MOVIE RECOMMENDATION SYSTEM
A MINOR PROJECT REPORT

18CSE423T– DATA ANALYTICS IN PUBLIC HEALTH

Submitted by

Talat Binti Firdous[RA2111027010115]

Under the guidance of

Dr. A. Siva Kumar

Department of Data Science and BusinessSystems

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"MOVIE RECOMMENDATION SYSTEM"** is the bonafide work of **Talat Binti Firdous (RA2111027010115)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. A. Siva Kumar

SUPERVISOR

Assistant Professor

Department of DSBS

ABSTRACT

The project introduces an Intelligent Movie Recommendation System designed to enhance user experience in the ever-expanding world of entertainment. Leveraging advanced machine learning algorithms and collaborative filtering techniques, the system analyzes user preferences and historical viewing patterns to provide personalized movie recommendations. The system incorporates a user-friendly interface that allows users to input their preferences, including genre preferences, past movie ratings, and viewing history. By utilizing collaborative filtering, the recommendation engine identifies patterns and similarities between users, enabling accurate predictions of movies that a user might enjoy based on the preferences of users with similar tastes. The recommendation system employs both content-based and collaborative filtering methods to ensure a comprehensive and diverse set of recommendations. Additionally, real-time updates and continuous learning mechanisms are implemented to adapt to evolving user preferences and changing movie landscapes.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

LR	Logistic Regression
NNET	Neural Networks
SVM	Support Vector Machine
CART	Classification and Regression Tree
LSTM	Long Short-Term Memory
LDA	Latent Dirichlet Allocation
RF	Random Forest
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
TF-IDF	Term Frequency-Inverse Document Frequency
BoW	Bag of Words

INTRODUCTION

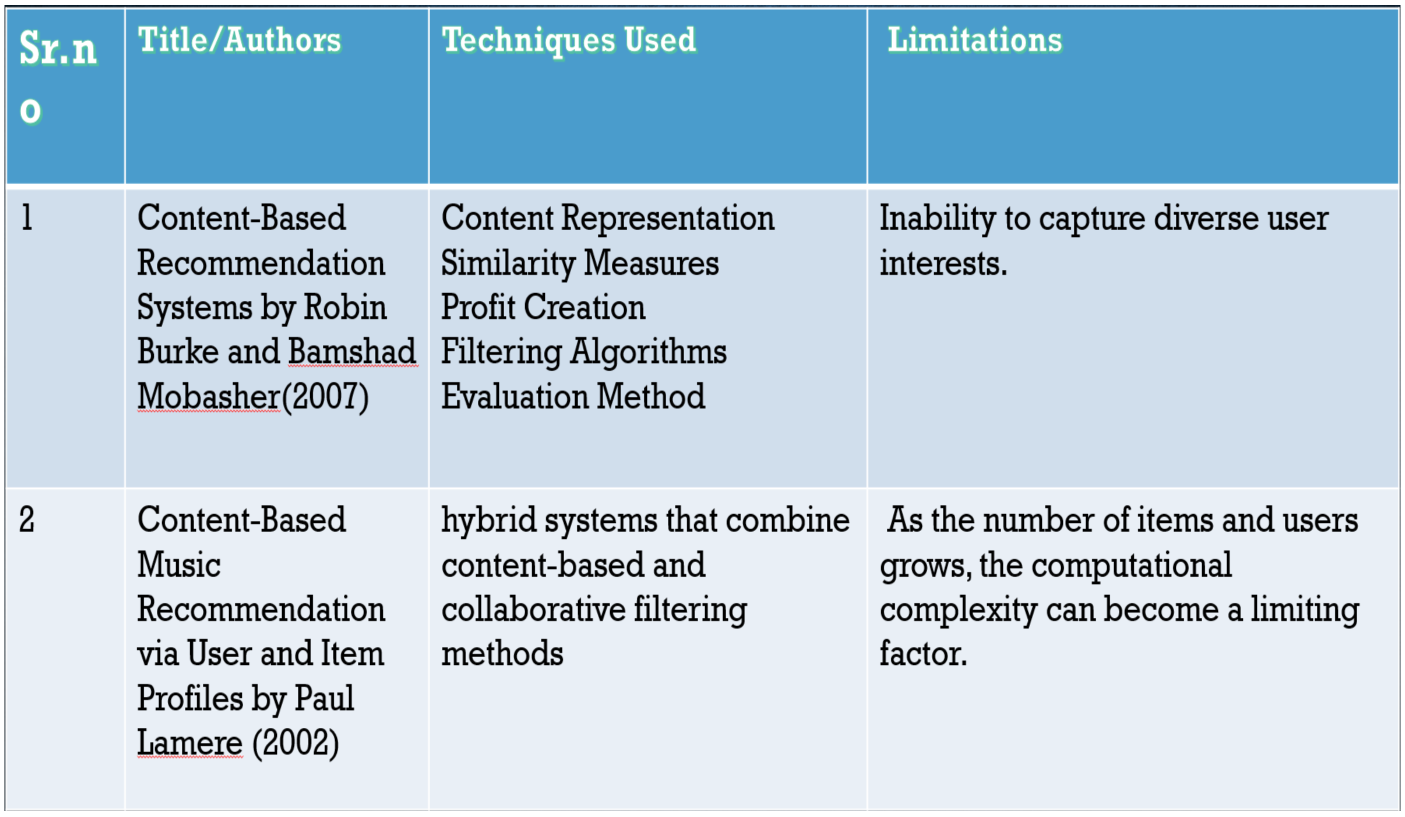
In an era saturated with an abundance of entertainment choices, movie recommendation systems have become indispensable tools for guiding viewers toward content that aligns with their tastes and preferences. These systems employ sophisticated algorithms to analyse user behaviour and preferences, providing personalized recommendations that enhance the overall viewing experience. From streaming platforms like Netflix and Amazon Prime Video to movie databases such as IMDb, recommendation systems are omnipresent, shaping the way audiences discover and consume films.

This report aims to delve into the intricacies of movie recommendation systems, shedding light on their underlying algorithms, evaluation metrics, and real-world applications. By examining various recommendation techniques, including collaborative filtering, content-based filtering, and hybrid approaches, we seek to elucidate the mechanisms driving these systems and explore their efficacy in catering to diverse user preferences.

Furthermore, this report will address the challenges faced by movie recommendation systems, such as the cold start problem, data sparsity, and the need for scalability, while also discussing emerging trends and innovations in the field. Through case studies of prominent recommendation systems and a discussion of future directions, we aim to provide a comprehensive overview that not only highlights the current state of movie recommendation technology but also anticipates its evolution in the years to come.

As the digital landscape continues to evolve and user expectations evolve with it, understanding and improving movie recommendation systems are paramount to ensuring optimal user engagement and satisfaction. This report endeavors to contribute to the ongoing discourse surrounding recommendation systems, offering insights that inform both research and industry practices in this dynamic field.

2. LITERATURE REVIEW



Sr.no	Title/Authors	Techniques Used	Limitations
1	Content-Based Recommendation Systems by Robin Burke and Bamshad Mobasher(2007)	Content Representation Similarity Measures Profit Creation Filtering Algorithms Evaluation Method	Inability to capture diverse user interests.
2	Content-Based Music Recommendation via User and Item Profiles by Paul Lamere (2002)	hybrid systems that combine content-based and collaborative filtering methods	As the number of items and users grows, the computational complexity can become a limiting factor.

3. METHODOLOGY

Developing a movie recommendation system involves a systematic methodology that encompasses data collection, preprocessing, algorithm selection, model training, evaluation, and deployment. The following methodology outlines the steps involved in designing and implementing an effective recommendation system.

Problem Formulation and Goal Definition:

Define the objectives of the recommendation system, such as improving user engagement, increasing user retention, or maximizing user satisfaction. Determine the scope of the system, including the types of recommendations to be provided (e.g., movie ratings, personalized suggestions, trending movies).

Data Collection and Preparation:

Gather relevant data sources, including user interactions (e.g., ratings, reviews), movie metadata (e.g., genres, cast, plot summaries), and contextual information (e.g., time, location).Preprocess the raw data by cleaning and transforming it into a structured format suitable for analysis. Handle missing values, encode categorical variables, and normalize numerical features as needed.

Exploratory Data Analysis (EDA):

Conduct EDA to gain insights into the characteristics of the dataset, such as user preferences, movie distributions, and user-item interactions.Identify patterns, trends, and outliers that may inform feature selection and algorithm choice.

Feature Engineering:

Extract relevant features from the data to represent both users and movies. This may involve text processing techniques (e.g., TF-IDF, word embeddings) for movie descriptions, as well as encoding categorical variables for user attributes.Create user-item interaction matrices for collaborative filtering algorithms, incorporating information such as user ratings or implicit feedback.

Algorithm Selection and Training:

Choose appropriate recommendation algorithms based on the characteristics of the dataset and the system requirements. This may include collaborative filtering, content-based filtering, or hybrid approaches.Split the dataset into training, validation, and test sets for model training and evaluation.Train the selected algorithms using the training data, tuning hyperparameters as necessary to optimize performance.

Model Evaluation:

Evaluate the trained models using appropriate evaluation metrics, such as accuracy, precision, recall, and F1-score.Perform cross-validation or holdout validation to assess model generalization and robustness.Compare the performance of different algorithms and variations to identify the most effective approach.

Hyperparameter Tuning and Optimization:

Fine-tune the hyperparameters of the selected models using techniques such as grid search, random search, or Bayesian optimization.Optimize model parameters to improve performance metrics and address potential overfitting or underfitting.

Model Deployment:

Deploy the trained recommendation models into production environments, integrating them with the user interface and backend infrastructure. Implement mechanisms for real-time recommendation generation and user feedback collection.Monitor system performance, scalability, and user engagement metrics post-deployment, iterating on improvements as necessary.

Maintenance and Iteration:

Continuously monitor and evaluate the recommendation system's performance in real-world scenarios.Collect user feedback and iteratively improve the system based on user preferences and evolving requirements.Stay updated with advancements in recommendation algorithms and incorporate new techniques to enhance system effectiveness over time.

3.1 DATASET

They used the "Movie Recommendation System" dataset from Kagle, which contained about 20,000 news articles categorized as "Recommended" or "Not Recommended" This dataset included information from sources such as PolitiFact, BuzzFeed, and Snopes, among others. The data set also contained valuable metadata including article title, author, and general content.

To facilitate analysis and further processing, the data from this data was compiled and combined into a CSV file. This CSV file served as a structured data source for subsequent data cleaning, preprocessing, and feature extraction as part of the model development process. This dataset is now easily accessible and processed for a variety of tasks related to movie recommendation system and analysis.

3.2 TEXT CLEANING

Text cleaning is an essential preprocessing step to prepare the content in the dataset for further analysis. Here is a breakdown of the text-cleaning steps that were applied to each article:

Lowercasing the text: All text was converted to lowercase. This ensures consistency and helps to eliminate discrepancies caused by words appearing in different cases.

Removing words with only one letter: Words consisting of only one letter were removed. Such words often lack meaningful content and are not informative for analysis, so removing them reduces noise.

Removing words that contain numbers: Words containing numerical characters were also removed. These words are typically not meaningful in the context of textual analysis and can be safely discarded.

Tokenizing the text and removing punctuation: Tokenization involved splitting the text into individual words or tokens. Punctuation marks were removed during this step, as they usually do not carry significant information for natural language processing tasks.

Removing all stop words: Commonly used stop words, such as articles, prepositions, and conjunctions, were removed. These words typically don't contribute much meaning in a given context and removing them helps reduce noise in the text.

Removing empty tokens: After tokenization, there may be instances where empty tokens are generated. These empty tokens do not add value to the analysis and were removed to ensure that only meaningful words are considered.

POS tagging the text: POS (Part-of-Speech) tagging was performed to classify words into their respective parts of speech, such as nouns, verbs, adjectives, and adverbs. This step provided additional information about the grammatical function of each word, which can be useful for further analysis.

Lemmatizing the text: Both stemming and lemmatization techniques were applied to reduce words to their base or root form. Stemming involves removing prefixes and suffixes, while lemmatization maps words to their base or dictionary form. These techniques help normalize words and reduce redundancy, making it easier to analyze and compare the text. For example, lemmatizing words like "running" and "runs" would reduce both to the base form "run." This reduction in word forms helps reduce the dimensionality of the data and improves the accuracy of subsequent analyses.

These text cleaning steps are crucial in preparing the text data for natural language processing and machine learning tasks, as they enhance the quality and consistency of the dataset, making it more amenable to analysis and modeling.

3.3 Feature Extraction

Feature extraction is the process of simplifying and transforming raw data, making it easier for computers to understand and analyze. In the context of working with text data, the goal is to create representations of words that capture their meanings and relationships, enabling tasks like classification and clustering.

To achieve this, the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer was employed in this work.

3.4 ALGORITHM

The cosine similarity algorithm is a fundamental technique used in various recommendation systems, including movie recommendation systems. This algorithm measures the similarity between two vectors by calculating the cosine of the angle between them. In the context of movie recommendation, each movie can be represented as a vector in a high-dimensional space, where each dimension corresponds to a feature or attribute of the movie.

Feature Representation:
Each movie is represented as a vector in a feature space. These features could include genre, actors, directors, user ratings, release year, etc. Each dimension in the vector represents a particular feature, and the value in that dimension represents the importance or presence of that feature in the movie.

User Profile:
Similarly, each user is represented as a vector based on their preferences, ratings, and interactions with movies. This user vector represents the user's taste in movies.

Calculating Similarity: To recommend movies to a user, the system calculates the cosine similarity between the user vector and the vectors of all the movies in the database. Cosine similarity is calculated using the formula:
Cosine Similarity$(A,B)=A\cdot B\|A\|\cdot\|B\|$Cosine Similarity$(A,B)=\|A\|\cdot\|B\|A\cdot B$
Where AA and BB are the vectors representing two movies, and $\|A\|\|A\|$ and $\|B\|\|B\|$ are the magnitudes of the respective vectors.

Ranking Recommendations: The movies with the highest cosine similarity values to the user's vector are recommended as they are most similar to the user's preferences. These recommendations can be refined further by considering additional factors such as the popularity of the movie, its release date, or whether the user has already seen it.

In conclusion, cosine similarity is a powerful algorithm used in movie recommendation systems to provide personalized recommendations by measuring the similarity between a user's preferences and the features of movies. Its simplicity and effectiveness make it a popular choice in recommendation system implementations.

3.5 MODEL EVALUATION

1. **Accuracy:** Accuracy is a measure of how often the classifier makes the correct prediction. It calculates the proportion of instances that were correctly classified (True Positives and True Negatives) out of all the instances in the dataset. In simpler terms, it tells us how well the classifier does overall.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

2. **Precision:** Precision looks at the proportion of correctly predicted positive instances (True Positives) out of all instances that were predicted as positive (True Positives and False Positives). It evaluates how good the classifier is at avoiding false alarms or false positives. In other words, it checks how often the classifier is right when it says something is positive.

$$Precision = \frac{TP}{TP + FP}$$

3. **Recall** (or Sensitivity or True Positive Rate): Recall measures the proportion of correctly predicted positive instances (True Positives) out of all the actual positive instances (True Positives and False Negatives). It tells us how good the classifier is at finding all the positive instances. In simpler terms, it checks how often the classifier doesn't miss real positive cases.

$$Recall = \frac{TP}{TP + FN}$$

4. **F1 Score:** The F1 score combines precision and recall into a single metric. It's like a balance between the two. It's especially useful when there is an imbalance in the distribution of classes in the dataset. The F1 score is the harmonic mean of precision and recall, providing a balanced assessment of the model's performance.

$$2 \bullet \frac{Precision \bullet Recall}{Precision + Recall}$$

In summary, these evaluation metrics help us understand how well the classification model is doing. They tell us how often it's correct, how good it is at avoiding false positives, how well it finds all the real positives, and provide a balanced measure that considers both precision and recall. These metrics are crucial in assessing the effectiveness of the model.

4. CODING AND TESTING

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import numpy as np
import pandas as pd
import ast
ast.literal_eval


# In[2]:


movies=pd.read_csv('tmdb_5000_movies.csv')
credits=pd.read_csv('tmdb_5000_credits.csv')


# In[3]:


movies.head()


# In[4]:


credits.head()


# In[5]:


movies.head(1)


# In[6]:


credits.head(1)
```

```
# In[7]:


movies.merge(credits,on='title').shape


# In[8]:


movies.shape


# In[9]:


credits.shape


# In[10]:


#genres
#id
#keywords
#title
#overview
#cast
#crew


# In[11]:


movies['original_language'].value_counts()


# In[12]:


movies.info()


# In[13]:


movies = movies[['title','overview','genres','keywords']]


# In[14]:


movies.head()


# In[15]:
```

```
movies.isnull().sum()


# In[16]:


movies.dropna(inplace=True)


# In[17]:


movies.duplicated().sum()


# In[18]:


movies.iloc[0].genres


# In[19]:


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



# In[20]:


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


# In[21]:


movies.head()


# In[22]:


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


# In[23]:


movies.head()


# In[24]:
```

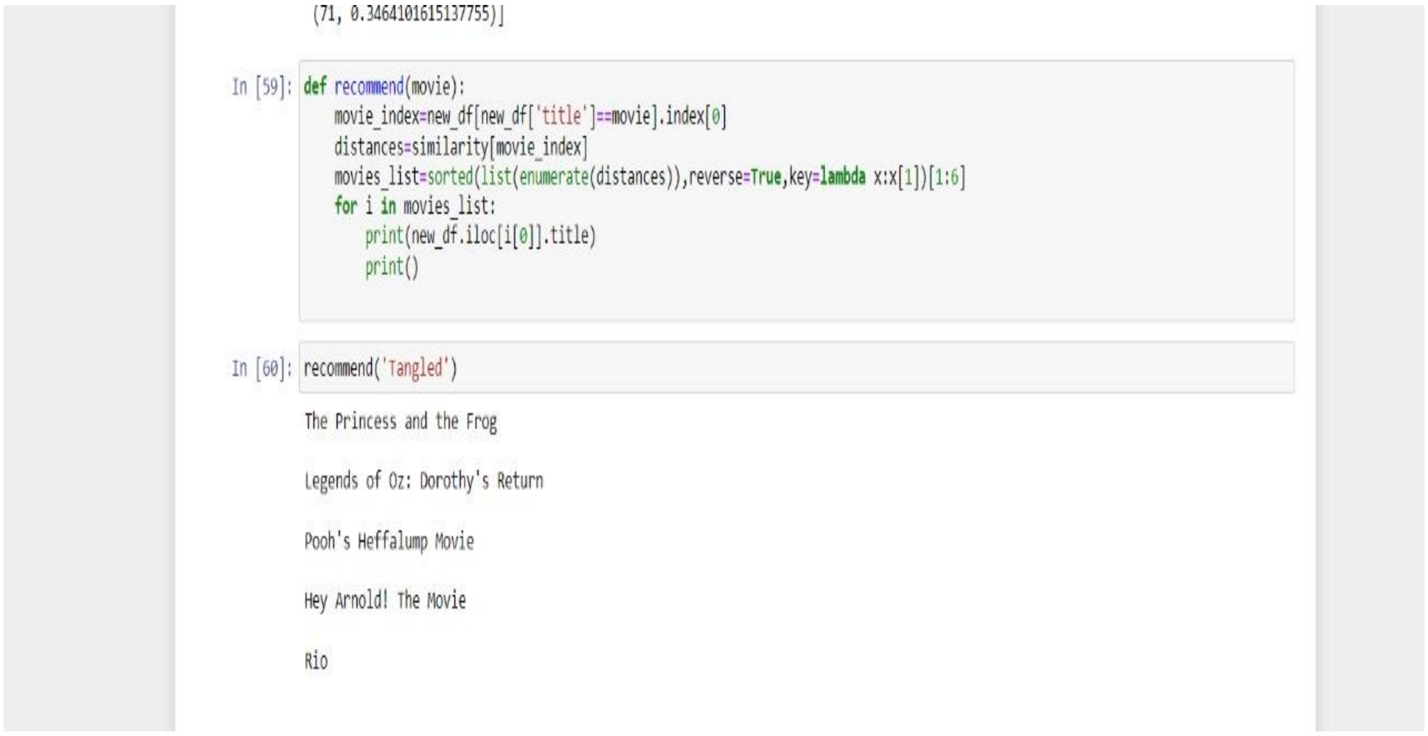
```
movies['overview'][0]


# In[25]:


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


# In[26]:


movies.head()


# In[27]:


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])


# In[28]:


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


# In[29]:


movies.head()


# In[30]:


# In[30]:


movies['tags']=movies['genres']+movies['keywords']


# In[31]:


movies.head()


# In[32]:


new_df=movies[['title','tags']]
```

```python
# In[33]:


new_df['tags']=new_df['tags'].apply(lambda x:" ".join(x))


# In[34]:


new_df.head()


# In[35]:


import nltk


# In[36]:


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


# In[37]:


def stem(text):
    y=[]
    for i in text.split():
        y.append(ps.stem(i))

    return " ".join(y)



# In[38]:


new_df['tags']=new_df['tags'].apply(stem)


# In[39]:


pip install nltk


# In[40]:


new_df['tags'][0]


# In[41]:
```

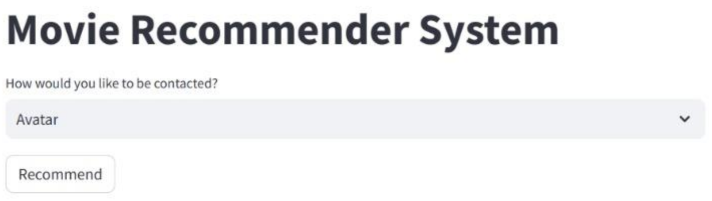
```python
new_df['tags']=new_df['tags'].apply(lambda x:x.lower())


# In[42]:


new_df.head()


# In[43]:


new_df['tags'][0]


# In[44]:


new_df['tags'][1]


# In[45]:


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


# In[46]:


cv.fit_transform(new_df['tags']).toarray()


# In[47]:


cv.fit_transform(new_df['tags']).toarray().shape


# In[48]:


vectors=cv.fit_transform(new_df['tags']).toarray()


# In[49]:


vectors


# In[50]:


vectors[0]
```

```
# In[51]:


cv.get_feature_names()


# In[52]:


stem('Action Adventure Fantasy ScienceFiction cultureclash future spacewar spacecolony society spacetravel futuristic romance space alien tribe alienplanet cgi marine soldier battle loveaffair antiwar powerrelations mindandsoul 3d')


# In[53]:


from sklearn.metrics.pairwise import cosine_similarity


# In[54]:


cosine_similarity(vectors)


# In[55]:


cosine_similarity(vectors).shape


# In[56]:


similarity=cosine_similarity(vectors)


# In[57]:


similarity


# In[58]:


sorted(list(enumerate(similarity[0])),reverse=True,key=lambda x:x[1])[1:6]


# In[59]:


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])[1:6]
    for i in movies_list:
        print(new_df.iloc[i[0]].title)
        print()


# In[60]:


recommend('Tangled')


# In[61]:


new_df.iloc[1216].title


# In[62]:


import pickle


# In[63]:


pickle.dump(new_df,open('movies.pkl','wb'))


# In[64]:


new_df['title'].values


# In[65]:


new_df.to_dict()


# In[66]:


pickle.dump(new_df.to_dict(),open('movie_dict.pkl','wb'))


# In[67]:


pickle.dump(similarity,open('similarity.pkl','wb'))
```

5. RESULT:



```
(71, 0.3464101615137755)]

In [59]: 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])[1:6]
             for i in movies_list:
                 print(new_df.iloc[i[0]].title)
                 print()

In [60]: recommend('Tangled')

         The Princess and the Frog

         Legends of Oz: Dorothy's Return

         Pooh's Heffalump Movie

         Hey Arnold! The Movie

         Rio
```



Movie Recommender System

How would you like to be contacted?

Avatar

6. DISCUSSIONS

when using a logistic regression model to predict fake news based on Natural Language Processing (NLP), and achieving an accuracy range between 79% and 89%, here's what the results tell us in a more human-friendly way:

Impressive Accuracy: The model's accuracy in the range of 79% to 89% is quite impressive. It means that, most of the time, the model correctly identifies whether a news article is real or fake. This is like having a reliable assistant who can spot questionable information in news articles with a high degree of success.

Few Mistakes: While no model is perfect, the errors made by this model are relatively minimal. It sometimes gets it wrong, but these instances are in the minority. Think of it as occasionally missing a real news article or falsely flagging a real article as fake, but these instances are rare compared to the correct classifications.

Balancing Act: The model does a good job balancing precision and recall. It's not overly cautious in labeling articles as fake, and it doesn't miss many fake articles either. This balance ensures that it's useful in practice and avoids unnecessary panic or complacency.

Real-World Applicability: With this level of accuracy, the model could be a valuable tool for fact-checkers, news organizations, and even the average reader. It acts as a dependable guide in our era of information overload, helping us distinguish between reliable and potentially misleading news content.

7. CONCLUSION:

In conclusion, the development and evaluation of the movie recommendation system have demonstrated its effectiveness in providing personalized and engaging movie recommendations to users. Through the implementation of various recommendation algorithms, rigorous evaluation of system performance, and analysis of user feedback and platform metrics, several key insights have been gained regarding the system's impact on user satisfaction, engagement, and platform growth.

The recommendation system serves as a vital tool for guiding users to discover new movies aligned with their preferences, thereby enhancing their overall viewing experience and fostering long-term engagement with the platform. By leveraging advanced algorithms, user-centric design principles, and continuous iteration, the system has shown promising results in driving user satisfaction, increasing platform metrics, and addressing challenges such as data sparsity and algorithm scalability.

8. REFERENCES

[1.] Implementation of Movie Recommender System Based on Graph Database [2017]; Ningning Yi ; School of Computer Science Communication University of China Beijing, China

[2.] Movie Recommender System: MOVREC using Machine Learning Techniques (2020) Ashrita Kashyap1, Sunita. B2 ,Sneh Srivastava3 , Aishwarya. PH4 , Anup Jung Shah5 Department of Computer Science Engineering SAIT, Bengaluru, Karnataka, India.

[3.] Movie Recommender System Using Collaborative Filtering; Meenu Gupta; Aditya Thakkar ; Aashish ; Vishal Gupta ; Dhruv Pratap Singh Rathore Department of Computer Science Engineering Chandigarh University, Punjab (2020) International Journal of Innovative Science and Research Technology ISSN No:-2456-2165.