VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018

A Project Report on

"NETFLIX MOVIE RECOMMENDATION"

Submitted in partial fulfillment of the requirements as a part of the

Artificial Intelligence and Machine Learning Internship

For the award of degree of

Bachelor of Engineeringin
Information Science and Engineering

Submitted by

HITHAISHINI S
1RN18IS052

Under the guidance of

Mrs. Kusuma S

Assistant Professor

Dept. of ISE

ESTD:2001
An Institute with a Difference

Department of Information Science and Engineering

RNS Institute of Technology

Channasandra, Dr. Vishnuvardhan Road, RR Nagar Post,
Bengaluru – 560 098

2020 -2021

R N S Institute of Technology

Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560 098

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

(NBA ACCREDITED FOR ACADEMIC YEARS 2018-19, 2019-20, 2020-21)

CERTIFICATE

Certified that the project work entitled ***"Netflix Movie Recommendation"*** has been successfully carried out by **Hithaishini S** bearing **1RN18IS052**, bonafide student of **RNS Institute of Technology** in partial fulfillment of the requirements for the award of degree in **Bachelor of Engineering in Information Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during academic year 2021-2022. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

Mrs. Kusuma S	**Dr. Suresh L**	**Dr. M K Venkatesha**
Internship Guide	Professor & HOD	Principal
Assistant Professor	Department of ISE	RNSIT
Department of ISE	RNSIT	

External Viva

Name of the Examiners	**Signature with Date**
1._____	**1.**_____
2._____	**2.**_____

DECLARATION

I, **Hithaishini S [1RN18IS052]** student of VIII Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Internship work entitled ***"Netflix Movie Recommendation"*** has been carried out by me and submitted in partial fulfillment of the requirements for the VIII Semester degree of **Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belagavi during academic year 2021-2022.

Place: Bengaluru **HITHAISHINI S - 1RN18IS052**

Date:

ABSTRACT

In a growing population and population's varying interests, there is rapid growth of data collection that has led to a new era of information. Data is being used to create more efficient systems and this is where Recommendation Systems come into play. Recommendation Systems are a type of information filtering systems as they improve the quality of search results and provides items that are more relevant to the search item or are related to the search history of the user.

They are used to predict the rating or preference that a user would give to an item. Almost every major tech company has applied them in some form or the other: Amazon uses it to suggest products to customers, YouTube uses it to decide which video to play next on auto-play, and Facebook uses it to recommend pages to like and people to follow. Moreover, companies like Netflix and Spotify depend highly on the effectiveness of their recommendation engines for their business and success.

This recommender system is a simple algorithm whose aim is to provide the most relevant information to a user by discovering patterns in a dataset. The algorithm rates the items and shows the user the items that they would rate highly. An example of recommendation in action is when you visit Amazon and you notice that some items are being recommended to you or when Netflix recommends certain movies to you.

ACKNOWLEGMENT

At the very onset I would like to place our gratitude to all those people who helped me in making the Internship a successful one.

Coming up, this internship to be a success was not easy. Apart from the sheer effort, the enlightenment of the very experienced teachers also plays a paramount role because it is they who guided me in the right direction.

First of all, I would like to thank the **Management of RNS Institute of Technology** for providing such a healthy environment for the successful completion of internship work.

In this regard, I express sincere gratitude to our beloved Principal **Dr. M K Venkatesha**, for providing us all the facilities.

We are extremely grateful to our own and beloved Professor and Head of Department of Computer Science and Engineering, **Dr. Suresh L**, for having accepted to patronize me in the right direction with all her wisdom.

We place our heartfelt thanks to **Mrs. Kusuma S** Assistant Professor, Department of Information Science and Engineering for having guided internship and all the staff members of the department of Information Science and Engineering for helping at all times.

I thank **Mr. Aman Upadhyay**, Professor, NASTECH, for providing the opportunity to be a part of the Internship program and having guided me to complete the same successfully.

I also thank our internship coordinator **Dr. R Rajkumar** Associate Professor, Department of Information Science and Engineering. I would thank my friends for having supported me with all their strength and might. Last but not the least, I thank my parents for supporting and encouraging me throughout. I have made an honest effort in this assignment.

Date:_____

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 General Introduction

A recommendation system or recommendation engine is a model used for information filtering where it tries to predict the preferences of a user and provide suggestion based on these preferences. These systems have become increasingly popular nowadays and are widely used today in areas such as movies, music, books, videos, clothing, restaurants, food, places and other utilities. These systems collect information about a user's preferences and behaviour, and then use this information to improve their suggestions in the future.

Movies are a part and parcel of life. There are different types of movies like some for entertainment, some for educational purposes, some are animated movies for children, and some are horror movies or action films. Movies can be easily differentiated through their genres like comedy, thriller, animation, action etc. Other way to distinguish among movies can be either by releasing year, language, director etc. Watching movies online, there are a number of movies to search in our most liked movies. Movie Recommendation Systems helps us to search our preferred movies among all of these different types of movies and hence reduce the trouble of spending a lot of time searching our favourable movies. So, it requires that the movie recommendation system should be very reliable and should provide us with the recommendation of movies which are exactly same or most matched with our preferences.

A large number of companies are making use of recommendation systems to increase user interaction and enrich a user's shopping experience. Recommendation systems have several benefits, the most important being customer satisfaction and revenue. Movie Recommendation system is very powerful and important system. But, due to the problems associated with pure collaborative approach, movie recommendation systems also suffer with poor recommendation quality and scalability issues.

1.2 Machine Learning

Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed.

The name machine learning was coined in 1959 by Arthur Samuel. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data – such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions, through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or infeasible; example applications include email filtering, detection of network intruders or malicious insiders working towards a data breach, optical character recognition (OCR), learning to rank, and computer vision.

Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning. Machine learning can also be unsupervised and be used to learn and establish baseline behavioural profiles for various entities and then used to find meaningful anomalies.

1.3 Artificial intelligence

Artificial intelligence (AI), sometimes called machine intelligence, is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and other animals. In computer science AI research is defined as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals. Colloquially, the term "artificial intelligence" is applied when a machine mimics "cognitive" functions that humans associate with other human minds, such as "learning" and "problem solving". Modern machine capabilities generally classified as AI include successfully understanding human speech, competing at the highest level in strategic game systems (such as chess and Go), autonomously operating cars, intelligent routing etc.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. For most of its history, AI research has been divided into subfields that often fail to communicate with each other. These sub-fields are based on technical considerations, such as

particular goals (e.g. "robotics" or "machine learning"), the use of particular tools ("logic" or artificial neural networks), or deep philosophical differences. Subfields have also been based on social factors (particular institutions or the work of particular researchers).

The traditional problems (or goals) of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence is among the field's long-term goals. Approaches include statistical methods, computational intelligence, and traditional symbolic AI. Many tools are used in AI, including versions of search and mathematical optimization, artificial neural networks, and methods based on statistics, probability and economics. The AI field draws upon computer science, mathematics, psychology, linguistics, philosophy and many others.

The field was founded on the claim that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the nature of the mind and the ethics of creating artificial beings endowed with human-like intelligence which are issues that have been explored by myth, fiction and philosophy since antiquity. Some people also consider AI to be a danger to humanity if it progresses unabated. Others believe that AI, unlike previous technological revolutions, will create a risk of mass unemployment.

In the twenty-first century, AI techniques have experienced a resurgence following concurrent advances in computer power, large amounts of data, and theoretical understanding. AI techniques have become an essential part of the technology industry, helping to solve many challenging problems in computer science, software engineering and operations research. In simple terms, AI aims to extend and augment the capacity and efficiency of mankind in tasks of remaking nature and governing the society through intelligent machines, with the final goal of realizing a society where people and machines coexist harmoniously together.

1.4 Problem Statement

The goal of the project is to recommend a movie to the user by calculating the similarities between different users and then recommend movies to them as per the ratings given by the different users of similar taste.

1.5 Methodology for movie recommendation

We create recommenders using demographic, content- based and collaborative filtering. While demographic filtering is elementary and cannot be used practically, Hybrid Systems can take advantage of content-based and collaborative filtering as the two approaches are proved to be more effective.

- Demographic Filtering- They offer generalized recommendations to every user, based on movie popularity and/or genre. The System recommends the same movies to users with similar demographic features. Since each user is different, this approach is considered to be too simple. The basic idea behind this system is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience.
- Content Based Filtering- They suggest similar items based on a particular item. This system uses item metadata, such as genre, director, description, actors, etc. for movies, to make these recommendations. The general idea behind these recommender systems is that if a person liked a particular item, he or she will also like an item that is similar to it.
- Collaborative Filtering- This system matches persons with similar interests and provides recommendations based on this matching. Collaborative filters do not require item metadata like its content-based counterparts.

1.5.1 Agile Methodology

1 Collecting the data sets: Collecting all the required data set from Kaggle web site. In this project we TMDB 5000 Movie Dataset and The Movies Dataset.

2 Data Analysis: make sure that that the collected data sets are correct and analysing the data in the csv files. i.e., checking whether all the column Felds are present in the data sets.

3 Algorithms: in our project we have only two algorithms one is cosine similarity and other is single valued decomposition are used to build the machine learning recommendation model.

4 Training and testing the model: once the implementation of algorithm is completed we have to train the model to get the result. We have tested it several times the model is recommend different set of movies to different users.

5 Improvements in the project: In the later stage we can implement different algorithms and methods for better recommendation.

CHAPTER 2

LITERATURE SURVEY

Over the years, many recommendation systems have been developed using either collaborative, content based or hybrid filtering methods. These systems have been implemented using various big data and machine learning algorithm.

Movie Recommendation System by K-Means Clustering and K-Nearest Neighbour:

A recommendation system collects data about the user's preferences either implicitly or explicitly on different items like movies. An implicit acquisition in the development of movie recommendation system uses the user's behaviour while watching the movies. On the other hand, an explicit acquisition in the development of movie recommendation system uses the user's previous ratings or history. The other supporting techniques that are used in the development of recommendation system is clustering. Clustering is a process to group a set of objects in such a way that objects in the same clusters are more similar to each other than to those in other clusters.

KMeans Clustering along with K-Nearest Neighbour is implemented on the movie lens dataset in order to obtain the best-optimized result. In existing technique, the data is scattered which results in a high number of clusters while in the proposed technique data is gathered and results in a low number of clusters. The process of recommendation of a movie is optimized in the proposed scheme. The proposed recommender system predicts the user's preference of a movie on the basis of different parameters. The recommender system works on the concept that people are having common preference or choice. These users will influence on each other's opinions. This process optimizes the process and has lower RMSE.

Different filtering techniques are used to optimize the predictions made by the system to recommend movies to users. The different types of filtering techniques are as listed below:

- Demographic Filtering- They offer generalized recommendations to every user, based on movie popularity and/or genre. The System recommends the same movies to users with similar demographic features. Since each user is different, this approach is considered to be too simple. The basic idea behind this system is that movies that are

more popular and critically acclaimed will have a higher probability of being liked by the average audience.

- Content Based Filtering- They suggest similar items based on a particular item. This system uses item metadata, such as genre, director, description, actors, etc. for movies, to make these recommendations. The general idea behind these recommender systems is that if a person liked a particular item, he or she will also like an item that is similar to it. The two types of content based filtering used in the proposed system are plot description based recommender and credits, genres and keywords based recommenders.
- Collaborative Filtering- Collaborative filtering systems analyse the user's behaviour and preferences and predict what they would like based on similarity with other users. There are two kinds of collaborative filtering systems: user-based recommender and item-based recommender.

CHAPTER 3

ANALYSIS

3.1 Objective of the project

- Improve retention

  Caters to the user's preferences and keeps them hooked to the application.
- Increase sales

  Can improve business by a great margin by giving various recommendations of different items.
- Form habits

  Influencing usage pattern in users.
- Accelerate work

  Helps the analysts for further research and reduces their work.

3.2 Scope of the project

The objective of this project is to provide accurate movie recommendations to users. The goal of the project is to improve the quality of movie recommendation system, such as accuracy, quality and scalability of system than the pure approaches. This is done using Hybrid approach by combining content based filtering and collaborative filtering, to eradicate the overload of the data, recommendation system is used as information filtering tool in social networking sites hence, there is a huge scope of exploration in this field for improving scalability, accuracy and quality of movie recommendation systems Movie Recommendation system is very powerful and important system. But, due to the problems associated with pure collaborative approach, movie recommendation systems also suffer' s with poor recommendation quality and scalability issues.

3.3 Hardware Requirements

- A PC with Windows/Linux OS
- Processor with 1.7-2.4gHz speed
- Minimum of 8gb RAM
- CPU: Intel Core i3 processor and above

- System type - 64bit OS

3.4 Software Description

- **Python**

  Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

  Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

  Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

  Python's design offers some support for functional programming in the Lisp tradition. It has filter, map and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

- **Dataset**

  A Dataset is the basic data container in PyMVPA. It serves as the primary form of data storage, but also as a common container for results returned by most algorithms. In this tutorial part we will take a look at what a dataset consists of, and how it works.

  Most datasets in PyMVPA are represented as a two-dimensional array, where the first axis is the samples axis, and the second axis represents the features of the samples. In the simplest case, a dataset only contains data that is a matrix of numerical values.

  For building this recommendation engine we have used the following datasets:

  TMDB 5000 Movie Dataset, The Movies Dataset.

The TMDB 5000 Movie dataset contains the following features:

movie_id - A unique identifier for each movie.

cast - The name of lead and supporting actors.

crew - The name of Director, Editor, Composer, Writer etc.

```
In [6]: df1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   movie_id  4803 non-null   int64 
 1   title     4803 non-null   object
 2   cast      4803 non-null   object
 3   crew      4803 non-null   object
dtypes: int64(1), object(3)
memory usage: 150.2+ KB
```

Figure 3.1: TMDB 5000 Movie Dataset

The Movies dataset has the following features:

budget - The budget in which the movie was made.

genre - The genre of the movie, Action, Comedy ,Thriller etc.

homepage - A link to the homepage of the movie.

id - This is infact the movie_id as in the first dataset.

keywords - The keywords or tags related to the movie.

original_language - The language in which the movie was made.

original_title - The title of the movie before translation or adaptation.

overview - A brief description of the movie.

popularity - A numeric quantity specifying the movie popularity.

production_companies - The production house of the movie.

production_countries - The country in which it was produced.

release_date - The date on which it was released.

revenue - The worldwide revenue generated by the movie.

runtime - The running time of the movie in minutes.

status - "Released" or "Rumored".

tagline - Movie's tagline.

title - Title of the movie.

vote_average - average ratings the movie recieved.

vote_count - the count of votes recieved.

```
In [5]: df2.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   budget                4803 non-null   int64
 1   genres                4803 non-null   object
 2   homepage              1712 non-null   object
 3   id                    4803 non-null   int64
 4   keywords              4803 non-null   object
 5   original_language     4803 non-null   object
 6   original_title        4803 non-null   object
 7   overview              4800 non-null   object
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object
 10  production_countries  4803 non-null   object
 11  release_date          4802 non-null   object
 12  revenue               4803 non-null   int64
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object
 15  status                4803 non-null   object
 16  tagline               3959 non-null   object
 17  title                 4803 non-null   object
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64
dtypes: float64(3), int64(4), object(13)
memory usage: 750.6+ KB
```

Figure 3.2: Movie Dataset

3.5 Software Tools

- **Anaconda distribution package (Jupyter Notebook)**

  Anaconda is a free and open-source distribution of the Python programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management system and deployment. Package versions are managed by the package management system conda. The anaconda distribution includes data-science packages suitable for Windows, Linux and MacOS.3

- **Python libraries**

  For the computation and analysis we need certain python libraries which are used to perform analytics. Packages such as sklearn, numpy, pandas, surprise and matplotlib are needed.

i. **Sklearn**

It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

ii. **NumPy**

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

iii. **Pandas**

Pandas is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures and data analysis tools. Unlike NumPy library which provides objects for multi-dimensional arrays, Pandas provides in-memory 2d table object called Data frame.

CHAPTER 4

SYSTEM DESIGN

4.1 System Architecture

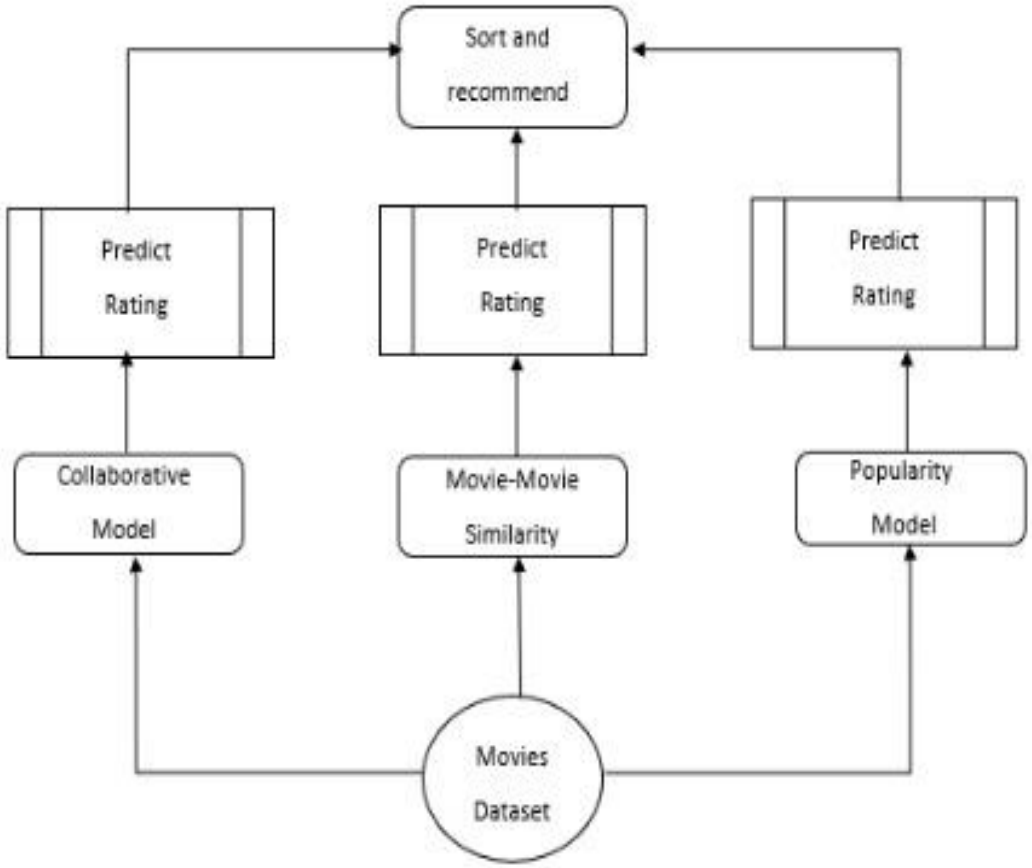


Figure 4.1: System Architecture

4.2 IPython Notebook

The notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results. The Jupyter notebook combines two components:

- A web application: A browser-based tool for interactive authoring of documents which combine explanatory text, mathematics, computations and their rich media output.

- Notebook documents: A representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects.

4.3 Algorithms

1. **Demographic Filtering**
    - we need a metric to score or rate movie
    - Calculate the score for every movie
    - Sort the scores and recommend the best rated movie to the users.

    These demographic recommender provide a general chart of recommended movies to all the users. They are not sensitive to the interests and tastes of a particular user. This is when we move on to a more refined system- Content Based Filtering.

2. **Content Based Filtering**

    In this recommender system the content of the movie (overview, cast, crew, keyword, tagline etc) is used to find its similarity with other movies. Then the movies that are most likely to be similar are recommended.

    - Plot description based recommender

      We will compute pairwise similarity scores for all movies based on their plot descriptions and recommend movies based on that similarity score. The plot description is given in the overview feature of our dataset.

      We see that over 20,000 different words were used to describe the 4800 movies in our dataset.

      With this matrix in hand, we can now compute a similarity score. There are several candidates for this; such as the Euclidean, the Pearson and the cosine similarity scores. There is no right answer to which score is the best. Different scores work well in different scenarios and it is often a good idea to experiment with different metrics.

      We will be using the cosine similarity to calculate a numeric quantity that denotes the similarity between two movies. We use the cosine similarity score since it is independent of magnitude and is relatively easy and fast to calculate. Mathematically, it is defined as follows:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Since we have used the TF-IDF vectorizer, calculating the dot product will directly give us the cosine similarity score. Therefore, we will use sklearn's **linear_kernel()** instead of cosine_similarities() since it is faster.

These are the following steps we'll follow :-

- ❖ Get the index of the movie given its title.
- ❖ Get the list of cosine similarity scores for that particular movie with all movies. Convert it into a list of tuples where the first element is its position and the second is the similarity score.
- ❖ Sort the aforementioned list of tuples based on the similarity scores; that is, the second element.
- ❖ Get the top 10 elements of this list. Ignore the first element as it refers to self (the movie most similar to a particular movie is the movie itself).
- ❖ Return the titles corresponding to the indices of the top elements.

While our system has done a decent job of finding movies with similar plot descriptions, the quality of recommendations is not that great.

- **Credits, Genres and Keywords Based Recommender**

  It goes without saying that the quality of our recommender would be increased with the usage of better metadata. That is exactly what we are going to do in this section. We are going to build a recommender based on the following metadata: the 3 top actors, the director, related genres and the movie plot keywords.

  The next steps are the same as what we did with our plot description based recommender. One important difference is that we use the **CountVectorizer()** instead of TF-IDF. This is because we do not want to down-weight the presence of an actor/director if he or she has acted or directed in relatively more movies. It doesn't make much intuitive sense.

We see that our recommender has been successful in capturing more information due to more metadata and has given us (arguably) better recommendations.

3. Collaborative Filtering

Our content based engine suffers from some severe limitations. It is only capable of suggesting movies which are close to a certain movie. That is, it is not capable of capturing tastes and providing recommendations across genres.

Also, the engine that we built is not really personal in that it doesn't capture the personal tastes and biases of a user. Anyone querying our engine for recommendations based on a movie will receive the same recommendations for that movie, regardless of who she/he is.

Therefore, in this section, we will use a technique called Collaborative Filtering to make recommendations to Movie Watchers. It is basically of two types:-

- **User based filtering**- These systems recommend products to a user that similar users have liked. For measuring the similarity between two users we can either use pearson correlation or cosine similarity. This filtering technique can be illustrated with an example. In the following matrixes, each row represents a user, while the columns correspond to different movies except the last one which records the similarity between that user and the target user. Each cell represents the rating that the user gives to that movie. Assume user E is the target.

	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You	Similarity(i, E)
A	2		2	4	5		NA
B	5		4			1	
C			5		2		
D		1		5		4	
E			4			2	1
F	4	5		1			NA

Figure 4.2 Movies Matrix 1

Since user A and F do not share any movie ratings in common with user E, their similarities with user E are not defined in Pearson Correlation. Therefore, we only need to consider user B, C, and D. Based on Pearson Correlation, we can compute the following similarity.

	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You	Similarity(i, E)
A	2		2	4	5		NA
B	5		4			1	0.87
C			5		2		1
D		1		5		4	-1
E			4			2	1
F	4	5		1			NA

Figure 4.3 Movies Matrix 2

From the above table we can see that user D is very different from user E as the Pearson Correlation between them is negative. He rated Me Before You higher than his rating average, while user E did the opposite. Now, we can start to fill in the blank for the movies that user E has not rated based on other users.

	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You	Similarity(i, E)
A	2		2	4	5		NA
B	5		4			1	0.87
C			5		2		1
D		1		5		4	-1
E	3.51*	3.81*	4	2.42*	2.48*	2	1
F	4	5		1			NA

Figure 4.4 Movies Matrix 3

Although computing user-based CF is very simple, it suffers from several problems. One main issue is that users' preference can change over time. It indicates that precomputing the matrix based on their neighboring users may lead to bad performance. To tackle this problem, we can apply item-based CF.

- **Item Based Collaborative Filtering** - Instead of measuring the similarity between users, the item-based CF recommends items based on their similarity with the items that the target user rated. Likewise, the similarity can be computed with Pearson Correlation or Cosine Similarity. The major difference is that, with item-based collaborative filtering, we fill in the blank

vertically, as oppose to the horizontal manner that user-based CF does. The following table shows how to do so for the movie Me Before You.

	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You
A	2		2	4	5	2.94*
B	5		4			1
C			5		2	2.48*
D		1		5		4
E			4			2
F	4	5		1		1.12*
Similarity	-1	-1	0.86	1	1	

Figure 4.5 Movies Matrix 4

It successfully avoids the problem posed by dynamic user preference as item-based CF is more static. However, several problems remain for this method. First, the main issue is ***scalability***. The computation grows with both the customer and the product. The worst case complexity is O(mn) with m users and n items. In addition, ***sparsity*** is another concern. Take a look at the above table again. Although there is only one user that rated both Matrix and Titanic rated, the similarity between them is 1. In extreme cases, we can have millions of users and the similarity between two fairly different movies could be very high simply because they have similar rank for the only user who ranked them both.

- **Single Value Decomposition**

  One way to handle the scalability and sparsity issue created by CF is to leverage a **latent factor model** to capture the similarity between users and items. Essentially, we want to turn the recommendation problem into an optimization problem. We can view it as how good we are in predicting the rating for items given a user. One common metric is Root Mean Square Error (RMSE). **The lower the RMSE, the better the performance**.

  Latent factor is a broad idea which describes a property or concept that a user or an item have. For instance, for music, latent factor can refer to the genre that the music belongs to. SVD decreases the dimension of the utility matrix by extracting its latent

factors. Essentially, we map each user and each item into a latent space with dimension r. Therefore, it helps us better understand the relationship between users and items as they become directly comparable.

One startling feature of this recommender system is that it doesn't care what the movie is (or what it contains). 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.

CHAPTER 5

IMPLEMENTATION

5.1 Code Snippets

Loading the TMDB 5000 Movies Datasets and merging both the csv files as in figure 5.1.

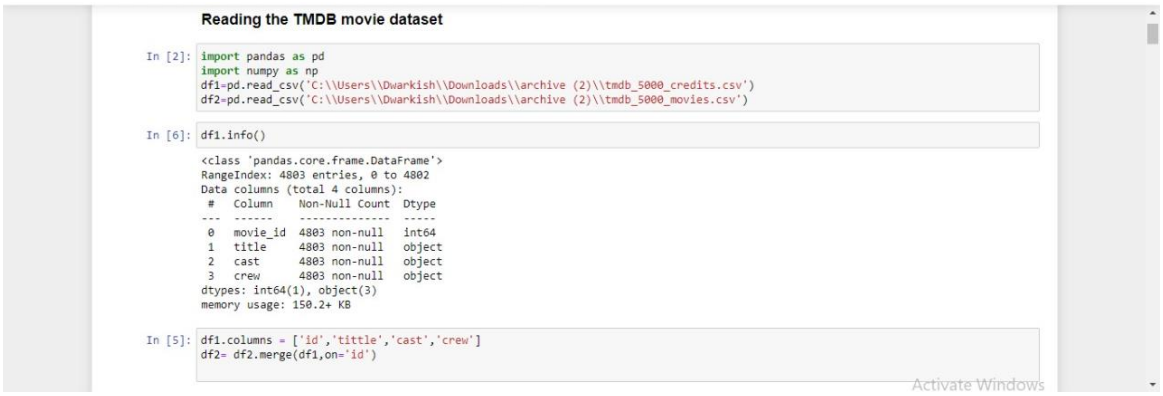
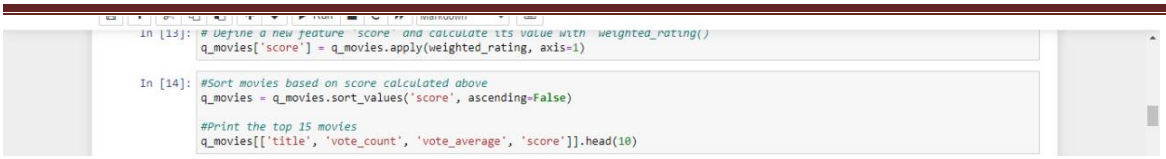


Figure 5.1 Reading the dataset

Calculating the values of C and m where C is the mean and m is the minimum votes so that we can filter out movies that qualify for the chart as shown in figure 5.2 for calculating weighted ratings.

Figure 5.2 Demographic filtering

Sort the DataFrame based on the score feature and output the title, vote count, vote average and weighted rating or score of the top 10 movies as shown in figure 5.3.



```
In [13]: # Define a new feature 'score' and calculate its value with 'weighted_rating()'
         q_movies['score'] = q_movies.apply(weighted_rating, axis=1)

In [14]: #Sort movies based on score calculated above
         q_movies = q_movies.sort_values('score', ascending=False)

         #Print the top 15 movies
         q_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)
```

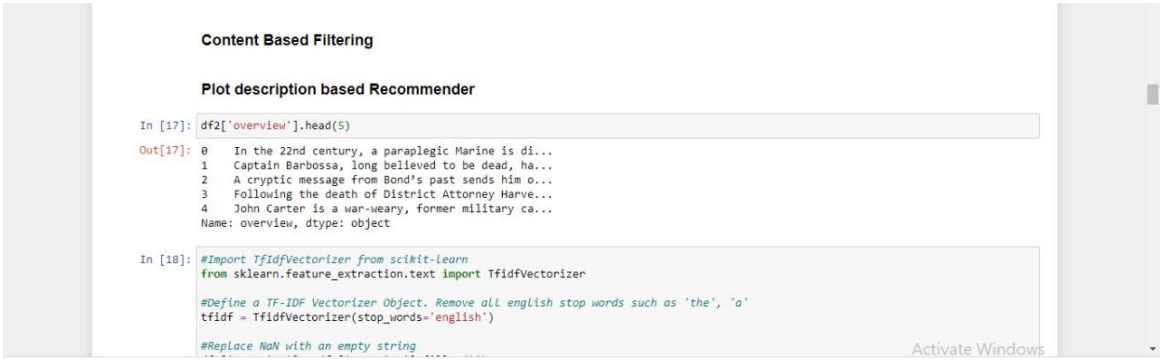
Figure 5.3 Top 10 Movies

```
In [15]: pop= df2.sort_values('popularity', ascending=False)
         import matplotlib.pyplot as plt
         plt.figure(figsize=(12,4))

         plt.barh(pop['title'].head(6),pop['popularity'].head(6), align='center',
                  color='skyblue')
         plt.gca().invert_yaxis()
         plt.xlabel("Popularity")
         plt.title("Popular Movies")
```

Figure 5.4 Plotting graph

Get the index of the movie given its title, get the list of cosine similarity scores for the particular movie with all movies. Convert it into a list of tuples where the first element is its position and the second is the similarity score. Sort the aforementioned list of tuples based on the similarity scores; that is, the second element.

Get the top 10 elements of this list. Ignore the first element as it refers to self (the movie most similar to a particular movie is the movie itself). Return the titles corresponding to the indices of the top elements as shown in figure 5.5, 5.6, 5.7.



Content Based Filtering

Plot description based Recommender

```
In [17]: df2['overview'].head(5)

Out[17]: 0    In the 22nd century, a paraplegic Marine is di...
         1    Captain Barbossa, long believed to be dead, ha...
         2    A cryptic message from Bond's past sends him o...
         3    Following the death of District Attorney Harve...
         4    John Carter is a war-weary, former military ca...
         Name: overview, dtype: object

In [18]: #Import TfIdfVectorizer from scikit-learn
         from sklearn.feature_extraction.text import TfidfVectorizer

         #Define a TF-IDF Vectorizer Object. Remove all english stop words such as 'the', 'a'
         tfidf = TfidfVectorizer(stop_words='english')

         #Replace NaN with an empty string
```

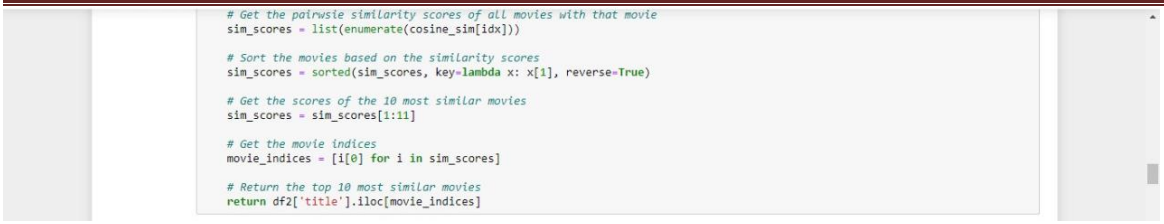
Activate Windows

Figure 5.5 Content Based Filtering (Plot based recommender)

```
         tfidf_matrix = tfidf.fit_transform(df2['overview'])

         #Output the shape of tfidf_matrix
         tfidf_matrix.shape

Out[18]: (4803, 20978)

In [19]: # Import linear_kernel
         from sklearn.metrics.pairwise import linear_kernel

         # Compute the cosine similarity matrix
         cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

In [20]: #Construct a reverse map of indices and movie titles
         indices = pd.Series(df2.index, index=df2['title']).drop_duplicates()

In [21]: # Function that takes in movie title as input and outputs most similar movies
         def get_recommendations(title, cosine_sim=cosine_sim):
             # Get the index of the movie that matches the title
             idx = indices[title]

             # Get the pairwsie similarity scores of all movies with that movie
             sim_scores = list(enumerate(cosine_sim[idx]))
```

Activate Windows

Figure 5.6 Function to get recommendation



```
    # Get the pairwise similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))

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

    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]

    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar movies
    return df2['title'].iloc[movie_indices]
```

Figure 5.7 Function to get recommendation (cont.)

Our data is present in the form of "stringified" lists , we need to convert it into a safe and usable structure. We'll write functions that will help us to extract the required information from each feature as shown in figure 5.8 and 5.9.

Credits, Genres and Keywords Based Recommender

In [24]:
```
# Parse the stringified features into their corresponding python objects
from ast import literal_eval

features = ['cast', 'crew', 'keywords', 'genres']
for feature in features:
    df2[feature] = df2[feature].apply(literal_eval)
```

In [25]:
```
# Get the director's name from the crew feature. If director is not listed, return NaN
def get_director(x):
    for i in x:
        if i['job'] == 'Director':
            return i['name']
    return np.nan
```


Figure 5.8 Content Based Filtering (Credits, Genres and Keywords Based Recommender)

In [26]:
```
# Returns the list top 3 elements or entire list; whichever is more.
def get_list(x):
    if isinstance(x, list):
        names = [i['name'] for i in x]
        #Check if more than 3 elements exist. If yes, return only first three. If no, return entire list.
        if len(names) > 3:
            names = names[:3]
        return names

    #Return empty list in case of missing/malformed data
    return []
```

In [27]:
```
# Define new director, cast, genres and keywords features that are in a suitable form.
df2['director'] = df2['crew'].apply(get_director)

features = ['cast', 'keywords', 'genres']
for feature in features:
    df2[feature] = df2[feature].apply(get_list)
```

In [28]:
```
# Print the new features of the first 3 films
df2[['title', 'cast', 'director', 'keywords', 'genres']].head(3)
```


Figure 5.9 Formatting data

The next step would be to convert the names and keyword instances into lowercase and strip all the spaces between them as shown in figure 5.10.

Out[28]:

	title	cast	director	keywords	genres
0	Avatar	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	James Cameron	[culture clash, future, space war]	[Action, Adventure, Fantasy]
1	Pirates of the Caribbean: At World's End	[Johnny Depp, Orlando Bloom, Keira Knightley]	Gore Verbinski	[ocean, drug abuse, exotic island]	[Adventure, Fantasy, Action]
2	Spectre	[Daniel Craig, Christoph Waltz, Léa Seydoux]	Sam Mendes	[spy, based on novel, secret agent]	[Action, Adventure, Crime]

In [29]:
```
# Function to convert all strings to lower case and strip names of spaces
def clean_data(x):
    if isinstance(x, list):
        return [str.lower(i.replace(" ", "")) for i in x]
    else:
        #Check if director exists. If not, return empty string
        if isinstance(x, str):
            return str.lower(x.replace(" ", ""))
        else:
            return ''
```

In [30]:
```
# Apply clean_data function to your features.
features = ['cast', 'keywords', 'director', 'genres']
```


Figure 5.10 Function to clean data

Metadata soup is a string that contains all the metadata that we want to feed to our vectorizer (namely actors, director and keywords). We use the CountVectorizer() instead of TF-IDF. This is because we do not want to down-weight the presence of an actor/director if he or she has acted or directed in relatively more movies. We can reuse get_recommendations() function by passing in the new cosine_sim2 matrix as second argument.

Figure 5.11 Importing Count vectorizer and Cosine similarity

Since the dataset we used before did not have userId(which is necessary for collaborative filtering) let's load another dataset. We'll be using the 'Surprise' library to implement SVD. When calculating RMSE we get a mean error of 0.89 approx which is more than good enough for our case. Then we train on dataset and arrive at predictions as shown in figure 5.12 and 5.13.

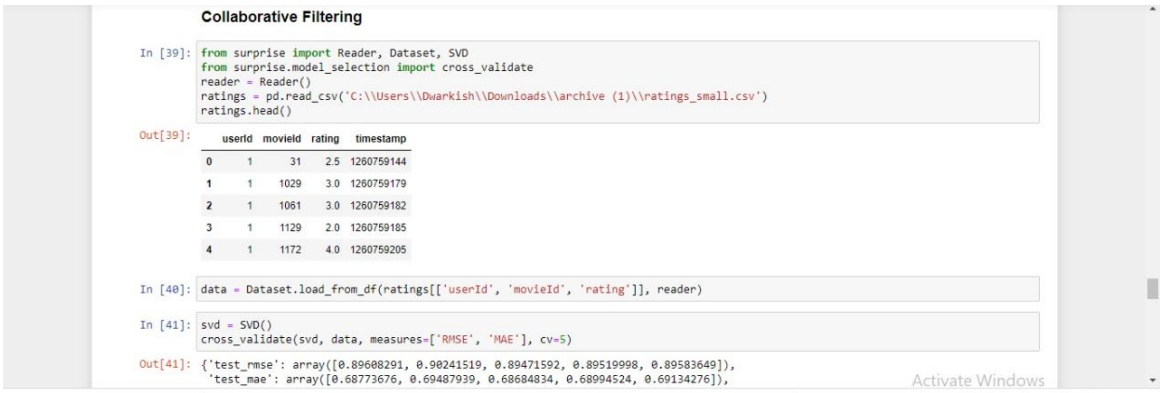


Figure 5.12 Implementing Collaborative Filtering using SVD

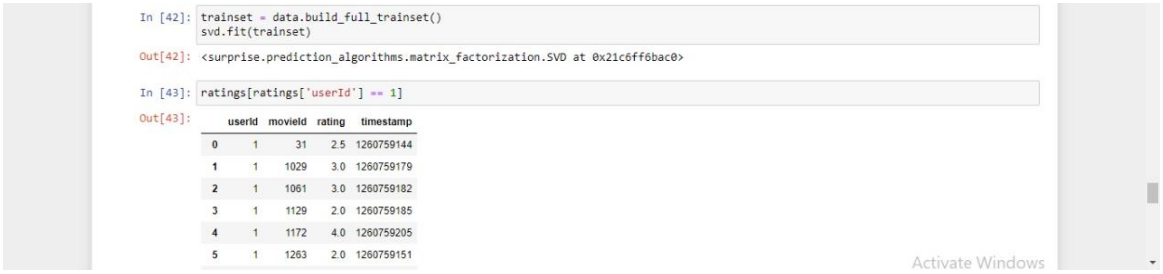


Figure 5.13 Training dataset

CHAPTER 6

TESTING

6.1 Unit Testing

Unit testing is the level of software testing where the individual units/components of a software are tested. The purpose is to validate each unit of the software performs as desired. Here the validation concepts are used to check whether the program is taking the inputs in the correct format or not. Table 6.1 show unit testing for dropping a specified column by using drop function.

Sl No. of test case	1
Name of test	Unit Testing
Item / Feature being tested	DataFrame object
Sample Input	Drop command to drop a column in the table
Expected output	Column gets dropped
Actual output	Specified column gets dropped
Remarks	Test succeeded

Table 6.1 Unit Test Case for dropping a column function

6.2 Integration Testing

Integration testing is also taken as integration and testing this is the major testing process where the units are combined and tested. Its main objective is to verify whether the major parts of the program is working fine or not. This testing can be done by choosing the options in the program and by giving suitable inputs it is tested. Table 6.3 shows integration testing.

Sl No. of test case	1
Name of test	Integration testing
Item / Feature being tested	Train-test split functionality
Sample Input	Input features and no of splits and check for the correct splitting of the dataframe.
Expected output	Split the dataframe according to given no of splits.
Actual output	Split the dataframe according to given no of splits.
Remarks	Test succeeded

Table 6.2 Integration test case for splitting the data using KFold

6.3 System Testing

System Testing (ST) is a black box testing technique performed to evaluate the complete system the system's compliance against specified requirements. The software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. Tables 6.3, 6.4and 6.5 show system test cases for predictions made by demographic filtering, content based filtering and collaborative filtering.

Sl No. of test case:	1
Name of test:	System testing
Item / Feature being tested:	Model's prediction on Movie
Sample Input:	Merged Dataset
Expected output:	Lists of top 10 movies
Actual output:	Lists of top 10 movies
Remarks:	Test succeeded

Table 6.3 System test case for model's prediction of demographic filtering

Sl No. of test case:	2
Name of test:	System testing
Item / Feature being tested:	Model's prediction on Movie
Sample Input:	Movie=Toy Story
Expected output:	Lists of all movies similar to toy story
Actual output:	Lists of all movies similar to toy story
Remarks:	Test succeeded

Table 6.4 System test case for model's prediction of content based filtering

Sl No. of test case	3
Name of test	System testing
Item / Feature being tested	Model's prediction on user rating
Sample Input	User id=1 Movie id=302
Expected output	Rating given by user based on past history(3)
Actual output	Rating given by user based on past history(2.75)
Remarks	Test succeeded

Table 6.5 System test case for model's prediction of collaborative filtering

CHAPTER 7

RESULTS

7.1 Result Snapshots

```
In [14]: #Sort movies based on score calculated above
         q_movies = q_movies.sort_values('score', ascending=False)

         #Print the top 15 movies
         q_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)
```

Out[14]:

	title	vote_count	vote_average	score
1881	The Shawshank Redemption	8205	8.5	8.059258
662	Fight Club	9413	8.3	7.939256
65	The Dark Knight	12002	8.2	7.920020
3232	Pulp Fiction	8428	8.3	7.904645
96	Inception	13752	8.1	7.863239
3337	The Godfather	5893	8.4	7.851236
95	Interstellar	10867	8.1	7.809479
809	Forrest Gump	7927	8.2	7.803188
329	The Lord of the Rings: The Return of the King	8064	8.1	7.727243
1990	The Empire Strikes Back	5879	8.2	7.697884

Figure 7.1 Recommendation by Demographic Filtering

```
In [15]: pop= df2.sort_values('popularity', ascending=False)
         import matplotlib.pyplot as plt
         plt.figure(figsize=(12,4))

         plt.barh(pop['title'].head(6),pop['popularity'].head(6), align='center',
                  color='skyblue')
         plt.gca().invert_yaxis()
         plt.xlabel("Popularity")
         plt.title("Popular Movies")
```

Out[15]: Text(0.5, 1.0, 'Popular Movies')

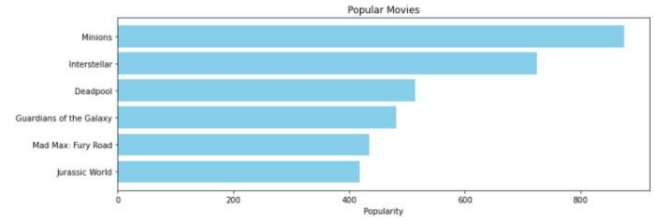


Figure 7.2 Graphical representation of Demographic Filtering

```
In [22]: get_recommendations('The Dark Knight Rises')
```

Out[22]:
```
65                          The Dark Knight
299                          Batman Forever
428                          Batman Returns
1359                                Batman
3854    Batman: The Dark Knight Returns, Part 2
119                          Batman Begins
2507                            Slow Burn
9           Batman v Superman: Dawn of Justice
1181                                 JFK
210                          Batman & Robin
Name: title, dtype: object
```

```
In [23]: get_recommendations('The Avengers')
```

Out[23]:
```
7               Avengers: Age of Ultron
3144                            Plastic
1715                            Timecop
4124                 This Thing of Ours
3311              Thank You for Smoking
3033                        The Corruptor
588     Wall Street: Money Never Sleeps
2136          Team America: World Police
1468                       The Fountain
1286                        Snowpiercer
Name: title, dtype: object
```


Figure 7.3 Recommendation by Content Based Filtering (Plot based recommender)

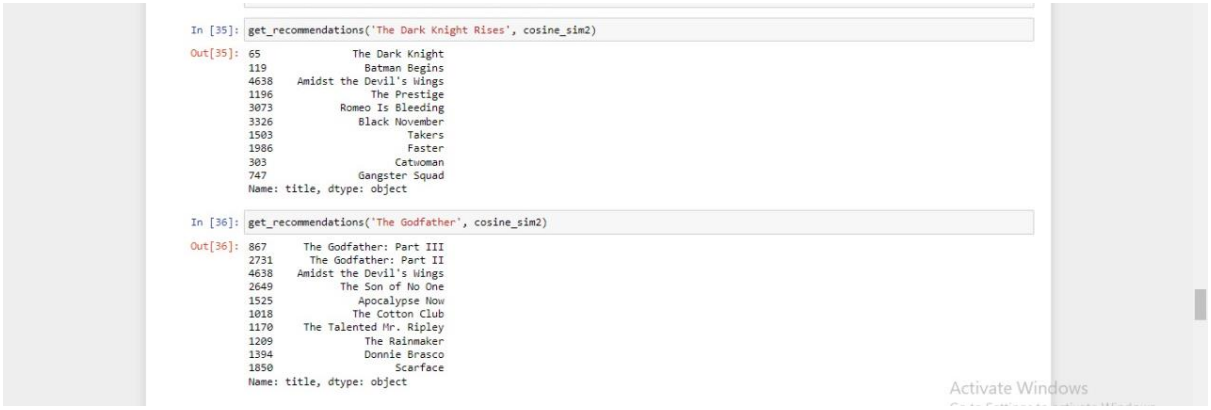


Figure 7.4 Recommendation by Content Based Filtering (Credits, Genres and Keywords Based Recommender)

```
In [44]: svd.predict(1, 302, 3)
Out[44]: Prediction(uid=1, iid=302, r_ui=3, est=2.7375526346714354, details={'was_impossible': False})
```

Figure 7.5 Prediction using SVD for Collaborative Filtering

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

8.1 Conclusion

We create recommenders using demographic, content- based and collaborative filtering. While demographic filtering is very elementary and cannot be used practically, Hybrid Systems can take advantage of content-based and collaborative filtering as the two approaches are proved to be almost complimentary.

In this project, to improve the accuracy, quality and scalability of movie recommendation system, a Hybrid approach by unifying content based filtering and collaborative filtering; using Singular Value Decomposition (SVD) as a classifier and Cosine Similarity is presented in the proposed methodology. Existing pure approaches and proposed hybrid approach is implemented on two different Movie datasets and the results are compared among them. Comparative results depicts that the proposed approach shows an improvement in the accuracy, quality and scalability of the movie recommendation system than the pure approaches. Also, computing time of the proposed approach is lesser than the other two pure approaches.

8.2 Future Enhancements

A possible extension of the hybrid model is creating a feature vector for model and training a neural network to learn the importance or the weights of each model. Such a linear combination model can highly improve the recommendations. Learning to rank using deep learning methods is another possible extension. Currently, we have implemented the models using the smaller dataset. Extending this to the larger dataset and observing the results is yet another future task. There is a need to work on the memory requirements of the proposed approach in the future. The proposed approach has been implemented here on two basic movie datasets. It can also be implemented on the Film Affinity datasets and the performance can be computed in the future. There is also a scope for the development of a user-friendly interface which makes use of the proposed recommendation engine.

REFERENCES

[1] https://www.ijert.org/research/recommender-systems-types-of-filtering-techniques-IJERTV3IS110197.pdf

[2] https://en.wikipedia.org/wiki/Recommender_system

[3] https://www.analyticssteps.com/blogs/what-are-recommendation-systems-machine-learning

[4] Kaggle

[5] 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS) Date of Conference: 23-25 Nov. 2018

[6] A Collaborative Filtering based Recommender System for Suggesting New Trends in Any Domain of Research Published in IEEE 2019

[7] Movie Recommendation System Using Item Based Collaborative Filtering International Journal of Innovative Research in Computer Science & Technology (IJIRCST), ISSN: 2347-5552, Volume-8, Issue-4, July 2020

[8] G. Vaitheeswaran , L. Arockiam "Hybrid Based Approach to Enhance the Accuracy of Sentiment Analysis on Tweets " IJCSET ,Vol 6, Issue 6, June , 2016.

[9] Chen, H. W., Wu, Y. L., Hor, M. K., & Tang, C. Y. (2017, July). Fully content-based movie recommender system with feature extraction using neural network. 2017 International conference on machine learning and cybernetics (ICMLC) (Vol. 2, pp. 504-509). IEEE.

[10] Çano, E., & Morisio, M. (2017). Hybrid recommender systems: a systematic literature review. Intelligent data analysis, 21(6), 1487-1524.

[11] Breese, J. S., Heckerman, D. and Kadie, C, „Empirical analysis of predictive algorithms for collaborative filtering". Computer Networks and ISDN Systems, 1998, pp. 43-52.

[12] Alspector, J., Koicz, A., and Karunanithi, N, „Feature-based andClique-based User Models for Movie Selection: A Comparative Study".User Modeling and User-Adapted Interaction 7,1997,pp. 279-304.