**DSC 478: FINAL PROJECT**

**Movie Recommendation System**

**By**

**Sai Santoshi Pravallika Devaguptapu**

**Introduction:**

Increasing of data day by day has created different opportunities to create new system and make life easier. one system which is developed form the data is recommendation system. A recommender system, or a recommendation system, is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. They are primarily used in commercial applications. The recommendation systems are playing a major role in any of the online platforms. Netflix, Amazon, Spotify etc. have keep on suggesting with different recommendations. Every online application tells us the “the customer who bought this item also bought this”.

In this project I have taken data from different datasets related to movies and user ratings from Kaggle. Developed recommendation systems based on derived IMDB ratings, using user ratings and using similarity based on genres and overview of movie using technics like Content based filtering and Collaborative Filtering.

“Netflix awarded a $1 million prize to a developer team in 2009, for an algorithm that increased the accuracy of the company’s recommendation system by 10 percent.”

Data:

Data has been collected from Kaggle platform.

Credits data: <https://www.kaggle.com/tmdb/tmdb-movie-metadata>

Contains movie ID, Caste, crew, title as columns. Shape of the data is 4803 rows and 4 columns.

Graphical user interface

Description automatically generated

Movie data: <https://www.kaggle.com/tmdb/tmdb-movie-metadata>

Contains 20 columns with and 4083 rows.

Graphical user interface, application

Description automatically generated

User ratings: <https://www.kaggle.com/rounakbanik/the-movies-dataset>

Contains 100004 rows and 4 columns

Table

Description automatically generated

Preprocessing and Transformations:

* Checking for missing values

In our data set homage and tagline columns have more missing values. Anyways these two columns are not the input for model so we can ignore these columns. Title columns has one missing value, so I dropped the entire row.

A picture containing chart

Description automatically generated

* Conversation of json data into list data

Genre column has data in the form of json. So, I converted the json data to list and then split the list and check number of movies in each genre.

A picture containing text

Description automatically generated

keywords column has data in the form of json. So, I converted the json data to list and then split the list and to check the most assigned keyword.

Text

Description automatically generated

* Data integrations

Merged the data sets on Movie\_id

Graphical user interface, text, application, email

Description automatically generated

Chart, histogram

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, histogram

Description automatically generated

A picture containing table

Description automatically generated

Chart, histogram

Description automatically generated

Chart

Description automatically generated

Chart, bar chart

Description automatically generated

### IMDB weighted Average:

We can use the average ratings of the movie as the score but using this won't be fair enough since a movie with 8.9 average rating and only 3 votes cannot be considered better than the movie with 7.8 as as average rating but 40 votes.

The metric is the numeric quantity based on which you rank movies. A movie is considered to be better than another movie if it has a higher metric score than the other movie. It is very important that you have a robust and reliable metric to build your chart upon to ensure a good quality of recommendations.

A picture containing shape

Description automatically generated

The following apply:

* v is the number of votes garnered by the movie.
* m is the minimum number of votes required for the movie to be in the chart (the prerequisite)
* R is the mean rating of the movie
* C is the mean rating of all the movies in the dataset
* You already have the values for v and R for every movie in the form of the vote\_count and vote\_average features respectively. Calculating C is extremely trivial.

A picture containing text

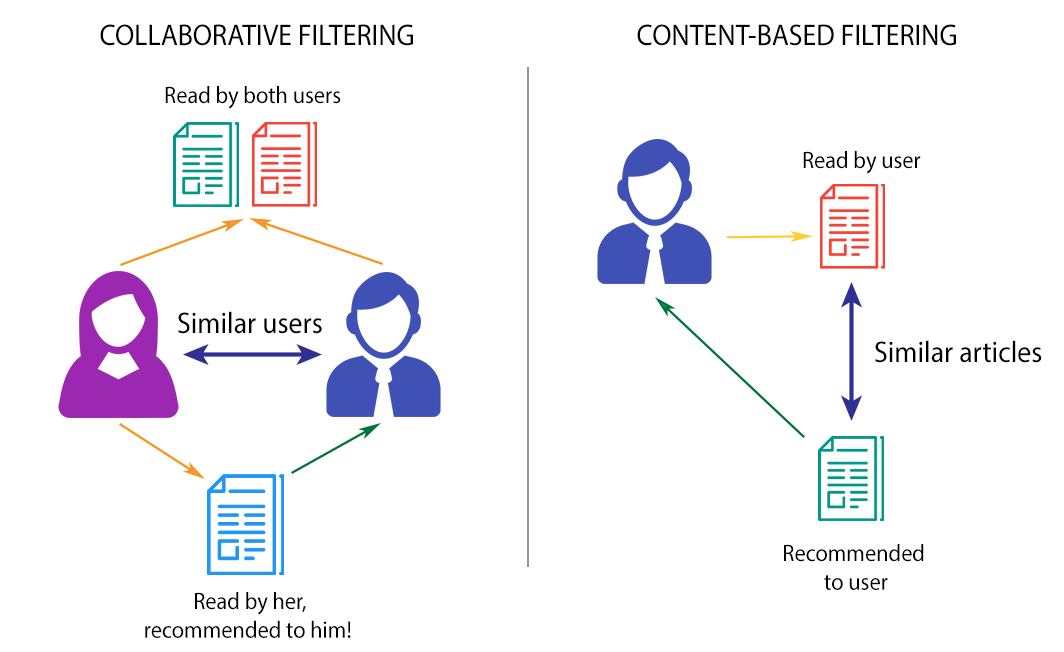
Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Content based filtering:

This filtering is based on the description or some data provided for that product. The system finds the similarity between products based on its context or description. The user’s previous history is taken into account to find similar products the user may like.



we willbebuilding recommender based on the movie genres and overiew. A fairly common approach for this problem is to use a tf-idf vectorizer. While this approach is more commonly used on a text corpus, it possesses some interesting properties that will be useful in order to obtain a vector representation of the data. The expression is defined as follows:

Text

Description automatically generated

Here we have the product of the term frequency, i.e. the amount of times a given term (genre) occurs in a document (genres of a movie), times the right side term, which basically scales the term frequency depending on the amount of times a given term appears in all documents (movies). The fewer movies that contain a given genre (df\_i), the higher the resulting weight. The logarithm is basically there to smooth the result of the division, i.e. avoids huge differences as a result of the right hand term.

Text

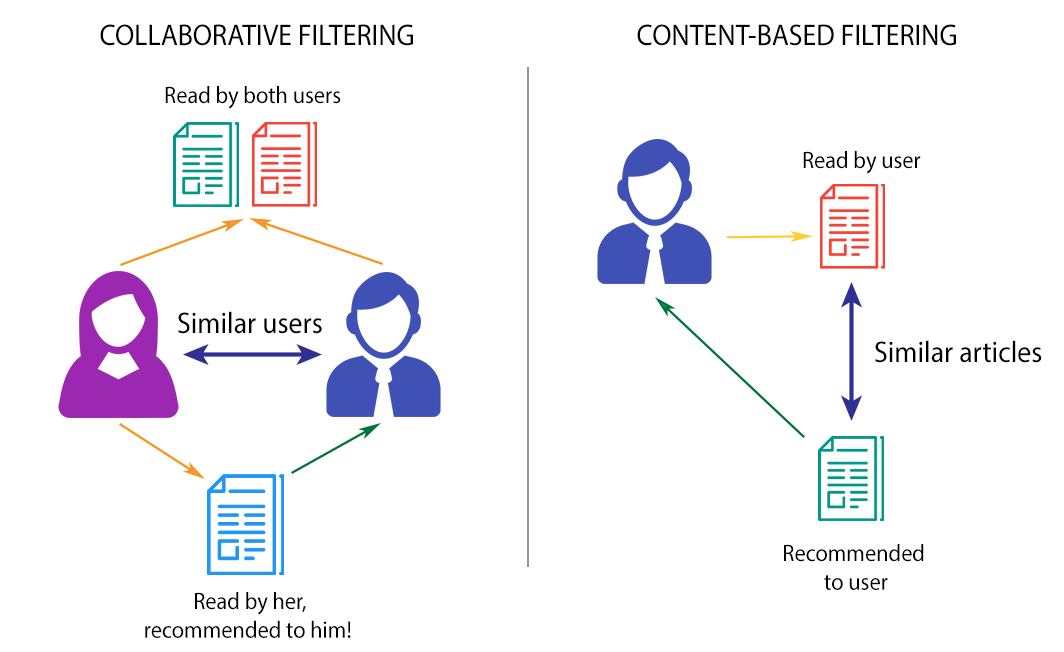
Description automatically generated

Graphical user interface, text, application

Description automatically generated

# **Collaborative filtering**

The recommendations are done based on the user’s behavior. History of the user plays an important role. For example, if the user ‘A’ likes ‘Coldplay’, ‘The Linkin Park’ and ‘Britney Spears’ while the user ‘B’ likes ‘Coldplay’, ‘The Linkin Park’ and ‘Taylor Swift’ then they have similar interests. So, there is a huge probability that the user ‘A’ would like ‘Taylor Swift’ and the user ‘B’ would like ‘Britney Spears’. This is the way collaborative filtering is done.



# KNN Search

The kNN algorithm measures distance to determine the “closeness” of instances. It then classifies an instance by finding its nearest neighbors and picks the most popular class among the neighbors.

We convert our table to a 2D matrix, and fill the missing values with zeros (since we will calculate distances between rating vectors). We then transform the values(ratings) of the matrix dataframe into a scipy sparse matrix for more efficient calculations.

We use unsupervised algorithms with sklearn.neighbors. The algorithm we use to compute the nearest neighbors is “brute”, and we specify “metric=cosine” so that the algorithm will calculate the cosine similarity between rating vectors. Finally, we fit the model.

Table

Description automatically generated

Text

Description automatically generated

# SVD

SVD in the context of recommendation systems is used as a collaborative filtering (CF) algorithm. Most CF algorithms are based on user-item rating matrix where each row represents a user, each column an item. The entries of this matrix are ratings given by users to items.

Graphical user interface, text, application, email

Description automatically generated

Results:

Below are the results for different recommendation systems:

|  |  |  |  |
| --- | --- | --- | --- |
| IMDB | Content based | KNN | SVD |
| Fight Club  The Dark Knight  Inception  The Godfather  Interstellar | Batman Forever    The Dark Knight  Batman    Sicko    Batman Returns | Jarhead  Interview with the Vampire  Flash dance  Lock, Stock and Two Smoking Barrels  The Thomas crown Affair | Men in Black II  Batman Begins  Sunshine  Crank  Pandora's Box |

Conclusion:

I have used different algorithms for recommend the systems. After getting the results from different recommenders (KNN, Content based, SVD) It will display all the recommendations in order have sorted the all the outputs based on occurrence and recommended the top movies.

Text, letter

Description automatically generated

Future work:

Including fuzzy logic and NLP gives better recommendation. So, as future work I would like to implement the NLP to get better recommendation system.