**C O V E N T R Y**
U N I V E R S I T Y

Faculty of Engineering, Environment and Computing

School of Computing, Electronics and Mathematics

Master of Science in Data Science and Computational Intelligence

7151 CEM- Computing Individual Research Project

# Comparison of Several Movie Recommendation Systems

Author: Guddu Kumar Shah

SID: 11402949

Supervisor: Prof. Xiang Fei

Submitted in partial fulfilment of the requirements for the Degree of Master of Science in Data Science and Computational Intelligence

Academic Year: 2022/23

# Declaration of Originality

I declare that this project is all my own work and has not been copied in part or in whole from any other source except where duly acknowledged. As such, all use of previously published work (from books, journals, magazines, internet etc.) has been acknowledged by citation within the main report to an item in the References or Bibliography lists. I also agree that an electronic copy of this project may be stored and used for the purposes of plagiarism prevention and detection.

# Statement of copyright

I acknowledge that the copyright of this project report, and any product developed as part of the project, belong to Coventry University. Support, including funding, is available to commercialise products and services developed by staff and students. Any revenue that is generated is split with the inventor/s of the product or service. For further information please see www.coventry.ac.uk/ipr or contact ipr@coventry.ac.uk.

# Statement of ethical engagement

I declare that a proposal for this project has been submitted to the Coventry University ethics monitoring website (https://ethics.coventry.ac.uk/) and that the application number is listed below (Note: Projects without an ethical application number will be rejected for marking)

Signed: Guddu Kumar Shah Date: 09-12-2022

Please complete all fields.

| | |
|---|---|
| First Name: | Guddu Kumar |
| Last Name: | Shah |
| Student ID number | 11402949 |
| Ethics Application Number | P143546 |
| 1st Supervisor Name | Xiang Fei |
| 2nd Supervisor Name | Saurav Sthapit |

**This form must be completed, scanned and included with your project submission to Turnitin. Failure to append these declarations may result in your project being rejected for marking.**

# Abstract

Recommendation Systems play an important role in lives of people for taking proper automated decisions. It helps the user in finding out relevant information from a variety of available data in order to make informed decisions. Recommendations for the movies in the movie recommendation system are based on similarity among users which is known as Collaborative Filtering) or based on the activities of particular users (Content Based Filtering). A better recommendation system can be developed by combining collaborative and content based filtering in order to overcome the limitations of collaborative and content based filtering generally. There are numerous similarity measures available for finding out the similarity between users for recommendations. Since Recommendation systems are assisting users in classifying users with similar interests, these systems are very paramount for e-commerce applications and several websites. In this study, several recommendation systems add analysed and compared namely, simple recommendation system, content based recommendation system, collaborative filtering recommendation system, hybrid approach. It is common for social networking sites like Facebook to recommend friends, LinkedIn to recommend jobs, YouTube to recommend videos, Netflix to recommend movies, Amazon to recommend products, etcetera to use recommendation systems to increase their profits and benefit their customers. To explore the research into recommendation systems, this study primarily reviews the different methods and techniques used for movie recommendation.


Keywords- Content based filtering, collaborative filtering, singular value decomposition, hybrid approach, movie recommendation system.

# Table of Contents

# Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. Xiang Fei who has guided me and helped me throughout this research work. This work would not have been possible without his continuous guidance.

I would also like to thank my friends and family who have believed me and motivated me to complete this work and helped me in achieving my dreams with their best wishes.

Lastly, I would also like to pay my gratitude to Coventry University which gave me opportunity to complete my studies at international level-institute.

# 1 Introduction

Recommendation systems can be considered as an informative tool which facilitates users in finding out the items which they want from an extensive list of products. A recommendation system functions by forecasting the rating that a particular user gives to that item (Konstan & Riedl, 2012, p. 101). In this way, it allows the user to choose from a list of available options the best solution for his or her needs. Recommendation systems are used by many companies to provide their users with the right service and increase their profits, such as Netflix, YouTube, Amazon, and others. Yet, the research on these systems remains relevant because finding what the user wants from the available resources is a daunting task as the users keep changing their choices and preferences with time. In today's world, everything a user buys online is based on recommendations. Nowadays when a user buys things online, there is a recommendation system that works in the background, suggesting items based on the user's previous purchases. It can be the times when a user is buying a book, listening to music, watching a movie, etc. These recommendation engines enable users to look for items according to their preferences. Thus, it is very important to build and analyse different recommender systems.

## *1.1 Background to the Project*

The recommendation system has become an integral part of many e-commerce applications. It gathers information about the user's preferences by two methods, such as implicitly gathering information, or explicitly by asking the user. These systems collect user preferences for a variety of items such as shopping, tourism, taxi, TV, movies. In implicit data collection, users' behaviour is typically observed, such as how they watch movies, purchased products, or download applications. In contrary to this, directly procuring information, however, requires collecting previous ratings or history pertaining to the user. With the proliferation of information, users find it almost impossible to find relevant information that they want. Additionally content providers have a hard time making their content stand out from the crowd in the age of information overload. Thereby, recommender systems are developed by researchers and companies in order to solve these contradictions. In order to serve both content providers and users, recommender system connects users and information. In one sense, it helps users find valuable information and in another it pushes the information to specific users. Thus, this can be considered as the win-win situation for both **content providers and customers**.

There are several recommender systems currently which are based on the user information. The intention of this study is to find answers to the following research questions.

- What is the difference between different Recommendation Systems available?
- How are these Recommendation systems able to generate recommendations?
- How can the similarity among movies be predicted for recommendations?
- Does increasing weightage of any particular feature such as cast, crew, or genre have any impact on generated recommendations?

## *1.2 Project Objectives*

The aim of this study is to analyse, develop, and compare the behaviour of several movie recommender systems on a similar dataset. In order to achieve the desired goal of this work, there are several objectives which need to be accomplished.

The objectives of this work are given below.

- To investigate and select dataset suitable for generating recommendations by the model.

- To review the working of different recommender systems such as General Recommender System, Content-Based, and Collaborative Filtering based.
- To implement the reviewed recommender systems using Python programming language
- To design the Hybrid approach based recommendation system
- To generate the movie recommendations when any of the information subjected to movie or user is passed as an input to the system.
- To identify how the similarity among the movies be calculated to generate recommendations.
- To experiment the impact of different weightage given to features on movie recommendations.
- To assess and compare the performance of systems developed qualitatively.

### 1.2.1 Gather client and user requirements

The requirement of this study is to generate the movie recommendations to the user when given the name of movie or user information. Several recommendations should be presented to the user such as:

- General Popular movies which can be recommended to anyone irrespective of their personalised choices.
- Top movies according to the genre such as Popular Comedy movies, Action movies, Drama movies, etc.
- Movie recommendations having similar overview or tagline with movies that have been previously watched.
- Movie recommendations having similar cast, director, or keywords with movies that have been previously watched.
- Movie recommendations which are personalised for users.

### 1.2.2 Analyse and model the requirements

The above requirements have been analysed. It can be said that different Recommendation systems are required to put forward the recommendations to the users. Thus, several recommendation systems need to be developed which can generate these outcomes.

- General popular movies can be recommended by considering the weighted rating formula which helps in calculating the general popularity of the movie among all the given movies.
- In order to generate popular movie recommendations according to the genres, the data should be extracted initially for the given genre. The weighted rating formula can be applied on the extracted data to generate the popular movies according to the genres.
- Content-Based recommender systems can be used to generate recommendations in context of movie description such as overview or tagline; or movie meta data such as cast, crew, etc.
- Collaborative Filtering can be used to generate recommendations which are tailored to the users.

## *1.3 Applications of Recommender Systems*

The area of recommendation system is very extensive, and it is used throughout all kinds of filled since it saves time. Many real-life applications have incorporated recommendation systems in

them such as services, social media, e-commerce, entertainment etc (Kumar & Reddy, 2014, p. 5254). Recommendation systems are widely used for listening to music, watching movies, or any TV programme in entertainment area. The world's largest shopping site in the e-commerce field is Amazon. Some people use it for buying books, for buying home goods or clothing, and some people use it for everything. In this way the whole world relies on these e-commerce sites for one thing or another. There are several more applications of Recommender Systems which are given below:

Travel service: Here, recommendations offer you various travel sites to guarantee a smooth commute. You will find sites such as Road trippers that make planning any road trip easy. Hooper, on the other hand, helps you figure out when is the best time to book your flight. They all have recommendation systems in their background (Heung et al., 2001, p. 259).
Fashion: Recommendation Systems are also helpful in suggesting clothing items to the users. There are various shopping sites and applications that make use of such systems to generate recommendations for the users such as Myntra, SHEIN, Amazon, etc.

Music Recommendation: There are numerous mediums that are based on recommendation systems such as Pandora, Spotify, JioSavan, Ganna, etc which provide music options to users.

News: Various applications that suggest articles, news, blog post, content from top publishers are based on recommendation systems such as Google News, apple news, pocket, tweet deck, News360, etc.

Movie Recommendation: Several entertainment platforms such as Netflix, hotstar, voot, amazon prime, Sony LIV, Alt Balaji, etc make use of recommendation systems to generate movie choices for users.

## *1.4 Challenges in Recommender Systems*

There are certain issues that are faced by majority of the recommendation systems. Most commonly faced challenges are Cold Start Problem, Data Sparsity, and Scalability. The brief overview of these issues is given below.

Cold Start Problem: To find a match, there must be enough users within the recommendation system. For instance, to find a matching item or user, we must match against the set of users we have available. New users have an empty profile at the beginning of their accounts since they have not treated any items and the system doesn't know their taste. This makes it difficult for the system to make recommendations in such scenarios. Similarly, with new items, which have not been rated by any other user because they are new. Both problems can be solved by utilising hybrid approaches (Bobadilla et al., 2013, p. 109).

Data Sparsity: Since the majority of users do not rate items com it is very hard to find users who have rated the same items. As a result, it is challenging to find a group of users who read the items. Recommending an item can be a difficult task when there is little information about the individual (Sharma & Mann, 2013, p. 8).

Scalability: the collaborative filtering process uses massive amounts of data to make reliable decisions, requiring a larger number of resources. As information grows exponentially, the coast and accuracy of the processing grow as well (Mansur et al., 2017, p. 3).

## *1.5 Overview of This Report*

This section presents the report outline. The work is divided into Eleven chapters. Chapter 1 presents the introduction of the work giving overview about the recommendation systems, applications, challenges faced by them, aims and objectives of study, and report structure. Chapter 2 presents the literature review in which several recommendation systems are overviewed and discussed along with some prior works done by co-authors associated with them. Chapter 3 presents the Research methodology adopted for this study. Chapter 4 presents the design of the systems developed in this study. Chapter 5 presents the Requirements section where the necessary requirements to implement the proposed design solution is discussed. Chapter 6 presents the Implementation where the steps taken to build several recommendation systems are discussed. The results obtained with these systems are discussed in Chapter 7. Chapter 8 presents the Project Management section where the project schedule is illustrated using Gantt Charts along with some risk management, quality management analysis. Conclusion of the study is discussed in Chapter 9 of this work where the achievement of the work is discussed along with some possible future work in the study. Critical Appraisal of the work has been discussed in Chapter 10. Chapter 11 presents the own learnings and experience gained while performing this work. The last section of the work discusses the References and Bibliography where all the resources used are specified.

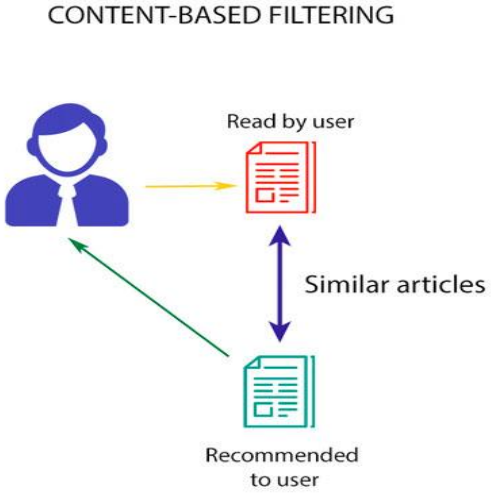
# 2 Literature Review

This section presents the analysis of several recommendation systems available along with some overview of prior works done by the authors in the similar context. There are three broad categories of Recommender systems- Content-Based Filtering, Collaborative Filtering, and Hybrid Filtering. Recommendation system powered by Content based recommendation gathers data from the user explicitly by rating or implicitly by clicking a link and generates a user profile based on that information. Recommendations are generated by making use of user profile. Items are recommended by the system to the users based on their past activity. In contrast to Content based filters, Collaborative filtering finds users who are similar to the given user, and then recommends items or products based on the fact that these users will also like the item because they have similar tests. As both of techniques have their own strengths and weaknesses, a hybrid approach has been developed to overcome their weaknesses. These approaches can be used in different ways. Content based filtering can be applied first, followed by collaboration recommended, or integrating both filters in one model to generate results. This type of modification can also alleviate problems associated with the recommender systems such as cold start, data sparsity, and scalability. Several recommendation systems are discussed below.

## *2.1 Content-Based Filtering*

It is type of filtering used in the recommendation system to increase the accuracy. It works on the input of user past activity and give recommendation to user based on that, like if anyone searches on the internet about drama movies a lot in the past then, the system can find that user like drama movies and recommends him only these kinds of movies only and similar highly rated movies. It is also known as the cognitive filtering system (Li et al., 2012, p. 277). If we take another example, let just say if someone like current affairs content then this system will recommend similar content news channel, blogs etc. Content-based filtering only focuses on a particular user interest instead of focusing on other user interaction along with main user as collaborative filtering method does. This approach only focuses on each individual user's ideas, thoughts and gives prediction based on his interest. Content based filtering first checks the user's preferences before recommending movies or products to him/her.

**Figure 1**
**Content-based Filtering**



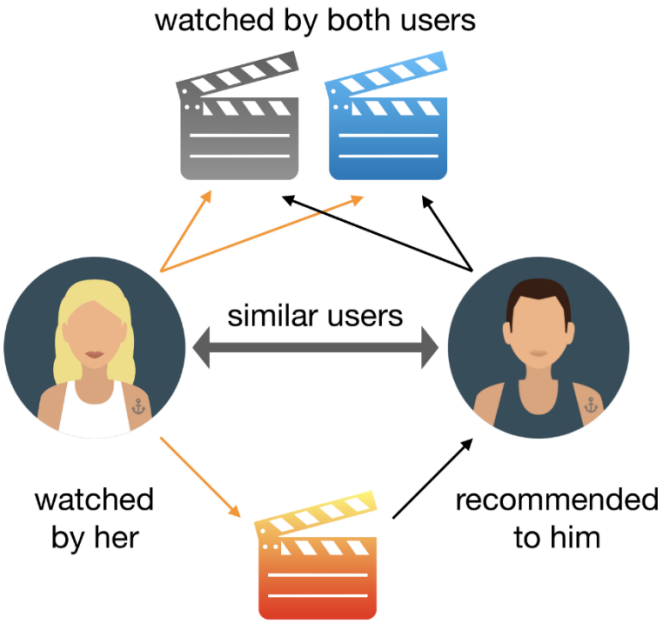

**Adapted from Yash (2019)**

The content-based filtering technique ensures a good match between the movie and the rating that is given by the user. This is done by checking the genre categories in the user profile to determine which movies are given high ratings by the user. The figure above illustrates the working of content-based recommender system. In the content-based approach, no other users' data is required to make recommendations. New items can be suggested based on the user's unique test. No first raters are required. In addition, it offers content features that enables us to explain why certain items are recommended. Finding specific features, such as images or movies of a particular genre, can sometimes be daunting. This problem is referred as over specialisation. Users are never recommended anything outside their user profiles. It is easy to miss recommending an item to users since the item does not have enough information about it.

## *2.2 Collaborative Filtering*

The recommendation system is used by many e-commerce sites presently, such as Amazon, CDNow, Drugstor, MovieFinder, etc. There is a huge amount of data available from these sources. As we live in this hectic world, no one has time to search through thousands of hundreds of items and select the ones that appeal to their taste. Collaboration filtering is one of the ways to filter the data and provide the relevant information to the users. The collaborative filtering technique is a well known technique for recommending items. It suggests relevant items to the user based on the choices of their neighbours. In this method, the similarity between the user and his neighbour is determined first, and then a prediction of the items is made. There can be n users, and this technique finds the similar user from a list of users. Users are compared based on their ratings for the items in question. In this way, the comparison continues, and the desired result is achieved. In this strategy, ratings are taken from the large catalogue of ratings given by the users for a given item. This collection, called the user-item matrix, records the ratings for every item in that large catalogue (Khatwani & Chandak, 2016, p. 625).

**Figure 2**
**Collaborative-Filtering Approach**




**Adapted From vitalflux (2022)**

The figure above shows the Collaborative-Filtering approach. Both the users 1 and 2 saw some same movies, thus, identified as similar users. Thus, movie watched by user 1 is recommended to user 2 since there are high chances of another user liking the movie.

Collaborative Filtering methods can further be split into two approaches- Memory Based Methods and Model-Based Methods. In memory-based methods, similarity measures are calculated based on explicit user ratings in order to identify neighbours and make predictions. This method examines the interest of the user in a particular item. After analysing the users' view for an item, it searches for similar users who also share the same interest using the utility matrix. Therefore, this method mainly uses system memory to predict similar users. The unknown rating of any user can be generated based on the user item rating matrix (utility matrix) if similar users are found. Finally, recommendations can be provided based on these results.

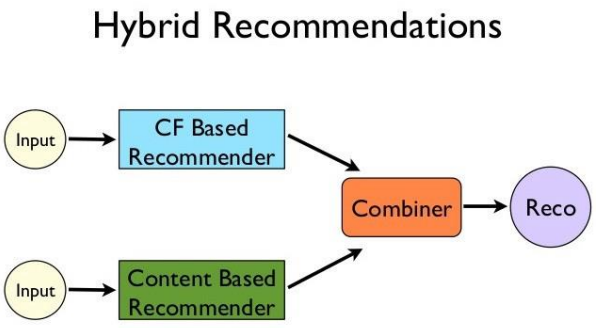
Model based approach methods used the ratings of each user to develop a user model that estimates the expected value of unrated items based on these ratings (Liu et al., 2018, p. 275). The model is created a utility metrics based on user ratings for any given item and can be calculated using a machine learning or data mining algorithm. As a result of the utility metrics, a model is developed to generate predictions for the users with the help of the given data. The model-based approaches can be further classified into a variety of categories such as decision tree, artificial neural network, clustering, association rule mining, regression etc. A number of model-based approaches exist, including Latent Semantic models like Latent Semantic Analysis, Latent Semantic Indexing, and Dimensionality Reduction methods like Matrix Factorization, Singular Value Decomposition (SVD) and others. These techniques tend to solve sparsity problem as well which occurs in general recommendation systems. In this work, Collaborative Filtering based recommendation system has been developed using Singular Value Decomposition (SVD) technique.

Due to the fact that collaborative filtering works with any kind of item, no feature selection is required. the biggest advantage of collaborative filtering. A typical problem is that no unrated item can be recommended by it, as it suffers from first traitor problem. Furthermore, there is a popularity bias issue which arises when something may not be liked by the user, but it is recommended by the system because of its popularity.

## *2.3 Hybrid Recommender System*

This method of filtering information is based on combining collaborative filtering with content based filtering and generating user recommendations based on the features of the movie. This approach is a mixture of two approaches that is content based filtering and collaborative filtering. This filtering system tends to solve the challenges faced by the collaborative filtering recommendation systems and content based filtering recommendation systems such as cold start.

**Figure 3**
**Hybrid Recommender System**




**Adapted From miro.medium**

The figure above shows the hybrid recommendation system. It can be observed from the figure that initially the input is passed to content-based and collaborative filter approaches. The combiner then combines both the approaches and generates the necessary recommendations.

These systems tend to solve various challenges faced by the prior systems such as sparsity problem, cold start problem, and grey sheep problem. These systems usually have high implementation cost due to the combination of two approaches and sometimes, also lead to increased complexity of the system.

## *2.4 Cosine Similarity Measure*

There are numerous measures that can be used to identify the similarity between different users and items such as Cosine Similarity, Jaccard Similarity, Pearson Correlation Similarity, Mean Square Distance, etc. In this study, recommendations systems are using Cosine Similarity for identifying the similarity between movies and users. Cosine Similarity is used to measure the degree of similarity between two non-zero vectors in the inner product space. The angle between two vectors is measured by cosine similarity. The values of similarity between two non-zero vectors, u and v can be obtained by calculating the dot product of these two vectors and can be represented as:

$$u \cdot v = \|u\| \cdot \|v\| \cdot \cos\theta$$

Where u and v are two non-zero vectors and θ is the angle between them.

## *2.5 Related Work*

This section provides the overview of some of the work done by previous other related to the recommender systems or the approaches discussed above.

A Content-Based Filtering Recommender System has been proposed by the (Son & Kim, 2017, p. 404). using multi-attribute networks. Network analysis is used to identify various items based on all attributes, thereby solving over specialisation issues. Comparing Content based filtering, Linked open data, and Feature Weighting, results obtained by the authors also revealed improvements in sparsity and scalability. The authors used Movie lens dataset for conducting experiments.

Both the approaches of Collaborative Filtering- User Based and Item-based are used by (Wu et al., 2018, p. 13). in their study. The similarity between the users have been obtained using Pearson Correlation Similarity. The authors also used Nearest N User Neighbourhood algorithm for obtaining N similar users to form a group for them. The similarity between the items have been found out using another algorithm known as Log-Likelihood Similarity. Hadoop Distributed File System (HDFS) has been used to store the item-based recommender. The authors used Yahoo Research Web Scope Database dataset.

(Nilashi et al., 2018, p. 677). used Collaborative filtering recommendation system for building a movie recommendation system. The authors conquered the problem of Sparsity and Scalability using the Ontology and Dimensionality Reduction technique. Hadoop programming language was used by (Tianqi et al., 2017, p. 47). for implementation of Movie Recommendation system. The recommender was based on Collaborative filtering approaches based on items. The authors used Movie Lens dataset comprising of 1000 records for rating.

(Yang et al., 2018, p. 277). proposed hybrid recommendation approach by combining social similarities and genres of the movie with collaborative filtering method. The proposed methodology solved the sparsity problem and worked in two stages. The first stage is used to obtain the candidate set using Matrix Factorization using the ratings by training the training dataset. The unknown ratings are predicted after finding the candidate set. The obtained ratings are then sorted to obtain the final candidate set for each user comprising of several top items. The movies are recommended during the second stage using feature selection TF-IDF method for calculating the similarity between users. The authors used Movie Lens dataset. Thy observed that hybrid approach produced better recommendations than Collaborative filtering based methods.

(Bharti & Gupta, 2019, p. 110). used Movie Lens dataset for recommending movies to the users. The authors used two different approaches for generating recommendations depending on the users. Content-Based Filtering approach was used for new users whilst the collaborative filtering approach was used for old users. The authors have used Cosine and Pearson similarity to find the similarities for generating recommendations. The information about user and movie is stored in the database using Hive.

## *2.6 Summary of Literature Review*

After reviewing the above work and literature, it has been observed that there are three major categories of Recommender Systems namely, Content-Based Recommender System, Collaborative Filtering based, and Hybrid approach based recommender system. There are several problems identified with Content-Based and Collaborative filtering based systems such as Cold Start, Data Sparsity, etc. These challenges can be overcome by using hybrid approaches where both the above discussed approaches combine together to form a hybrid recommender.

It has also been observed that there are numerous similarity calculating measures that can be used to identify the similarity scores among different users and items. Furthermore, it has also been analysed that there are not much evaluating metrics available for recommendations except for Collaborative filtering approaches. The reason can be due to the fact that there are never any 100% accurate recommendations. There can be certain situations when the recommendations are correct and useful for one user whilst not useful for others. Thus, these need to be assessed qualitatively only. Majority of the studies are based on the different versions of Movie Lens dataset having different number of attributes, movie ratings, users, etc.

One of the goal of the study is to analyse different recommendation systems which has been achieved in this section. The next step is to implement these systems to analyse their behaviour. Thus, different recommendation systems have been developed in this work using the dataset MovieLens which has been used in majority of the studies. The similarity criteria adopted is Cosine Similarity for calculating the similarity among the movies and users for generating recommendations. It can be analysed from above works that none of the studies have discussed about popularity calculation which can be done using IMDB weighted rating formula. Thus, these studies generate the recommendations considering popularity of the movies as well.

# 3 Methodology

The research work has been done using the Saunders Research Onion methodology. A research methodological first to the process by which the study researchers accomplished, starting with the selection of the topic, studying the literature, and testing the developed software itself. In this research work, several methodologies have been adopted to implement the movie recommender system such as content based filtration methods, collaborative filtering based methods, hybrid filtering recommendation methods.

As illustrated by the Saunders Research Onion approach, Compared to peeling an onion layer by layer and reaching the court through the outer cover. As part of the research methodology coma several essential steps are involved, such as the research philosophy, the research approach, the research strategy, the research choices, and the time horizon. The following brief description of each research step can be found below.

Research Philosophy can be thought of as the top most layer of an onion it can be divided into two major types- Ontology an Epistemology. Ontology specifies the nature of reality which is further defined by three philosopher stances-Objectivism, Constructivism, and Pragmatism. The concept of epistemology, on the other hand, explains what constitutes acceptable knowledge and how that knowledge may be acquired. There are three stances that can be applied to Epistemology- Positivism, Critical realism, and Interpretivism. **Positivism** research philosophy has been used in this research.

After peeling away the outer most layer, the next layer represents the Research Approach. Inductive and Deductive are two of the most common approaches to research. Research approaches that use the inductive approach start from specifics to generals. These types of approaches apply where adequate research is available, therefore theory development is required. Alternatively, deductive research operates from general to specific. Researchers start by reviewing existing theories, generating hypothesis, and thereafter testing these hypotheses. It is common practise to use a deductive approach for the purpose of conducting research com especially in the field of epistemology, particularly with positivism. This study has used a **deductive approach** for researching the central question.

A research strategy is the next inner layer. This layer signifies what data has been gathered. There are many types of research strategies, such as action research, case studies, surveys, experimentation, grounded theory, and ethnography, archival research, etc. It is common for studies to use combinations of different research strategies. In this study, **experimental research** is used for the implementation an analysis of the different recommender systems.

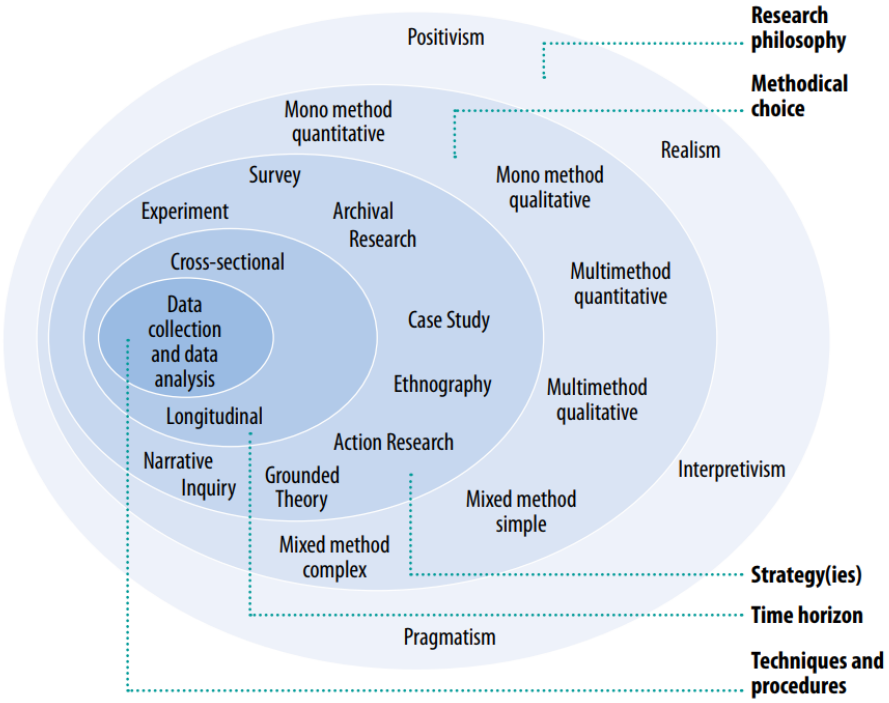
The next layer present is research choices which are determined by factors such as data collection, analysis methods, and research style. Research can be quantitative, qualitative, or a mixed methods approach. There are three types of research methods-mono, mixed, and multi methods. The research can be considered as mono method if the research is either quantitative or qualitative. Research that uses both qualitative and quantitative data and also uses quantitative and qualitative analysis can be considered as mixed research. On the other hand, research that uses both qualitative and quantitative data but analysis is done either qualitative or quantitative, can be considered as multi method. The research choice used is **Mixed-methods**.

Time horizons is the next layer present after research choices. Cross-Sectional and Longitudinal are the two main timeframes. Cross-sectional has short time frame. A snapshot view is obtained at a particular time in this horizon for the situation under investigation. In contrary to this, that time

frame is longitudinal when a sample is studied over a long period for understanding the behaviours and events. In this study the time frame of study is short, thereby **cross-sectional** time horizon is used.

A final stage of the research methodology involved deciding that techniques and procedures. Here it is decided what techniques and procedures to use, what data to use, how to analyse the data, and how to deal with ethical issues.

**Figure 4**
**Saunders Research Onion**




**Adapted From imgur.com**

The figure above shows the overview of layers or steps present in the research methodology adopted in this study which is Saunders Research Onion.

# 4 Requirements

The intention of this work is to analyse the recommendations generated by several recommender systems. Since the recommender systems need data to generate the recommendations, thus, the major requirement of this study is a large-scale dataset. The dataset should comprise of several information about users and movies which can be some descriptive information of movies such as Overview, Title, reviews of people, etc, or it should possess some user information such as his interests, ratings given by him, etc.

Apart from dataset, another major requirement of this work is the tool where these recommender systems can be implemented along with the programming language to be used. There is no denying fact that python programming language is very useful while working with development. Since it offers various packages to handle data such as numpy, pandas, python is a suitable programming language to be used in this work. Collaborative filtering requires analysis of large natural language text, thus, Python can be proved helpful for that also by offering various pre-processing libraries helpful for cleaning data and surprise module. Furthermore, python is easy to use language and has simple syntax structure.

In order to work with large datasets, large number of computational resources such as RAM, GPU, Disk storage, etc are required to execute the experiments. To conquer these challenges, the recommender systems have been developed on Google Colab which is a cloud-based tool offering large RAM and disk storage space. Additionally, it provides GPU access also which helps in executing the experiments quickly.

Thus, the requirements of this work can be summarised as three major requirements namely, Dataset, Software Tool, and Programming Language and its packages.

## *4.1 Dataset Description*

As discussed above, the dataset is one of the primarily requirements of this study. Thus, this section provides the overview how this requirement has been fulfilled by selecting appropriate dataset. There are numerous datasets available in public repositories such as Kaggle, UCI, etc. Several datasets have been examined to select the appropriate dataset for study. The dataset selected for this study is an ensemble of data collected from TMDB and GroupLens. The overview of dataset selected is given below:

The dataset comprises of six csv files which can be proved useful for implementation of different recommendation systems.

**Movies_metadata.csv file**- It is the primary and important file present in the dataset. It comprises of information of nearly 45000 movies which have been featured in the full MovieLens dataset. Features present in this dataset are 'adult', 'belongs_to_collection', 'budget', 'genres', 'homepage', 'id', 'imdb_id', 'original_language', 'original_title', 'overview', 'popularity', 'poster_path', 'production_companies', 'production_countries', 'release_date', 'revenue', 'runtime', 'spoken_languages', 'status', 'tagline', 'title', 'video', 'vote_average', 'vote_count'. There are 24 features present in this dataset. The data types, null count present in the dataset are given below.

**Table 1**
**Movies Metadata Overview**

| Column Number in dataframe | Feature Name | Non-Null Value Count | Data Type |
|---|---|---|---|
| 0 | adult | 45466 non-null | object |
| 1 | belongs_to_collection | 4494 non-null | object |
| 2 | budget | 45466 non-null | object |
| 3 | genres | 45466 non-null | object |
| 4 | homepage | 7782 non-null | object |
| 5 | id | 45466 non-null | object |
| 6 | imdb_id | 45449 non-null | object |
| 7 | original_language | 45455 non-null | object |
| 8 | original_title | 45466 non-null | object |
| 9 | overview | 44512 non-null | object |
| 10 | popularity | 45461 non-null | object |
| 11 | poster_path | 45080 non-null | object |
| 12 | production_companies | 45463 non-null | object |
| 13 | production_countries | 45463 non-null | object |
| 14 | release_date | 45379 non-null | object |
| 15 | revenue | 45460 non-null | float64 |
| 16 | runtime | 45203 non-null | float64 |
| 17 | spoken_languages | 45460 non-null | object |
| 18 | status | 45379 non-null | object |
| 19 | tagline | 20412 non-null | object |
| 20 | title | 45460 non-null | object |
| 21 | video | 45460 non-null | object |
| 22 | vote_average | 45460 non-null | float64 |
| 23 | vote_count | 45460 non-null | float64 |

The table 1 shows the features present in the movies metadata file. The features present in the dataset are self- explanatory. The values present for these features are shown in the figure below.

**Figure 5**
**Transpose of Top 3 rows of Movies Metadata file**

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| adult | False | False | False | False |
| belongs_to_collection | {'id': 10194, 'name': 'Toy Story Collection', ... | NaN | {'id': 119050, 'name': 'Grumpy Old Men Collect... | NaN |
| budget | 30000000 | 65000000 | 0 | 16000000 |
| genres | [{'id': 16, 'name': 'Animation'}, {'id': 35, '... | [{'id': 12, 'name': 'Adventure'}, {'id': 14, '... | [{'id': 10749, 'name': 'Romance'}, {'id': 35, ... | [{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam... |
| homepage | http://toystory.disney.com/toy-story | NaN | NaN | NaN |
| id | 862 | 8844 | 15602 | 31357 |
| imdb_id | tt0114709 | tt0113497 | tt0113228 | tt0114885 |
| original_language | en | en | en | en |
| original_title | Toy Story | Jumanji | Grumpier Old Men | Waiting to Exhale |
| overview | Led by Woody, Andy's toys live happily in his ... | When siblings Judy and Peter discover an encha... | A family wedding reignites the ancient feud be... | Cheated on, mistreated and stepped on, the wom... |
| popularity | 21.946943 | 17.015539 | 11.7129 | 3.859495 |
| poster_path | /rhIRbceoE9lR4veEXuwCC2wARtG.jpg | /vzmL6fP7aPKNKPRTFnZmiUfciyV.jpg | /6ksm1sjKMFLbO7UY2i6G1ju9SML.jpg | /16XOMpEaLWkrcPqSQqhTmeJuqQl.jpg |
| production_companies | [{'name': 'Pixar Animation Studios', 'id': 3}] | [{'name': 'TriStar Pictures', 'id': 559}, {'na... | [{'name': 'Warner Bros.', 'id': 6194}, {'name'... | [{'name': 'Twentieth Century Fox Film Corporat... |
| production_countries | [{'iso_3166_1': 'US', 'name': 'United States o... | [{'iso_3166_1': 'US', 'name': 'United States o... | [{'iso_3166_1': 'US', 'name': 'United States o... | [{'iso_3166_1': 'US', 'name': 'United States o... |
| release_date | 1995-10-30 | 1995-12-15 | 1995-12-22 | 1995-12-22 |
| revenue | 373554033.0 | 262797249.0 | 0.0 | 81452156.0 |
| runtime | 81.0 | 104.0 | 101.0 | 127.0 |
| spoken_languages | [{'iso_639_1': 'en', 'name': 'English'}] | [{'iso_639_1': 'en', 'name': 'English'}, {'iso... | [{'iso_639_1': 'en', 'name': 'English'}] | [{'iso_639_1': 'en', 'name': 'English'}] |
| status | Released | Released | Released | Released |
| tagline | NaN | Roll the dice and unleash the excitement! | Still Yelling. Still Fighting. Still Ready for... | Friends are the people who let you be yourself... |
| title | Toy Story | Jumanji | Grumpier Old Men | Waiting to Exhale |
| video | False | False | False | False |
| vote_average | 7.7 | 6.9 | 6.5 | 6.1 |
| vote_count | 5415.0 | 2413.0 | 92.0 | 34.0 |

**Keywords.csv file**- This file comprises of keywords for the movies present in the metadata file. The keywords present in the file are present in the form of JSON object which needs to be cleaned

before analysis. The top 5 rows of the file is shown below in the figure. It can be observed from the figure that in addition to keywords, the id for all the movies is also present. This is unique identifier for each movie. Thus, this id column can be used to map necessary information about movies present in different files such as movies metadata.

**Figure 6**
**Top 5 rows of the Keywords.csv file**

| | id | keywords |
|---|---|---|
| 0 | 862 | [{'id': 931, 'name': 'jealousy'}, {'id': 4290,... |
| 1 | 8844 | [{'id': 10090, 'name': 'board game'}, {'id': 1... |
| 2 | 15602 | [{'id': 1495, 'name': 'fishing'}, {'id': 12392... |
| 3 | 31357 | [{'id': 818, 'name': 'based on novel'}, {'id':... |
| 4 | 11862 | [{'id': 1009, 'name': 'baby'}, {'id': 1599, 'n... |

**Credits.csv file**- This file comprises of information regarding cast and crew for the movies present in the metadata file. The information present in the file is present in the form of JSON object which needs to be cleaned before analysis. The top 5 rows of the file are shown below in the figure. It can be observed from the figure that in addition to cast and crew, the id for all the movies is also present. This is unique identifier for each movie. Thus, this id column can be used to map necessary information about movies present in different files such as movies metadata.

**Figure 7**
**Top 5 rows of the Credits.csv file**

| | cast | crew | id |
|---|---|---|---|
| 0 | [{'cast_id': 14, 'character': 'Woody (voice)',... | [{'credit_id': '52fe4284c3a36847f8024f49', 'de... | 862 |
| 1 | [{'cast_id': 1, 'character': 'Alan Parrish', '... | [{'credit_id': '52fe44bfc3a36847f80a7cd1', 'de... | 8844 |
| 2 | [{'cast_id': 2, 'character': 'Max Goldman', 'c... | [{'credit_id': '52fe466a9251416c75077a89', 'de... | 15602 |
| 3 | [{'cast_id': 1, 'character': "Savannah 'Vannah... | [{'credit_id': '52fe44779251416c91011acb', 'de... | 31357 |
| 4 | [{'cast_id': 1, 'character': 'George Banks', '... | [{'credit_id': '52fe44959251416c75039ed7', 'de... | 11862 |

**Links.csv file-** This file comprises of movieId, imbID, and tmbID for the movies. All these ids are unique identifier for the movie. The top 5 rows of the file are shown below in the figure. These columns can be used to map necessary information about movies present in different files such as movies metadata, ratings, or credits.

**Figure 8**
**Top 5 rows of links.csv file**

| | movieId | imdbId | tmdbId |
|---|---|---|---|
| 0 | 1 | 114709 | 862.0 |
| 1 | 2 | 113497 | 8844.0 |
| 2 | 3 | 113228 | 15602.0 |
| 3 | 4 | 114885 | 31357.0 |
| 4 | 5 | 113041 | 11862.0 |

**Links_small.csv file-** This file comprises of movieId, imbID, and tmbID for the movies. All these ids are unique identifier for the movie. This file is a subset of links.csv file. The file contains information of 9000 movies whilst the original links file comprises information of nearly 45000 movies. This file can be used in case if there are less computational resources available for collaborative filtering.

**Ratings_small.csv file-** This file comprises of userId, movieId, rating, and timestamp for the movies. The file can be used to access the user details such as his id, movies watched by him using movieId, ratings given by him at the timestamp specified. The figure below shows the top 5 rows of the ratings file.

**Figure 9**
**Top 5 rows of ratings_small.csv file**

| | userId | movieId | rating | timestamp |
|---|---|---|---|---|
| 0 | 1 | 31 | 2.5 | 1260759144 |
| 1 | 1 | 1029 | 3.0 | 1260759179 |
| 2 | 1 | 1061 | 3.0 | 1260759182 |
| 3 | 1 | 1129 | 2.0 | 1260759185 |
| 4 | 1 | 1172 | 4.0 | 1260759205 |

The dataset present in Kaggle has been originally created by obtaining data from two sources- TMDB and GroupLens. The files Movie Details, Credits, and keywords have been extracted from the TMDB open API. The ratings of the movie and link to the movie have been taken from the GroupLens. The movie data present in this dataset consists of information of movies which are released before or in July 2017. The combined dataset is available on public source Kaggle and can be used for research and study purposes.

## *4.2 Required Libraries*

There are certain Python libraries that are required to implement and analyse recommender systems which needs to be imported prior to using them. Some of the essential libraries are discussed below:

Os, zipfile- these libraries are used for file handling operations. Since the dataset in this study has been loaded directly from Kaggle, thus these libraries have been used for performing these operations.

Numpy, Pandas- These libraries are very useful python libraries. These facilitate in performing several mathematical operations on data and helps machine in understanding it better for decision making.

Matplotlib, Seaborn- Some of the visualisation libraries are also essential to import so that analysis can be visualised if required.

NLTK library- There are various functions offered by NLTK packages for handling textual data such as for Stemming and applying other pre-processing tasks.

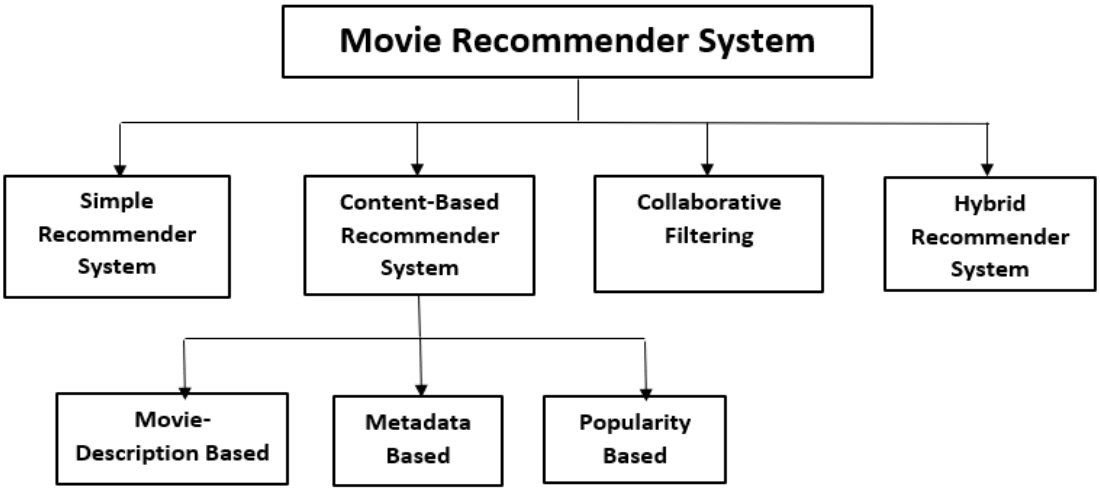
Sklean library- This package offers numerous of functions which are essential for machines to make decisions. In this study, several functions such which are essential for forming TF-IDF and Count Matrix, calculating pairwise similarity such as Linear_Kernel and Cosine_similarity are imported.

Surprise- This module has been imported to implement the Collaborative filtering approach as it is not being built from scratch, instead model-based approach is used.

# 5 Design

This section presents the block diagrams for the Recommendation Systems developed. The figure below shows the hierarchical structure of several Movie Recommender Systems. All these systems have been implemented in this study whose high-level block diagrams are also discussed in this section.

**Figure 10**
**Recommendation Systems Developed**

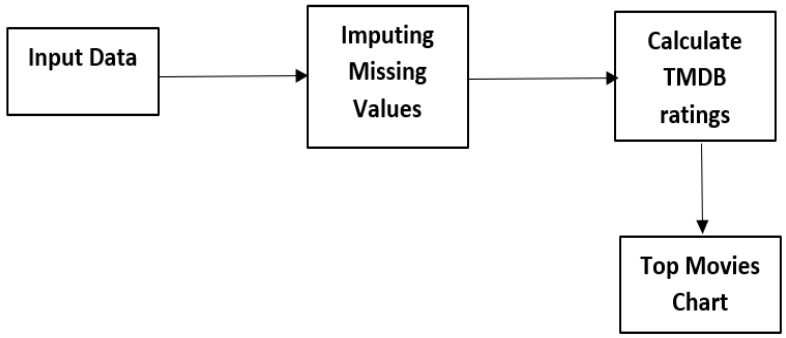



## *5.1 Simple Recommender System*

It can be observed from above figure that there are four broad recommender systems. The Content-Based Recommender systems have been implemented using three different approaches shown in the diagram. Furthermore, another variant of Simple recommender system is also implemented which generates recommendations according to genre of movies. Thus, **there are seven recommender systems implemented and assessed** in this study.

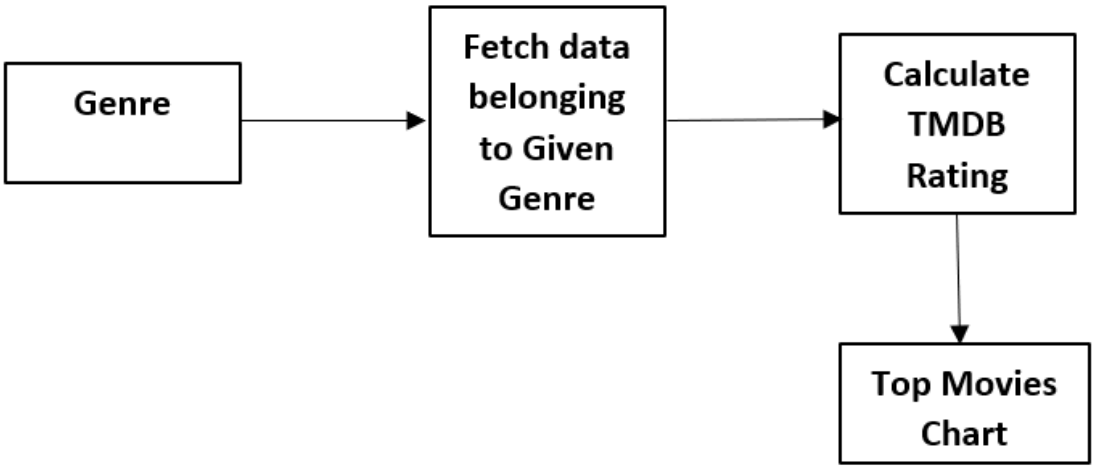
The figure below shows the block diagram for Simple Recommender System. The input to this system is entire movies metadata file. The outcome of this system is the chart comprising of top movies obtained using IMDB weighted rating formula. The detailed implementation of this system has been discussed in the next section.

**Figure 11**
**Block Diagram of Simple Recommender System**



The figure below shows the block diagram for second type of Simple Recommender System. The input to this system is genre or category of movie in which user is interested. The outcome of this system is the chart comprising of top movies of the given genre obtained using IMDB weighted rating formula. The detailed implementation of this system has been discussed in the next section.

**Figure 12**
**Block Diagram for Movie Recommender System Displaying Top Movies for Given Genre**




## *5.2 Content-Based Recommender System*

The figure below shows the block diagram illustrating the prior steps required for generating recommendation from the system using Movie Description and Meta data. The detailed implementation of these steps is discussed in the next section.

**Figure 13**
**Block Diagram for Data Pre-processing Steps for Movie Description Based Recommender System**

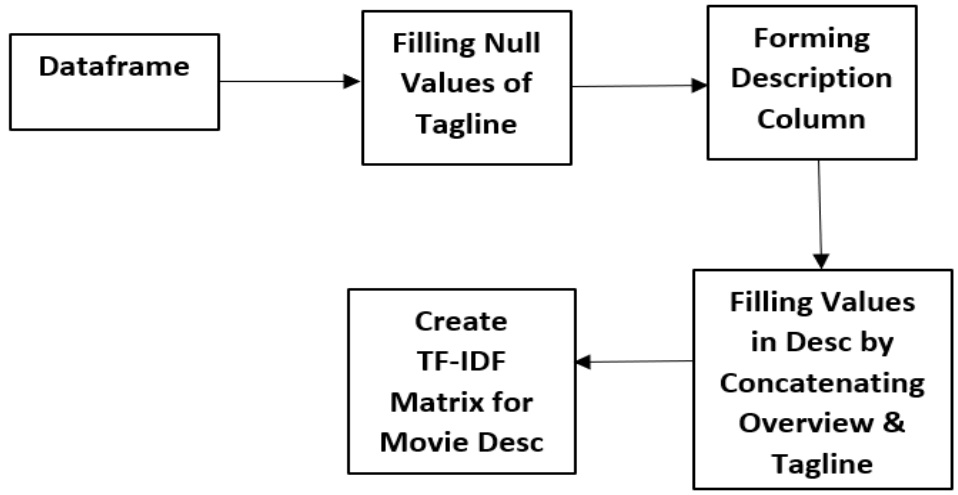
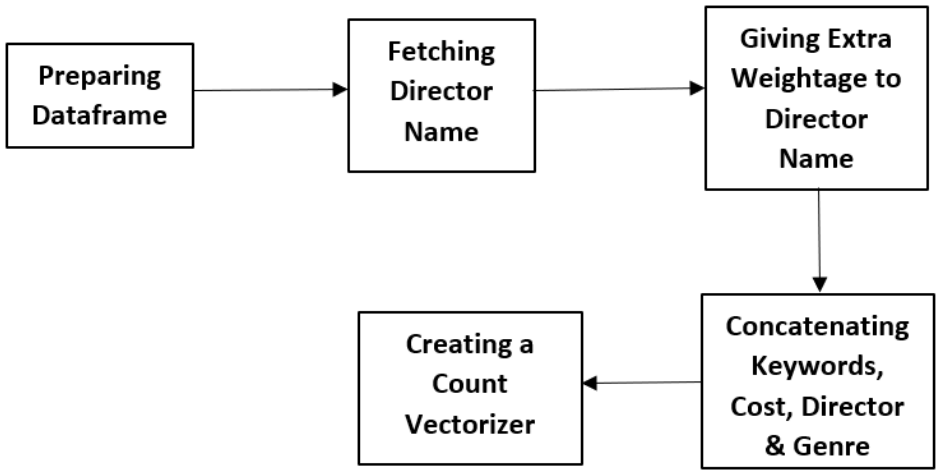
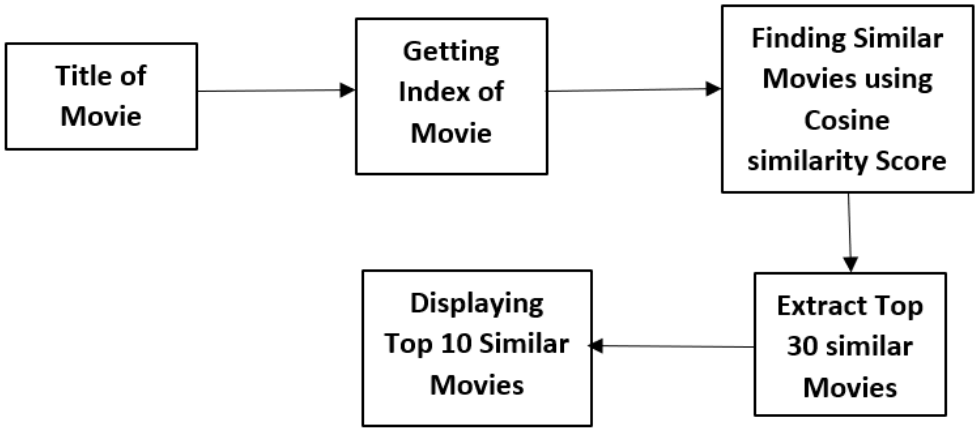



**Figure 14**
**Block Diagram for Data Pre-processing Steps for Meta-Data Based Recommender System**



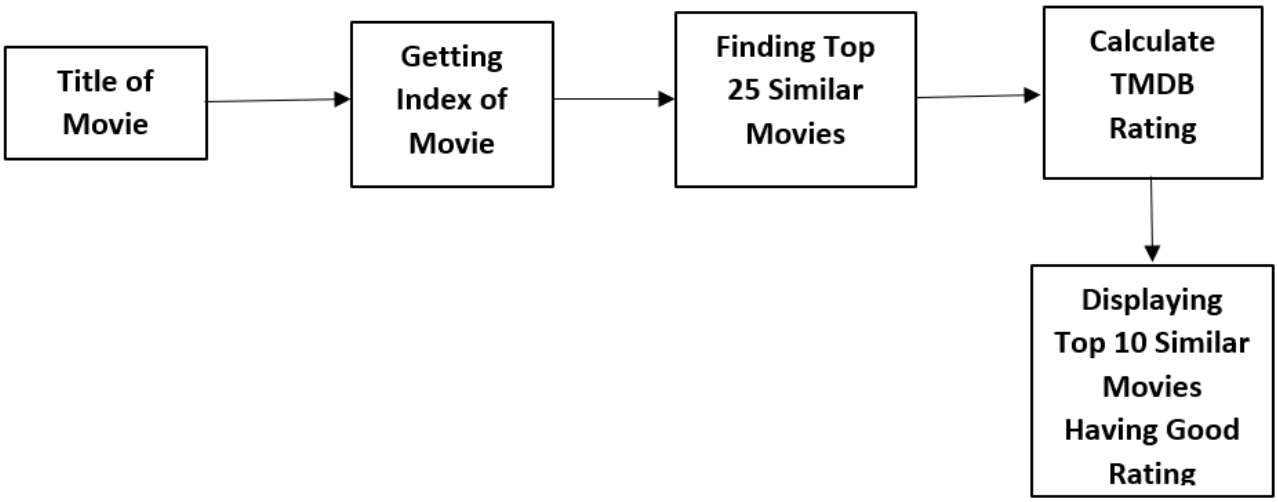

The figure below shows the block diagram for generating recommendations using Content-Based Recommender System. The input to this system is title of movie whose similar movies need to be generated. The outcome of this system is the chart comprising of similar movies of the given movie name obtained using Cosine Similarity Matrix. The same function is used for both movie description based and meta data-based recommendation system. These systems are different in context of their pre-processing steps, thus, resulting in different Pair-wise similarity matrix. The detailed implementation of these systems has been discussed in the next section.

**Figure 15**
**Block Diagram of Generating Recommendations for Content Based Recommender System**




The figure below shows the block diagram for improved Content-Based Recommender System which apart from considering movie content such as meta data or description, also considers the popularity of movies obtained using weighted rating formula. The input to this system is title of movie whose similar movies need to be generated. The outcome of this system is the chart comprising of similar movies of the given movie name obtained using Cosine Similarity Matrix and sorted according to their popularity. The Implementation section presents the steps taken to implement this system.

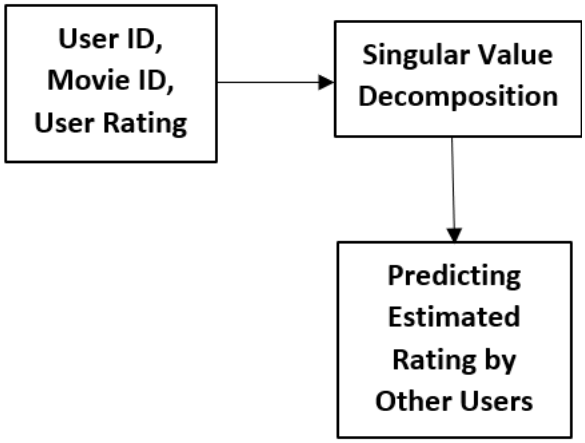
**Figure 16**
**Block Diagram for Popularity & Ratings Based Improved Content Based Recommender System**




## *5.3 Collaborative Filtering Based Recommender System*

The figure below shows the block diagram for Collaborative filtering-based movie recommender system. The approach follows the use of Singular Value Decomposition. The input to this system is the user id, movie id, and user rating. The output of the system is the estimated rating for the movie according to the user. In this study, model-based approach of Collaborative filtering has been used.

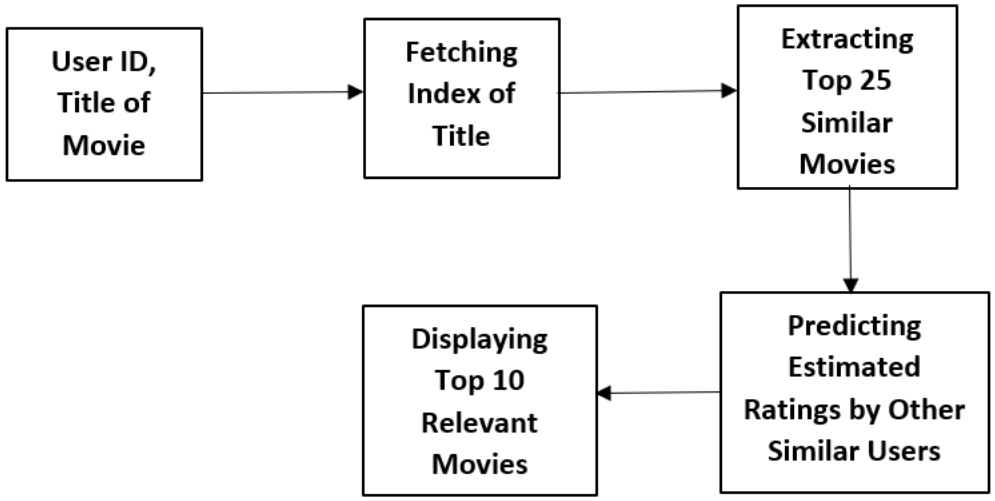
**Figure 17**
**Block Diagram for Collaborative Filtering Approach Based Recommender Systems**




## *5.4 Hybrid Filtering Based Recommender System*

The figure below shows the block diagram for generating recommendations using Hybrid Filtering-Based Recommender System. The input to this system is title of movie whose similar movies need to be generated and id of user for whom the recommendations need to be generated. The outcome of this system is the chart comprising of similar movies of the given movie name obtained for the user given along with the predicted estimated movie ratings. This approach utilises the architecture of both Content-Based and Collaborative Filtering based Recommender Systems. The detailed implementation of this systems has been discussed in the next section.

**Figure 18**
**Block Diagram for Hybrid Filtering Based Recommender Systems**



# 6 Implementation

This section gives the description of steps taken to implement several recommender systems to generate recommendations for movies. There are numerous recommender systems developed in work using the Python programming language. Python programming language offers numerous packages that help in applying various operations on data. All the recommenders have been built on Google Colab so that advantage of GPU and large storage can be taken. The detailed explanation of all the recommenders is given below.

## *6.1 Simple Recommender System*

There are two variants of simple recommender system developed in this work. The first recommender system is generating recommendations in general. It is considering the entire dataset for producing a list of Top Movies. The output of this recommender system is not tailored to the users. Indeed, it generates the similar movie recommendations to every user on the basis of popularity of movie and genre. It is based on the assumption that large audience generally tend to watch popular movies.

Initially the TMDB ratings are calculated for the movies. The weighted rating can be found using the given formula:

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

Where

v = number of votes for the movie,

m = minimum votes required to be listed in the chart,

R = average rating of the movie, and

C = mean vote across the whole report

The values of v and R are present in the dataset. But the values of c and m need to be calculated. The code snippet below shows the code to extraction of these values.

**Figure 19**
**Code Snippet showing the gathering the values of v, R, C, and m**

```
#getting the value of v (number of votes) from dataset
vote_counts = md[md['vote_count'].notnull()]['vote_count'].astype('int')

#getting the value of R (average rating of the movie) from dataset
vote_averages = md[md['vote_average'].notnull()]['vote_average'].astype('int')

#Calculating the value of mean votes C
C = vote_averages.mean()
C

#Calculating the value of m which is minimum votes required to be listed in the chart
m = vote_counts.quantile(0.95)
m
```

It is observed from the figure that value of v represented by '*vote_counts*' is extracted where the data is not null in the dataset. These values are then converted to integer values. The similar pattern can be followed for getting the value of R which is average rating of the movie represented by the 'vote*_averages'*. The mean value is calculated for all the average rating of the movies using the *mean( )* function of python. The value of m signifies the minimum number of votes which every movie should have to be present in the chart. The quantile value is set to be 95% as threshold or

cutoff value. The values obtained for C and m are 5.24 and 434 respectively. This implies that **a movie should at least have 434 votes and 5.244 rating in order to be qualified to be present in the Top Movie Chart**.

**Figure 20**
**Code Snippet showing the filters applied on original dataset**

```
#Creating a dataframe with useful columns only
#Title, Year, Vote Count, Vote Average, Popularity, Genres
#Extracting information about movies where vote count and vote average is not null
# and vote count should be greater than value of m = 434 (here)
qualified = md[(md['vote_count'] >= m) &
               (md['vote_count'].notnull()) &
               (md['vote_average'].notnull())][['title', 'year', 'vote_count', 'vote_average', 'popularity', 'genres']]

#Converting vote count and vote average values from float to int
qualified['vote_count'] = qualified['vote_count'].astype('int')
qualified['vote_average'] = qualified['vote_average'].astype('int')
qualified.shape
```

The characteristics obtained for top movies are then applied to the movie dataset to filter out the top movies. The movies which satisfy the given criteria can be stored in different dataframe. There are 45466 movies present in the dataset used. Only, **2274 movies qualified** the filters and may present on the top movies chart based on their ratings.

**Figure 21**
**Code Snippet showing the calculations of IMDB's weighted rating**

```
#Defining function for calculating IMDB's weighted rating
def weighted_rating(x):
    v = x['vote_count']
    R = x['vote_average']
    return (v/(v+m) * R) + (m/(m+v) * C)
```

```
#Calculating and adding IMDB weighted rating in dataframe
qualified['wr'] = qualified.apply(weighted_rating, axis=1)
```

```
#Sorting the values for movies according to IMDB weighted rating
#Fetching only Top 250 movies
qualified = qualified.sort_values('wr', ascending=False).head(250)
```

The '*weighted rating'* function is defined to calculate the IMDB rating of movie. These values are calculated for the qualified 2274 movies. The values obtained are sorted in descending order so that popular movies come on top. The top 250 popular movies are selected. The recommendations made by the simple recommender system are present in Results section.

The next variant of simple recommender systems built is displaying movies according to the **Genres**. This means top 15 movies for comedy genre, another 15 for Family genre, etc. There are more than 30 genres available in the dataset which are shown below in the figure.

**Figure 22**
**Genres present in the movie dataset**

```
#Different categories/genres of movies
gen_md.genre.unique()
```

```
array(['Animation', 'Comedy', 'Family', 'Adventure', 'Fantasy', 'Romance',
       'Drama', 'Action', 'Crime', 'Thriller', 'Horror', 'History',
       'Science Fiction', 'Mystery', 'War', 'Foreign', nan, 'Music',
       'Documentary', 'Western', 'TV Movie', 'Carousel Productions',
       'Vision View Entertainment', 'Telescene Film Group Productions',
       'Aniplex', 'GoHands', 'BROSTA TV',
       'Mardock Scramble Production Committee', 'Sentai Filmworks',
       'Odyssey Media', 'Pulser Productions', 'Rogue State', 'The Cartel'],
      dtype=object)
```

The recommender system accepts the name of the genre. The first step is to fetch the movie details belonging to the given genre. The similar pattern to the simple recommender system is then followed afterwards where the values of v, m, c, and r are calculated. The filters are applied to the newly formed dataset containing movies belonging to the given genre. The next step is to calculate the IMDB weighted rating for the qualified movies. Top 250 movies belonging to the given genre having high IMDB weighted ratings are selected and returned. The working code of the same is shown in the figure below.

**Figure 23**
**Code Snippet showing extracting top movies for the given genre**

```
#Displaying Top 250 movies belonging to the specified genre
#by using IMDB weighted ratings for each set of movies belonging to similar genres
def build_chart(genre, percentile=0.95):
    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
```

## *6.2 Content-Based Recommender System*

This recommender system has been developed to conquer the issues occurring in Simple Recommender systems which can only generate general recommendations without taking users' choices and interests in considerations. These systems are based on the assumptions that if a user likes or watches any movie, the user might be interested in watching movies similar to that. Thus, similar movies can be obtained and displayed to the user. This similarity can be based on Movie Description such as Title, Overview, or Tagline, or Movie Cast, Crew, Keywords, Genre, etc. Thus, in this thesis, two types of Content-based Recommender Systems have been developed which are Movie Description Based and Metadata based.

Since, there is limited availability of resources, and the dataset consists of nearly 45000 movies. Thus, the content-based recommender system has been developed using the small dataset which is a subset of original dataset comprising of 9099 movies.

### 6.2.1 Movie Description Based Recommender System

**Data Preparation-** The null values present in the attribute is initially imputed. The description column is created for movies by concatenating the 'overview' and 'tagline' columns. The null values present in the description column are filled with spaces.

**Creating a TF-IDF vectorizer-** The next step is to create a TF-IDF vectorizer for the descriptions using the '*tfidfVectorizer'* function. The vectorizer consists of several parameters whose values are set as analyzer as *'word',* ngram_range as (1,2), stop words as *'English'*. The resulting matrix is of the shape (9099, 268124). Here, 9099 shows the movie instances whilst the 268124 represents the feature vectors present in these movie descriptions.

**Forming Similarity Matrix-** The next step is to form the matrix containing the pair-wise similarity between the movies based on description. Cosine similarity has been used in this study to assess the similarity between two movies. The cosine similarity has been obtained by finding the dot product of the TF-IDF vectors. To determine the dot product of the TF-IDF vectors, *'linear kernel'* function of sklearn library is used. The result is the matrix comprising of the pair-wise similarity values between the movies.

**Generating Recommendations-** *Get_recommendations( )* function has been defined to generate the recommendations for user on the basis of similarity between description of movies. The code snippet below shows the function defined for generating the similar recommendations for the given movie name. It can be observed in the figure that the function accepts the name of the movie as an input. The index of the movie is fetched afterwards. The values of Cosine Similarity are obtained for the extracted index value. These similarity scores are then sorted in descending order so that most relevant score appears on the top. Top 30 similarity scores are then extracted from the list. The title of the movies are extracted for the given movie indices and displayed.

**Figure 24**
**Code Snippet for Movie description-based Recommender System**

```
#Generating Recommendations
def get_recommendations(title):
    #Getting the index of the title of the movie
    idx = indices[title]
    print("Index of the movie is: ",idx)

    #Extracting cosine similarity of the given movie with all the remaining movies
    sim_scores = list(enumerate(cosine_sim[idx]))
    print(sim_scores)

    #Sorting Similarity Score in descending order
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    print('Sorted Similarity Score in descending order:')
    print(sim_scores)

    #Extracting Top 30 similarity score
    sim_scores = sim_scores[1:31]

    #Extracting movie names for the matched indices
    movie_indices = [i[0] for i in sim_scores]
    print('Recommended Movie indices:')
    print(movie_indices)

    return titles.iloc[movie_indices]
```

### 6.2.2 Meta Data Based Recommender System

This recommender system generates movie recommendations keeping in mind the necessary details about movies such as cast, crew, keywords, etc known as meta data. In the current dataset, there is no metadata information available. There are separate csv files given – *credits.csv* and *keywords.csv* containing information about meta data. The figure below shows the top 5 rows of both the csv files.

**Figure 25**
**Top 5 rows of credits.csv file**

| | cast | crew | id |
|---|---|---|---|
| 0 | [{'cast_id': 14, 'character': 'Woody (voice)',... | [{'credit_id': '52fe4284c3a36847f8024f49', 'de... | 862 |
| 1 | [{'cast_id': 1, 'character': 'Alan Parrish', '... | [{'credit_id': '52fe44bfc3a36847f80a7cd1', 'de... | 8844 |
| 2 | [{'cast_id': 2, 'character': 'Max Goldman', 'c... | [{'credit_id': '52fe466a9251416c75077a89', 'de... | 15602 |
| 3 | [{'cast_id': 1, 'character': "Savannah 'Vannah... | [{'credit_id': '52fe44779251416c91011acb', 'de... | 31357 |
| 4 | [{'cast_id': 1, 'character': 'George Banks', '... | [{'credit_id': '52fe44959251416c75039ed7', 'de... | 11862 |

**Figure 26**
**Top 5 rows of keywords.csv file**

| | id | keywords |
|---|---|---|
| 0 | 862 | [{'id': 931, 'name': 'jealousy'}, {'id': 4290,... |
| 1 | 8844 | [{'id': 10090, 'name': 'board game'}, {'id': 1... |
| 2 | 15602 | [{'id': 1495, 'name': 'fishing'}, {'id': 12392... |
| 3 | 31357 | [{'id': 818, 'name': 'based on novel'}, {'id':... |
| 4 | 11862 | [{'id': 1009, 'name': 'baby'}, {'id': 1599, 'n... |

**Dataframe Preparation**- It can be seen from figure above credits.csv file contains features- cast, crew, and id; keywords.csv file contains the features- id and keywords. Id is here acting as the primary key comprising of necessary information about the movie with the given id. Thus, this id can be used to merge the above fetched information with the existing dataframe. It is important to merge this fetched information with existing dataframe because the current dataframe comprises of more details about movie such as name of the movie, year of release, etc.

After merging the necessary information in lieu of cast, crew, and keywords with the current dataframe, there are 45463 instances which are same as before but 28 features now. Since the number of records are very large, larger computational resources would be required. Thus, the size of the dataset has been reduced using the smaller set data file. The smaller subset of file is already present in the source of the dataset and has been extracted from there only. The subset consists of nearly 9219 instances. Thus, the dataset which would now be used by recommender system comprises of 9219 instances and 28 attributes.

The next step is to calculate the cast and crew size and adding them to the dataframe. It can be observed from looking at the size of cast and crew that there are large number of cast and crew available. It is not wise to consider all the cast members. Generally, Top 3 cast members play dominant role in movie and influence user to like or dislike it. Thus, top 3 cast members are selected for each movie.

**Figure 27**
**Displaying cast, crew, keywords, cast_size, and crew_size in dataframe**

| title | video | vote_average | vote_count | year | cast | crew | keywords | cast_size | crew_size |
|---|---|---|---|---|---|---|---|---|---|
| Toy Story | False | 7.7 | 5415.0 | 1995 | [{'cast_id': 14, 'character': 'Woody (voice)',... | [{'credit_id': '52fe4284c3a36847f8024f49', 'de... | [{'id': 931, 'name': 'jealousy'}, {'id': 4290,... | 13 | 106 |
| Jumanji | False | 6.9 | 2413.0 | 1995 | [{'cast_id': 1, 'character': 'Alan Parrish', '... | [{'credit_id': '52fe44bfc3a36847f80a7cd1', 'de... | [{'id': 10090, 'name': 'board game'}, {'id': 1... | 26 | 16 |
| Grumpier Old Men | False | 6.5 | 92.0 | 1995 | [{'cast_id': 2, 'character': 'Max Goldman', 'c... | [{'credit_id': '52fe466a9251416c75077a89', 'de... | [{'id': 1495, 'name': 'fishing'}, {'id': 12392... | 7 | 4 |
| Waiting to Exhale | False | 6.1 | 34.0 | 1995 | [{'cast_id': 1, 'character': 'Savannah 'Vannah... | [{'credit_id': '52fe44779251416c91011acb', 'de... | [{'id': 818, 'name': 'based on novel'}, {'id':... | 10 | 10 |
| Father of the Bride Part II | False | 5.7 | 173.0 | 1995 | [{'cast_id': 1, 'character': 'George Banks', '... | [{'credit_id': '52fe44959251416c75039ed7', 'de... | [{'id': 1009, 'name': 'baby'}, {'id': 1599, 'n... | 12 | 7 |

The important crew member which influences the movie is Director. Thus, name of the director is fetched from the crew and added separately in the dataframe. The weightage is given to director. Hence, the director's name has been replicated thrice to generate recommendations according to the director. Additionally, columns- cast and keywords are pre-processed so that it can help in generating recommendations. The figure below shows the pre-processed dataframe for metadata-based movie recommender systems.

**Figure 28**
**Pre-processed Dataframe for Metadata-based Movie Recommender System**

| year | cast | crew | keywords | cast_size | crew_size | director |
|---|---|---|---|---|---|---|
| 1995 | [tomhanks, timallen, donrickles] | [{'credit_id': '52fe4284c3a36847f8024f49', 'de... | [jealousy, toy, boy, friendship, friends, riva... | 13 | 106 | [johnlasseter, johnlasseter, johnlasseter] |
| 1995 | [robinwilliams, jonathanhyde, kirstendunst] | [{'credit_id': '52fe44bfc3a36847f80a7cd1', 'de... | [board game, disappearance, based on children'... | 26 | 16 | [joejohnston, joejohnston, joejohnston] |
| 1995 | [waltermatthau, jacklemmon, ann-margret] | [{'credit_id': '52fe466a9251416c75077a89', 'de... | [fishing, best friend, duringcreditsstinger, o... | 7 | 4 | [howarddeutch, howarddeutch, howarddeutch] |
| 1995 | [whitneyhouston, angelabassett, lorettadevine] | [{'credit_id': '52fe44779251416c91011acb', 'de... | [based on novel, interracial relationship, sin... | 10 | 10 | [forestwhitaker, forestwhitaker, forestwhitaker] |
| 1995 | [stevemartin, dianekeaton, martinshort] | [{'credit_id': '52fe44959251416c75039ed7', 'de... | [baby, midlife crisis, confidence, aging, daug... | 12 | 7 | [charlesshyer, charlesshyer, charlesshyer] |

It can be seen from above figure that the cast column consists of three names only which are also processed, pre-processed keywords column, and director column comprising of director's name thrice.

**Metadata Preparation**- The meta data used for generating recommendations is produced by joining keywords, cast, director, and genres of movie. The figure below shows the metadata prepared for generating recommendations for movie.

**Figure 29**
**Metadata Preparation**

```
0        jealousi toy boy friendship friend rivalri boy...
1        boardgam disappear basedonchildren'sbook newho...
2        fish bestfriend duringcreditssting waltermatth...
3        basedonnovel interracialrelationship singlemot...
4        babi midlifecrisi confid age daughter motherda...
                             ...
40952    friendship sidneypoitier wendycrewson jayo.san...
41172    bollywood akshaykumar ileanad'cruz eshagupta t...
41225    bollywood hrithikroshan poojahegde kabirbedi a...
41391    monster godzilla giantmonst destruct kaiju hir...
41669    music documentari paulmccartney ringostarr joh...
Name: soup, Length: 9219, dtype: object
```

**Count Vectorizer**- The next step is to form a count vectorizer matrix containing the number of times a word appears in a document. The count vectorizer is applied for the metadata which has been prepared. The vectorizer consists of several parameters whose values are set as analyzer as *'word',* ngram_range as (1,2), stop words as *'English'.*

**Calculating Pair-wise similarity between movies-** The next step is to prepare the similarity matrix containing the pair-wise similarity for the movies. The *'cosine_similarity( )'* function is used to calculate the similarity among the movies. The function takes input as resulting count matrix from count vectorizer and calculate the similarity among them.

**Generating Recommendations-** The *'get_recoomendations( )'* function defined for generating recommendations based on the movie description is used for generating the recommendations based on metadata also. The results obtained are discussed in next section.

### 6.2.3 Popularity and Ratings Content-Based Recommender System

The recommendations generated by both prior recommender systems are considering only either movie description such as overview and tagline or movie metadata such as cast, crew, genre, keywords. They are not generating recommendations considering ratings and popularity of the movies. This results in sometimes, poor recommendations. The user should be displayed popular and good quality content in the order. Thus, this system tends to improve the recommendations generated by the meta data recommender system by applying weighted rating formula. The function used to generate the improved recommendations is shown below.

**Figure 30**
**Function to improve Recommendations**

```
def improved_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[1:26]
    movie_indices = [i[0] for i in sim_scores]

    movies = smd.iloc[movie_indices][['title', 'vote_count', 'vote_average', 'year']]
    vote_counts = movies[movies['vote_count'].notnull()]['vote_count'].astype('int')
    vote_averages = movies[movies['vote_average'].notnull()]['vote_average'].astype('int')
    C = vote_averages.mean()
    m = vote_counts.quantile(0.60)

    qualified = movies[(movies['vote_count'] >= m) & (movies['vote_count'].notnull()) & (movies['vote_average'].notnull())]
    qualified['vote_count'] = qualified['vote_count'].astype('int')
    qualified['vote_average'] = qualified['vote_average'].astype('int')
    qualified['wr'] = qualified.apply(weighted_rating, axis=1)
    qualified = qualified.sort_values('wr', ascending=False).head(10)
    return qualified
```

## *6.3 Collaborative Filtering Based Recommender System*

There are certain drawbacks of Content-Based Recommender Systems that these systems do not generate the recommendations subjected to personal choices and interests of users, To conquer this issue, this system is generating recommendations capturing tastes of users. Unlike Content-based recommender systems, these systems do not produce similar recommendations for all the users. Instead, they generate the recommendations differently for each user depending on their personal interests and choice. This system does not consider movie content such as overview, tagline, cast, crew, genre, popularity, director, indeed, it considers the behaviour of other similar users.

In this study, the collaborative filtering-based recommender system has been developed using Singular Value Decomposition (SVD) algorithm from Surprise library. This algorithm generates the recommendations minimising the root mean square error. The dataset used for this Recommender system is given below:

**Figure 31**
**Dataframe used for Collaborative filtering**

| | userId | movieId | rating | timestamp |
|---|---|---|---|---|
| **0** | 1 | 31 | 2.5 | 1260759144 |
| **1** | 1 | 1029 | 3.0 | 1260759179 |
| **2** | 1 | 1061 | 3.0 | 1260759182 |
| **3** | 1 | 1129 | 2.0 | 1260759185 |
| **4** | 1 | 1172 | 4.0 | 1260759205 |

The *cross_validate( )* function has been used to run a cross validation according to the 5 folds defined in the cv argument of the function. The function computes root mean squared error and mean absolute error as accuracy measure.

**Figure 32**
**Collaborative Filtering Approach using SVD**

```
svd = SVD()
cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=5)
```

```
{'test_rmse': array([0.89881492, 0.89536055, 0.8977806 , 0.90440227, 0.89197405]),
 'test_mae': array([0.69423804, 0.68695171, 0.69130627, 0.69649013, 0.68833663]),
 'fit_time': (8.15396499633789,
  5.602790832519531,
  5.810094356536865,
  7.058130741119385,
  5.5311970710754395),
 'test_time': (0.1584477424621582,
  0.49848270416259766,
  0.17769360542297363,
  0.15530180931091309,
  0.3657872676849365)}
```

## *6.4 Hybrid Recommender System*

In this approach of recommender systems, both the approaches namely- Content Based and Collaborative filtering are joined together to recommend movies to the user. The small dataset has been used to build the hybrid recommender system. The values of two features of movie are fetched- movieId and tmbId from the small dataset. Afterwards, title of the movie is fetched to prepare a dataframe shown in figure below.

**Figure 33**
**Top 5 rows of the dataframe comprising movieId, tmbid, and title of movies**

| title | movieId | id |
|---|---|---|
| Toy Story | 1 | 862.0 |
| Jumanji | 2 | 8844.0 |
| Grumpier Old Men | 3 | 15602.0 |
| Waiting to Exhale | 4 | 31357.0 |
| Father of the Bride Part II | 5 | 11862.0 |

The following snippet shows the implementation of Hybrid Recommender System.

**Figure 34**
**Code Snippet Showing Hybrid Recommender System Approach**

```
#building hybrid recommender
def hybrid(userId, title):
    #fetching index of the title
    idx = indices[title]
    #fetching tmdbID
    tmdbId = id_map.loc[title]['id']
    movie_id = id_map.loc[title]['movieId']

    #Calculating Similarity
    sim_scores = list(enumerate(cosine_sim[int(idx)]))
    #Sorting cosine similarity in reverse order
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    #Slicing top 25 movies
    sim_scores = sim_scores[1:26]
    #fetching movie indices of top 25 movies
    movie_indices = [i[0] for i in sim_scores]

    #Displaying info of top 25 movies
    movies = smd.iloc[movie_indices][['title', 'vote_count', 'vote_average', 'year', 'id']]

    #collaborative filtering
    #predicting estimated ratings using SVD
    movies['est'] = movies['id'].apply(lambda x: svd.predict(userId, indices_map.loc[x]['movieId']).est)
    movies = movies.sort_values('est', ascending=False)
    return movies.head(10)
```

The hybrid recommender system accepts user id and title of the movie to generate the recommendations. Initially, the indexes of the movies are fetched whose title is given to the function. The similarity scores are fetched for the given movie title from the matrix containing movie similarity score. The matrix used consists of pair-wise similarity score between the movies based on meta data of the movie. The scores obtained are sorted in the descending order and topmost 25 similar movies are fetched.

The information such as title of the movie, vote count, average voting, year of release, movie id about top 25 movies is then extracted from the dataframe. The collaborative filtering is then used to predict the estimated ratings using SVD. The movies are then sorted according to the estimated ratings tailored to the given user id.

# 7 Testing and Results

This section presents the recommendations obtained with the implemented recommender systems.

## *7.1 Simple Recommender System*

This recommender system generates the movie recommendations irrespective of the user's choice. The top 15 movie chart that can be recommended to any user is shown below.

**Figure 35**
**Top 15 movie chart**

| | title | year | vote_count | vote_average | popularity | genres | wr |
|---|---|---|---|---|---|---|---|
| **15480** | Inception | 2010 | 14075 | 8 | 29.108149 | [Action, Thriller, Science Fiction, Mystery, A... | 7.917588 |
| **12481** | The Dark Knight | 2008 | 12269 | 8 | 123.167259 | [Drama, Action, Crime, Thriller] | 7.905871 |
| **22879** | Interstellar | 2014 | 11187 | 8 | 32.213481 | [Adventure, Drama, Science Fiction] | 7.897107 |
| **2843** | Fight Club | 1999 | 9678 | 8 | 63.869599 | [Drama] | 7.881753 |
| **4863** | The Lord of the Rings: The Fellowship of the Ring | 2001 | 8892 | 8 | 32.070725 | [Adventure, Fantasy, Action] | 7.871787 |
| **292** | Pulp Fiction | 1994 | 8670 | 8 | 140.950236 | [Thriller, Crime] | 7.868660 |
| **314** | The Shawshank Redemption | 1994 | 8358 | 8 | 51.645403 | [Drama, Crime] | 7.864000 |
| **7000** | The Lord of the Rings: The Return of the King | 2003 | 8226 | 8 | 29.324358 | [Adventure, Fantasy, Action] | 7.861927 |
| **351** | Forrest Gump | 1994 | 8147 | 8 | 48.307194 | [Comedy, Drama, Romance] | 7.860656 |
| **5814** | The Lord of the Rings: The Two Towers | 2002 | 7641 | 8 | 29.423537 | [Adventure, Fantasy, Action] | 7.851924 |
| **256** | Star Wars | 1977 | 6778 | 8 | 42.149697 | [Adventure, Action, Science Fiction] | 7.834205 |
| **1225** | Back to the Future | 1985 | 6239 | 8 | 25.778509 | [Adventure, Comedy, Science Fiction, Family] | 7.820813 |
| **834** | The Godfather | 1972 | 6024 | 8 | 41.109264 | [Drama, Crime] | 7.814847 |
| **1154** | The Empire Strikes Back | 1980 | 5998 | 8 | 19.470959 | [Adventure, Action, Science Fiction] | 7.814099 |
| **46** | Se7en | 1995 | 5915 | 8 | 18.45743 | [Crime, Mystery, Thriller] | 7.811669 |

It can be observed from above results that movies- **Inception, The Dark Knight, and Interstellar** are among the Top 3 popular movies among the movies present in the dataset. Since the Inception got the maximum number of votes which is 14075, thus it is present on the top of chart. The movies present in the chart have been sorted according to the column *'wr'* which shows the weighted ratings obtained for each movie. Though The Dark Knight has high popularity, but it has less number of vote counts, thus weighted rating calculated for the Inception is more than The Dark Knight.

Afterwards, top 15 movies belonging to the 'Comedy' and 'Romance' genre are extracted and results are displayed below.

**Figure 36**
**Top 15 Comedy movies**

| | title | year | vote_count | vote_average | popularity | wr |
|---|---|---|---|---|---|---|
| **351** | Forrest Gump | 1994 | 8147 | 8 | 48.307194 | 7.843994 |
| **1225** | Back to the Future | 1985 | 6239 | 8 | 25.778509 | 7.799850 |
| **18465** | The Intouchables | 2011 | 5410 | 8 | 16.086919 | 7.771794 |
| **22841** | The Grand Budapest Hotel | 2014 | 4644 | 8 | 14.442048 | 7.737837 |
| **2211** | Life Is Beautiful | 1997 | 3643 | 8 | 39.39497 | 7.674556 |
| **10309** | Dilwale Dulhania Le Jayenge | 1995 | 661 | 9 | 34.457024 | 7.408901 |
| **732** | Dr. Strangelove or: How I Learned to Stop Worr... | 1964 | 1472 | 8 | 9.80398 | 7.316986 |
| **3342** | Modern Times | 1936 | 881 | 8 | 8.159556 | 7.025524 |
| **883** | Some Like It Hot | 1959 | 835 | 8 | 11.845107 | 6.992045 |
| **26564** | Deadpool | 2016 | 11444 | 7 | 187.860492 | 6.929222 |
| **1236** | The Great Dictator | 1940 | 756 | 8 | 9.241748 | 6.928845 |
| **13724** | Up | 2009 | 7048 | 7 | 19.330884 | 6.888022 |
| **22131** | The Wolf of Wall Street | 2013 | 6768 | 7 | 16.382422 | 6.883711 |
| **30315** | Inside Out | 2015 | 6737 | 7 | 23.985587 | 6.883213 |
| **13746** | The Hangover | 2009 | 6324 | 7 | 23.947351 | 6.876149 |

**Figure 37**
**Top 15 Romance movies**

| | title | year | vote_count | vote_average | popularity | wr |
|---|---|---|---|---|---|---|
| **351** | Forrest Gump | 1994 | 8147 | 8 | 48.307194 | 7.869860 |
| **10309** | Dilwale Dulhania Le Jayenge | 1995 | 661 | 9 | 34.457024 | 7.582757 |
| **876** | Vertigo | 1958 | 1162 | 8 | 18.20822 | 7.298862 |
| **40251** | Your Name. | 2016 | 1030 | 8 | 34.461252 | 7.235471 |
| **883** | Some Like It Hot | 1959 | 835 | 8 | 11.845107 | 7.117619 |
| **1132** | Cinema Paradiso | 1988 | 834 | 8 | 14.177005 | 7.116921 |
| **19901** | Paperman | 2012 | 734 | 8 | 7.198633 | 7.041055 |
| **37863** | Sing Street | 2016 | 669 | 8 | 10.672862 | 6.984338 |
| **1639** | Titanic | 1997 | 7770 | 7 | 26.88907 | 6.916316 |
| **19731** | Silver Linings Playbook | 2012 | 4840 | 7 | 14.488111 | 6.869789 |
| **40882** | La La Land | 2016 | 4745 | 7 | 19.681686 | 6.867399 |
| **23437** | Maleficent | 2014 | 4607 | 7 | 19.467404 | 6.863766 |
| **22168** | Her | 2013 | 4215 | 7 | 13.829515 | 6.852269 |
| **20910** | The Great Gatsby | 2013 | 3885 | 7 | 17.598936 | 6.840970 |
| **23512** | The Fault in Our Stars | 2014 | 3868 | 7 | 16.274653 | 6.840341 |

It can be seen from the results that the movie Forest Grump is the most popular movie belonging to the comedy and romance genre released in the year 1994. The movie received 8147 votes from the users, average voting of 8, popularity of 48.30. The calculated weighted rating for the movie is 7.8439 according to the Comedy genre dataframe and 7.869 according to the Romance genre dataframe. There is a difference in the weighted rating values since the values of C and M calculated are different for both of the dataframes. Additionally, the movie belongs to more than

one genre in the dataset, Therefore, the movie 'Forest Grump' is present in movie charts of both the genres.

**Observations from Simple Recommender System**- It should also be observed that all the recommendations generated by simple recommender system is general for all the users. The results can be viewed as the content displayed under the **'Top Trending'** section which is same for all the users. The trending movies can also be accessed by the user according to the genres such as Popular Comedy Movies, Popular Action Movies, etc. Several entertainment providers use this algorithm to display general recommendations for user such as Netflix, YouTube, Amazon.
There are certain limitations observed for Simple Recommender system such as these are not capable of putting forward the options according to user's interest.

## *7.2 Content-Based Recommender System*

### 7.2.1 Movie-Description Based Recommender System

**Figure 38**
**Getting Recommendations according to Movie Description of 'Jumanji'**

```
get_recommendations('Jumanji').head(10)

Index of the movie is:  1
[(0, 0.006804755671748422), (1, 1.0000000000000007), (2, 0.015310620291169728), (3, 0.0), (4, 0.002236809940247703 2), (5, 0.014747265950384272), (6, 0.0), (7, 0.0), (8, 0.033137051354
Sorted Similarity Score in descending order:
[(1, 1.0000000000000007), (8889, 0.06504282809768652), (8608, 0.057312344608660 42), (6392, 0.053803762848217), (8154, 0.05222633255408065), (3196, 0.05211723461442551), (8670, 0.05126
Recommended Movie indices:
[8889, 8608, 6392, 8154, 3196, 8670, 5356, 8211, 6323, 4082, 5803, 1644, 2080, 6545, 6285, 4173, 8410, 1618, 4027, 7308, 1063, 7048, 4337, 6199, 8164, 7564, 8569, 6557, 5475, 6293]
8889                      Pixels
8608      Guardians of the Galaxy
6392                   Stay Alive
8154               Wreck-It Ralph
3196           Dungeons & Dragons
8670                        Ouija
5356     Night of the Living Dead
8211              Would You Rather
6323                Grandma's Boy
4082    The Giant Spider Invasion
Name: title, dtype: object
```

As discussed above, initially the movie name 'Jumanji' has been passed to the function to generate recommendations.

The index of the movie is then fetched which is 1.

The next step is to obtain the similarity score with different movies. Thus, (0, 0.006), (1, 1.00), (2, 0.0153),…. are the pair-wise similarity scores which are fetched using Cosine Similarity score matrix. 0 here shows the index of the movie and second argument 0.006 shows the similarity score obtained of movie with 0 index and 1 index.

The similarity scores are then sorted in descending order giving the maximum similar movie on the top. It can be seen that movie with index 1 has the similarity score of 1 signifying the movie itself. The next top most similar movie is with index 8889, 8608, and so on. Top 30 movies are selected on the basis of similarity score. Top 10 are displayed along with the title as shown in figure above.

It can be observed that user who has watched 'Jumanji' movie, the system would recommend similar movies to the user like *'Pixels', 'Guardians of the Galaxy', 'Stay Alive', 'Wreck-It Ralph', 'Dungeons & Dragons', 'Ouija', 'Night of the Living Dead', 'Would you Rather', 'Grandma's Boy', 'The Giant Spider Invasion'*.

Similarly, if the user has watched movie 'The Dark Knight', the system would recommend movies like *'The Dark Knight Rises', 'Batman Forever', 'Batman Returns', 'Batman: Under the Red*

*Hood', 'Batman', 'Batman: Year One', 'Batman: Mask of the Phantasm', 'JKF', 'Batman: The Dark Knight Returns, Part 1'.*

**Figure 39**
**Getting Recommendations according to Movie Description of 'The Dark Knight'**

```
Index of the movie is:  6900
[(0, 0.0), (1, 0.007774131635450035), (2, 0.0), (3, 0.0), (4, 0.0), (5, 0.004610245611341961), (6, 0.0), (7, 0.0), (8, 0.0039417018132425 67), (9, 0.0), (10, 0.0), (11, 0.0150339017235
Sorted Similarity Score in descending order:
[(6900, 0.9999999999999991), (7931, 0.17137364862331894), (132, 0.12244415240106257), (1113, 0.10088975721500881), (8227, 0.08476241853836834), (7565, 0.084196920814179), (524, 0.0816
Recommended Movie indices:
[7931, 132, 1113, 8227, 7565, 524, 7901, 2579, 2696, 8165, 6144, 7933, 5511, 4489, 7344, 7242, 3537, 2893, 1135, 8680, 8917, 1240, 6740, 1652, 6667, 4028, 8371, 8719, 3730, 4160]
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
```

### 7.2.2 Metadata based Recommender System

This system has generated recommendations for the user when the user passes movie title. Considering the movie title, the system extracts the necessary meta data about the given movie title and generate recommendations further based on the similarity in the metadata.

**Figure 40**
**Getting Recommendations according to Meta data of 'The Dark Knight'**

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

Index of the movie is:  6981
[(0, 0.0), (1, 0.0), (2, 0.0), (3, 0.020965696734438367), (4, 0.0), (5, 0.08481041912882635), (6, 0.0), (7, 0.04774099160262887), (8, 0.04193139346887673), (9, 0.05017452060042545), (10, 0.020965696734438 36
Sorted Similarity Score in descending order:
[(6981, 1.0000000000000018), (8031, 0.4927569020810309), (6218, 0.4685756640036763), (6623, 0.34375418461967155), (2085, 0.29584010465345556), (7648, 0.2713848825944775), (4145, 0.2657887169768753), (3381,
Recommended Movie indices:
[8031, 6218, 6623, 2085, 7648, 4145, 3381, 8613, 7659, 1134, 8927, 5943, 1260, 9024, 4021, 5809, 7362, 7561, 7582, 8001, 2754, 132, 2131, 2448, 5098, 8026, 8727, 9121, 7133, 217]
8031          The Dark Knight Rises
6218                  Batman Begins
6623                 The Prestige
2085                    Following
7648                    Inception
4145                     Insomnia
3381                      Memento
8613                  Interstellar
7659    Batman: Under the Red Hood
1134                 Batman Returns
Name: title, dtype: object
```

It can be observed that if the user has watched movie 'The Dark Knight', the system would recommend movies like *'The Dark Knight Rises', 'Batman Begins, 'The Prestige', 'Following', 'Inception', 'Insomnia', 'Momento', 'Interstellar', 'Batman: Under the Red Hood', 'Batman Returns'.*

It can be easily observed that movies recommended by the system have director 'Christopher Nolan'. The system has given more weightage to Director Name while generating recommendations instead of giving weightage to other metadata such as cast, genres, or keywords. Thus, it can be said that weightage of feature highly affects the recommendations. Several other features can also be experimented by increasing their weightage and generating recommendations.

### 7.2.3 Popularity and Ratings based Recommender System

Since it is possible that movies generated by above systems are not always popular because they do not take popularity and ratings into consideration. Thus, this system generates the improved recommendations by reviewing popularity and ratings using weighted rating formula used in Simple Recommender system. The recommendations generated for the movie 'The Dark Knight' are shown below.

**Figure 41**
**Improved Recommendations**

| | title | vote_count | vote_average | year | wr |
|---|---|---|---|---|---|
| **7648** | Inception | 14075 | 8 | 2010 | 7.917588 |
| **8613** | Interstellar | 11187 | 8 | 2014 | 7.897107 |
| **6623** | The Prestige | 4510 | 8 | 2006 | 7.758148 |
| **3381** | Memento | 4168 | 8 | 2000 | 7.740175 |
| **8031** | The Dark Knight Rises | 9263 | 7 | 2012 | 6.921448 |
| **6218** | Batman Begins | 7511 | 7 | 2005 | 6.904127 |
| **1134** | Batman Returns | 1706 | 6 | 1992 | 5.846862 |
| **132** | Batman Forever | 1529 | 5 | 1995 | 5.054144 |
| **9024** | Batman v Superman: Dawn of Justice | 7189 | 5 | 2016 | 5.013943 |
| **1260** | Batman & Robin | 1447 | 4 | 1997 | 4.287233 |

It can be observed that if the user has watched movie 'The Dark Knight', the system would recommend movies like *'Inception', 'Interstellar', 'The Prestige', 'Momento', 'The Dark Knight Rises', 'Batman Begins', 'Batman Returns', 'Batman Forever', 'Batman v Superman: Dawn of Justice', 'Batman & Robin'*. It can be observed from results that some of the movies which have been recommended earlier are no longer recommended. The order of recommendation has been changed. The system is generating more popular movies on top.

## *7.3 Collaborative Filtering*

This system is generating recommendations which are tailored according to the users' personal interests and choices. This system is generating recommendations using the Singular Value Decomposition (SVD) offered by the Surprise library. The estimated ratings are predicted for user with id 1 for the movie id 302. The SVD predicted that user 1 might give 2.84 ratings to the movie 302. This system does not consider about popularity, description, or meta data of the movie. Instead, it generates the ratings on the basis of how other users similar to the given user have given the ratings to the movie.

## *7.4 Hybrid Recommendation System*

**Figure 42**
**Recommendations using Hybrid Approach for user 1 and movie Avatar**

| | title | vote_count | vote_average | year | id | est |
|---|---|---|---|---|---|---|
| **1011** | The Terminator | 4208.0 | 7.4 | 1984 | 218 | 3.129897 |
| **522** | Terminator 2: Judgment Day | 4274.0 | 7.7 | 1991 | 280 | 3.124812 |
| **974** | Aliens | 3282.0 | 7.7 | 1986 | 679 | 3.115874 |
| **8401** | Star Trek Into Darkness | 4479.0 | 7.4 | 2013 | 54138 | 3.071065 |
| **8658** | X-Men: Days of Future Past | 6155.0 | 7.5 | 2014 | 127585 | 3.052325 |
| **1668** | Return from Witch Mountain | 38.0 | 5.6 | 1978 | 14822 | 3.014696 |
| **1621** | Darby O'Gill and the Little People | 35.0 | 6.7 | 1959 | 18887 | 2.870933 |
| **2014** | Fantastic Planet | 140.0 | 7.6 | 1973 | 16306 | 2.870143 |
| **4347** | Piranha Part Two: The Spawning | 41.0 | 3.9 | 1981 | 31646 | 2.844025 |
| **4017** | Hawk the Slayer | 13.0 | 4.5 | 1980 | 25628 | 2.844025 |

The hybrid system generates the movies- *'The Terminator', 'Terminator 2:Judgement Day', 'Aliens', 'Star Trek into darkness', 'X-Men: Days of Future Past', 'Return from Witch Mountain', 'Darby O'Gill and the Little People', 'Fantastic Planet', 'Piranha Part Two: The Spawning', 'Hawk the Slayer'* for the user with id 1 and movie title – Avatar.

**Figure 43**
**Recommendations using Hybrid Approach for user 500 and movie Avatar**

| | title | vote_count | vote_average | year | id | est |
|---|---|---|---|---|---|---|
| **522** | Terminator 2: Judgment Day | 4274.0 | 7.7 | 1991 | 280 | 3.434330 |
| **1621** | Darby O'Gill and the Little People | 35.0 | 6.7 | 1959 | 18887 | 3.262900 |
| **1668** | Return from Witch Mountain | 38.0 | 5.6 | 1978 | 14822 | 3.135922 |
| **1011** | The Terminator | 4208.0 | 7.4 | 1984 | 218 | 3.074656 |
| **8658** | X-Men: Days of Future Past | 6155.0 | 7.5 | 2014 | 127585 | 3.058195 |
| **1376** | Titanic | 7770.0 | 7.5 | 1997 | 597 | 3.001172 |
| **7088** | Star Wars: The Clone Wars | 434.0 | 5.8 | 2008 | 12180 | 2.993309 |
| **922** | The Abyss | 822.0 | 7.1 | 1989 | 2756 | 2.973688 |
| **974** | Aliens | 3282.0 | 7.7 | 1986 | 679 | 2.973176 |
| **2014** | Fantastic Planet | 140.0 | 7.6 | 1973 | 16306 | 2.928536 |

The hybrid system generates the movies- *'Terminator 2:Judgement Day', 'Darby O'Gill and the Little People' , 'Return from Witch Mountain', 'The Terminator', 'X-Men: Days of Future Past', 'Titanic', 'Star Wars: The Clone Wars', 'The Abyss', 'Aliens', 'Fantastic Planet'* for the user with id 500 and movie title – Avatar.

It can be seen from above results that some of the movies like *'Aliens', 'Darby O'Gill and the Little People' ,'Return from Witch Mountain', 'The Terminator'* are generated for both the users but with different estimated ratings.

The order of the movie recommendation is also different for both of the users for the same movie. This proves that Hybrid Recommender system is giving recommendations according to the users' personal choices and tastes. This system is not going to generate general recommendations for all the users, instead generating personalised recommendations.

## *7.5 Comparison of Results*

This section shows the comparison of several Recommendations systems implemented. After studying more about different recommendation systems and implementing them, it can be said that the comparison between different recommender system approaches such as- Content Based, Collaborative filtering based, or hybrid filtering based is not feasible. These systems accept different inputs and generate different outcomes accordingly. The outcomes of all these approaches are useful in different context. The results obtained from one approach cannot supersede the results obtained using different approach.

However, the comparison of systems which are based on single approach is feasible. In this study, three recommender systems have been developed which are using Content-Based Recommendation approach. These are generating recommendations based on Movie Description, Metadata based, and popularity based. Although, the results are not wise to compare since different criteria is used to obtain recommendations. However, the recommendations obtained from different systems when the same movie name 'The Dark Knight' is given as input are summarised below in the table.

**Table 2**
**Recommendations from Different Content-Based Recommender Systems for movie 'The Dark Knight'**

| Movie Description Based | Metadata Based | Popularity Based |
|---|---|---|
| The Dark Knight Rises | The Dark Knight Rises | Inception |
| Batman Forever | Batman Begins | Interstellar |
| Batman Returns | The Prestige | The Prestige |
| Batman: The Dark Knight Returns, Part 2 | Following | Memento |
| Batman: Under the Red Hood | Inception | The Dark Knight Rises |
| Batman | Insomnia | Batman Begins |
| Batman: Year One | Memento | Batman Returns |
| Batman: Mask of the Phantasm | Interstellar | Batman Forever |
| JFK | Batman: Under the Red Hood | Batman v Superman: Dawn of Justice |
| Batman: The Dark Knight Returns, Part 1. | Batman Returns | Batman & Robin |

It can be observed from Table 2 that many of the movies such as 'The Dark Knight Rises', 'Batman Returns' satisfy all the criterion and thus recommended by all the three systems. Also, these are just top 10 movies from these recommender systems. There can be more intersection in the remaining data as well.

Thus, it can be said that all the approaches reviewed have their own applications and usefulness. The comparison among these approaches is not feasible.

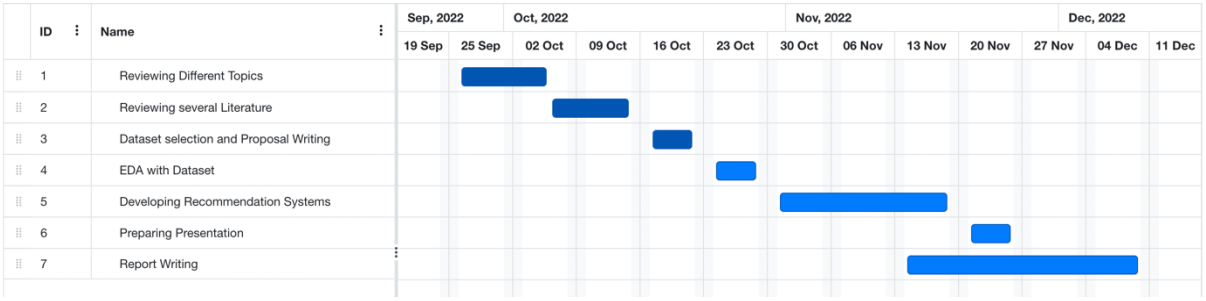
# 8 Project Management

This section presents the project schedule followed, risk management for the project, quality management, and Social, Legal, Ethical and Professional Considerations.

## *8.1 Project Schedule*

Project Schedule helps in adhering to the timelines and thus resulting in timely completion of the work. The figure below shows the planned schedule for the work prior to the start of work. All the aims and objectives set out for the work need to be completed in a time span of 11-12 weeks.

**Figure 44**
**Gantt Chart for Project Timelines**




The brief summary of each phase and time allocated is given below.

Topic Selection is very time-consuming and difficult task. This phase consumed nearly 1.5 weeks of the plan. The next step is to review several literatures subjected to the topic selected. It is very paramount to review the existing work as it helps in formulating the research questions and methodology.

The fourth week is utilised in searching appropriate dataset. Along with dataset search, this week is also utilised in finalising the research questions, aims and objectives, project plan, research methodology. A proposal document is framed in this duration which can be presented to give brief overview of the work.

As the dataset comprises of large number of features present in six different files, thus, the fifth week is utilised in gaining better understanding of the files present, their features, and their importance and use in developing different recommendation systems.

The next three phases are utilised in design of the systems to be developed, their implementation phase, and the additional things to be done to improve the performance of the recommenders built. The majority of the coding is ended by the eighth week. However, some changes and more analysis are also done during the report documentation period.

The ninth week is utilised in preparing the presentation illustrating the overview of study along with the results obtained with the experiments.

The reporting work is started from eighth week only since it is very important phase. The results obtained, learnings, conclusion of the work should be presented in a good manner. Thus, reporting started early and completed by the end of eleventh week.

The entire project schedule has been followed strictly resulting in completion of work in time.

## *8.2 Risk Management*

This section gives the overview of the risk strategies adopted to minimise the failure of work. There are no major risks involved in the study. However, certain things are taken care of while working such as

- Importing all the necessary packages and libraries so that functions offered by them can be used without any error.
- The syntax of the language is followed cautiously so that no error can be caused due to it.
- The textual data is pre-processed so that recommendations can be generated of good quality since the similarity matrix is obtained with pre-processed data only.
- The use of Google Colab ensures that there is no program failure due to limited resources since the necessary RAM and storage is present in Google Colab.

## *8.3 Quality Management*

It is very important to assess the quality of any work being done since it helps in understanding the reliability of the work. Thus, there are numerous performance metrics available for assessing the performance of several systems. However, the quality evaluation of recommender systems cannot be done by using these measures since the performance metrics require knowledge about actual outcomes which can be compared with the predicted outcome. There is not recommendations possible which are 100% accurate. Thus, it is not feasible to obtain the truth knowledge for the content-based and hybrid filtering approaches especially. Thus, the performance of systems developed in this work has been assessed qualitatively.

## *8.4 Social, Legal, Ethical and Professional Considerations*

There can be problem of ethics in the movie information. However, this information, ratings are available on public resources. Thus, there is not any issue subjected to data confidentiality and user privacy. Furthermore, the dataset used has also been referenced to give credits to the owner. Additionally, all the literatures that have been reviewed are referenced illustrating the credits to the authors.

# 9 Critical Appraisal

This section presents the brief discussion of the work, its results, learning and inferences. The aim of the study is to analyse, implement, and compare different movie recommendations systems. There are numerous movie recommendations systems available which are based on three broad categories namely Content-based filtering, collaborative filtering, and Hybrid filtering. Many recommender systems can be developed on the basis of these approaches. In this study, seven different recommendations systems have been implemented using the subset of MovieLens dataset which is available on Kaggle.

During reviewing literature, the knowledge of different recommendation systems has been gained. This phase helps in analysing the differences between all the aforementioned approaches. This phase built the understanding of need of different approaches, their importance, and usefulness in applications. The methodologies adopted in study has been adopted after this phase only.

All the recommendation systems have been successfully implemented using libraries of python programming language. All the recommendation systems except the collaborative filtering have been implemented from scratch. Singular Value Decomposition algorithm offered by the surprise module is used to implement the collaborative approach. The outcomes of all the recommendation systems have been analysed.

It has been inferred that simple recommender system generates the general recommendations which is same for everyone irrespective of users' choices and tastes. The recommendations can be made among all the data available or via classes such as top movies in the year 2010, 2016; top movies according to the genre; top movies according to cast; or any other features. The outcomes are based on the IMDB weighted rating formula.

The content-based recommender systems generated outcomes based on the content such as tagline or movie overview and keywords, director, etc. The results obtained are satisfactory but only this recommender system cannot fulfil the all the requirements of the recommender systems since it is not capable of generating recommendations which are personalised for users.

The collaborative filtering approach used in this study predicted the estimated rating for the given user and movie. There is not much work done in context of Collaborative filtering approach in this study due to lack of understanding and timing. However, it can be taken as future recommendation to enhance the work and learnings.

The next recommender system is the combination of above two approaches- Content Based and Collaborative filtering. The outcomes obtained using this recommender system is personalised for users. During result analysis, it has been observed that different recommendations are generated for different users when the same movie is given as the input. Thus, it can be inferred that the system is generating personalised recommendations.

The objective of the study is also to determine the best recommender system. However, it has been analysed that there are no best recommender systems. Every approach has its own limitations, advantages, and usefulness. Every recommender system is equally important depending on the user requirements. The major limitation of this study is that it is not feasible to determine the reliability of outcomes and thus systems due to the lack of data available in context of ground truth for recommendations.

# 10 Conclusions

This section provides the summary of the work that has been done along with the discussion of how aims and objectives of the work has been achieved. The section also presents further the future work that can be done to enhance the work. There is increasing evidence that recommender systems can be beneficial for addressing some of the information overload phenomenon due to the internet. With the evolution of recommender systems, there is an evolution of the web as well. Recommender systems tend to collect information about users from three different sources- content-based data from watched movies, demographic data from the profiles of users, and memory-based data which has been collected from item preferences of users. This information can be used by such systems to generate recommendations for the users.

## *10.1 Achievements*

The purpose of the study is to study, analyse, implement, and compare different Movie Recommendation Systems. While reviewing some theoretical frameworks, it has been initially identified that there are three major categories of Movie Recommendation Systems- Content-Based Filtering Recommendation Systems, Collaborative Filtering based Recommendation systems, and Hybrid Filtering based Recommendation Systems. While studying more from several resources, it has been identified that Content-based filtering can further be done on the basis of movie description such as Overview or tagline and movie meta data such as Actors, Crew, Director, Genre, Keywords, etc. Thus, the recommendation systems implemented along with their observations is discussed below.

Simple Recommender System- This recommender system generates general recommendations which can be viewed as Top movies on home page section. This system generated recommendations without considering users' choices, similarity. Instead, IMDB weighted rating formula has been used to generate the popular movies.

Popular Movie Recommendations according to the Genres- This recommender system is a variant of Simple Recommender system generating top popular recommendations. In this system, the recommendations are generated using the movie genres. The popular recommendations can be obtained from the model by giving it any genre such as Popular Drama movies, Popular Action movies, etc. These outcomes can be used by content providers to display recommendations according to the category on their page.

Movie Description Based Recommender Systems (Content-Based Recommendation System)- This recommender system generates recommendations for the user on the similarity of movie overview and taglines. These outcomes can be used by content providers to recommend similar movies in context of movie description to users which they have watched previously.

Movie Meta Data Based Recommender Systems (Content-Based Recommendation System)- This recommender system generates recommendations for the user on the similarity of movie metadata such as actors, director, keywords, and genres. These outcomes can be used by content providers to recommend similar movies in context of movie metadata to users which they have watched previously. The importance of contribution of Director feature is checked with this recommender. It has been observed that when any feature is given more weightage, the recommender systems give more priority to those features and generate recommendations accordingly. For instance, when the input to the system is 'The Dark Knight', the system returned majority of the movies having same director Christopher.

Popularity and Content-Based Recommender Systems- It has been observed from the results of content-based recommender systems that the recommendations are not sorted according to the popularity. In some instances, movies which are not of good quality are also being showed to the user. Moreover, the recommendations are also not sorted as per popularity. Thus, this system improved the recommendations of Content-Based Recommender system by using IMDB weighted rating formula.

Collaborative filtering Based Recommender system- This recommender has been implemented using Singular Value Decomposition (SVD). This predicts the estimated rating for the movie that user has not watched but watched by other similar users.

Hybrid Filtering Recommender System- In this system, the input to the system is title of the movie and user id. The recommendations are generated for different users given the same movie title. It has been observed that different set of movie recommendations have been generated with different weighted ratings. This is due to the fact that hybrid recommender systems generate recommendations using both approaches. The recommendations are tailored according to the user's choices, thereby, different order of recommendations for different users.

As discussed earlier also in the study, the quality of the recommendations have been assessed qualitatively. There is no evaluation metric used to compare the performance of these systems. Instead, the recommendations are generated by different systems for the same movie name. The recommendations are then analysed and compared with each other. The systems have been implemented using Python programming language on Google Colab using the MovieLens dataset extracted from Kaggle. All the objectives of the study have been fulfilled successfully in this work.

## *10.2 Future Work*

This section presents the ways in which the work can be extended further. Some of the potential recommendations are discussed below.

Use Memory-Based Collaborative Filtering- Currently, in this study, a model based collaborative filtering approach using SVD has been implemented and reviewed. The work can be enhanced further by analysing and implementing memory based collaborative approaches which are based on machine learning algorithms.

Use Deep Learning Methods- Since deep learning algorithms have given remarkable performance for several Natural Language Processing tasks such as Sentiment analysis, Text Classification, etc. Thus, deep learning techniques can also be used to implement Movie Recommender systems.

Experimenting weightage to other features- In this study, the weightage has been given to director name while generating recommendations based on meta data. Several other features can also be given different weightage and the results can be analysed and compared with the results obtained prior to giving weightage.

Using Some Other Dataset- The current study is based on MovieLens dataset. The study can further be enhanced by using different dataset having different features.

Introducing List of movies disliked by user- Since the user data is very crucial and helpful in generating recommendations, thus, the information about the movies which users do not like can also be proved beneficial while generating more personalised recommendations.

# 11 Student Reflections

This section presents the personal learning gained in this work. This project has been very challenging for me since it includes massive theory and implementation from scratch. There are large number of different systems developed. Thus, coding phase was very time-consuming and daunting.

I believe that I need to gain more understanding of Collaborative filtering. There are two approaches of Collaborative filtering- Memory Based and Model-Based. Currently, in this study, only model-based approach has been analysed. Thus, the better understanding of collaborative filtering can be gained while implementing memory-based recommender systems using deep learning or machine learning methods.

More research needs to be done in finding the evaluation criteria of general recommender systems, or how the reliability of these systems can be analysed. I will still continue to work, research, and learn more about this study.

# Bibliography and References

Bharti, R., & Gupta, D. (2019). Recommending top N movies using content-based filtering and collaborative filtering with hadoop and hive framework. In Recent Developments in Machine Learning and Data Analytics (pp. 109-118). Springer, Singapore.DOI: http://dx.doi.org/10.1007/978-981-13-1280-9 10

Bobadilla, J., Ortega, F., & Hernando, A. (2013). A. Guti errez,"Recommender systems survey,". Knowl.-Based Syst, 46, 109-132. DOI: http://dx.doi.org/10.1016/j.knosys.2013.03.012

Heung, V. C., Qu, H., & Chu, R. (2001). The relationship between vacation factors and socio-demographic and travelling characteristics: The case of Japanese leisure travellers. Tourism management, 22(3), 259-269.

imgur, Saunders Research Onion
https://i.imgur.com/vUQEitV.png

Jain, K. N., Kumar, V., Kumar, P., & Choudhury, T. (2018). Movie recommendation system: hybrid information filtering system. In Intelligent Computing and Information and Communication (pp. 677-686). Springer, Singapore.

Khatwani, S., & Chandak, M. B. (2016, September). Building Personalized and Non Personalized recommendation systems. In 2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT) (pp. 623-628). IEEE. DOI: http://dx.doi.org/10.1109/ICACDOT.2016.7877661

Konstan, J. A., and Riedl, J. (2012). "Recommender systems: From algorithms to user experience", User Modeling and User-Adapted Interaction 22(1-2), (101-123). DOI: http://dx.doi.org/10.1007/s11257-011-9112-x

Krzywicki, A., Wobcke, W., Kim, Y. S., Cai, X., Bain, M., Mahidadia, A., & Compton, P. (2015). Collaborative filtering for people-to-people recommendation in online dating: Data analysis and user trial. International Journal of Human-Computer Studies, 76, 50-66.

Kumar, P. V., & Reddy, V. R. (2014). A survey on recommender systems (RSS) and its applications. International Journal of Innovative Research in Computer and Communication Engineering, 2(8), 5254-5260.

Li, H., Cai, F., & Liao, Z. (2012, August). Content-based filtering recommendation algorithm using HMM. In 2012 Fourth International Conference on Computational and Information Sciences (pp. 275-277). IEEE.

Mansur, F., Patel, V., & Patel, M. (2017, March). A review on recommender systems. In 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS) (pp. 1-6). IEEE. DOI: http://dx.doi.org/10.1109/ICIIECS.2017.8276182

Miro.medium, Hybrid Recommender System
https://miro.medium.com/max/1270/1*XH3CT3gwQtwtOLvL-n48pg.jpeg

Nilashi, M., Ibrahim, O., & Bagherifard, K. (2018). A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. Expert Systems with Applications, 92, 507-520.DOI: http://dx.doi.org/10.1016/j.eswa.2017.09.058

Patel, A., Thakkar, A., Bhatt, N., & Prajapati, P. (2019). Survey and evolution study focusing comparative analysis and future research direction in the field of recommendation system specific to collaborative filtering approach. In Information and Communication Technology for Intelligent Systems (pp. 155-163). Springer, Singapore. DOI: http://dx.doi.org/10.1007/978-981-13-1742-216

Rahul, K., & Om, V. P. (2017). An effective collaborative movie recommender system with cuckoo. Egyptian Informatics Journal, 18, 105-112. DOI: http://dx.doi.org/10.1016/j.eij.2016.10.002

Sahoo, A. K., Pradhan, C., & Mishra, B. S. P. (2019, April). SVD based privacy preserving recommendation model using optimized hybrid item-based collaborative filtering. In 2019 international conference on communication and signal processing (ICCSP) (pp. 0294-0298). IEEE.

Sharma, M., & Mann, S. (2013). A survey of recommender systems: approaches and limitations. International Journal of Innovations in Engineering and Technology, 2(2), 8-14.

Son, J., & Kim, S. B. (2017). Content-based filtering for recommendation systems using multiattribute networks. Expert Systems with Applications, 89, 404-412.

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. Advances in artificial intelligence, 2009.

Vitalflux (2022), Collaborative-Filtering Approach
https://vitalflux.com/wp-content/uploads/2022/08/collaborative-filtering-recommender-system.png

Wu, C. M., Garg, D., and Bhandary, U. (2018) “Movie Recommendation System Using Collaborative Filtering”, IEEE 9th International Conference on Software Engineering and Service Science, (pp. 11-15).

Yash(2019), Content-Based Filtering
https://www.yash.com/wp-content/uploads/2019/03/recommendation-system11.jpg

Yang, C., Chen, X., Liu, L., Liu, T., & Geng, S. (2018, June). A hybrid movie recommendation method based on social similarity and item attributes. In International Conference on Swarm Intelligence (pp. 275-285). Springer, Cham.

Zhang, R., Bao, H., Sun, H., Wang, Y., & Liu, X. (2016). Recommender systems based on ranking performance optimization. Frontiers of Computer Science, 10(2), 270-280.

Zhou, T., Chen, L., & Shen, J. (2017, July). Movie recommendation system employing the user-based cf in cloud computing. In 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC) (Vol. 2, pp. 46-50). IEEE.

# Appendix A – Link to Dataset and Code

The link to the dataset used in this study is:
https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset

The link to the code file illustrating the implementation of the work is:
https://colab.research.google.com/drive/1Jyl6cG9297BQTzAjnrS8nV9F3Xou18Q5?usp=sharing

# Appendix B – Certificate of Ethics Approval

The figure below shows the screenshot of certificate of ethics approval.

Movie Recommendation System P143546

Coventry University

## Certificate of Ethical Approval

| | |
|---|---|
| Applicant: | Guddu Kumar Shah |
| Project Title: | Movie Recommendation System |

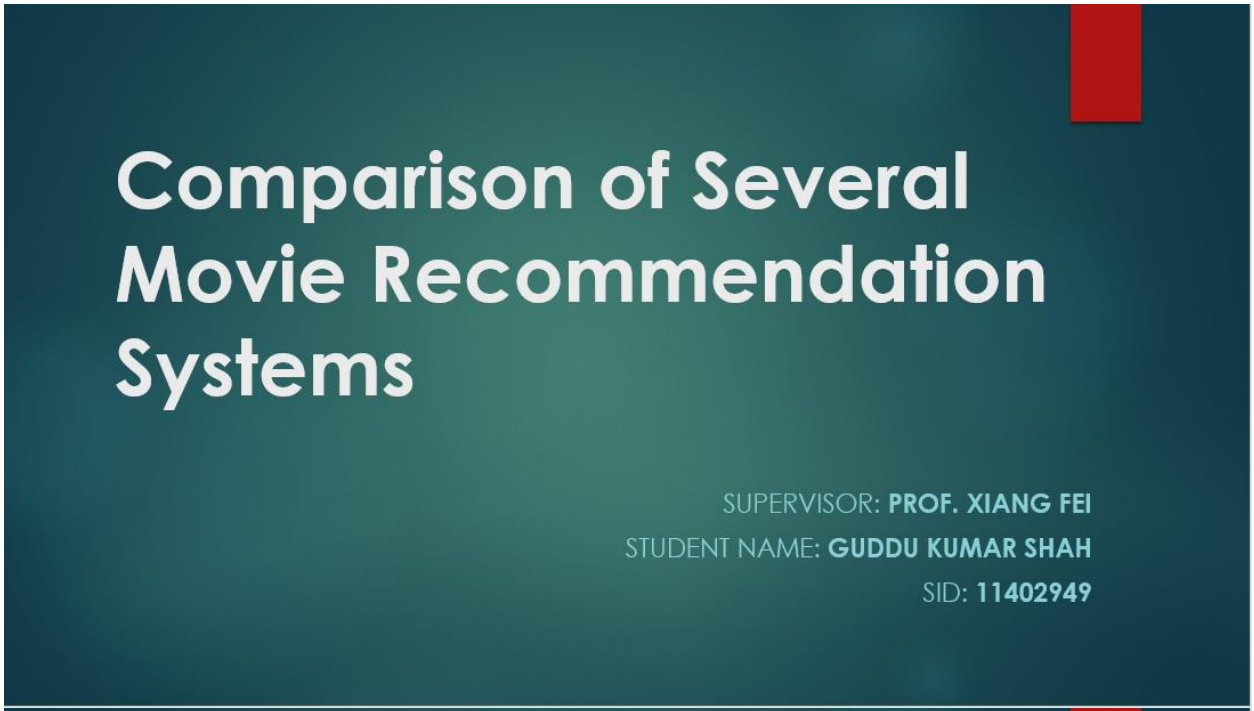
This is to certify that the above named applicant has completed the Coventry University Ethical Approval process and their project has been confirmed and approved as Low Risk

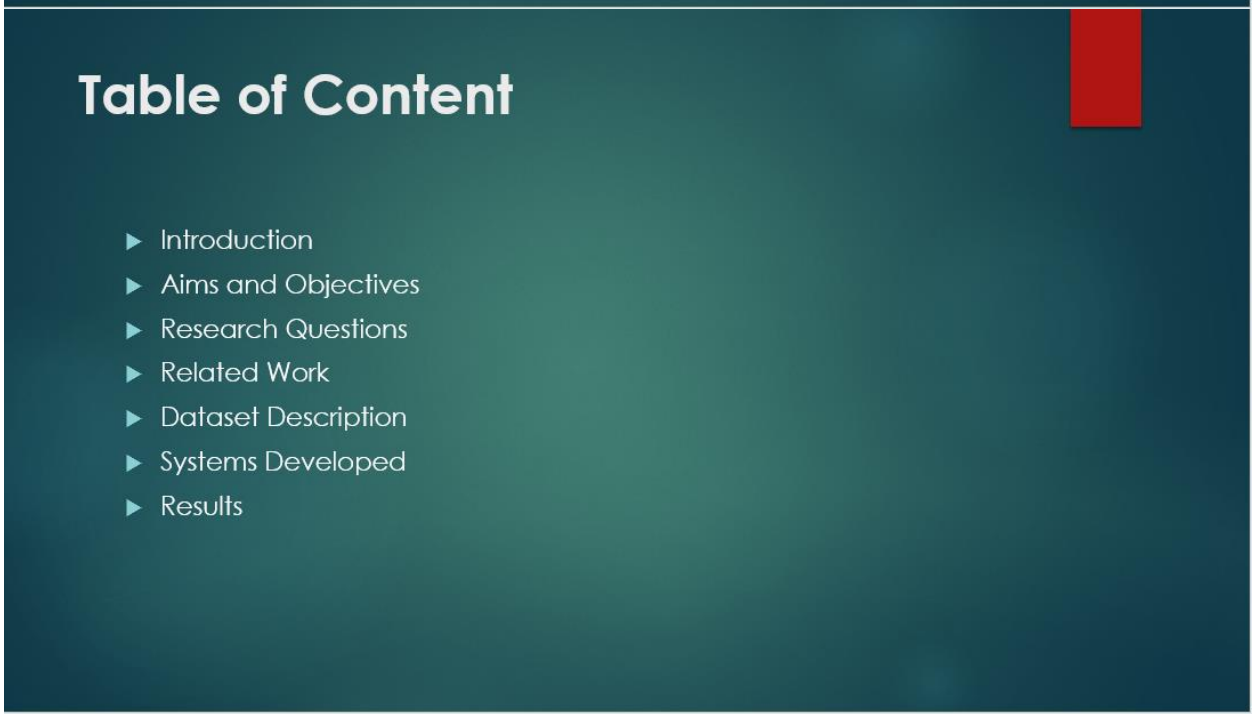
| | |
|---|---|
| Date of approval: | 18 Oct 2022 |
| Project Reference Number: | P143546 |

Guddu Kumar Shah (7151CEM) Page 1 18 Oct 2022

# Appendix C – Project Presentation

The figures below show the screenshots of project presentation submitted.



# Introduction

- The Recommender System helps users find content and reduce information overload by predicting their interests and making recommendations based on their interests.
- It is a challenging task for users to search information of their interest due to information overload. Additionally, Content Providers also face a challenge to differentiate their content from the rest.
- Thus, the outcomes can be useful for both **Customers** finding information on the internet and **Content Providers** pushing information on the internet.

# Aims and Objectives

The aim of this study is to implement and study the behaviour of several Movie Recommender Systems on the same dataset.

The objectives of this study are given below:

- Searching for suitable dataset
- Studying about different Recommender systems
- Generating movie recommendations which are personalised to users; according to the genres; some trending movies in general; recommendations considering Movies metadata such as Cast, Crew, Taglines, etc.
- Studying the impact of weightage of different features on recommendations.

# Research Questions

- Exploring different types of Recommender Systems that can be developed based on user information, movie details.
- What role does user details play in generating recommendations?
- What information of movies can be used to generate recommendations?
- How does the different weightage given to specific features of movies such as Cast, Crew impact the Recommender Systems?
- Given a movie name, how can the similar movies be identified?

# Related Work

- **Reddy et al.,2019**

Algorithm- Content-Based Filtering

Dataset- MovieLens

The authors considered genres only for generating recommendations and suggested that various other features can also be taken into consideration.

- **Ching-She Wu et al.,2018**

Algorithm- Collaborative Filtering

Dataset- Yahoo Research Web Scope Database

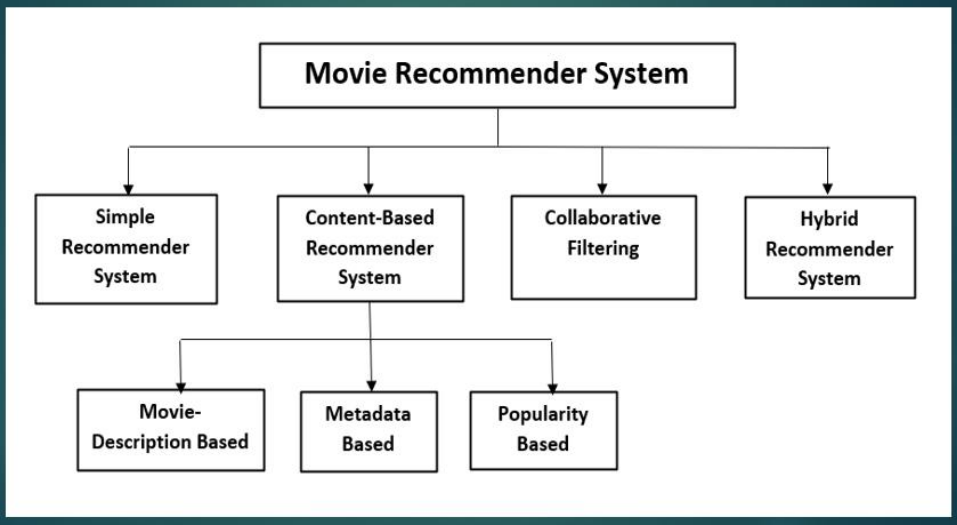
The authors developed user based and item based recommender systems and form groups of similar movies using K nearest neighbour algorithm.

# Dataset Description

- The dataset has been taken from Kaggle whose origin is TMDB API.
- Consists of 45466 rows and 24 features.

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| adult | False | False | False | False | False |
| belongs_to_collection | {'id': 10194, 'name': 'Toy Story Collection', ... | NaN | {'id': 119050, 'name': 'Grumpy Old Men Collect... | NaN | {'id': 96871, 'name': 'Father of the Bride Col... |
| budget | 30000000 | 65000000 | 0 | 16000000 | 0 |
| genres | [{'id': 16, 'name': 'Animation'}, {'id': 35, '... | [{'id': 12, 'name': 'Adventure'}, {'id': 14, '... | [{'id': 10749, 'name': 'Romance'}, {'id': 35, ... | [{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam... | [{'id': 35, 'name': 'Comedy'}] |
| homepage | http://toystory.disney.com/toy-story | NaN | NaN | NaN | NaN |
| id | 862 | 8844 | 15602 | 31357 | 11862 |
| imdb_id | tt0114709 | tt0113497 | tt0113228 | tt0114885 | tt0113041 |
| original_language | en | en | en | en | en |
| original_title | Toy Story | Jumanji | Grumpier Old Men | Waiting to Exhale | Father of the Bride Part II |
| overview | Led by Woody, Andy's toys live happily in his ... | When siblings Judy and Peter discover an encha... | A family wedding reignites the ancient feud be... | Cheated on, mistreated and stepped on, the wom... | Just when George Banks has recovered from his ... |
| popularity | 21.946943 | 17.015539 | 11.7129 | 3.859495 | 8.387519 |
| poster_path | /rhIRbceoE9lR4veEXuwCC2wARtG.jpg | /vzmL6fP7aPKNKPRTFnZmiUfciyV.jpg | /6ksm1sjKMFLbO7UY2i6G1ju9SML.jpg | /16XOMpEaLWkrcPqSQqhTmeJuqQl.jpg | /e64sOI48hQXyru7naBFyssKFxVd.jpg |
| production_companies | [{'name': 'Pixar Animation Studios', 'id': 3}] | [{'name': 'TriStar Pictures', 'id': 559}, {'na... | [{'name': 'Warner Bros.', 'id': 6194}, {'name':... | [{'name': 'Twentieth Century Fox Film Corporat... | [{'name': 'Sandollar Productions', 'id': 5842}... |
| production_countries | [{'iso_3166_1': 'US', 'name': 'United States o... | [{'iso_3166_1': 'US', 'name': 'United States o... | [{'iso_3166_1': 'US', 'name': 'United States o... | [{'iso_3166_1': 'US', 'name': 'United States o... | [{'iso_3166_1': 'US', 'name': 'United States o... |
| release_date | 1995-10-30 | 1995-12-15 | 1995-12-22 | 1995-12-22 | 1995-02-10 |
| revenue | 373554033.0 | 262797249.0 | 0.0 | 81452156.0 | 76578911.0 |
| runtime | 81.0 | 104.0 | 101.0 | 127.0 | 106.0 |
| spoken_languages | [{'iso_639_1': 'en', 'name': 'English'}] | [{'iso_639_1': 'en', 'name': 'English'}, {'iso... | [{'iso_639_1': 'en', 'name': 'English'}] | [{'iso_639_1': 'en', 'name': 'English'}] | [{'iso_639_1': 'en', 'name': 'English'}] |
| status | Released | Released | Released | Released | Released |
| tagline | NaN | Roll the dice and unleash the excitement! | Still Yelling. Still Fighting. Still Ready for... | Friends are the people who let you be yourself... | Just When His World Is Back To Normal... He's ... |
| title | Toy Story | Jumanji | Grumpier Old Men | Waiting to Exhale | Father of the Bride Part II |
| video | False | False | False | False | False |
| vote_average | 7.7 | 6.9 | 6.5 | 6.1 | 5.7 |
| vote_count | 5415.0 | 2413.0 | 92.0 | 34.0 | 173.0 |

# Systems Developed



# Simple Recommender System

- Generating general Recommendations to every user on the basis of popularity of movie and genre. Personalized recommendations are not given to the user.
- Idea- There are high chances that large audience will like more popular movies.
- Implementation- Sort the movies on the basis of Ratings and popularity and then display top movies of the list.

# Simple Recommender System

IMDB's weighted rating formula is used to construct the Top Movies Chart. The formula is given as:

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

where,

v = number of votes for the movie

m = minimum votes required to be listed in the chart

R = average rating of the movie

C = mean vote across the whole report

## Content-Based Recommendation System

- To overcome drawbacks of Simple Recommender Systems.
- Idea- Generating Personalized Recommendations by building 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.

1 **Movie-Description Based-** Considering Movie Overviews and Taglines.

2 **Metadata Based-** Considering Movie Cast, Crew, Keywords, and Genre.

3 **Popularity Based-** Considering Popularity along with movie data.

- Cosine Similarity is used to find the similarity among the movies.

## Collaborative Filtering Recommendation System

- Limitations of Content Based movies- Recommending movies close to a certain movie only without considering genres; not recommending movies considering personal tastes of users.
- This system is designed to conquer the limitations of Content-Based system.
- Idea- Users similar to each other can be used to predict how much user will like a particular product or service that haven't been experienced yet by the user but liked by similar users.
- Algorithm used- Singular Value Decomposition (SVD)

# Hybrid Recommendation System

- Combination of Content-Based Recommender and Collaborative Filtering.

Input- UserID and Title of Movie

Output- Similar movies sorted on the basis of expected ratings by that particular user.

# Results- Simple Recommender System

A chart of Top 15 movies has been displayed. A movie must have at least 434 votes on TMDB, 5.244 or above as average rating to be recommended.

| | title | year | vote_count | vote_average | popularity | genres | wr |
|---|---|---|---|---|---|---|---|
| 15480 | Inception | 2010 | 14075 | 8 | 29.108149 | [Action, Thriller, Science Fiction, Mystery, A... | 7.917588 |
| 12481 | The Dark Knight | 2008 | 12269 | 8 | 123.167259 | [Drama, Action, Crime, Thriller] | 7.905871 |
| 22879 | Interstellar | 2014 | 11187 | 8 | 32.213481 | [Adventure, Drama, Science Fiction] | 7.897107 |
| 2843 | Fight Club | 1999 | 9678 | 8 | 63.869599 | [Drama] | 7.881753 |
| 4863 | The Lord of the Rings: The Fellowship of the Ring | 2001 | 8892 | 8 | 32.070725 | [Adventure, Fantasy, Action] | 7.871787 |
| 292 | Pulp Fiction | 1994 | 8670 | 8 | 140.950236 | [Thriller, Crime] | 7.868660 |
| 314 | The Shawshank Redemption | 1994 | 8358 | 8 | 51.645403 | [Drama, Crime] | 7.864000 |
| 7000 | The Lord of the Rings: The Return of the King | 2003 | 8226 | 8 | 29.324358 | [Adventure, Fantasy, Action] | 7.861927 |
| 351 | Forrest Gump | 1994 | 8147 | 8 | 48.307194 | [Comedy, Drama, Romance] | 7.860656 |
| 5814 | The Lord of the Rings: The Two Towers | 2002 | 7641 | 8 | 29.423537 | [Adventure, Fantasy, Action] | 7.851924 |
| 256 | Star Wars | 1977 | 6778 | 8 | 42.149697 | [Adventure, Action, Science Fiction] | 7.834205 |
| 1225 | Back to the Future | 1985 | 6239 | 8 | 25.778509 | [Adventure, Comedy, Science Fiction, Family] | 7.820813 |
| 834 | The Godfather | 1972 | 6024 | 8 | 41.109264 | [Drama, Crime] | 7.814847 |
| 1154 | The Empire Strikes Back | 1980 | 5998 | 8 | 19.470959 | [Adventure, Action, Science Fiction] | 7.814099 |
| 46 | Se7en | 1995 | 5915 | 8 | 18.45743 | [Crime, Mystery, Thriller] | 7.811669 |

Top 15 movie generated using Simple Recommender Systems

## Results- Content Based Recommender System

| Movie Description Based | Metadata Based | Popularity Based |
|---|---|---|
| The Dark Knight Rises | The Dark Knight Rises | Inception |
| Batman Forever | Batman Begins | Interstellar |
| Batman Returns | The Prestige | The Prestige |
| Batman: The Dark Knight Returns, Part 2 | Following | Memento |
| Batman: Under the Red Hood | Inception | The Dark Knight Rises |
| Batman | Insomnia | Batman Begins |
| Batman: Year One | Memento | Batman Returns |
| Batman: Mask of the Phantasm | Interstellar | Batman Forever |
| JFK | Batman: Under the Red Hood | Batman v Superman: Dawn of Justice |
| Batman: The Dark Knight Returns, Part 1. | Batman Returns | Batman & Robin |

Outputs for the Movie- The Dark Knight

In Meta data based recommender, the system has recognised that director of The Dark Knight and made recommendations accordingly due to the high weightage given to the director.
Popularity Based Recommender also takes into consideration popularity and ratings along with movie name.

## Results- Collaborative Filtering

- SVD algorithm has been used to predict the estimated rating for user for a given movie.
- Input: Userid, Movieid
- Output: Estimated Ratings for user whose id is given.

For ex: For user having userId-1, there are high chances that user will give 2.844 rating to the movie with id 302.

It works purely on the basis of an assigned movie ID and tries to predict ratings based on how the other users have predicted the movie.

# Results- Hybrid Recommender System

| | title | vote_count | vote_average | year | id | est |
|---|---|---|---|---|---|---|
| 1011 | The Terminator | 4208.0 | 7.4 | 1984 | 218 | 3.129897 |
| 522 | Terminator 2: Judgment Day | 4274.0 | 7.7 | 1991 | 280 | 3.124812 |
| 974 | Aliens | 3282.0 | 7.7 | 1986 | 679 | 3.115874 |
| 8401 | Star Trek Into Darkness | 4479.0 | 7.4 | 2013 | 54138 | 3.071065 |
| 8658 | X-Men: Days of Future Past | 6155.0 | 7.5 | 2014 | 127585 | 3.052325 |
| 1668 | Return from Witch Mountain | 38.0 | 5.6 | 1978 | 14822 | 3.014696 |
| 1621 | Darby O'Gill and the Little People | 35.0 | 6.7 | 1959 | 18887 | 2.870933 |
| 2014 | Fantastic Planet | 140.0 | 7.6 | 1973 | 16306 | 2.870143 |
| 4347 | Piranha Part Two: The Spawning | 41.0 | 3.9 | 1981 | 31646 | 2.844025 |
| 4017 | Hawk the Slayer | 13.0 | 4.5 | 1980 | 25628 | 2.844025 |

When userid is 1 and movie is 'Avatar'

| | title | vote_count | vote_average | year | id | est |
|---|---|---|---|---|---|---|
| 522 | Terminator 2: Judgment Day | 4274.0 | 7.7 | 1991 | 280 | 3.434330 |
| 1621 | Darby O'Gill and the Little People | 35.0 | 6.7 | 1959 | 18887 | 3.262900 |
| 1668 | Return from Witch Mountain | 38.0 | 5.6 | 1978 | 14822 | 3.135922 |
| 1011 | The Terminator | 4208.0 | 7.4 | 1984 | 218 | 3.074656 |
| 8658 | X-Men: Days of Future Past | 6155.0 | 7.5 | 2014 | 127585 | 3.058195 |
| 1376 | Titanic | 7770.0 | 7.5 | 1997 | 597 | 3.001172 |
| 7088 | Star Wars: The Clone Wars | 434.0 | 5.8 | 2008 | 12180 | 2.993309 |
| 922 | The Abyss | 822.0 | 7.1 | 1989 | 2756 | 2.973688 |
| 974 | Aliens | 3282.0 | 7.7 | 1986 | 679 | 2.973176 |
| 2014 | Fantastic Planet | 140.0 | 7.6 | 1973 | 16306 | 2.928536 |

When userid is 500 and movie is 'Avatar'

It can be observed from both the figures that different recommendations are obtained for different users given the same movie name. This shows that recommendations are more personalized and tailored towards the given users.

# Conclusion

- In this study, six different Recommender systems for movies have been reviewed. The dataset used in this study is MovieLens dataset comprising of nearly 45000 records.
- The recommender systems have been developed using Python Programming Language on Google Colab. Several packages such as numpy, pandas, tf-idf vectors, surprise, etc have been used for building recommender systems.
- There are several performance metrics that can be used to assess the performance of these systems but these metrics require ground truth knowledge about which recommendations are correct. It is not feasible to obtain this information at a large scale according to this problem. Thus, the performance of each recommender has been assessed qualitatively.


Thank You!!