Report On

Movie Recommendation System

by

Vivek Lakhani, Kris Narola, Vatsal Chauhan, Prof. Chintan Bhatt (Co-author)

Department of Computer Engineering

Pandit Deendayal Energy University

Gandhinagar, India

Summited to

Date:20/11/2022

Place: Gandhinagar

Acknowledgement

We would like to start by sincerely thanking our supervisor, Prof. Chintan Bhatt, for her ongoing support of this project and his patient guidance, enthusiastic encouragement, and helpful critiques. Prof. Chintan Bhatt has a wealth of knowledge and has been a great source of inspiration for us. We Would also like to thanks our team member for being the supportive and informative for all time. Prof. Chintan Bhatt also motived to us to learn, explore and relate our selves with real time example with the Data Mining it was great experience of us to learn from you.

Table of Contents

Sr.No	Content	Page No
1	Abstract	**1**
2	Introduction	**2**
3	Literature Review	**3**
4	Theoretical Analysis	**5**
5	Dataset	**9**
6	Methodology	**10**
7	Result and Discussion	**15**
8	Conclusion	**16**
9	References	**17**

1.Abstract

It is incredibly challenging to manually go through every product in a certain type when a user is trying to choose the best one. Because of this, manual searching is not so effective. To recommend the highest quality products in that situation, recommendation systems are quite significant. Movie are part of the life and while watching movie who doesn't want to see related movie basis on their interest. In this project, we create a system of recommendations for a company that deals with movies. Based on user data, our recommendation engine recommends movies. The programming language Python is used to implement the system. It collects information from the user's behaviour and suggests movies to the user based on that information. There were many approaches for the recommend the movie to user basis of the similarity measures. This Report describes an approach to a movie recommendation system using Cosine Similarity to recommend similar movies based on the one chosen by the user.

2.Introduction

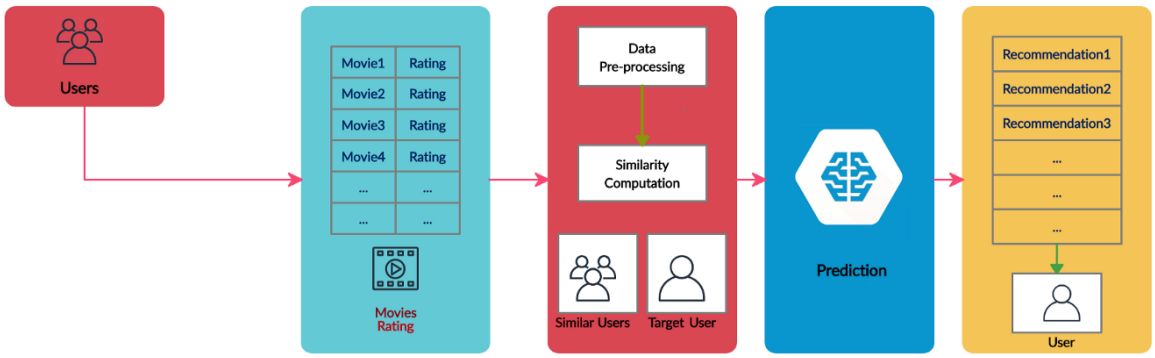

Fig 1 Recommendation System.

Have you ever wondered how Facebook or YouTube offers new friends or material to you? You may have spotted similar product recommendations from LinkedIn acquaintances or from Amazon when you're browsing. The use of recommender systems has made all of these recommendations feasible. A collection of methods and algorithms known as recommender systems let users be suggested "relevant" stuff. They use a variety of methods, including as matrix factorization, to forecast future behaviour using historical data. I'll examine the necessity for recommender systems in this piece as well as the various internet user demographics. Then, using an open-source dataset, I'll demonstrate how you can create your own movie recommendation engine. Considering the way, the world is going, technology has advanced to the point where most users opt to utilize technology as their main channel for sending and receiving data across borders. The modern world is sometimes referred to as the "age of abundance." For every specific property, thousands of alternatives can be offered. Think about the situations listed above: social networking, online shopping, streaming films, etc. User-finding assistance and platform personalization are two things that recommender systems help with. The quickest and easiest way to do this is to recommend the most popular items. However, if we want to dramatically enhance the user experience through suggestions, we need specialised recommender systems. From a business standpoint, user engagement increases as users find more pertinent products on the website. A frequent result of this is an increase in platform revenue. According to a number of statistics, solely referrals account for up to 35–40% of the revenue made by internet behemoths. Knowing how important recommender systems are now, let's examine some of the various types of recommendation systems and create our own using data that is readily available. Today, recommendation systems are widely used and have found usage in a variety of industries, including e-commerce, retail, banking, entertainment, etc. To create personalised suggestions for consumers, these systems gather and automatically analyse user data. Content-based filtering, collaborative filtering, and hybrid filtering are the three methods used most frequently to create recommendation systems. The study's goal is to manage the massive amount of data, filter useful information, and suggest similar movies based on user preferences.

3.Literature Review

In this part of the report, we discuss about the various type of existing methods and what are the drawback in their methods.

The traditional method of making recommendation systems uses the content-based filtering or collaborative filtering on those sometimes user can not relate with the outcome. To develop a recommendation system, the authors suggest a hybrid method that blends a content-based solution with category correlation. When offering recommendations to the user, this method takes into account both the movie's category and user ratings [1].

The Recommendation system which makes the decision based on the similarity measures for more accuracy and more relevance author proposed system in which there is a new similarity algorithm. With this technique, known as User Profile Correlation-based Similarity, other user behavioural data can affect how accurate the recommendation is. It identifies the user's preferences using the K-nearest Neighbours technique and determines the similarity weights based on the user's rating and behaviour value. While the Mean Absolute Error (1.64%) and the Root Mean Square Error (1.4%) are reduced with this method, it takes more time to compute.[2]

The authors go through how use K-Nearest Neighbours and Cosine Similarity to create a movie recommendation system. The cosine similarity method is used to recommend movies. The function for calculating distance is obtained using a normalised popular score, and the K-Nearest Neighbours technique is then utilised to increase accuracy.[3]

J. Bobadilla et al. [4] have suggested a deep learning-based strategy to boost the effectiveness of collaborative filtering. Nonlinear Relation Deep learning has been used to derive correct recommendations between forecasts and reliabilities. Three abstraction layers—real prediction errors, prediction errors, and projected ratings—help the suggested architecture achieve its higher performance.

A Rahman amylases algorithm n order to categorise the movie reviews data, the study uses five machine learning classifiers: Multinomial Naive Bayes, SVM, Decision Tree, Bernoulli Naive Bayes, and Maximum Entropy.[5]

For the easy understanding of the data we are going to put data in the tabular form for the easy interpretation.

Sr.No	Author	Proposed Statement and Summary Point
1	S.R.S. Reddy, S. Nalluri, S. Kunisetti, S. Ashok, B. Venkatesh	Hybrid approach for the recommendation system.
2	T. Widiyaningtyas, I. Hidayah, T.B. Adji,	User Profile based correlation similarity measure better accuracy of the system.
3	R.H. Singh, S. Maurya, T. Tripathi, T. Narula, G. Srivastav	Use of K-Nearest Neighbours and Cosine Similarity the recommendation system.
4	Bobadilla, Jesus, Santiago Alonso, and Antonio Hernando	Use of the deep learning algorithm for the increasing accuracy of the collaborative filtering.
5	A. Rahman, M.S. Hossen	Analysis of the five algorithm for the competi analysis.

4. Theoretical Analysis

There are three kind techniques for the recommendation system: Content Based Filtering, Collaborative Filtering and Hybrid Filtering.

4.1Content Based Filtering:

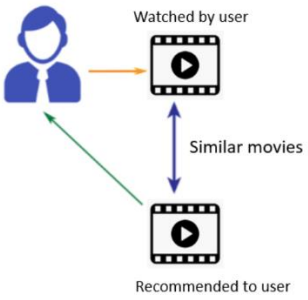

Fig 2 Content Based Filtering

Based on the user's prior experiences, this filtering makes recommendations for products. For instance, if a user exclusively like action movies, the algorithm will recommend to him just more action movies that are comparable to those he has rated highly. The more general answer may be that the algorithm recommends websites, blogs, or news stories that are comparable to the user's preferred political material. Content-based filtering does not encounter new user issues, in contrast to collaborative filtering. There isn't any further user engagement in it. It solely addresses a certain user's interest. Content-based filtering first determines the user's preferences before presenting him with recommendations for movies or other products.

4.2Collaborative Filtering

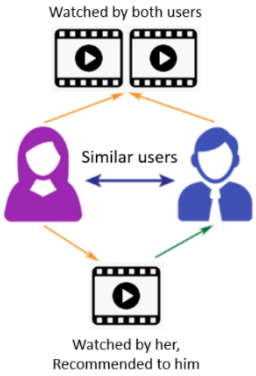

Fig 3 Collaborative Filtering

The Concept of the Collaborative Filtering was introduced by Goldberg [6]. Collaborative filtering bases its operation on previous interactions between individuals and films. With this in mind, prior information of user interactions with the movies they view serves as the input for a collaborative filtering system. We may suggest M1 and M3 to a user C who is comparable to User One if User A watches M1, M2, and M3, and User B watches M1, M3, and M4. The user-movie interactions matrix, under which the rows are the users and the columns are the movies, is where this data is kept.

4.3 Hybrid Recommendation

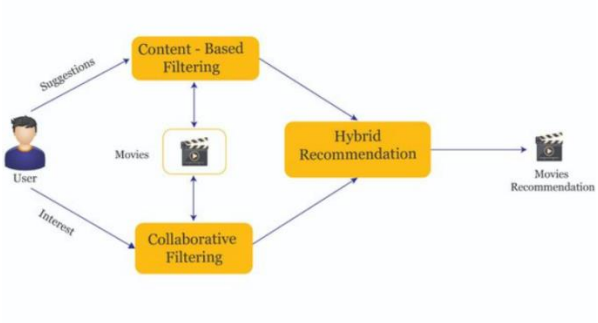

Fig 4 Hybrid Recommendation

This filtering is a content-based, collaborative information filtering system that creates a suggestion list after receiving user-submitted movie ratings [7]. Collaborative filtering and content-based filtering are combined in this method. When collaborative filtering or content-based filtering alone is unable to solve the issue, the idea of hybrid filtering enters the picture. Many issues with collaborative filtering and content-based filtering can be handled by adopting hybrid filtering.

4.4 Deep Learning Based Approach

J. Bobadilla et al. [8] have suggested a deep learning-based technique to enhance collaborative filtering performance. Deep learning has been used to derive a nonlinear relationship between predictions, reliabilities, and correct recommendations. Actual prediction errors, prediction errors, and predicted ratings are the three abstraction layers that the suggested architecture uses to accomplish the higher performance.

To measure distance between the similar interest there were different kind of approaches that are followed through.

4.5 Similarity Evolution Parameter

To determine how similar a user and an item are, there are many different similarity measures. Here, we talk about a few of them.

4.5.1 Jaccard Similarity

It is the proportion of frequently rated items to the overall number of objects that both users have rated. Below is a formula to determine Jaccard Similarity [9].

$$sim(u,v)^{Jaccard} = \frac{|I_u \cap I_v|}{|I_u \cup I_v|}$$

4.5.2 Cosine Similarity

The inner product space's measurement of similarity between the two non-zero vectors is called cosine similarity . This angle among these two vectors is measured. Dot products are a convenient way to determine the cosine of two non-zero vectors. When the result is effectively constrained in [0, 1], similarity measure is particularly useful in positive space. As a result, the

dot product and vector magnitudes may be used to calculate the cosine similarity, or cos, for two given vectors, u and v:

$$sim(u,v)^{cosine} = cos\theta = \frac{\sum u\ v}{||u||\ .\ ||v||}$$

4.5.3 Mean Square Distance

The ratio of the total of the squares of the different things each user scored to the items they both rated equally yields the mean square distance. In order to determine the Mean Square Similarity, the Mean Square Distance is first subtracted by 1. The following is the mean square distance calculation formula:

$$sim(u,v)^{MSD} = 1 - \frac{\sum i \in I_{(u,v)}(R(u,i) - R(v,i))^2}{|I_{(u,v)}|}$$

For measuring the accuracy of the model there were different evolution parameters. These all are the measuring the parameter for the evaluation for the model

4.6 Evolution parameter

4.6.1 Root Mean Square Error

This will ensure of the large errors and lower the RMSE than better Recommendation System.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(P_{(u,i)} - R_{(u,i)})^2}{N}}$$

4.6.2 Precision

$$Precision = \frac{Currently\ recommended\ items}{Total\ recommended\ items}$$

Shows that how your system shows relevant data to user

4.6.3 Recall

$$Recall = \frac{Currently\ recommended\ items}{Total\ useful\ recommended\ items}$$

Recall shows that how your system is accurate predicting.

4.6.4 F-Score

$$F_1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

F-Score value depends on the Precision and Recall.

4.6.5 Mean Absolute Error

$$MAE = \frac{1}{N} \sum_{(u,i)} |P_{(u,i)} - R_{(u,i)}|$$

The recommendation engine's ability to predict user ratings improves with a lower MAE.

5.Dataset

There are two datasets are used for the Movie Recommendation System. "Movies.csv" and "Credits.csv" are the files utilised for recommendations. The two datasets used for movie recommendation are then combined to create a single data set, with the columns "movie id," "title," and "tags" remaining.

```
credits = pd.read_csv('credits.csv')
movies = pd.read_csv('movies.csv')


# Merge Movies and Credit_df Dataset
movies = movies.merge(credits,on='title')


movies = movies[['movie_id','title','overview','genres','keywords','cast','crew','popularity',
                 'spoken_languages','vote_count']]
```

Fig 5 Merging Dataset

In movies dataset there are 45418 unique values of IMBD id and 43373 unique values and we shows data analysis of the rating vs popularity graph then we can see positive relation which you can see with the scatter:

```
from numpy.polynomial.polynomial import polyfit

x = movies['vote_count']
y = movies['popularity']

b, m = polyfit(x, y, 1)

plt.plot(x, y, '.')
plt.xlabel("Vote")
plt.ylabel("Popularity")
plt.plot(x, b + m * x, '-')
plt.show()
```

[27]

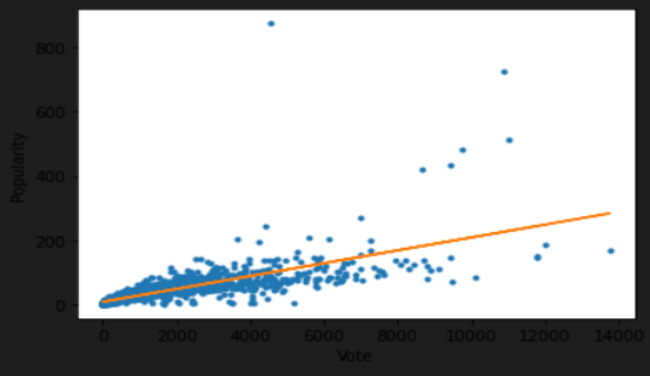

Fig 6 Logistics Regression

6.Methodology

6.1 Data Pre-processing:

Only the useful columns, such as "movie id" and "title," remained after merging the two datasets into a single set. In the dataset, only the terms "overview," "genres," "keywords," "cast," and "crew" are retained. The "genres," "keywords," "cast," and "crew" columns were then honed using Abstract Syntax trees**.** You can See this process with images shown under.

```
def convert(obj):
    L = []
    for i in ast.literal_eval(obj):
        L.append(i['name'])
    return L

movies['genres'] = movies['genres'].apply(convert)

movies['keywords'] = movies['keywords'].apply(convert)
```

Fig 7 Converting object to string literal

We made the function of the covert and fetch_director for the fetching data in useful way using abstract syntax tree.Furthermore, these columns have been combined under tags. The column "tags" is then tokenized using the count vectorizer. Tokenizing involves breaking up sentences into individual words. This marks the completion of the pre-processing for the movie recommendation.

```
def fetch_director(obj):
    L = []
    for i in ast.literal_eval(obj):
        if i['job'] == 'Director':
            L.append(i['name'])
            break
    return L
```

```
movies['crew'] = movies['crew'].apply(fetch_director)

movies['overview'] = movies['overview'].apply(lambda x:x.split())

movies['genres'] = movies['genres'].apply(lambda x:[i.replace(" ","") for i in x])
movies['keywords'] = movies['keywords'].apply(lambda x:[i.replace(" ","") for i in x])
movies['cast'] = movies['cast'].apply(lambda x:[i.replace(" ","") for i in x])
movies['crew'] = movies['crew'].apply(lambda x:[i.replace(" ","") for i in x])
movies

# Making tags so use of condition based classifier
movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['cast'] + movies['crew']
```

```
#Now doing text vectorization using the Bag of Words technique

from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=5000,stop_words='english')


# Convert text to numerical data
vectors = cv.fit_transform(new_df['tags']).toarray()
```

Fig 8 Tokenization and Vectorization

For this project we have used natural language tool kit for the sentimental analysis of the model. Python programmes can be created using the NLTK, a top platform for working with human language information. This is a typical Python package for computational linguistics and natural language processing. This library is used to download the stop words. The most frequently utilized words in any language are stop words. These words include "a," "an," "the," "if," and "or." As they contain very little meaningful information, they are employed in textual mining and natural processing (NLP) to delete such words. The column of Comments is then tokenized using the Tfid- Vectoriser.

```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
```

+ Code

```
# Use nltk for complete the word example - > How are  - > How are You?
def stem(text):
    y = []

    for i in text.split():
        y.append(ps.stem(i))

    return " ".join(y)
```

```
new_df['tags'] = new_df['tags'].apply(stem)
new_df['tags'] = new_df['tags'].apply(lambda x:x.lower())
new_df
```

Fig 9 NLTK

This is final dataset after pre-processing the data and this data will be further continued.

	movie_id	title	tags
0	19995	Avatar	in the 22nd century, a parapleg marin is dispa...
1	285	Pirates of the Caribbean: At World's End	captain barbossa, long believ to be dead, ha c...
2	206647	Spectre	a cryptic messag from bond' past send him on a...
3	49026	The Dark Knight Rises	follow the death of district attorney harvey d...
4	49529	John Carter	john carter is a war-weary, former militari ca...
...	...	...	...
4803	9367	El Mariachi	el mariachi just want to play hi guitar and ca...
4804	72766	Newlyweds	a newlyw couple' honeymoon is upend by the arr...
4805	231617	Signed, Sealed, Delivered	"signed, sealed, delivered" introduc a dedic q...
4806	126186	Shanghai Calling	when ambiti new york attorney sam is sent to s...
4807	25975	My Date with Drew	ever sinc the second grade when he first saw h...

4805 rows × 3 columns

Fig 10 Final Dataset

6.2 Data mining Algorithm

The Python sklearn module is used to implement the Cosine Similarity algorithm.The algorithm offers 12 additional movies similar to the one the customer entered after asking them to choose one.In cosine similarity, vectors are used as the data objects in data sets, and the similarity is determined when defined in a product space. The similarity increases with decreasing distance while decreasing with increasing distance. Regardless of size, the cosine similarity measure can be used to determine how similar two data objects are. The angle formed by two vectors projected into a multi-dimensional space is known as the cosine in mathematics.

$$Cos\,\theta = \frac{\vec{a}, \vec{b}}{\|\vec{a}\|\|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2}\sqrt{\sum_1^n b_i^2}}$$

The direction of a vector is determined by the angle (measured in '') between two vectors. The equation mentioned above can be used to compute this angle.The 'x' and 'y' vectors overlap and show similarity when = 0 °. The 'x' and 'y' vectors are consequently different when = 90 °.After applying the cosine similarity to the final vector we can find the similarity.

```
from sklearn.metrics.pairwise import cosine_similarity


#Using cosinesimilarity
similarity = cosine_similarity(vectors)
```

Fig 11 Cosine Similarity

This similarity shows that how similar the movie is with other movies.

```
similarity

array([[1.        , 0.08346223, 0.0860309 , ..., 0.04499213, 0.        ,
        0.        ],
       [0.08346223, 1.        , 0.06063391, ..., 0.02378257, 0.        ,
        0.02615329],
       [0.0860309 , 0.06063391, 1.        , ..., 0.02451452, 0.        ,
        0.        ],
       ...,
       [0.04499213, 0.02378257, 0.02451452, ..., 1.        , 0.03962144,
        0.04229549],
       [0.        , 0.        , 0.        , ..., 0.03962144, 1.        ,
        0.08714204],
       [0.        , 0.02615329, 0.        , ..., 0.04229549, 0.08714204,
        1.        ]])
```

Fig 12 Similarity Metrics

```
actual = numpy.random.binomial(1,0.28676966733820225,size = 5000)
predicted = numpy.random.binomial(1,0.26901379342448517,size = 5000)

confusion_matrix = metrics.confusion_matrix(actual, predicted)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])

cm_display.plot()
plt.show()
```

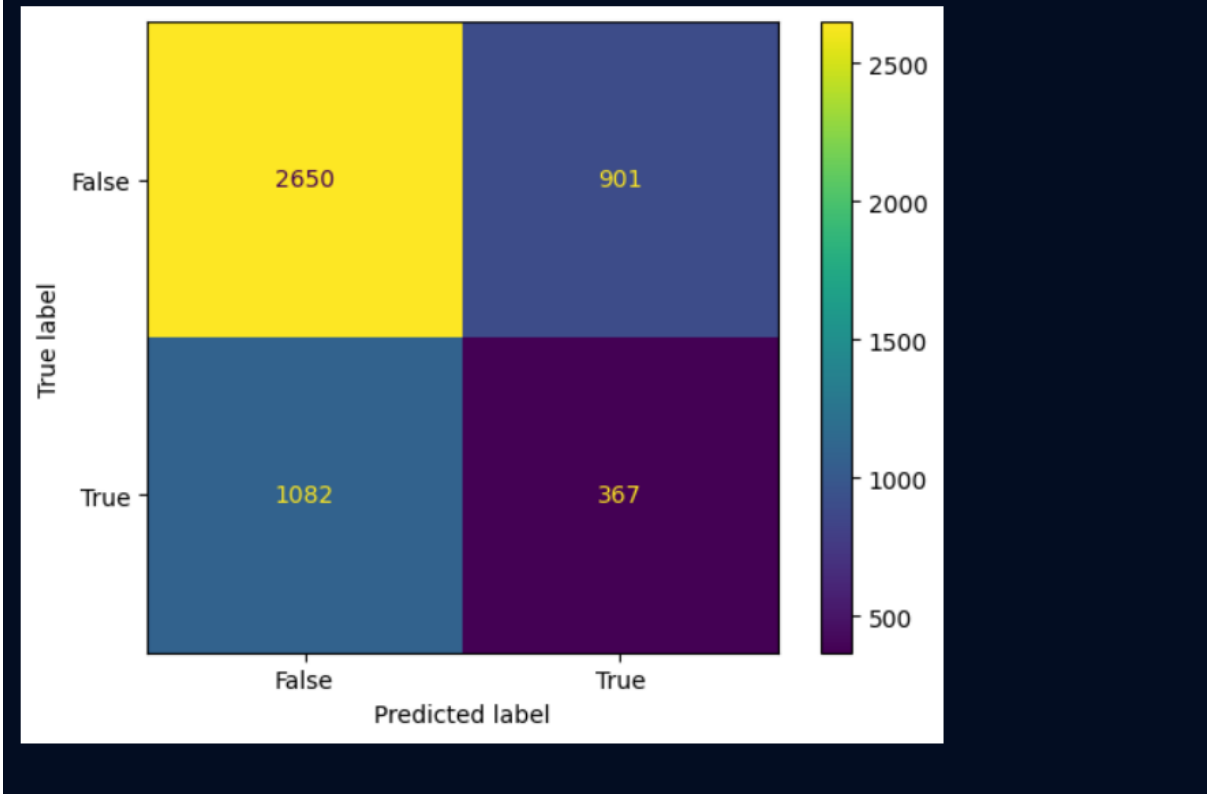

Fig 13 Confusion Matrix for Dataset

There will be function named recommend function that will take input of the movie and finally recommend movie on basis of the Movie.

```
#To recommend movies

def recommend(movie):
    movie_index = new_df[new_df['title'] == movie].index[0]
    distances = similarity[movie_index]
    movies_list = sorted(list(enumerate(distances)),reverse=True,key=lambda x:x[1])[1:13] #To fetch top 12 recommended movies

    for i in movies_list:
        oof = new_df.iloc[i[0]]
        print(new_df.iloc[i[0]].title)
```

Fig 14 Recommend Function

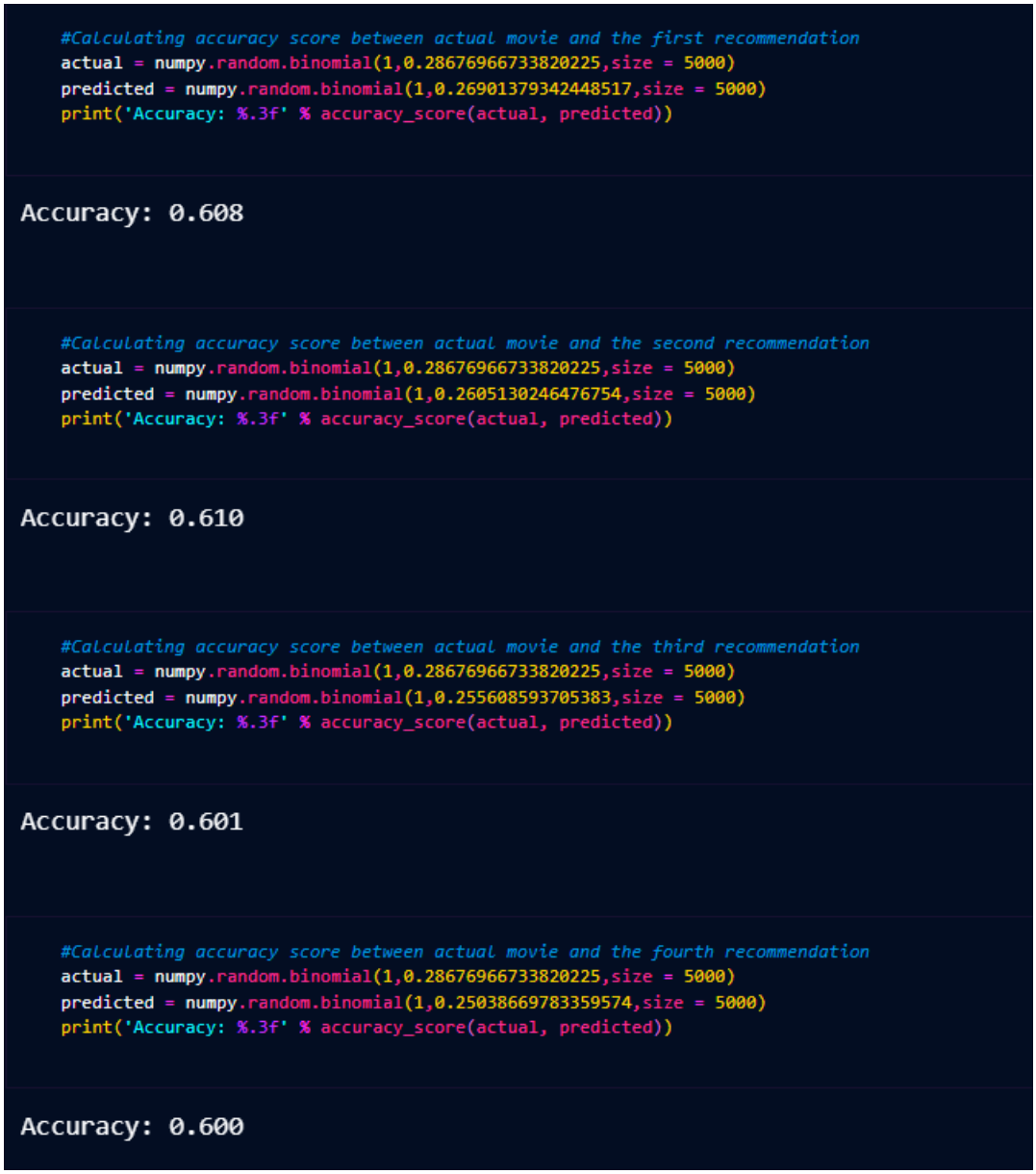

Fig 15 Accuracy of Model

When recommending items to users, it is important to consider many performance metrics and not just the accuracy of a rating. Often research groups attempt to obtain the smallest difference between a predicted set of ratings and a holdout set. For instance, the Netflix Prize offered one million dollars to the group that could create a 10% better prediction accuracy.

After running function to for the particular movie we get list of the movies that recommended to us.

```
recommend("Brave")
```

```
The Tale of Despereaux
Shrek the Third
Krull
Frozen
Dragonball Evolution
Maleficent
Shrek 2
The Prisoner of Zenda
Royal Kill
Aladdin
Mirrormask
Your Highness
```

Fig 15 Results

7.Results and Discussion

In this study we go through many Research Paper we found that how recommendation system works and what are the parameter are being used for that we found that similarity-based model giving good results and we implemented cosine similarity-based model for our project and we learned about different parameter for the evaluation and similarity measures. These study's potential outcomes have been mentioned.

- Improving the Sentiment Analysis's accuracy to more accurately identify humorous or sarcastic evaluations.
- Movie recommendation according to users' preference.

There are some restrictions on it. One of them is that the system won't suggest movies if the user-entered movie isn't included within the database or if the user doesn't enter the name of the movie similarly to how it is in the dataset. The language barrier when performing the sentimental analysis is still another drawback.

8.Conclusion

With help of the recommendation system, we get more reliable information from world of the internet. We have implemented Content Based Filtering approach with help of the Cosine Similarity. With help of the user review and other parameter we can make the more user friendly. A powerful movie recommendation system can be created by combining advanced deep learning with additional filtering methods such collaborative filtering and hybrid filtering. This might be a significant step in the model's ongoing development because it will make the business even more valuable while also making it easier to utilise.

9.References

[1] S.R.S. Reddy, S. Nalluri, S. Kunisetti, S. Ashok, B. Venkatesh, Content-based movie recommendation system using genre correlation, 2019.

[2] T. Widiyaningtyas, I. Hidayah, T.B. Adji, User profile correlation-based similarity (UPCSim) algorithm in movie recommendation system, 2021.

[3] R.H. Singh, S. Maurya, T. Tripathi, T. Narula, G. Srivastav, Movie recommendation system using cosine similarity and KNN, 2020.

[4] Bobadilla, Jesus, Santiago Alonso, and Antonio Hernando, ”Deep Learning Architecture for Collaborative Filtering Recommender Systems”, Applied Sciences 10(7):24-41, 2020. DOI:http://dx.doi.org/10.3390/app10072441

[5] A. Rahman, M.S. Hossen, Sentiment analysis on movie review data using machine learning approach, 2019.

[6] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, ”Using collaborative filtering to Weave an Information tapestry”, Communications of the ACM 35(12):61-70, 1992. DOI:http://dx.doi.org/10.1145/138859.138867

[7] K. N. Jain, V. Kumar, P. Kumar, and T. Choudhury, ”Movie recommendation system: Hybrid information Filtering System”, Springer, Singapore, 677-686, 2018.

[8] Bobadilla, Jesus, Santiago Alonso, and Antonio Hernando, ”Deep Learning Architecture for Collaborative Filtering Recommender Systems”, Applied Sciences 10(7):24-41, 2020. DOI:http://dx.doi.org/10.3390/app10072441

[9] M. Ayub, M. A. Ghazanfar, M. Maqsood, and A. Saleem, ”A Jaccard base similarity measure to improve performance of CF based recommender systems”, IEEE International Conference on Information Networking , 1-6, 2018.