# Developing a title Recommendation System using K-Means Clustering with IMDB Data from 1950 to 2023

**Anthony Aramouny – Charbel Daccache – Roula Irani – Jana Hasan Kasem**


# Abstract

This project aimed to create a title recommendation system using the K-means clustering algorithm, focusing on utilizing PySpark and SQL to develop a robust and efficient solution. To achieve this objective, two research papers were reviewed to inform the choice of the K-means approach. The data for the study was procured from the IMDb database (Kaggle), and tables were joined using SQL. Exploratory data analysis and machine learning modeling were performed using PySpark. The model successfully clustered titles into six groups, providing up to 10 recommendations for a given movie or series based on the user's viewing history. The results demonstrate that K-means is an effective approach for creating a title recommendation engine by identifying common attributes between features and clustering observations. Future research could enhance the system by assigning ranks and weights to different features, allowing the engine to tailor recommendations based on individual user preferences.


# Introduction

The Internet Movie Database (IMDb) is a widely used website for researching movies, TV shows, and games. It first appeared in 1990, and today Amazon owns and operates it.

The Kaggle dataset entitled "IMDb Dataset - From 1950 to 2023" includes a collection of data related to over 85,000 movies, TV shows, and video games. The dataset includes information on the titles, release dates, genres, ratings, and other relevant details.

The dataset presents a challenge due to the presence of different tables that lack interconnectivity, as well as inconsistencies and inaccuracies within the data. The dataset exhibits instances of redundancy, absence, and inaccuracies in specific data fields. Furthermore, there exist divergences in the representation of specific information, such as film titles or genres in different languages.
This circumstance may pose challenges in handling the data and deriving significant insights from it.

Despite the previously mentioned difficulties, the IMDb Dataset - From 1950 to 2023 represents a valuable asset for reviewing patterns in the world of entertainment throughout history. By using suitable techniques and tools, it is feasible to clean and transform data for the purpose of obtaining valuable insights such as trends over time of the most popular movie genres and the most highly-rated films in

history. In addition, the development of a movie recommendation system would facilitate the identification of users' preferences.

# Literature review

**Predicting the IMDB rating by using EDA and machine learning Algorithms**
**Prayas Dixit*, Shahzeb Hussain, Gaurav Singh Infosys Limited, Pune, Maharashtra, India**
**Research in International Journal of Scientific Research in Computer Science Engineering and Information Technology · August 2020**

In this paper, the authors aim to predict the IMDB rating of a movie based on various factors such as duration, budget, genre, and language, among others. They have utilized historical movie data to predict the success of upcoming movies with the goal of achieving maximum accuracy.

In their literature review, they reference several studies and sources that have contributed to their research:
Latif and Afzal (2016) discussed the prediction of movie popularity using machine learning techniques.
Ahmad et al. (2017) focused on movie success prediction using data mining.
Siva et al. (2012) explored box-office opening prediction of movies based on hype analysis through data mining.
VR et al. (2017) investigated predicting movie success based on IMDb data.
Anantharaman et al. presented a study on movie success prediction using data mining.
In addition to these studies, the authors also referred to online resources such as Towards Data Science and Machine Learning Mastery to gather more information on techniques and methodologies used in their research.
Overall, the literature review highlights several approaches to predicting movie success and underscores the importance of various factors such as duration, budget, genre, language, and more in accurately predicting IMDB ratings. The authors did not specifically discuss duration and genre in the context of their literature review. However, they mentioned that in their own study, some of the most important features for predicting the IMDB rating of a movie were num_voted_users, duration, budget, main_genre, and language.
By including duration and genre as significant features in their models, the authors implicitly acknowledged the importance of these factors in predicting movie success. While the paper does not delve deeply into the individual impact of these features, it demonstrates that they play a crucial role in the overall predictive models built using machine learning algorithms.

**Weight based movie recommendation system using K-means algorithm.**
**Md. Tayeb Himel; Mohammed Nazim Uddin; Mohammad Arif Hossain; Yeong Min Jang**
**2017 International Conference on Information and Communication Technology Convergence (ICTC)**

In this paper, the authors introduced a weight-based title recommender system that utilizes a combination of user preferences and title attributes to generate recommendations. The system employs the k-means clustering algorithm to group titles into clusters, which can be used to refine recommendations. The system has been implemented in Python using popular libraries such as pandas, NumPy, and scikit-learn. The performance of the recommender system was evaluated through informal evaluations with a small set of users, comparing the quality of recommendations with and without clustering. The results showed that the clustered recommendations had higher RMSE, MSE, and MAE values, indicating a higher difference in results quality, and thus suggesting that the clustered results were more polished and preferable to the users. The user evaluation also supported this claim. In the future, the authors plan to experiment with different tunings for the algorithm and explore the use of various machine learning and clustering algorithms for comparison. They also aim to implement a web-based user interface with a user database and tailor the learning model to each user.

# Experimental Design and Evaluation

“IMDb Dataset – From 1888 to 2023” is a dataset that contains information of movie titles and their details from 1888 to 2023, this data set is from IMDb website and uploaded on Kaggle.

It is subdivided into 7 datasets. The data cleaning was done on the following tables however, we dropped the tables that don’t have any sufficient insights such as: IMDb Names, IMDB title AKAS, and IMDb Title episode.

## Datasets Description & Challenges Faced:

### 1) IMDb title basics dataset:

| Feature | Data Type | Description |
|---|---|---|
| tconst | String | An alphanumeric unique identifier of the title. |
| titleType | String | The type/format of the title (e.g. movie, short, tvseries, tvepisode, video, etc). |
| primaryTitle | String | The more popular title / the title used by the filmmakers on promotional materials at the point of release. |
| originalTitle | String | The original title, in the original language. |
| isAdult | Boolean | Indicates whether the title is an adult title or not. A value of 0 represents a non-adult title, while a value of 1 represents an adult title. |
| startYear | YYYY | Represents the release year of a title. In the case of TV series, it is the series start year. |
| endYear | YYYY | TV series end year. ‘\N’ for all other title types. |
| runtimeMinutes | Integer | The primary runtime of the title, in minutes. |
| genres | array of strings | Includes up to three genres associated with the title. |

In this dataset, 'tconst' column was renamed to read 'title_id', 'titleType' to read 'title_type', 'runtimeMinutes' to read 'runtime', and 'genres' into 'genre'.
We dropped 'primaryTitle', 'endYear', 'isAdult', and 'original_title' columns and we changed the types of 'title_id', 'title_type', and 'genre' into **string** and 'startYear', 'runtime' into **integer**.
Then, we noticed that we have ‘\N’ value in our table, therefore we defined a function to replace it with 'Nan'.

### 2) IMDb Title Crew dataset:

| Feature | Data Type | Description |
|---|---|---|
| tconst | String | alphanumeric unique identifier of the title |
| directors | array of strings | director(s) of the given title identified by their nconsts |
| writers | array of strings | writer(s) of the given title identified by their nconsts |

In this dataset, 'tconst' column was renamed to read 'title_id' and 'directors' into 'director'.
We dropped the 'writers' column. Moreover, we changed the type of 'title_id' and 'director' into **string**.

Then, we noticed that we have '\N' value in our table, therefore we defined a function to replace it with 'NaN'.

**3) IMDb Title principles:**

| Feature | Data Type | Description |
|---|---|---|
| tconst | String | alphanumeric unique identifier of the title |
| ordering | Integer | a number to uniquely identify rows for a given titleId |
| nconst | String | alphanumeric unique identifier of the name/person |
| category | String | the category of job that person was in |
| job | String | the specific job title if applicable, else '\N' |
| characters | String | the name of the character played if applicable, else '\N' |

In this dataset, 'tconst' column was renamed to read 'title_id' and 'nconst' to read 'id'.
We dropped ordering, job, and characters columns. Moreover, we changed the type of 'title_id', 'id', category into string.
Then, we noticed that we have '\N' value in our table, therefore we defined a function to replace it with NaN.
In addition, we selected only the actor/actress in the category column and grouped the data frame by 'title_id' and joined the 'id' column values.

**4) IMDb Title Ratings dataset:**

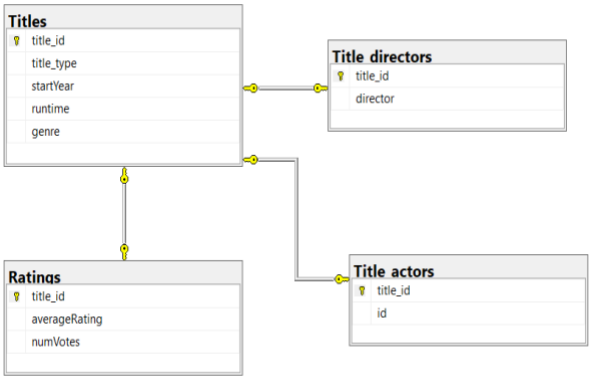
| Feature | Data Type | Description |
|---|---|---|
| tconst | string | alphanumeric unique identifier of the title |
| averageRating | float | weighted average of all the individual user ratings |
| numVotes | integer | number of votes the title has received |

In this dataset, we renamed the 'tconst' to read 'title_id', changed the type of 'title_id' into **string**, 'averageRating' into **float**, and 'numVotes' into **integer**.
We noticed that this dataset contains ratings related to title ids not existing in the title basics dataset. Therefore, we joined two datasets, IMDb title ratings and IMDb title basics, and we kept the needed ratings.

## Data modeling:

The project involved handling big data related to titles and ratings from multiple datasets. To streamline the data processing and analysis, **SQL Server Management Studio** was used to create a database named IMDB and construct tables for Titles, Title_Directors, Ratings, and Title_Actors. To import the data into the IMDB database, the raw CSV files were imported as flat files. During this process, it was noted that some values in the CSV files were delimited by a comma ',', which caused import errors. To mitigate this issue, a data cleansing step was performed where commas within values were replaced by '-'.

Once the tables were successfully created and the data was imported, a join operation was performed to merge the information from the different tables. The join was based on the primary key of the Titles table, which was titled **'title_id'**. By performing this join operation, a comprehensive and unified dataset was created that could be analyzed and utilized for modeling and visualization purposes.




The following query was used to join the four tables:

```
SELECT t.title_id, t.title_type, t.startYear,
t.runtime, t.genre, td.director as directors,
ta.id as actors, r.averageRating, r.numVotes
FROM Titles t

INNER JOIN Title_directors td
ON t.title_id = td.title_id
INNER JOIN Ratings r
ON t.title_id = r.title_id
INNER JOIN Title_actors ta
ON t.title_id = ta.title_id
```

The above query retrieved a dataset that includes a range of movie-related attributes, such as title type, start year, runtime, genre, director(s), actor(s), average rating, and number of votes. It can be used for various types of analysis and modeling, such as identifying trends over time, exploring popular genres, assessing the success of directors, and more insights discussed next.

## Building a Recommendation Model using K-Means Clustering:

Building a recommendation system using K-Means clustering is an effective approach for suggesting movies to users based on their past preferences. The IMDB dataset provides a wealth of information about movies, including details about the cast (such as the name of the directors and actors), genre, rating, and runtime. By leveraging this data, we can create a clustering algorithm that groups similar movies together and recommends new movies to users based on their past preferences.

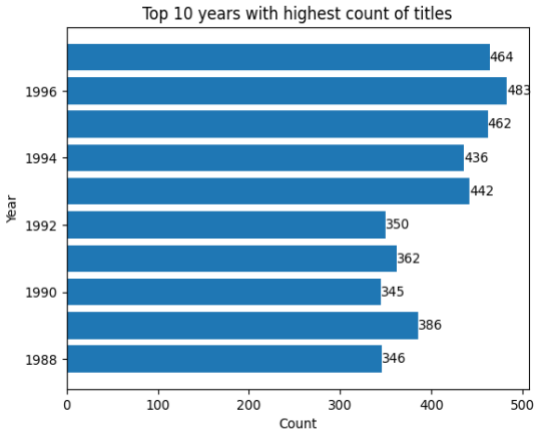
The movie recommendation system using K-Means involves several steps. Firstly, we load the data from the CSV file using Spark, and then we filter the Dataframe to select movies released in the last 70 years with a minimum number of votes and runtime. Next, we explode the "directors," "actors," and "genres" columns to split movies with multiple directors or actors into multiple rows. After that, we drop duplicate rows based on all columns. We apply one-hot encoding to the "genre," "directors," and "actors" columns to convert them into binary vectors, and also convert numerical columns to float type and normalize them using MinMaxScaler to have a range of values between 0 and 1. We then drop all the original categorical and numerical columns except the final features, which consist of binary vectors and normalized numerical values.

Finally, we train the K-Means model with a range of K values from 5 to 11, and we choose the K value that gives the highest Silhouette score. Based on the Silhouette score, the K-Means model with k=6 is the best. We make predictions on the entire dataset and evaluate the Silhouette score for each K value, which measures how similar a data point is to its own cluster compared to other clusters. The resulting clusters are used to recommend movies to users based on their preferences.

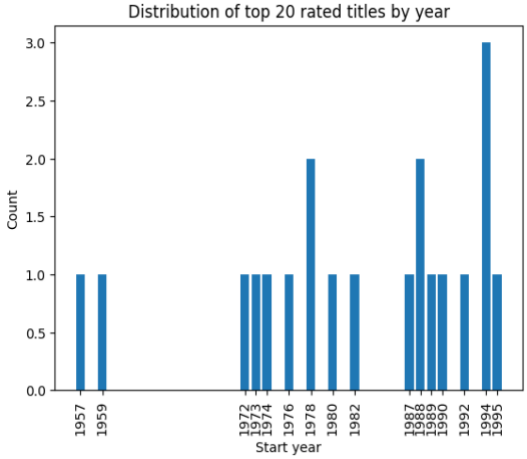
# Results

## EDA:

As part of our analysis of the IMDB dataset, we conducted exploratory data analysis (EDA) to gain insights into the trends and patterns of title production, ratings, and genre preferences over the years.

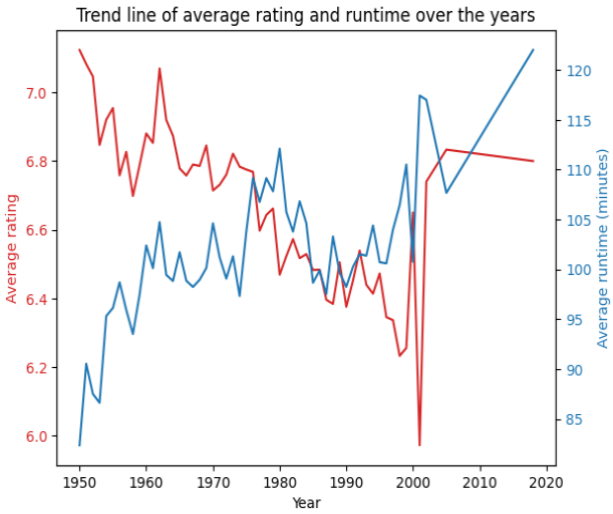



**Years with the highest number of movies using bar charts:** Through our analysis, we found that there has been a steady increase in movie production over the years, with the highest number of movies produced in 1996 with 483 movies.

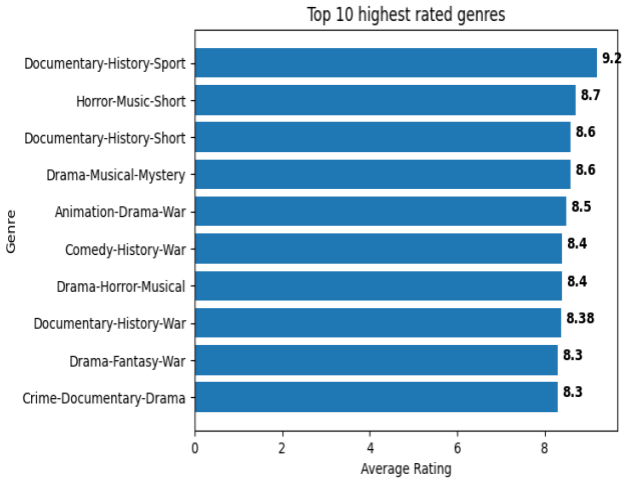



**Distribution of Top 20 Rated Movies by Year using bar charts:** Among the top 20 highest-rated movies in the last 70 years, 1994 has the most with three movies, followed by 1978 and 1988 with two movies each

Based on the analysis of the two above plots, it can be inferred that there is no clear correlation between the production of high-quality movies and the time period they were released, as none of the movies released in the last 30 years have made it to the top 20 list of highest-rated movies.




**Relationship Between Runtime and Average Rating of Movies Across the Years using line charts:** Contrary to popular belief, a longer runtime does not necessarily translate to a higher rating. In fact, over the years, the runtime of movies has significantly increased, while average ratings have seen a decrease. However, it is important to note that this trend may be affected by the sheer number of movies produced. In recent decades, a visible trend has emerged where the average ratings of movies have remained stable despite an increase in their runtime.




**Highest rated combination of genres:** This chart also revealed that historic documentaries about sport and war, as well as short historic documentaries, have received the highest ratings. Short horrors have taken second place, while dramatic musical mysteries complete the top four. This information could be valuable for producers seeking to create a highly rated movie.

## K-Means Clustering:

The K-means clustering model was able to group titles into six clusters and provide up to 10 recommendations for a movie or TV show based on the user's viewing history. This shows that K-means is a useful method for creating a recommendation engine by identifying shared features and clustering observations. Further research could improve the system by assigning importance to different features and customizing recommendations based on the user's preferences.

# Conclusions and recommendations

The study successfully demonstrated the effectiveness of the K-means clustering algorithm in creating a title recommendation system using PySpark and SQL. The system was able to cluster titles into six groups and provide up to ten recommendations based on the user's viewing history. The results suggest that clustering can be an effective approach to identify common attributes between features and group observations with similar characteristics.

## Recommendations:

Incorporate user feedback: The system could be enhanced by allowing users to provide feedback on the recommendations provided to improve the accuracy and relevance of future recommendations.

Weighted clustering: Future research could explore the possibility of assigning ranks and weights to different features, allowing the engine to tailor recommendations based on individual user preferences.

Expansion to other datasets: The system could be improved by incorporating data from other sources, such as user reviews or social media, to provide a more comprehensive understanding of user preferences.

Advanced machine learning techniques: Other machine learning techniques, such as deep learning or neural networks, could be explored to improve the accuracy of recommendations and further enhance the system's performance.

Real-time processing: To improve the user experience, real-time processing could be implemented to provide recommendations as users watch movies or series, making the system more dynamic and responsive to users' needs.