

**REAL TIME RESEARCH PROJECT ON**  
**CRAFTING PERSONALIZED MOVIE RECOMMENDATIONS: EXPLORING**  
**THE MOVIELENS DATASET AND WORD CLOUD VISUALIZATION FOR**  
**RECOMMENDER SYSTEMS**

**BACHELOR OF TECHNOLOGY**  
**IN**  
**COMPUTER SCIENCE AND ENGINEERING**  
**(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)**

Submitted By

**Syeda Sanobar Ali-22RA1A6673**  
**Mukku Shiva Sai Ganesh-22RA1A6677**  
**Wasalwar Shivam-22RA1A6678**  
**Boga Supriya-22RA1A6683**

Under the guidance of  
**Dr.S.Sankar Ganesh**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)**

**KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY (UGC-**  
**AUTONOMOUS, AFFILIATED TO JNTUH, GHANPUR(V),**  
**GHATKESAR(M),**  
**MEDCHAL(D)-501301)**  
**2023- 2024**

# KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY

(Affiliated to JNTUH,UGC-AUTONOMOUS, Ghanpur(V), Ghatkesar(M), Medchal(D)-501301)



## CERTIFICATE

This is to certify that the project work entitled **“Crafting Personalized Movie Recommendations: Exploring the MovieLens Dataset and Word Cloud Visualization for Recommender System”** is submitted by Ms.Syeda Sanobar Ali(22RA1A6673), Mr.Mukku Shiva Sai Ganesh(22RA1A6677), Mr.Wasalwar Shivam(22RA1A6678),Ms.Boga Supriya(22RA1A6683) in B.Tech II-II semester **Computer Science and Engineering –AI&ML** is a recorded bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

**Internal Examiner**

**HOD**

**Co-ordinator**

**External Examiner**

## **DECLARATION**

We Syeda Sanobar Ali(22RA1A6673), Mukku Shiva Sai Ganesh(22RA1A6677),Wasalwar Shivam(22RA1A6678),Boga Supriya(22RA1A6683) hereby declare that the results embodied in this project dissertation entitled “**CRAFTING PERSONALIZED MOVIE RECOMMENDATIONS: EXPLORING THE MOVIELENS DATASET AND WORD CLOUD VISUALIZATION FOR RECOMMENDER SYSTEMS**” is carried out by us during the year 2024 in partial fulfillment of the award of Bachelor of Technology in **Computer Science & Engineering(AI&ML)** from **Kommuri Pratap Reddy Institute of Technology**. It is an authentic record carried out by us under the guidance of **Dr.S.Sankar Ganeshan**, Associate Professor, Department of **COMPUTER SCIENCE & ENGINEERING (AI&ML)**.

**Date:**

**Place:**

**BY**

**Syeda Sanobar Ali-22RA1A6673**

**Mukku Shiva Sai Ganesh-22RA1A6677**

**Wasalwar Shivam-22RA1A6678**

**Boga Supriaya-22RA1A6683**

## ACKNOWLEDGEMENT

It gives us immense pleasure to acknowledge with gratitude, the help and support extended throughout the project report from the following:

We will be very much grateful to almighty, parents who made us capable of carrying out our job.

We express our profound gratitude to **Dr.E.RAVINDRA, Principal of Kommuri Pratap Reddy Institute of Technology**, who has encouraged in completing our project report successfully.

We are grateful to **Dr. J . SRIKANTH** who is our **Head of the Department, AI&ML** for his amiable ingenious and adept suggestions and pioneering guidance during the project report.

We express our gratitude and thanks to the Project Coordinator **DR. S. SANKAR GANESH**, Associate Professor, CSE(AI&ML) of our department for his contribution for making it success with in the given time duration.

We express our deep sense of gratitude and thanks to **Internal Guide, Dr.S.Sankar Ganesh , Associate Professor, CSE(AI&ML)** for his guidance during the project report.

We are also very thankful to our **Management, Staff Members** and all **Our Friends** for their valuable suggestions and timely guidance without which we would not have been completed it.

**BY**

**Ms.Syeda Sanobar Ali -22RA1A6673**  
**Mr.Mukku Shiva Sai Ganesh-22RA1A6677**  
**Mr.Wasalwar Shivam-22RA1A6678**  
**Ms.Supriya Boga-22RA1A6683**



## KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY COMPUTER SCIENCE AND ENGINEERING (AI&ML)

### VISION OF THE INSTITUTE

To emerge as a premier institute for high quality professional graduates who can contribute to economic and social developments of the Nation.

### MISSION OF THE INSTITUTE

Mission	Statement
IM1	To have holistic approach in curriculum and pedagogy through industry interface to meet the needs of Global Competency.
IM2	To develop students with knowledge, attitude, employability skills, entrepreneurship, research potential and professionally Ethical citizens.
IM3	To contribute to advancement of Engineering & Technology that would help to satisfy the societal needs.
IM4	To preserve, promote cultural heritage, humanistic values and Spiritual values thus helping in peace and harmony in the society.

## **VISION OF THE DEPARTMENT**

To Provide Quality Education in Computer Science for the innovative professionals to work for the development of the nation.

## **MISSION OF THE DEPARTMENT**

<b>Mission</b>	<b>Statement</b>
<b>DM1</b>	Laying the path for rich skills in Computer Science through the basic knowledge of mathematics and fundamentals of engineering
<b>DM2</b>	Provide latest tools and technology to the students as a part of learning Infrastructure
<b>DM3</b>	Training the students towards employability and entrepreneurship to meet the societal needs.
<b>DM4</b>	Grooming the students with professional and social ethics

## **PROGRAM EDUCATIONAL OBJECTIVES (PEOS)**

<b>PEO'S</b>	<b>Statement</b>
<b>PEO1</b>	Demonstrate technical skills, competency in AI&ML and exhibit team management capability with proper communication in a job environment.
<b>PEO2</b>	Support the growth of economy of a country by starting enterprise with a lifelong learning attitude.
<b>PEO3</b>	Carry out research in the advanced areas of AI&ML and address the basic needs of the society.
<b>PEO4</b>	To develop research attitude among the students in order to carry out research in cutting edge technologies, solve real world problems and provide technical consultancy services.

## PROGRAM OUTCOMES

<b>PO1</b>	<b>Engineering Knowledge:</b> Apply the knowledge of mathematics, science, Engineering fundamentals, and an engineering specialization to the solution of complex Engineering problems.
<b>PO2</b>	<b>Problem Analysis:</b> Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
<b>PO3</b>	<b>Design/development of Solutions:</b> Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
<b>PO4</b>	<b>Conduct investigations of complex problems:</b> Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
<b>PO5</b>	<b>Modern tool usage:</b> Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
<b>PO6</b>	<b>The engineer and society:</b> Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
<b>PO7</b>	<b>Environment and sustainability:</b> Understand the impact of the professional engineering solutions in societal and environmental context, and demonstrate the knowledge of, and need for sustainable development.
<b>PO8</b>	<b>Ethics:</b> Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.



<b>PO9</b>	Individual and team network: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
<b>PO10</b>	Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, being able to comprehend and write effective reports and design documentation, make Effective presentations, and give and receive clear instructions.
<b>PO11</b>	Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work,as a member and leader in a team, to manage projects and in multidisciplinary environment.
<b>PO12</b>	Life-Long learning: Recognize the need for, and have the preparation and able to engage in independent and life-long learning in the broadest context of technological change.

## PROGRAM SPECIFIC OUTCOMES

<b>PSO1</b>	Demonstrate the knowledge of human cognition, Artificial Intelligence, Machine Learning and data engineering for designing intelligence systems.
<b>PSO2</b>	Apply computational knowledge and project development skills to provide innovative solutions.
<b>PSO3</b>	Use tools and techniques to solve problems in AI&ML.

# **CRAFTING PERSONALIZED MOVIE RECOMMENDATIONS: EXPLORING THE MOVIELENS DATASET AND WORD CLOUD VISUALIZATION FOR RECOMMENDER SYSTEMS**

## **ABSTRACT**

The development of movie recommendation systems began in the late 1990s with the rise of e-commerce platforms and streaming services. Early methods relied heavily on collaborative filtering, where recommendations were based on user behavior and preferences. In the past, movie recommendations were often made by friends, family, or through movie critics and television programs. People relied on word-of-mouth, printed reviews, and televised recommendations to decide what movies to watch. The traditional system of movie recommendations was limited by a lack of personalization and scalability. It relied heavily on subjective opinions and could not cater to the unique tastes and preferences of individual users, often leading to unsatisfactory movie choices. The motivation behind developing machine learning-based movie recommendation systems is to provide personalized, accurate, and scalable recommendations that enhance user satisfaction and engagement. By leveraging vast amounts of data, these systems can uncover patterns and preferences that are not immediately apparent, offering a more tailored viewing experience. The proposed system for movie recommendations leverages advanced machine learning techniques to provide personalized suggestions. It integrates collaborative filtering, content-based filtering, and hybrid models to analyze user data and predict preferences. Collaborative filtering identifies patterns in user behavior by comparing the preferences of similar users, while content-based filtering analyzes movie attributes such as genre, actors, and directors to match user interests. A hybrid model combines these approaches to enhance recommendation accuracy. User interactions, such as viewing history, ratings, and search queries, are continuously collected and processed. Machine learning algorithms, including matrix factorization and deep learning, are employed to detect latent factors and complex patterns within the data. These models are trained and fine-tuned to improve over time, adapting to evolving user tastes. The system can also incorporate additional data sources, such as social media activity and demographic information, to further refine recommendations. By providing a seamless and personalized viewing experience, this approach enhances user satisfaction and engagement on streaming platforms like Netflix and Amazon Prime.

## **TABLE OF CONTENTS**

<b>ABSTRACT.....</b>	<b>X</b>
<b>CHAPTER – 1.....</b>	<b>4</b>
<b>INTRODUCTION.....</b>	<b>4</b>
1.1 Objective.....	4
1.2 Research Motivation.....	5
1.3 Problem Statement .....	6
1.4 Need and Significance .....	7
<b>CHAPTER – 2.....</b>	<b>8</b>
<b>LITERATURE SURVEY.....</b>	<b>8</b>
<b>CHAPTER 3.....</b>	<b>15</b>
<b>EXISTING SYSTEM .....</b>	<b>15</b>
3.1 Overview.....	15
3.2 Limitations .....	16
<b>CHAPTER 4.....</b>	<b>18</b>
<b>PROPOSED METHODOLOGY .....</b>	<b>18</b>
4.1 Overview.....	18
4.2 Data Preprocessing .....	20
4.3 THE BUILD CHART FUNCTION: .....	21
4.4 FILTRATION STRATEGIES FOR MOVIE RECOMMENDATION SYSTEMS .....	23
1. Content-Based Filtering.....	23
2. Collaborative Filtering.....	23
How to Build a Movie Recommendation System in Python? .....	24
<b>CHAPTER 5.....</b>	<b>25</b>
<b>SOFTWARE ENVIRONMENT .....</b>	<b>25</b>
5.1 What is Python? .....	25
5.2 Advantages of Python.....	26
5.3 Modules Used in Project.....	32

1.TensorFlow .....	32
2.NumPy .....	32
3.Pandas .....	32
4.Scikit – learn .....	33
5.4 Install Python Step-by-Step in Windows and Mac .....	34
<b>CHAPTER 6.....</b>	<b>41</b>
<b>SYSTEM REQUIREMENTS .....</b>	<b>41</b>
6.1 Software Requirements.....	41
6.2 Hardware Requirements .....	41
<b>CHAPTER 7.....</b>	<b>42</b>
<b>FUNCTIONAL REQUIREMENTS.....</b>	<b>42</b>
7.1 Output Design .....	42
7.2 Output Definition .....	42
7.3 Input Design.....	43
7.4 Input Stages.....	43
7.5 Input Types .....	43
7.6 Input Media .....	44
7.7 Error Avoidance .....	44
7.8 Error Detection .....	45
7.9 Data Validation .....	45
7.10 User Interface Design .....	45
7.11 User Initiated Interfaces.....	45
7.12 Computer-Initiated Interfaces .....	46
7.13 Error Message Design.....	46
7.14 Performance Requirements .....	46
<b>CHAPTER 8.....</b>	<b>47</b>
<b>SOURCE CODE.....</b>	<b>47</b>
<b>CHAPTER 9.....</b>	<b>55</b>
<b>RESULTS AND DISCUSSION.....</b>	<b>55</b>
9.1 IMPLEMENTATION AND DESCRIPTION .....	55
9.1.1 Description .....	56

9.2 Dataset Description: .....	56
9.3 Result and Description.....	58
<b>CHAPTER 10 .....</b>	<b>60</b>
<b>CONCLUSION AND FEATURE SCOPE.....</b>	<b>60</b>
10.1 Conclusion .....	60
10.2 Future Scope.....	60

# CHAPTER – 1

## INTRODUCTION

In the era of digital streaming, personalized movie recommendations have become essential for enhancing user experience and engagement. Traditional methods often rely on generic lists and ratings, which may not fully capture individual preferences and viewing habits. These conventional approaches often lead to less satisfying viewing experiences, as they fail to account for the diverse tastes and nuanced preferences of each user. This project aims to address these shortcomings by exploring the Movielens dataset, a comprehensive source of user ratings and movie metadata. By leveraging this rich dataset, the project seeks to develop an advanced recommender system that can provide more accurate and personalized movie suggestions.

One of the innovative aspects of this project is the use of word cloud visualization to highlight key features and patterns within the data. This visualization technique will help in understanding the most significant attributes that influence user preferences and how they correlate with movie choices. By analyzing user ratings and movie metadata, and visualizing these key features, the system can identify trends and preferences that are not immediately obvious through traditional methods.

The ultimate goal of this project is to enhance the overall viewing experience by offering highly personalized movie recommendations. By catering to individual tastes and preferences, this advanced recommender system aims to increase user satisfaction and engagement. This approach not only improves the accuracy of recommendations but also fosters a more enjoyable and tailored viewing experience for users, making it a valuable tool in the competitive landscape of digital streaming platforms.

### 1.1 Objective

The primary objective of this research is to develop a sophisticated and highly personalized movie recommendation system by leveraging the MovieLens dataset and advanced machine learning techniques. This system aims to deliver movie suggestions that are not only accurate but also tailored to individual user preferences, thereby enhancing the overall user experience on streaming platforms.

To achieve this, the research integrates collaborative filtering, content-based filtering, and hybrid models. Collaborative filtering involves analyzing user behavior and preferences to identify patterns and similarities among users. This method allows the system to make recommendations based on the collective experience of users with similar tastes. Content-based filtering, on the other hand, examines the attributes of movies—such as genre, cast, director, and plot—to find films that align with a user’s stated interests and viewing history.

A hybrid model is employed to combine the strengths of both collaborative and content-based filtering. This model enhances recommendation accuracy by utilizing multiple data sources and approaches. By incorporating machine learning algorithms such as matrix factorization and deep learning, the system can detect latent factors and complex patterns within the data, continuously improving its recommendations as it processes more user interactions.

The research also explores the integration of additional data sources, such as social media activity and demographic information, to refine recommendations further. This comprehensive approach ensures that the recommendation system can adapt to the evolving tastes and preferences of users, providing a seamless and engaging viewing experience on platforms like Netflix and Amazon Prime.

## 1.2 Research Motivation

The motivation behind this research stems from the limitations of traditional movie recommendation methods and the growing demand for personalized content in the digital age. Historically, movie recommendations were based on subjective opinions from friends, family, critics, or popular media. These recommendations lacked scalability and personalization, often resulting in unsatisfactory movie choices for many users. With the advent of e-commerce and streaming services, there is a significant opportunity to transform how users discover and enjoy movies.

Machine learning and big data analytics offer the potential to revolutionize movie recommendations by providing highly personalized and accurate suggestions. As streaming platforms accumulate vast amounts of user data, the ability to analyze this data to uncover hidden

patterns and preferences becomes increasingly important. Personalized recommendations can significantly enhance user satisfaction and engagement, driving the success of these platforms.

Additionally, the rise of social media and the availability of demographic data provide valuable insights that can further refine recommendations. By leveraging these data sources, the research aims to develop a system that not only meets but exceeds user expectations in terms of personalization and accuracy. The ultimate goal is to create a recommendation system that continuously adapts to user preferences, ensuring a more enjoyable and engaging viewing experience.

### 1.3 Problem Statement

The traditional system of movie recommendations is inherently limited by its reliance on subjective opinions and lack of scalability. Users often struggle to find movies that align with their unique tastes and preferences, leading to a suboptimal viewing experience. Despite the abundance of content available on streaming platforms, users are frequently overwhelmed by the choices and unable to discover movies that truly resonate with them.

This research addresses the need for a personalized and scalable movie recommendation system that leverages advanced machine learning techniques. The primary challenge lies in effectively analyzing vast amounts of user data to uncover hidden patterns and preferences. Collaborative filtering and content-based filtering each have their strengths, but individually, they are insufficient to meet the complex demands of modern users.

A hybrid model that integrates these approaches, combined with machine learning algorithms such as matrix factorization and deep learning, offers a promising solution. This system must be capable of continuously learning and adapting to evolving user tastes, providing increasingly accurate and personalized recommendations over time. Additionally, incorporating data from social media and demographic information poses a technical challenge but holds significant potential to enhance the system's effectiveness.



The proposed solution aims to overcome these challenges and deliver a seamless and engaging user experience on streaming platforms. By providing tailored movie suggestions, the system can increase user satisfaction and engagement, ultimately driving the success of streaming services.

## 1.4 Need and Significance

1. **Enhanced User Experience:** Personalized recommendations significantly improve user satisfaction by suggesting movies that align with individual tastes and preferences. This leads to a more enjoyable viewing experience and increased user engagement on streaming platforms.
2. **Scalability:** The traditional recommendation system is limited by its reliance on subjective opinions and cannot scale to meet the needs of millions of users. A machine learning-based system can analyze vast amounts of data to provide personalized recommendations at scale.
3. **Accurate Recommendations:** By leveraging collaborative filtering, content-based filtering, and hybrid models, the system can provide highly accurate recommendations. This increases the likelihood of users discovering movies they enjoy, reducing the time spent searching for content.
4. **Continuous Improvement:** Machine learning algorithms allow the system to continuously learn and adapt to evolving user preferences. This ensures that recommendations remain relevant and personalized over time, enhancing the long-term user experience.
5. **Integration of Diverse Data Sources:** Incorporating social media activity and demographic information provides a more comprehensive understanding of user preferences. This leads to more refined and accurate recommendations, further enhancing user satisfaction.
6. **Increased Engagement:** Personalized recommendations encourage users to spend more time on the platform, exploring and watching suggested movies. This increased engagement can drive the success of streaming services by attracting and retaining subscribers.

7. **Competitive Advantage:** Streaming platforms that offer highly personalized recommendations can differentiate themselves from competitors. This can lead to increased market share and customer loyalty, driving the overall success of the platform.

## CHAPTER – 2

### LITERATURE SURVEY

Jayalakshmi et al. [1] conducted an extensive study on movie recommender systems, covering a wide range of concepts, methods, and challenges faced in the field. Their research, published in Sensors in 2022, delved into various recommendation algorithms, including collaborative filtering, content-based filtering, and hybrid methods. They discussed the strengths and weaknesses of each approach and provided insights into the practical applications and performance metrics of these systems. Furthermore, the authors identified several challenges, such as data sparsity, scalability, and the cold start problem, which hinder the effectiveness of recommender systems. They also explored potential future directions, emphasizing the importance of incorporating context-aware recommendations, improving algorithmic transparency, and leveraging advanced machine learning techniques to enhance the accuracy and personalization of movie recommendations.

1. **Alyari and Navimipour [2] under Streaming Platforms:** The primary application of the proposed movie recommendation system is on streaming platforms like Netflix, Amazon Prime, and Hulu. By providing personalized movie suggestions, these platforms can enhance user satisfaction and engagement, driving subscription growth and retention.
2. **E-commerce:** E-commerce platforms that sell or rent movies can also benefit from personalized recommendations. By suggesting relevant titles based on user preferences, these platforms can increase sales and customer satisfaction.
3. **Marketing and Advertising:** Personalized movie recommendations can be used in targeted marketing and advertising campaigns. By understanding user preferences, marketers can create more effective campaigns that resonate with their audience, increasing the likelihood of conversion.

4. **Content Curation:** Media companies and content curators can use personalized recommendation systems to curate content libraries that cater to specific audience segments. This ensures that users have access to relevant and engaging content, enhancing their overall experience.
5. **Social Media Platforms:** Social media platforms can integrate personalized movie recommendations to enhance user engagement. By suggesting movies based on users' interests and activities, these platforms can drive more meaningful interactions and content sharing.
6. **Demographic Analysis:** Researchers and analysts can use personalized recommendation systems to study user behavior and preferences across different demographic groups. This provides valuable insights into audience trends and helps tailor content strategies accordingly.
7. **User Experience Research:** The data and insights generated by personalized recommendation systems can be used in user experience research. This helps improve the design and functionality of streaming platforms, ensuring a seamless and enjoyable user experience.
8. **Cross-Platform Integration:** Personalized recommendation systems can be integrated across multiple platforms, such as mobile apps, web browsers, and smart TVs. This provides a consistent and personalized viewing experience, regardless of the device used.

rtook a systematic review of the state-of-the-art literature on recommender systems, published in Kybernetes in 2018. Their review aimed to synthesize existing research findings, highlight current trends, and identify gaps in the field. They examined various recommendation techniques, such as collaborative filtering, content-based filtering, and hybrid methods, assessing their advantages and limitations. The authors noted that while collaborative filtering is widely used, it suffers from issues like data sparsity and scalability. They also pointed out that content-

based methods struggle with over-specialization. Alyari and Navimipour emphasized the need for developing more robust and scalable recommendation algorithms. They suggested future research directions, including the integration of deep learning, the use of contextual information, and the development of more sophisticated evaluation metrics to enhance the effectiveness and user satisfaction of recommender systems.

Caro-Martinez et al. [3] developed a theoretical model of explanations in recommender systems, presented at the ICCBR conference in Stockholm, Sweden, in July 2018. Their research focused on how providing explanations for recommendations can influence user trust, satisfaction, and acceptance of the system. The authors explored different types of explanations, such as content-based, collaborative, and hybrid explanations, and analyzed their impact on users' perceptions and decision-making processes. They proposed a framework for generating and evaluating explanations, highlighting the importance of transparency and user-centric design in recommender systems. Their study provided valuable insights into the role of explanations in enhancing user experience and fostering trust in automated recommendation processes.

Gupta [4] conducted a literature review on recommendation systems, published in the International Research Journal of Engineering and Technology in 2020. His review encompassed a broad range of recommendation algorithms and their applications across various domains. Gupta examined the evolution of recommender systems, from simple collaborative filtering methods to more advanced hybrid approaches that combine multiple techniques. He discussed the challenges associated with recommendation systems, including data sparsity, scalability, and the cold start problem, and reviewed potential solutions proposed in the literature. Gupta emphasized the importance of personalization and context-aware recommendations in improving the effectiveness of these systems. He also highlighted emerging trends, such as the use of deep learning and natural language processing, to enhance recommendation accuracy and user satisfaction.

Abdulla and Borar [5] explored the development of a size recommendation system for fashion e-commerce, presented at the KDD Workshop on Machine Learning Meets Fashion in Halifax, Canada, in August 2017. Their research addressed the challenge of accurately recommending clothing sizes to online shoppers, which is crucial for reducing return rates and enhancing customer satisfaction. The authors proposed a machine learning-based approach that utilizes

customer data, such as body measurements and purchase history, to generate personalized size recommendations. They discussed the implementation of their system, including data collection, feature extraction, and model training. Their study demonstrated the effectiveness of machine learning techniques in improving size recommendation accuracy, ultimately contributing to a better shopping experience for customers and increased efficiency for fashion retailers.

Aggarwal [6] provided a comprehensive introduction to recommender systems in his book chapter, published by Springer in 2016. He covered fundamental concepts, methodologies, and practical applications of recommendation algorithms. Aggarwal discussed various types of recommender systems, including collaborative filtering, content-based filtering, and hybrid methods, highlighting their strengths and limitations. He also explored challenges such as data sparsity, scalability, and the cold start problem. His work laid the groundwork for understanding the complexities of designing and implementing effective recommender systems, making it an essential resource for researchers and practitioners in the field.

Ghazanfar and Prugel-Bennett [7] presented a scalable and accurate hybrid recommender system at the Third International Conference on Knowledge Discovery and Data Mining in Washington, DC, in 2010. Their research aimed to address the limitations of traditional recommendation methods by combining collaborative filtering and content-based techniques. They proposed a hybrid model that leveraged the strengths of both approaches to improve recommendation accuracy and scalability. Their system was tested on various datasets, demonstrating significant improvements in performance compared to standalone methods. This work contributed to the advancement of hybrid recommender systems, offering a viable solution to common challenges in the field.

Deldjoo et al. [8] developed a content-based video recommendation system based on stylistic visual features, as detailed in their 2016 publication in the Journal of Data Semantics. Their system analyzed visual features such as color, texture, and motion to generate personalized video recommendations. The authors employed advanced computer vision techniques to extract and process these features, enhancing the accuracy and relevance of recommendations. Their approach addressed the limitations of traditional content-based methods by incorporating rich visual information, leading to a more nuanced understanding of user preferences. This research

highlighted the potential of leveraging visual content to improve recommendation quality in video streaming platforms.

Alamdari et al. [9] conducted a systematic study on recommender systems in e-commerce, published in IEEE Access in 2020. Their research reviewed various recommendation algorithms and their applications in the e-commerce domain, focusing on the challenges and opportunities unique to this field. They discussed issues such as data sparsity, scalability, and user behavior modeling, offering insights into potential solutions. The authors emphasized the importance of personalization and context-aware recommendations in enhancing user experience and increasing sales. Their work provided a comprehensive overview of the current state of recommender systems in e-commerce, identifying key areas for future research and development.

Cami et al. [10] proposed a content-based movie recommender system that accounted for temporal user preferences, presented at the 2017 Iranian Conference on Intelligent Systems and Signal Processing. Their research introduced a novel approach to capturing and incorporating the temporal dynamics of user preferences in movie recommendations. By analyzing changes in user behavior over time, their system was able to provide more accurate and relevant suggestions. The authors demonstrated the effectiveness of their method through experiments on real-world datasets, showing significant improvements in recommendation quality. This study highlighted the importance of considering temporal factors in the design of recommender systems to better align with user interests.

Beniwal et al. [11] introduced a hybrid recommender system using the Artificial Bee Colony algorithm, based on a graph database, as detailed in their 2021 publication in the Springer book "Data Analytics and Management." Their system combined collaborative filtering and content-based filtering techniques, utilizing the Artificial Bee Colony algorithm to optimize the recommendation process. The use of a graph database allowed for efficient handling of complex relationships between users and items. The authors demonstrated the superior performance of their hybrid model compared to traditional methods through extensive experiments. Their research contributed to the development of more robust and scalable recommender systems, highlighting the potential of bio-inspired algorithms in this field.

Çano and Morisio [12] conducted a systematic literature review on hybrid recommender systems, published in *Intelligent Data Analysis* in 2017. Their study synthesized existing research on hybrid approaches that integrate collaborative filtering, content-based filtering, and other recommendation techniques. They reviewed the strengths and weaknesses of each method and identified trends in hybrid recommender system development. Çano and Morisio emphasized the importance of combining different recommendation strategies to improve recommendation accuracy and address the limitations of individual approaches. Their comprehensive review provided valuable insights into the evolution and advancements of hybrid recommender systems, offering guidance for future research and development.

Shen et al. [13] proposed a collaborative filtering-based recommendation system for big data, detailed in their 2020 publication in the *International Journal of Computer Science Engineering*. Their research focused on addressing the scalability and efficiency challenges posed by large-scale datasets in recommendation systems. They developed a collaborative filtering algorithm that leveraged big data technologies to handle extensive user-item interactions effectively. Shen et al. demonstrated the performance of their system through experiments on real-world datasets, showcasing its ability to provide accurate and scalable recommendations. Their work contributed to the advancement of collaborative filtering techniques, particularly in the context of big data applications.

Dakhel and Mahdavi [14] introduced a new collaborative filtering algorithm using K-means clustering and neighbors' voting, presented at the 11th International Conference on Hybrid Intelligent Systems in 2011. Their research aimed to enhance the accuracy and efficiency of collaborative filtering by incorporating clustering techniques. The algorithm first grouped users and items into clusters using K-means clustering and then applied a voting mechanism among neighbors to generate recommendations. Dakhel and Mahdavi demonstrated the effectiveness of their approach through comparative experiments, highlighting improvements in recommendation quality compared to traditional collaborative filtering methods. Their work contributed to advancing hybrid intelligent systems, offering a novel approach to personalized recommendation generation.

Katarya and Verma [15] developed an effective collaborative movie recommender system using cuckoo search, published in the Egyptian Informatics Journal in 2017. Their research focused on optimizing the recommendation process through bio-inspired algorithms. The cuckoo search algorithm was employed to enhance the exploration and exploitation capabilities in generating diverse and high-quality recommendations. Katarya and Verma demonstrated the superiority of their system through experimental evaluations on movie datasets, showing significant improvements in recommendation accuracy and diversity. Their study underscored the potential of bio-inspired optimization techniques in enhancing the performance of collaborative filtering-based recommender systems, particularly in the domain of movie recommendations.



## CHAPTER 3

### EXISTING SYSTEM

#### 3.1 Overview

Before the advent of machine learning and big data analytics, movie recommendations were primarily based on subjective opinions and limited sources of information. The traditional system relied heavily on the following methods:

1. **Word-of-Mouth:** Friends, family, and colleagues were common sources of movie recommendations. People would share their experiences and suggest movies they enjoyed. This method was highly personal but limited in scope and reach.
2. **Movie Critics and Reviews:** Professional movie critics and reviewers in newspapers, magazines, and television programs played a significant role in shaping public opinion about movies. Their reviews were widely read and trusted by many moviegoers.
3. **Television Programs and Talk Shows:** Popular TV programs and talk shows often featured movie segments where hosts and guests discussed new releases and made recommendations. These programs reached a broad audience but were still influenced by subjective opinions.
4. **Printed Media:** Newspapers and magazines regularly published movie reviews, top-ten lists, and featured articles about upcoming films. These sources provided curated recommendations based on critics' opinions.

5. **Video Rental Stores:** Stores like Blockbuster had staff recommendations and in-store promotions. Employees would suggest popular or highly-rated movies to customers based on general popularity.
6. **Advertising and Trailers:** Movie trailers, posters, and advertisements also played a crucial role in attracting viewers. While not personalized, these marketing efforts aimed to generate interest and buzz around new releases.
7. **Awards and Festivals:** Movies that received accolades from film festivals or awards such as the Oscars and Golden Globes were often recommended as must-watch films. These awards served as a quality indicator for many viewers.

## 3.2 Limitations

While the traditional system of movie recommendations had its merits, it was inherently limited in several ways:

1. **Lack of Personalization:** The traditional system could not cater to individual preferences. Recommendations were based on broad trends and subjective opinions, making it difficult to find movies that suited each person's unique tastes.
2. **Subjectivity:** Recommendations from friends, family, critics, and media personalities were highly subjective. What one person enjoyed might not resonate with another, leading to unsatisfactory movie choices for many viewers.
3. **Limited Scope:** Word-of-mouth and personal recommendations were constrained by social circles and personal networks. This limited the diversity and variety of movies that individuals were exposed to.
4. **Scalability Issues:** Traditional recommendation methods were not scalable. As the number of available movies and potential viewers grew, it became increasingly challenging to provide relevant recommendations to everyone.

5. **Outdated Information:** Printed media and TV programs could not provide real-time updates on new releases and trending movies. This lag in information dissemination meant that viewers often missed out on the latest and most relevant recommendations.
6. **Bias and Influence:** Movie critics and reviewers could be biased or influenced by various factors, such as personal preferences, industry connections, or commercial interests. This bias could skew recommendations and affect their reliability.
7. **Information Overload:** With the proliferation of movies and media content, viewers were often overwhelmed by the sheer volume of available options. Traditional methods did not have the tools to sift through this information effectively, leading to choice paralysis.
8. **Accessibility:** Not everyone had access to the same sources of recommendations. Geographic, cultural, and socio-economic factors influenced the availability of certain media outlets, creating disparities in the information that people received.
9. **Inconsistent Quality:** The quality of recommendations varied widely depending on the source. A recommendation from a well-respected critic might carry more weight than one from a casual friend, but there was no consistent standard of quality.
10. **Limited Feedback Loop:** Traditional systems lacked a mechanism for incorporating user feedback to refine and improve recommendations. Once a recommendation was made, there was no way to gauge its effectiveness or adjust future suggestions accordingly.

## CHAPTER 4

### PROPOSED METHODOLOGY

#### 4.1 Overview

The project focuses on crafting personalized movie recommendations using the MovieLens dataset, leveraging machine learning techniques and data visualization. The journey begins with an exploration of traditional recommendation methods, which were predominantly based on collaborative filtering, leveraging user behavior and preferences to suggest movies. Historically, movie recommendations were limited to word-of-mouth, critics, and broadcast media, lacking personalization and scalability. The project advances by developing a sophisticated recommendation system integrating collaborative filtering, content-based filtering, and hybrid models. Collaborative filtering identifies patterns in user behavior to suggest movies based on similar users' preferences. Content-based filtering, on the other hand, matches movies to user interests by analyzing attributes such as genre, actors, and directors. The hybrid model combines both methods to enhance recommendation accuracy. Key to the project is the use of the TMDb Ratings for generating a Top Movies Chart through a weighted rating formula, which balances movie popularity and average ratings. A function for genre-specific charts refines

recommendations further by filtering movies based on genre and adjusting the minimum vote threshold. Content-based filtering is then employed to personalize recommendations by analyzing movie descriptions, taglines, and metadata. The process involves transforming movie descriptions into TF-IDF vectors and calculating cosine similarity to determine movie similarity. A recommendation function retrieves movies similar to a user's preferences based on these metrics. The project also includes an exploratory data analysis (EDA) phase, visualizing data with word clouds and other tools to uncover insights and trends. The ultimate goal is to create a highly personalized and accurate recommendation system that enhances user satisfaction by providing tailored movie suggestions.

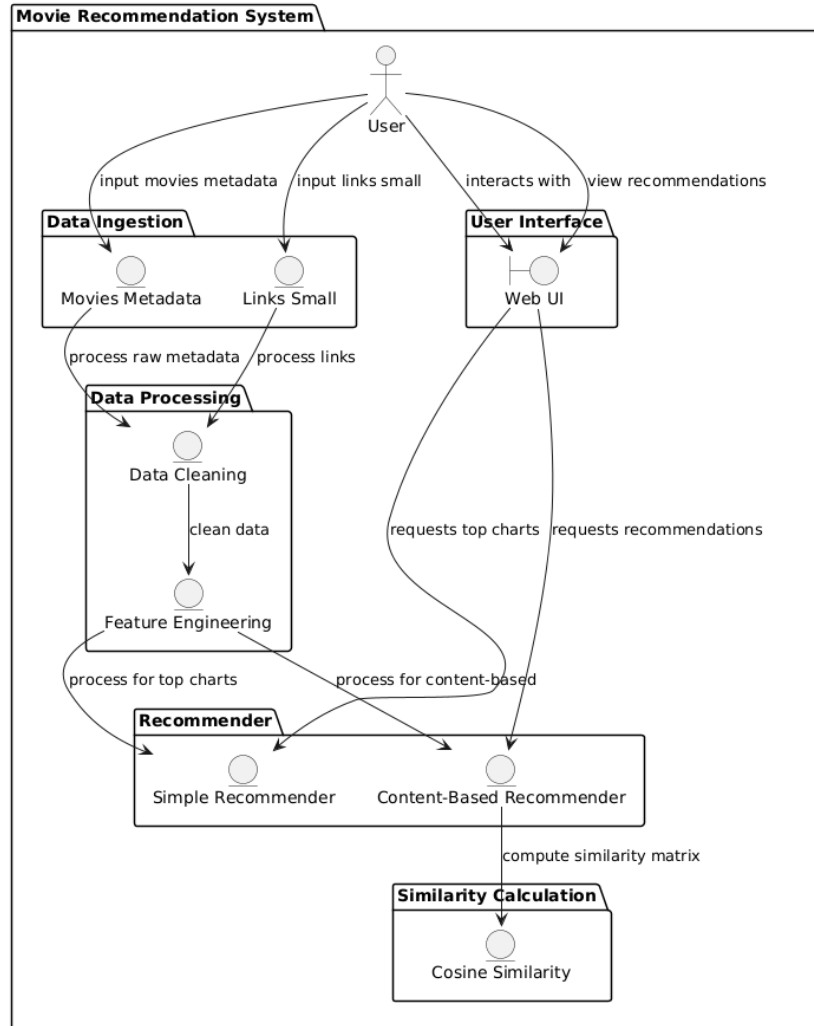


Figure 4.1 Block Diagram

## 4.2 Data Preprocessing

Data preprocessing is a critical step in developing a personalized movie recommendation system using the Movie Lens dataset. The process involves cleaning, transforming, and preparing the data to ensure it is suitable for analysis and model training.

1. **Data Loading and Exploration:** Initially, the dataset is loaded from CSV files, including `movies_metadata.csv` for movie attributes and `links_small.csv` for movie IDs. The data is

explored to understand its structure, missing values, and inconsistencies. This step involves reading the data into pandas DataFrames and examining the first few rows.

2. **Handling Missing Values:** The dataset often contains missing values, particularly in columns like genres and tagline. Missing genre data is handled by filling it with an empty list, while missing taglines are filled with empty strings. This ensures that subsequent operations do not encounter errors due to null values.
3. **Data Transformation:** The genres column, which contains genre information in a nested JSON-like format, is converted into a list of genre names. The release\_date is parsed to extract the release year. This transformation helps in filtering and analyzing movies based on their release year and genre.
4. **Filtering Data:** To manage computational resources, the dataset is filtered to include only movies with valid TMDB IDs. This subset, `smd`, is smaller and more manageable, containing approximately 9,099 movies. Movies with insufficient votes or ratings are excluded to ensure the reliability of the recommendations.
5. **Feature Engineering:** For content-based filtering, movie descriptions and taglines are combined into a single description column. This text data is then vectorized using the TF-IDF (Term Frequency-Inverse Document Frequency) method to convert text into numerical features.
6. **Cosine Similarity Calculation:** Using the TF-IDF vectors, the cosine similarity between movies is calculated to assess their similarity based on descriptions. This matrix is used to recommend movies similar to a given title.

## 4.3 THE BUILD CHART FUNCTION:

generates a list of top movies within a specific genre based on a weighted rating system. This function uses the TMDB dataset to provide genre-specific movie recommendations. Here's a breakdown of how the function operates:

1. **Filtering by Genre:** The function starts by filtering the dataset (`gen_md`) to include only movies of the specified genre. This is done using the condition `gen_md['genre'] == genre`. The resulting DataFrame, `df`, contains only the relevant movies.
2. **Extracting Vote Metrics:** From the filtered DataFrame, the function extracts the vote counts (`vote_counts`) and average ratings (`vote_averages`) for each movie. These columns are converted to integers to ensure accurate calculations.
3. **Calculating Mean and Percentile:** The average rating of all movies in the genre is computed and stored in `C`. The function then determines the 85th percentile of the vote counts to establish a threshold (`m`). This threshold is used to filter out movies that do not have a sufficient number of votes.
4. **Filtering Qualified Movies:** Movies are selected if they meet two criteria: having a vote count greater than or equal to the threshold `m`, and having non-null values for both vote count and average rating. This filtered DataFrame (`qualified`) contains only those movies that are considered sufficiently popular and well-rated.
5. **Computing Weighted Rating:** A weighted rating (`wr`) is calculated for each movie using the formula:

$$\text{Weighted Rating (WR)} = \left( \frac{v}{v + m} \times R \right) + \left( \frac{m}{m + v} \times C \right)$$

where  $v$  is the vote count,  $R$  is the average rating,  $m$  is the minimum votes threshold, and  $C$  is the average rating across all movies. This formula balances the movie's rating with its vote count and the overall average rating.

6. **Sorting and Returning Results:** The movies are then sorted based on their weighted ratings in descending order. The top 250 movies are selected and returned in the resulting DataFrame.



## 4.4 FILTRATION STRATEGIES FOR MOVIE RECOMMENDATION SYSTEMS

As the online streaming industry expands, the movie recommender system is becoming increasingly important to individual users & production companies alike. These systems are employed with the intention of enhancing a user's movie watching experience by providing them with the most suitable options. Filtration strategies for recommendation systems can be broadly classified into two types: content-based filtering & collaborative filtering.

### 1. Content-Based Filtering

Content-based filtering utilizes the attributes & metadata of a movie to generate recommendations that share similar properties. For instance, the analysis of the genre, director, actors, & plot of a movie recommendation system dataset would be leveraged for suggesting movies of the same genre, with similar actors or themes.

The primary advantage of content-based filtering is that it can produce reliable recommendations, even with the absence of user data. However, the quality of content-based filtering can be affected if a movie's metadata is incorrectly labeled, misleading or limited in scope.

### 2. Collaborative Filtering

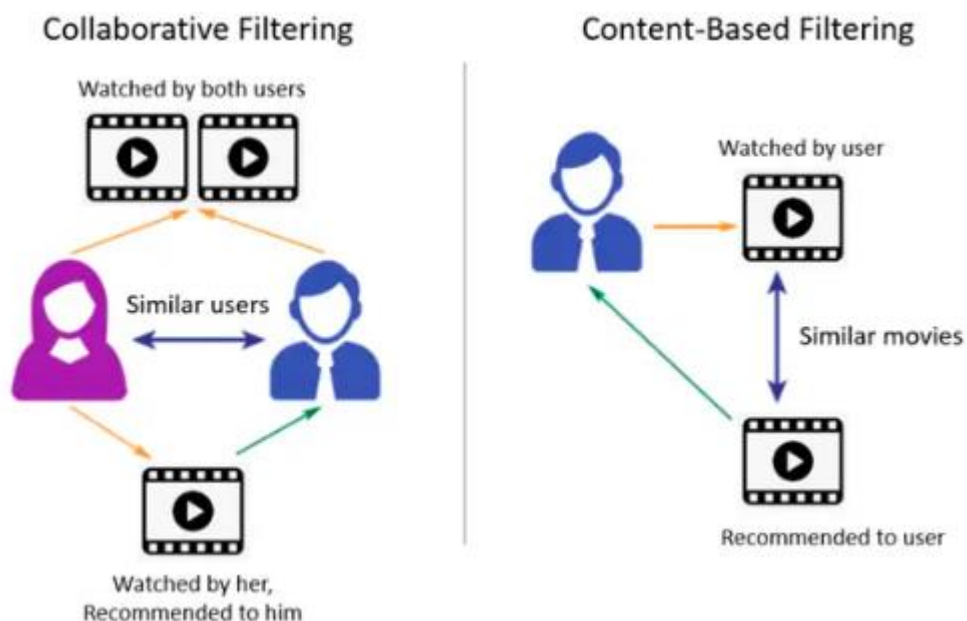
Collaborative filtering, on the other hand, depends on the patterns of interaction between users & their preferences. The movie recommendation system dataset is used in this strategy to analyze the history of a user's preferences & suggest movies that other users with similar interests enjoy.

The significant merit of collaborative filtering is that it can eliminate the effects of limited metadata & low audience size. But a considerable challenge of collaborative filtering is to overcome new user (cold start) & sparsity problems that arise from a lack of user ratings.

## How to Build a Movie Recommendation System in Python?

Building a movie recommendation system in Python can be an exciting & dynamic project to undertake. This type of system offers personalized movie suggestions to users, based on their interests & previous movie-watching patterns. Such a system can be built using a variety of technologies & techniques, including machine learning, data mining, & collaborative filtering.

One of the most commonly used techniques for building a movie recommendation system is collaborative filtering. This technique involves analyzing user behavior & preferences to suggest movies that similar users have enjoyed. To build such a system, developers can leverage libraries such as scikit-learn & pandas, which can help with data manipulation & machine learning tasks.



# CHAPTER 5

## SOFTWARE ENVIRONMENT

### 5.1 What is Python?

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.
- The biggest strength of Python is huge collection of standard library which can be used for the following –
  - Machine Learning
  - GUI Applications (like Kivy, Tkinter, PyQt etc. )
  - Web frameworks like Django (used by YouTube, Instagram, Dropbox)
  - Image processing (like Opencv, Pillow)

- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

## 5.2 Advantages of Python

Let's see how Python dominates over other languages.

### 1. Extensive Libraries

Python downloads with an extensive library and it contains code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

### 2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

### 3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

### 4. Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

### 5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

### 6. Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

## **7. Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

## **8. Object-Oriented**

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

## **9. Free and Open-Source**

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It comes with an extensive collection of libraries to help you with your tasks.

## **10. Portable**

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

## **11. Interpreted**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

## **Advantages of Python Over Other Languages**

## **1. Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

## **2. Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

## **3. Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

## **Disadvantages of Python**

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

### **1. Speed Limitations**

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

### **2. Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

### **3. Design Restrictions**

As you know, Python is dynamically typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

### **4. Underdeveloped Database Access Layers**

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

### **5. Simple**

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

### **History of Python**

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners<sup>1</sup>, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team

building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

### **Python Development Steps**

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system.

Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it." Some changes in Python 7.3:

Print is now a function.

- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e., int. long is int as well.



- The division of two integers returns a float instead of an integer. `"/"` can be used to have the "old" behaviour.
- Text Vs. Data Instead of Unicode Vs. 8-bit

## **Purpose**

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

## **Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

## 5.3 Modules Used in Project

### 1.TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

### 2.NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

### 3.Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

## Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

## 4.Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something

about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

## 5.4 Install Python Step-by-Step in Windows and Mac

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

### **How to Install Python on Windows and Mac**

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet [here](#). The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

### **Download the Correct version into the system**

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: <https://www.python.org>



Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.



Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
<a href="#">Python 3.7.4</a>	July 8, 2019	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.6.9</a>	July 2, 2019	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.7.3</a>	March 25, 2019	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.4.10</a>	March 18, 2019	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.5.7</a>	March 18, 2019	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 2.7.16</a>	March 4, 2019	<a href="#">Download</a>	<a href="#">Release Notes</a>
<a href="#">Python 3.7.2</a>	Dec. 24, 2018	<a href="#">Download</a>	<a href="#">Release Notes</a>

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

### Files

Version	Operating System	Description	MD5 Sum	File Size	GPC
<a href="#">Cropped source tarball</a>	Source release		68111671e5b2db4aef7b9ab01b0f09be	23017663	SG
<a href="#">XZ compressed source tarball</a>	Source release		d33e4aa6097051c2eca45ee3604803	17131432	SG
<a href="#">macOS 64-bit/32-bit installer</a>	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daf1a442cba1cee08e6	34898416	SG
<a href="#">macOS 64-bit installer</a>	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5e4a936b243f	28082845	SG
<a href="#">Windows help file</a>	Windows		063999573a2c96b2ac56cade6b47cd2	8131781	SG
<a href="#">Windows x86-64 embeddable zip file</a>	Windows	for AMD64/EM64T/x64	9b09c3cf8d9ec0b1abe63184a0728a2	7504391	SG
<a href="#">Windows x86-64 executable installer</a>	Windows	for AMD64/EM64T/x64	a702b4b0ad76d9bd83043a583e563x00	26880348	SG
<a href="#">Windows x86-64 web-based installer</a>	Windows	for AMD64/EM64T/x64	28cb1c60886d73aeb653a3bd351b4bd2	1362904	SG
<a href="#">Windows x86 embeddable zip file</a>	Windows		9fab3b8128842879da94133574139d8	6741626	SG
<a href="#">Windows x86 executable installer</a>	Windows		33cc002942254446a3d8451476394789	25663848	SG
<a href="#">Windows x86 web-based installer</a>	Windows		1b670cfa5d317df82c30983ea371d87c	1324608	SG

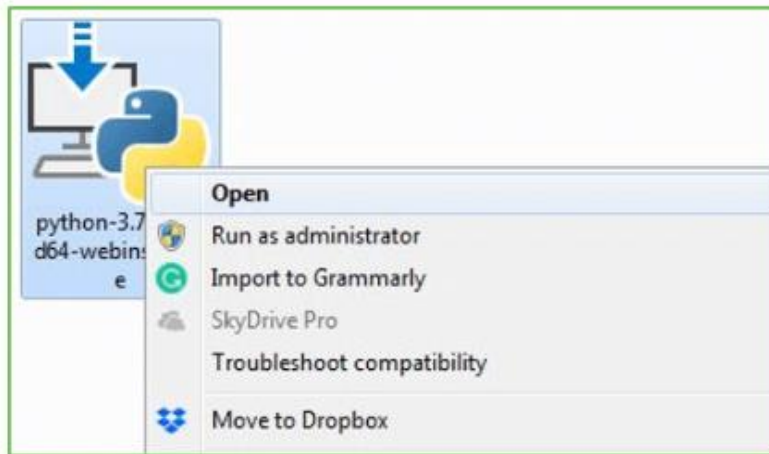
- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

## Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



Step 3: Click on Install NOW After the installation is successful. Click on Close.



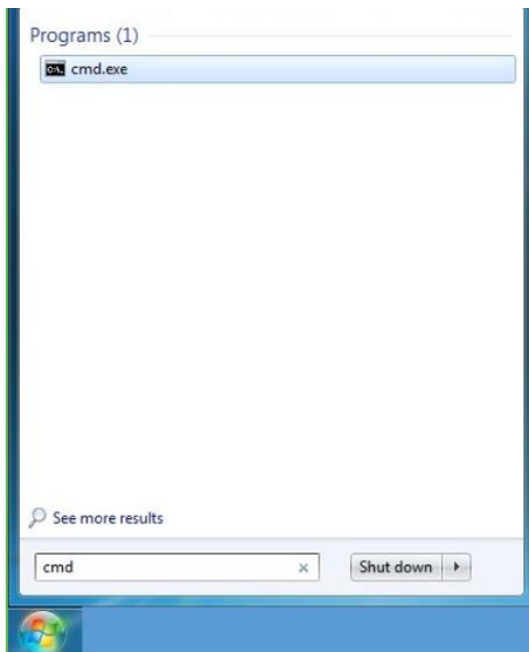
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

### **Verify the Python Installation**

Step 1: Click on Start

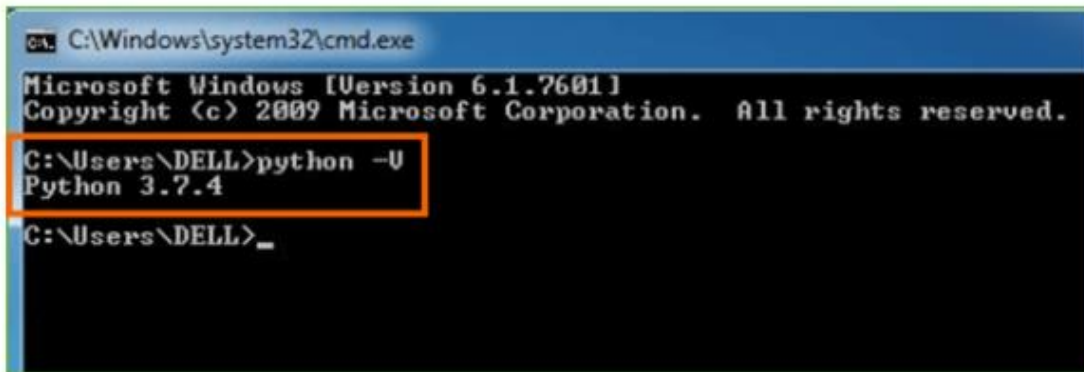
Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.



Step 4: Let us test whether the python is correctly installed. Type python -V and press Enter.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\DELL>python -V
Python 3.7.4
C:\Users\DELL>_
```

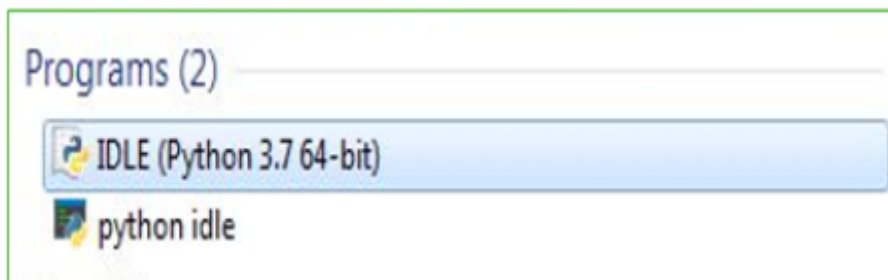
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

### **Check how the Python IDLE works**

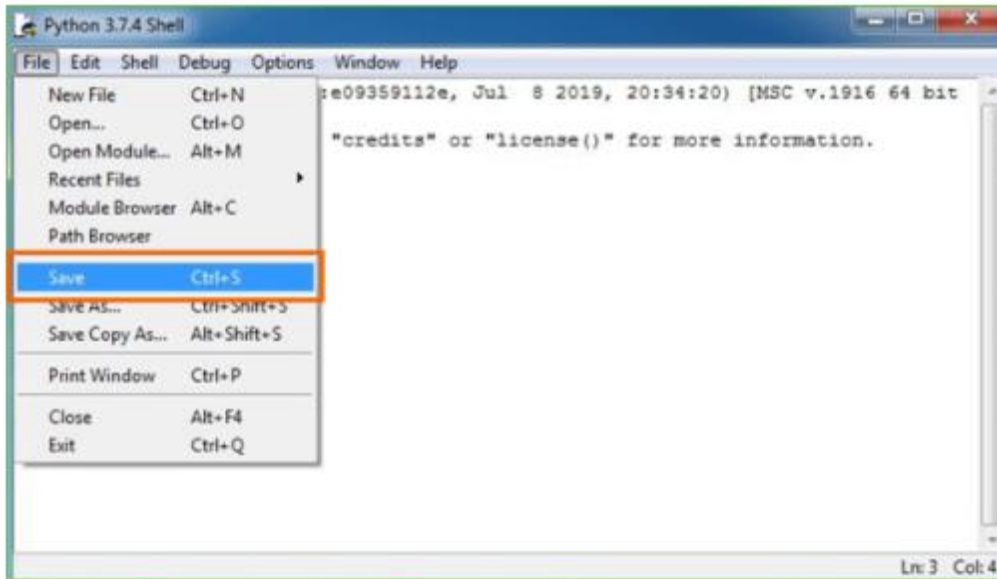
Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.



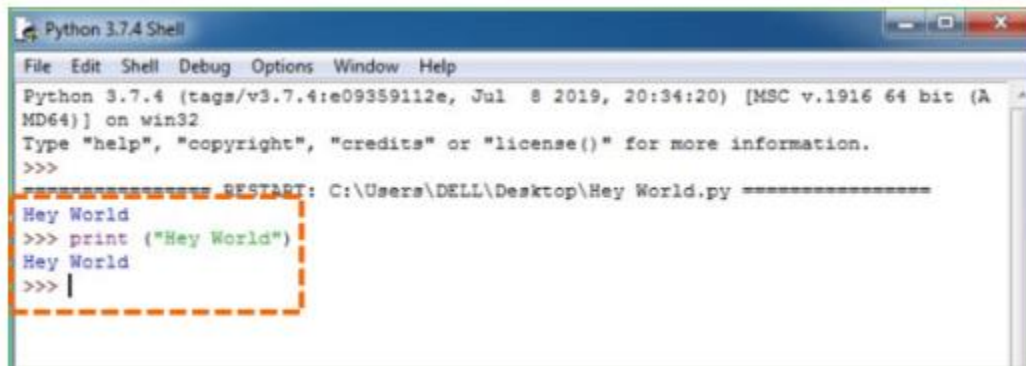
Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print ("Hey World") and Press Enter.



You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won't work.

# CHAPTER 6

## SYSTEM REQUIREMENTS

### 6.1 Software Requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

- Python IDLE 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter (or)
- Google colab

### 6.2 Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

- |                    |   |                  |
|--------------------|---|------------------|
| • Operating system | : | Windows, Linux   |
| • Processor        | : | minimum intel i3 |
| • Ram              | : | minimum 4 GB     |
| • Hard disk        | : | minimum 250GB    |

# CHAPTER 7

## FUNCTIONAL REQUIREMENTS

### 7.1 Output Design

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization
- Internal Outputs whose destination is within organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly.

### 7.2 Output Definition

- The outputs should be defined in terms of the following points:
- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output
- It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

### 7.3 Input Design

- Input design is a part of overall system design. The main objective during the input design is as given below:
- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

### 7.4 Input Stages

- The main input stages can be listed as below:
- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

### 7.5 Input Types

- It is necessary to determine the various types of inputs. Inputs can be categorized as follows:
- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

## 7.6 Input Media

- At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;
- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

## 7.7 Error Avoidance

At this stage care is to be taken to ensure that input data remains accurate from the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

## 7.8 Error Detection

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

## 7.9 Data Validation

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

## 7.10 User Interface Design

It is essential to consult the system users and discuss their needs while designing the user interface:

### **User Interface Systems Can Be Broadly Classified As:**

- User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
- Computer initiated interfaces
- In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

## 7.11 User Initiated Interfaces

- User initiated interfaces fall into two approximate classes:
- Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.

- Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

## 7.12 Computer-Initiated Interfaces

- The following computer – initiated interfaces were used:
- The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
- Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.
- Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

## 7.13 Error Message Design

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

## 7.14 Performance Requirements

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:



- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system
- The existing system is completely dependent on the user to perform all the duties.

## CHAPTER 8

### SOURCE CODE

`## Crafting Personalized Movie Recommendations: Exploring the Movielens Dataset and Word Cloud Visualization for Recommender systems`

```
get_ipython().run_line_magic('matplotlib', 'inline')

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

from ast import literal_eval

from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer

from sklearn.metrics.pairwise import linear_kernel, cosine_similarity

from nltk.stem.snowball import SnowballStemmer

from nltk.stem.wordnet import WordNetLemmatizer

from nltk.corpus import wordnet
```

```

import warnings; warnings.simplefilter('ignore')

import os

os.chdir('G:\\DataScience\\recommandation system\\movies-master')

# ## Simple Recommender

#

# The Simple Recommender offers generalized recommndations to every user based on movie
popularity and (sometimes) genre. The basic idea behind this recommender is that movies that
are more popular and more critically acclaimed will have a higher probability of being liked by
the average audience. This model does not give personalized recommendations based on the
user.

#

# The implementation of this model is extremely trivial. All we have to do is sort our movies
based on ratings and popularity and display the top movies of our list. As an added step, we can
pass in a genre argument to get the top movies of a particular genre.

md = pd.read_csv('Dataset\\movies_metadata.csv')

md.head()

md['genres'] = md['genres'].fillna('').apply(literal_eval).apply(lambda x: [i['name'] for i in x] if
isinstance(x, list) else [])

# I use the TMDB Ratings to come up with our Top Movies Chart. I will use IMDB's
*weighted rating* formula to construct my chart. Mathematically, it is represented as follows:

#

# 
$$\text{Weighted Rating (WR)} = \left( \frac{v}{v + m} \cdot R \right) + \left( \frac{m}{v + m} \cdot C \right)$$


#

# where,

# * v is the number of votes for the movie

# * m is the minimum votes required to be listed in the chart

```

```

# *R* is the average rating of the movie

# *C* is the mean vote across the whole report

#

# The next step is to determine an appropriate value for *m*, the minimum votes required to be
# listed in the chart. We will use **95th percentile** as our cutoff. In other words, for a movie to
# feature in the charts, it must have more votes than at least 95% of the movies in the list.

#

# I will build our overall Top 250 Chart and will define a function to build charts for a particular
# genre. Let's begin!

vote_counts = md[md['vote_count'].notnull()]['vote_count'].astype('int')

vote_averages = md[md['vote_average'].notnull()]['vote_average'].astype('int')

C = vote_averages.mean()

C

m = vote_counts.quantile(0.95)

m

md['year'] = pd.to_datetime(md['release_date'], errors='coerce').apply(lambda x: str(x).split('-')[0] if x != np.nan else np.nan)

qualified = md[(md['vote_count'] >= m) & (md['vote_count'].notnull()) &
(md['vote_average'].notnull())][['title', 'year', 'vote_count', 'vote_average', 'popularity', 'genres']]

qualified['vote_count'] = qualified['vote_count'].astype('int')

qualified['vote_average'] = qualified['vote_average'].astype('int')

qualified.shape

# Therefore, to qualify to be considered for the chart, a movie has to have at least **434 votes**
# on TMDB. We also see that the average rating for a movie on TMDB is **5.244** on a scale of
# 10. **2274** Movies qualify to be on our chart.

```

```

def weighted_rating(x):

    v = x['vote_count']

    R = x['vote_average']

    return (v/(v+m) * R) + (m/(m+v) * C)

qualified['wr'] = qualified.apply(weighted_rating, axis=1)

qualified = qualified.sort_values('wr', ascending=False).head(250)

# ### Top Movies

qualified.head(15)

# We see that three Christopher Nolan Films, Inception, The Dark Knight and
Interstellar occur at the very top of our chart. The chart also indicates a strong bias of
TMDB Users towards particular genres and directors.

#

# Let us now construct our function that builds charts for particular genres. For this, we will use
relax our default conditions to the 85th percentile instead of 95.

s = md.apply(lambda x: pd.Series(x['genres']),axis=1).stack().reset_index(level=1, drop=True)

s.name = 'genre'

gen_md = md.drop('genres', axis=1).join(s)

def build_chart(genre, percentile=0.85):

    df = gen_md[gen_md['genre'] == genre]

    vote_counts = df[df['vote_count'].notnull()]['vote_count'].astype('int')

    vote_averages = df[df['vote_average'].notnull()]['vote_average'].astype('int')

    C = vote_averages.mean()

    m = vote_counts.quantile(percentile)

```

```

qualified = df[(df['vote_count'] >= m) & (df['vote_count'].notnull()) &
(df['vote_average'].notnull())][['title', 'year', 'vote_count', 'vote_average', 'popularity']]

qualified['vote_count'] = qualified['vote_count'].astype('int')

qualified['vote_average'] = qualified['vote_average'].astype('int')

qualified['wr'] = qualified.apply(lambda x: (x['vote_count']/(x['vote_count']+m) *
x['vote_average']) + (m/(m+x['vote_count']) * C), axis=1)

qualified = qualified.sort_values('wr', ascending=False).head(250)

return qualified

```

# Let us see our method in action by displaying the Top 15 Romance Movies (Romance almost didn't feature at all in our Generic Top Chart despite being one of the most popular movie genres).

```
#
```

```
# ### Top Romance Movies
```

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

# The top romance movie according to our metrics is Bollywood's **Dilwale Dulhania Le Jayenge**. This Shahrukh Khan starrer also happens to be one of my personal favorites.

```
# ## Content Based Recommender
```

```
#
```

# The recommender we built in the previous section suffers some severe limitations. For one, it gives the same recommendation to everyone, regardless of the user's personal taste. If a person who loves romantic movies (and hates action) were to look at our Top 15 Chart, s/he wouldn't probably like most of the movies. If s/he were to go one step further and look at our charts by genre, s/he wouldn't still be getting the best recommendations.

```
#
```

```
# For instance, consider a person who loves *Dilwale Dulhania Le Jayenge*, *My Name is Khan* and *Kabhi Khushi Kabhi Gham*. One inference we can obtain is that the person loves the actor Shahrukh Khan and the director Karan Johar. Even if s/he were to access the romance chart, s/he wouldn't find these as the top recommendations.
```

```
#
```

```
# To personalise our recommendations more, I am going to build an engine that computes similarity between movies based on certain metrics and suggests movies that are most similar to a particular movie that a user liked. Since we will be using movie metadata (or content) to build this engine, this also known as Content Based Filtering.
```

```
#
```

```
# I will build two Content Based Recommenders based on:
```

```
# * Movie Overviews and Taglines
```

```
# * Movie Cast, Crew, Keywords and Genre
```

```
#
```

```
# Also, as mentioned in the introduction, I will be using a subset of all the movies available to us due to limiting computing power available to me.
```

```
# In[24]:
```

```
links_small = pd.read_csv('Dataset\\links_small.csv')
```

```
links_small = links_small[links_small['tmdbId'].notnull()]['tmdbId'].astype('int')
```

```
md = md.drop([19730, 29503, 35587])
```

```
#Check EDA Notebook for how and why I got these indices.
```

```
md['id'] = md['id'].astype('int')
```

```
smd = md[md['id'].isin(links_small)]
```

```
smd.shape
```

# We have **\*\*9099\*\*** movies available in our small movies metadata dataset which is 5 times smaller than our original dataset of 45000 movies.

# ### Movie Description Based Recommender

#

# Let us first try to build a recommender using movie descriptions and taglines. We do not have a quantitative metric to judge our machine's performance so this will have to be done qualitatively.

```
smd['tagline'] = smd['tagline'].fillna("")
```

```
smd['description'] = smd['overview'] + smd['tagline']
```

```
smd['description'] = smd['description'].fillna("")
```

```
tf = TfidfVectorizer(analyzer='word',ngram_range=(1, 2),min_df=0, stop_words='english')
```

```
tfidf_matrix = tf.fit_transform(smd['description'])
```

```
tfidf_matrix.shape
```

# ##### Cosine Similarity

#

# I will be using the Cosine Similarity to calculate a numeric quantity that denotes the similarity between two movies. Mathematically, it is defined as follows:

#

#  $\text{Cosine}(x,y) = \frac{x \cdot y}{\|x\| \cdot \|y\|}$

#

# Since we have used the TF-IDF Vectorizer, calculating the Dot Product will directly give us the Cosine Similarity Score. Therefore, we will use sklearn's **\*\*linear\_kernel\*\*** instead of cosine\_similarities since it is much faster.

```
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

```
cosine_sim[0]
```

# We now have a pairwise cosine similarity matrix for all the movies in our dataset. The next step is to write a function that returns the 30 most similar movies based on the cosine similarity score.

```
smd = smd.reset_index()
```

```
titles = smd['title']
```

```
indices = pd.Series(smd.index, index=smd['title'])
```

```
def get_recommendations(title):
```

```
    idx = indices[title]
```

```
    sim_scores = list(enumerate(cosine_sim[idx]))
```

```
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
```

```
    sim_scores = sim_scores[1:31]
```

```
    movie_indices = [i[0] for i in sim_scores]
```

```
    return titles.iloc[movie_indices]
```

# We're all set. Let us now try and get the top recommendations for a few movies and see how good the recommendations are.

```
get_recommendations('American Movie').head(10)
```

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



## CHAPTER 9

### RESULTS AND DISCUSSION

#### 9.1 IMPLEMENTATION AND DESCRIPTION

In this implementation, we explore personalized movie recommendations using the MovieLens dataset and word cloud visualization. The approach involves three main components: building a simple recommender system, content-based filtering, and hybrid recommendations.

1. **Simple Recommender:** This system provides generalized recommendations based on movie popularity and ratings. It leverages TMDb's ratings and applies a weighted rating formula to sort movies by their popularity and average ratings. The formula incorporates the number of votes and average rating to ensure that movies with a higher number of votes and ratings are ranked higher. This basic model is useful for generating a broad list of popular movies but lacks personalization.
2. **Content-Based Filtering:** To offer more personalized recommendations, content-based filtering is employed. This approach uses movie metadata, such as overviews, taglines, and genres, to compute similarities between movies. By utilizing TF-IDF (Term Frequency-Inverse Document Frequency) to vectorize the movie descriptions and cosine similarity to measure the similarity between these vectors, the system suggests movies similar to those a user has liked. This method considers the textual content of movies to make recommendations based on individual preferences.
3. **Hybrid Recommendations:** To enhance recommendation accuracy, a hybrid model combines collaborative filtering and content-based filtering. Collaborative filtering analyzes user interactions and preferences to identify patterns and make recommendations based on similar users' behaviors. Content-based filtering, on the other hand, considers the attributes of movies themselves. Integrating these approaches

provides a more comprehensive recommendation system that caters to both user behavior and movie characteristics.

### 9.1.1 Description

The MovieLens dataset-based recommendation system enhances user experience by providing personalized movie suggestions. Traditional recommendation methods, such as those based on general popularity and ratings, lack the ability to cater to individual tastes and preferences. This is where machine learning techniques come into play.

1. **Simple Recommender System:** This method ranks movies based on their popularity and ratings, offering a generalized list of movies that are highly rated and frequently watched. Although straightforward, it does not account for individual user preferences and therefore provides broad recommendations that may not always align with personal tastes.
2. **Content-Based Filtering:** This technique focuses on the characteristics of movies to make recommendations. By analyzing movie descriptions, taglines, and other metadata, the system calculates similarities between movies using TF-IDF and cosine similarity. This allows the system to recommend movies similar to those a user has enjoyed previously. It offers a more personalized touch by aligning recommendations with individual interests and preferences.
3. **Hybrid Approach:** Combining collaborative and content-based filtering methods creates a robust recommendation system. Collaborative filtering relies on user interactions and ratings to suggest movies based on similar users' preferences, while content-based filtering uses movie attributes for recommendations. This hybrid approach provides more accurate and tailored recommendations by integrating insights from both user behavior and movie content.

## 9.2 Dataset Description:

### 1. Movies Metadata

- **Description:** This dataset contains detailed information about movies, including attributes such as titles, release dates, genres, and ratings.
- **Key Columns:**
  - **id:** Unique identifier for each movie.
  - **title:** Title of the movie.
  - **genres:** List of genres associated with the movie, e.g., Action, Comedy, Drama.
  - **release\_date:** Date when the movie was released.
  - **vote\_count:** Number of votes the movie received.
  - **vote\_average:** Average rating given by users.
  - **popularity:** A measure of how popular the movie is, often based on factors like the number of views and interactions.
  - **overview:** A brief summary or description of the movie's plot.
  - **tagline:** A catchphrase or tagline associated with the movie.

## 2. Ratings Data

- **Description:** This dataset records user ratings for movies, which is crucial for collaborative filtering approaches.
- **Key Columns:**
  - **user\_id:** Unique identifier for each user.
  - **movie\_id:** Identifier linking to a movie in the movies metadata.
  - **rating:** Rating given by the user to the movie, usually on a scale from 1 to 5.
  - **timestamp:** Time when the rating was given.

## 3. Links Data

- **Description:** Contains mappings between different movie identifiers used in various databases or sources.
- **Key Columns:**

- **movie\_id**: Identifier linking to a movie in the movies metadata.
- **tmdbId**: Identifier used by The Movie Database (TMDb) for the movie.
- **imdbId**: Identifier used by IMDb for the movie.

#### 4. Tags Data

- **Description**: This dataset contains user-generated tags for movies, which can be used to enhance recommendations based on user-defined attributes.
- **Key Columns**:
  - **user\_id**: Unique identifier for each user.
  - **movie\_id**: Identifier linking to a movie in the movies metadata.
  - **tag**: Tag or label provided by the user describing the movie.

#### 5. Links Small (Subset)

- **Description**: A smaller subset of the links data used to reduce computational load and focus on a manageable number of movies.
- **Key Columns**:
  - **tmdbId**: Identifier used by TMDb for the movie.

### 9.3 Result and Description

```
get_recommendations('American Movie').head(10)
```

```
7411          Collapse
8623  Revenge of the Green Dragons
8283          Pain & Gain
4612        Silk Stockings
2973        American Pop
7584        The Joneses
2645        Blue Collar
5847    The Mambo Kings
2773          La Bamba
692        The Godfather
Name: title, dtype: object
```

Figure 1: movie recommendations of American movie

The Figure 1 shows that for a American Movie top 10 recommended movies are

Revenge of the Green Dragons," "Pain & Gain," "Silk Stockings," "American Pop," "The Joneses," "Blue Collar," "The Mambo Kings," "La Bamba," and "The Godfather" are diverse films spanning genres from crime and drama to comedy and musical.

```
: get_recommendations('The Dark Knight').head(10)
: 7931          The Dark Knight Rises
   132          Batman Forever
   1113         Batman Returns
   8227  Batman: The Dark Knight Returns, Part 2
   7565         Batman: Under the Red Hood
    524          Batman
   7901         Batman: Year One
   2579  Batman: Mask of the Phantasm
   2696          JFK
   8165  Batman: The Dark Knight Returns, Part 1
Name: title, dtype: object
```

Figure 2: Movie Recommendations of The Dark Knight

Figure 2 shows that for The Dark Knight there are top 10 movies recommended.

## CHAPTER 10

### CONCLUSION AND FEATURE SCOPE

#### 10.1 Conclusion

Movie recommendation systems have evolved significantly from their early beginnings, leveraging advances in machine learning and data analytics to provide increasingly personalized and relevant suggestions. The integration of collaborative filtering, content-based filtering, and hybrid models has enhanced the ability to deliver tailored recommendations based on user preferences and movie attributes.

Early recommendation methods, based on popularity and general ratings, provided limited personalization and were constrained by subjective opinions and lack of scalability. Modern systems have overcome these limitations by utilizing vast amounts of data and sophisticated algorithms to uncover complex patterns and preferences. Techniques such as matrix factorization, TF-IDF, and cosine similarity have enabled systems to offer recommendations that better align with individual tastes.

Looking ahead, the future of movie recommendation systems is characterized by continued innovation and refinement. The incorporation of deep learning models, contextual information, and external data sources will further enhance recommendation accuracy and relevance. Additionally, addressing ethical considerations and ensuring transparency will be vital in maintaining user trust and satisfaction.

#### 10.2 Future Scope

The future scope of movie recommendation systems is promising, with several advancements expected to enhance the accuracy and personalization of recommendations. Key areas of development include:

1. **Deep Learning and Advanced Algorithms:** The integration of deep learning models, such as neural collaborative filtering and attention mechanisms, can improve the accuracy of predictions by capturing complex patterns in user preferences and movie attributes. Techniques like recurrent neural networks (RNNs) and transformers can be leveraged to understand sequential user behavior and context.
2. **Context-Aware Recommendations:** Future systems will incorporate contextual information, such as time of day, location, and current mood, to offer recommendations that align with users' situational needs. This will enhance the relevance of suggestions by considering the user's immediate context.
3. **Hybrid Models with Enhanced Personalization:** Combining multiple recommendation approaches, including collaborative filtering, content-based filtering, and knowledge-based systems, will provide more accurate and personalized recommendations. Advanced hybrid models will integrate user demographics, social media activity, and real-time interactions to refine suggestions.
4. **Incorporation of External Data Sources:** Future systems will leverage additional data sources, such as social media trends, user-generated content, and demographic information, to enhance recommendation quality. By integrating diverse datasets, systems can offer more nuanced and relevant suggestions.
5. **Ethical and Explainable AI:** As recommendation systems become more sophisticated, addressing ethical considerations and ensuring transparency will be crucial. Developing explainable AI models that provide insights into how recommendations are generated will improve user trust and satisfaction.
6. **Real-Time Adaptation:** Implementing real-time adaptation techniques will allow systems to respond dynamically to changing user preferences and behaviors. This will ensure that recommendations remain relevant as user interests evolve.

## REFERENCES

- [1] Jayalakshmi, Sambandam, Narayanan Ganesh, Robert Čep, and Janakiraman Senthil Murugan. 2022. "Movie Recommender Systems: Concepts, Methods, Challenges, and Future Directions" *Sensors* 22, no. 13: 4904. <https://doi.org/10.3390/s22134904>
- [2] Alyari, F.; Navimipour, N.J. Recommender systems: A systematic review of the state of the art literature and suggestions for future research. *Kybernetes* 2018, 47, 985.
- [3] Caro-Martinez, M.; Jimenez-Diaz, G.; Recio-Garcia, J.A. A theoretical model of explanations in recommender systems. In *Proceedings of the ICCBR, Stockholm, Sweden, 9–12 July 2018*.
- [4] Gupta, S. A Literature Review on Recommendation Systems. *Int. Res. J. Eng. Technol.* 2020, 7, 3600–3605.
- [5] Abdulla, G.M.; Borar, S. Size recommendation system for fashion e-commerce. In *Proceedings of the KDD Workshop on Machine Learning Meets Fashion, Halifax, NS, Canada, 14 August 2017*.
- [6] Aggarwal, C.C. An Introduction to Recommender Systems. In *Recommender Systems*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 1–28.
- [7] Ghazanfar, M.A.; Prugel-Bennett, A. A scalable, accurate hybrid recommender system. In *Proceedings of the 2010 Third International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 9–10 January 2010*.
- [8] Deldjoo, Y.; Elahi, M.; Cremonesi, P.; Garzotto, F.; Piazzolla, P.; Quadrana, M. Content-Based Video Recommendation System Based on Stylistic Visual Features. *J. Data Semant.* 2016, 5, 99–113.
- [9] Alamdari, P.M.; Navimipour, N.J.; Hosseinzadeh, M.; Safaei, A.A.; Darwesh, A. A Systematic Study on the Recommender Systems in the E-Commerce. *IEEE Access* 2020, 8, 115694–115716.



- [10] Cami, B.R.; Hassanpour, H.; Mashayekhi, H. A content-based movie recommender system based on temporal user preferences. In Proceedings of the 2017 3rd Iranian Conference on Intelligent Systems and Signal Processing (ICSPIS), Shahrood, Iran, 20–21 December 2017.
- [11] Beniwal, R.; Debnath, K.; Jha, D.; Singh, M. Hybrid Recommender System Using Artificial Bee Colony Based on Graph Database. In Data Analytics and Management; Springer: Berlin/Heidelberg, Germany, 2021; pp. 687–699.
- [12] Çano, E.; Morisio, M. Hybrid recommender systems: A systematic literature review. *Intell. Data Anal.* 2017, 21, 1487–1524.
- [13] Shen, J.; Zhou, T.; Chen, L. Collaborative filtering-based recommendation system for big data. *Int. J. Comput. Sci. Eng.* 2020, 21, 219–225.
- [14] Dakhel, G.M.; Mahdavi, M. A new collaborative filtering algorithm using K-means clustering and neighbors' voting. In Proceedings of the 11th International Conference on Hybrid Intelligent Systems (HIS), Malacca, Malaysia, 5–8 December 2011; pp. 179–184.
- [15] Katarya, R.; Verma, O.P. An effective collaborative movie recommender system with cuckoo search. *Egypt. Inform. J.* 2017, 18, 105–112.