Netflix Recommendation System

Your Name

TABLE OF CONTENTS

ABSTRACT

The recommendation system has become an essential part of many online platforms, including movie and TV streaming services like Netflix. Recommender systems refer to systems that suggest items to consumers based on various criteria. These systems predict the movies that are most probable for consumers to have an interest in. Recommender services are utilized by Netflix to help customers find appropriate movies or items.

The purpose of this project is to develop a recommendation system for Netflix that can suggest new movies and TV shows to users. The system uses several content-based filtering techniques such as Basic Clustering and K-Means Clustering, KNN, Naïve Bayes, Decision Tree and Logistic Regression and to identify users' preferences and generate personalized recommendations. Then, describe the implementation and functionality of the proposed system, specifically in relation to a movie recommender system. Different evaluation measures are employed to assess the accuracy of the implemented recommendation system. The findings indicate that some of these systems are the same with Netflix's existing recommendation system in both accuracy and personalization.

Keywords:

Recommender System, Content-Based Filtering, K-Means, KNN, Movie recommendation

INTRODUCTION

The art of storytelling has always been deeply ingrained in human nature. Throughout history, major technological advancements have transformed the way stories are told, resulting in more immersive and attractive stories. From the days of our ancestors gathering around a fire in a cave to modern-day digital platforms, such as television and the internet, storytelling has continued to evolve. With the advent of the internet, storytelling has been revolutionized once again, providing new and innovative ways to engage audiences worldwide. The impact of the internet on storytelling is profound, and its influence is expected to continue to grow in the coming years.

Netflix, a leading streaming service with over 200 million subscribers [1] worldwide, offers a vast collection of movies and TV shows. However, with so many options, it can be challenging for users to decide what to watch. To address this challenge, we have developed a recommendation system that utilizes machine learning algorithms to analyze user data and provide personalized recommendations.

Netflix's recommendation system is a machine learning-based system that provides personalized content recommendations to its users. The aim of the system is to improve the user experience by making it easier for users to discover new content and increasing the probability that they will find something they enjoy watching.

Research Question

This research aims to explore the performance of various recommendation algorithms on Netflix's dataset, and investigate how they can be modified to better assist users with different factors and how can different recommendation algorithms be optimized to improve the user experience on Netflix?

Research Challenges

Representing movie content effectively is one of the significant research challenges in content-based recommendation systems. The challenge involves developing techniques to extract relevant features from movie data that can be used for accurate recommendations. Another research challenge in content-based systems is the cold-start problem [2]. This challenge arises when there is little or no historical data available for new users or movies. This issue is particularly challenging in content-based systems as it relies on historical data to identify similarities between movies. Another challenge for content-based recommendation systems is the risk of over-specialization. If the system only recommends movies that are similar to those the user has already watched, it may not expose users to new or diverse content, limiting their options.

LITERATURE REVIEW

The effectiveness of a recommendation system largely relies on the user's input and the correlation between the products and Content based [3], [4] collaborative [5] and hybrid [6] filtering are the three different approaches.

A. Content-based Filtering

Recommendation systems rely on product similarity and work best when product properties are easily identified. For movie recommendations, a content-based system creates a user profile using provided data and employs Term Frequency and Inverse Document Frequency concepts [7] to determine the importance of a movie based on word frequency in a collection of documents.

B. Collaborative Filtering

Collaborative filtering is a technique used by recommendation systems to provide users with personalized suggestions. The two main types are memory-based and model-based. Memory-based filtering uses the similarity between users or items to generate recommendations, while model-based filtering uses unsupervised learning methods to identify user preferences and item attributes. Both types use similarity matrices to measure similarity.

C. Hybrid Filtering

A hybrid recommendation system is a combination of two or more recommendation algorithms to provide more accurate and diverse recommendations to users [8]. It combines the advantages of different recommendation methods to overcome their limitations and improve the overall performance of the system.

In this paper, we proposed a recommended system based on content filtering with several techniques.

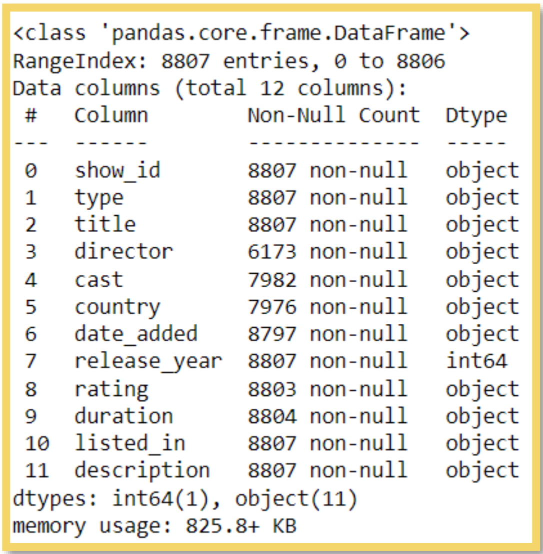
DATA EXPLORATION AND PREPARATION

The recommendation engine model that utilizes content-based filtering operates by analyzing factors such as cast, genre, director, and descriptions of movies or TV shows. Following the user's input of a movie or TV show that they enjoyed, the engine must search the database and present the user with a list of 5 recommended movies. The aim is to provide the user with the top five movies based on their previous viewing history.

A. About dataset

This paper uses the Netflix movies and TV shows dataset available in the Kaggle [8]. The dataset includes more than 8,800 records and 12 columns, including:

- General info: ID, title, type, director, country, cast, description, duration.
- Date fields: the release year and the added date to the catalog.
- Categorization: rating and genre in which the show is listed.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

Figure 1

B. Data Pre-processing

- **Data Cleaning:** Data pre-processing is a method used in data mining which involves converting unrefined data that is disorganized, inconsistent, and could have diverse errors. Data pre-processing is necessary to address these problems [9]. The idea is to build a recommendation engine based on content filtering. The text data needs to be converted and used as numbers with machine learning tools. First, the data should be cleaned up and made more suitable to find similar movies or TV shows. From the dataset, "duration" and "data_added" columns are dropped because they are not needed. Then, missing values in the "rating" column are filled with correct values obtained from IMDb and Netflix, and the values in this column are also changed to terms that are easy to understand. "cast", "director", and "country" columns contain NaN values which are filled with "Unknown".

- **Bag Of Words:** BoW, which stands for "Bag of Words," is a popular technique used in feature selection and classification. This method is currently popular due to its ability to effectively select and classify features by creating bags for each type of instance. Due to its versatile nature, BoW is applicable in various domains. Its key strengths lie in its ability to classify and categorize texts, and to compute weights for individual words. These weights correspond to the frequency of occurrence for each word within the text [10].
  In the project a new column called "bag_of_words" was populated with strings that are derived from four other columns in the dataframe: "director", "cast", "listed_in", and "description". The codes applied a series of transformations to each one. The text in each column is converted to lowercase, all spaces are removed, and the resulting string is split into a list of words. Next, the code created a list of strings by concatenating the first three words from each of the four columns of interest. These strings then joined together with spaces to form a single string, which is assigned to the "bag_of_words" column for that row.

- **CountVectorizer:** CountVectorizer is a feature extraction technique in natural language processing (NLP) that transforms a given text corpus into a matrix of word counts. It converts a collection of text documents to a matrix of token counts, where each row represents a document and each column represents a word in the corpus vocabulary. In the project after computing the frequency of each word or term in a document, we assigned a weight to words based on their frequency. For instance, let k_i be the unique terms in document d_j, and let c_{ij} represent their respective counts. Then, the content of document d+ can be defined as:

$$Content(d_j) = \{n_{1j}, n_{2j}, ...\}$$

  In this way, CountVectorizer represents each document as a bag-of-words model, where the order of the words is ignored and only their frequency is considered. The resulting matrix can then be used as input to machine learning algorithms for classification, clustering

C. Exploratory Data Analysis

Netflix is currently the most widely used platform for watching movies. By analysing the Netflix dataset and after pre-processing, it has been revealed that the platform offers a total of 8797 movies and TV shows, of which 6131 are movies and 2666 are TV shows. The distribution type of these movies and TV shows can be visualized in Figure 2 using Python.

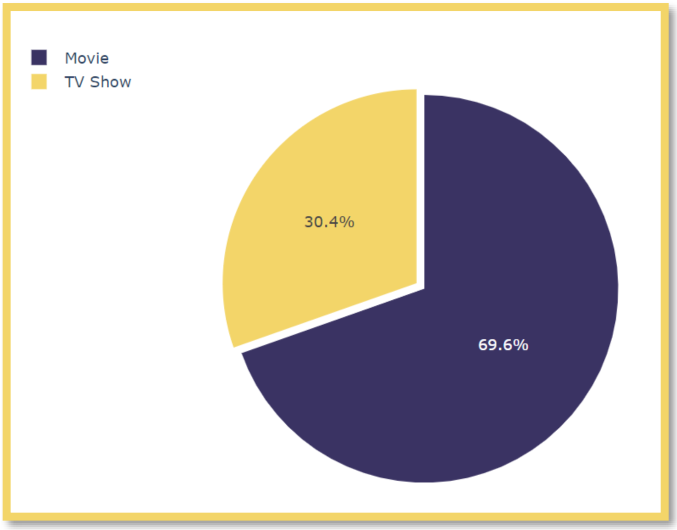


Figure 2

The movie category represents 69.6% of the total content, while the TV show category represents 30.4%. The movies contents are most popular on Netflix.

The word cloud in Figure 3 shows the most commonly occurring words in the titles which in this case are Love, Girl, Man, Life. It's noteworthy that numerous movies have similar keywords in their titles.

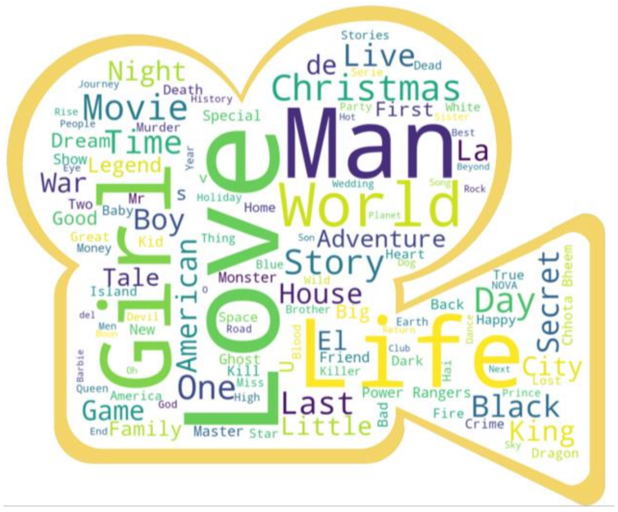


Figure 3

Figure 4 shows the chart would help you understand the relationship between the countries and the relative number of movie contents produced by each. As you can see, The United States produces approximately 2700 movie contents, which is a significantly larger number than the movie contents produced by other countries.

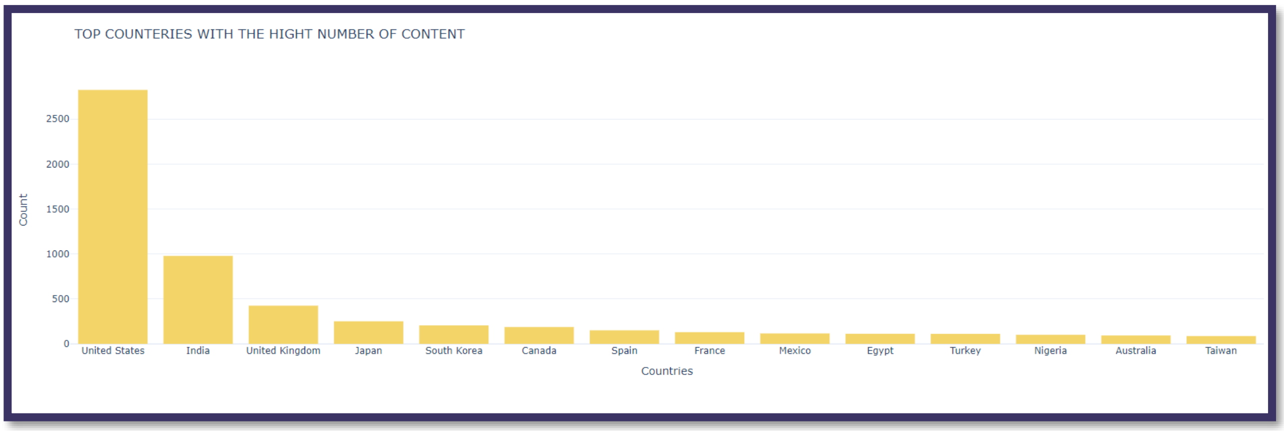


Figure 4

The chart in Figure 5 displays that there is a strong association between independent movies and dramas. In other words, independent movies are much more likely to be in the drama genre. Additionally, the chart reveals that there are very few international movies in the children's genre.

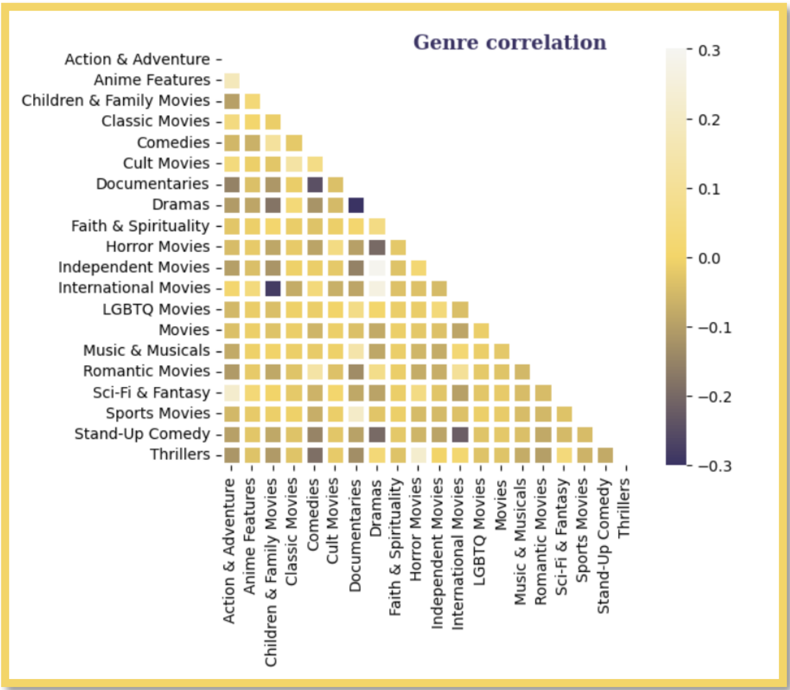


Figure 5

MODEL AND ALGORITHM SELECTION

A recommendation system (RS) is a type of Decision Support System (DSS) that provides item recommendations by analysing and consolidating information about the user's preferences [11]. The system assists users in making decisions by offering useful information that considers their priorities and concerns [12]. Recommendation systems (RSs) are widely used in popular e-commerce websites like Amazon, Netflix, Pandora, etc. These RSs suggest various types of items to users, including news, articles, people, URLs, and more. They play a significant role in enhancing the user experience [13]. The project worked on Netflix movie recommendation system and the diagram presented below depicts the pipeline representation of the methodology that we have developed.

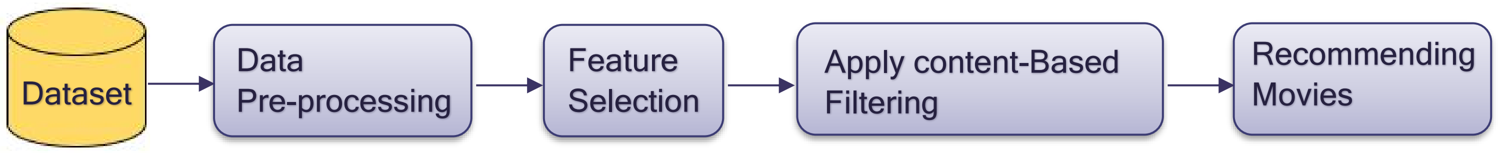


Figure 6

We also used Python to develop our methods for recommendation system.

Content-Based Filtering

The Content-based filtering (CB) aims to suggest movies or items that are similar to movies the user has previously watched. CB offers recommendations based on the information from the products [14], such as the movie title, year, actors, and genre, rather than just relying on similarity by rating.

To apply this approach, it is crucial to have data that describes each item, and having a user profile that details their preferences is also helpful. The objective is to understand the user's preferences and then suggest items that are comparable to their preferences [15].

Additionally, in situations where a user is new to a platform and we have no prior information on them, this recommender system is used to find similar items by analyzing contents. We use clustering models for our recommendation system.

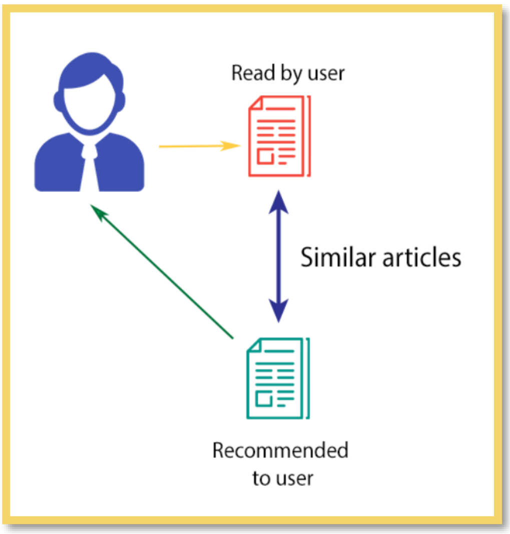


Figure 7

The main intent of a content-based recommender system is to measure the similarity between items. One of the most popular methods for modelling items in unlabelled data is clustering.

Algorithm 1: Basic Clustering

As we have utilized the CountVectorizer, computing the dot product will give us the cosine similarity score. Cosine similarity score is often used to determine the similarity between two documents based on their word frequency vectors and it is relatively easy and fast to calculate.

$$similarity = \cos(\theta) = \frac{A \cdot B}{||A||||B||} = \frac{\sum_{n}^{i=1} A_i \times B_i}{\sqrt{\sum_{n}^{i=1} (A_i)^2} \times \sqrt{\sum_{n}^{i=1} (B_i)^2}}$$

The values are graded from -1 to 1, with -1 indicating complete dissimilarity and 1 indicating complete similarity. To determine the similarity between movies, the results produced by CountVectorizer are utilized, and the cosine similarity score is obtained by computing the dot product.

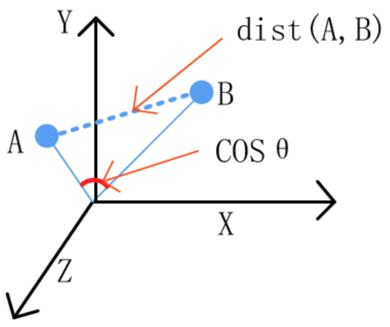
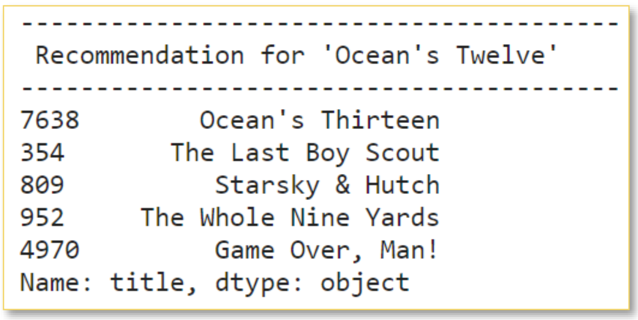


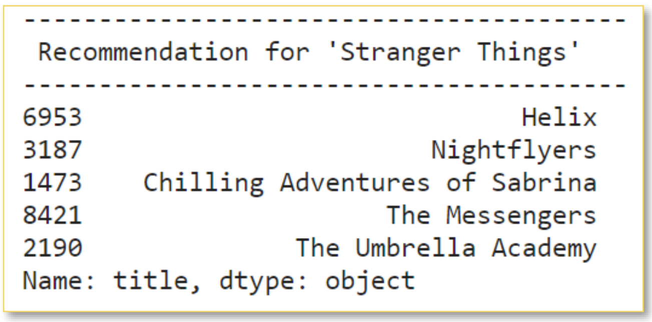
Figure 8

As Figure 8 shows, the smaller angle between the feature vectors signifies a higher cosine similarity, which implies that the two movies are more alike.

To optimize the system's performance speed, we will replace the use of cosine_similarities() with sklearn's linear_kernel(). This will provide a faster computation time for generating similarity scores [16].



```
----------------------------------------
 Recommendation for 'Ocean's Twelve'
----------------------------------------
7638         Ocean's Thirteen
354         The Last Boy Scout
809            Starsky & Hutch
952       The Whole Nine Yards
4970           Game Over, Man!
Name: title, dtype: object
```



```
---------------------------------------
 Recommendation for 'Stranger Things'
---------------------------------------
6953                              Helix
3187                        Nightflyers
1473     Chilling Adventures of Sabrina
8421                     The Messengers
2190               The Umbrella Academy
Name: title, dtype: object
```

Figure 9

Algorithm 2: K-Mean clustering

The method involves the utilization of CountVectorizer for converting text into a vector of token frequencies, while cosine_similarity is applied as the similarity matrix metric. Additionally, the closeness of nodes is determined through the use of Adamic Adar measure, which takes into account the shared neighbors.

Adamic Adar Measure

Adamic Adar Measure is employed to exhibit the shared elements among movies, such as clusters, countries, persons, and categories, in a graphical representation. This measure determines the closeness between nodes based on their mutual neighbors. For two nodes, x and y (representing two movies), the function N (one_node) is used to retrieve the set of neighboring nodes to one_node.

$$A(x,y) = \sum_{u \in N(x) \cap N(y)} \frac{1}{\log(N(u))}$$

When x and y have a common node u that is connected to numerous neighboring nodes, this node is irrelevant. If N(u) is high → 1/log(N(u)) is not high.
if x and y share a node u that is not connected to many neighboring nodes, this node is deemed relevant. In such a scenario, If N(u) is not high → 1/log(N(u)) is higher.
The Adamic Adar Measure calculates the total inverse logarithmic degree centrality of shared neighbors between two nodes that have a common set of adjacent nodes. The measure is based on the idea that common elements become less important when predicting a connection between two nodes if the neighborhood is very large. On the other hand, when a small number of nodes share common elements, they become more significant. A value of "0" suggests that two nodes are not closely related, while higher values indicate that the nodes are more closely related [17].

K-Mean Algorithm

In the second method we use k-mean clustering that is the most important algorithm in data mining [18]. K-means clustering is a type of learning method that is both unsupervised and semi-parametric. Its main function is to group data, with each group represented by their centers. These groups are typically referred to as "classes" in classification. When new data is inputted, it is assigned to the closest center in the group. In the realm of recommendation systems, K-means clustering is often used to group users or items together. Once the system has identified a group, it provides recommendations that are tailored to the specific characteristics of that group. The algorithm for K-means clustering is described below [19].

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \, ||\boldsymbol{x}_n - \boldsymbol{m}_n||^2$$

- N is the total number of data points,
- K is the number of clusters
- x_n is the vector of measurement n
- m_k is the mean for cluster k
- r_{nk} is an indicator variable that indicates whether to assign x_n to k

Edge weights assignment

It is not possible to calculate the weights of all movies in the Netflix dataset with absolute certainty. Nevertheless, for the purposes of opinion propagation, the resource allocation method can be used to approximately determine the edge weights of movies. The edge weights are assigned to each connection between two nodes, based on the number of common neighbors present and the degree of the nodes. The weight, indicated by w(x, y), in the edge weight between nodes x and y is determined using the resource allocation method.

$$w(x, y) = 1 + \sum \frac{1}{N(u)}$$

Where

$$u \in x \cap y$$

If there is only one path between nodes x and y, u is assigned a value of zero, indicating that there is no common node between x and y, and the weight of the link is set to 1. The higher the edge weight, the more similar the movies are considered to be. This is similar to the strategy used by operating systems to allocate resources to user programs.

Table 1 displays the five movies that the model suggests for the “Ocean's Twelve”.

Recommendation for ‘Stranger Things’	
Movies	**corresponding weights**
Ocean's Thirteen	6.672296
The Man Who Feels No Pain	3.655610
Up in the Air	2.533039
Hail, Caesar!	2.533039
Good Night, and Good Luck	2.398322

Table 1. The corresponding weights between ' Ocean's Twelve ' and five movies

A graph is created by connecting a set of nodes with links between them. Figure 10 displays the connections among actor, categories, director, country, cluster, and Countvectorizer using cosine similarity based on the features like actors, directors, genre and description. The recommendation function retrieves all the neighbor nodes for a given movie and collects all the related movies into a dictionary. The movies are then ordered based on their weights, which are determined by the number of neighbors each node has. The top five related movies are recommended to the user who searched for the initial movie.

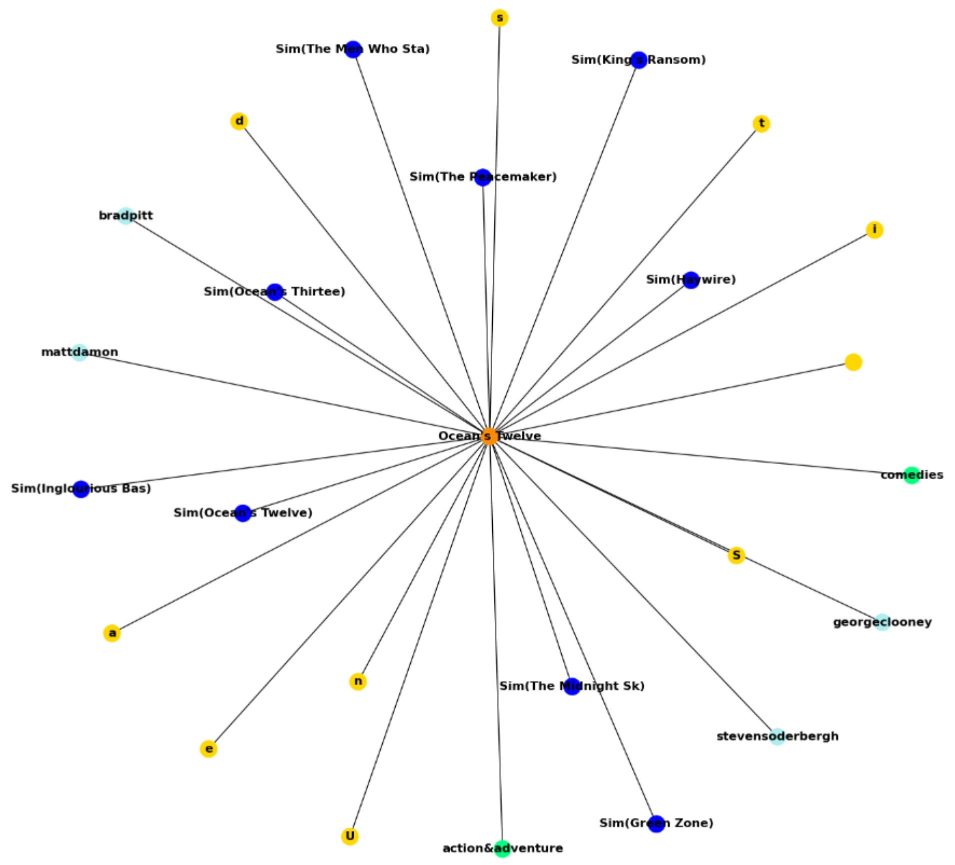


Figure 10. The relation between “Ocean's Twelve” and other fields such as categories, actor, director, country, and similarity

The purpose of the recommendation feature is to allow users to explore the surrounding area of their desired movie. This area includes details on the movie's actors, directors, countries, and categories. Additionally, the feature enables users to explore the neighborhood of each adjacent movie, uncovering movies that share similar attributes with the target film. The end result is the calculation of the Adamic Adar measure.

The findings are presented using a Python sub-graph. The figure 11 shows the recommendations made by the model after the user watches "Ocean's Twelve" and it can be observed that the " Ocean's Thirteen " movie is identified by the algorithm as the next most similar movie. It also displays the closeness between the movies.

The recommendation model is better since it analyses more fields from the dataset, such as similarity based on description, categories, actors and directors, clusters, and countries, before recommending the five movies to the user. After that the model analyses the neighborhood of the target movies and then explores the neighborhood of each neighbor to discover the movies that share a node with the target field and the resulting final calculation by Adamic Adar measure [20].

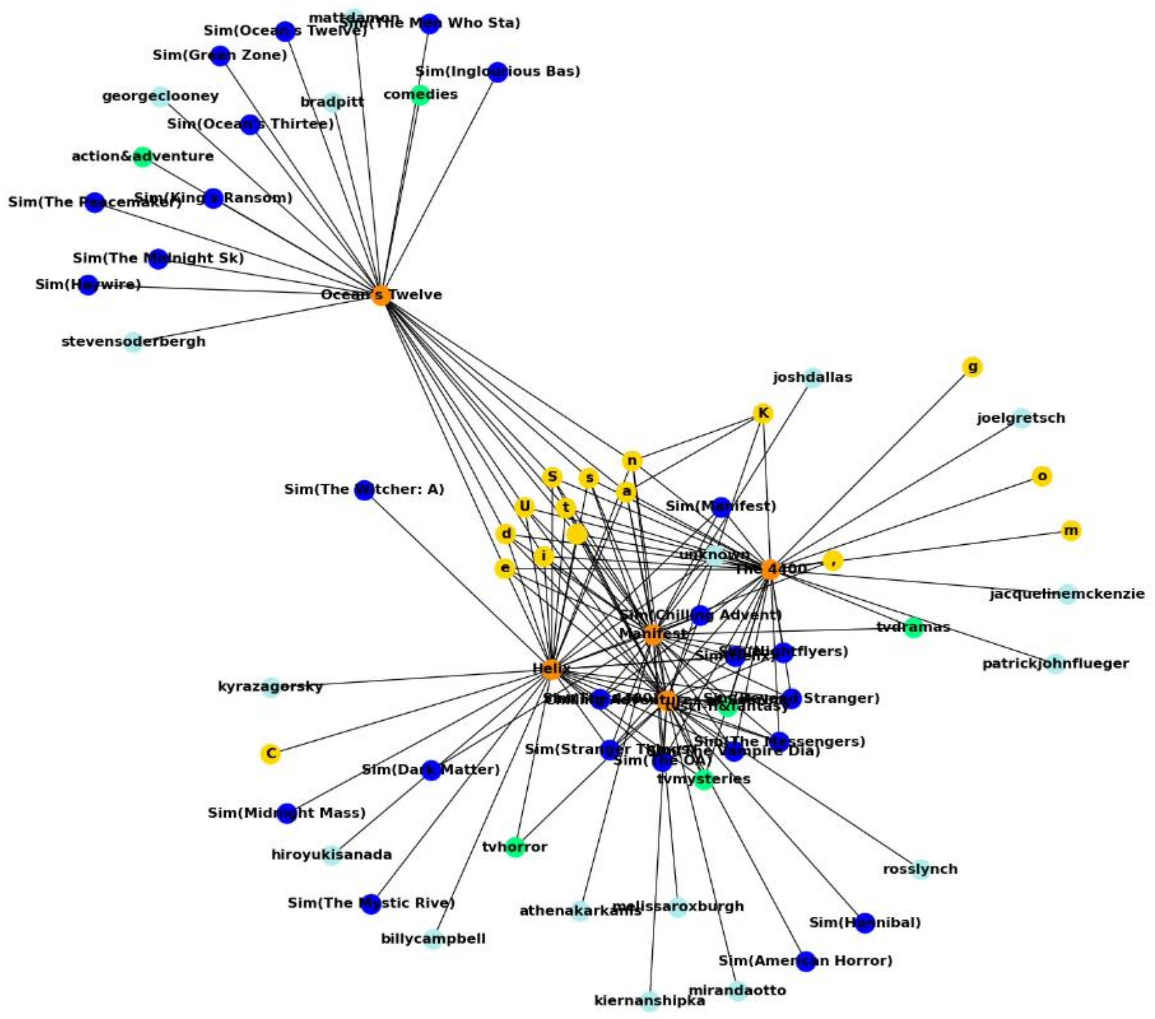


Figure 11, Top recommendations for Ocean's Twelve

The figure 12 illustrates the recommendations for " Stranger Things"

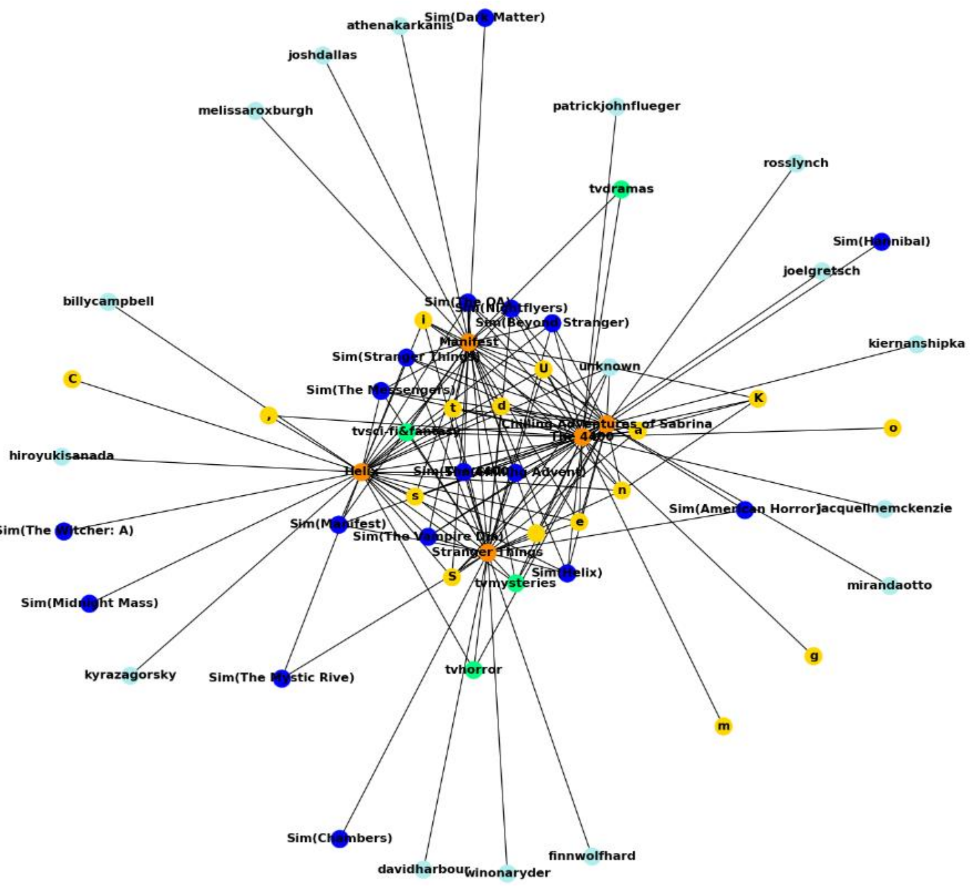


Figure 12, Top recommendations for Stranger Things

Algorithm 3: K-Nearest Neighbour (KNN)

After converting categorical variables to numerical values and transforming the text data into a format that can be used in machine learning models, we use the bag-of-words technique to represent each movie features word as a feature of the dataset [21]. This is done using the CountVectorizer class from the scikit-learn library.

In this algorithm the K Nearest Neighbour (K-NN) is applied to obtain movie recommendations. K-NN uses the K closest examples in the training data and finds the top N recommendations based on similarities in the feature space [22].

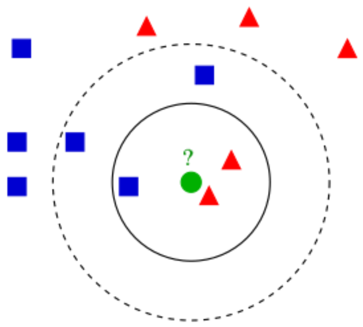


Figure 13

The recommendation algorithm uses a combination of Euclidean distance and dissimilarity metrics to calculate the distance between movies. Euclidean distance measures the distance of two points [23]:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}.$$

In our code the Euclidean distance between movies is calculated based on their features such as type, rating, and country, with an additional weighting factor assigned to the difference in release year. The dissimilarity metric, based on the cosine similarity of the movie features, is also incorporated in the distance calculation. The calculation below represents our model unique distance metric:

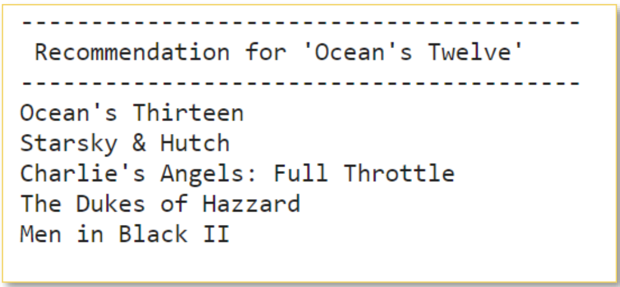
$$\text{distance} = \text{euclidean\_distance}_{type,country,rating}$$

$$+\alpha\ \text{euclidean\_distance}_{release\_year}$$

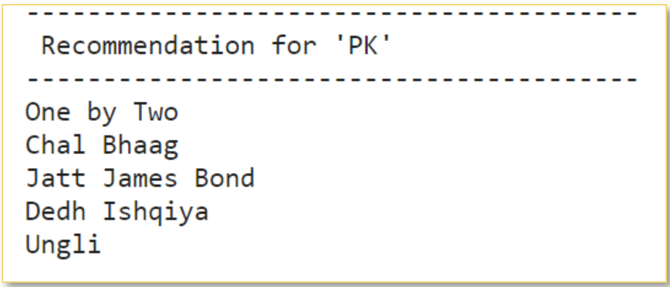
$$+\beta\ \text{cosine\_dissimilarity}_{bag\_of\_words}$$

We will reduce the weight of the release year column in the Euclidean distance calculation compared to the encoded type, country, and rating columns by multiplying it with a factor α, default value is 0.1. We can also adjust the impact of the bag of words feature that includes genre, cast, and director by using the parameter β, default value is 2.

The recommendation system is implemented as a function called 'get_recommendation_knn' that takes a movie title as input and outputs a list of recommended movies. The function first locates the index of the input movie title in the dataset and then calculates the distance between that movie and all other movies in the dataset using the total_distance function [24].



```
----------------------------------------
 Recommendation for 'Ocean's Twelve'
----------------------------------------
Ocean's Thirteen
Starsky & Hutch
Charlie's Angels: Full Throttle
The Dukes of Hazzard
Men in Black II
```



```
----------------------------------------
 Recommendation for 'PK'
----------------------------------------
One by Two
Chal Bhaag
Jatt James Bond
Dedh Ishqiya
Ungli
```

Figure 14

Algorithm 4: Decision Tree

The Decision Trees (DTs) was chosen as the classifier/model for the proposed RS due to its simplicity and interpretability, making it easy for decision-makers to understand the decision-making process. In addition, decision trees are suitable for addressing technical issues such as sparsity and scalability. A decision tree consists of nodes and leaves, with the first node being the root node, and the subsequent nodes being internal nodes that test a particular attribute to create a binary or multi split. The leaf nodes represent the output of the classification, which is the final decision for the instance from the test data. As Figure 15 shows, in order to suggest a movie to users, it is necessary to navigate through the decision tree from the root node to the leaf node.

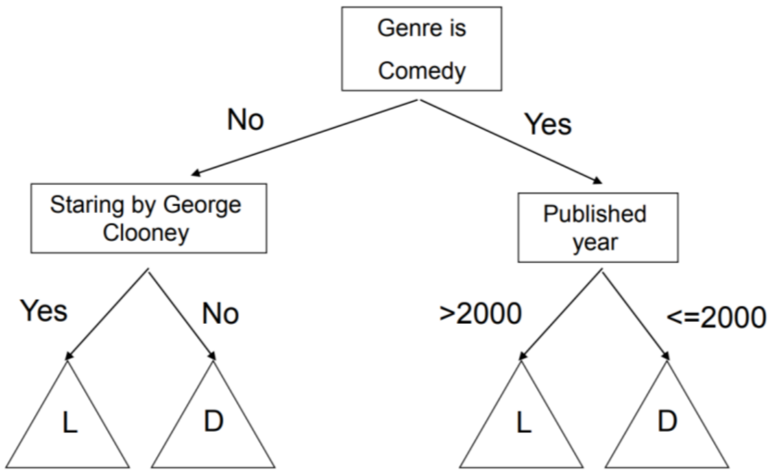


Figure 15

Decision Trees are a type of supervised learning method that uses decision rules derived from features to predict response values. In the Netflix movies and TV shows dataset, there is not any labeled data that is used to train the DTs algorithm, so we used the K-Mean Clustering's recommendation results as a labeled feature for DTs.
The objective of our code is to make predictions for cluster labels of movies by analysing their characteristics such as type, rating, country, and category. After determining the cluster of an input movie, the code recommends similar movies by choosing other movies in the same cluster or those with similar features. The categorical features are encoded using the LabelEncoder function and the dataset is then split into training and testing sets. The DecisionTreeClassifier is used to train the model on the training set and predict the cluster labels of movies in other clusters. The recommended movie list is a combination of movies in the same cluster as the input movie and top 10 similar movies from other clusters.

Algorithm 5: Logistic Regression

Logistic regression [25] is a popular type of generalized linear model that is commonly employed for probability prediction. The hypothesis for logistic regression can be denoted as $h_\theta(x)$, and is expressed as follows:

$$h_\theta(x) = g(\theta^T x)$$

where

$$g(z) = \frac{1}{1 + e^{-z}}$$

is named the logistic function or the sigmoid function [26].

Up until now, the logistic regression model has found widespread application in various recommendation systems. One instance of this is demonstrated by Bartz et al. from Yahoo!, who implemented logistic regression and collaborative filtering techniques to construct an advertising recommendation system [27]. A news recommendation system was established by Lau et al. from Stanford University through the utilization of logistic regression and Naive Bayes classifiers [28]. An approach centered around logistic regression was suggested by Quevedo et al. from the University of Oviedo as a means of tag recommendation [29]. In the project, we used logistic regression for the movie recommendation system, and like the decision tree algorithm that was applied before, the results of K-mean clustering were used to train the model. The recommendation function was used in this algorithm is same with the decision tree algorithm.

Algorithm 6: Naive Bayes Classifier

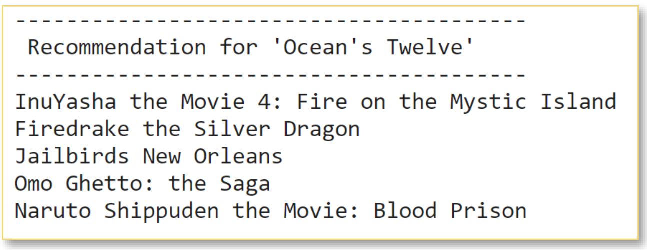
The Naive Bayes classifier is founded on the Bayes theorem and assumes strong (Naive) independence among the input features. This makes it particularly useful for scenarios that involve high-dimensional input data. The Bayes theorem is employed to determine the probability of a document 'd' belonging to a certain class 'C_j':

$$P(C_j|d) = \frac{P(C_j)P(d|C_j)}{P(d)},$$

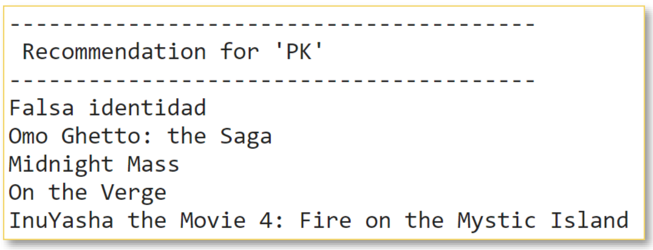
The terms 'posterior', 'prior', 'likelihood', and 'evidence' are used to represent P(C_j |d), P(C_j), P(d|C_j), and P(d) of this calculation, respectively.

In our approach, we set the process in three phases, as depicted in Figure 1 below:

- In the first phase, the labels obtained from the k-means clustering and the 'CountVectorizer' function is used to transform the 'bag_of_words' column into a matrix of token counts.
- In the second phase, the Naive Bayes classifier used as a classification and recommendation model.
- In the third phase, the function 'get_recommendation_nb' is created to take in two input arguments, 'title' and 'top'. It extracts the categories of the movie that corresponds to the input title, then transforms these categories into a matrix of token counts. The function then utilizes a method to generate a list of probabilities for each category. After sorting these probabilities in a descending order, it returns a list of movie recommendations based on the categories that have the highest probabilities. The number of recommendations is limited to the value of 'top'.



```
----------------------------------------
 Recommendation for 'Ocean's Twelve'
----------------------------------------
InuYasha the Movie 4: Fire on the Mystic Island
Firedrake the Silver Dragon
Jailbirds New Orleans
Omo Ghetto: the Saga
Naruto Shippuden the Movie: Blood Prison
```

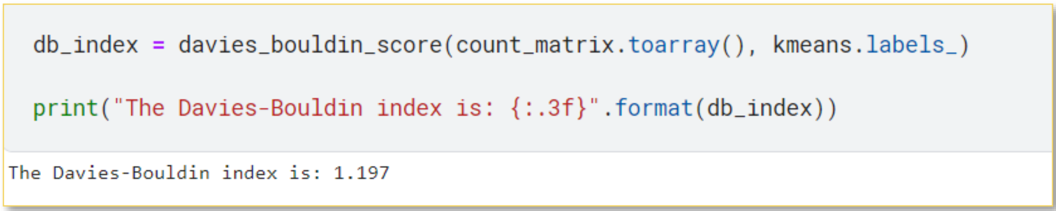


```
----------------------------------------
 Recommendation for 'PK'
----------------------------------------
Falsa identidad
Omo Ghetto: the Saga
Midnight Mass
On the Verge
InuYasha the Movie 4: Fire on the Mystic Island
```

Figure 16

EXPERIMENT RESULTS

The Davies-Bouldin index is an evaluation metric used to measure the quality of clustering results. It is calculated as the average similarity between each cluster and its most similar cluster, relative to the average dissimilarity between each cluster and its most dissimilar cluster. A lower Davies-Bouldin index indicates that the clustering results are of higher quality, as the clusters are more distinct and less overlapping [30].
Figure 17 shows the Davies-Bouldin index for the k-mean clustering, which suggests that the clustering results were of good quality. This means that while the clusters may be relatively distinct, they may also have some degree of overlap or similarity.



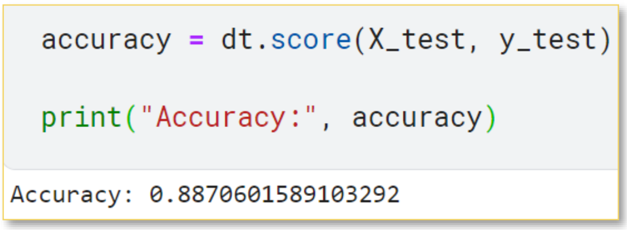
```
db_index = davies_bouldin_score(count_matrix.toarray(), kmeans.labels_)

print("The Davies-Bouldin index is: {:.3f}".format(db_index))
```

```
The Davies-Bouldin index is: 1.197
```

Figure 17

The naive Bayes model achieved a Davies-Bouldin index of 1.317, which suggests that the clustering results were of slightly higher quality than the k-mean clustering results. It's important to note that the accuracy of the supervised learning models may not be as reliable as the Davies-Bouldin index, since we used the results of k-mean clustering as labels instead of actual data. This means that the accuracy results may not accurately reflect the true performance of the models. However, the accuracy score of the decision tree model on the test set is 0.88 (Figure 18), which means that the model predicted the correct target label for 88.7% of the samples in the test set. This is a relatively high accuracy score, indicating that the model is performing well on the test set. However, the accuracy of the logistic regression model was lower at 0.460(Figure 19), which suggests that this model may not be as reliable as the decision tree model.

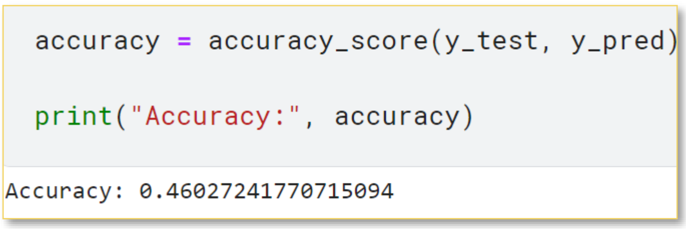


```
accuracy = dt.score(X_test, y_test)

print("Accuracy:", accuracy)
```

```
Accuracy: 0.8870601589103292
```

Figure 18



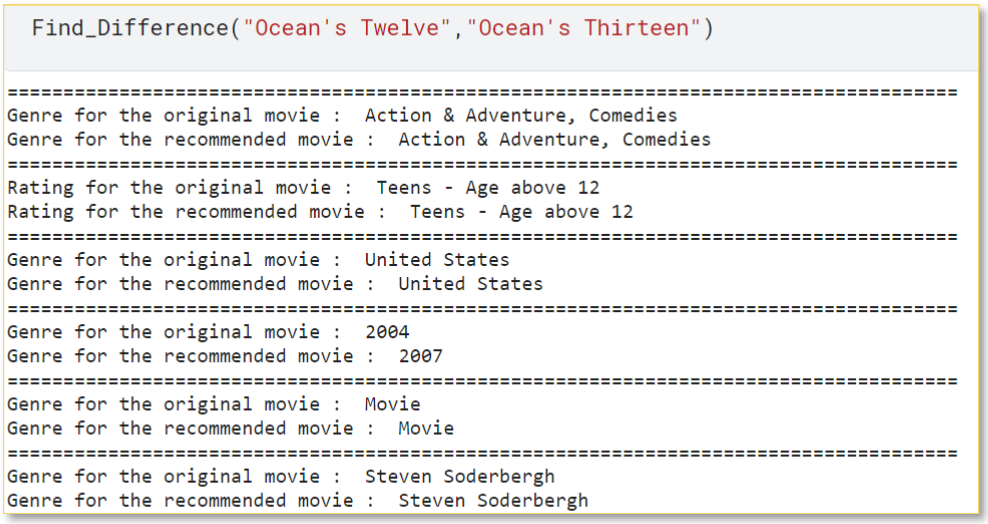
```
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)
```

```
Accuracy: 0.46027241770715094
```

Figure 19

We also compared the features of an original movie and a recommended movie using KNN. We found that the first recommended movie had similar features to the original movie in terms of genre, rating, country, director, and type of media.



```
Find_Difference("Ocean's Twelve","Ocean's Thirteen")
```

```
=====================================================================================
Genre for the original movie :  Action & Adventure, Comedies
Genre for the recommended movie :  Action & Adventure, Comedies
=====================================================================================
Rating for the original movie :  Teens - Age above 12
Rating for the recommended movie :  Teens - Age above 12
=====================================================================================
Genre for the original movie :  United States
Genre for the recommended movie :  United States
=====================================================================================
Genre for the original movie :  2004
Genre for the recommended movie :  2007
=====================================================================================
Genre for the original movie :  Movie
Genre for the recommended movie :  Movie
=====================================================================================
Genre for the original movie :  Steven Soderbergh
Genre for the recommended movie :  Steven Soderbergh
```

Figure 20

However, it's important to note that the accuracy of the supervised learning models may not be as reliable because we used the results of k-mean clustering as labels instead of actual data. Therefore, after comparison of various recommendation Systems, we can observe that the K-Mean Clustering algorithm has the highest accuracy of recommendations for an individual.

ETHICAL ISSUES

Making inaccurate predictions is a major ethical issue for movie recommendation systems [31], as it can result in dissatisfied users and financial losses for both the platform and its users. Therefore, it is crucial to ensure that the system is dependable and precise in its suggestions to prevent users from being disappointed and causing damage to the platform's reputation.

Nevertheless, ethical concerns may arise when collecting user data to enhance the recommendation system. It is important to protect user privacy and their personal information [31] from unauthorized access or misuse. Additionally, it is crucial to be transparent about the collection and usage of user data and to obtain user consent before doing so.

To sum up, while movie recommendation systems can significantly enhance the viewing experience for users, it is essential to consider ethical factors. These include precision, user privacy, transparency.

CONCLUSION

In conclusion, we developed a content-based recommendation system using an unlabeled dataset of movies and TV shows from Netflix. This recommendation system with the combination of many evaluation metrics makes itself relevant and useful. This paper focused on the content-based filtering and implementation on K-Mean, K nearest neighbour, basic clustering and 3 supervised learning algorithms and finding the ”N” recommendations. A comparative study of the different techniques we can conclude that using K-Mean Clustering would give better accuracy and performance results. Additional factors from the dataset which may provide more insights can be considered in the future. Nevertheless, identifying correlations between customer or a user satisfaction to give recommendations with various factors can also be a way in the future.

REFERENCE

1. Streaming services like Netflix have been one of the few 'winners' during the Covid-19 pandemic. [Online]. Available: Link
2. Lika, B., Kolomvatsos, K., Hadjiefthymiades, S. (2014) Facing the cold start problem in recommender systems. Expert Systems with Applications, Pages 2065-2073.
3. Lops, P., de Gemmis, M., Semeraro, G. (2011) Content-based Recommender Systems: State of the Art and Trends. Recommender Systems Handbook. Pages 73-105.
4. Hatami, M., Pashazadeh, S. (2014) Improving results and performance of collaborative filtering-based recommender systems using cuckoo optimization algorithm,Int J Comput Appl, Pages 46-51.
5. Huang, Z., Zeng, D., Chen, H. (2007) A comparison of collaborative filtering algorithms for e-commerce IEEE Intell Syst. Pages 68-78.
6. Burke, R., (2007) Hybrid web recommender systems, Adapt Web, Pages 377-408.
7. Le, J. The 4 recommendation engines that can predict your movie tastes. [Online]. Available: Link
8. Netflix Movies and TV Shows Dataset. [Online]. Available: Link
9. Himel, Md. T., Uddin, M., Hossain, M., Jang, Y. (2017). Weight based movie recommendation system using K-means algorithm. International Conference on Information and Communication Technology Convergence (ICTC).
10. Qader, A. W. (2019). An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges. International Engineering Conference (IEC).
11. Haubl, G. and Trifts, V. (2000) Consumer Decision Making in Online Shopping Environments: The Effects of Interactive Decision Aids. Marketing Science, Pages 4-21.
12. Ricci, F., Rokach, L., Shapira, B. (2011) Introduction to Recommender Systems Handbook, in Recommender Systems Handbook, Ricci, F., Rokach, L., Shapira, B., Kantor, P. B. Springer US. Pages 1-35.
13. Resnick, P., Varian, H. R. (1997) Recommender Systems. Commun ACM. Pages 56-58.
14. Aggarwal, C. (2016). Recommender Systems: The Textbook. Springer Publishing Company.
15. Adomavicius, G., Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering. Pages 734-749.
16. Netflix EDA and Movie Recommendation System. [Online]. Available: Link
17. Havolli, A., Maraj, A., Fetahu, L. (2022). Building a content-based recommendation engine model using Adamic Adar Measure; A Netflix case study. 11th Mediterranean Conference on Embedded Computing.

18. Schein, A. I., Popescul, A., Ungar, L. H., Pennock, D. M. (2002) Methods and metrics for cold-start recommendations, Proce. 25th annual international ACM SIGIR conference on Research and development in information retrieval, Pages 253-260.
19. Gong, S. (2010) A Collaborative Filtering Recommendation Algorithm Based on User Clustering and Item Clustering. Journal of software. Pages 745-752.
20. Recommendation engine with network. [Online]. Available: Link
21. Juluru, K., Shih, H., Murthy, K., Elnajjar, P. (2021) Bag-of-Words Technique in Natural Language Processing. Published online in Radiographics. Pages 1420-1426.
22. Bose, D. (2020). Movie recommendation system with Collaborative Filtering using K-NN. Letterkenny Institute of Technology.
23. Cheng, Z. (2021). A comparison of cosine similarity vs Euclidean distance in ALS recommendation engine. [Online]. Available: Link
24. Modified K Nearest Neighbours Recommender. [Online]. Available: Link
25. Zhu, X., Nie, Y., Jin S. (2015). Spammer Detection on Online Social Networks Based on Logistic Regression[M]//Web-Age Information Management. Springer International Publishing. Pages 29-40
26. Liu, P. (2000). Analyses of regular fuzzy neural networks for approximation capabilities[J]. Fuzzy Sets and Systems. Pages 329-338.
27. Bartz, K., Murthi, V., Sebastian, S. (2006) Logistic regression and collaborative filtering for sponsored search term recommendation. Second workshop on sponsored search auctions.
28. Lau, C. W. (2011). News Recommendation System Using Logistic Regression and Naive Bayes Classifiers.
29. Quevedo, J. R., Montanes, E., Ranilla, J. (2010). Ranked tag recommendation systems based on logistic regression. International Conference on Hybrid Artificial Intelligence Systems. Pages 237-244
30. Mughnyanti, M., Efendi, S., Zarlis, M. (2020). Analysis of determining centroid clustering x-means algorithm with davies-bouldin index evaluation. IOP Publishing Ltd. Page 725
31. Milano, S., Taddeo, M., Floridi, L. (2020) Recommender systems and their ethical challenges. AI & SOCIETY. Pages 957-967