Project Report

On

MOVIE RECOMMENDATION SYSTEM

Submitted in the partial fulfillment for the award of

Post Graduate Diploma in Big Data Analytics (PG-DBDA)

from Know-IT ATC, CDAC ACTS, Pune-411004

Guided by:

Mr. Milind Kapase

Submitted By:

Aryan Sharma	(220343025009)
Roshan Kature	(220343025019)
Mandar Tannu	(220343025021)
Sarang Vasugade	(220343025035)
Swarupraj Bhosale	(220343025045)

CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Aryan Sharma	(220343025009)
Roshan Kature	(220343025019)
Mandar Tannu	(220343025021)
Sarang Vasugade	(220343025035)
Swarupraj Bhosale	(220343025045)

have successfully completed their project on

MOVIE RECOMMENDATION SYSTEM

under the guidance of Mr. Milind Kapase.

ACKNOWLEDGMENT

This project **"Movie Recommendation System"** was a great learning experience for us and we are submitting this work to Know-IT ATC, CDAC ACTS, Pune.

We are very grateful to mention the name of **Mr. Milind Kapase** for his valuable guidance on this project. His continuous guidance and support helped us overcome various obstacles and intricacies during the course of the project work.

We are highly grateful to **Mr. Vaibhav Inamdar**, the Center Coordinator at Know-IT, Pune, for his guidance and support whenever necessary while we were pursuing the Post Graduate Diploma in Big Data Analytics (PG-DBDA) through C-DAC ACTS in Pune.

Our most heartfelt thanks go to **Mr. Shrinivas Jadhav** (Vice-President, Know-IT, Pune) who provided all the required support and his kind coordination to provide all the necessities like required hardware, internet facility, and extra lab hours to complete the project, throughout the course and till date, here in Know-IT ATC, CDAC ACTS, Pune.

From:

Aryan Sharma	(220343025009)
Roshan Kature	(220343025019)
Mandar Tannu	(220343025021)
Sarang Vasugade	(220343025035)
Swarupraj Bhosale	(220343025045)

TABLE OF CONTENTS

ABSTRACT

This project presents a scalable Movie Recommendation System that leverages Natural Language Processing (NLP) and K-Nearest Neighbors (KNN) to generate content-based movie suggestions. The dataset comprises over 722,000 movies, including attributes such as title, genres, language, overview, popularity, budget, and revenue. To efficiently handle large-scale data, the preprocessing pipeline is implemented using PySpark, encompassing tasks like handling missing values, removing redundant records, normalizing numerical features with MinMaxScaler, and performing text-based feature engineering by merging multiple textual attributes. Additionally, web scraping techniques are employed to enrich data completeness by filling missing values in key fields such as genres, overview, and credits.

The recommendation model utilizes Sentence-BERT (SBERT) embeddings, specifically the all-MiniLM-L6-v2 transformer, to convert movie descriptions into dense vector representations. These embeddings enable efficient similarity-based retrieval when processed through the KNN algorithm. By identifying the nearest neighbors for a given movie query, the system generates personalized recommendations based on content similarity. This approach highlights the effectiveness of NLP-driven embeddings combined with KNN-based retrieval, offering an efficient and scalable solution for delivering accurate, content-focused movie recommendations.

1. INTRODUCTION

The rapid expansion of digital content has made recommendation systems essential for enhancing user experience in domains like movies, music, and e-commerce. A Movie Recommendation System helps users find relevant films by analyzing content similarities. Advancements in Natural Language Processing (NLP) and machine learning have significantly improved the accuracy and efficiency of such systems.

This report presents a Movie Recommendation System that utilizes NLP-based embeddings and K-Nearest Neighbors (KNN) to suggest similar movies. The dataset, containing over 722,000 movies, includes attributes such as title, genres, language, overview, popularity, budget, and revenue.

For efficient large-scale data processing, we employ PySpark. Preprocessing steps include handling missing values, removing redundant records, normalizing numerical features, and combining textual attributes for feature engineering. Additionally, web scraping is used to enrich missing data.

The recommendation system leverages Sentence-BERT (SBERT) embeddings to convert movie descriptions into vector representations, which are processed using KNN for similarity-based retrieval.

This study aims to assess the effectiveness of NLP-driven embeddings and KNN-based retrieval in content-based recommendations. We explore optimal techniques to enhance recommendation accuracy and relevance.

Dataset and Features

The dataset, sourced from www.kaggle.com, provides comprehensive metadata and descriptive attributes for movie analysis. It enables pattern recognition in movie preferences, forming the foundation for an effective recommendation model.

The goal is to develop a robust content-based filtering system using Sentence-BERT embeddings and KNN retrieval, optimizing recommendations through semantic similarity and large-scale data processing.

2. SYSTEM REQUIREMENTS

Hardware Requirements

- Processor: Intel Core i5 or higher
- Platform - Windows 7 or above
- RAM - 8 GB or more
- Peripheral Devices - Keyboard, Monitor, Mouse
- Wifi connection with minimum 2 Mbps speed

Software Requirements

- Programming - Python, Machine Learning
- Tools - PySpark
- Dashboard - Tableau
- UI - Streamlit

3. FUNCTIONAL REQUIREMENTS

Data Processing

- Load, clean, and preprocess the dataset using PySpark.
- Handle missing values via imputation or removal.
- Normalize numerical features using MinMaxScaler.
- Convert text-based attributes (title, genres) into vector representations.
- Use web scraping to enrich missing metadata.

Recommendation System

- Generate recommendations based on content similarity.
- Use Sentence-BERT (SBERT) for movie description embeddings.
- Apply K-Nearest Neighbors (KNN) for similarity-based retrieval.

User Interface & Dashboard

- Provide a web-based search interface for movie recommendations.
- Display relevant movie details with filtering options (genre, language, popularity).

Performance & Scalability

- Efficiently handle a dataset of 722,000+ movies.
- Ensure recommendations are generated within 1-2 seconds.
- Support future dataset expansion.

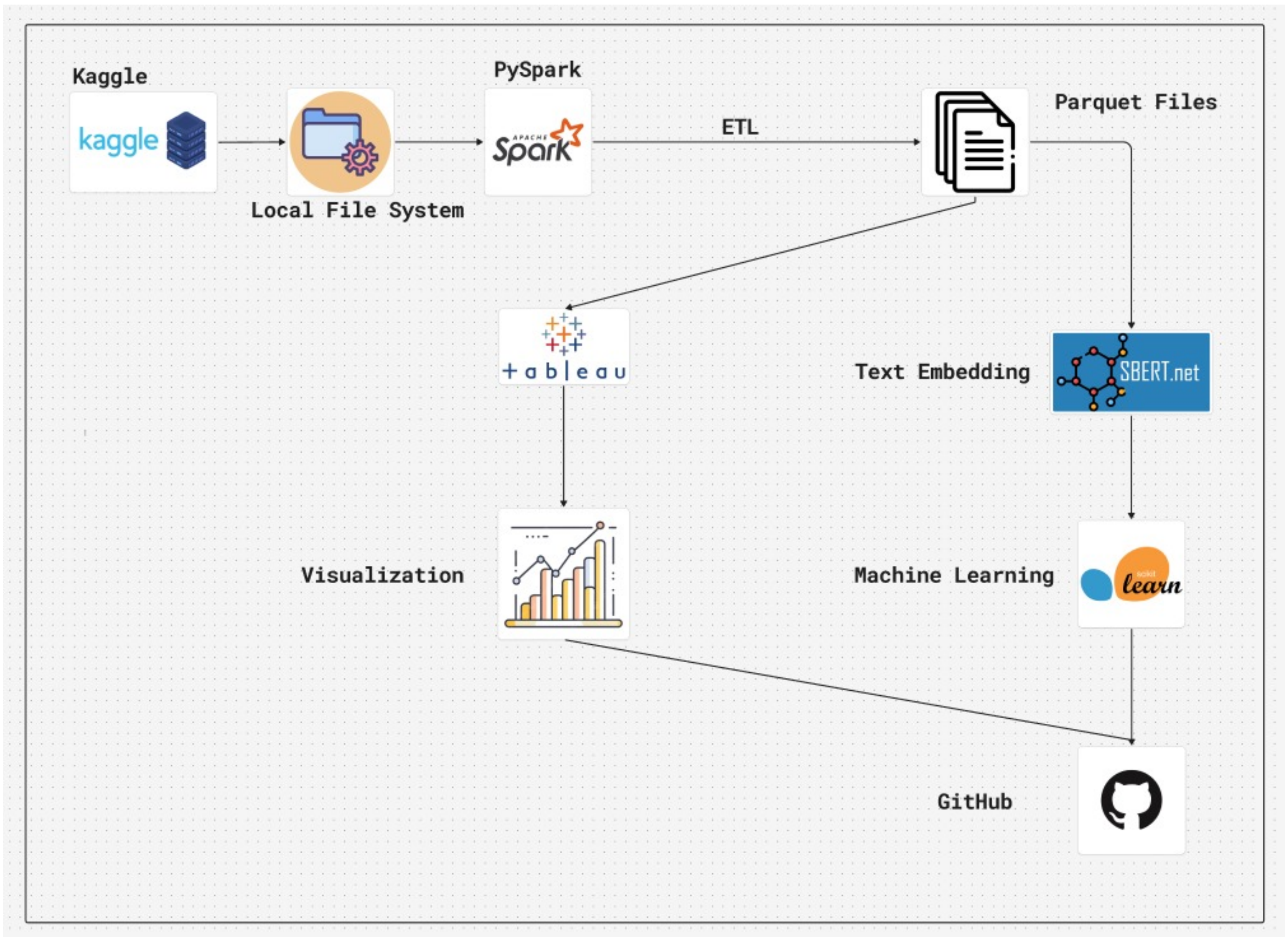
Dashboard & Insights

- Use Tableau to visualize trends, popularity, and recommendation patterns.

Integration & Deployment

- Deploy on a local server with web-based access.
- No additional software installation required for users.

4. SYSTEM ARCHITECTURE



5. METHODOLOGY

This project employs content-based filtering with K-Nearest Neighbors (KNN) and cosine similarity to recommend movies based on their content. The methodology includes data preprocessing, feature engineering, model development, and evaluation.

Data Collection & Preprocessing

The dataset used is the Millions of Movies dataset from Kaggle, containing movie titles, genres, summaries, and other metadata. Preprocessing steps included:

Handling missing values by imputing or removing incomplete records. Normalizing text (lowercasing, punctuation removal, and stop word elimination). Tokenization and lemmatization for text processing.

Feature Engineering

To capture semantic relationships between movies, the following features were used:

Combined Textual Features: Merging titles, genres, and summaries into a single text column. Text Embeddings: Using the SentenceTransformer model (all-MiniLM-L6-v2) to convert text into dense numerical vectors.

Similarity Computation & Model Development

Cosine similarity was used to measure similarity between movie embeddings. KNN was implemented with cosine similarity as the distance metric. The model was trained on precomputed embeddings to find top-N nearest neighbors.

Recommendation Generation

For a given movie, its embedding is compared against all other movies to retrieve the most similar ones. The top-N movies with the highest similarity scores are recommended.

6. MACHINE LEARNING ALGORITHM

The core machine learning algorithms utilized in this project include:

SentenceTransformer Model ('all-MiniLM-L6-v2')This pre-trained model is part of the Sentence Transformers library, designed to generate high-quality sentence embeddings. It captures semantic meaning from textual data, converting combined movie information (titles, genres, plots) into numerical vectors suitable for similarity calculations.

K-Nearest Neighbors (KNN)KNN is a non-parametric, instance-based learning algorithm used to identify similar items based on feature proximity. In this project:

The model leverages cosine similarity as the distance metric to measure how closely related two movies are.

It identifies the 'K' most similar movies to a given movie, forming the basis of the recommendation system.

KNN's simplicity and efficiency make it well-suited for content-based recommendation tasks, especially when combined with high-dimensional embeddings.

Cosine SimilarityAlthough not a machine learning algorithm in itself, cosine similarity plays a crucial role in this system. It quantifies the similarity between two vectors (movie embeddings) by calculating the cosine of the angle between them. This metric is particularly effective in high-dimensional spaces where the magnitude of the vectors is less important than their orientation.

Together, these algorithms create a robust framework for identifying and recommending movies based on content similarity, ensuring accurate and contextually relevant suggestions.

7. DATA VISUALIZATION AND REPRESENTATION

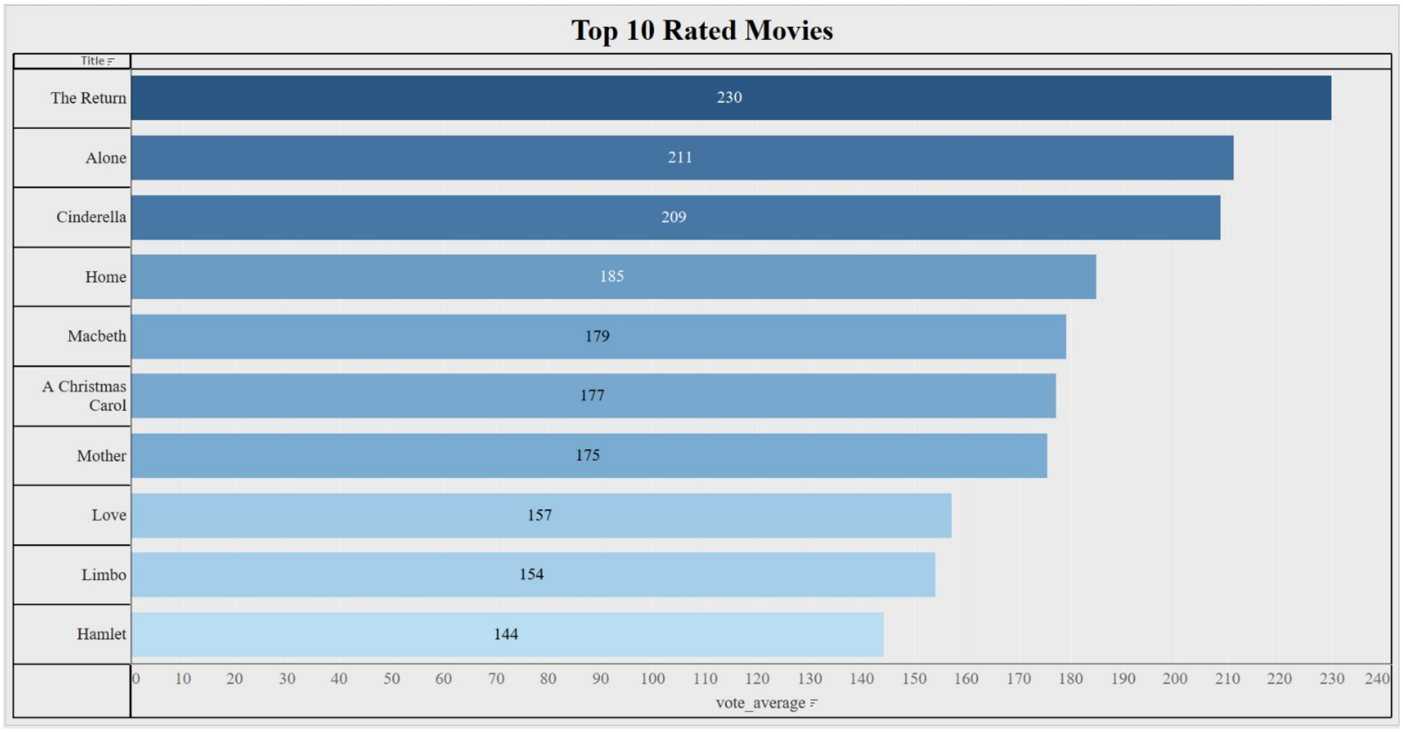


Figure 1: Top 10 Rated Movies

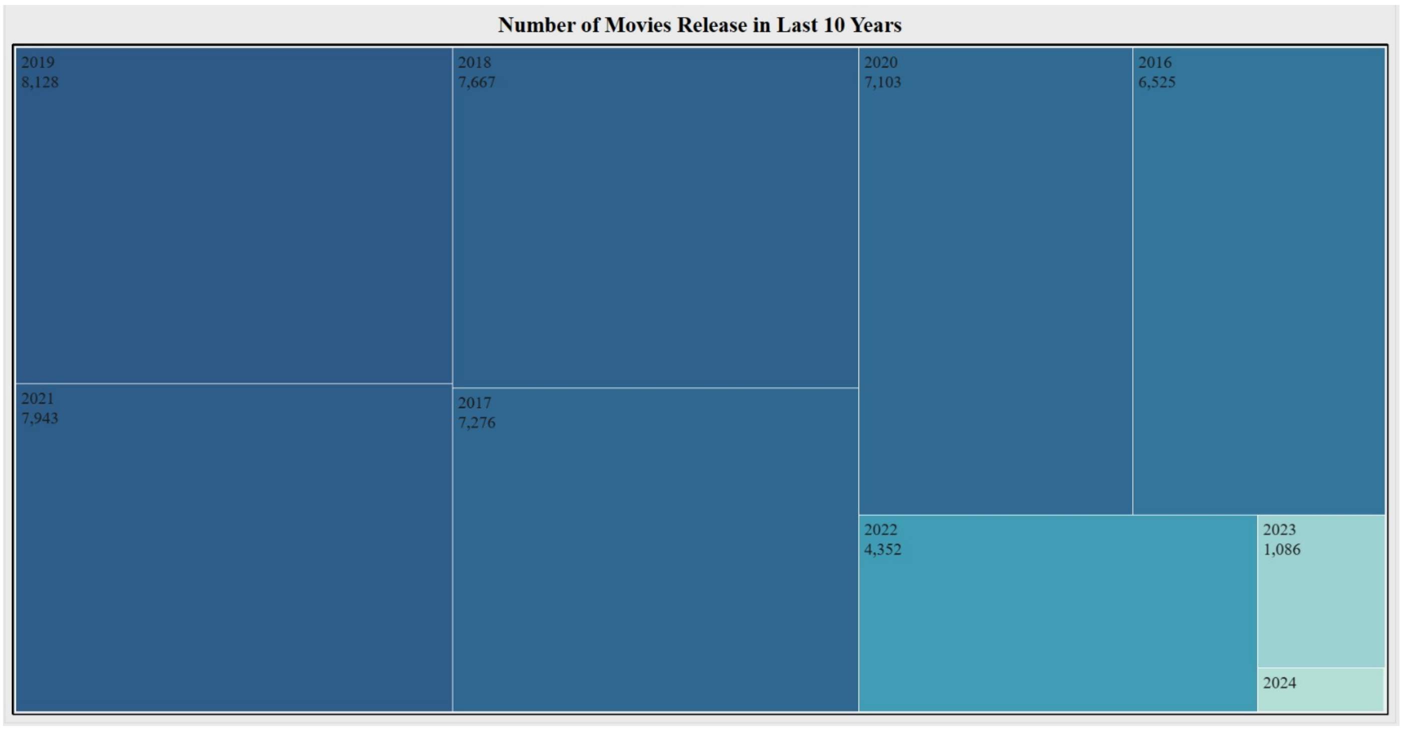


Figure 2: No of Movies Release in Last 10 Years

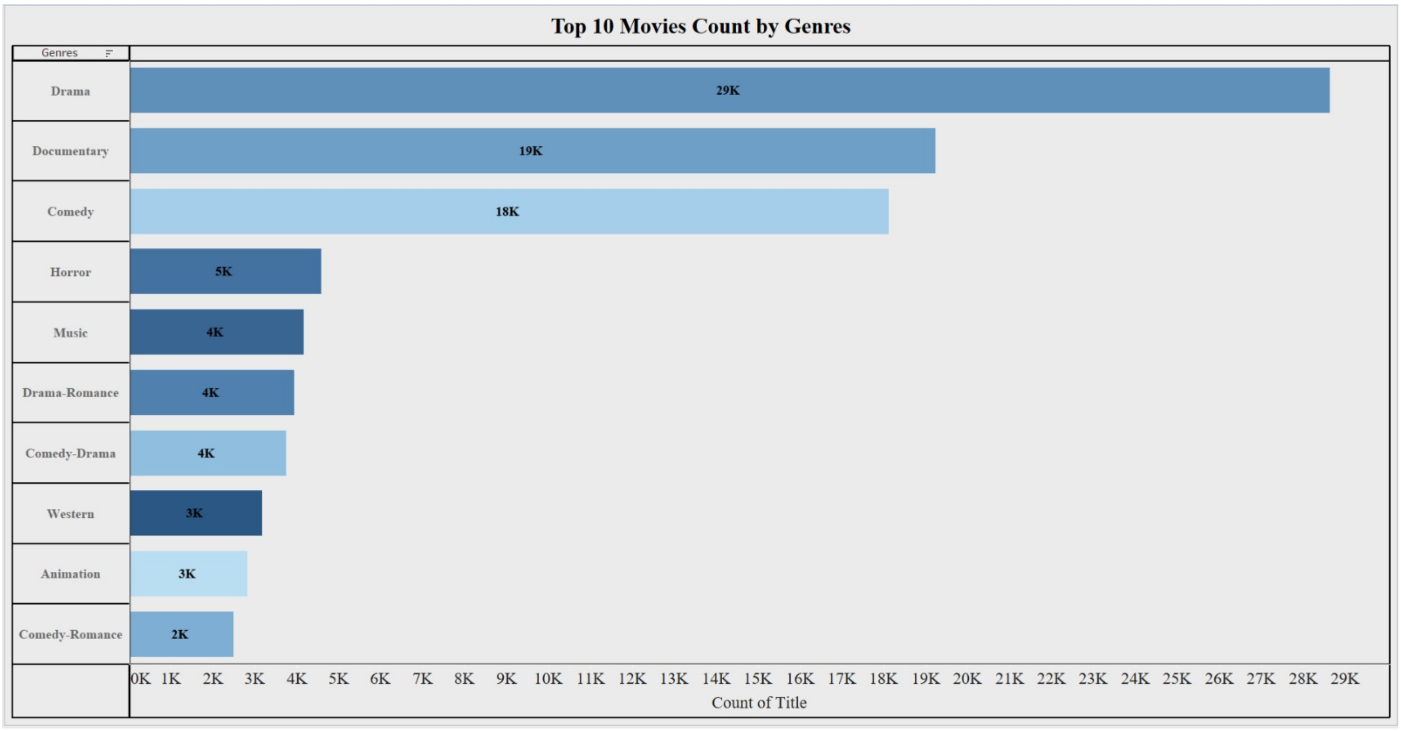


Figure 3: Top 10 Movies Count by Genres

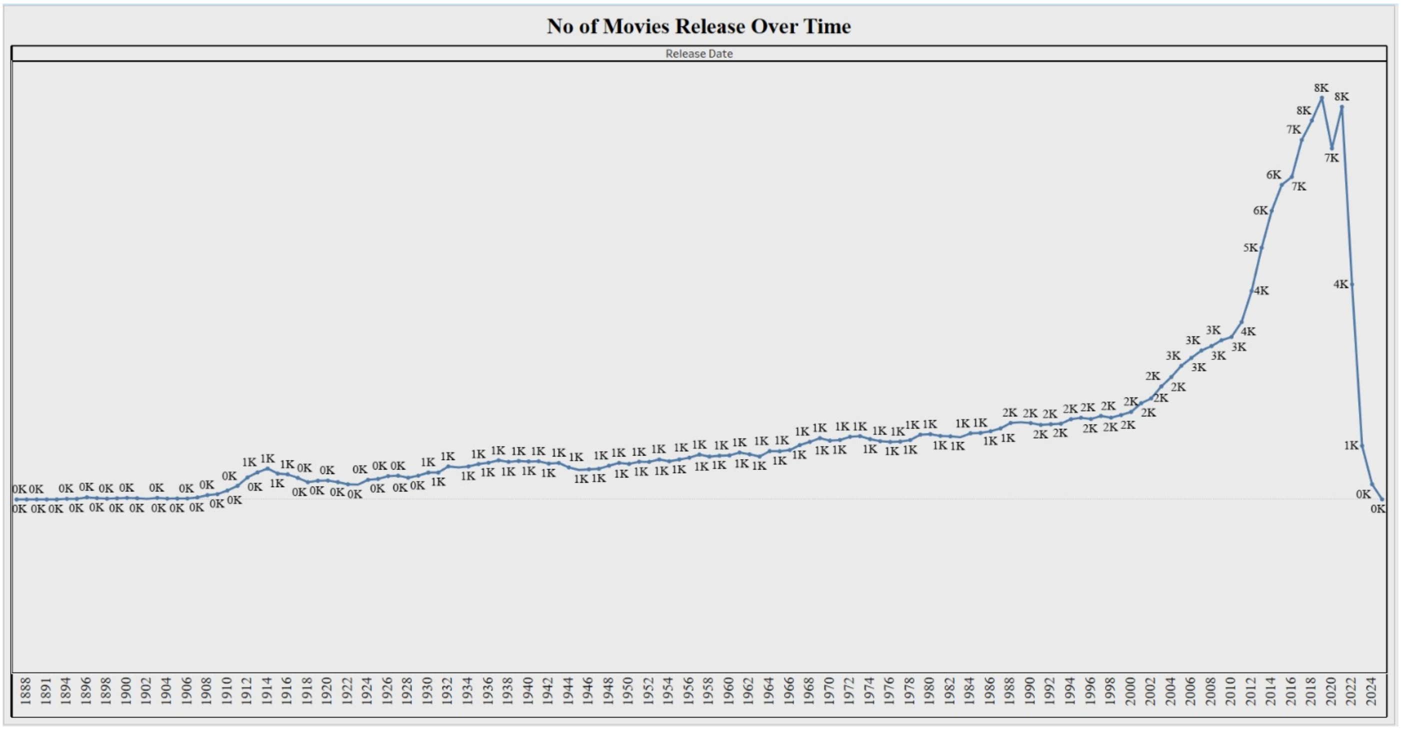


Figure 4: No of Movies Release Over Time

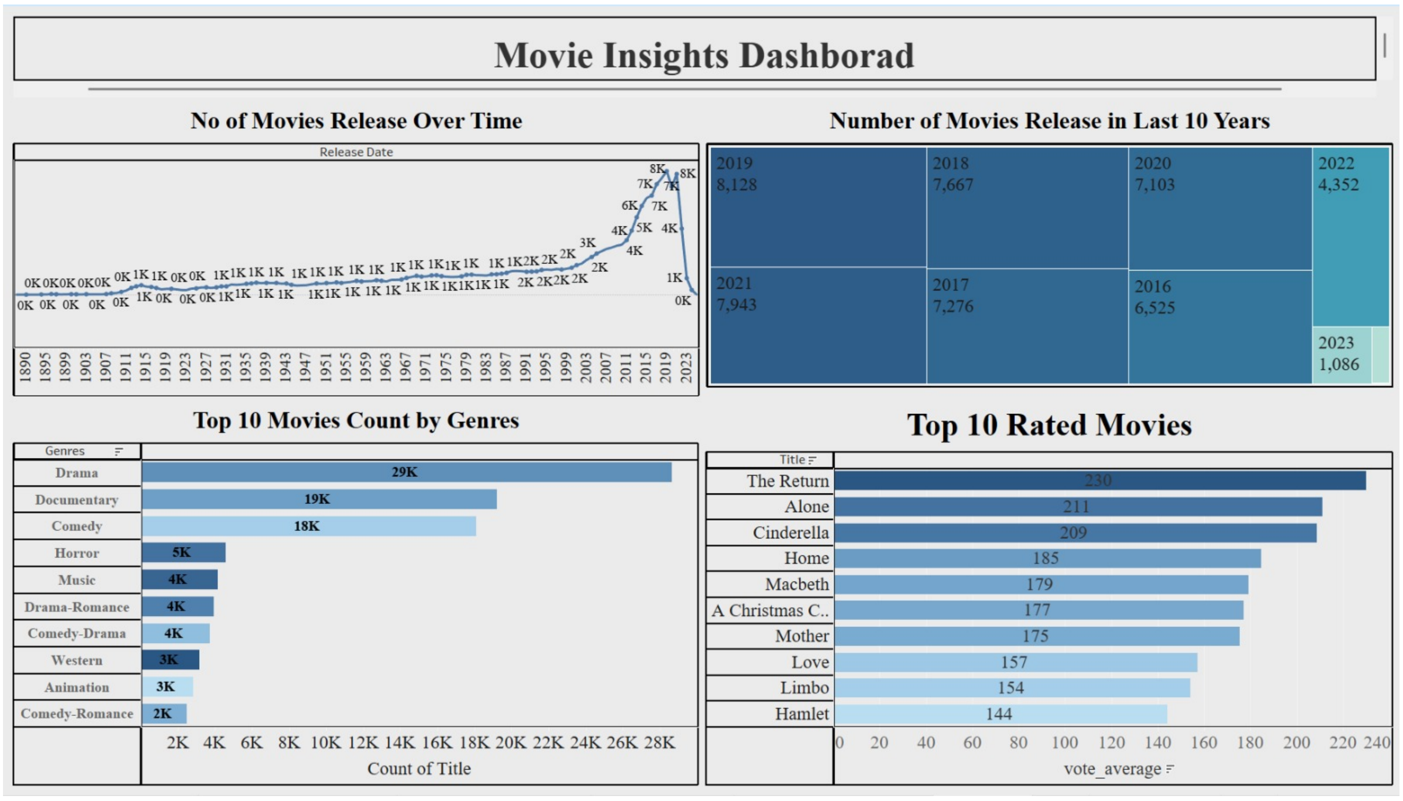
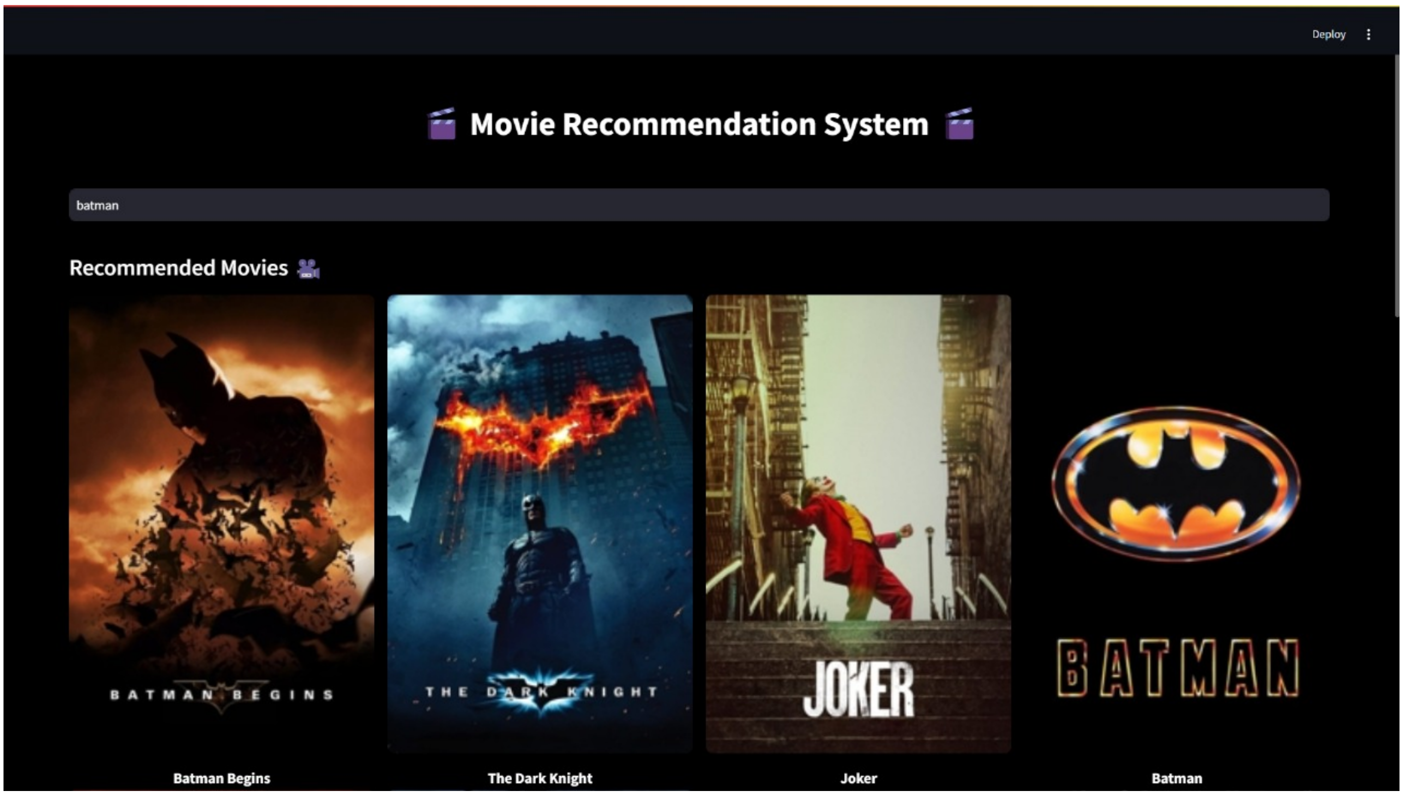


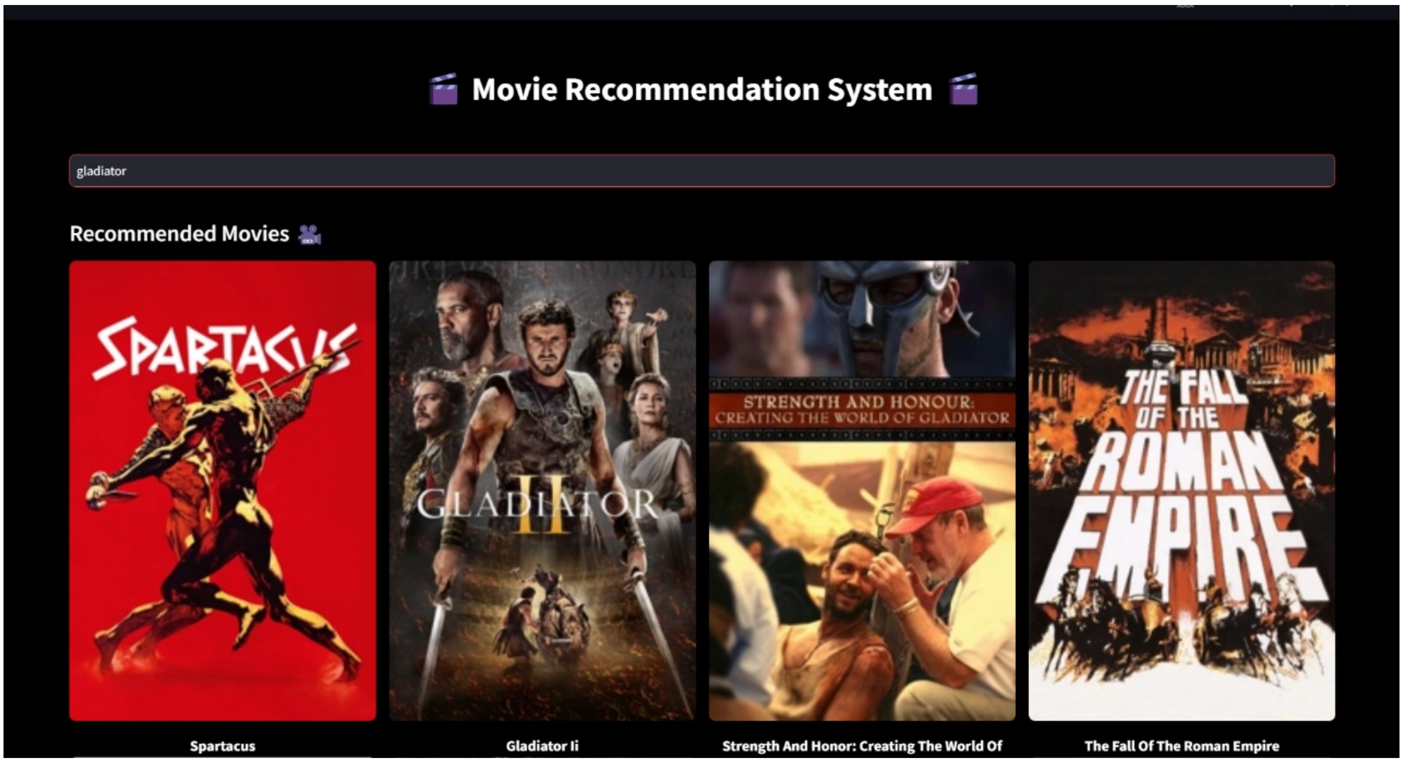
Figure 5: Dashboard

9. RESULT

```
print(get_recommendations("Singham"))
```

```
['Singham', 'Sarkar', 'Simmba', 'Indian', 'Kuttavum Shikshayum', 'Mardaani', 'Thalai Nagaram', 'Toofan Singh']
```




Movie Recommendation System
gladiator
Recommended Movies
Spartacus
Gladiator Ii
Strength And Honor: Creating The World Of
The Fall Of The Roman Empire

8. CONCLUSION AND FUTURE SCOPE

Conclusion

In this project, we developed a content-based movie recommendation system using BERT embeddings for titles, overviews, and original language. The system preprocesses data using Apache Spark, ensuring efficient handling of large datasets. Users can enter a movie name on the web-based platform, and the system provides recommendations based on semantic similarities.

To retrieve similar movies efficiently, we implemented the K-Nearest Neighbors (KNN) algorithm, which finds the top-k closest movies based on their BERT embeddings. KNN helps in fast and scalable similarity searches, making our recommendation process both accurate and efficient.

By leveraging deep learning-based text embeddings and KNN for nearest neighbor search, our model captures contextual meanings better than traditional methods like TF-IDF and cosine similarity. This leads to more relevant and personalized recommendations. However, since our dataset lacks user interaction data (such as ratings), we could not implement collaborative filtering or hybrid models.

Overall, this system provides an efficient, scalable, and intelligent solution for personalized movie recommendations, improving user experience compared to basic keyword-based approaches.

Future Scope

1. **Personalized User Profiles**
   - Implement user accounts where preferences, watch history, and liked movies are stored.
   - Use this data to provide highly personalized recommendations based on individual user behavior.
2. **Hybrid Recommendation System**
   - Combine content-based filtering (BERT embeddings) with collaborative filtering (user behavior-based suggestions) for improved accuracy.
   - Implement a weighted approach to balance both methods and provide diverse recommendations.

3. **Trending & Popular Movie Section**
   - Introduce a section that dynamically updates based on real-time popularity trends from sources like TMDb and IMDb.
   - Use metrics such as recent searches, high ratings, and box office performance to display the most popular movies.
4. **YouTube Trailer Integration**
   - Embed YouTube trailers for recommended movies, allowing users to watch trailers directly within the platform.
   - Enhance user engagement by integrating auto-play trailer previews when hovering over a movie.
5. **Watchlist & Bookmark Feature**
   - Allow users to save movies to a personal watchlist for later viewing.
   - Provide a bookmarking system to save favorite recommendations and revisit them easily.
6. **Multi-Language Support**
   - Expand the system to recommend movies in multiple languages based on user preferences.
   - Use machine translation models to provide localized movie descriptions and recommendations for non-English speakers.

REFERENCES

Movies Daily Update Dataset

`https://www.kaggle.com/datasets/akshaypawar7/millions-of-movies`