GraphicEra
Deemed to be University
Accredited by NAAC with Grade A
NBA Accredited Program in CSE, ECE & ME
Approved by AICTE, Ministry of HRD, Govt. of India

MINI PROJECT REPORT

PROBLEM STATEMENT:

RECOMMENDATION SYSTEM

Submitted By: Gaurisha Maheshwari

Branch: CSE(Core)

Project No: 17

University Roll No: 2013325

Section: C

Class Roll No: 18

MOTIVATION

We now live in what some call the “era of abundance”. For any given product, there are sometimes thousands of options to choose from. Think of the examples above: streaming videos, social networking, online shopping; the list goes on. Recommender systems help to personalize a platform and help the user find something they like.

With the ever-growing volume, complexity and availability of online information, recommender systems have been an effective key solution to overcome such information overload.

METHODOLOGY

The rapid growth of data collection 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.

Simply a **Recommendation System** is a filtration program whose prime goal is to predict the “ratings” or “preferences” of a user towards a domain-specific item or item. In our case, this domain-specific item is **movie,** therefore the main four of our recommendation systems is to filter and predict only those movies which a user would prefer given some data about the user him or herself.

Different Filtering Strategies

- **Content-Based Filtering:** This filtration strategy is based on the data provided about the items. The algorithm recommends products that are **similar** to the ones that a user has liked in the **past**. This similarity (generally cosine similarity) is computed from the data we have about the items as well as the user’s past preferences.

- **Collaborative Based Filtering:** This filtration strategy is based on the combination of the user’s behaviour and comparing and contrasting that with **other users’** behaviour in the database. The history of **all users** plays an important role in this algorithm.

  There are two types of Collaborative filtering:

1. User-based Collaborative filtering

The basic idea here is to find users that have **similar past preference patterns** as the user 'A' has had and then recommending him or her items liked by those similar users which 'A' has not encountered yet. This is achieved by making a matrix of items each user has rated/viewed/liked/clicked depending upon the task at hand, and then computing the similarity score between the users and finally recommending items that the concerned user isn't aware of but users similar to him/her are and liked it.

2. Item-based Collaborative Filtering

The concept in this case is to find similar movies instead of similar users and then recommending similar movies to that 'A' has had in his/her past preferences. This is executed by finding every pair of items that were rated/viewed/liked/clicked by the same user, then measuring the similarity of those rated/viewed/liked/clicked across all user who rated/viewed/liked/clicked both, and finally recommending them based on similarity scores.

In this implementation, when the user searches for a movie we will recommend the top 10 similar movies using our movie recommendation system. We will be using an item-based collaborative filtering algorithm for our purpose.

Getting the data up and running

We have imported libraries which we will be using in our movie recommendation system. Also, we'll import the dataset by adding the path of the CSV file.

Movie dataset has

- movieId- once the recommendation is done, we get a list of all similar movie Id and get the tittle for each movie from this dataset.

- genres-which is not required for this filtering approach.

Rating dataset has-

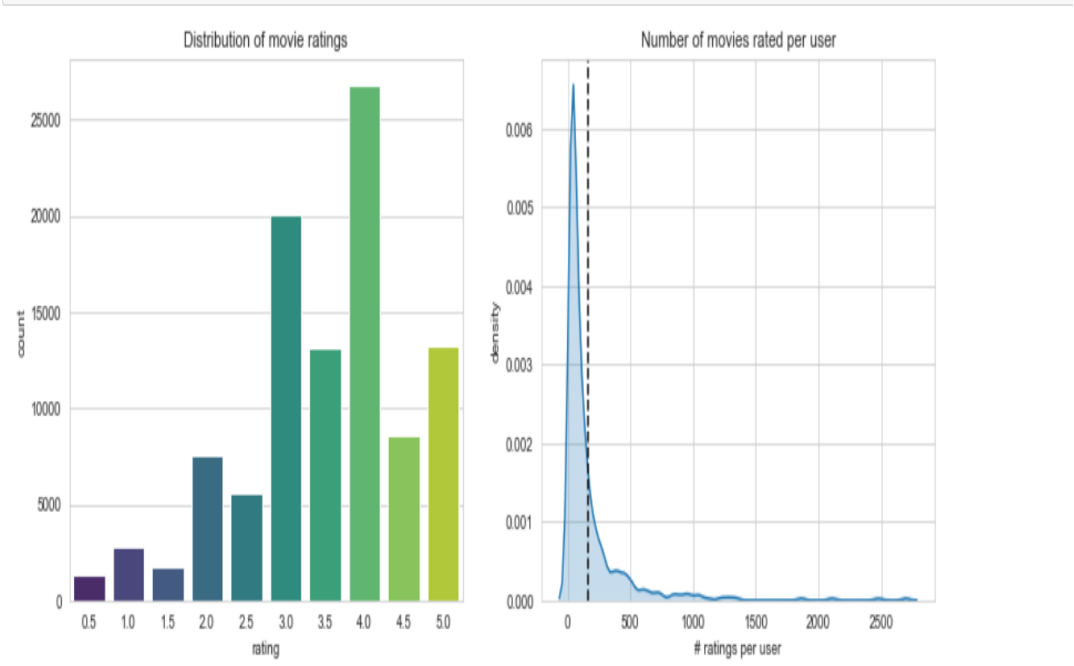
- userId- unique for each user
- movieId-using this feature, we take the tittle of the movie from the movie dataset
- ratings-Ratings given by each user

VISUALIZATION

We have done some visualization using matplotlib and seaborn library which is used for representing data.

Countplot()-:method has been used to show the counts of ratings using bars .We have set the palette , this parameter is used show colours for different levels . In this we have used viridis colour.

Kdeplot()-: Kernel density plot function is used to plot the data against the single variable. It represents the probability distribution of the data values as the area under the plotted curve .



By the above graph we can see that lot of movies(25000) has been rated 4 and also the number of rated per user can be analysed to be around 160-180

We will make a new dataframe where each column representing each userId and each row representing each unique movieId. Fixing the Nan with 0 will make things understandable for the algorithm. Therefore, it will make things easier to understand work with.

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
movieId																					
1	4.0	0.0	0.0	0.0	4.0	0.0	4.5	0.0	0.0	0.0	...	4.0	0.0	4.0	3.0	4.0	2.5	4.0	2.5	3.0	5.0
2	0.0	0.0	0.0	0.0	0.0	4.0	0.0	4.0	0.0	0.0	...	0.0	4.0	0.0	5.0	3.5	0.0	0.0	2.0	0.0	0.0
3	4.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 610 columns

As we are recommending movies, we don't want movies that were rated by a small number of users. Similarly, user who have rated only a handful of movies should also not be taken into account.

- So, in this project a minimum of 10 users should have voted a movie to qualify.
- To qualify a user, a minimum of 50 movies should have voted a movie to qualify.

userId	1	4	6	7	10	11	15	16	17	18	...	600	601	602	603	604	605	606	607	608	610
movieId																					
1	4.0	0.0	0.0	4.5	0.0	0.0	2.5	0.0	4.5	3.5	...	2.5	4.0	0.0	4.0	3.0	4.0	2.5	4.0	2.5	5.0
2	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	...	4.0	0.0	4.0	0.0	5.0	3.5	0.0	0.0	2.0	0.0
3	4.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0
5	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	2.5	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0
6	4.0	0.0	4.0	0.0	0.0	5.0	0.0	0.0	0.0	4.0	...	0.0	0.0	3.0	4.0	3.0	0.0	0.0	0.0	0.0	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
174055	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
176371	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
177765	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	4.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
179819	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
187593	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Our final_data set where most of the values are sparse. We are using only small dataset but for the original dataset, our system may run out of computational resources. Python's **SciPy** provides tools for creating sparse matrices using multiple data structures, as well as tools **crs_matrix** for converting a dense matrix stored in Numpy array to a sparse matrix. The sparse matrix representation outputs the row-column tuple where the matrix contains non-zero values along with those values. This will reduce the sparsity and time complexity.

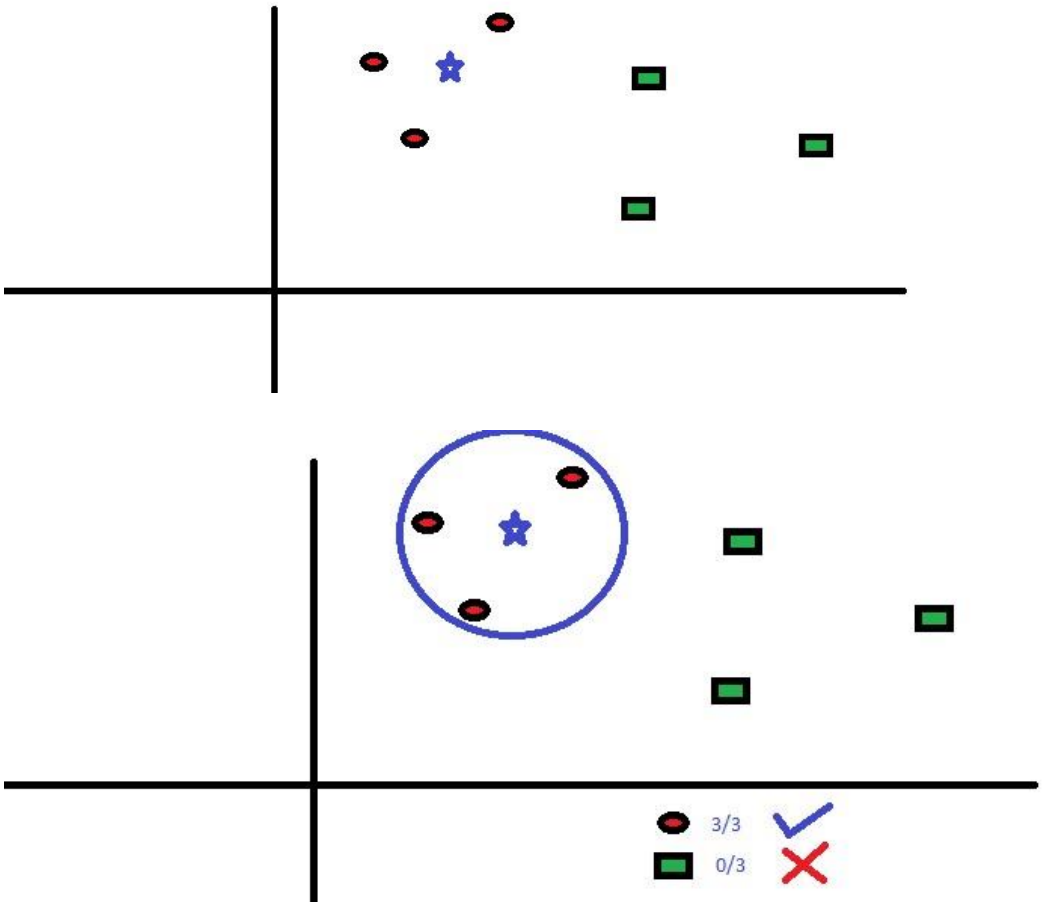
Making the movie recommendation system model

We will be using **the KNN** algorithm to compute similarity with **cosine** distance metric which is very fast and more preferable than Pearson coefficient.

- **K-Nearest Neighbour** is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs' images and based on the most similar features it will put it in either cat or dog category.

  How does KNN algorithm work?

  Suppose there are two categories, and we have a new data point, so this data point lies in which categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



We can implement a KNN model by following the below steps:

1. Load the data
2. Initialise the value of k
3. For getting the predicted class, iterate from 1 to total number of training data points
    1. Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it's the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
    2. Sort the calculated distances in ascending order based on distance values
    3. Get top k rows from the sorted array
    4. Get the most frequent class of these rows
    5. Return the predicted class

Making the recommendation function

The working principle is very simple. We first check if the movie name input is in the database and if it is we use our recommendation system to **find similar movies** and sort them based on their similarity distance and output only the **top 10** movies with their distances from the input movie.

Conclusion

Recommender systems are a powerful new technology for extracting additional value for a business from its user databases. These systems help users find items they want to buy from a business. Recommender systems benefit users by enabling them to find items they like.

Recommender System are used in variety of areas, example taking the form of playlist generators for video and music services, product recommender for online stores. There are also popular recommender system like restaurants, travel products and also helps to increase sales.