**Movies Data Analysis**

**By:**

Aya Ali Abdelazim 202000866

Donia Ahmed Abdelaziz 202000542

Farah Waleed Etman 202001464

Gannat Elsayed Mohamed 202000750

Sherif Ali Mahmoud 202000658

**Submitted in fulfillment of the course**

**CSCI461: Introduction to Big Data**

**Abstract**

This documentation presents a project that performs comprehensive analysis of movie data including personalized movie recommendations, sentiment analysis on reviews, predicting movie revenues, and providing insightful answers to movie-related questions like which director has the highest average movie popularity? These analytics were done for the purpose of improving user experiences, gaining insightful knowledge about movies, and assisting with decision-making processes.

**Data Source:**

Our source was “Kaggle.com” website from which we obtained 5 datasets that contained different movies with a variety of features, and we were able to integrate 3 of the datasets to work on them together, while also integrating the fourth and fifth datasets to analyze the features included.

**Technologies Used:**

* Apache Spark
* Python libraries

**Expected Outcomes:**

The project provides valuable insights and analytics on movie data and audience sentiment and provides results that can be of great value to filmmakers in future movie-related decisions.

**Introduction**

The project contains an extensive analysis of movie data including personalized movie recommendations by providing a list of 10 movies similar to the one that the user has seen by recognizing similar attributes between them. As well as sentiment analysis on movie reviews to understand public opinion, feedback and overall preferences. Prediction of movie revenues was also done by checking which features were correlated the most to movie revenues and considering features like genre, director, runtime to predict the revenue. Prediction of revenues can be of great interest to filmmakers and can assist in decision making regarding upcoming movies. In the project we also answer other important movie-related questions like; which director has the highest average movie popularity? In order to obtain insightful knowledge about the film industry. Understanding audience preferences, market trends, and the effects of many elements on a movie’s performance are vital in the film industry, as it operates in a highly competitive and constantly changing environment. So, the motivation behind this project was to use data analytics to find significant patterns, correlations, and insights that can help production companies and filmmakers make informed decisions, increase user satisfaction, and enhance their experiences. In the following sections of this documentation, we will dive into the methodologies, implementation, and results of each aspect of the project.

**Methodology**

1. **Proposal & Design:**
   1. **Data source:**

We obtained 5 datasets from Kaggle website and integrated 3 of them to have the best possible needed features and also integrated the fourth and the fifth datasets to make further analysis. The first dataset “rotten\_tomatos\_movies” is a csv file that initially contained 143,258 rows and 16 columns including (id, title, genre, director, box office, etc). The second dataset “rotten\_tomatos\_movie\_reviews” focused on movie reviews information and, contained initially was 1444963 rows and 11columns. The third dataset “movies\_metadata” contained further information about the movie including (runtime, original language, revenue, etc). After cleaning and integrating the datasets, we had data of 960382 rows and 16 columns. The fourth dataset “mubi\_movie\_data” contained 226575 rows and 10 columns including (release\_year, movie\_popularity, director\_name, etc). The fifth and final dataset “mubi\_ratings\_data” contained 15520005 rows and 13 columns including (rating\_score, rating\_id, user\_id, etc)

* 1. **Technologies used:**
* Python was used due to its extensive libraries (Pandas), data processing capabilities, and visualization tools.
* Spark was used to provide a powerful and flexible platform for processing big data efficiently. Some of Spark benefits is the speed, scalability, ease of use and interactive analytics. It enables you to tackle complex data processing tasks at scale, achieve high performance, and leverage a wide range of tools and libraries within the Spark ecosystem.
  1. **Expected outcome:**

The project provides valuable insights and analytics on movie data including: personalizing movie recommendations, applying sentiment analysis on movie reviews, predicting movie revenues, checking the correlation between movie popularity and the release year, checking which director (from the directors included in the data) has the highest average movie popularity, seeing which year had the highest number of English spoken movies, and examining how has the popularity of English spoken movies changed over the years?

1. **Implementation & Analysis:**

To start, we would clean the data, integrate it and then, take a look at what needs to be modified in it so that the data is all set for us to start our analysis. Our analysis would include predicting the revenue for the movies with linear regression, applying sentiment analysis for the reviews and perform personalized movie recommendation. Some other analysis would be done on the data to gain useful insights.

* 1. **Data Cleaning, Integration & exploration:**

We started our implementation by cleaning two datasets by removing the duplicates, and dropping the columns with large null values. Then, we integrate the datasets by the column “title” and chose the columns that we needed for our analysis. We needed a third dataset to integrate with the first two to minimize the null values for the revenue column so, after cleaning it and dropping the columns we don’t need, we integrated it also by the column title. The dataset now contains two different columns for the revenue so, we needed to check if the movie contains null value in a column and an actual value in another, we would consider the actual value and if the two columns contained actual values, we would consider the first. We then drop the rows with null values in both columns and drop the two revenue columns and store the results in a new column “total\_revenue”. For some values in the total revenue column, it contained M (million) or K (thousand), so we transformed these to their integer values so that we can use them later. Some movies had multiple reviews in different rows, so we grouped them so that no movie would be repeated in multiple rows and each movie would have all its reviews in a list for further usage. However, the lists contained duplicate reviews, so we transformed the lists into sets then into lists again to remove any duplicate reviews.

* 1. **Linear Regression:**
* Spark Implementation:

We attempted to forecast the total income of films based on a variety of features, including ("director, release date, language, overview, popularity, production companies, runtime, vote avg, vote count"). Due to the fact that most of these attributes were categorical, the first step was to transform them to numerical using the "StringIndexer" function, which is a component of the "pyspark.ml.featrue" library, each object is initialized with the names of an input column and an output column. Then, using the 'Pipeline' class, a pipeline was formed, with a list of 'StringIndexer' objects being passed as stages. Then, we check if "\_INDEX" is present in each column name in the list by iterating over them, if "\_INDEX" is present in a column name; add it to the list “lis”. This list should be updated to remove the column "total\_revenue\_INDEX". Then create a "VectorAssembler" object and specify the output column as "features" and the input columns as the items in the "lis" list. Then we start applying the transform method of the "VectorAssembler" to the dataframe and save the outcome in “assembled\_data”. And now we train the model. To do this, we constructed a Linear Regression object and set the features column to "features" and the label column to "total\_revenue\_INDEX". The 'fit' technique was then used to fit the Linear Regression model to the train data. Using the 'transform' method of the Linear Regression Model obtained from the 'fit' stage, we generated predictions on the test data. With the label column set to "total\_revenue\_INDEX" and the prediction column set to "prediction," we generated a "RegressionEvaluator" object. Using the “evaluate” method of the “RegressionEvaluator”, we lastly assessed the predictions using the RMSE (Root Mean Squared Error) and R2 (R-squared) metrics.

* Python Implementation:

To implement linear regression using python we first find the correlation between the total revenue and the other features, we needed to transform all categorical features to numerical. After doing that, the column genres needed to be expanded as a movie can be of multiple genres so, each genre became a column and its values may be 1 if the movie is that genre, 0 otherwise. We then build our linear regression model and passed all the convenient columns and split the data into train data and test data. We decided to go for 30% as the test size then, we fitted the data to our model and predicted the total revenue. Finally, we tested the model’s accuracy by r2 score.

* 1. **Sentiment analysis:**
* Spark Implementation:

We conducted sentiment analysis on the “reviews” column to analyze the reviews that the audience left on the movies included in our data. In order to ensure that only unique reviews are analyzed, we first deleted any duplicate rows from the DataFrame. We then used the "SentimentIntensityAnalyzer" function from the "vaderSentiment" library to start the sentiment analyzer. Then, we set up a function class instance to conduct the sentiment analysis. We create a function called "analyze\_sentiment" that analyzes sentiment based on the text input it receives. To get sentiment scores for the supplied text, the “polarity\_scores()” method of the “SentimentIntensityAnalyzer” was used within the function. To determine the overall sentiment polarity, subtract the compound score from the sentiment scores. Then, we classified the review sentiment as "Positive" if the compound score is greater than or equal to 0.05, "Negative" if it is less than or equal to -0.05, or "Neutral" in all other cases, and returned the sentiment label. To use sentiment Analysis UDF Register; install the required Apache Spark module for User-Defined Functions (UDFs). The “StringType” function from “pyspark.sql.types” was also used. We used the 'udf()' method to register the “analyze\_sentiment” function as a UDF and specified the return type as 'StringType()' (assuming that the sentiment label is of a string type). And “analyze\_sentiment\_udf” should be assigned the registered UDF. UDF was applied to the "reviews" column using the "withColumn()" method, and the registered UDF "analyze\_sentiment\_udf" was applied to the "reviews" column to generate a new column called "sentiment". Use “df5["reviews"]” as the input to the UDF, which will analyze each value in the "reviews" column using the sentiment analysis function and add the results to the new "sentiment" column.

* Python Implementation:

To perform sentiment analysis using python; the first step was to join the reviews that were written in the form of lists of individual reviews for each movie into a single string that contains all the reviews for a certain movie. Then we use the “SentimentIntensityAnalyzer” class from the “nltk.sentiment” module. This class is used to analyze sentiment in text by computing sentiment scores. The “polarity\_scores()” method is used to calculate sentiment scores for each review and the calculated scores are then assigned to a new column named “sentiment\_score” in the dataset. Based on the determined sentiment scores, each review is given a sentiment category. Reviews with scores above a specified positive threshold (0.2) are labeled as 'Positive', reviews with scores below a specified negative threshold (-0.2) are labeled as 'Negative', and reviews with scores between these thresholds are labeled as 'Neutral'. The sentiment categories are assigned to a new column named “sentiment\_category” in the dataset.

* 1. **Movie recommendations:**
* Spark Implementation:

First, the “regexp\_replace” function from the “pyspark.sql.functions” module is used to replace specific patterns in the "genre" column of the data frame. Each replacement is followed by casting the column to the "string" data type. Then, additional functions and modules are imported from “pyspark.ml.feature” and “pyspark.ml.linalg”. The code creates a new column called "index" using “monotonically\_increasing\_id” and drops the original “index” column. Next, a new column called “combined\_features” is created by concatenating the "overview" and "genre" columns. A pipeline is defined for text processing and feature extraction, including stages such as “RegexTokenizer”, “StopWordsRemover”, “CountVectorizer”, `IDF`, “VectorAssembler”, and “Normalizer”. The pipeline is fitted to the DataFrame, resulting in a transformed DataFrame called “df\_transformed”. A user-defined function (UDF) called “cosine\_similarity” is defined to calculate the cosine similarity between two vectors, and it is registered. A cross-join is performed between “df\_transformed” and itself to obtain all pairwise combinations, and the “cosine\_similarity\_udf” is applied to calculate the similarity between the feature vectors. The user is prompted to enter their favorite movie name, and “df\_similarity” is filtered to select rows matching the user's input. The resulting DataFrame,   
“df\_selected”, is ordered by similarity in descending order. Finally, the top 10 recommended movies from “df\_selected” are printed.

* Python Implementation:

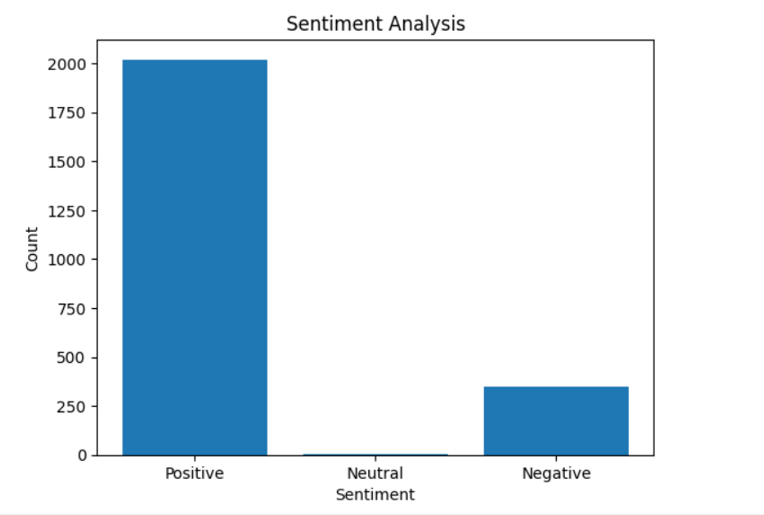
The code begins by modifying the "genre" column in the dataframe (`df5`) to ensure consistency in the genre labels. It uses the `str.replace()` method to replace specific genre labels with standardized versions, such as removing commas and ensuring consistent spacing. Next, the code combines the "overview" and "genre" columns into a single feature called "combined\_features". This concatenation allows for a comprehensive representation of each movie's content and genre. To convert the text data into numerical feature vectors, the code uses the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer from the `TfidfVectorizer` class. It applies the vectorizer to the "combined\_features" column and generates feature vectors (`feature\_vectors`). After obtaining the feature vectors, the code calculates the similarity scores between movies using cosine similarity. The `cosine\_similarity()` function from the scikit-learn library is employed for this purpose. To recommend movies based on user input, the code prompts the user to enter their favorite movie name. It then finds the closest matches for the input using the `get\_close\_matches()` function from the `difflib` module. The list of all movie titles from the dataframe is compared with the user's input, and a list of close matches is generated. The code presents the close match options to the user and asks them to choose the intended movie by number. Once the user selects the movie, the code retrieves the index of the selected movie in the dataframe. The code then generates a list of similarity scores between the selected movie and all other movies using `enumerate()` and `similarity[index\_of\_the\_movie]`. The list is sorted in descending order based on the similarity scores using the `sorted()` function. Finally, the code prints the top 10 movies with the highest similarity scores as movie recommendations for the user.

* 1. **Correlation analysis and other movie-related questions answered:**

To find if the relation between movie popularity and the movie released year is strong enough, we made a correlation analysis and plotted the result. Another question we wanted to answer is which director has the highest average movie popularity so, we grouped the movies popularity by the director’s name and then for each group we calculated the average and compared each director average. The following question to answer was which year had the most English released movies, to answer this question we found the unique values for the movies’ language column and then filter all the English spoken movies and group them by the year released and count how many movies were released in each year. Our final analysis was to plot how the popularity of English spoken movies changed over time.

**Results and Analysis**

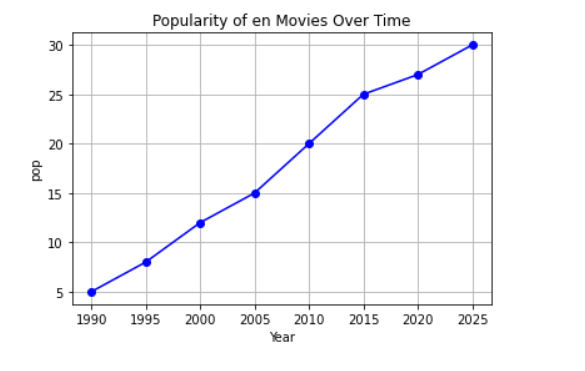
After exploring our data and performing different analytic techniques, some results were reached, and they will be discussed in this section.

* The most correlated features to the movie’s revenue were TMDB vote count, adventure and budget. That led us to the conclusion that the adventure genre is the most genre to gain revenue which means that the adventure genre is the most popular genre among the genres included in our dataset. Also, budget being correlated with the movie’s revenue means that the more money spent on the movie production, the more money it makes.
* A linear regression model was built to predict the movies’ revenues. We build the model using python and measured the accuracy with r2 score and it reached 49% and we implemented the linear regression model using spark and the accuracy was tested to be 31.7%. We deduced the reason behind the accuracy being relatively low was due to the analyzed dataset was from 3 different datasets integrated together.
* A sentiment analysis was applied on the movies’ reviews written by audience to express their opinion, and most of the movies included in our dataset had positive reviews.
* For the movie recommendations, we were able to provide a personalized list of 10 movies similar to the movie that the user enters.

A screenshot of a computer

Description automatically generated with medium confidence

* A screen shot of a graph

  Description automatically generated with low confidenceFrom the correlation analysis done between movie release year and movie popularity, we wanted to check which years had the most popular movies so that further analysis could be done on these years to check which factors affected the popularity the most.
* In our dataset, the director with the highest average movie popularity was “Stanley Kubrick”. Results of similar analysis can be a factor that can assist movie production companies in decision making.
* A graph was plotted to show how the popularity of English spoken movies changed over time. We noticed that the popularity of these movies is increasing gradually.

A picture containing text, screenshot

Description automatically generated

**Conclusion and Future Work**

In conclusion, this project conducted a comprehensive analysis of movie data, including personalized movie recommendations, sentiment analysis of movie reviews, and prediction of movie revenues. The results offer insightful information about audience preferences, factors that affect earnings in the film business and more. These results can help production businesses and filmmakers improve their decisions by utilizing these data analytics tools. While the current project has offered useful details on movie data analysis, the methods and results presented can be used as a starting point for additional studies and real-world applications in the film industry, and there are several areas for future work that can further increase our understanding of the film industry and improve the accuracy of predictions. Potential future work can include:

* **Enhancing Sentiment Analysis**: Deeper analysis can be done on movie reviews to not only classify them into positive, negative and neutral reviews, but also identifying certain aspects of a film that contribute to these sentiments. This level of analysis can offer more thorough understandings of the factors influencing audience opinions.
* **Models That Can Predict Box Office Success**: Adding extra factors like marketing campaigns, release dates, star power, and competition can help create improved prediction models for box office success. Filmmakers can make better decisions and improve their marketing strategies by using more precise revenue prediction models.

**References**

[1] <https://www.kaggle.com/datasets/andrezaza/clapper-massive-rotten-tomatoes-movies-and-reviews?select=rotten_tomatoes_movies.csv>

[2] <https://www.kaggle.com/datasets/andrezaza/clapper-massive-rotten-tomatoes-movies-and-reviews?select=rotten_tomatoes_movie_reviews.csv>

[3] <https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=movies_metadata.csv>

[4] <https://www.kaggle.com/datasets/clementmsika/mubi-sqlite-database-for-movie-lovers?select=mubi_movie_data.csv>

[5] <https://www.kaggle.com/datasets/clementmsika/mubi-sqlite-database-for-movie-lovers?select=mubi_ratings_data.csv>

**Appendix**

Github link that contains the code:

<https://github.com/doniaahmed8/BigData_Project>