Movie Recommendation System Using Cosine Similarity

[1]R. Sivarjun, [2]K. SivaRamaKrisha, [3]Dr. M. Sireesha

[1, 2]B. Tech Student, Department of Computer Science & Engineering,
Narasaraopeta Engineering College, Narasaraopeta, Andhra Pradesh, India.

[3]Professor, Department of Computer Science & Engineering,
Narasaraopeta Engineering College, Narasaraopeta, Andhra Pradesh, India.

[1]sivarjun90@gmail.com, [2]shivaram1352071@gmail, [3]moturisireesha@gmail.com

Abstract -- A Recommendation system is a system that predicts or recommends items based on the data previously stored. A movie Recommendation System is a system that recommends movies based on the user search. Every problem has one solution but, the way to find the solution varies from person to person. For recommendation, we need to consider more than one attribute to recommend the best movie which is liked by the user. Here, In this movie recommendation system, we are going to recommend movies with the help of cosine similarity. A recommendation systems are becoming more powerful tool for improving user experience, increasing engagement, promoting movies, and generating revenue. Personalized movie recommendation systems can provide personalized movie suggestions to users based on their preference and viewing history. This system enhances the user experience, as users are more likely to enjoy movies that align with their interest. A good recommendation can increase user engagement by keeping users on platform for longer period of time. This is because users are more likely to continue watching movies that are interested in. Which is also used to promote new and lesser known movies. In this paper, the movie dataset is taken from the tmdb movies dataset. Analysis and pre-processing of data are done in the python programming language.

1. INTRODUCTION

In our day-to-day life, we are going to use many applications on our mobile. For example, consider the application flip kart, When are searched for an item in that application it will show similar items to your search. Even if you close that particular application it will send you a notification about similar items on your search again and again. Like this, there are many applications we are going to use in our daily life which recommends our favorite products. That means the recommendations are possible only when there is data about a user. Therefore, large organizations collect data from different users in multiple actions. Here, the data is stored in various formats. From that data, the organizations will recommend or predict what the user like most. How to recommend something? Mainly there are three ways to recommend something [5, 12].

Those are:

1. Content-Based Recommendations
2. Collaborative Recommendations
3. Hybrid Recommendations

Here, let us go with one by one.

A. Content-Based Recommendations:

Content-Based Recommendations will be made recommendations based on the user search. For example, consider the content-based movie recommendation system, If a user searches for a particular movie, the system will show the results which are most similar to the search, or if a user liked a movie, Here also it will show similar movies to the liked movie. The data used to recommend a movie is dependent on the particular user. Data is nothing but information or raw data about a user or something. Recommendations will change from person to person [6, 9]. For better understanding about Content-Based Recommendation System consider Fig 1.

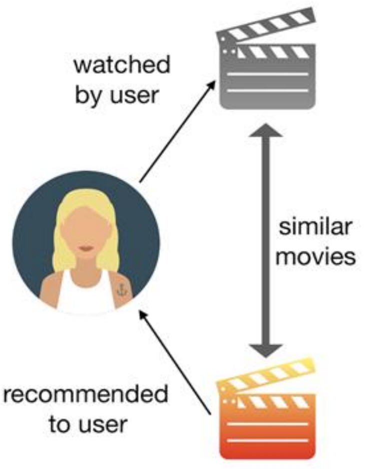


Fig 1: Example for Content-Based Recommendations

B. Collaborative Recommendations:

Collaborative Recommendations will be made recommendations based on the similarity between the users either positively or negatively. Here, The data of multiple users are used to make recommendations. For example, consider the collaborative movie recommendation system, Let us consider User-A, User-B, and User-C. User-A liked some movies, User-B liked some movies and User-C liked some movies. If User-A and User-B like almost all similar movies, Then the movies which is not visited by User-A and which are liked by User-B will recommend to User-A and wise versa. If User-A-liked movies are almost all negative to User-C, Therefore here it shows that the movies which are disliked by User-A are liked by User-C. Taking this into consideration the collaborative movie recommendation system will recommend the movies to the movies to User-A which are disliked by User-B. This is how the collaborative recommendations are done [7, 10]. For better understanding about Collaborative Recommendation System consider Fig 2.

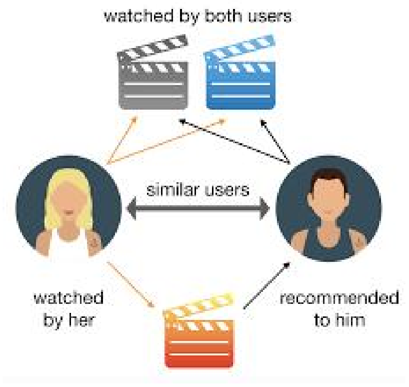


Fig 2: Example for Collaborative Recommendation

C. Hybrid Recommendations:

Both content-based recommendation system and collaborative recommendation systems have individual advantages and disadvantages to overcome the Hybrid Recommendation System introduced. Hybrid Recommendation is nothing but the combination of both advantages of a content-based recommendation system and a collaborative recommendation system [8, 11]. Fig 3 gives you the better explanation about how the Hybrid Recommendation System works.

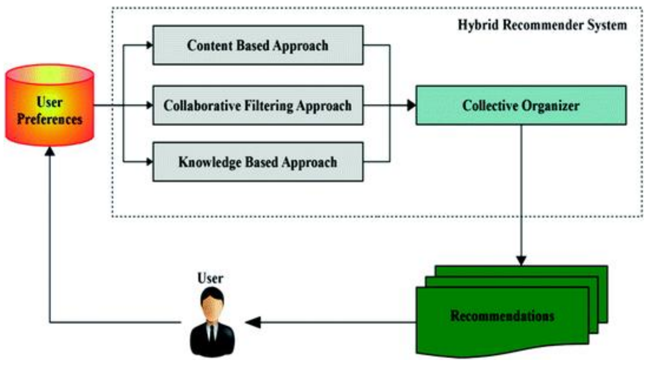


Fig 3: Example for Hybrid Recommendations

2. LITERATURE SURVEY

As we discussed above there are ways to made recommendations. Making genres as one matrix, Ratings which are above “3” will noted as “1” and which are below the “3” will rated as “-1” then finding the dot product of those two matric, finally finding the Euclidian distance between the users the recommendations are made [1].

The article "Matrix Factorization Techniques for Recommender Systems" by Koren, Bell, and Volinsky discusses the use of matrix factorization techniques to improve the accuracy of recommender systems. The authors begin by explaining the challenges of building a recommender system, namely the problem of sparsity, where there are many missing values in the data matrix that represents user-item interactions [2]. Matrix factorization techniques aim to overcome this problem by decomposing the data matrix into two lower-rank matrices, which can then be used to predict missing values. The authors then describe two popular matrix factorization techniques: Singular Value Decomposition (SVD) and Alternating Least Squares (ALS). They explain the advantages and disadvantages of each technique and compare their performance on real-world datasets. The article also discusses several extensions and modifications of matrix factorization techniques, such as incorporating side information and using regularization to prevent overfitting. Finally, the authors provide some insights into the practical implementation of matrix factorization techniques, including the choice of hyperparameters, handling missing values, and dealing with large datasets. Overall, the article provides a comprehensive overview of matrix factorization techniques for recommender systems, their advantages and limitations, and practical considerations for their implementation. The authors demonstrate the effectiveness these techniques on real-world datasets and highlight the potential for further improvements and extensions [2].

Wang et al. (2015) proposed a novel deep learning approach for collaborative filtering-based recommendation systems, called Collaborative Deep Learning (CDL). The CDL model consists of two parts: a deep neural network for learning feature representations of user and item data, and a collaborative filtering component that combines these feature representations to make recommendations. The feature representation learning component uses a stacked auto encoder to encode user and item data into low-dimensional feature vectors. The collaborative filtering component then combines these feature vectors to estimate the rating that a user would give to an item. The authors evaluated the CDL model on two large-scale datasets: the MovieLens dataset and the Netflix dataset [3]. They compared the performance of CDL with several state-of-the-art collaborative filtering algorithms, including SVD, SVD++, and NMF. The experimental results showed that CDL outperformed these baseline algorithms on both datasets, achieving significant improvements in recommendation accuracy. Overall, the CDL model demonstrates the potential of deep learning techniques for improving the performance of collaborative filtering-based recommendation systems, and it provides a promising direction for future research in this area. The feature representation learning component uses a stacked autoencoder to encode user and item data into low-dimensional feature vectors. The collaborative filtering component then combines these feature vectors to estimate the rating that a user would give to an item [3].

The paper "Collaborative Deep Learning for Recommender Systems" by Wang et al. proposes a novel approach for building recommender systems using deep learning techniques. The authors begin by highlighting the limitations of traditional collaborative filtering techniques, such as sparsity and scalability issues [4]. They propose a solution based on deep learning, which is a type of neural network that can learn complex representations of the input data.

The proposed approach, called Collaborative Deep Learning (CDL), consists of a three-layer neural network that takes as input user-item interaction data and outputs a predicted rating for each user-item pair. The first layer of the network represents the users, the second layer represents the items, and the third layer combines the user and item representations to generate the predicted ratings. The authors also propose an extension to the basic CDL model that incorporates auxiliary information, such as user and item attributes, to improve the accuracy of the recommendations. The paper provides experimental results on two large-scale datasets that demonstrate the superiority of the CDL approach over traditional collaborative filtering techniques and other state-of-the-art deep learning-based recommender systems [4]. The authors also perform ablation studies to evaluate the contribution of each component of the CDL model.

Overall, the paper presents a novel and effective approach for building recommender systems using deep learning techniques. The proposed CDL model addresses some of the limitations of traditional collaborative filtering techniques and provides state-of-the-art performance on real-world datasets. The authors also highlight the potential for future research to extend the CDL approach to incorporate more complex user-item interactions and to handle dynamic environments [4].

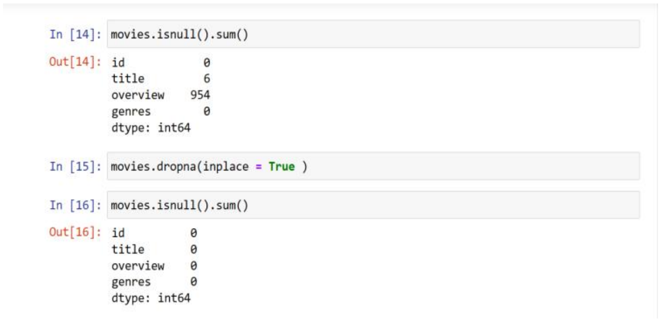
3. CONTENT BASED MOVIE RECOMMENDATION SYSTEM USING COSINE SIMILARITY

In this paper, we are mainly going to discuss the Content-Based Movie Recommendation System. A simple recommendation takes a single attribute and a complex recommendation takes multiple attributes. In this Movie Recommendation System, the attributes considered are Id, Title, Overview, Cast, Crew, and Keywords.

Id: It is unique to every record in the dataset.
Title: This denotes the title of the movie.
Overview: It consists of a summary of the movie Which plays an important role while predicting the movie.
Cast: It denotes the actors and actresses present in the movie.
Crew: It consists of the people like the director, actor,… background roles of the movie. Keywords: It carries the important word used in a particular movie.

Algorithm:
Step 1: Collect the dataset and analyze it.
Step2: Data pre-processing
a. Future extraction
b. Dropping missing values
c. Dropping duplicate values
d. Visualizing each end of every attribute
e. Data cleaning
Step 3: Finding vectors with CountVectorization.
Step 4: Finding Similarity by using Cosine-Similarity.
Step5: Building a model recommending the top 5 highest similarity movies for the searched or selected movie.

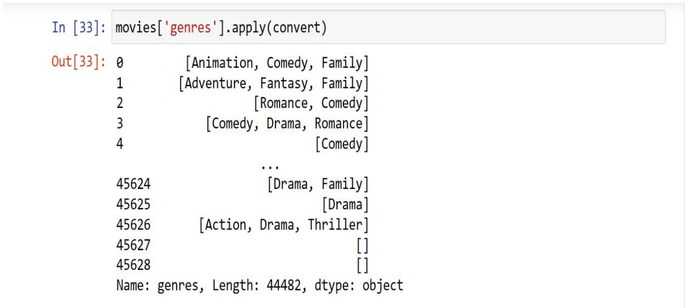
4. RESULTS



```
In [14]: movies.isnull().sum()

Out[14]: id            0
         title         6
         overview    954
         genres        0
         dtype: int64

In [15]: movies.dropna(inplace = True )

In [16]: movies.isnull().sum()

Out[16]: id          0
         title       0
         overview    0
         genres      0
         dtype: int64
```

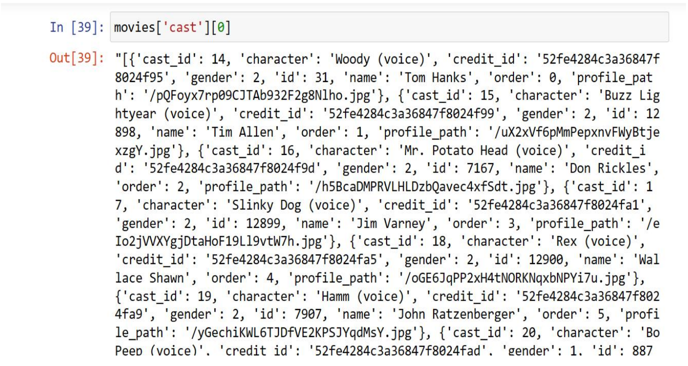
Fig 4: Dropping null values

The genre attribute consists of data like -
movies. iloc[0].genres
"[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id': 10751, 'name':
'Family'}]"



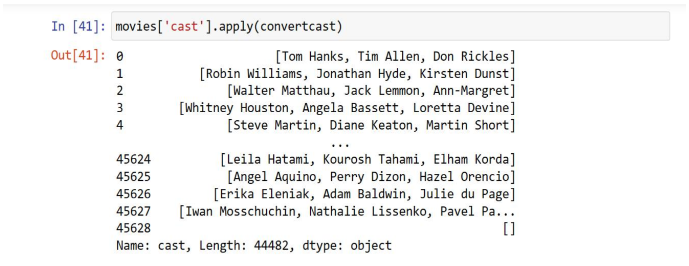
```
In [33]: movies['genres'].apply(convert)

Out[33]: 0        [Animation, Comedy, Family]
         1       [Adventure, Fantasy, Family]
         2                 [Romance, Comedy]
         3          [Comedy, Drama, Romance]
         4                          [Comedy]
                             ...
         45624               [Drama, Family]
         45625                       [Drama]
         45626      [Action, Drama, Thriller]
         45627                            []
         45628                            []
         Name: genres, Length: 44482, dtype: object
```

Fig 5: After cleaning Genres attribute



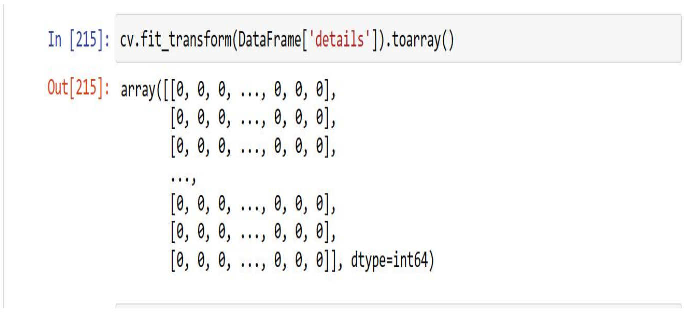
```
In [39]: movies['cast'][0]

Out[39]: "[{'cast_id': 14, 'character': 'Woody (voice)', 'credit_id': '52fe4284c3a36847f
         8024f95', 'gender': 2, 'id': 31, 'name': 'Tom Hanks', 'order': 0, 'profile_pat
         h': '/pQFoyx7rp09CJTAb932F2g8Nlho.jpg'}, {'cast_id': 15, 'character': 'Buzz Lig
         htyear (voice)', 'credit_id': '52fe4284c3a36847f8024f99', 'gender': 2, 'id': 12
         898, 'name': 'Tim Allen', 'order': 1, 'profile_path': '/uX2xVf6pMmPepxnvFWyBtje
         xzgY.jpg'}, {'cast_id': 16, 'character': 'Mr. Potato Head (voice)', 'credit_i
         d': '52fe4284c3a36847f8024f9d', 'gender': 2, 'id': 7167, 'name': 'Don Rickles',
         'order': 2, 'profile_path': '/h5BcaDMPRVLHLDzbQavec4xfSdt.jpg'}, {'cast_id': 1
         7, 'character': 'Slinky Dog (voice)', 'credit_id': '52fe4284c3a36847f8024fa1',
         'gender': 2, 'id': 12899, 'name': 'Jim Varney', 'order': 3, 'profile_path': '/e
         Io2jVVXYgjDtaHoF19Ll9vtW7h.jpg'}, {'cast_id': 18, 'character': 'Rex (voice)',
         'credit_id': '52fe4284c3a36847f8024fa5', 'gender': 2, 'id': 12900, 'name': 'Wal
         lace Shawn', 'order': 4, 'profile_path': '/oGE6JqPP2xH4tNORKNqxbNPYi7u.jpg'},
         {'cast_id': 19, 'character': 'Hamm (voice)', 'credit_id': '52fe4284c3a36847f802
         4fa9', 'gender': 2, 'id': 7907, 'name': 'John Ratzenberger', 'order': 5, 'profi
         le_path': '/yGechiKWL6TJDfVE2KPSJYqdMsY.jpg'}, {'cast_id': 20, 'character': 'Bo
         Peep (voice)', 'credit_id': '52fe4284c3a36847f8024fad', 'gender': 1, 'id': 887
```

Fig 6: Before cleaning cast attribute



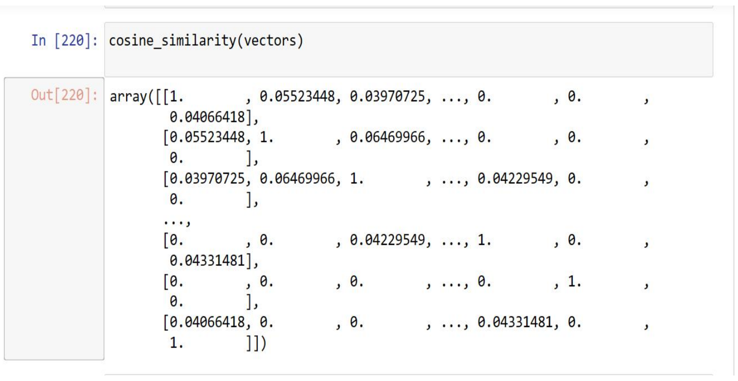
```
In [41]: movies['cast'].apply(convertcast)

Out[41]: 0                         [Tom Hanks, Tim Allen, Don Rickles]
         1              [Robin Williams, Jonathan Hyde, Kirsten Dunst]
         2                  [Walter Matthau, Jack Lemmon, Ann-Margret]
         3       [Whitney Houston, Angela Bassett, Loretta Devine]
         4                  [Steve Martin, Diane Keaton, Martin Short]
                                        ...
         45624             [Leila Hatami, Kourosh Tahami, Elham Korda]
         45625             [Angel Aquino, Perry Dizon, Hazel Orencio]
         45626            [Erika Eleniak, Adam Baldwin, Julie du Page]
         45627       [Iwan Mosschuchin, Nathalie Lissenko, Pavel Pa...
         45628                                                      []
         Name: cast, Length: 44482, dtype: object
```

Fig 7: After cleaning cast attribute



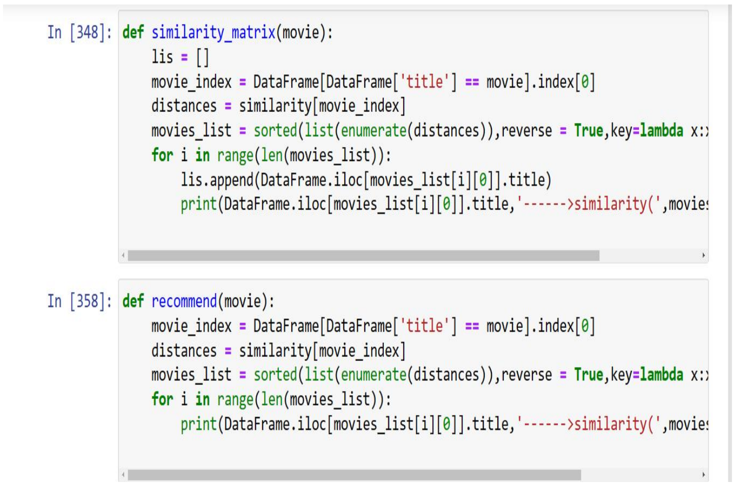
```
In [215]: cv.fit_transform(DataFrame['details']).toarray()

Out[215]: array([[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]], dtype=int64)
```

Fig 8: Text data is converted into vectors



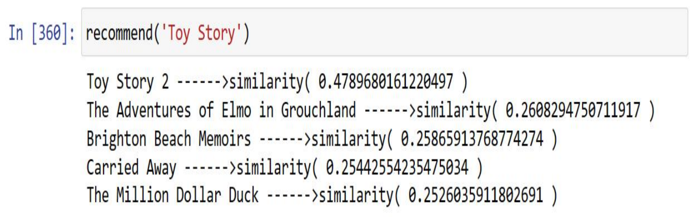
```
In [220]: cosine_similarity(vectors)

Out[220]: array([[1.        , 0.05523448, 0.03970725, ..., 0.        , 0.        ,
                  0.04066418],
                 [0.05523448, 1.        , 0.06469966, ..., 0.        , 0.        ,
                  0.        ],
                 [0.03970725, 0.06469966, 1.        , ..., 0.04229549, 0.        ,
                  0.        ],
                 ...,
                 [0.        , 0.        , 0.04229549, ..., 1.        , 0.        ,
                  0.04331481],
                 [0.        , 0.        , 0.        , ..., 0.        , 1.        ,
                  0.        ],
                 [0.04066418, 0.        , 0.        , ..., 0.04331481, 0.        ,
                  1.        ]])
```

Fig 9: Finding the similarity between the vectors



```
In [348]: def similarity_matrix(movie):
              lis = []
              movie_index = DataFrame[DataFrame['title'] == movie].index[0]
              distances = similarity[movie_index]
              movies_list = sorted(list(enumerate(distances)),reverse = True,key=lambda x::
              for i in range(len(movies_list)):
                  lis.append(DataFrame.iloc[movies_list[i][0]].title)
                  print(DataFrame.iloc[movies_list[i][0]].title,'------>similarity(',movies

In [358]: def recommend(movie):
              movie_index = DataFrame[DataFrame['title'] == movie].index[0]
              distances = similarity[movie_index]
              movies_list = sorted(list(enumerate(distances)),reverse = True,key=lambda x::
              for i in range(len(movies_list)):
                  print(DataFrame.iloc[movies_list[i][0]].title,'------>similarity(',movies
```

Fig 10: Function for finding similarity for given movie and recommendations for the given movie



```
In [360]: recommend('Toy Story')

          Toy Story 2 ------>similarity( 0.4789680161220497 )
          The Adventures of Elmo in Grouchland ------>similarity( 0.2608294750711917 )
          Brighton Beach Memoirs ------>similarity( 0.25865913768774274 )
          Carried Away ------>similarity( 0.25442554235475034 )
          The Million Dollar Duck ------>similarity( 0.2526035911802691 )
```

Fig 11: Recommendations for given movie with similarity

5. CONCLUSION AND FUTURE WORK

In this paper, the content-based movie recommendation system is based on attributes named like genres, keywords, cast, crew, and overview. To increase the accuracy of the prediction multiple attributes are used. Here, we are recommending the top 5 highest similar movies to the selected or searched movie. Cross-validation will represent whether the recommendations given by the model are correct or not. Based on the similarity of the movie we are displaying them in descending order and recommending the top movies which have the highest similarity. By using different machine learning algorithms we can improve the accuracy level of the prediction. It is a simple recommendation system using machine learning algorithms. Further, we can improve the model using various algorithms with high accuracy.

REFERENCES

1. SRS Reddy, Sravani Nalluri, Subramanyam Kunisetti, S. Ashok and Venkatesh ----- Content-Based Movie Recommendation System Using Genre Correlation.
2. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.
3. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017). Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web (pp. 173-182).
4. Wang, H., Wang, N., Yeung, D. Y., & Shi, W. (2015). Collaborative deep learning for recommender systems. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1235-1244).
5. https://medium.com/mlearning-ai/what-are-the-types-of-recommendation-systems-3487cbafa7c9.
6. https://medium.com/web-mining-is688-spring-2021/content-based-movie-recommendation-system-72f122641eab#:~:text=Content%20Based%20Recommendation%20System%3A%20It,a%20show%20similar%20to%20it.
7. https://towardsdatascience.com/tensorflow-for-recommendation-model-part-1-19f6b6dc207d.
8. https://towardsdatascience.com/creating-a-hybrid-content-collaborative-movie-recommender-using-deep-learning-cc8b431618af.

9. https://analyticsindiamag.com/how-to-build-a-content-based-movie-recommendation-system-in-python/.
10. https://ieeexplore.ieee.org/document/9155879.
11. https://ieeexplore.ieee.org/document/9510058.
12. https://www.researchgate.net/publication/346651834_Various_Types_of_Movies_Recommendation_System.
13. Ghuli, P., Ghosh, A., Shettar, R.: A collaborative filtering recommendation engine in a distributedenvironment. In: 2014 International Conference on Contemporary Computing and Informatics (IC3I). IEEE (2014).
14. Zhao, L., et al.: Matrix factorization + for movie recommendation. In: IJCAI (2016).
15. Bhatt, B.: A review paper on machine learning based recommendation system. Int. J. Eng. Dev. Res. (2014).