DSC 511 - Group Project

Iliana Frantzia, Eleni Yiasoumi, Rebecca Hailou

IMDb Non-Commercial Datasets

Project Overview

This project focuses on performing exploratory and advanced analytics on the IMDb Non-Commercial Datasets (https://developer.imdb.com/non-commercial-datasets/) using Apache Spark. The aim is to derive insights into factors that influence movie ratings and popularity, apply predictive modeling techniques, perform text analysis and build a content-based recommendation system.

Our main objective is to analyze and identify key factors that impact the success of a movie. We then use these insights to train machine learning models capable of estimating a movie's IMDb rating and provide personalized movie recommendations.

Dataset Description

We are working on the IMDb Non-Commercial dataset. Each dataset is a gzipped, tab-separated-values (TSV) formatted file encoded in UTF-8. The first line in each file contains column headers. Missing values are represented with '\N'. We begin by exploring each dataset separately to understand their features and determine which ones are important for achieving our goal. Once selected, the datasets are cleaned and merged to proceed with the analysis.

The seven available datasets are:

File Name	Description
name.basics.tsv.gz	Info on actors, directors, writers
title.akas.tsv.gz	Alternative titles across regions/languages
title.basics.tsv.gz	Core info about titles (type, year, genres)
title.crew.tsv.gz	Director and writer information
title.episode.tsv.gz	TV episode info (season, episode, series)
title.principals.tsv.gz	Main cast and crew
title.ratings.tsv.gz	IMDb average rating and number of votes

Final Dataset Schema

After merging all 7 IMDb datasets and dropping irrelevant columns and rows with null values, we constructed a clean, analysis-ready dataset.

We also filtered the dataset to include only entries where `titleType = 'movie'`, excluding TV episodes, shorts, and series, in order to focus our analysis on movies.

We saved the resulting dataset in two formats:

- `.zip` — for portability and distribution
- `.parquet` — for optimized performance in Apache Spark

Columns Overview

- nconst: Person unique ID
- tconst: Title unique ID
- primaryTitle: Movie title
- isAdult: 0 = non-adult, 1 = adult
- startYear: Release year of the movie
- runtimeMinutes: Duration of the movie
- genres: Genres
- averageRating: IMDb average rating
- numVotes: Number of user votes
- directors: Directors ID
- writers: Writers ID
- category: Role (e.g., actor, director)
- primaryName: Name of person
- primaryProfession: Person's profession
- knownForTitles: Known titles for a person

Target variable: `averageRating`

Exploratory Data Analysis (EDA)

We performed a comprehensive EDA on the IMDb dataset using PySpark DataFrames. The main steps included:

- **Data Loading & Inspection:** Loaded the combined movie dataset and inspected its schema, row samples and descriptive statistics.
- **Data Cleaning:** Handled missing values in key columns like runtimeMinutes, numVotes and averageRating by either filtering or imputing appropriate values. Casted types to ensure numeric operations were valid.
- **Feature Parsing:** Exploded multi-valued columns such as genres to allow genre-level analysis.
- **Statistical Summaries:** Computed summary statistics for numeric columns and examined the distributions of ratings, runtimes and votes.
- **Genre Analysis:** Identified the most common genres and analyzed their average ratings and popularity.
- **Temporal Trends:** Explored movie release trends over the years (by decades) and how average ratings evolved with time. However ,there are no strong linear correlations between the selected numerical IMDb features.

Key Findings from EDA

Rating Distribution:

The average rating for most movies is above 6.5 and the distribution is mostly symmetric, indicating a balanced perception of movie quality.

Number of Votes:

While most movies have moderate vote counts, a few movies received an exceptionally high number of votes, showing strong popularity outliers.

Runtime Insights:

We removed movies with invalid runtimes (such as, 0 or greater than 600 minutes). The majority of films have runtimes between 80–120 minutes, which aligns with typical industry standards. Average runtime increased with each decade, reflecting a trend toward longer films over time.

Adult Content Distribution:

The isAdult field is highly imbalanced ,where the majority of movies are intended for general audiences rather than adult-only.

Role & Profession Demographics:

The most frequent entries in both category and primaryProfession are actors and actresses. There are more male actors than female actresses, indicating a gender imbalance in the dataset.

Genre Popularity:

The most common genres are Drama, Comedy, Romance and Documentary, showing a dominance of emotional and narrative-driven content.

Temporal Trends:

Both the average number of votes and average runtime increase over the decades, which is consistent with the evolution of film popularity and production scale.

This analysis helped us clean and understand the structure of our data, preparing it for further advanced analytics like recommendation systems, graph analysis and text analysis.

Advanced Analysis

1. Text Analysis

Text analysis was a crucial step in our project to convert unstructured textual data into a structured format suitable for machine learning workflows. We applied a comprehensive pipeline of preprocessing techniques to clean and prepare textual features. This included:

- Tokenization
- Normalization
- Lemmatization

These steps were essential to reduce vocabulary size, eliminate noise, and standardize input for effective vectorization and semantic analysis.

We focused on key textual columns such as:

- primaryTitle
- genres
- primaryProfession

During the cleaning phase, we encountered multilingual content, which made stopword removal challenging. Although we could not feasibly eliminate every irrelevant word manually, we made a strong effort to remove as many as possible to improve downstream performance.

We adopted two separate preprocessing pipelines:

- One for machine learning tasks (e.g., sentiment prediction) – this version excluded lemmatization, as we observed a slight drop in performance when it was included.
- One for topic detection ,here, lemmatization was included, as it significantly improved the coherence of extracted topics.

2. Machine Learning

After preprocessing and transforming the text data into feature vectors, we implemented a machine learning pipeline to extract patterns and make predictions. The pipeline followed a standard process:

- Feature Extraction
- Model Training
- Validation
- Evaluation

We used Apache Spark MLlib to develop scalable machine learning models that support distributed processing. Both supervised and unsupervised techniques were explored based on task requirements.

To enhance prediction performance, we incorporated sentiment scores (computed via **TextBlob**) as features. These scores provided insight into the overall tone of each movie's description, offering valuable information for regression modeling.

Model Comparison

The models aimed to predict sentiment scores. Below is a summary of model performance:

Model	R^2	RMSE	Notes
Gradient Boosted Trees	0.378	0.1305	Best performer
Random Forest	~0.35	Higher	Runner-up
Linear Regression	Lower	Higher	Poor generalization
Decision Tree	Lower	Higher	Weak performance

Clustering

Genre-Based Clustering

We applied unsupervised clustering using the KMeans algorithm on movie genres to identify common patterns in movie types.

Feature Construction: `genre features` that we got from text analysis before and use it for clustering.

Optimal k Selection: Used Silhouette Scores to evaluate different values of k. The best performance was observed at k = 6.

Clustering Results: Fitted KMeans with k=6 to assign each movie to one of six genre-based clusters. Extracted top genres for each cluster by analyzing the cluster centers.

Assigned intuitive labels to each cluster based on genre composition:

Cluster Top Genres Label

Cluster	Top Genres	Label
0	Fantasy, Drama, Comedy, Adventure, Action	Fantasy & Action
1	Comedy, Documentary, Action, Romance, Adventure	Diverse Popular Genres
2	Drama, Comedy, Romance, Crime, Action	Mainstream Drama & Romance
3	Horror, Thriller, Drama, Mystery, Comedy	Thriller & Mystery
4	Drama, History, War, Biography, Romance	Historical & Biographical
5	Documentary, Biography, History, Drama, Music	Informative & Cultural Features

Cluster Distribution: Visualized the number of movies per cluster, showing Cluster 2 as the most dominant with ~120K movies.

PCA Projection: Applied PCA to project genre features into 2D, showing that clusters are well-separated, especially clusters 0, 2, and 4. This clustering helped group the movies into genre-based themes and offers insights into genre prevalence and diversity across the dataset.

Topic Detection

To uncover underlying themes in the textual data, we applied unsupervised topic modeling techniques.

- **K-Means Clustering (Unigrams vs. Bigrams)**

Initially used K-Means clustering on unigram-based TF-IDF features to detect textual patterns. To enhance contextual understanding, we extended the feature set with bigrams and re-applied clustering. Evaluated clustering performance using silhouette scores, comparing scores before and after including bigrams. The improvement in silhouette scores demonstrated that bigrams contributed to more cohesive and well-separated clusters.

- **Latent Dirichlet Allocation (LDA)**

We then applied LDA (Latent Dirichlet Allocation) for probabilistic topic modeling, successfully extracting 7 distinct topics from the lemmatized movie titles.

Identified semantically coherent themes, including:

- Holiday & Family
- Horror & Thriller
- Romantic & Emotional narratives
- Supernatural, Mystery, Religion

Overall, LDA proved effective in uncovering latent thematic structures and provided interpretable groupings aligned with common movie genres and narrative styles.

3. Recommendation System

We implemented a content-based recommendation system using PySpark, designed to suggest similar movies based on user preferences and movie features. This system leverages vector similarity techniques to identify relevant titles within the IMDb dataset.

Feature Engineering

Combined multiple columns using VectorAssembler, including:

Numeric features: `startYear`, `runtimeMinutes`, `isAdult`, `averageRating`, `numVotes`

Encoded features: `genre_features`, `profession_features`, `tfidf_features`, `director_features`,`writer_featu

Standardized the feature vectors using StandardScaler to ensure comparability across attributes.

Similarity Computation

Two methods were tested for similarity:

Cosine Similarity: Focuses on vector direction, suitable for sparse/high-dimensional data.

Euclidean Distance: Measures raw distance in the vector space, used for comparison.

Movie-to-Movie Recommendation

Set a target movie (e.g., Titanic) and retrieved its feature vector.

Calculated cosine similarity between the target and all other movies.

Returned top 10 most similar movies excluding the target based on the features we set each time.

Personalized User Recommendations

Allowed users to specify a list of 3 favorite movies (e.g., Percy Jackson, The Maze Runner, Chronicles of Narnia).

Averaged their feature vectors into a user profile vector.

Recommended movies based on their similarity to this aggregated user profile.

Results aligned with user preferences (e.g., fantasy/adventure themes), but also revealed areas for improvement , filtering out less relevant genres.

Final Feature Selection & Similarity Metric

After testing various combinations, we selected the following features as the most relevant for generating movie recommendations:

- genre
- startYear
- isAdult
- director
- writer

These attributes capture both thematic and stylistic elements of movies, enabling more meaningful similarity comparisons.

We also determined that cosine similarity was the most appropriate distance metric for our content-based system. It focuses on the direction of vectors rather than magnitude, making it ideal for high-dimensional and scaled feature spaces. This choice provided better alignment with user preferences and thematic grouping compared to Euclidean distance.

4. Graph Analysis

We used GraphFrames to perform graph analysis on the IMDb dataset by constructing a movie similarity graph based on shared directors. This approach enabled us to explore structural relationships between films and uncover patterns in creative collaborations.

Graph Construction

Vertices: Represented each movie using its unique ID (tconst) and title (primaryTitle).

Edges: A directed edge was created between two movies if they shared the same director. That is, if Director X directed both Movie A and Movie B, an edge was formed from A $\rightarrow$ B.

Graph Metrics & Interpretation

Calculated degree centrality for each movie (the number of connections a movie has based on shared directors).

Found that some nodes had very high connectivity—for instance:

Venice 70: Future Reloaded had 1804 connections, acting as a hub of collaboration and indicating wide directorial overlap with other movies.

Observed multiple disconnected components, reflecting independent creative clusters in the industry.

In addition to the graph analysis techniques successfully applied (such as degree centrality, triangle count, and network visualization), we also attempted to run advanced algorithms including PageRank, Shortest Paths, Label Propagation, and Connected Components using the GraphFrames library in PySpark. However, these operations did not execute as intended .