VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

MOVIE RECOMMENDATION SYSTEM

Course: CSE 3505 Foundations of Data Analytics
Faculty: Dr. R Priyadarshini

Class ID: CH2022231000372
Slot: F1
Project Type: C

Aditi Jaiswal 20BRS1184
Khushi Kashyap 20BCE1235

BONAFIDE CERTIFICATE

Certified that the project report for the course “CSE3505 - Foundations of Data Analytics” entitled “subject” is a bonafide work of Aditi Jaiswal (20BRS1184) and Khushi Kashyap (20BCE1235) who carried out the Project work under my supervision and guidance.

Prof Priyadarshini

School of Computer Science and Engineering

(SCOPE)

VIT University, Chennai

Chennai – 600 127

ACKNOWLEDGEMENT

I wish to express my sincere thanks and deep sense of gratitude to my project guide, Prof PRIYADARSHINI, for her consistent encouragement and valuable guidance offered to me in a pleasant manner throughout the course of the project work.

I am extremely grateful to Dr. GANESAN R, Dean of the School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Chennai, for extending the facilities of the school towards our project and for her unstinting support.

I take this opportunity to thank all the faculty of the school for their support and wisdom imparted to me throughout the course.

TABLE OF CONTENTS

1. ABSTRACT

Over the past years, the internet has broadened the horizon of various domains to interact and share meaningful information. As it is said that everything has its pros and cons therefore, along with the expansion of domain comes information overload and difficulty in extraction of data. How to quickly find one's favorite movie in a large number of movies has become a very important issue. To overcome this problem the recommendation system plays a vital role. Personalized recommendation systems can play an important role especially when the user has no clear target movie. It is used to enhance the user experience by giving fast and coherent suggestions. This paper describes an approach which offers generalized recommendations to every user, based on movie popularity and/or genre. Content-Based Recommender System is implemented using deep learning approaches. This paper also gives an insight into problems which are faced in content-based recommendation systems and we have made an effort to rectify them.

2. INTRODUCTION

A Recommendation System is a major area that helps a user to find out the information which is beneficial for him/her from a variety of data available. It is a tool for information filtering which is built to predict the "rating" or "preference" of a user based on his input. It collects data about the user's preferences directly or indirectly on different objects like movies, shows, shopping, tourism, TV etc. On the other hand, in the development of movie recommendation systems, the system uses the user's previous watched movies.

There are mainly two types of Movie Recommendation System: first, where recommendation is done based on similarity between users (Collaborative Filtering) and second, by considering particular user's activity (Content Based Filtering) which he wants to engage with. These are advanced filtration mechanisms that predict the possible movie choices of the concerned user and their preferences towards a domain-specific item. Collaborative filtering is the method to filter or calculate the items through the sentiments of others. Collaborative filtering first collects the movie ratings or preference given by IMDB and Rotten Tomatoes and then suggests movies to the different users based on similar tastes and interests in the past. On the other hand, Content Based Filtering filters out movies based on their individual content. It recommends movies to the user with a similar content base to those that are previously watched by the user.

In this project, we have implemented a simple Content Based Movie Recommendation System that uses the Cosine Similarity Algorithm. Cosine Similarity is implemented on this dataset in order to obtain the best-optimized result. The process of recommendation of a title is simplified in this method. The recommender system predicts the user's preference of a title on the basis of different parameters such as genre, cast, director, keywords and tagline. The recommender system works on the concept of distance. It finds similarity between each data point for a selected list of attributes from a dataset in the form of a similarity score. Higher similarity score between two data items indicates that the two are similar and vice versa. So based on the user's input title, movies are segregated from the dataset which have a high similarity score output with the user's input and these are recommended to the user. This process optimizes the process.

The work starts with section 1 as the Introduction section with the basics of data handling. Section 2 discusses the work done by the group for visualizing the data. Section 3 describes the evolution of the proposed recommendation system. Section 4 shows the algorithm of the proposed system. Section 5 shows the implementation of the proposed system.

2.1 RELATED WORK

Some of the common approaches of recommender system are:

1. Content-based filtering
2. Collaborative filtering
3. Hybrid filtering

A. Content Based Filtering

This approach filters the items based on the likings of the user. It gives results based on what the user has rated earlier. The method to model this approach is the Vector Space Model (VSM). It derives the similarity of the item from its description and introduces the concept of TF-IDF (Term Frequency-Inverse Document Frequency).

The similarity between item vectors can be computed by three methods:

1. Cosine similarity
2. Euclidean distance
3. Pearson's correlation

B. Collaborative Filtering

It depends upon the users who have similar interests and gives the result based on all the users. User-based: In user-based collaborative filtering, it is considered that a user will like the items that are liked by users with whom they have comparable taste. Item-based: Item-based collaborative-filtering is different, it expects the users to like items that are related to items that he has liked earlier.

C. Hybrid Filtering

Hybrid filtering can be known as a combination of collaborative filtering and content-based filtering. It is the most common and popular technique today. It avoids the weakness of every single recommender technique. There are some problems related to the recommendation system. They are: Cold-start problem: When a user registers for the first time, he has not watched any movie. So, the recommendation system does not have any movie based on which it can give results. This is called the cold-start problem. Recommendation system goes through this problem as a result of no previous record. This happens with every new user once. Data sparsity problem: This problem occurs when the user has rated very few items based on which it is difficult for the recommendation system to give accurate results. In this problem, the results given are not much similar to the expected result. Sometimes, the system fails to give successful results and generates weak recommendations. Also, data sparsity always leads to coverage problems. Scalability: Another problem associated with recommendation is scalability. In this, the encoding goes linearly on items. The system works efficiently when the data set is of limited size. As the data set increases, it becomes difficult for the recommendation system to give accurate results based on varying genres of movies. The scalability problem can be solved by dimensionality reduction techniques such as singular value decomposition (SVD). It also speeds up the generation of results and produces a reliable and efficient result. Synonym: When many words have a similar meaning then they are known as synonyms. In this problem, the system sometimes fails to understand the difference between two similar words and cannot produce the desired output. E.g.: movie and film have the same meaning but the recommendation system considers them as different words and because of this it fails to give the accurate output. Singular Value Decomposition (SVD), especially Latent Semantic Indexing is capable of solving the synonymy problem. In other recommendation systems, recommendations are based on the ratings and genres given by other users due to which it becomes difficult to give accurate results. This is called collaborative filtering.

This is a common method for a recommendation system. To make things easier for people and overcome this drawback, we have used another method for the recommendation which is content-based filtering. This method uses ratings and genres given by the user itself. It gives recommendations based on the movies watched by the user earlier. This helps us in giving accurate suggestions because the ratings and genres are given by the user only and depend entirely on the user's choice, not on any other person's choice. Previous research lacks the accuracy of true recommendations due to their dependency on other users. Here it is important to mention that recommendations can be based on a particular user and they assure to give recommendations on the mark as they depend on the likes and dislikes of a particular user. The proposed system is capable of storing a large amount of data and giving efficient results. Our recommendation system searches for the best movies that are similar to the movie that we have watched based on genre and gives the result.

2.2 Problem Statement

To create a movie recommendation system using Content Based Filtering which will be built using the artificial algorithm of Cosine Similarity that analyzes the user's favorite genre, cast, director, keywords and tagline and recommend movies according to their liking. Secondly, the response should be based on the preference of the particular user. The user will submit queries depending on their liking of the movies. The System should be able to analyze the liking and then accurately recommend movies to the user. And finally, providing related content out of relevant and irrelevant collection of items to users of online service providers. Our project aims to recommend movies to users based on content of items rather than other user's opinions.

2.3 Objective

In this project we are going to perform some exploratory data analysis to find some hidden trends and patterns in the dataset. The data is going to be loaded and read using pandas, which will be followed by some cleaning and processing of the data. Next, we will explore the dataset through visualizations and graphs using matplotlib and seaborn and finally answer some questions related to the dataset. Our project aims to filter the data using different algorithms and recommend the most relevant items to users. It first captures the past behavior of a customer and based on that, recommends products which the users might be likely to buy/watch. This is done by taking a movie input from the user. We are going to try to implement the predictive system using Cosine Similarity to produce top 30 suggestions for the user based on his/her input. It will help the users get personalized recommendations and enable him/her to make correct decisions, increase sales, redefine the users web browsing experience, retain the customers, enhance their online media viewing experience etc.

2.4 Statement

When it comes to OTT-Platforms the main pitfall is the content-recommendation for the subscriber. Companies often wrongly profile its audience and there is more time spent by the user on choosing the platform and/or the movie or TV show because of the vast availability of many options. This can be reverted by performing data analysis of the individual customer and using AI and ML algorithms to suggest a movie / show on cross-platform.

2.5 Preparing the data

The dataset used in this project: movies.csv has several columns of information for numerous movies, 24 columns for 4803 movies to be precise. It has been clubbed and filtered to form a unique dataset.

Numerical: - index, budget, id, popularity, release_date, revenue, runtime, vote_average, vote_count,

Categorical: - genres, homepage, keywords, original_language, original_title, overview, production_companies, production_countries, spoken_languages, status, tagline, title, cast, crew, director

2.6 Challenges

The challenges faced by the author include cleaning the data before being used for analysis, extracting features from the processed data necessary for visualizing and recommending content. Analyzing the final results to draw appropriate conclusions.

3. LITERATURE REVIEW

Shubhankar et al. [1] highlighted the main goal of ranking and modeling topic evolution, by an efficient algorithm. The topic, evolution, has become a challenging task over time for the researchers. They suggested the topic as a summary of its content, and also introduced a unique approach, that assigns the rank to a topic by applying PageRank algorithm, without considering the time of research publication [3]. Furthermore, they have categorized topics based on the set of the topic and closed keyword-set. The closed keyword-set were formed such that; phrases were selected from topics of the publications along with a user-defined minimum support. PageRank algorithm has been applied in iterative passion, to assign authorities to score on the research paper. The authority scores, based on popularity in the research community; however, the algorithm also identifies hot topics by evaluating them on a timeline and is shown as landmarks. They also tried to find fading topics; they first checked for topic detection, then evaluated landmark topics, and at the end tried to find fading topics. The algorithm proved as most effective and faster when tested on DBLP dataset.

Song et al. [2] emphasized the use of text clustering for Topic Detection. The text clustering dominates other algorithms in terms of time, computational complexity and cost. There are many ways to transmit data over a network; still, we need methods to avoid noise and irrelevant information. They considered the unstructured and scattered data over the internet as text copra and introduced a two-step algorithm for clustering the text copra. The first step uses C-Process to create overlapping clusters with the help of Canopy Clustering. The Canopy Clustering is usually applied before the K-mean algorithm to speed up clustering of large data-set. In the second step named K-means apply rough clustering on results based on the common points between the clusters. The K-Means uses the X-Means algorithm. The experiments have proved better performance than k-means clustering and single pass algorithms, and proved to be more suitable for detection of online topics.

Wu et al. [4] discussed CAR (Credible association rule) , a new method to relate and track documents. The CAR does not use prior knowledge of category structure as compared to other automated procedures. The other traditional processes detect related documents based on the topic of documents, with the help of some predefined rules or categorization. This method makes use of the term frequency-inverse document frequency (TF-IDF) as a feature pre-selection set. TF-IDF is a numerical statistic which shows how words are important to a document. After the feature subset selection, CAR and minimal clique algorithms were applied. These two algorithms use an adjacency matrix to produce credible association rules. The refinements, removes noise and common words with the high frequency that are not related to the topic of the document. A high level of reliability, availability and performance are achieved by applying refinements like Inverse Document Frequency (IDF) and quasi-maximal cliques.

Jo et al. in [5] proposed algorithms for the topic detection from linked textual corpus using the relationship of topics in terms of the distribution and the link distribution. Their algorithm generates a ranked list of the topics the method has shown effective results with arXiv and Citeseer. Jo et al. in [6] discovered rich patterns of topic evolution within built-in features of time-stamped documents. Instead of analyzing documents on the specific interval, the method focuses and treats topics separately as they appear over time in chronological order. The information is obtained such that; it qualifies the topic as either new or it has some similarity with existing topic. The result was visualized by chronological scanning on a graph known as topic evolution graph. The topological or time restrictions were not considered while building the graph. The nodes of the graph represent topics and the connection between the topic nodes represents the cross-citation relationship. This representation of information projects a huge amount of knowledge about topic evolution. Details about a single topic can be obtained by selecting a seed topic and studying its connections with other nodes. These connections change as time passes, the emergence of new topics adds new nodes to the graph and also changes connections between them. The testing was carried out with the ACM repository.

Jayashri et al. [7] discussed retrieval of temporal and event-based knowledge from a huge collection of historical documents. The method uses temporal text mining (TTM) and Topic detection and tracking (TDT) techniques. The TTM extracts important patterns related to time, like collecting term frequency in a sequence of time; it also helps in keeping track of modification in the words with respect to time. In TDT, a clustering problem called Adaptive Resonance Theory (ART) is used, that tracks unknown topics in the system and assigns them to previously identified topics. The Evaluation of such information is usually carried at the time of its arrival, which helps to consider temporal properties of the topics. The Evolution is implemented using an incremental algorithm. The experiments helped to discover new trends and identify trends that cannot be obtained from documents if analyzed individually.

Cui et al. [8] highlighted topic evolution in text data as an important task. The Text Flow has been introduced; which studies various patterns that appear from various topics. Three-level features selection was conducted namely keywords, critical events, and evolution trends, then these features were visualized via a coherent visualization technique that shows the complex and important relation between them. The Text Flow is an interactive visual analysis tool that helps users analyze how and why the correlated topics change over time. First patterns related to merging/splitting are discovered via hierarchical Dirichlet process employed in incremental passion followed by extraction of keyword correlation and critical events. The result can be visualized by a three-level directed acyclic graph such that the user can deduce various information at any level of consideration. This method helps users visually study the relation between two topics that is best as it represents this relationship in an interactive way.

Jin [9] focused on detecting topics with immense information available over the internet. They introduced a new method by combining Suffix Tree Clustering (STC) and Semantic analysis that approaches the problem in two steps. In the first step feature selection is done with the help of NLP algorithm, by selecting meaningful words for clustering and the weight is assigned to word using term frequency-inverse document frequenting (TF-IDF). The NLP results in parts of the sentences in the form of the noun, verb and named entity. The result of feature selection is supplied to STC to form clusters where the score is assigned to them. TDT is applied to track topics, focusing on topic drifting. This is an inherent difficulty of topic evolution, which occurs over time as new information emerges. The clustered contents are represented via VSM (Vector Space Model) by selecting only top K words that are added to the vector [10]. The semantic analysis is used to add significance to the topics and the significance can be measured by applying filters to the words and analyzing structure of words under a cluster which share the same meaning. The experiments proved that topics can be tracked effectively.

Zhang et al. [11] highlighted that news collection should be structured such that topic detection and clustering become easy. The vector space model (VSM) is one of the easiest and productive methods that can be used for the representation of topics, and the Information gain algorithm is used for feature selection. The features, then ranked by assigning a score to each of them. Those features are selected which have high scores among the feature selection set. The TF-IDF used to represent and score the features. Then K-means employed for partial clustering, each object assigned to a cluster center where distance based on the cosine distance, which assigns each feature to a cluster that is more similar in properties. The K-Means algorithm is not efficient when the dataset is large enough; it is used in conjunction with VSM, which facilitates by representing data in a form which is processed easily by K-Means algorithm. The process was evaluated by use of Topic detection and tracking method and verified that topic detection that is based on K-means outperforms if used for large-scale data.

Yue et al. [12] emphasized the importance of technology in finding relevant information in a huge set of fast-growing information. A topic detection algorithm based on K-means clustering is proposed by Authors [12]. This method identifies some keywords and assigns weight to it which then helps in detecting topics. Two methods are used to select keywords; the first method selects hot words from the database which is man-made. The second step selects words by means of the TF-IDF method which makes sure no word with high frequency is left behind, that is related to the document [13]. The method proposed by [12] is a two-step process. In the first step, documents are selected which are related to 14 categories, which are then parsed by a well-known Chinese parsing system known as ICTCLAS to obtain components of the document. Text vector is then created from the component by selecting feature words which are represented by a vector space model. The weight of featured words is calculated and assigned to them. Document summary is calculated using cosine formula, which is used to find similarity among documents. In the last step documents, clusters are created using the k-means algorithm. Various experiments are conducted to prove its efficiency.

Masada et al. [14] discussed various ways of connecting and referencing scientific documents that ultimately helps in finding a relationship between scientific documents based on the citation. Instead of text analysis, references are extracted with a model named TERESA. TERESA extends (LDA) Latent Dirichl et al. location, which models relation between documents in the form of transitions between them. TERESA focuses on the collection of the document as a whole to discover directed relations, thus providing a global view of relationships. Global view of the system introduces a problem as the huge amount of computation is required; to overcome this problem (VB) Vibrational Bayesian inference is used. VB accelerates GPUS compatible with NVidia CUDA, thus, hundreds of threads can be executed. As compared to another method which creates multiple local views which are difficult to conceptualize, this method provides a single view of all the relationships. Many iterations of this method will give different projections that show important information about topic evolution.

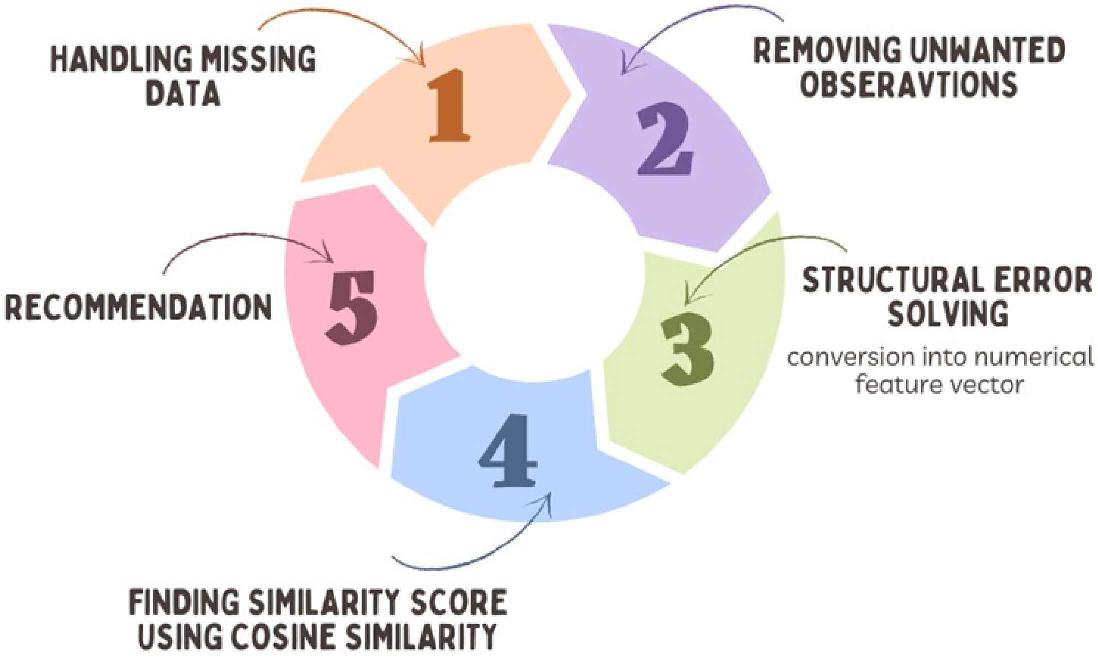
Lv et al. [15] identified various problems concerned with traditional approaches that consider the huge amount of information while tracking topics as useful information might be skipped. A model is proposed by [15] that makes use of similarity based on subtopic and events; it also makes use of time partition to study development history and location characteristics. First considering temporal characteristics of the topic they are partitioned into subtopics clusters based on time slices. The process of partitioning based on temporal characteristics is applied only once. Topics are strongly similar within a cluster. Based on temporal and attribute similarities subtopics are combined into events. Subtopic evolution relation is discovered by keeping in mind the events in which these are combined. Topics might change with time and changing events. This method is applied to a dataset of 500 news articles and proves to be effective.

Shah et al. in [16] proposed a model to handle the literature overloading and selecting the most relevant papers published in recent years. They tried to answer the questions like, how many articles are sufficient for a good literature review, and how much past year's literature will be enough to meet the required level for a good literature review?

Chen et al. [17] have focused on SSD-PLSA, a novel topic detection method. SDDPLSA is the combination of Semantic Dependency Distance SSD and PLSA. PLSA has proved to be one of the effective and efficient methods in Text Mining. Syntax and semantic information of text are also considered for the effectiveness of methods proposed by the Authors [17]. The SDD-PLSA method works in two steps; First step uses semantic features of sentences to group them; this is an important step. The second step uses the result of the first step which is a tree structure. Optimal semantic dependency between parent and child nodes is calculated, these nodes are linked with the help of dependency grammar. The link between a parent node and child node is called dependency link [18]. In Dependency Grammar, a word that represents the meaning of the whole sentence is selected as the head word and all others that are dependent on the headwords are descendants. In the second step variation of the PLSA classifier is applied to the result of the first step. Experiments show that the result of SDD-PLSA is more efficient and accurate than PLSA if used alone.

However, these methods mentioned above are immature and not comprehensive, still there is a need for a system that can organize the published material in an informative manner.

4. ARCHITECTURE DESIGN



HANDLING MISSING DATA
1
REMOVING UNWANTED OBSERAVTIONS
2
STRUCTURAL ERROR SOLVING
conversion into numerical feature vector
3
4
FINDING SIMILARITY SCORE USING COSINE SIMILARITY
5
RECOMMENDATION

1. Handling Missing Data

Replacing all the null values in the dataset with a null string for facilitating the cosine score calculation which cannot be computed using null values otherwise.

2. Removing Unwanted Observations

Combining the relevant attributes (columns) into a single list for every data point (movie). Here relevant attributes refer to all the features of the movies that provide pertinent information of the movie which can be useful in finding similarity and finally providing good and accurate recommendations.

3. Solving Structural Error

Converting the list of relevant attributes constructed in the previous step into a feature vector. This basically means converting the categorical string values into numerical values.

4. Finding Similarity Score Using Cosine Similarity

Finding similarity score for every movie with every other datapoint in the dataset using the feature vector calculated in the previous step with the use of the Cosine Similarity Algorithm.

5. Movie Recommendation

Taking a movie title name input from the user, finding the input movie in the dataset and getting the list of similar movies to the input based on the cosine similarity score. The higher the similarity score, the higher is the similarity. Hence top 30 similar movies sorted in descending order based on the score are recommended to the user.

4.1 Proposed Method: Cosine Similarity

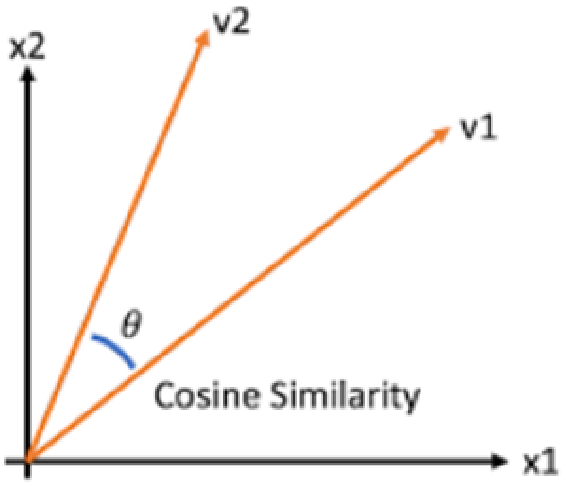
In Data Mining, similarity measure refers to distance with dimensions representing features of the data object, in a dataset. If this distance is less, there will be a high degree of similarity, but when the distance is large, there will be a low degree of similarity.

Some of the popular similarity measures are: –

1. Euclidean Distance.
2. Manhattan Distance.
3. Jaccard Similarity.
4. Minkowski Distance.
5. Cosine Similarity.

Cosine similarity

Cosine Similarity among two objects measures the angle of cosine between the two objects. It compares two documents on a normalized scale. It can be done by finding the dot product between the two identities.



As the above diagram shows, the angle between v1 and v2 is Ɵ. Lesser the angle between the two vectors, more is the similarity. It means if the angle between two vectors is small, they are almost alike each other and if the angle between the two vectors is large then the vectors are very different from each other.

Cosine similarity is a popular similarity measure, helpful in determining how similar the data objects are irrespective of their size. For example, we can measure the similarity between two sentences using Cosine Similarity. In cosine similarity, data objects in a dataset are treated as a vector.

The formula to find the cosine similarity between two vectors is –

$$\boxed{Cos(x, y) = x . y / ||x|| * ||y||}$$

where,

- x . y = product (dot) of the vectors 'x' and 'y'.
- ||x|| and ||y|| = length of the two vectors 'x' and 'y'.
- ||x|| * ||y|| = cross product of the two vectors 'x' and 'y'.

Example: -

Consider an example to find the similarity between two vectors – ‘x’ and ‘y’, using Cosine Similarity.

The ‘x’ vector has values, $x = \{3, 2, 0, 5\}$

The ‘y’ vector has values, $y = \{1, 0, 0, 0\}$

The formula for calculating the cosine similarity is: $Cos(x, y) = x \, . \, y \, / \, ||x|| * ||y||$

$x \, . \, y = 3*1 + 2*0 + 0*0 + 5*0 = 3$

$||x|| = \sqrt{} \, (3)^2 + (2)^2 + (0)^2 + (5)^2 = 6.16$

$||y|| = \sqrt{} \, (1)^2 + (0)^2 + (0)^2 + (0)^2 = 1$

$\therefore \; Cos(x, y) = 3 \, / \, (6.16 * 1) = 0.49$

The dissimilarity between the two vectors ‘x’ and ‘y’ is given by: –

$$\boxed{\therefore \; Dis(x, y) = 1 - Cos(x, y)}$$

$= 1 - 0.49 = 0.51$

- The cosine similarity between two vectors is measured in ‘θ’.
- If θ = 0°, the ‘x’ and ‘y’ vectors overlap, thus proving they are similar.
- If θ = 90°, the ‘x’ and ‘y’ vectors are dissimilar.

Advantages:

- The cosine similarity is beneficial because even if the two similar data objects are far apart by the Euclidean distance because of the size, they could still have a smaller angle between them. Smaller the angle, higher the similarity.
- When plotted on a multi-dimensional space, the cosine similarity captures the orientation (the angle) of the data objects and not the magnitude.

5. MODULE EXPLANATION

1. HANDLING MISSING DATA
    a. We may end up building a biased machine learning model which will lead to incorrect results if the missing values are not handled properly. Missing data can lead to a lack of precision in the statistical analysis.
    b. Replacing all the null values in the dataset with a null string for facilitating the cosine score calculation which cannot be computed using null values otherwise.

2. REMOVING UNWANTED OBSERVATIONS
    a. As a part of the data preparation process, data cleansing which includes removing unwanted observations allows for accurate, defensible data that generates reliable visualizations, models, and business decisions.
    b. Out of all the data we require only the most relevant one. Combining the relevant attributes (columns) into a single list for every data point (movie) and making decisions based on it.

3. SOLVING STRUCTURAL ERROR
    a. Structural errors are when you measure or transfer data and notice strange naming conventions, typos, or incorrect capitalization. These inconsistencies can cause mislabeled categories or classes. For example, you may find “N/A” and “Not Applicable” both appear, but they should be analyzed as the same category.
    b. Converting the list of relevant attributes into a feature vector, i.e. converting strings into numerical values.

4. FINDING SIMILARITY SCORE USING COSINE SIMILARITY
    a. The formula for calculating the cosine similarity is : Cos(x, y) = x . y / ||x|| * ||y|| x

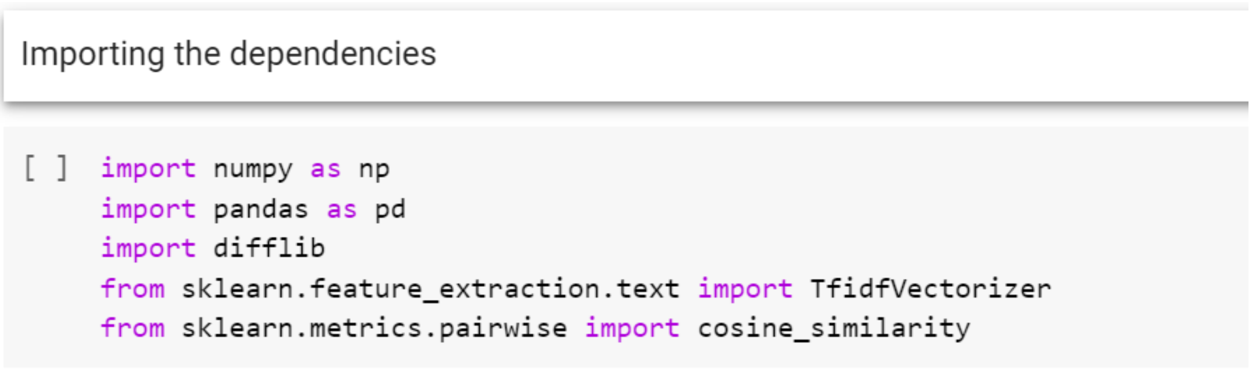
5. Finding similarity score for every data point with every movie using the feature vector of the movies.

6. MOVIE RECOMMENDATION
    a. The intuition behind the content based movie recommendation system using

cosine algorithm is to analyze and make a decision on the basis of a user liking a particular movie or show then he/she might like a movie or a show similar to it.

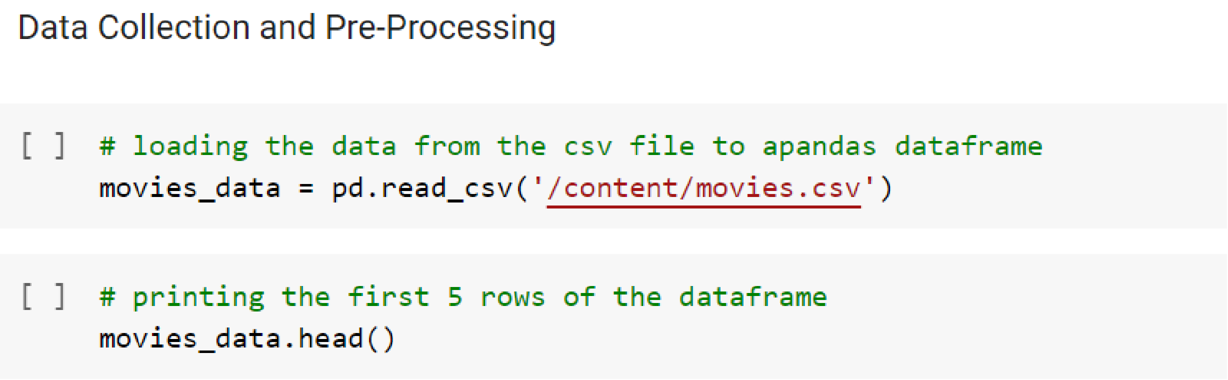
b. Taking a movie name input from the user, finding the movie in the dataset and getting the list of similar movies to the input based on the cosine similarity score.

6. IMPLEMENTATION



Importing the dependencies

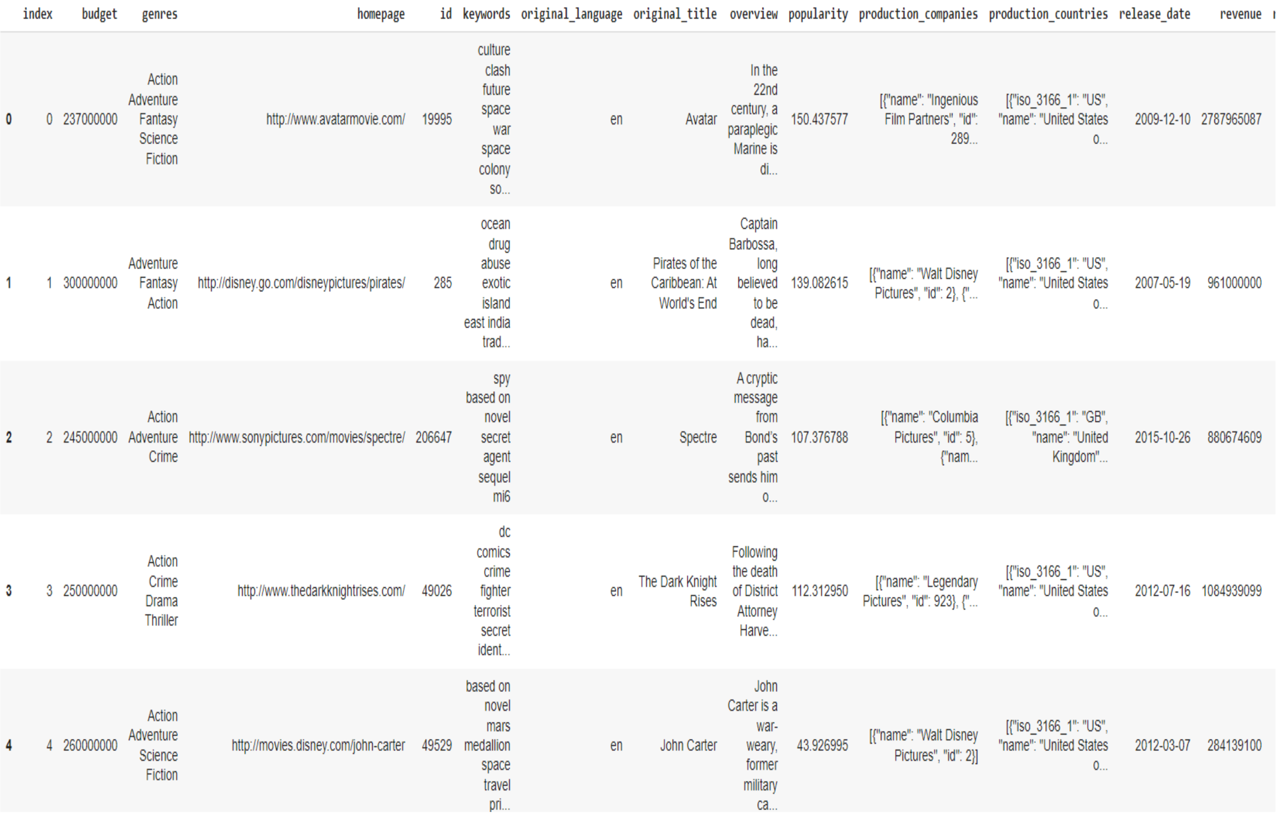
```
[ ]  import numpy as np
     import pandas as pd
     import difflib
     from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn.metrics.pairwise import cosine_similarity
```



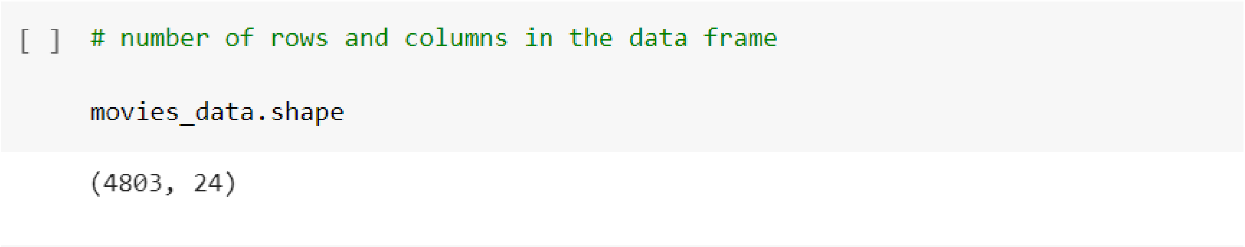
Data Collection and Pre-Processing

```
[ ]  # loading the data from the csv file to apandas dataframe
     movies_data = pd.read_csv('/content/movies.csv')
```

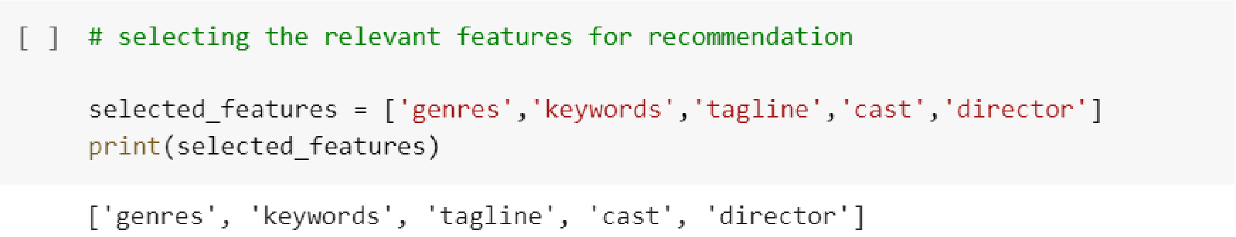
```
[ ]  # printing the first 5 rows of the dataframe
     movies_data.head()
```



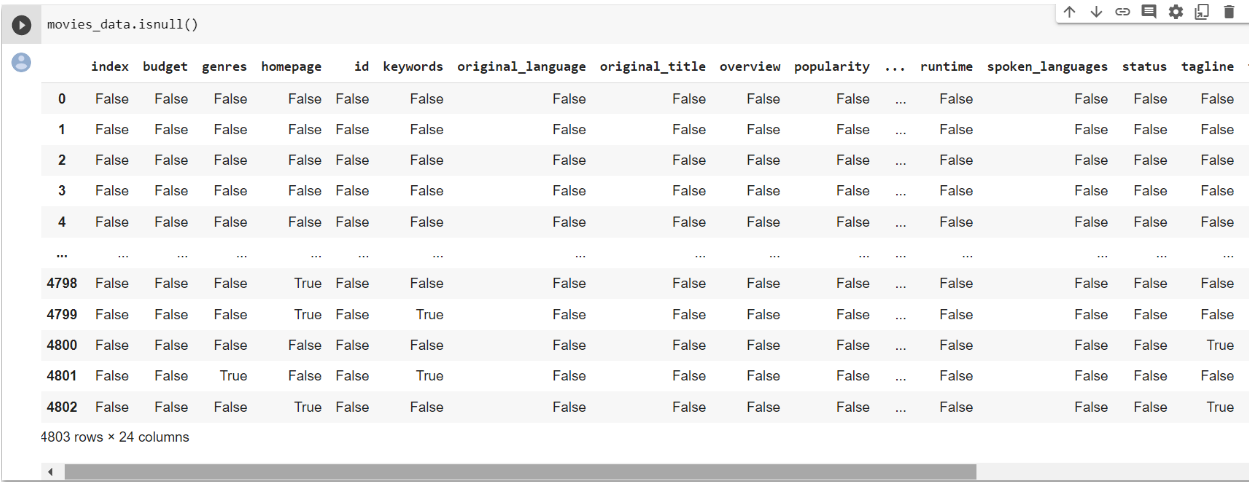
	index	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies	production_countries	release_date	revenue
0	0	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	culture clash future space war space colony so...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...	[{"iso_3166_1": "US", "name": "United States o...	2009-12-10	2787965087
1	1	300000000	Adventure Fantasy Action	http://disney.go.com/disneypictures/pirates/	285	ocean drug abuse exotic island east india trad...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt Disney Pictures", "id": 2}, {"...	[{"iso_3166_1": "US", "name": "United States o...	2007-05-19	961000000
2	2	245000000	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	206647	spy based on novel secret agent sequel mi6	en	Spectre	A cryptic message from Bond's past sends him o...	107.376788	[{"name": "Columbia Pictures", "id": 5}, {"nam...	[{"iso_3166_1": "GB", "name": "United Kingdom"...	2015-10-26	880674609
3	3	250000000	Action Crime Drama Thriller	http://www.thedarkknightrises.com/	49026	dc comics crime fighter terrorist secret ident...	en	The Dark Knight Rises	Following the death of District Attorney Harve...	112.312950	[{"name": "Legendary Pictures", "id": 923}, {"...	[{"iso_3166_1": "US", "name": "United States o...	2012-07-16	1084939099
4	4	260000000	Action Adventure Science Fiction	http://movies.disney.com/john-carter	49529	based on novel mars medallion space travel pri...	en	John Carter	John Carter is a war-weary, former military ca...	43.926995	[{"name": "Walt Disney Pictures", "id": 2}]	[{"iso_3166_1": "US", "name": "United States o...	2012-03-07	284139100



```
[ ]  # number of rows and columns in the data frame

     movies_data.shape

     (4803, 24)
```



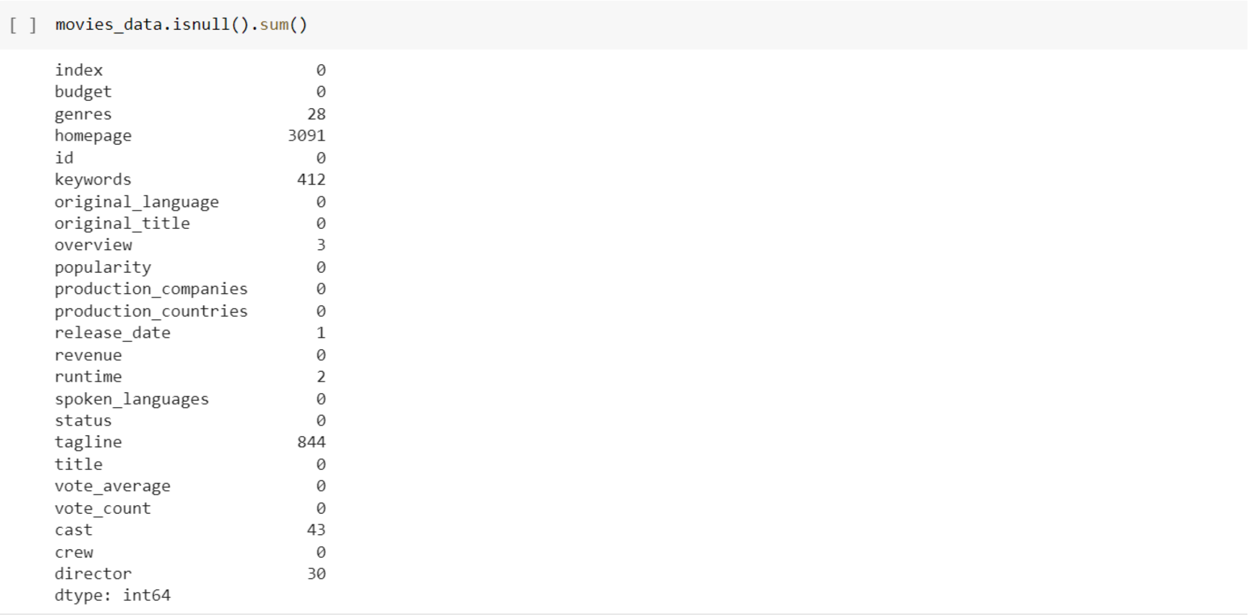
```
[ ]  # selecting the relevant features for recommendation

     selected_features = ['genres','keywords','tagline','cast','director']
     print(selected_features)

     ['genres', 'keywords', 'tagline', 'cast', 'director']
```



```
movies_data.isnull()
```

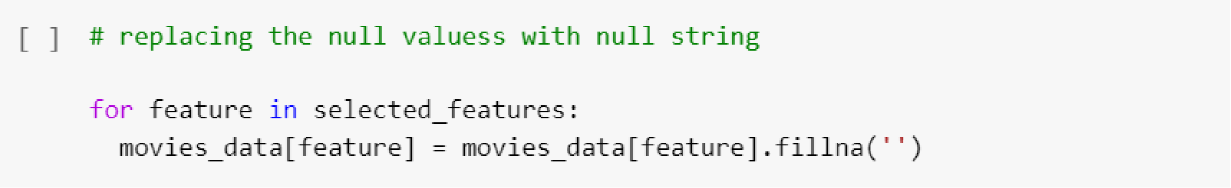
	index	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	...	runtime	spoken_languages	status	tagline
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4798	False	False	False	True	False	False	False	False	False	False	...	False	False	False	False
4799	False	False	False	True	False	True	False	False	False	False	...	False	False	False	False
4800	False	False	False	False	False	False	False	False	False	False	...	False	False	False	True
4801	False	False	True	False	False	True	False	False	False	False	...	False	False	False	False
4802	False	False	False	True	False	False	False	False	False	False	...	False	False	False	True

4803 rows × 24 columns

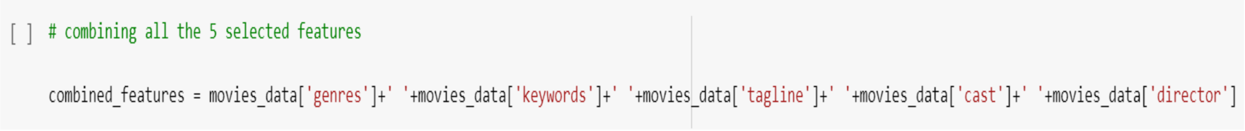


```
[ ] movies_data.isnull().sum()
```

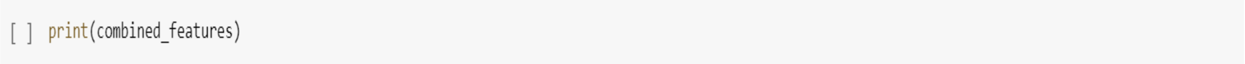
```
index                    0
budget                   0
genres                  28
homepage              3091
id                       0
keywords               412
original_language        0
original_title           0
overview                 3
popularity               0
production_companies     0
production_countries     0
release_date             1
revenue                  0
runtime                  2
spoken_languages         0
status                   0
tagline                844
title                    0
vote_average             0
vote_count               0
cast                    43
crew                     0
director                30
dtype: int64
```



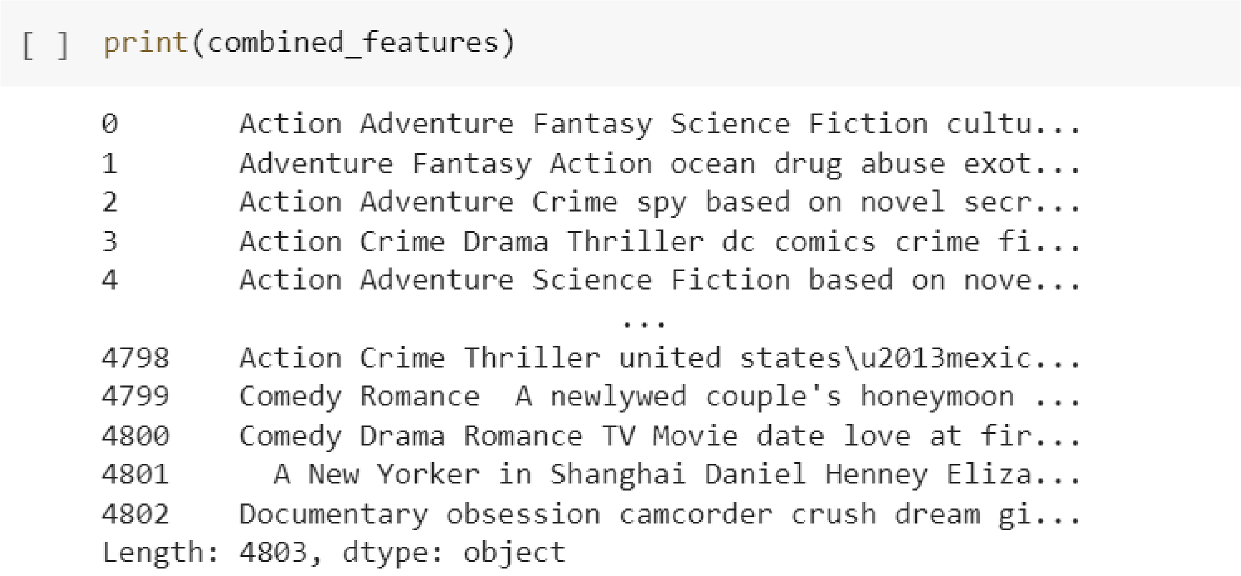
```
[ ] # replacing the null valuess with null string

    for feature in selected_features:
      movies_data[feature] = movies_data[feature].fillna('')
```



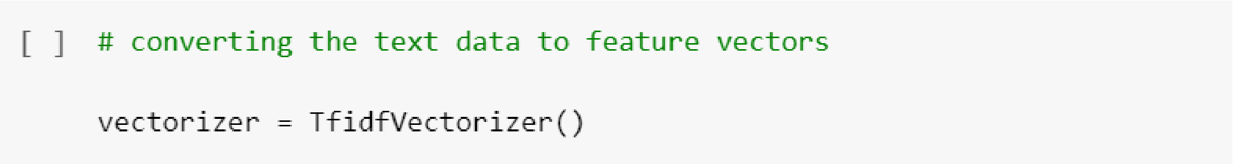
```
[ ] # combining all the 5 selected features

    combined_features = movies_data['genres']+' '+movies_data['keywords']+' '+movies_data['tagline']+' '+movies_data['cast']+' '+movies_data['director']
```



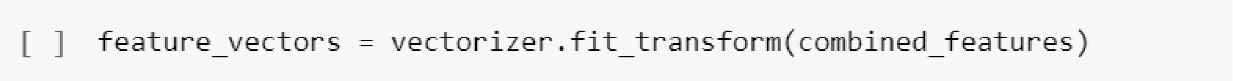
```
[ ] print(combined_features)
```



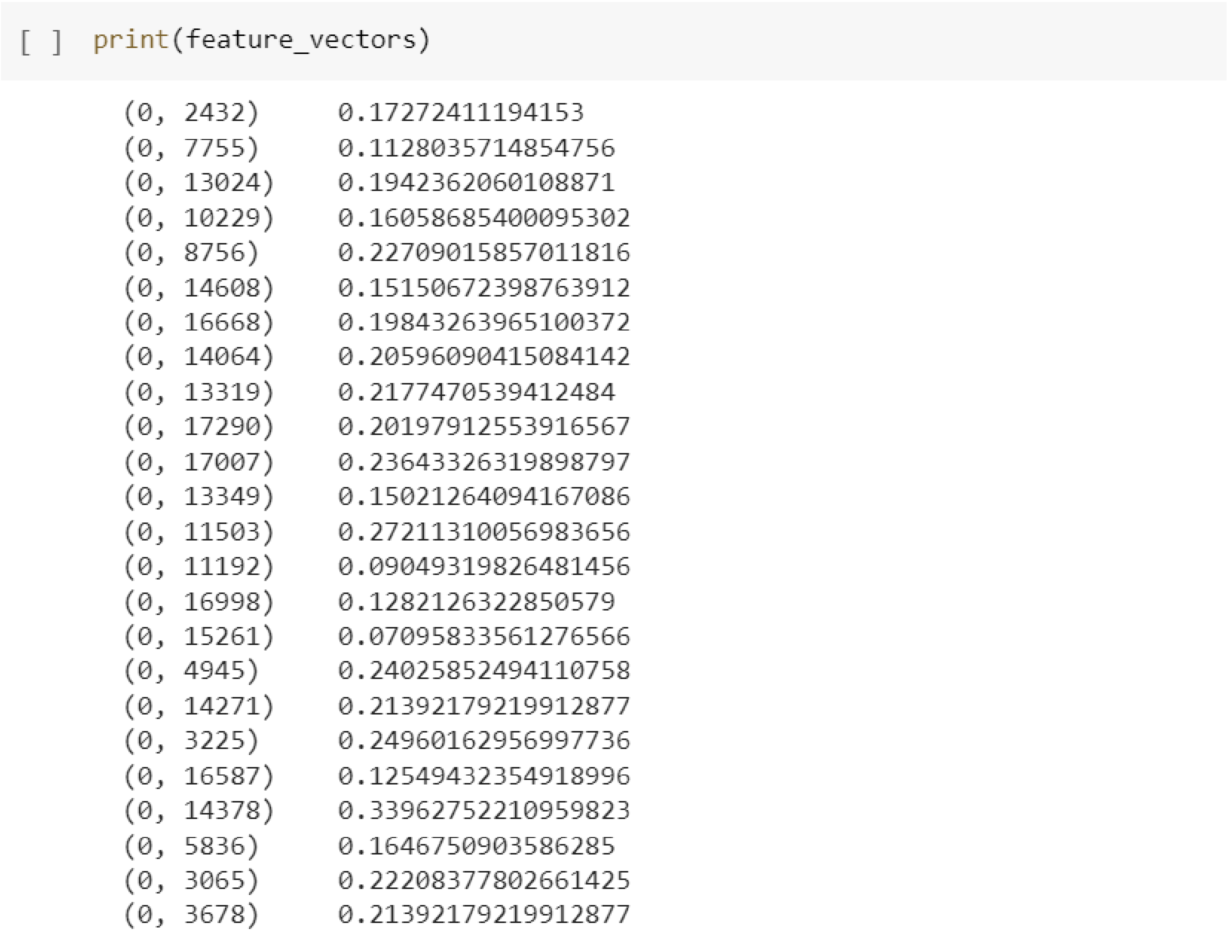
```
[ ]  print(combined_features)

     0       Action Adventure Fantasy Science Fiction cultu...
     1       Adventure Fantasy Action ocean drug abuse exot...
     2       Action Adventure Crime spy based on novel secr...
     3       Action Crime Drama Thriller dc comics crime fi...
     4       Action Adventure Science Fiction based on nove...
                                   ...
     4798    Action Crime Thriller united states\u2013mexic...
     4799    Comedy Romance  A newlywed couple's honeymoon ...
     4800    Comedy Drama Romance TV Movie date love at fir...
     4801      A New Yorker in Shanghai Daniel Henney Eliza...
     4802    Documentary obsession camcorder crush dream gi...
     Length: 4803, dtype: object
```



```
[ ]  # converting the text data to feature vectors

     vectorizer = TfidfVectorizer()
```

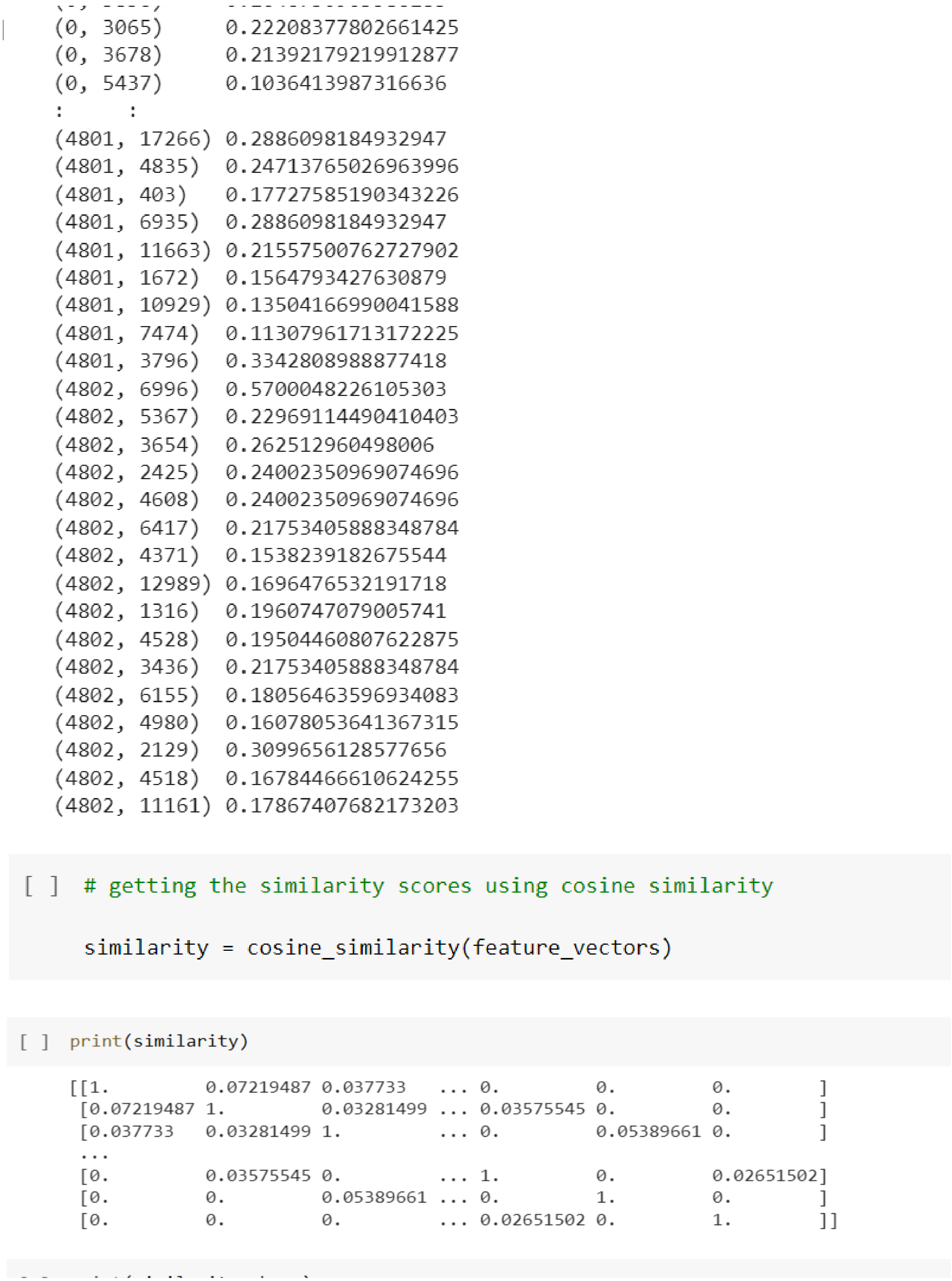


```
[ ]  feature_vectors = vectorizer.fit_transform(combined_features)
```



```
[ ]  print(feature_vectors)
```

```
  (0, 2432)	0.17272411194153
  (0, 7755)	0.1128035714854756
  (0, 13024)	0.1942362060108871
  (0, 10229)	0.16058685400095302
  (0, 8756)	0.22709015857011816
  (0, 14608)	0.15150672398763912
  (0, 16668)	0.19843263965100372
  (0, 14064)	0.20596090415084142
  (0, 13319)	0.2177470539412484
  (0, 17290)	0.20197912553916567
  (0, 17007)	0.23643326319898797
  (0, 13349)	0.15021264094167086
  (0, 11503)	0.27211310056983656
  (0, 11192)	0.09049319826481456
  (0, 16998)	0.1282126322850579
  (0, 15261)	0.07095833561276566
  (0, 4945)	0.24025852494110758
  (0, 14271)	0.21392179219912877
  (0, 3225)	0.24960162956997736
  (0, 16587)	0.12549432354918996
  (0, 14378)	0.33962752210959823
  (0, 5836)	0.1646750903586285
  (0, 3065)	0.22208377802661425
  (0, 3678)	0.21392179219912877
```

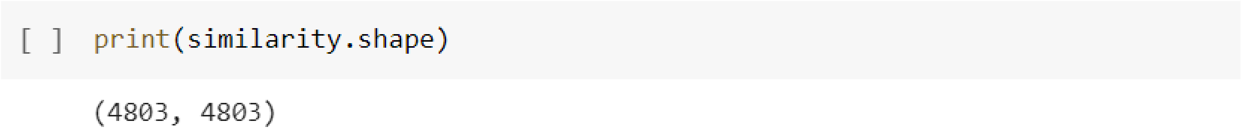


```
  (0, 3065)	0.22208377802661425
  (0, 3678)	0.21392179219912877
  (0, 5437)	0.1036413987316636
  :	:
  (4801, 17266)	0.2886098184932947
  (4801, 4835)	0.24713765026963996
  (4801, 403)	0.17727585190343226
  (4801, 6935)	0.2886098184932947
  (4801, 11663)	0.21557500762727902
  (4801, 1672)	0.1564793427630879
  (4801, 10929)	0.13504166990041588
  (4801, 7474)	0.11307961713172225
  (4801, 3796)	0.3342808988877418
  (4802, 6996)	0.5700048226105303
  (4802, 5367)	0.22969114490410403
  (4802, 3654)	0.262512960498006
  (4802, 2425)	0.24002350969074696
  (4802, 4608)	0.24002350969074696
  (4802, 6417)	0.21753405888348784
  (4802, 4371)	0.1538239182675544
  (4802, 12989)	0.1696476532191718
  (4802, 1316)	0.1960747079005741
  (4802, 4528)	0.19504460807622875
  (4802, 3436)	0.21753405888348784
  (4802, 6155)	0.18056463596934083
  (4802, 4980)	0.16078053641367315
  (4802, 2129)	0.3099656128577656
  (4802, 4518)	0.16784466610624255
  (4802, 11161)	0.17867407682173203
```

```
[ ]  # getting the similarity scores using cosine similarity

     similarity = cosine_similarity(feature_vectors)
```

```
[ ]  print(similarity)
```

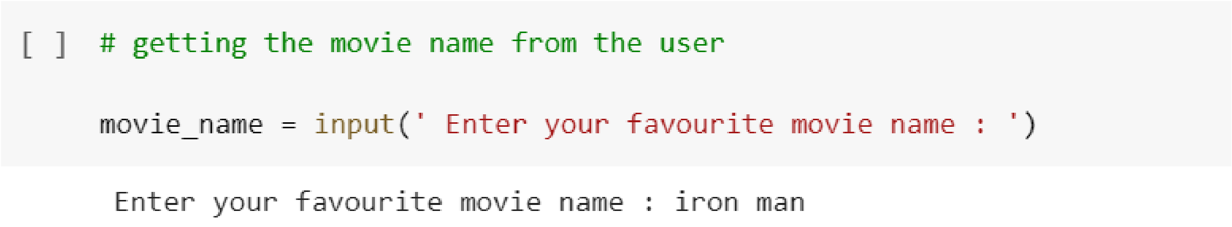
```
[[1.         0.07219487 0.037733   ... 0.         0.         0.        ]
 [0.07219487 1.         0.03281499 ... 0.03575545 0.         0.        ]
 [0.037733   0.03281499 1.         ... 0.         0.05389661 0.        ]
 ...
 [0.         0.03575545 0.         ... 1.         0.         0.02651502]
 [0.         0.         0.05389661 ... 0.         1.         0.        ]
 [0.         0.         0.         ... 0.02651502 0.         1.        ]]
```



```
print(similarity.shape)
```

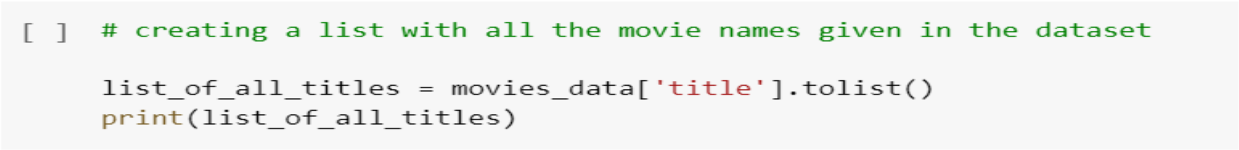
```
(4803, 4803)
```

Getting the movie name from the user

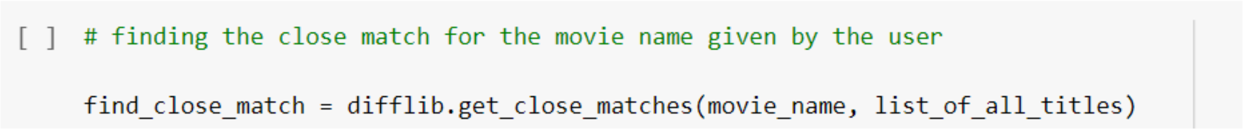


```
# getting the movie name from the user

movie_name = input(' Enter your favourite movie name : ')
```

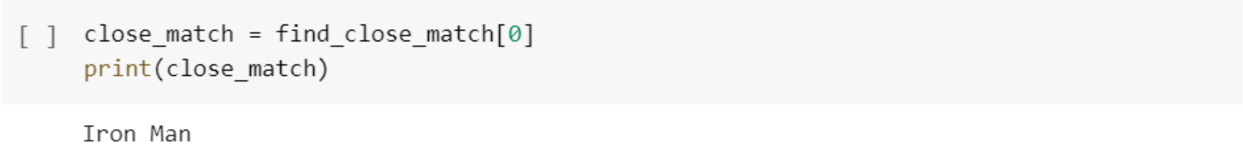
```
 Enter your favourite movie name : iron man
```



```
# creating a list with all the movie names given in the dataset

list_of_all_titles = movies_data['title'].tolist()
print(list_of_all_titles)
```

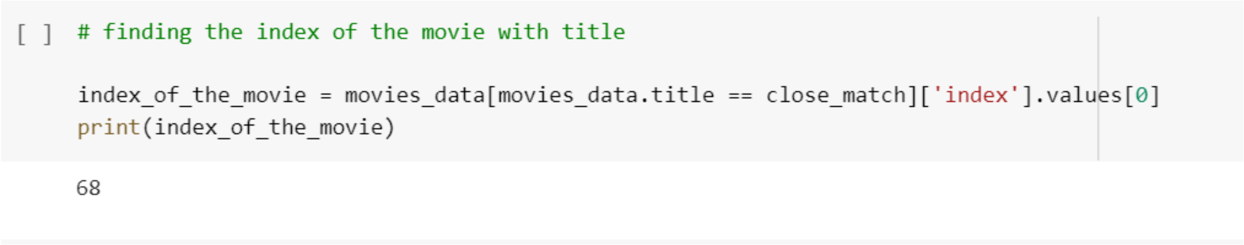


```
# finding the close match for the movie name given by the user

find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)
```



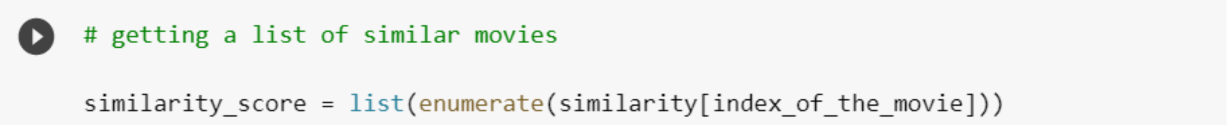
```
close_match = find_close_match[0]
print(close_match)
```

```
Iron Man
```

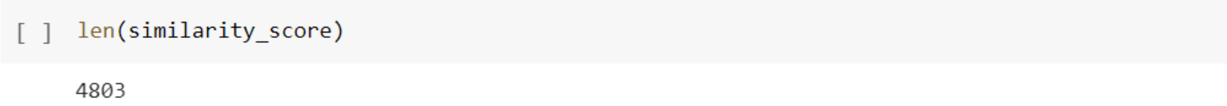


```
# finding the index of the movie with title

index_of_the_movie = movies_data[movies_data.title == close_match]['index'].values[0]
print(index_of_the_movie)
```

```
68
```

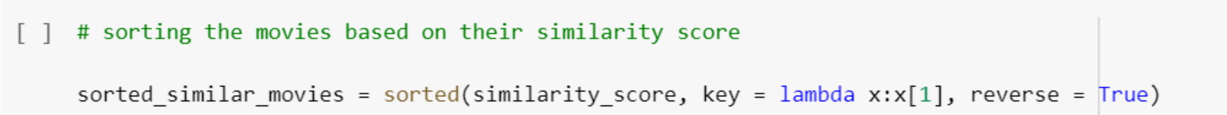


```python
# getting a list of similar movies

similarity_score = list(enumerate(similarity[index_of_the_movie]))
```

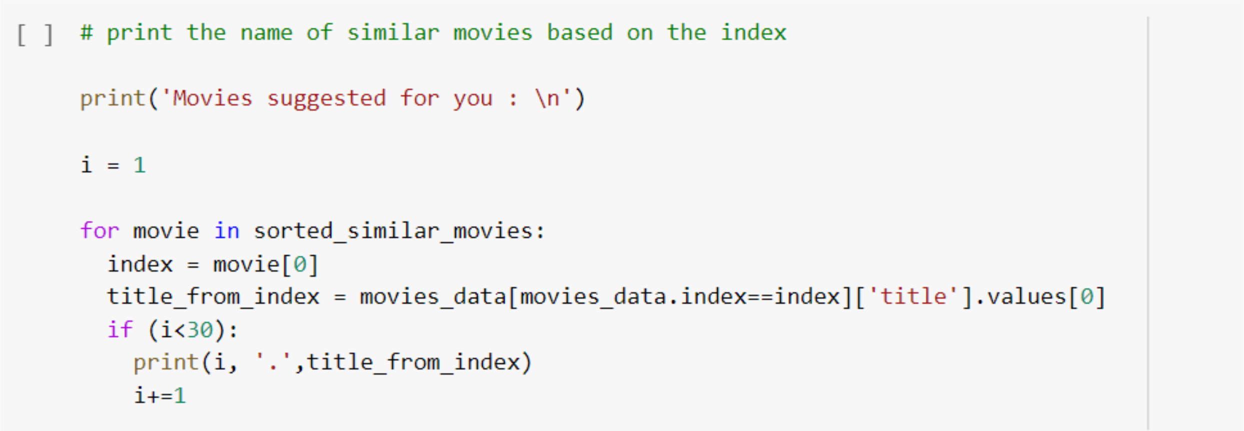


```python
len(similarity_score)
```

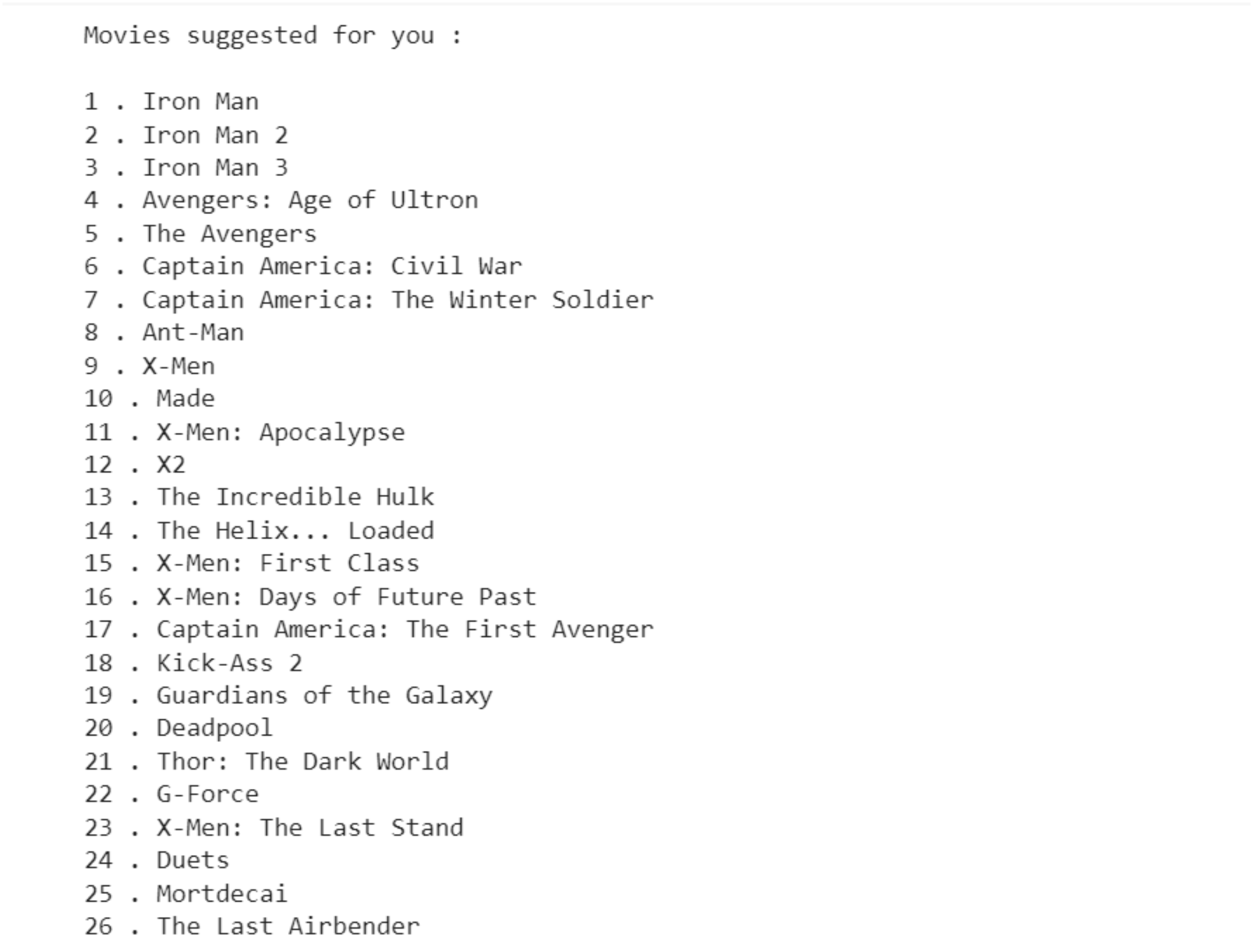
```
4803
```



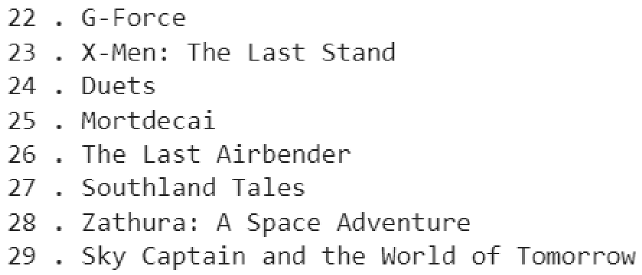
```python
# sorting the movies based on their similarity score

sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse = True)
```



```python
# print the name of similar movies based on the index

print('Movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
  index = movie[0]
  title_from_index = movies_data[movies_data.index==index]['title'].values[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
```

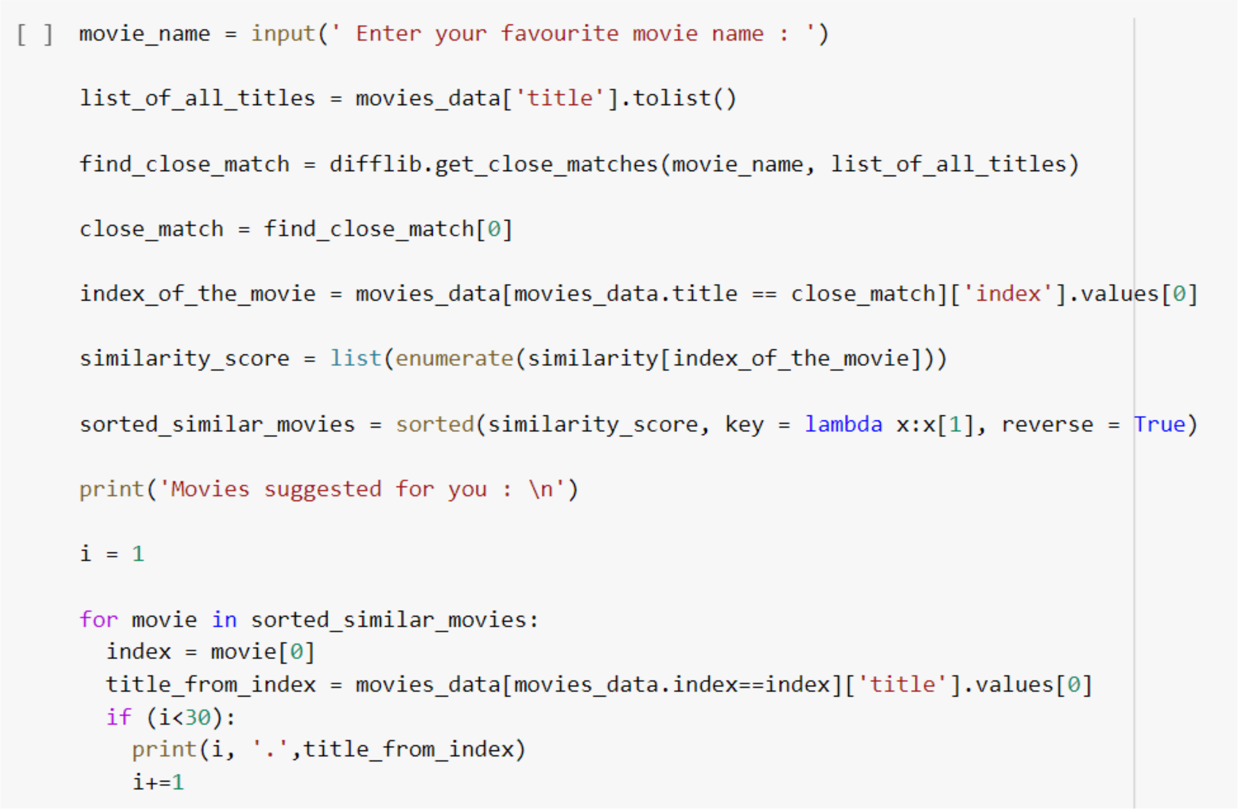


```
Movies suggested for you :

1 . Iron Man
2 . Iron Man 2
3 . Iron Man 3
4 . Avengers: Age of Ultron
5 . The Avengers
6 . Captain America: Civil War
7 . Captain America: The Winter Soldier
8 . Ant-Man
9 . X-Men
10 . Made
11 . X-Men: Apocalypse
12 . X2
13 . The Incredible Hulk
14 . The Helix... Loaded
15 . X-Men: First Class
16 . X-Men: Days of Future Past
17 . Captain America: The First Avenger
18 . Kick-Ass 2
19 . Guardians of the Galaxy
20 . Deadpool
21 . Thor: The Dark World
22 . G-Force
23 . X-Men: The Last Stand
24 . Duets
25 . Mortdecai
26 . The Last Airbender
```

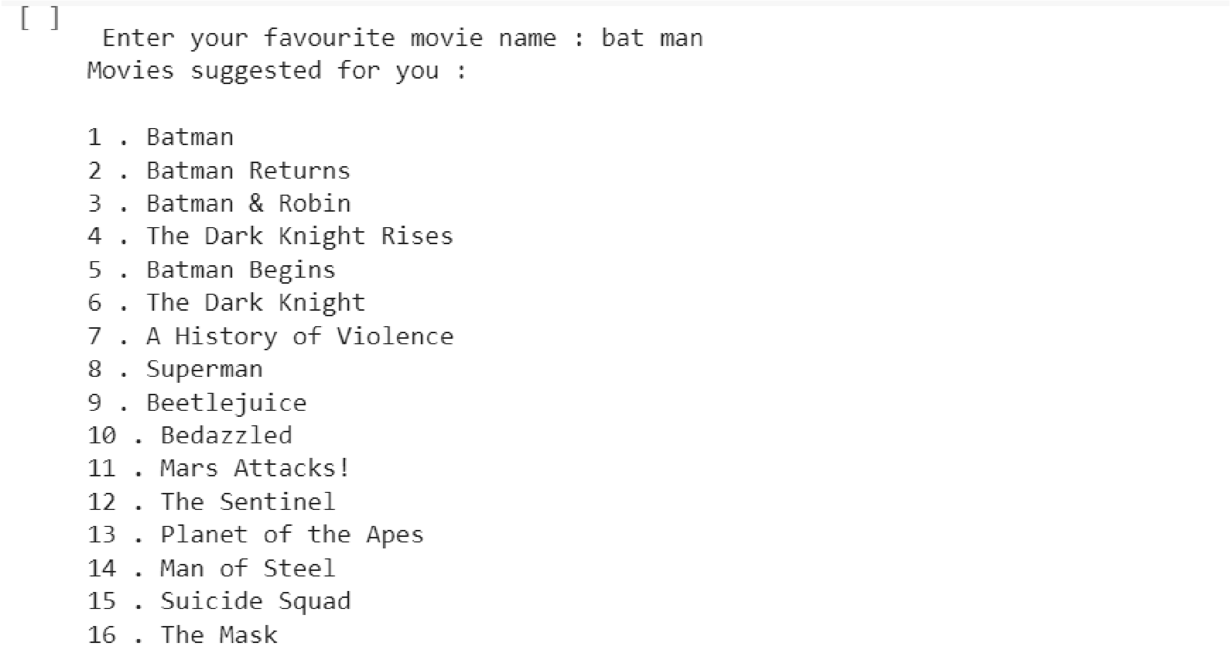


```
22 . G-Force
23 . X-Men: The Last Stand
24 . Duets
25 . Mortdecai
26 . The Last Airbender
27 . Southland Tales
28 . Zathura: A Space Adventure
29 . Sky Captain and the World of Tomorrow
```

Movie Recommendation Sytem

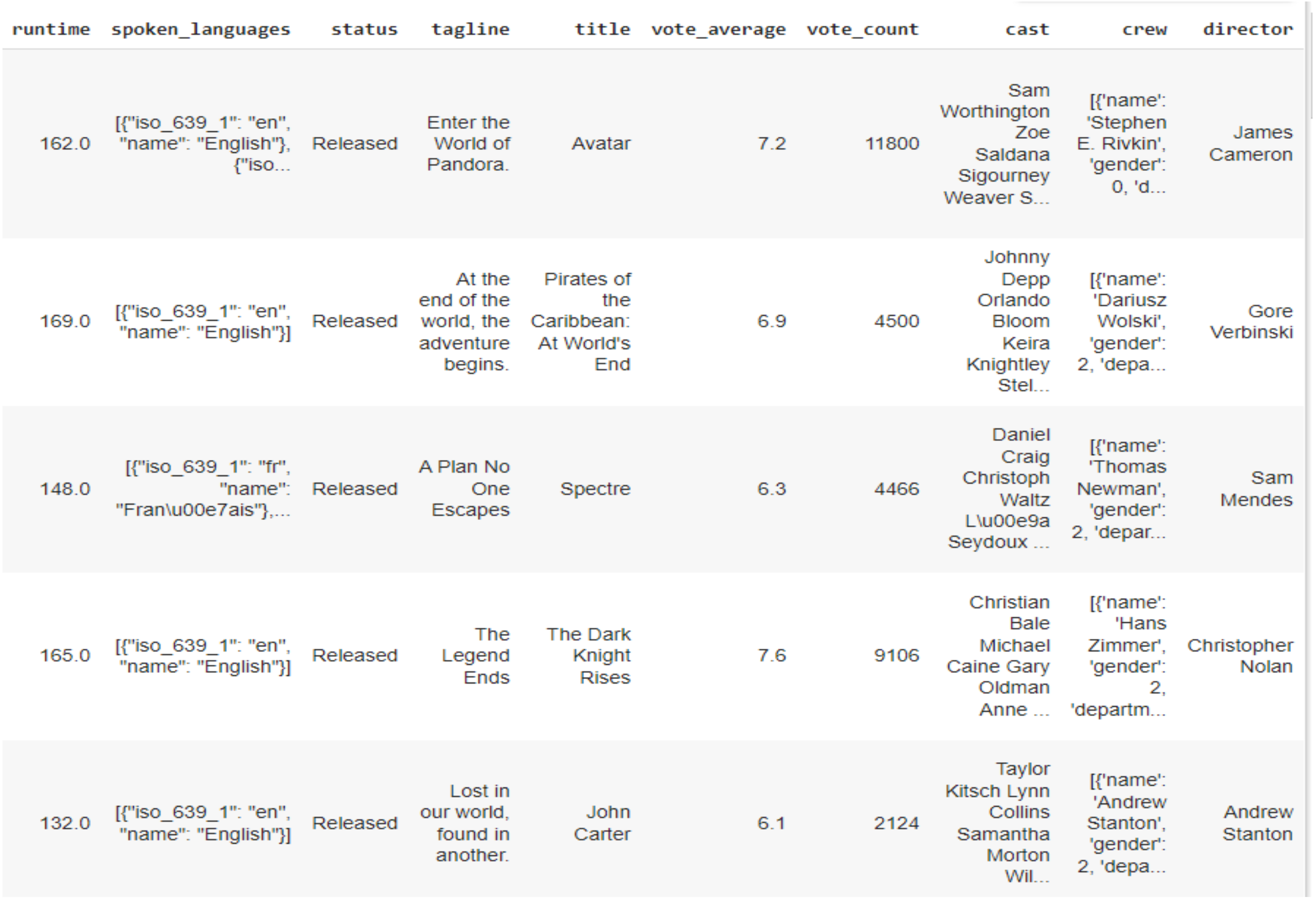


```
movie_name = input(' Enter your favourite movie name : ')

list_of_all_titles = movies_data['title'].tolist()

find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)

close_match = find_close_match[0]

index_of_the_movie = movies_data[movies_data.title == close_match]['index'].values[0]

similarity_score = list(enumerate(similarity[index_of_the_movie]))

sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse = True)

print('Movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
  index = movie[0]
  title_from_index = movies_data[movies_data.index==index]['title'].values[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
```



```
 Enter your favourite movie name : bat man
Movies suggested for you :

1 . Batman
2 . Batman Returns
3 . Batman & Robin
4 . The Dark Knight Rises
5 . Batman Begins
6 . The Dark Knight
7 . A History of Violence
8 . Superman
9 . Beetlejuice
10 . Bedazzled
11 . Mars Attacks!
12 . The Sentinel
13 . Planet of the Apes
14 . Man of Steel
15 . Suicide Squad
16 . The Mask
```

```
16 . The Mask
17 . Salton Sea
18 . Spider-Man 3
19 . The Postman Always Rings Twice
20 . Hang 'em High
21 . Spider-Man 2
22 . Dungeons & Dragons: Wrath of the Dragon God
23 . Superman Returns
24 . Jonah Hex
25 . Exorcist II: The Heretic
26 . Superman II
27 . Green Lantern
28 . Superman III
29 . Something's Gotta Give
```



runtime	spoken_languages	status	tagline	title	vote_average	vote_count	cast	crew	director
162.0	[{"iso_639_1": "en", "name": "English"}, {"iso...	Released	Enter the World of Pandora.	Avatar	7.2	11800	Sam Worthington Zoe Saldana Sigourney Weaver S...	[{'name': 'Stephen E. Rivkin', 'gender': 0, 'd...	James Cameron
169.0	[{"iso_639_1": "en", "name": "English"}]	Released	At the end of the world, the adventure begins.	Pirates of the Caribbean: At World's End	6.9	4500	Johnny Depp Orlando Bloom Keira Knightley Stel...	[{'name': 'Dariusz Wolski', 'gender': 2, 'depa...	Gore Verbinski
148.0	[{"iso_639_1": "fr", "name": "Fran\u00e7ais"},...	Released	A Plan No One Escapes	Spectre	6.3	4466	Daniel Craig Christoph Waltz L\u00e9a Seydoux ...	[{'name': 'Thomas Newman', 'gender': 2, 'depar...	Sam Mendes
165.0	[{"iso_639_1": "en", "name": "English"}]	Released	The Legend Ends	The Dark Knight Rises	7.6	9106	Christian Bale Michael Caine Gary Oldman Anne ...	[{'name': 'Hans Zimmer', 'gender': 2, 'departm...	Christopher Nolan
132.0	[{"iso_639_1": "en", "name": "English"}]	Released	Lost in our world, found in another.	John Carter	6.1	2124	Taylor Kitsch Lynn Collins Samantha Morton Wil...	[{'name': 'Andrew Stanton', 'gender': 2, 'depa...	Andrew Stanton

7. PERFORMANCE ANALYSIS

According to many researches, we come to know that for evaluating performance of a movie recommendation system only accuracy is not the criteria but we should also consider other important parameters. Recommendation quality is also an equally significant metric. Performance Evaluation Metrics can be divided into two parts :

Evaluation Metrics based on the recommendation algorithm

Evaluation Metrics independent from the recommendation algorithm

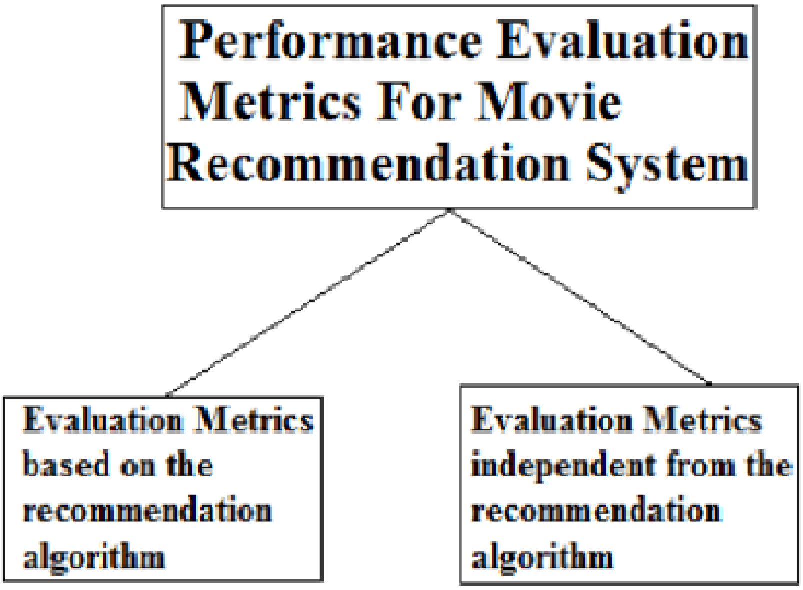

Performance Evaluation Metrics For Movie Recommendation System
Evaluation Metrics based on the recommendation algorithm
Evaluation Metrics independent from the recommendation algorithm

Figure 1: Performance Evaluation Metrics

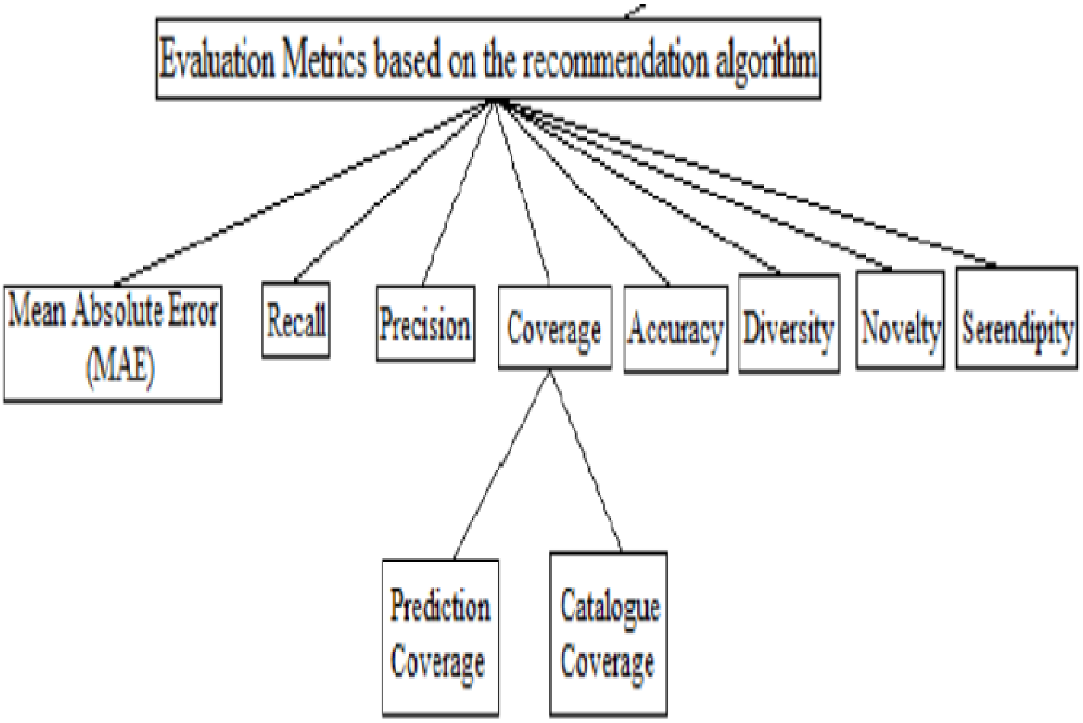
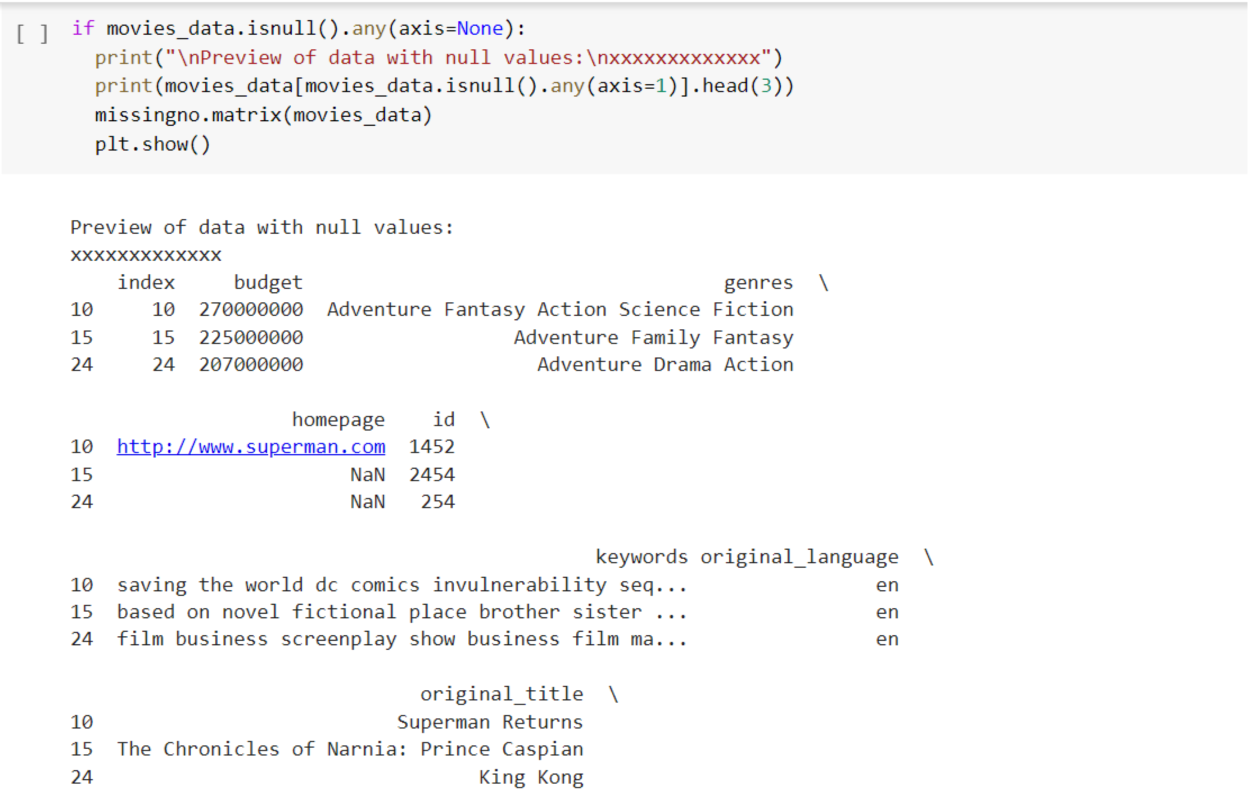


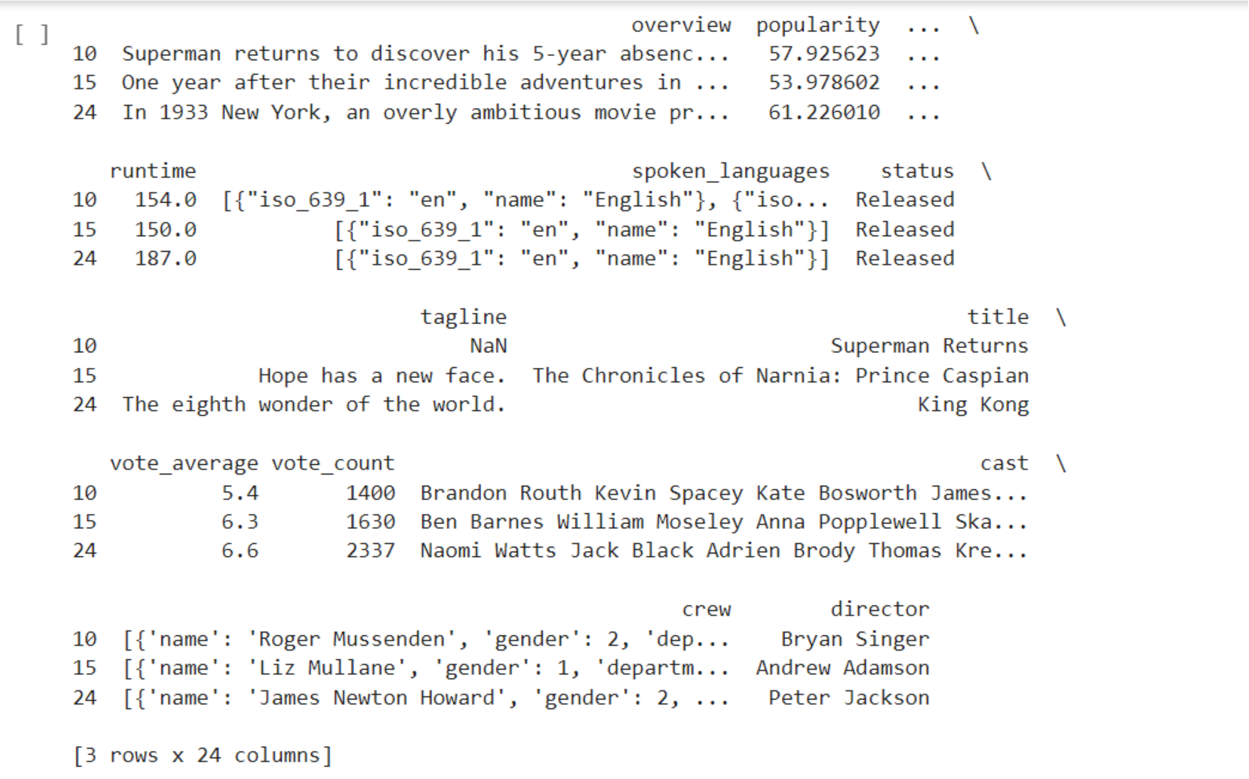
Figure 2: Evaluation Metrics based on the recommendation algorithm

8. DATA VISUALIZATION

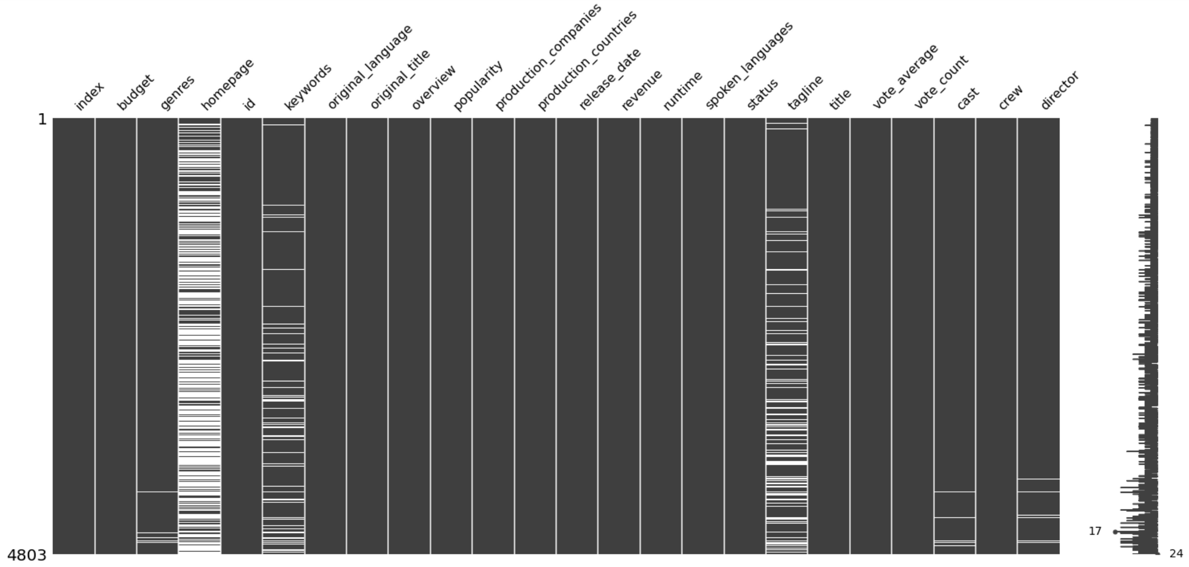


```
[ ]  if movies_data.isnull().any(axis=None):
       print("\nPreview of data with null values:\nxxxxxxxxxxxxx")
       print(movies_data[movies_data.isnull().any(axis=1)].head(3))
       missingno.matrix(movies_data)
       plt.show()
```

```
Preview of data with null values:
xxxxxxxxxxxxx
    index     budget                                   genres  \
10     10  270000000  Adventure Fantasy Action Science Fiction
15     15  225000000                 Adventure Family Fantasy
24     24  207000000                   Adventure Drama Action

                  homepage    id  \
10  http://www.superman.com  1452
15                     NaN  2454
24                     NaN   254

                                             keywords original_language  \
10  saving the world dc comics invulnerability seq...                en
15  based on novel fictional place brother sister ...                en
24  film business screenplay show business film ma...                en

                             original_title  \
10                          Superman Returns
15  The Chronicles of Narnia: Prince Caspian
24                                 King Kong
```

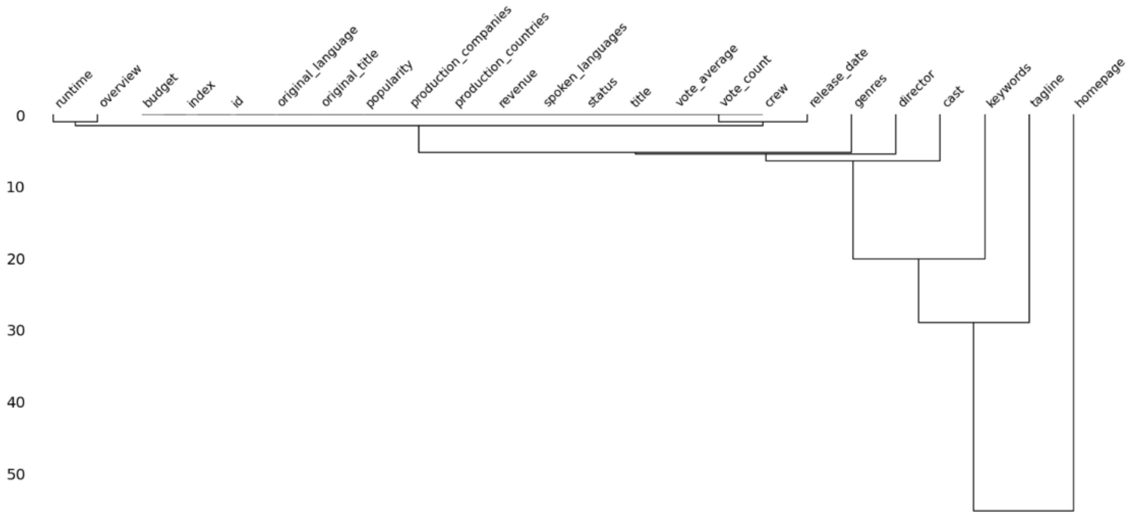


```
[ ]                                         overview  popularity  ...  \
    10  Superman returns to discover his 5-year absenc...   57.925623  ...
    15  One year after their incredible adventures in ...   53.978602  ...
    24  In 1933 New York, an overly ambitious movie pr...   61.226010  ...

       runtime                                   spoken_languages    status  \
    10   154.0  [{"iso_639_1": "en", "name": "English"}, {"iso...  Released
    15   150.0           [{"iso_639_1": "en", "name": "English"}]  Released
    24   187.0           [{"iso_639_1": "en", "name": "English"}]  Released

                               tagline                                     title  \
    10                             NaN                          Superman Returns
    15             Hope has a new face.  The Chronicles of Narnia: Prince Caspian
    24  The eighth wonder of the world.                                 King Kong

       vote_average vote_count                                               cast  \
    10          5.4       1400  Brandon Routh Kevin Spacey Kate Bosworth James...
    15          6.3       1630  Ben Barnes William Moseley Anna Popplewell Ska...
    24          6.6       2337  Naomi Watts Jack Black Adrien Brody Thomas Kre...

                                                     crew         director
    10  [{'name': 'Roger Mussenden', 'gender': 2, 'dep...    Bryan Singer
    15  [{'name': 'Liz Mullane', 'gender': 1, 'departm...  Andrew Adamson
    24  [{'name': 'James Newton Howard', 'gender': 2, ...    Peter Jackson

    [3 rows x 24 columns]
```

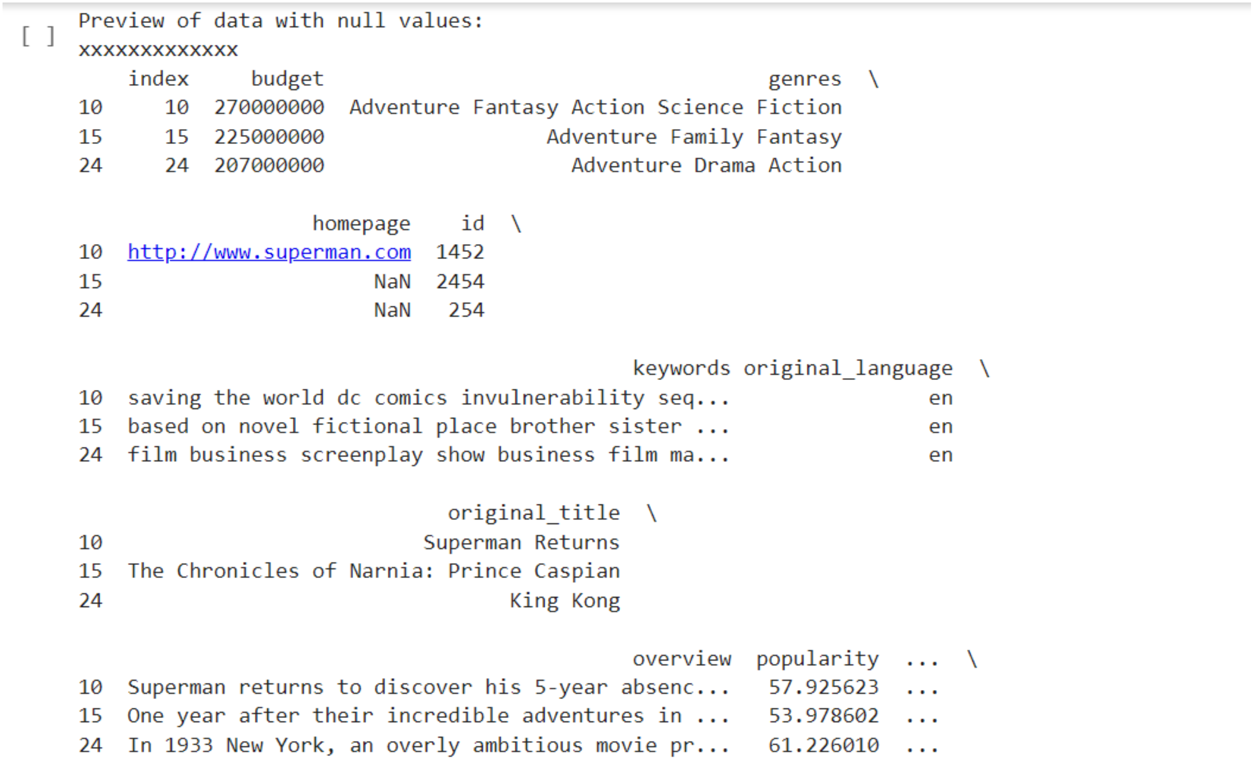


```
[ ]  if movies_data.isnull().any(axis=None):
       print("\nPreview of data with null values:\nxxxxxxxxxxxxx")
       print(movies_data[movies_data.isnull().any(axis=1)].head(3))
       missingno.dendrogram(movies_data)
       plt.show()
```

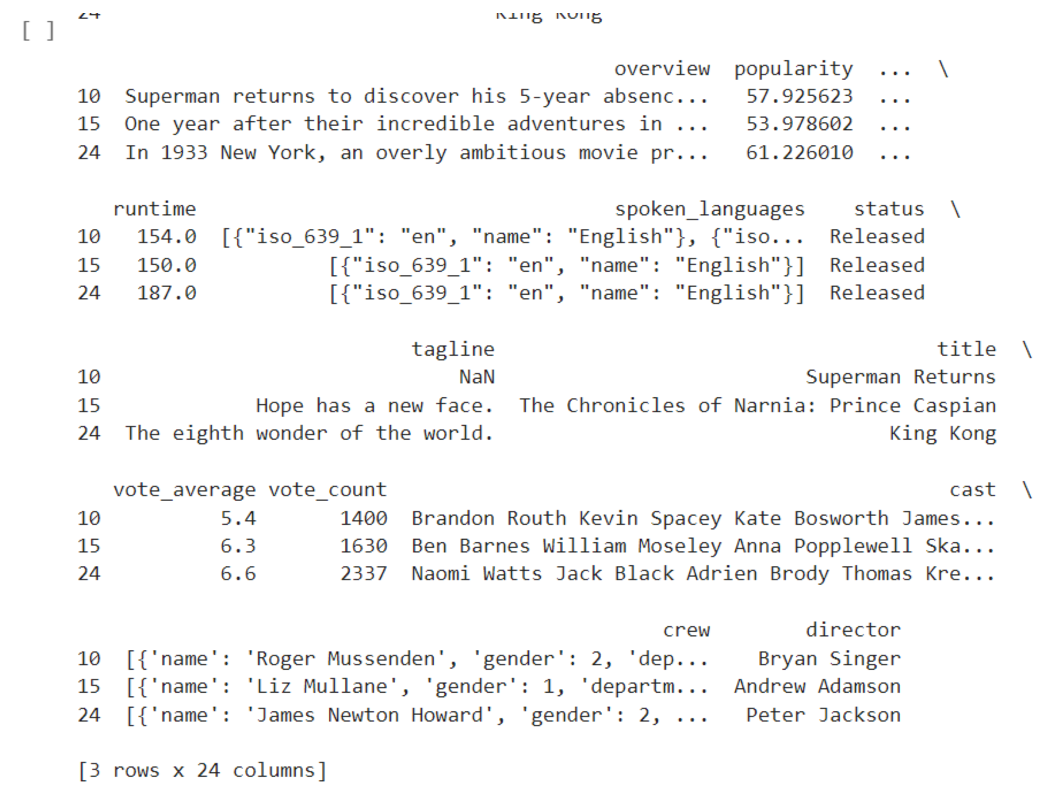
```
[ ]
    Preview of data with null values:
    xxxxxxxxxxxxx
        index     budget                                    genres  \
    10     10  270000000  Adventure Fantasy Action Science Fiction
    15     15  225000000                   Adventure Family Fantasy
    24     24  207000000                     Adventure Drama Action

                       homepage    id  \
    10  http://www.superman.com  1452
    15                      NaN  2454
    24                      NaN   254

                                                 keywords original_language  \
    10  saving the world dc comics invulnerability seq...                en
    15  based on novel fictional place brother sister ...                en
    24  film business screenplay show business film ma...                en

                                  original_title  \
    10                          Superman Returns
    15  The Chronicles of Narnia: Prince Caspian
    24                                 King Kong

                                                 overview  popularity  ...  \
    10  Superman returns to discover his 5-year absenc...   57.925623  ...
    15  One year after their incredible adventures in ...   53.978602  ...
    15  One year after their incredible adventures in ...   53.978602  ...
[ ] 24  In 1933 New York, an overly ambitious movie pr...   61.226010  ...

       runtime                                    spoken_languages    status  \
    10   154.0  [{"iso_639_1": "en", "name": "English"}, {"iso...  Released
    15   150.0           [{"iso_639_1": "en", "name": "English"}]  Released
    24   187.0           [{"iso_639_1": "en", "name": "English"}]  Released

                            tagline                                    title  \
    10                          NaN                         Superman Returns
    15           Hope has a new face.  The Chronicles of Narnia: Prince Caspian
    24  The eighth wonder of the world.                                King Kong

       vote_average vote_count                                             cast  \
    10          5.4       1400  Brandon Routh Kevin Spacey Kate Bosworth James...
    15          6.3       1630  Ben Barnes William Moseley Anna Popplewell Ska...
    24          6.6       2337  Naomi Watts Jack Black Adrien Brody Thomas Kre...

                                                     crew         director
    10  [{'name': 'Roger Mussenden', 'gender': 2, 'dep...     Bryan Singer
    15  [{'name': 'Liz Mullane', 'gender': 1, 'departm...   Andrew Adamson
    24  [{'name': 'James Newton Howard', 'gender': 2, ...    Peter Jackson

    [3 rows x 24 columns]
```



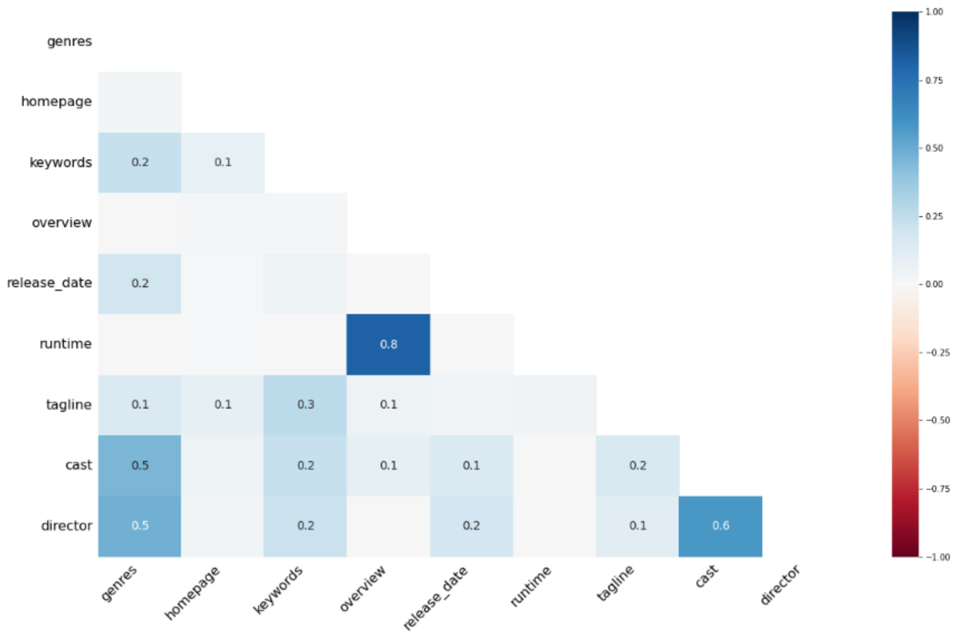
```
[ ] if movies_data.isnull().any(axis=None):
      print("\nPreview of data with null values:\nxxxxxxxxxxxxx")
      print(movies_data[movies_data.isnull().any(axis=1)].head(3))
      missingno.heatmap(movies_data)
      plt.show()
```



```
[ ]  Preview of data with null values:
     xxxxxxxxxxxxx
         index     budget                                    genres  \
     10     10  270000000  Adventure Fantasy Action Science Fiction
     15     15  225000000                  Adventure Family Fantasy
     24     24  207000000                    Adventure Drama Action

                        homepage    id  \
     10  http://www.superman.com  1452
     15                      NaN  2454
     24                      NaN   254

                                                  keywords original_language  \
     10  saving the world dc comics invulnerability seq...                en
     15  based on novel fictional place brother sister ...                en
     24  film business screenplay show business film ma...                en

                                    original_title  \
     10                            Superman Returns
     15  The Chronicles of Narnia: Prince Caspian
     24                                   King Kong

                                                  overview  popularity  ...  \
     10  Superman returns to discover his 5-year absenc...   57.925623  ...
     15  One year after their incredible adventures in ...   53.978602  ...
     24  In 1933 New York, an overly ambitious movie pr...   61.226010  ...
```



```
[ ]  24                                  King Kong

                                                  overview  popularity  ...  \
     10  Superman returns to discover his 5-year absenc...   57.925623  ...
     15  One year after their incredible adventures in ...   53.978602  ...
     24  In 1933 New York, an overly ambitious movie pr...   61.226010  ...

        runtime                                   spoken_languages    status  \
     10   154.0  [{"iso_639_1": "en", "name": "English"}, {"iso...  Released
     15   150.0           [{"iso_639_1": "en", "name": "English"}]  Released
     24   187.0           [{"iso_639_1": "en", "name": "English"}]  Released

                              tagline                                     title  \
     10                           NaN                          Superman Returns
     15             Hope has a new face.  The Chronicles of Narnia: Prince Caspian
     24  The eighth wonder of the world.                                 King Kong

        vote_average vote_count                                               cast  \
     10          5.4       1400  Brandon Routh Kevin Spacey Kate Bosworth James...
     15          6.3       1630  Ben Barnes William Moseley Anna Popplewell Ska...
     24          6.6       2337  Naomi Watts Jack Black Adrien Brody Thomas Kre...

                                                      crew        director
     10  [{'name': 'Roger Mussenden', 'gender': 2, 'dep...    Bryan Singer
     15  [{'name': 'Liz Mullane', 'gender': 1, 'departm...  Andrew Adamson
     24  [{'name': 'James Newton Howard', 'gender': 2, ...   Peter Jackson

     [3 rows x 24 columns]
```



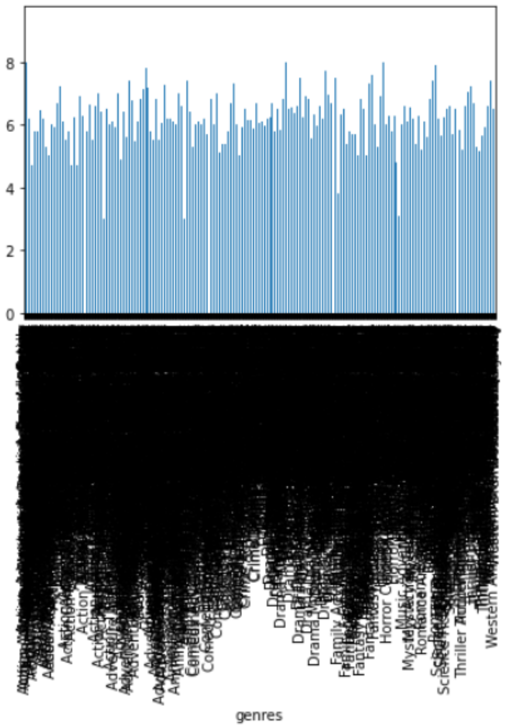
```
[ ]  display(movies_data.describe().T)
```

	count	mean	std	min	25%	50%	75%	max
index	4803.0	2.401000e+03	1.386651e+03	0.0	1200.50000	2.401000e+03	3.601500e+03	4.802000e+03
budget	4803.0	2.904504e+07	4.072239e+07	0.0	790000.00000	1.500000e+07	4.000000e+07	3.800000e+08
id	4803.0	5.716548e+04	8.869461e+04	5.0	9014.50000	1.462900e+04	5.861050e+04	4.594880e+05
popularity	4803.0	2.149230e+01	3.181665e+01	0.0	4.66807	1.292159e+01	2.831350e+01	8.755813e+02
revenue	4803.0	8.226064e+07	1.628571e+08	0.0	0.00000	1.917000e+07	9.291719e+07	2.787965e+09
runtime	4801.0	1.068759e+02	2.261193e+01	0.0	94.00000	1.030000e+02	1.180000e+02	3.380000e+02
vote_average	4803.0	6.092172e+00	1.194612e+00	0.0	5.60000	6.200000e+00	6.800000e+00	1.000000e+01
vote_count	4803.0	6.902180e+02	1.234586e+03	0.0	54.00000	2.350000e+02	7.370000e+02	1.375200e+04

```
[ ]  print(movies_data['vote_average'].unique())
```

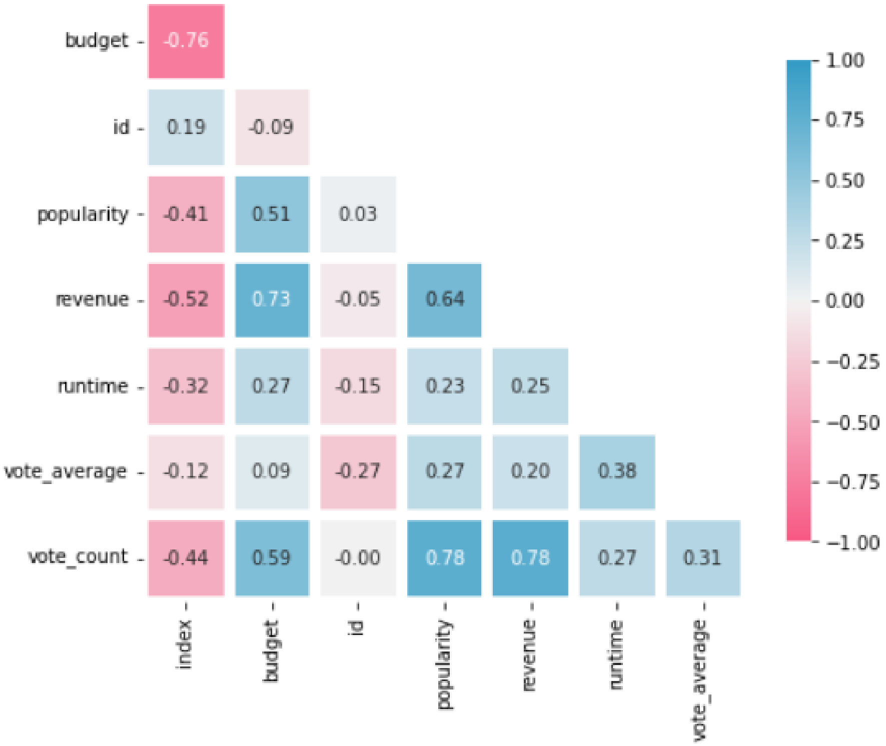
```
[ 7.2  6.9  6.3  7.6  6.1  5.9  7.4  7.3  5.7  5.4  7.   6.5  6.4  6.2
  7.1  5.8  6.6  7.5  5.5  6.7  6.8  6.   5.1  7.8  5.6  5.2  8.2  7.7
  5.3  8.   4.8  4.9  7.9  8.1  4.7  5.   4.2  4.4  4.1  3.7  3.6  3.
  3.9  4.3  4.5  3.4  4.6  8.3  3.5  4.   2.3  3.2  0.   3.8  2.9  8.5
  1.9  3.1  3.3  2.2  0.5  9.3  8.4  2.7 10.   1.   2.   2.8  9.5  2.6
  2.4]
```

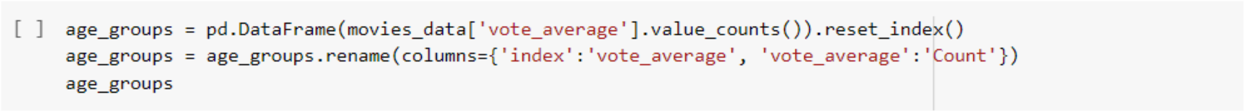
```
[ ] movies_data.groupby('genres')['vote_average'].mean().plot.bar()
    plt.show()
```



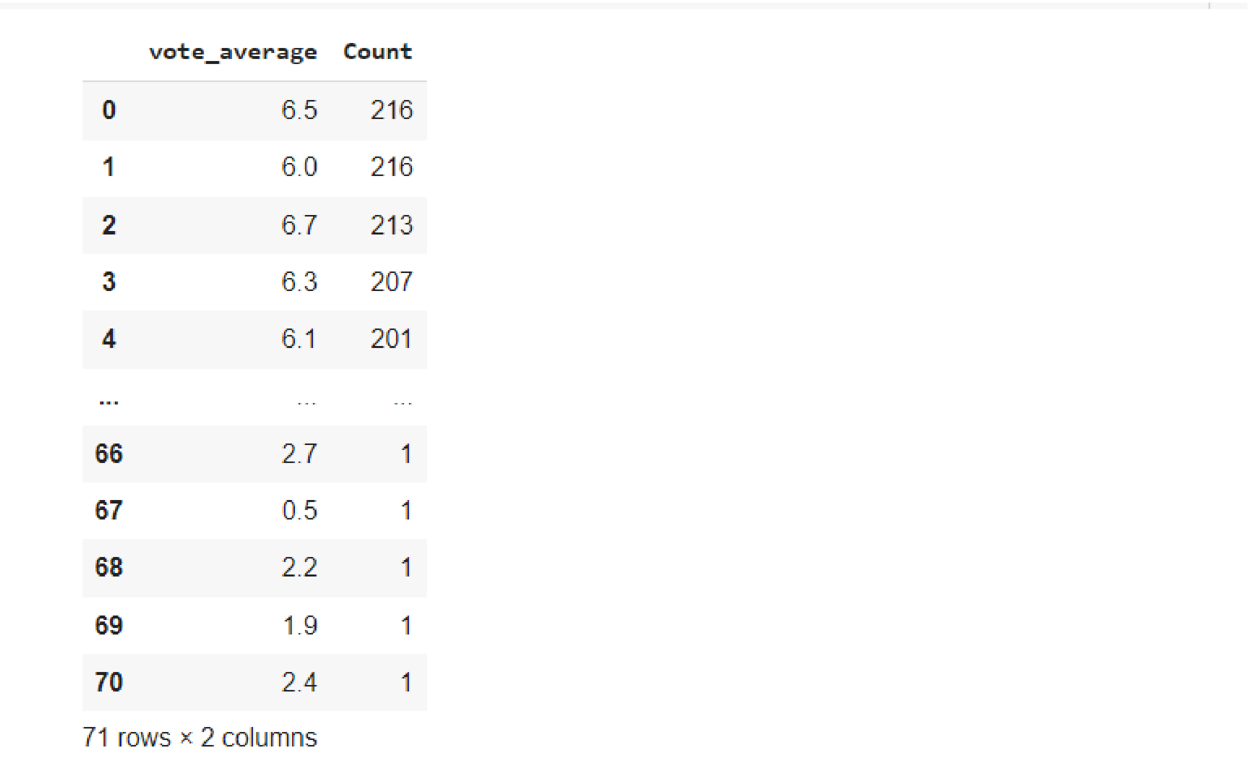
EXPLORATORY DATA ANALYSIS WITH THE HELP OF VISUALIZATIONS

```
[ ] data_corr = movies_data.corr()
    fig, ax = plt.subplots(figsize=(8, 6))

    # mask
    mask = np.triu(np.ones_like(data_corr, dtype=np.bool))

    # adjust mask and df
    mask = mask[1:, :-1]
    corr = data_corr.iloc[1:,:-1].copy()

    # color map
    cmap = sns.diverging_palette(0, 230, 90, 60, as_cmap=True)

    # plot heatmap
    sns.heatmap(corr, mask=mask, annot=True, fmt=".2f",
                linewidths=5, cmap=cmap, vmin=-1, vmax=1,
                cbar_kws={"shrink": .8}, square=True)

    # ticks
    yticks = [i.upper() for i in corr.index]
    xticks = [i.upper() for i in corr.columns]
```



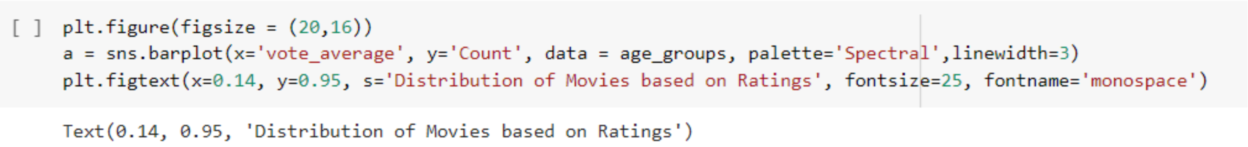


```
[ ]  age_groups = pd.DataFrame(movies_data['vote_average'].value_counts()).reset_index()
     age_groups = age_groups.rename(columns={'index':'vote_average', 'vote_average':'Count'})
     age_groups
```



	vote_average	Count
0	6.5	216
1	6.0	216
2	6.7	213
3	6.3	207
4	6.1	201
...	...	...
66	2.7	1
67	0.5	1
68	2.2	1
69	1.9	1
70	2.4	1

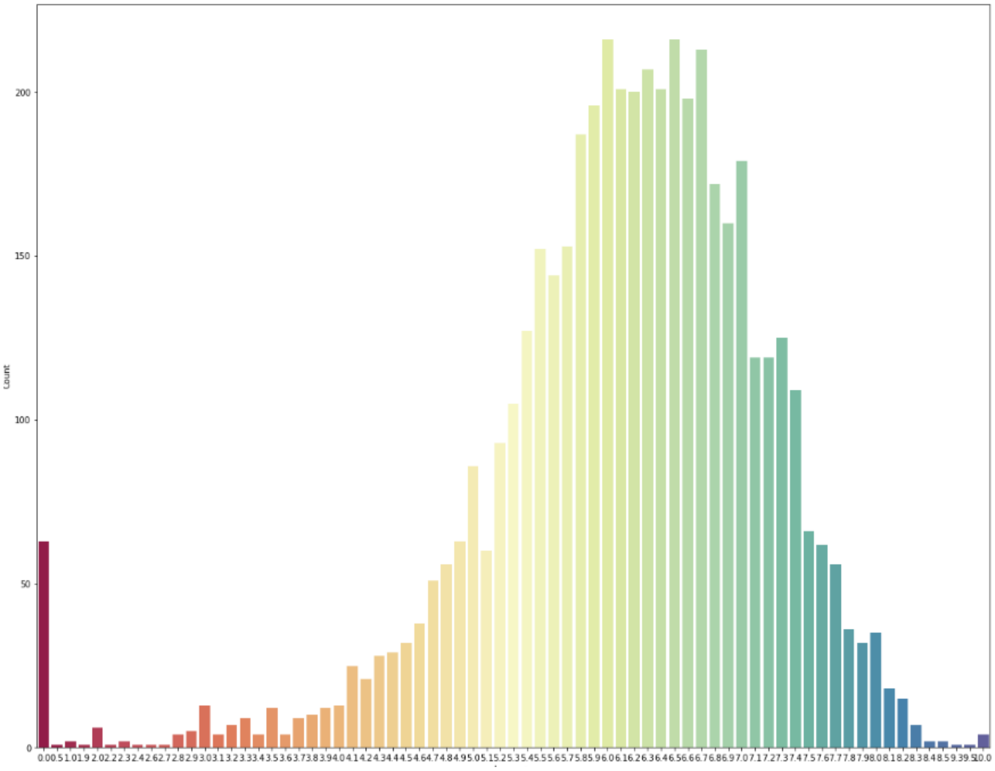
71 rows × 2 columns



```
[ ] plt.figure(figsize = (20,16))
    a = sns.barplot(x='vote_average', y='Count', data = age_groups, palette='Spectral',linewidth=3)
    plt.figtext(x=0.14, y=0.95, s='Distribution of Movies based on Ratings', fontsize=25, fontname='monospace')
```

```
Text(0.14, 0.95, 'Distribution of Movies based on Ratings')
```

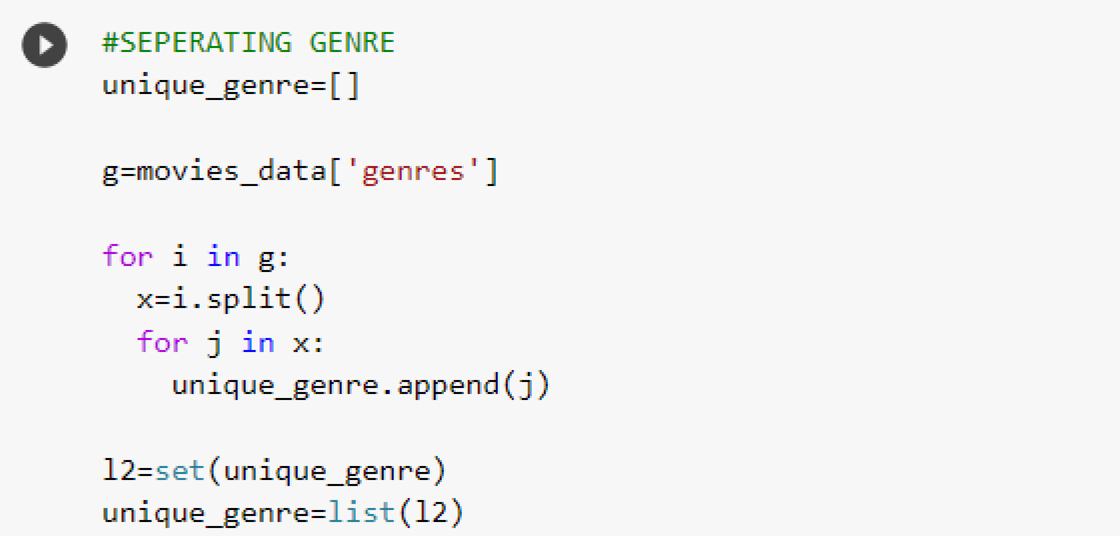
```
Distribution of Movies based on Ratings
```



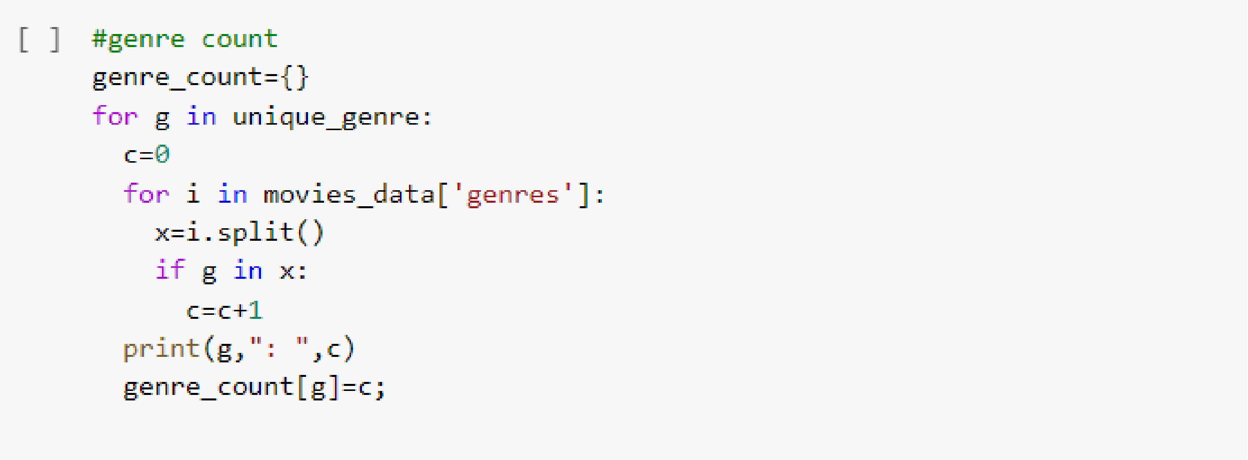
GENRE BASED ANALYSIS

for i in movies_data['genres']: x=i.split() for j in x: unique_genre.append(j)

l2=set(unique_genre) unique_genre=list(l2) print(unique_genre)



```
#SEPERATING GENRE
unique_genre=[]

g=movies_data['genres']

for i in g:
  x=i.split()
  for j in x:
    unique_genre.append(j)

l2=set(unique_genre)
unique_genre=list(l2)
```

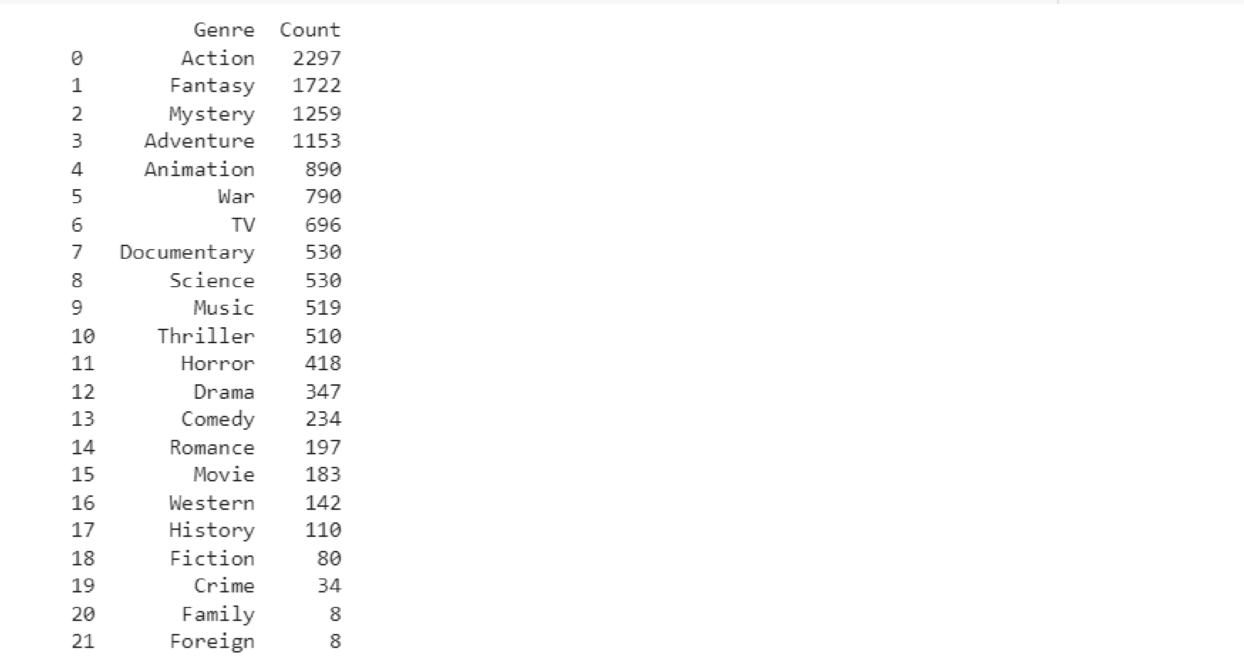


```
#genre count
genre_count={}
for g in unique_genre:
  c=0
  for i in movies_data['genres']:
    x=i.split()
    if g in x:
      c=c+1
  print(g,": ",c)
  genre_count[g]=c;
```

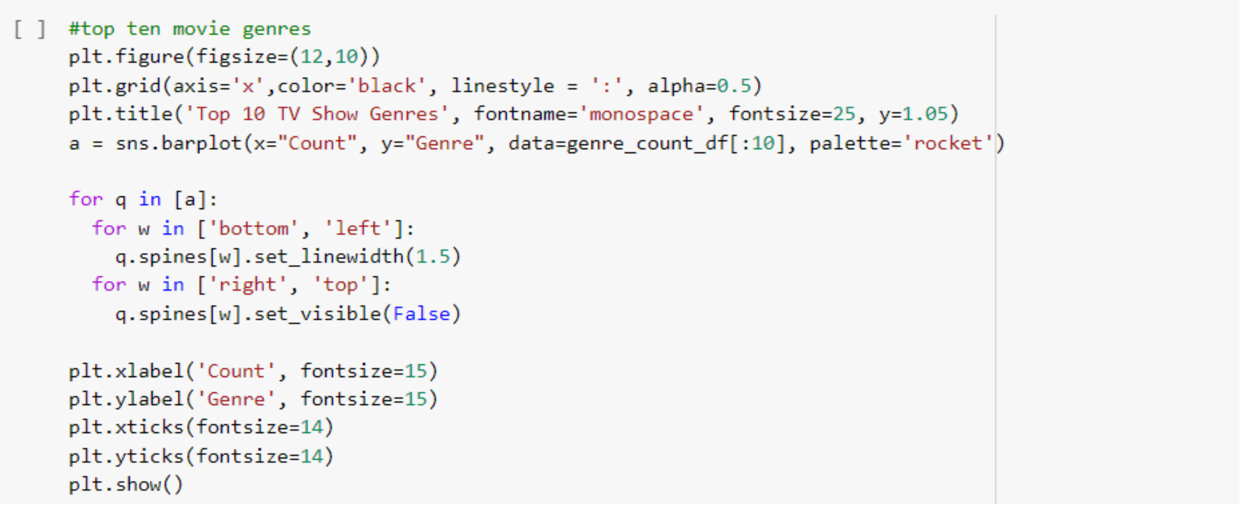
```
Action :  1153
Fantasy :  418
Mystery :  347
Adventure :  790
Animation :  234
War :  142
TV :  8
Documentary :  110
Science :  530
Music :  183
Thriller :  1259
Horror :  519
Drama :  2297
Comedy :  1722
Romance :  890
Movie :  8
Western :  80
History :  197
Fiction :  530
Crime :  696
Family :  510
Foreign :  34
```

[ ]

```python
#sorting most popular genres
sorted_value_index = np.argsort(genre_count.values())
dictionary_keys = list(genre_count.keys())
sorted_genre_count = {dictionary_keys[i]: sorted(
    genre_count.values(), reverse=True)[i] for i in range(len(dictionary_keys))}

#genre_count_df = pd.DataFrame.from_dict(sorted_genre_count, orient ='index')
sorted_count=list(sorted_genre_count.values())
sorted_genre=list(sorted_genre_count.keys())

genre_count_df=pd.DataFrame(list(zip(sorted_genre,sorted_count)))
genre_count_df.columns=['Genre','Count']
print(genre_count_df)
```

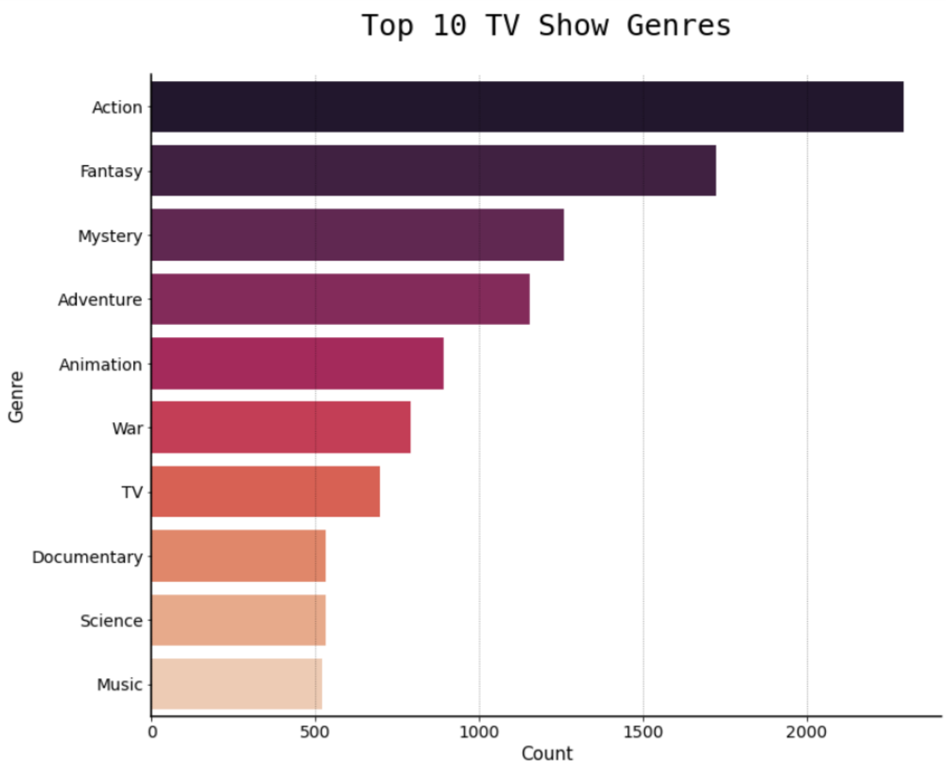


```
           Genre  Count
0         Action   2297
1        Fantasy   1722
2        Mystery   1259
3      Adventure   1153
4      Animation    890
5            War    790
6             TV    696
7    Documentary    530
8        Science    530
9          Music    519
10      Thriller    510
11        Horror    418
12         Drama    347
13        Comedy    234
14       Romance    197
15         Movie    183
16       Western    142
17       History    110
18       Fiction     80
19         Crime     34
20        Family      8
21       Foreign      8
```



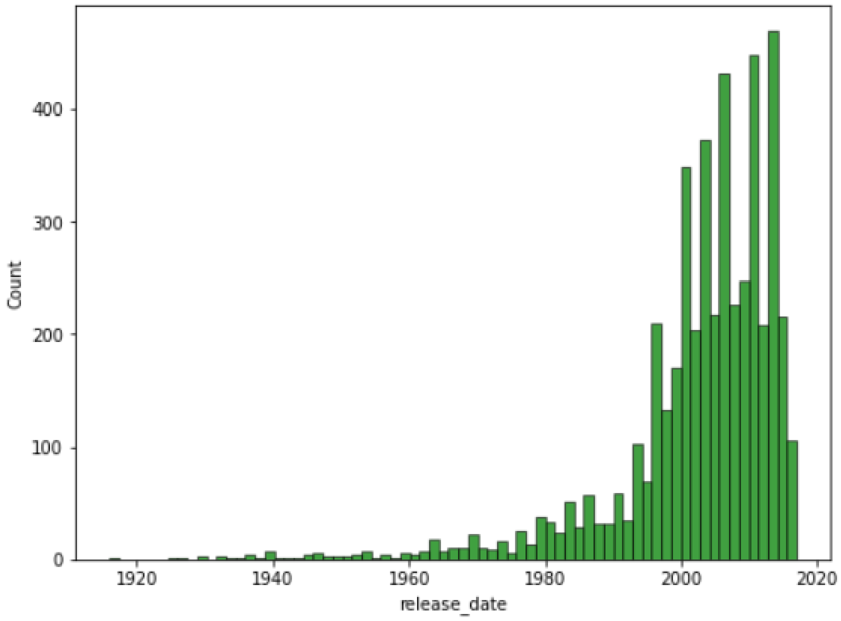
```
[ ]  #top ten movie genres
     plt.figure(figsize=(12,10))
     plt.grid(axis='x',color='black', linestyle = ':', alpha=0.5)
     plt.title('Top 10 TV Show Genres', fontname='monospace', fontsize=25, y=1.05)
     a = sns.barplot(x="Count", y="Genre", data=genre_count_df[:10], palette='rocket')

     for q in [a]:
       for w in ['bottom', 'left']:
         q.spines[w].set_linewidth(1.5)
       for w in ['right', 'top']:
         q.spines[w].set_visible(False)

     plt.xlabel('Count', fontsize=15)
     plt.ylabel('Genre', fontsize=15)
     plt.xticks(fontsize=14)
     plt.yticks(fontsize=14)
     plt.show()
```

[ ]



```
[ ]  #highest count of movies released

     plt.subplots(figsize=(8,6))
     sns.histplot(pd.DatetimeIndex(movies_data["release_date"]).year,kde=False, color="green")
```

```
     <matplotlib.axes._subplots.AxesSubplot at 0x7fbe4688a5d0>
```



8. CONCLUSION

- To conclude, a recommender system powered by content-based filtering performed using the cosine similarity algorithm can make better recommendations for users by suggesting them movies that have similar key features like the IMDb votes, average IMDb rating, genre, release year, casts, directors, user tags, etc

- Our movie recommender system will be having a pretty good prediction performance since our approach uses content based filtering to give results according to the specific user's preferences.

- Techniques like Clustering, Similarity and Classification are used to get better recommendations thus increasing precision and accuracy.

- In future we can work on a recommender using clustering and similarity as well as use a hybrid approach including collaborative filtering along with content-based filtering for better performance.

- Some limitations of content based filtering are that it can only make recommendations based on existing interests of the user, it does not consider the fact that what do other users think of an item, thus low quality item recommendations may occur sometimes.

9. APPENDIX AND REFERENCES

[1] K. Shubhankar, A. P. Singh and V. Pudi, “An Efficient Algorithm for Topic Ranking and Modeling Topic Evolution”, Proceedings of the 22nd International Conference on Database and Expert Systems Applications, Toulouse, France, (2011).

[2] Y. Song, J. Du and L. Hou, “A Topic Detection Approach Based on Multi-level Clustering”, 2012 31st Chinese Control Conference, Hefei, (2012), pp. 3834-3838.

[3] C. Wartena and R. Brussee, “Topic Detection by Clustering Keywords”, In: Proc. of the 19th International Workshop on Database and Expert Systems Applications, Turin, (2008), pp.54-58, September 1-5.

[4] L. Wu, B. Xiao, Z. Lin and Y. Lu, “A Practical Approach to Topic Detection Based on Credible Association Rule Mining”, 2012 3rd IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC), Beijing, (2012), pp.227-231, September 21-23.

[5] Y. Jo, C. Lagoze and C.L. Giles, “Detecting Research Topics via the Correlation Between the Graphs and Texts”, In. Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, (2007), pp.370-379.

[6] Y. Jo, J. E. Hopcroft and C. Lagoze, “The Web of Topics: Discovering the Topology of Topic Evolution in a Corpus”, In. Proceedings of the 20th International Conference on World Wide Web, New York, NY, USA, (2011), pp.257-266.

[7] M. Jayashri, P. Chitra. “Topic Clustering and Topic Evolution Based On Temporal Parameters”, 2012 International Conference on Recent Trends in Information Technology (ICRTIT), Chennai, Tamil Nadu, (2012), pp.559-564, April 19-21.

[8] W. Cui, S. Liu, L. Tan, C. Shi, Y. Song, Z. J. Gao, H. Qu and X. Tong, “TextFlow: Towards Better Understanding of Evolving Topics in Text”, IEEE Transactions on Visualization and Computer Graphics, vol.17, no. 12, (2011), pp.2412-2421.

[9] Y. Jin. “A Topic Detection and Tracking method combining NLP with Suffix Tree Clustering”, 2012 International Conference on Computer Science and Electronics Engineering (ICCSEE), Hangzhou, vol. 3, (2012), pp. 227-230.

[10] P.D. Turney and P. Pantel, “From Frequency to Meaning: Vector Space Models of Semantics'', Journal of Artificial Intelligence Research, vol. 37, (2010), pp. 141-188.

[11] D. Zhang and S. Li. “Topic Detection Based on K-means”, 2011 International Conference on Electronics, Communications and Control (ICECC), Ningbo, (2011), pp. 2983-2985.

[12] L. Yue, S. Xiao, X. Lv, T. Wang, “Topic Detection Based On Keyword”, 2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), Jilin, (2011), pp. 464-467.

[13] C. D. Manning, P. Raghavan and H. Schütze, “Introduction to Information Retrieval”, Cambridge University Press, New York, NY, USA, (2008), pp.118-120.

[14] T. Masada and A. Takasu, “Extraction of Topic Evolutions from References in Scientific Articles and Its GPU Acceleration”, In CIKM, (2012), pp. 1522–1526.

[15] N. Lv, J. Luo, Y. Liu, Q. Wang, Y. Liu and H. Yang, “Analysis of Topic Evolution Based on Subtopic Similarity”, 2009. CINC, 09. International Conference on Computational Intelligence and Natural Computing, Wuhan, vol. 2, (2009), pp. 506-509.

[16] A, Shah, K. Khowaja and A. S. Shah “A Model for Handling Overloading of Literature Review Process for Social Science”, In Proceedings of International Conference on Advanced Research in Business and Social Sciences 2015, Kuala Lumpur, Malaysia, (2015), pp-335-341.

[17] Y. Chen, Y. Yang, H. Zhang, H. Zhu, and F. Tian, “A Topic Detection Method Based on Semantic Dependency Distance and PLSA”, In: Proceeding of 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD), (2012), pp-703-708.

[18] S. M. Krishna and S. D. Bhavani, “An Efficient Approach for Text Clustering Based on Frequent Itemsets”, European Journal of Scientific Research, vol. 42, no. 3, (2010), pp. 385-396

[19] Abdul Ahad, Muhammad Fayaz and Abdul Salam Shah, “Navigation through Citation Network Based on Content Similarity Using Cosine Similarity Algorithm”, International Journal of Database Theory and Application Vol.9, No.5 (2016), pp.9-20

[20] Ramni Harbir Singh, Sargam Maurya, Tanisha Tripathi, Tushar Narula, Gaurav Srivastav, “Movie Recommendation System using Cosine Similarity and KNN”,

International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-9 Issue-5, June 2020

[21] Bei-Bei CUI, "Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm", ITM Web of Conferences 12, 04008 (2017)

[22] https://labelyourdata.com/articles/movie-recommendation-with-machine-learning

[23] Goyani, Mahesh & Chaurasiya, Neha (2020). A Review of Movie Recommendation System : Limitations, Survey and Challenges.

[24] IJRTI (ijcrt.org)

[25] Movies recommendation system in R Studio, Machine learning (slideshare.net)

10. PUBLICITY

GITHUB LINK-1

GITHUB LINK-2
KHUSHIKASHYAP-WEB/FDA_CSE3505_PROJECT (github.com)