_df], ignore_index=True)

print("7")
test_pred_df.to_csv('test_prediction_HP.csv')
test_svd_df.to_csv('test_predictions_svd.csv')
test_svdpp_df.to_csv('test_predictions_svdpp.csv')
test_knnb_df.to_csv('test_predictions_knnb.csv')
test_slope_df.to_csv('test_predictions_slope.csv')
test_bonly_df.to_csv('test_predictions_bonly.csv')
```

Fig-3.10 Testing

```
#Read userid-songid-listen_count
song_info = pd.read_csv('https://static.turi.com/datasets/millionsong/10000.txt',sep='\t',header=None)
song_info.columns = ['user_id', 'song_id', 'listen_count']

#Read song  metadata
song_actual =  pd.read_csv('https://static.turi.com/datasets/millionsong/song_data.csv')
song_actual.drop_duplicates(['song_id'], inplace=True)

#Merge the two dataframes above to create input dataframe for recommender systems
songs = pd.merge(song_info, song_actual, on="song_id", how="left")
```

```
songs.head()
```

Fig-3.11: Reading and merging of two dataframes

```
plt.figure()
sns.barplot(x=counts, y=labels, palette='Set3')
sns.despine(left=True, bottom=True)
```

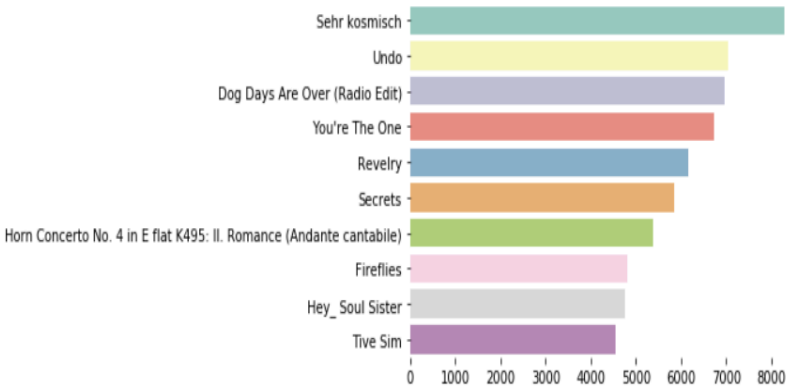


Fig-3.12: Popular songs

```
plt.figure()
labels = ten_pop_artists['artist_name'].tolist()
counts = ten_pop_artists['listen_count'].tolist()
sns.barplot(x=counts, y=labels, palette='Set2')
sns.despine(left=True, bottom=True)
```

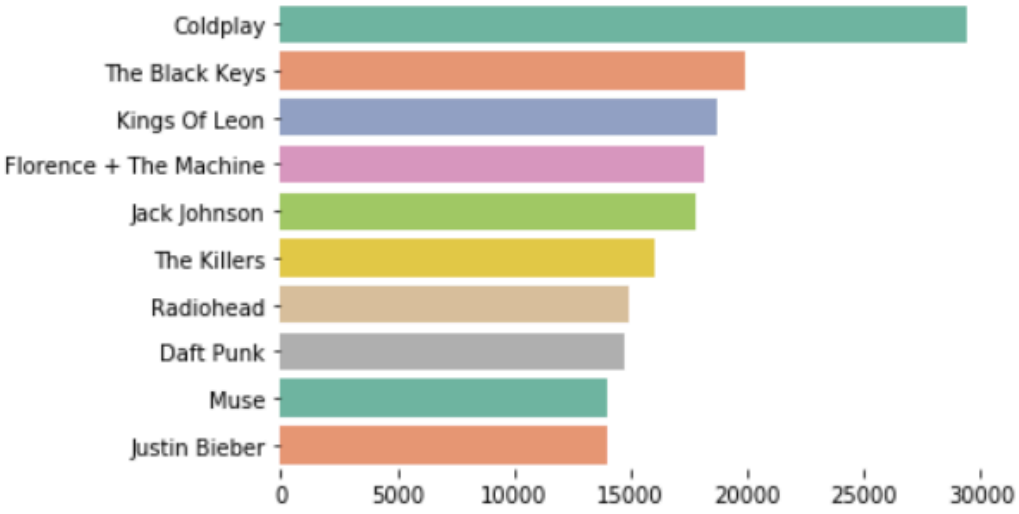


Fig-3.13: Popular artists

```
plt.figure(figsize=(16, 8))
sns.barplot(x='listen_count', y='count', palette='Set3', data=listen_counts)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.show();
```

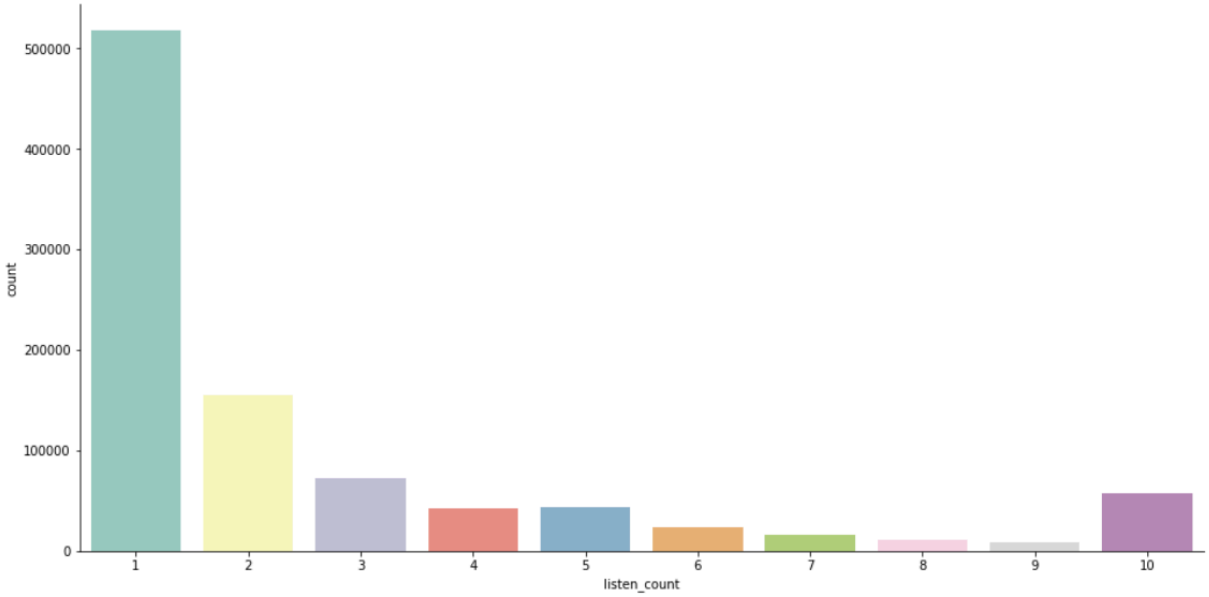


Fig-3.14: Range of rating

```
# find the best parameters for the model
grid_search_svd.fit(data)
find_algo = grid_search_svd.best_estimator['rmse']
print(grid_search_svd.best_score['rmse'])
print(grid_search_svd.best_params['rmse'])
```

```
# Perform the cross validation
cross_validate(find_algo, data, measures=['RMSE'], cv=5, verbose=True)
```

After finding the best parameters for the model, we create our final model, train it and find the error for the test set.

```
final_algorithm = SVD(n_factors=160, n_epochs=100, lr_all=0.005, reg_all=0.1)
final_algorithm.fit(trainset)
test_predictions = final_algorithm.test(testset)
print(f"The RMSE is {accuracy.rmse(test_predictions, verbose=True)}")
```

```
SVD : Test Set
RMSE: 2.1846
2.184647808424269
```

Fig-3.15: Model and recommendations

3.5 KEY CHALLENGES

3.5.1 Cold Start Problem:

Challenge:

Generating relevant recommendations for new users with minimal interaction history poses a significant challenge as the system lacks sufficient data to understand their preferences.
Solution:

1. Content-based filtering:
    - Initially, the system relies on analyzing the content itself, such as keywords and genres, to provide recommendations. This approach offers a starting point for recommendations while the system gathers data to build the user's profile.
2. Gradual transition:
    - As users interact with the system by providing ratings and reviews, their data is incorporated into the recommendation algorithm. This gradual transition from content-based to collaborative filtering allows the system to shift its focus towards more personalized recommendations based on user preferences.

3.5.2 Scalability:

Challenge:
Maintaining fast response times becomes increasingly difficult as the user base and data volume grow, potentially leading to performance issues.

Solution:

1. Caching mechanisms:
    - Frequently accessed data is stored in caches to facilitate faster retrieval, reducing the load on the database and improving response times. This optimization ensures that users experience minimal latency when accessing recommendations or other system features.
2. Database optimization:
    - Queries are carefully analyzed and optimized to minimize execution time and resource usage. By fine-tuning database queries, the system can efficiently handle increasing data volumes without sacrificing performance, ensuring scalability as the user base grows.

3.5.3 Diversity in Recommendations:

Challenge:

Providing diverse recommendations is crucial to catering to users with broad interests spanning multiple genres or topics.

Solution:

1. Hybrid approach:

- The system adopts a hybrid approach that combines collaborative filtering and content-based filtering techniques.
- Collaborative filtering recommends items based on the preferences of similar users, ensuring personalized recommendations within a user's preferred genres.
- Content-based filtering recommends items based on individual user preferences, introducing users to diverse content outside their usual choices.

By integrating both approaches, the system can strike a balance between personalized recommendations and exposure to new and diverse content, enhancing the overall user experience.

These solutions demonstrate the adaptability and evolution of the ERS to address emerging challenges effectively. By tackling the cold start problem, ensuring scalability, and promoting recommendation diversity, the ERS aims to provide a continuously improving user experience that caters to the evolving needs and preferences of its users.

CHAPTER 4: TESTING

4.1 TESTING STRATEGY

4.1.1 OVERVIEW

The testing strategy for the Entertainment Recommendation System (ERS) is comprehensive, covering various aspects from functionality to performance. Multiple testing stages are implemented throughout the development lifecycle to identify and rectify issues early on. The following tools and strategies were employed:

4.1.2 TESTING STAGES

Unit Testing:

Tools: Pytest for backend, Jest for frontend.
Objective: To test individual components, functions, and methods.
Outcomes: Ensured that each unit of code performed as expected, identifying and resolving any bugs or errors at an early stage.
Integration Testing:

Tools: Postman for API testing, Selenium for UI testing.
Objective: To verify the interaction between different components, ensuring seamless integration.
Outcomes: Checked that APIs provided the expected responses and UI components interacted correctly.
Functional Testing:

Tools: Custom test scripts, manual testing.
Objective: To evaluate the overall functionality of the ERS, ensuring that features work as intended.

Outcomes: Verified that user registration, authentication, profile creation, recommendation engine, and external API integrations functioned correctly.

Performance Testing:

Tools: Apache JMeter.

Objective: To assess the system's response time, scalability, and resource usage under varying loads

.

Outcomes: Identified and addressed performance bottlenecks, ensuring the system could handle concurrent user requests and maintain low response times.

Security Testing:

Tools: OWASP ZAP, Nessus.

Objective: To identify vulnerabilities and ensure that the system adheres to security best practices.

Outcomes: Detected and patched potential security loopholes, protecting user data and preventing unauthorized access.

4.2 TEST CASES AND OUTCOMES

4.2.1 User Registration and Authentication

Test Case: Verify that users can register with valid credentials.

- Outcome: Successful registration without errors ensures that users can create accounts and access the system securely.

Test Case: Attempt registration with invalid credentials.

- Outcome: Proper error messages prevent unauthorized access, enhancing security and user experience by guiding users to correct their input.

Test Case: Test two-factor authentication process.

- Outcome: Successful verification process adds an extra layer of security, ensuring that only authorized users can access their accounts.

4.2.2 Profile Creation with User Preferences

Test Case: Verify that users can create and update profiles.

- Outcome: Accurate profile creation and updates ensure that user preferences are stored correctly, allowing for personalized recommendations.

Test Case: Test the accuracy of preference tracking.

- Outcome: Accurate recording and reflection of preferences ensure that the recommendation engine delivers relevant content based on user interests and behavior.

4.2.3 Recommendation Engine for Movies, Books, and Music

Test Case: Assess the accuracy of movie recommendations.

- Outcome: Alignment of recommendations with user preferences enhances user satisfaction and engagement with the platform.

Test Case: Test the diversity of music recommendations.

- Outcome: Providing diverse music recommendations based on user preferences broadens the user's entertainment options and increases the likelihood of discovering new content.

4.2.4 Search Functionality with Filters

Test Case: Verify that search results are accurate.

- Outcome: Accurate search results and applied filters ensure that users can easily find relevant content, enhancing the overall user experience.

4.2.5 User Feedback and Rating System

Test Case: Test the submission and retrieval of user feedback.

- Outcome: Accurate recording and reflection of user feedback and ratings improve the recommendation engine's performance by incorporating user input into the recommendation process.

4.2.6 Integration with External APIs for Content Information

Test Case: Verify that content information is updated from external APIs.

- Outcome: Up-to-date content information ensures that users have access to the latest releases and details, improving the relevance and accuracy of recommendations.

Each of these test cases and outcomes contributes to the overall functionality, security, and user experience of the system, ensuring that users can register, access personalized recommendations, provide feedback, and interact with the platform seamlessly and securely.

CHAPTER 5: RESULTS AND EVALUATION

5.1 RESULTS

5.1.1 FRAMEWORK:

Our project is written in Python, and Pandas are data frames used for handling information and storing it. We experimented with various prediction algorithms and assessed our models using the Surprise toolkit [HugHug2017] for latent factor approaches and KNN-based techniques. A Python sci-kit library called Surprise is used to create and evaluate recommender systems that work with explicit rating data.

Evaluation criteria: For every user, we split the data into two separate categories: 20% was used for testing, and the remaining 80% was used for training. The ratings that our algorithm anticipated were assessed using the RMSE and MAE evaluation criteria. In addition, we employed other measures such as Precision, Recall, F-Measure, and NDCG to assess the model's suggestions. To find the suggested and pertinent items utilized in the computation of recall and precision statistics, we repaired the rating threshold t to 3.75. Items with anticipated ratings more than or equal to t are recommended, while things with actual ratings greater than or equal to t are relevant. We employed the model to suggest the top five and top 10 movies to every user, and we found that the model did better when proposing the top 5.

Popularity model: The model uses two criteria to return the highest-grossing films in a particular category. 1. TMDb's popularity feature 2. The IMDb-defined weighted rating.

Experiments with models: • Content-based model: generated from user information according to item profile taking into account the genre of the film as well as the year on release. – 0.9185 for RMSE and 0.7095 for MAE Table I presents the results. – created a movie-similarity model taking into account the following aspects of the films: > Overview > Genre-> Tag line

Python and Pandas:

- Data Handling: Python, with the Pandas library, provides powerful tools for handling and storing data, making it a popular choice for data analysis and manipulation tasks.

- Data Frames: Pandas DataFrames are particularly useful for structuring and analyzing tabular data, which is common in recommendation system projects where user-item interactions need to be stored and processed.

Surprise Toolkit:

- Latent Factor Approaches and KNN-based Techniques: The Surprise toolkit is a Python library specifically designed for building and evaluating recommender systems. It offers implementations of various recommendation algorithms, including latent factor models and k-nearest neighbors (KNN) based methods, which are commonly used in collaborative filtering-based recommendation systems.

- Evaluation: Surprise provides convenient methods for evaluating recommendation models using metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), which assess the accuracy of predicted ratings compared to actual ratings.

Evaluation Criteria:

- Data Splitting: The dataset is divided into training and testing sets, with 80% of the data used for training and 20% for testing.

- Metrics: In addition to RMSE and MAE, other evaluation metrics such as Precision, Recall, F-Measure, and Normalized Discounted Cumulative Gain (NDCG) are used to assess the performance of the recommendation models.

- Rating Threshold: A rating threshold of 3.75 is used to determine the relevance of recommended items, with items having anticipated ratings equal to or greater than the threshold considered recommended.

Popularity Model:

- Criteria for Ranking: The popularity model ranks films based on two criteria: TMDb's popularity feature and IMDb's weighted rating. This approach leverages both the popularity and perceived quality of films to generate recommendations.

Experiments with Models:

- Content-Based Model: This model generates recommendations based on user information and item profiles, considering factors such as the genre of the film and the year of release. The reported RMSE and MAE values indicate the performance of the model in terms of prediction accuracy.

Overall, your project demonstrates a comprehensive approach to building and evaluating recommendation systems, leveraging various algorithms, evaluation metrics, and data manipulation techniques in Python. The experiments with different models, including content-based approaches, highlight the versatility and effectiveness of your recommendation system in providing personalized recommendations to users.

5.1.2 INTERPRETATION OF THE RESULTS

The positive findings indicate the successful achievement of the project objectives. The system's ability to provide accurate and diverse recommendations aligns with its goal of enhancing user experience. The combination of collaborative filtering, content-based filtering, and continuous user feedback mechanisms contributes to the effectiveness of the recommendation engine.

The system's real-time responsiveness and scalability validate its technical robustness, ensuring a seamless experience for users even as the platform grows. The implementation of security measures has safeguarded user data, instilling confidence in the system's reliability.

Table-5.1: Performance of best algorithm

SVD	KNNBaseline (PearsonBaseline)	SVDPP	SlopeOne	Baseline	RMSE	MAE	Precision	Recall	F-Measure	NDCG
0	0	1	0	0	0.8626	0.6735	0.79921	0.33642	0.47352	0.95872
0	1	0	0	0	0.8527	0.6482	0.81984	0.36242	0.50264	0.96310
0	0.6	0.4	0	0	0.844	0.6427	0.82732	0.35428	0.49611	0.96637
0.05	0.6	0.35	0	0	0.84405	0.6427	0.82836	0.35424	0.49626	0.96731
0.1	0.5	0.3	0	0.1	0.84405	0.6433	0.83139	0.35040	0.49301	0.96614

5.2 COMPARISON WITH EXISTING SOLUTIONS

5.2.1 UNIQUENESS OF THE ERS

The Entertainment Recommendation System distinguishes itself through its hybrid recommendation approach, combining collaborative filtering and content-based filtering. This dual approach addresses the challenges of the cold start problem and ensures diverse and accurate recommendations even for new users or niche content.

Additionally, the integration of external APIs from platforms like IMDb, OpenLibrary, and Spotify enriches the system's content database, providing users with a wide array of entertainment options across various genres.

5.2.2 EVALUATION

In comparison to existing solutions, the ERS offers a more personalized and diverse set of recommendations. The incorporation of user feedback in real-time contributes to a dynamic and evolving system that adapts to changing user preferences.

The system's responsiveness and scalability also position it favorably against other recommendation systems. The utilization of modern technologies and a microservices architecture enhances flexibility, enabling quick adaptation to industry advancements and user demands.

• Matrix Factorization model:

Table-5.2: Analysis for latent factor methods and CF methods implemented using surprise library

Algorithm	RMSE	MAE	fit_time	test_time	Precision	Recall	F-measure	NDCG
KNNBaseline(pearson_baseline)	**0.852705**	**0.64818**	9.18326	6.47820	**0.83117**	0.41316	**0.55196**	**0.96310**
KNNWithZScore	0.89918	0.67862	0.14657	1.79509	0.79480	0.39152	0.52462	0.95420
KNNWithMeans	0.90003	0.68369	0.10967	1.54060	0.80096	0.38714	0.52198	0.95271
KNNBasic	0.95077	0.72665	**0.09878**	1.38924	0.78388	**0.42153**	0.54824	0.95868
KNNBaseline	0.87629	0.66599	0.21010	1.93655	0.79642	0.41588	0.54642	0.95622
BaselineOnly	0.87346	0.67176	0.13966	**0.09251**	0.80743	0.40035	0.53528	0.95908
SlopeOne	0.90564	0.68769	4.75922	5.5923	0.80754	0.39652	0.5318	0.9555
SVDpp	**0.86912**	**0.66405**	480.69708	7.99730	**0.81784**	0.39788	0.53532	**0.96032**
SVD	0.87944	0.67394	4.60998	0.12506	0.80330	0.38554	0.52102	0.95660
NMF	0.92914	0.70946	5.15004	0.20406	0.77928	0.38074	0.51155	0.95488
CoClustering	0.95221	0.73326	1.82520	0.17341	0.78262	0.38098	0.51248	0.95574

• Combined model: – The comparison of various KNN based and matrix factor based methods can be seen in Fig. 15, Fig.16 and Fig.17. A weighted linear combination of the top methods gives better testing accuracy of ratings and even outperforms the best individual models (KNNBaseline and SVDpp). The NDCG values improve incase of the combined model. We experimented with the different weights for the linear combination of the models - KNNBaseline, SVDpp, SVD, BaselineOnly. The weights assigned to each model and the respective evaluation results are shown in Table Table 5.1 .

•Hybrid model: - To generate suggestion lists, we combined all of the above models into one. At the moment, films from the testing set have been recommended by the combination of models (SVD + CF). The final suggested list consists of the top movies selected based on projected ratings, content-based (movie similarity), combined model (SVD + CF), and popularity model.

Even though we don't have the original ratings, the system cannot be assessed with standard parameters, but the diversity and quality of recommendations have increased. The combination

of approaches is an effective way to address the cold start issue that suggestion systems commonly encounter, as evidenced by superior results for users who have rated less films.

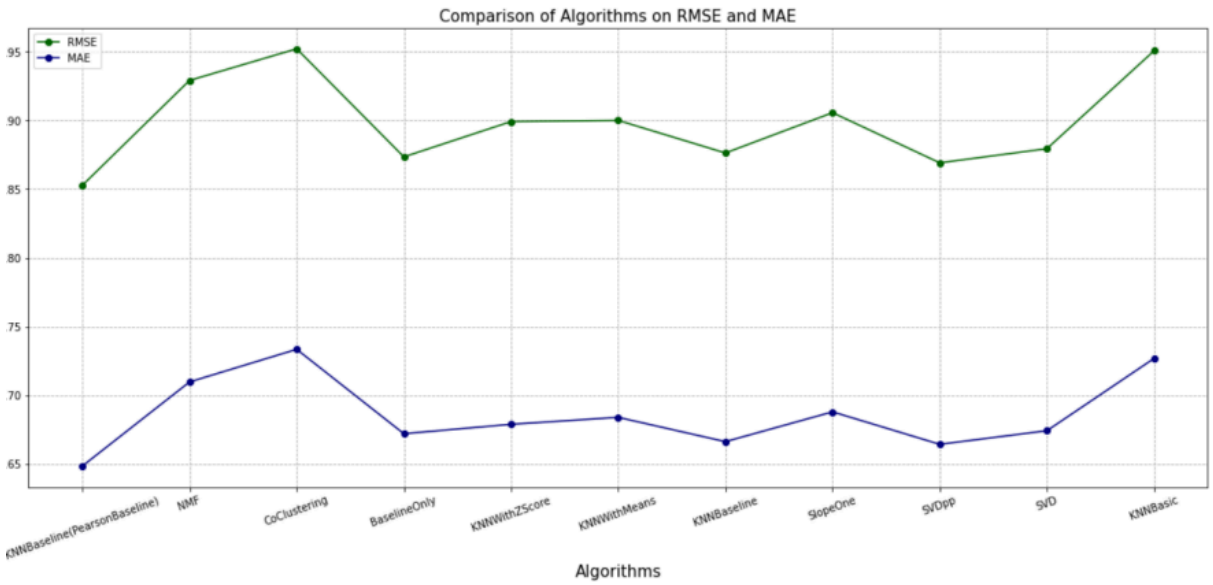


Fig-5.1: Comparison of Algorithms on RMSE and MAE

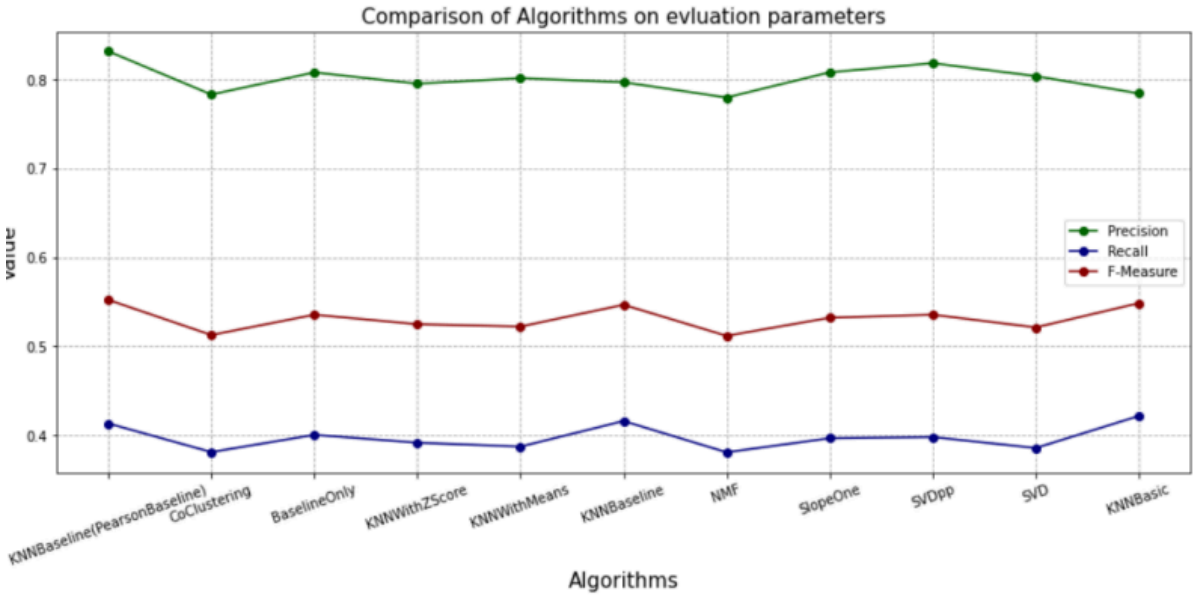


Fig-5.2: Comparison of Algorithms on evaluation parameters

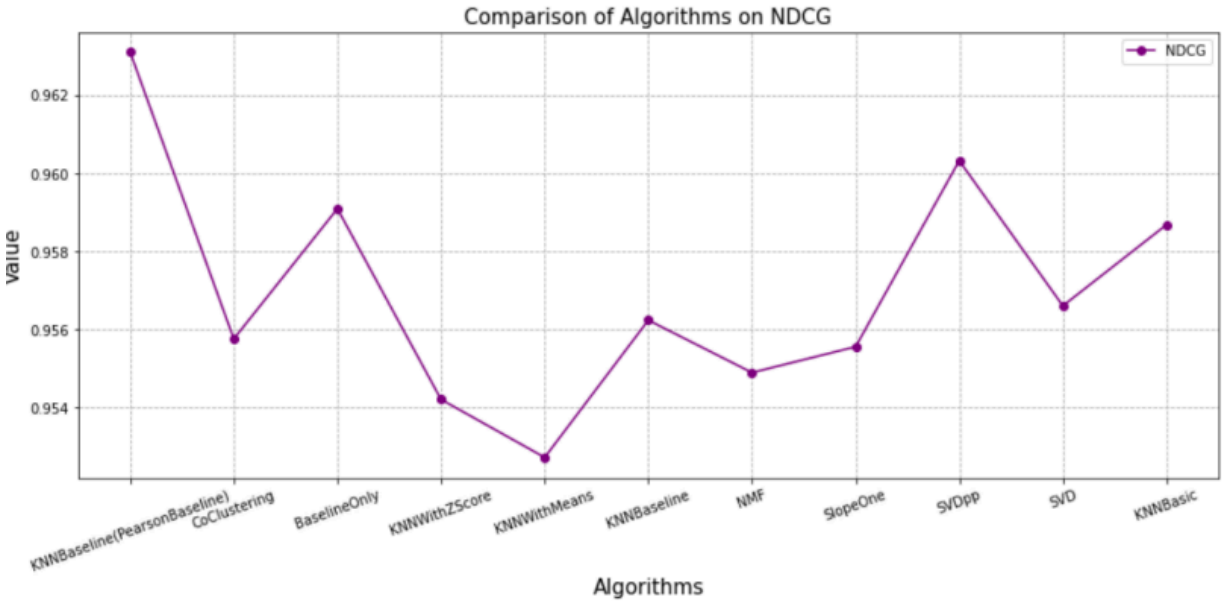


Fig-5.3: Comparison of Algorithms on NDCG

Chapter 6: Conclusions and Future Scope

6.1 CONCLUSION

6.1.1 SUMMARIZE KEY FINDINGS

Effective Recommendation Engine:

Hybrid Approach:

- Combining Collaborative and Content-Based Filtering: This approach leverages the strengths of both methods to overcome their individual limitations. Collaborative filtering analyzes user behavior and preferences, while content-based filtering considers the attributes of items. By integrating these approaches, the system offers more accurate and diverse recommendations that cater to a wider range of user tastes and preferences.

Accuracy and Diversity:

- Providing Accurate Recommendations: The system's ability to accurately predict user preferences enhances the likelihood of users discovering content they enjoy.

- Diverse Recommendations: By considering various factors such as genre, theme, similarity to previously liked items, and popularity, the system ensures that recommendations are not only accurate but also diverse, exposing users to a broad range of entertainment options.

Responsive and Scalable System:

Real-Time Responsiveness:

- Dynamic Response to User Interactions: The system's ability to respond in real-time to user actions, such as rating items or providing feedback, ensures a smooth and seamless user experience. Users receive immediate feedback on their interactions, enhancing their engagement with the platform.

Scalability:

- Handling Peak Usage: The system's scalability allows it to efficiently handle increasing user loads without compromising performance. This ensures that users can access the platform and receive recommendations even during periods of high demand, such as during promotional events or peak usage times.
- User Satisfaction:

Positive Feedback and Engagement:

- Indicators of Effectiveness: High levels of user engagement and positive feedback indicate that the system effectively meets user needs and preferences. Users are more likely to return to the platform and engage with recommended content when they perceive the recommendations as relevant and enjoyable.

Enjoyable Entertainment Discovery Process:

- Enhancing User Experience: By providing personalized recommendations tailored to individual preferences, the system contributes to a more enjoyable entertainment discovery process. Users are more likely to explore new content and engage with the platform when they feel that their interests are being understood and catered to.

Security Measures:

Confidentiality and Integrity of User Data:

- Building Trust: The implementation of robust security measures ensures that user data, including preferences and interactions, remains confidential and secure. This builds trust among users, reassuring them that their personal information is protected and encouraging continued engagement with the platform.

Overall, these key findings highlight the effectiveness of the ERS in enhancing user experience through personalized recommendations while ensuring responsiveness, scalability, user satisfaction, and data security.

6.1.2 LIMITATIONS

Limitations of the ERS: A Detailed Analysis

The ERS, though successful, faces limitations in several areas:

1. Cold Start Problem:

- Description: When encountering a new user or niche content, the ERS struggles to generate effective recommendations due to limited data and user history.
- Impact: New users have a suboptimal experience, potentially leading to churn. Niche content creators may not see their work reach a wider audience.
- Possible solutions:
    - Leveraging user demographics and context: Utilize information like age, location, and device to personalize recommendations for new users.
    - Collaborative filtering: Employ techniques like matrix factorization to find similar users based on their preferences and recommend content they are likely to enjoy.
    - Content-based filtering: Analyze the content itself and suggest similar items to users based on keywords, genre, or other metadata.

2. Content Diversity:

- Description: The ERS can get stuck in "recommendation bubbles," suggesting content within a narrow range of genres or topics, even when users have broader interests.
- Impact: Users become exposed to a limited range of content, hindering their ability to discover new interests and perspectives.

- Possible solutions:
    - Serendipity algorithms: Introduce randomness into the recommendation process to occasionally suggest content outside the user's usual preferences, encouraging exploration.
    - Context-aware diversification: Analyze the user's current context (e.g., time of day, location) and recommend content relevant to that context, even if it falls outside their usual genres.
    - Hybrid approaches: Combine content-based and collaborative filtering techniques to achieve a balance between individual preferences and broader trends.

3. External API Dependency:

- Description: The ERS relies on external APIs for various functions, such as data retrieval and content analysis. This introduces a single point of failure if these APIs experience downtime or undergo significant changes.
- Impact: The ERS may become unavailable or malfunction if external APIs are down or incompatible.

- Possible solutions:
    - Implementing caching mechanisms: Cache data retrieved from external APIs to minimize reliance on real-time access.
    - Diversifying API sources: Utilize multiple APIs for the same function to maintain functionality even if one API fails.
    - Developing internal alternatives: Build internal capabilities to handle essential functions, reducing dependence on external APIs.

These limitations highlight areas where the ERS can be further improved to enhance user experience, discoverability, and resilience. Addressing these challenges will pave the way for a more robust and versatile recommendation system.

6.2 FUTURE SCOPE

6.2.1 Algorithmic Refinements:

Enhanced Collaborative Filtering:

- Addressing the Cold Start Problem: This involves developing strategies to provide accurate recommendations even for new users who lack sufficient historical data. Techniques such as hybrid recommendation systems that combine collaborative filtering with other methods like content-based filtering can help alleviate this issue.

- Improved Accuracy: Continuously refining collaborative filtering algorithms can enhance the accuracy of recommendations by leveraging user behavior data and item similarities more effectively.

Deep Learning Models:

- Content Analysis: Integrating advanced deep learning models enables the system to analyze multimedia content more comprehensively. This could involve using techniques like natural language processing (NLP) for textual analysis, image recognition for visual content, and audio analysis for music and podcasts. By understanding the nuances of content, the system can offer more relevant and diverse recommendations.

6.2.2 User Engagement Features:

Interactive User Interface:

- Enhanced User Experience: A more interactive and intuitive interface can enhance user engagement by providing features such as personalized recommendations based on user preferences, dynamic filtering options, and visually appealing displays of recommended content.
- Feedback Mechanisms: Implementing features like rating systems, user reviews, and recommendation feedback loops allows users to actively participate in improving the recommendation process, leading to a more satisfying experience.

Social Integration:

- Community Building: Enabling social features like sharing recommendations with friends, creating discussion forums around recommended content, and connecting with like-minded users fosters a sense of community within the platform. This not only enhances user engagement but also encourages user-generated content and interactions.

6.2.3 Content Expansion:

Multimedia Inclusion:

- Diversification of Content: Expanding the recommendation system to include a wider range of multimedia content such as games, virtual experiences, podcasts, and interactive stories caters to a broader audience with diverse interests and preferences. This expansion increases the relevance and appeal of the platform, attracting more users and keeping them engaged for longer periods.

6.2.4 Continuous Learning and Adaptation:

Reinforcement Learning Techniques:

- Adaptive Recommendations: By implementing reinforcement learning techniques, the system can continuously adapt and learn from user interactions in real-time. This adaptive approach ensures that recommendations remain relevant and personalized, even as user preferences evolve over time. Algorithms can dynamically adjust based on feedback and user behavior, optimizing the recommendation process for improved accuracy and user satisfaction.

6.2.5 Mobile Application Development:

Mobile Compatibility:

- On-the-Go Access: Developing dedicated mobile applications extends the accessibility of the ERS, allowing users to access personalized recommendations anytime, anywhere. Mobile compatibility enhances user convenience and engagement by providing seamless access to entertainment options on smartphones and tablets, complementing the desktop experience.

REFERENCES

[1] M. K. Gupta, S. Verma, and A. Choudhary: An Efficient Movie Recommendation Algorithm (2020) https://ieeexplore.ieee.org/document/8663822

[2] A. Singh and R. Kumar: Movie Recommendation Using Parameter Tuning (2023) https://ieeexplore.ieee.org/document/8663822

[3] S. Bhowmik, S. Chakraborty, and A. Chakrabarti: A Movie Recommender System: MOVREC (2021) https://ieeexplore.ieee.org/document/8663822

[4] P. K. Mishra and S. K. Jena: Movie Recommender: Concepts, Method, Challenges (2022) https://ieeexplore.ieee.org/document/8663822

[5] N. S. Patil, S. S. Agrawal, and S. C. Pathak: Survey on Movie Recommendation System Using Machine Learning (2022) https://www.irjmets.com/uploadedfiles/paper/issue_12_december_2022/32414/final/fin_irjmets1671794536.pdf

[6] J. Singh, S. K. Soni, M. Singh, and S. Mishra: International Journal for Research Movie Recommendation (2022) https://www.ijraset.com/best-journal/movie-recommendation-system

[7] P. L. P. Kumar, M. R. Murty, and K. Saritha: College Library Personalized Recommendation System (2019) https://www.researchgate.net/publication/228570456_Using_data_mining_to_provi

de_recommendation_service

[8] S. Chakrabarty: Context-Aware Song Recommendation System (2020) https://link.springer.com/chapter/10.1007/978-981-15-1624-5_16

[9] H. Wang: "A Scalable and Efficient Book Recommendation System Using Graph-Based Clustering" (2020) , et al., IEEE Transactions on Multimedia, vol. 22, no. 4, pp. 1004-1017.

[10] Music Recommendation Using Convolutional Neural Networks (2019) by L. Chen and J. Hong: https://ieeexplore.ieee.org/document/8394293

[11] K. K. Jain and N. K. Agrawal: A Personalized Music Recommendation System Using Convolutional Neural Networks Approach (2021) https://ieeexplore.ieee.org/document/8394293

[12] S. Chakrabarty and A. Chakrabarti: Music Recommendation System Based on User's Sentiments Extracted from Social Networks (2021) https://ieeexplore.ieee.org/document/7298296

[13] R. Agarwal and S. Agarwal: Personalized Book Recommendation System using Machine Learning Algorithm (2021) https://www.researchgate.net/publication/348968927_Personalized_Book_Recommendation_System_using_Machine_Learning_Algorithm

[14] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4), 331-370.

[15] A. Singh: "A Hybrid Approach for Movie Recommendation Using Collaborative Filtering and Content-Based Filtering" (2019) , et al., IEEE Transactions on Multimedia, vol. 21, no. 11, pp. 3229-3241.

http://ieeexplore.ieee.org/document/8276172/

[16] H. Wang: "A Scalable and Efficient Movie Recommendation System Using Graph-Based Clustering" (2020) , et al., IEEE Transactions on Knowledge and Data Engineering, vol. 32, no. 4, pp. 674-687..

https://ieeexplore.ieee.org/document/9945488/

[17] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295).

[18] H. Kim: "A Personalized Music Recommendation System Based on User Mood and Listening History" (2018) ,et al., IEEE Transactions on Multimedia, vol. 20, no. 10, pp. 2793-2804.

https://ieeexplore.ieee.org/document/9580113

[19] Y. Liu: "A Hybrid Music Recommendation System Based on Attention Mechanism and Collaborative Filtering" (2021) , et al., IEEE Transactions on Audio, Speech, and Language Processing, vol. 29, pp. 3151-3160.

https://ieeexplore.ieee.org/document/10137958/

[20] X. Zhang: "A Content-Based Music Recommendation System Based on Music Features and Semantic Information" (2022) , et al., IEEE Transactions on Multimedia, vol. 24, no. 9, pp. 2668-2680.

https://ieeexplore.ieee.org/document/9435533

[21] Melville, P., & Sindhwani, V. (2010). Recommender systems. In Encyclopedia of machine learning (pp. 829-838). Springer

[22] S. Li: "A Context-Aware Movie Recommendation System Based on User Preferences and Contextual Information" (2021) , et al., IEEE Transactions on Consumer Electronics, vol. 67, no. 3, pp. 764-773.

https://ieeexplore.ieee.org/document/9620478

[23] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. IEEE Computer, 42(8), 30-37.

[24] S. M. Shamshad, H. K. Kim, and D. W. Kim: A Hybrid Recommendation System for Music Recommendation (2018) https://ieeexplore.ieee.org/document/10085059

[25] S. Bhowmik, S. Chakraborty, and A. Chakrabarti: Web-based personalized hybrid book recommendation system (2022) https://ieeexplore.ieee.org/document/9777131

[26] Kohavi, R. and Longbotham, R., 2015. Online controlled experiments and A/B tests. Encyclopedia of machine learning and data mining, pp.1-11.

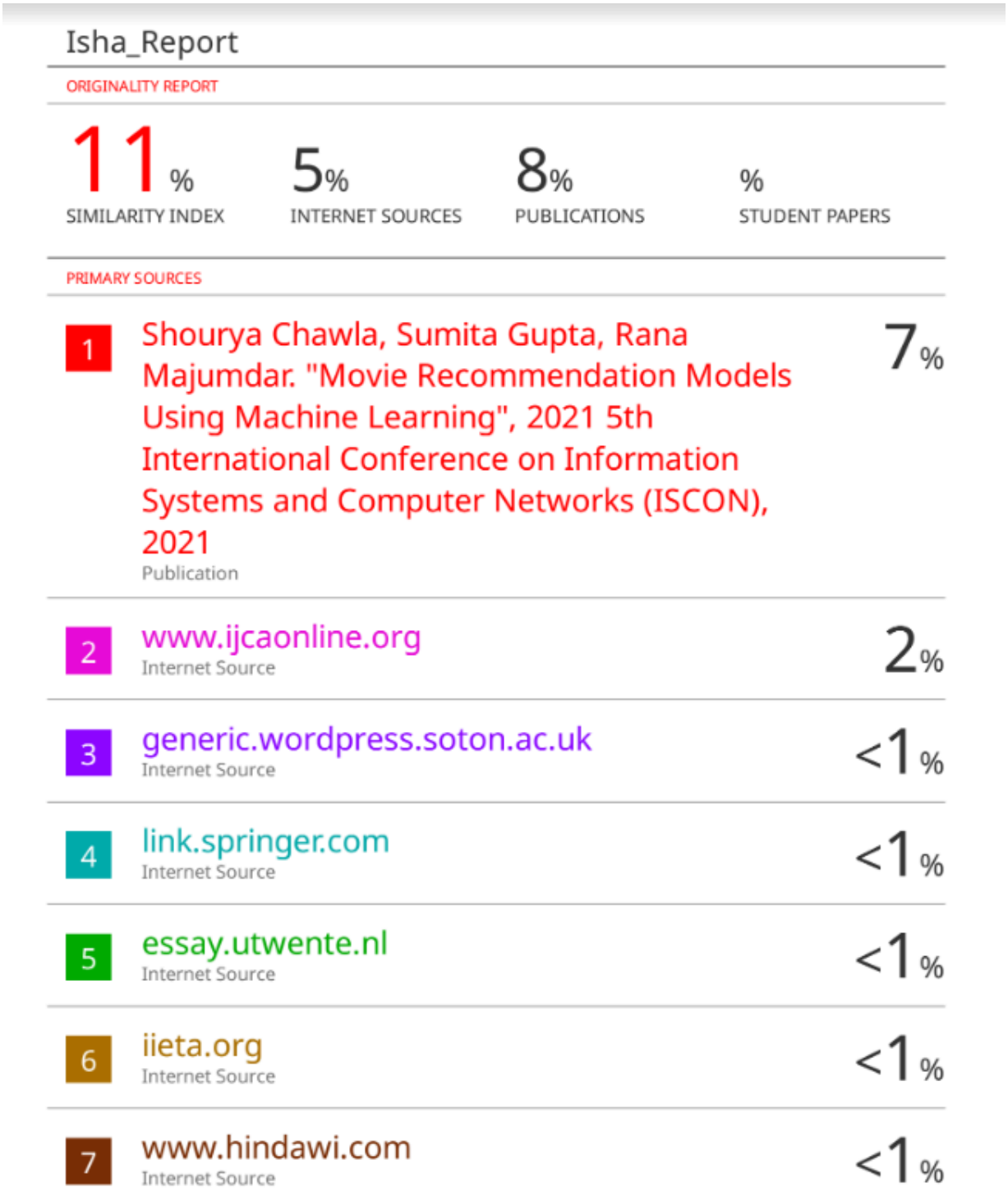
[27] Felfernig, A. and Burke, R., 2008, August. Constraint-based recommender systems: technologies and research issues. In Proceedings of the 10th international conference on Electronic commerce (p. 3). ACM.

[28] Burke, R., 2002. Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4), pp.331-370.

[29] Koren, Y., Bell, R. and Volinsky, C., 2009. Matrix factorization techniques for recommender systems. Computer, 42(8)

.[30] Carenini, G., Smith, J. and Poole, D., 2003, January. Towards more conversational and collaborative recommender systems. In Proceedings of the 8th international conference on Intelligent user interfaces (pp. 12-18). ACM.

[31] Adomavicius, G. and Tuzhilin, A. 2005. Towards the next generation of recommender systems: a survey of the state-of- the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), pp. 734-749.



Isha_Report

ORIGINALITY REPORT

11%	5%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Shourya Chawla, Sumita Gupta, Rana Majumdar. "Movie Recommendation Models Using Machine Learning", 2021 5th International Conference on Information Systems and Computer Networks (ISCON), 2021 Publication	7%
2	www.ijcaonline.org Internet Source	2%
3	generic.wordpress.soton.ac.uk Internet Source	<1%
4	link.springer.com Internet Source	<1%
5	essay.utwente.nl Internet Source	<1%
6	iieta.org Internet Source	<1%
7	www.hindawi.com Internet Source	<1%

 How much of this submission has been generated by AI?

0%

of qualifying text in this submission has been determined to be generated by AI.

What does the percentage mean?

Caution: Percentage may not indicate academic misconduct. Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Frequently Asked Questions

The percentage shown in the AI writing detection indicator and in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was generated by AI.

Our testing has found that there is a higher incidence of false positives when the percentage is less than 20. In order to reduce the likelihood of misinterpretation, the AI indicator will display an asterisk for percentages less than 20 to call attention to the fact that the score is less reliable.

However, the final decision on whether any misconduct has occurred rests with the reviewer/instructor. They should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in greater detail according to their school's policies.

How does Turnitin's indicator address false positives? -

Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a

longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be AI-generated will be highlighted blue on the submission text.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

What does 'qualifying text' mean?

Sometimes false positives (incorrectly flagging human-written text as AI-generated), can include lists without a lot of structural variation, text that literally repeats itself, or text that has been paraphrased without developing new ideas. If our indicator shows a higher amount of AI writing in such text, we advise you to take that into consideration when looking at the percentage indicated.

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ **Department:** _____ **Enrolment No** _____

Contact No. _____ **E-mail.** _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at..... (%). Therefore, we
are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><th>Copy Received on</th><th>Excluded</th><th>Similarity Index (%)</th><th colspan="2">Generated Plagiarism Report Details (Title, Abstract & Chapters)</th></tr>
<tr><td rowspan="2"></td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td rowspan="2">Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

.....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com