**‘Movie Recommender System’**

**OPIM 5512 –**

**Data Science using Python**

**Team 7**

Rahul Choubisa

Swathi Sukumar

Deepa Pasunooru

Amulya Balmuri

Vamsi Krishna Malempati

Satish Kumar MNV

**TABLES OF CONTENTS**

[**Introduction**](about:blank) **[3](#_heading=h.bbhj0ckyc7y)**

[Applications of Recommender systems](about:blank) [3](#_heading=h.41kc2h456w2z)

[Benefits of Recommendation Systems](about:blank) [4](#_heading=h.2rcchwlh6dcu)

[How do recommenders work](about:blank) [4](#_heading=h.iun785d21v1z)

[**Literature**](about:blank) **Survey and What’s New 4**

**Exploratory Data Analysis 5**

[Data & Findings](about:blank)5

[**Implementation**](about:blank) **7**

[Content Based Implementation](about:blank) 7

[Content Based Filtering Model results](about:blank) 8

[Collaborative Filtering Model](about:blank) 9

**Recommendation Engine UI** **10**

[**Re**](about:blank)**sults 11**

[**Conclusion**](about:blank) **and Future work 11**

[**References**](about:blank) **12**

[**Appendix**](about:blank) **13**

### Introduction

A movie recommendation system, also known as a movie recommender system, is a machine learning-based method of filtering or predicting users' preferences for films based on their prior decisions and actions. It is a cutting-edge filtration system that anticipates user preferences and potential movie selections for a certain domain-specific item, in this case movies.

What are recommendation Systems?

In the incredibly busy environment we live in today, recommendation systems are becoming more and more crucial. With the numerous duties that need to be completed in the finite 24 hours, people are always pressed for time. Recommendation systems are crucial because they enable people to make the best decisions without having to exert their cognitive abilities.

Basically, the aim of a recommendation system is to find content that a particular user could find interesting. To further personalize lists of relevant and entertaining content for each user or individual, a variety of characteristics must be considered. The algorithms behind recommendation systems, which are based on artificial intelligence, sift through all available possibilities to compile a unique list of options that are interesting and pertinent to a particular user. These results are based on the user's profile, search and browsing history, what other people with similar characteristics and demographics are watching, and how likely you are to watch those movies. With the use of heuristics and the facts at hand, this is accomplished.

#### Applications of Recommender systems:

– Advertising Messages

– Movies

– Books

– Music Tracks

– News Articles

– Restaurants

– Future Friends (Social Network Sites)

– Courses in e-learning

– Jobs

– Research Papers

– Investment Choices

– TV Programs

– Citations

– Clothes

– Online Mates (Dating Services)

– Supermarket Goods

**Real world Examples**

**Youtube:**

Whether users are aware of it or not, recommender systems are among the most prevalent applications of machine learning that they will come across. It drives YouTube "recommended videos" and Facebook and Twitter "curated timelines."

**Amazon Prime:**

Amazon started building an algorithm that would be able to examine things posted by users and ascertain the buying preferences of specific customers in tandem with the advancement of AI technology.

The Amazon algorithm is a recommendation engine made up of numerous crucial components in charge of analyzing various types of data. Artificial intelligence and machine learning-based technology have made this possible.

**Netflix:**

The company's technology is its main strength. especially their system of recommendations. A subfield of the study of information filtering systems is the recommendation system (Recommender system, 2020). Prior to reaching a person, information filtering systems deal with deleting redundant information from the data stream.

### Literature

Movie recommendation systems are everywhere. Be it Netflix, Amazon Prime, Youtube, Hulu, all of them have their own unique recommender system. For our project, we have referred to the type of algorithms and techniques followed in the Introduction to Recommender systems[1]. We also have referred to the TF-IDF and similarity score on Chan’s Jupyter[2] for calculating the metrics while modeling the recommender systems. These two sources have been useful for us to gather the insights followed for building the recommender model using Content and Collaborative Filtering.

**What’s New in our Project?**

We have created a dynamic Google Form that consists of the polls like what is the favorite genre and rating each movie out of 5. We have collected the real-time data from our class and have used that data to compare the results we obtained from our recommender model. See Appendix [1] and Appendix [2].

**EDA**

Movie recommendation algorithm. For the sources, see Appendix [3] and Appendix [4], respectively. Two separate files are created for each dataset. Dataset 1 includes 20,000,263 anonymous ratings for about 27,278 films submitted by 138,493 people, in contrast. By adding new columns, deleting unnecessary columns, turning column values into lists, and filtering the rows grouped by criteria required for the study, the datasets have been cleaned. In the data dictionary, the columns' specifics and a description are provided. For the data dictionaries of datasets 1 and 2, respectively, see Appendix [5] and [6]. The common variable imdbId has been joined with the two datasets.

To illustrate the data, various graphs have been generated depending on the factors available in the dataset. We can see from the Figure in Appendix [7] that the three genres with the most popular films are drama, comedy, and action, which suggests that the movie recommender system should concentrate more on these genres. Figure 1 Appendix [8] depicts the top users who have given the majority of the films high ratings. The business can create new inputs to the database by recognizing and elevating top contributors. Figures in Appendix [9] and [10] provide examples of films with high and low average ratings, respectively. These findings can aid the system in giving movies a higher priority and promoting them more aggressively based on the average rating value.Finding the user's interest in a certain genre is crucial, and the statistics in Appendices [11] and [12] describe the proportions of each genre with the highest and lowest ratings, respectively. When compared to other genres, Horror and Children's movies appear to garner less attention, with the biggest number of films receiving lower ratings. It is clear that the majority of Film-Noir and War films have achieved success with the highest ratings. A specific 20-year period, from 1921 to 2020, is highlighted in the Appendix [13] figure that shows the frequency of movie ratings. With a count for 3.5 to 4 rating being the highest and followed by 2.5 to 3 and below, we can see the similar pattern for all the time periods.

Ratings of 4.5 to 5. Figure [14] of the Appendix shows a similar pattern in terms of genre ratings, however drama appears to have received more 4.5–5 ratings than other genres.

Dataset 2 has a rating value based on the voting average on a scale of 10 for EDA. The numbers

from Appendix [15] and [16] show the movies with the highest vote average and vote count. Vote average and vote count for movies can be a crucial factor in determining how popular a film is. The frequency of movie votes by genre is shown in Figure in Appendix [17], and we can see that most films have a vote average between 6.1 and 8, followed by 4.1 to 6. A word cloud has been produced based on the title, and the figure in Appendix [18] displays the most frequent words. These words can be employed in the recommender system in delivering familiar results

| **Column Name** | **Description** |
| --- | --- |
| movieId | ID of the Movie |
| title | Title of the Movie |
| genres | Genre of the Movie |
| userId | ID of the User who rated the Movies |
| rating | Rating given by the User for a Movie |
| year | Year of the Movie being released |

# 

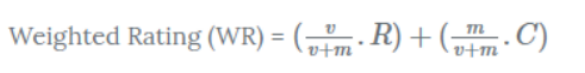
# 

# 

# 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. Appendix [11] figure that shows the basic idea of how a typical user watches the content.

We require a metric to evaluate or rank movies. Create a score for each film. Sort the ratings and give the users the top-rated movie. We could use the movie's average ratings as the score, but it wouldn't be fair enough because a movie with an average rating of 8.9 and only 3 votes couldn't be deemed superior to a movie with an average rating of 7.8 but 40 votes. We will so use the weighted rating (wr) provided by IMDB, which is as follows:

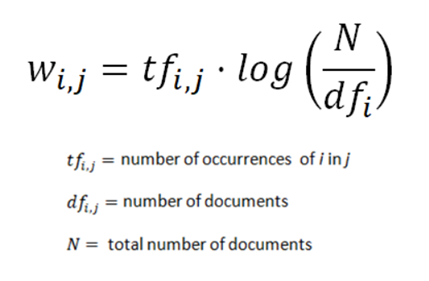


v is the number of votes for the movie; m is the minimum votes required to be listed in the chart; R is the average rating of the movie; And C is the mean vote across the whole report.

All movies will have pairwise similarity scores depending on their genre computed, and those values will be used to make movie recommendations. The overview component of our dataset includes a description of the plot. Let's examine the data first.

Everyone among us who has even a passing familiarity with text processing is aware that each overview's word vector needs to be converted. For each overview, we will now compute the Term Frequency-Inverse Document Frequency (TF-IDF) vectors.

If we were wondering what term frequency is, it's the proportion of times a word appears in a document over all instances, which is expressed as (term instances/total instances). The relative count of documents that contain a term is known as the "inverse document frequency," and it is expressed as log(number of documents/documents with term). Each word's total significance to the documents in which it appears is equal to TF. \* IDF

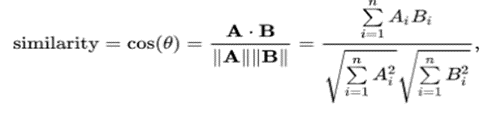


This will provide you with a matrix where each row, as before, represents a movie and each column, as before, represents a word from the overview vocabulary (all the words that appear in at least one document). This is done in order to lessen the weight given to words that frequently appear in plot summaries and, as a result, their significance in determining the final similarity score.

We see that over 19 different words were used to describe the 2500 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:



We're going to create a function that accepts a movie title as an input and returns a list of the ten films that are most comparable to it. First, we require a reverse mapping between DataFrame indices and movie titles. To put it another way, we require a method of determining the index of a movie given its title in our metadata DataFrame..

We can now define our recommendation function because we are in a good position. We'll proceed in the following manner: -

1. Obtain the movie's index using the title as a guide.
2. Get a list of the cosine similarity scores between that movie and all other movies. Make it into a list of tuples with the position as the first member and the similarity score as the second.
3. The second element is to order the aforementioned list of tuples according to the similarity scores.
4. Obtain the top ten items on this list. Ignore the first piece since it is about you (the movie most similar to a particular movie is the movie itself).
5. Send back the titles for the top items' matching indices.

# Collaborative Filtering

In this project we focussed on two types of filtering techniques to build our recommendation systems. Collaborative filtering and content based filtering were the two types.

In collaborative filtering, we build our recommendation systems by finding similar patterns or information of the users. It uses other users’ preferences and taste to recommend new items to a user. Find similar users (or items) to recommend new items which where liked by those users,

This filtering technique can be used to recommend by taking the rating of a particular user based on user ratings for other movies and others’ ratings for all movies. This concept is widely used in recommending

movies, news, applications, and so many other items.

In our example , we followed the following steps . 1. Merge the datasets : Comprehensive list of userid,movie id and ratings were formed by merging the user id, movie id, ratings by combining different csv datasets.

2.Pivot the dataset: To form a NxM table with ratings as its values and user-id and movie-id as its columns and index, a pivot table was created this way.

3.Convert pivot table into a vector matrix: Using the csr\_matrix library, the sparse matrix function enables arithmetic operations by converting the pivot table into matrices.

4.Find the K-Nearest Neighbors: For a randomly chosen movie, we make use of the sparse matrix values and compute the K Nearest Neighbors .

5.Use cosine similarity: The distance measure within KNN model is cosine similarity. Cosine similarity measures the angle between two vectors in order to determine how similar these values are.

This way we were able to identify the top K movies measured by cosine similarity distance for a given chosen movie . The assumption here is that the given chosen movie is the most liked one for the user .

An increase in distance in cosine similarity implies smaller angles between vectors which means it is more similar to each other

**Recommendation Engine UI**

We create a recommendation engine UI using pycharm software. The typical steps after installing the software to create the code were -

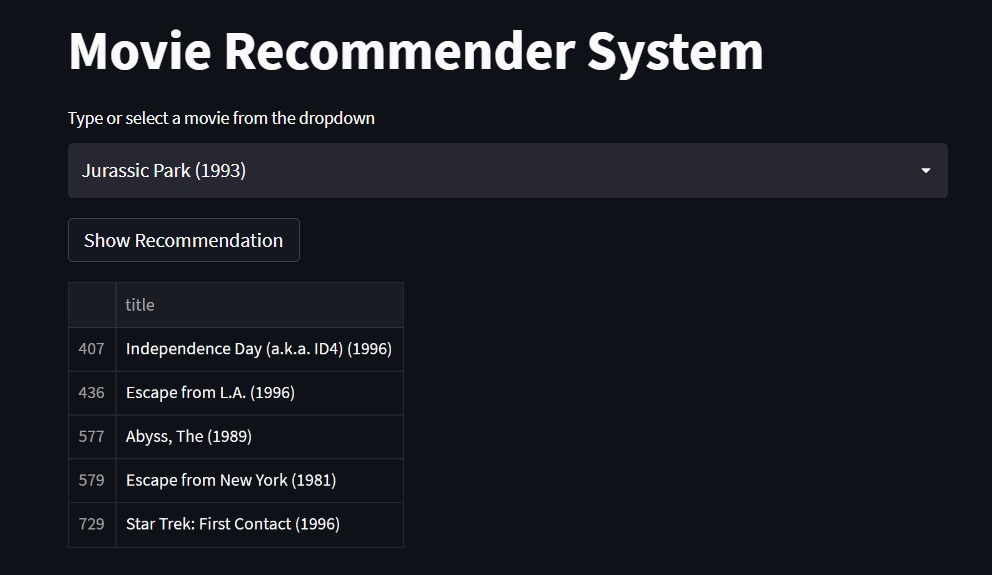
1. Import pickle library movies and cosine similarity data into files on the local system using pickle.dump method.
2. Open this dump file into the pycharm software just by right click and paste them and Load them using pickle.load method.
3. Import the required libraries (pickle,streamlite and pandas)
4. Create an object to save all the movie titles.
5. Create the function almost the same as python recommendation system function and take the input as movie title and return the recommended movies.
6. Create an if statement for the button and run the function inside it.
7. Run it using streamlit run file\_name.py which will open a webpage in local.

Now we can see the UI as given in Appendix[]. We can use it to recommande movies just by selecting an already watched movie and clicking on recommendation. That will provide all the recommended movies for that particular movie which the user will like.

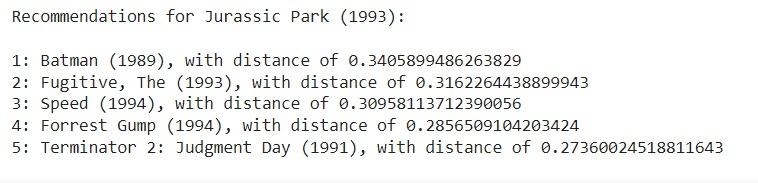
**Results:**

To evaluate the top recommendations produced by our recommendation system, we compared the recommendations for the movie “Jurassic Park(1993)” by using both content based and collaborative filtering.

The recommended movies by using content based filtering is as follows:



The recommended movies by using collaborative filtering is as follows:



**Conclusion and Future Work:**

New possibilities for finding personalized information on the Internet are made possible by recommender systems. It also enables users to access goods and services that are not immediately available to users on the system, which helps to relieve the issue of information overload, which is a fairly typical situation with information retrieval systems. We develop a method that prioritizes addressing the user's individual preferences, and users are given movies based on his prior reviews. This tactic aids in enhancing recommendation accuracy.

In order to provide an integrated solution to the recommendation problem, this project provides a thorough review of various methods for the implementation of movie recommendation systems. On the one hand, it suggests the Collaborative filtering strategy, which uses the preferences of other users and suggests a movie. On the other hand, it suggests a Content-based technique based on clustering films according to their plots using the Tf/Idf weighting scheme, which provides a solution to the problem of movie recommendation in the absence of user preference data.

The future work includes the creation of a hybrid model fusing the Collaborative and clustering techniques. Despite not being the most precise method, the user clustering strategy offers a sufficient means of identifying commonalities amongst users in terms of their preferences for various movie genres. It might serve as the foundation for a probabilistic method, in which the probability of each prediction would only be calculated for users who belonged to the same cluster. In addition, it is thought appropriate to test dimensionality reduction strategies other than the traditional method.

**References:**

[1]. [Introduction To Recommender Systems- 1: Content-Based Filtering And Collaborative Filtering | by Abhijit Roy | Towards Data Science](https://towardsdatascience.com/introduction-to-recommender-systems-1-971bd274f421)

[2]. [TF-IDF and similarity scores | Chan`s Jupyter (goodboychan.github.io)](https://goodboychan.github.io/python/datacamp/natural_language_processing/2020/07/17/04-TF-IDF-and-similarity-scores.html#Comparing-linear_kernel-and-cosine_similarity)

**Appendix:** 