Implementation of a big data system for predicting movie popularity

Alessia Marcolini 194274

alessia.marcolini@studenti.unitn.it

Anna Bertani 214047

anna.bertani@studenti.unitn.it

Introduction

The main objective of this project is to design and implement a big data system that analyses and predicts the popularity of movies by looking at popularity of the trailer on Youtube and ratings from RottenTomatoes.

The first question that came to mind was to define the concept of *popularity* and, then, to investigate if a movie can be considered popular or not.

Although this interrogative was very fascinating, the main point was to understand and to use the right architectures and analytics engines for data processing.

This report is organized as follows. The data we used and the preprocessing steps are explained in the first section. The second section describes the methodology adopted, that is which data storage systems we used to handle the data, and a description of the predictive analysis. The final part is dedicated to the results found and conclusions.

Data description

Movie popularity depends on several factors, such as classical factors, including cast, producer, director, etc, or the social factors in form of response of the society on various

online platforms.[1] Therefore, we are integrating different data sources, namely OpusData, OMDb and YouTube.

We retrieve movie data from the OpusData website[2], "the most in-depth collection of movie financial information, containing information on over 25,000 movies and counting, including budgets, US and international box office, US home market consumer sales and rentals, as well as biographical information on over 120,000 actors, directors, producers and technicians".[3]

We consider only movies produced from 2010 onwards and we select the following movie attributes: rating (expressed using the Motion Picture Association of America[4] system), genre and sequel (expressed as 0 or 1, where 1 means that that movie is a sequel of another one, 0 otherwise), the total box office revenue (given by the sum of domestic and international box office revenue) and the production budget. The movies with no data for box office revenue or production budget are removed from the movies collection. So our initial database consists of 1164 movies, which we refer to as the *OpusData*.

We also extract related movie data of three different websites: the Internet Movie Database[5] (IMDB), RottenTomatoes[6] and Metacritic[7].

We use the OMDb (Open Movie Database) Python Library to retrieve data from IMDB, Rotten Tomatoes and Metacritic.

We use the movie names from *OpusData* to search and retrieve the following information: the runtime (expressed in minutes), the director, the actors, the production country, the awards, the number of votes on IMDB and the ratings from IMDB, Rotten Tomatoes and Metacritic. Regarding the awards, we distinguish among the number of awards nominations and the number of awards won. Awards such as the Golden Globe, the Oscar and the BAFTA has been counted separately from the other awards. Being rankings coming from different metric systems, we scale them between 0 and 1.

[1] Bhave, Anand, et al. "Role of different factors in predicting movie success." 2015 International Conference on Pervasive Computing (ICPC). IEEE, 2015.
[2] https://www.opusdata.com/data.php
[3] Ibid. 2.
[4] https://www.motionpictures.org/film-ratings/
[5] https://www.imdb.com/
[6] https://www.rottentomatoes.com/
[7] https://www.metacritic.com/

Moreover, we use YouTube video API data to increase the number of features of our dataset. From the movie title, we extract video statistics for the official trailer of the movie: the numbers of views and the number of like/dislike.

From these information we derive the following scores, which are expected to be predictors of movie popularity:

- engagement score, calculated as $engagement\ score = \frac{likes + dislikes}{views}$
- positive engagement score, calculated as $positive\ engagement\ score = \frac{likes}{dislikes}$

In order to estimate the movie popularity, we decide that a movie is popular and successful if the profit of the movie is positive, as suggested by Rhree et. al[8].

The profit is calculated as:

$$profit = \frac{total\ box\ office}{2} - production\ budget$$

If the profit is positive, the movie is considered to be successful; it is labelled as a flop otherwise.

After these preprocessing steps, our dataset consists of 616 movies.

Methodology

Predictive Analysis

We trained a Random Forest classifier to predict the popularity of a movie as a binary label, i.e. "success", comprising 3/5 of the dataset) vs "flop", comprising 2/5 of the dataset (see section Data Description). The data were first splitted into training set and test set with the partition 70/30. We leveraged `ParamGridBuilder` to create the parameters grid used to select the best number of trees (n_t) of Random Forest among the values $n_t \in [100, 300, 500]$. To evaluate the performance of the model and to avoid overfitting effects, we performed our experiments in a 5-Fold Cross Validation schema. We chose accuracy to assess the performance of the classifier: given the balance between the two classes, we can consider the accuracy as a reliable evaluation metric.

The best model ($n_t = 500$) could predict the popularity of a movie with an accuracy of 0.730 on the test set, showing very little overfitting (ACC= 0.855 on the training set).

[8] Rhee, Travis Ginmu, and Farhana Zulkernine. "Predicting movie box office profitability: A neural network approach." *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2016.

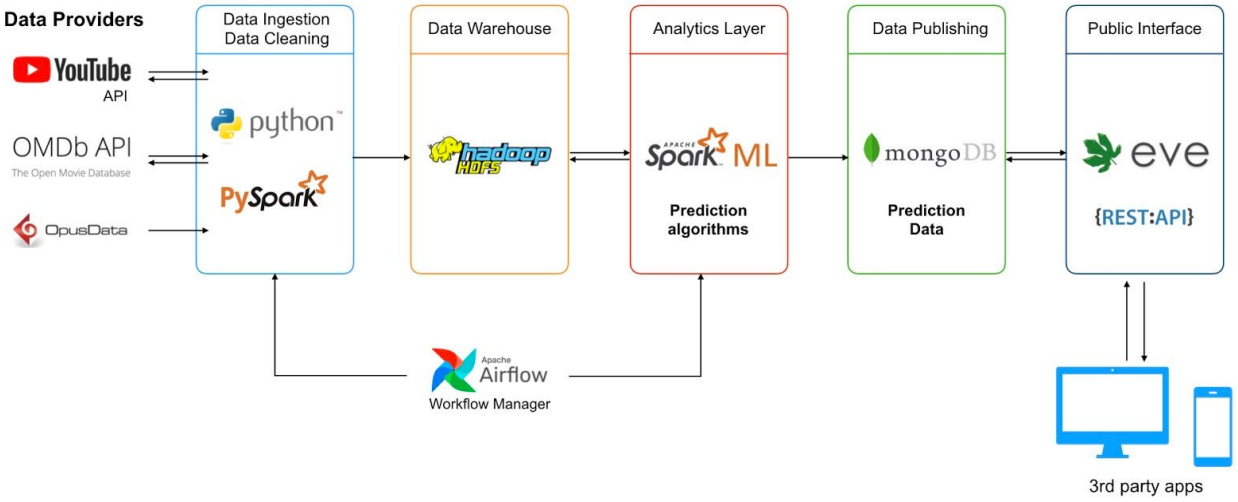
The internal ranking of Random Forest revealed that the three most important features used to predict the success of a movie were the number of votes of imdb, genre and nominations.
As the data are constantly updated, the model is trained on a daily basis, meaning that the prediction could change every day.

Architecture overview

After the stages of data ingestion and data cleaning, the data were saved on a HDFS cluster, which is suitable to store and analyze huge amount of data. We setup a local HDFS cluster using a Docker container for the DataNode and another container for the NameNode.
All of the data preprocessing steps were carried out using Python and PySpark (in particular using the SQL module). The Random Forest classifier was built using PySpark ML.
The classifier predictions are then saved into a MongoDB instance, a distributed noSQL document-based database.
In order to make available movie popularity predictions for 3rd-party clients, we use the Eve Python library[9], an open source REST API framework, powered by Flask, MongoDB, and Redis, offering native support for MongoDB data stores. We provide the `movie_popularity` endpoint, which offers all the resources saved in the database; the results can be filtered providing the `imdb_id` of the requested movie as an "additional lookup" of the endpoint. The response will contain the most recent information related to the specific movie.
We use Apache Airflow as the workflow manager and scheduler: our preprocessing and analytics tasks are organised following a dependency DAG, which Airflow make sure are executed only if the respective prior tasks are successful. Our entire workflow is scheduled to be executed once a day, in order to retrieve the latest data from OMDb and YouTube API and to train a new classifier on the updated data.
The described architecture, including the HDFS cluster, the Spark cluster and the MongoDB instance, is entirely hosted on a Microsoft Azure Virtual Machine.

[9] https://docs.python-eve.org/en/stable/

Here below a simple graphical representation of our project.



Conclusions

In this project, we implemented a big data system for predicting movie popularity. Although it was not straightforward to give a definition for popularity, we have found that the popularity of a movie can be accurately predicted by some classical features like votes of imdb, genre and nominations for an award.

We also defined the feature of *success,* evaluating in this way if a movie is considered as a success or a failure.

Furthermore, also YouTube video statistics demonstrates to have an effect on the movie popularity.

Moreover, big data tools such as HDFS and Spark demonstrate to be fundamental for the realization and implementation of this big data system and this was also an opportunity to learn much more about big data architectures.