UNIVERSITÀ DI PISA

Master's degree in Artificial Intelligence and Data Engineering

Data Mining and Machine Learning

MovieBox

Academic year 2021-2022

Francesco Hudema, Massimo Merla

Github link: https://github.com/MrFransis/MovieBox

Table of contents

Introduction

MovieBox is an application which allows users to create a playlist of movies and get suggestion on new movies to watch based on their tastes.
The application can automatically classify the movies into their genre.
The classification part uses a machine learning algorithm to classify the correct genre, selecting it among the most frequent genres.

Dataset

To create the movie genre classifier, we used a labelled dataset of movies obtained from The Movie Database (www.themoviedb.org), a community build movie database.
We extract the movies metadata using a Python library for TMDb API (https://github.com/AnthonyBloomer/tmdbv3api).
For our scope we downloaded 38MB of movies data, in total (~56k entities).

The dataset has the following structure:

	adult	backdrop_path	genre_ids	tmdb_id	imdb_id	original_language	overview	popularity	poster_path	production_companies	production_countries	release_date	title	vote_average	vote_count	casts
0	0	/hQ4pYslbP22TMXOUdSfC2mjWrO0.jpg	Drama	2	tt0094675	fi	Taisto Kasurinen is a Finnish coal miner whose...	10	/ojDg0PGvs6R9xYFodRct2kdI6wC.jpg	Villealfa Filmproductions	Finland	1988-10-21	Ariel	7	154	Turo Pajala,Susanna Haavisto,Matti Pellonpää,E...
1	0	/l94l89eMmFKh7na2a1u5q67VgNx.jpg	Drama	3	tt0092149	fi	An episode in the life of Nikander, a garbage ...	8	/njO1hspawPof0mJmlgfjuLyJuRN.jpg	Villealfa Filmproductions	Finland	1986-10-17	Shadows in Paradise	7	154	Matti Pellonpää,Kati Outinen,Sakari Kuosmanen,...
2	0	/bCKLsO2MJWzYYBJoavGOZN5iqGn.jpg	Crime	5	tt0113101	en	It's Ted the Bellhop's first night on the job....	16	/75aHn1NOYXh4M7L5shoeQ6NGykP.jpg	Miramax,A Band Apart	United States of America	1995-12-09	Four Rooms	6	2087	Tim Roth,Jennifer Beals,Antonio Banderas,Valer...
3	0	/5aXp2s4l6g5PcMMesIj63mx8hmU.jpg	Action	6	tt0107286	en	While racing to a boxing match, Frank, Mike, J...	12	/rYFAvSPiQUCebayLcxyK79yvtvV.jpg	Universal Pictures,Largo Entertainment,JVC	Japan	1993-10-15	Judgment Night	6	224	Emilio Estevez,Cuba Gooding Jr.,Denis Leary,St...
4	0	NaN	Documentary	8	tt0825671	en	Timo Novotny labels his new project an experim...	3	/x7Sz339F2oC8mBf0DHCQpKizXaL.jpg	inLoops	Austria	2006-01-01	Life in Loops (A Megacities RMX)	8	18	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
56379	0	/hLZkOUrh4m7DYbzv3jDmzmag43.jpg	Drama	99994	tt0027727	en	Best friends Kenneth Reynolds and Raymond Jord...	1	/bXD0LiSJqEF1EMDPk55EU7desdw.jpg	Republic Pictures	United States of America	1936-05-26	Hearts in Bondage	4	5	James Dunn,Mae Clarke,David Manners,Charlotte ...
56380	0	/jQvfxZDmxbKujlPJaJrYxKyunmEjpg	Action	99995	tt0032021	en	When her brother is killed by sabotage, Irene ...	1	/oAdH287KAa0CnuLk24S16W9i9ez.jpg	RKO Radio Pictures	United States of America	1939-04-14	They Made Her a Spy	0	0	Sally Eilers,Allan Lane,Fritz Leiber,Frank M. ...
56381	0	/uAG9D0mTEHhYsJ20W8PkBi35Dn8.jpg	Drama	99997	tt0027196	en	An army sergeant inspires his son to become an...	1	/enNyJlQOMDnvOOCRsUEN9k67gxJ.jpg	Metro-Goldwyn-Mayer	United States of America	1935-03-23	West Point of the Air	7	1	Wallace Beery,Robert Young,Lewis Stone,Maureen...
56382	0	/wS0lvCKBe2FdCHhXQg5PU8lozbG.jpg	Drama	99998	tt0032130	en	Jerry tries to out compete his older brother C...	1	/zMKaZT963XmhGTmC5q23j45Kt9n.jpg	Warner Bros. Pictures	United States of America	1939-02-11	Wings of the Navy	6	3	George Brent,Olivia de Havilland,John Payne,Fr...

Figure 1

Requirements

Functional Requirements:

Unregistered User:

- Unregistered User can register a Registered User account on the application

Registered User:

- Registered User can search for Movies by title
- Registered User can add and remove Movies from his/her personal list
- Registered User can browse suggestion about Movies.

Admins:

- Admins can add/remove and update movies in the application’s database

Non-Functional Requirements:

- The application must be user-friendly
- The application needs to provide fast response to the user

Specifications

Main Actors

• Unregistered User: User that is not registered on the application, to access he must sign-up.
• Registered User: User that is registered, he can access application by logging-in.
• Administrator: User can add new movies to the database.

Use Case Diagram

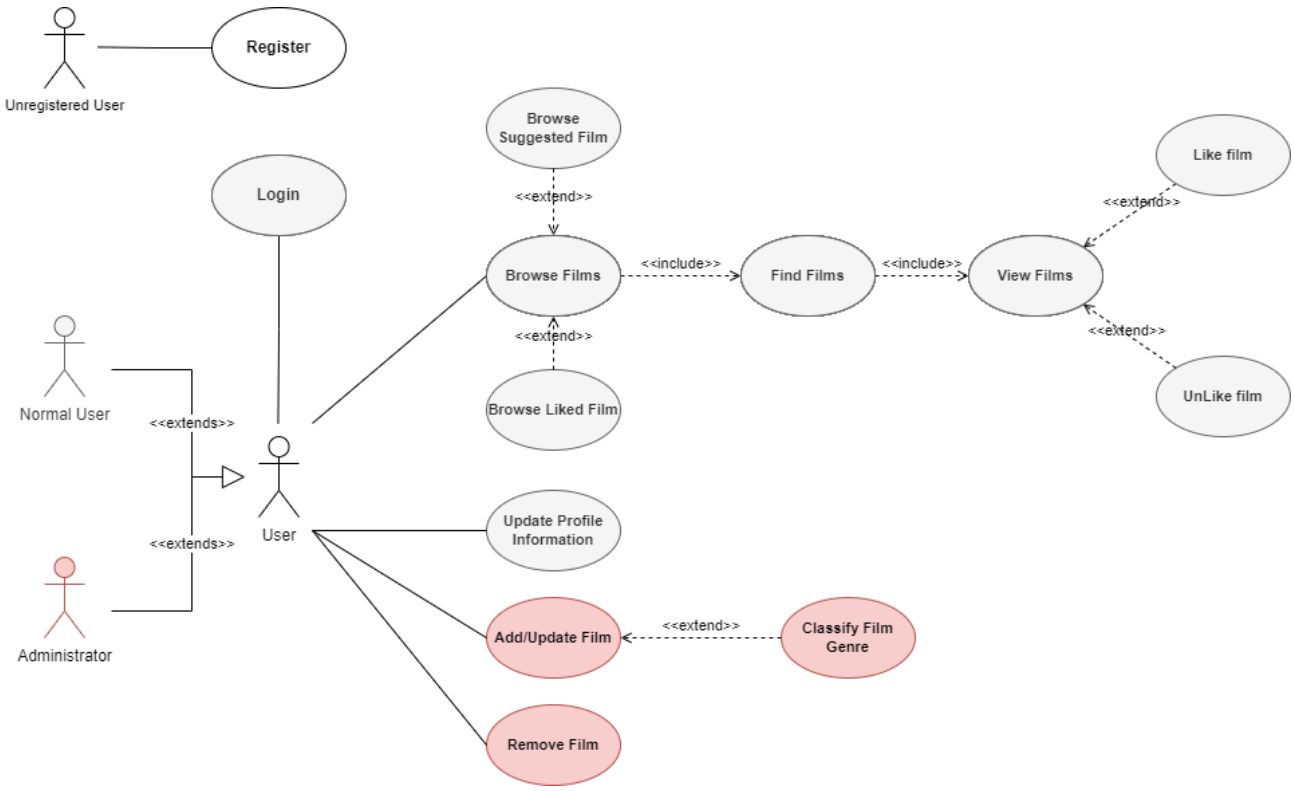


Figure 2

Legend:

- The circles in grey describe actions available to normal Users and Administrators.
- The circles in red describe actions available to Administrators.

UML Class Diagram

There are two main entities: User and Film. User is a generalization of the two main actors (User and Administrator) of the use case diagram, it resolved adding one attribute to the class user for specify the role.

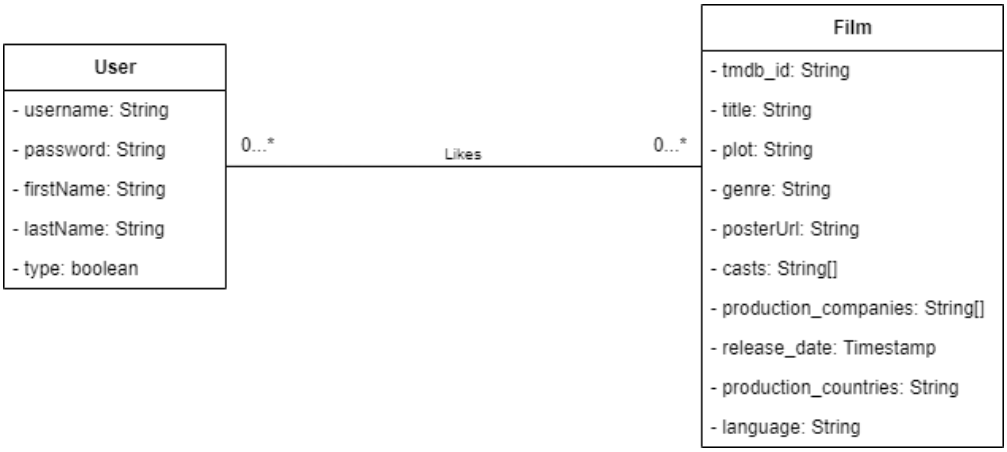


Figure 3

User:
The class implements the main actor of application.
Attributes:

- username: Username of the user
- password: Password of the user
- firstName: First name of the user
- lastName: Last name of the user
- type: Identifies the user's role (false: User, true: Administrator)

Film:
The class implements the film entity.
Attributes:

- tmdb_id: Id of tmdb_id film
- title: Title of the film
- plot: Overview of the film
- genre: Genre to which the film belongs
- posterUrl: Link of the film cover
- casts: List of actors in the film
- production_companies: List of film production companies
- realease_date: Publication date
- production_country: Country of production of the film
- language: Original language

Data Model

The data are stored in a graph database, using Neo4J DBMS, to store the relationships between users and movies.

The graphDB nodes with their attributes are the following:

- User: {username, password, firstname, lastname, type}
- Film: {tmdb_id, title, plot, gerne_ids, posterURL, casts, production companies, release_date, production, country, language}

Between the two entities there is the following relationship:

- (:User)-[:LIKES]->(:Film): each user can likes one or more movies, and it is used for showing suggestion based on movies genre and likes.

There is also the necessity to use some constraints:

- The username of the user must be unique and always present
- The id of the movies must be unique and always present

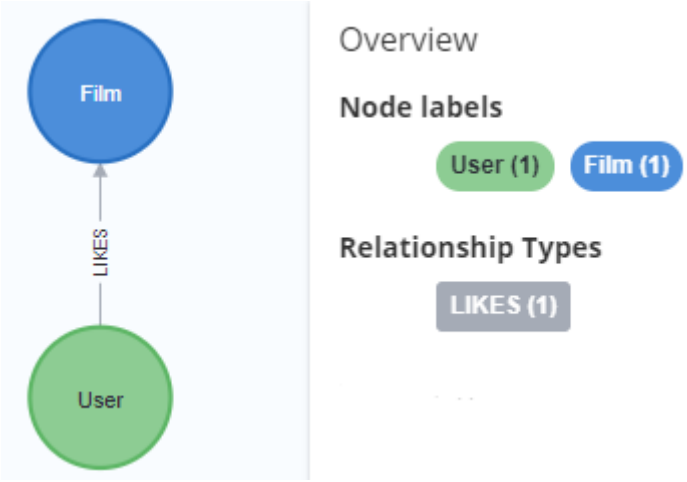


Figure 4

Machine Learning

Categorizing movies makes it easier for the viewer to discover what he or she likes and will want to see. Movie genres are formed by conventions that change over time as new genres are invented and the use of old ones are discontinued. Often, works fit into multiple genres by way of borrowing and recombing these conventions.

Precisely identifying the genre to which the movies belong is not a simple task, as the number of genres is extremely vast, and many movies can belong to multiple genres.
We decided to retain only the top seven most frequent genres present into the dataset, that are “drama, crime, action, documentary, animation, comedy and horror”.

In this study we had to deal with text classification using machine learning technique that assigns a set of predefined categories, our movies genre, to open-ended text.
We must consider also that not all classifications algorithms support multi-class classification, many algorithms are designed for binary classification and do not natively support more than two classes, for this reason we chose to use a One-vs-rest strategy, that is build-in in several algorithms implemented in scikit-learn library. This strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other class.

Analysis

The analysis of the dataset was performed in Python, using scikit-learn an open-source library that provides a wide range of tools for data analysis; all the pre-processing and algorithms evaluation steps are present in the Jupyter notebook files hosted on the GitHub repository.

Data Pre-processing

In this step we had to remove all the incomplete or null rows, and those that had a too short plot that can compromise the task to distinguish the genre to which it belongs and hold only the movies’ information whose genre is among the kept ones. In Figure 5 is shown the distribution of the genre considered for classification.

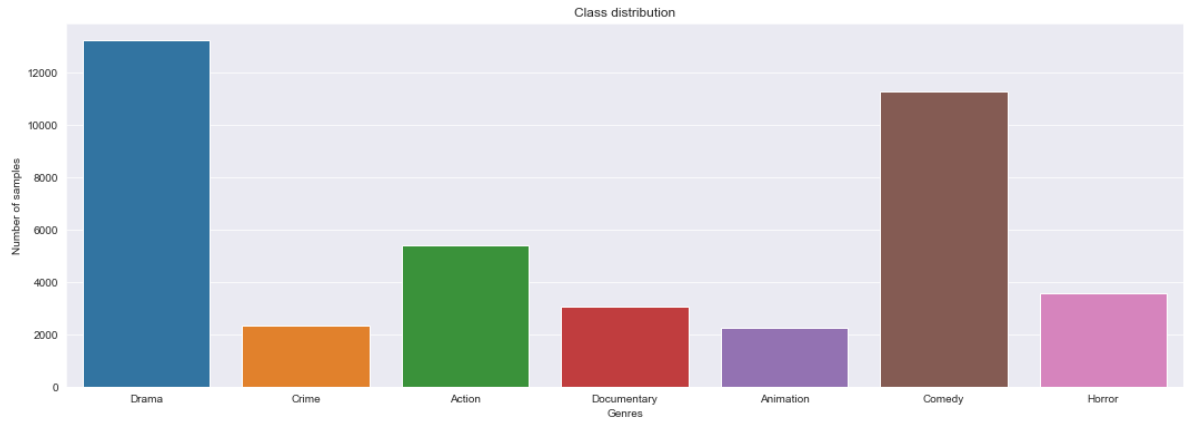


Figure 5

It was also necessary to do a text cleaning operation:

- remove punctuation and extra-spaces
- make all the text lowercase
- remove stop words

genre_ids	overview
Drama	taisto kasurinen is a finnish coal miner whose...
Drama	an episode in the life of nikander a garbage m...
Crime	its ted the bellhops first night on the job an...
Action	while racing to a boxing match frank mike john...
Documentary	timo novotny labels his new project an experim...
...	...
Drama	best friends kenneth reynolds and raymond jord...
Action	when her brother is killed by sabotage irene e...
Drama	an army sergeant inspires his son to become an...
Drama	jerry tries to out compete his older brother c...

Figure 6

Since different films belonging to different genres, they have terms in common that can lead to errors in the classification, we have analyzed the most frequent terms for each genre and eliminated those that appear frequently and are common in different genres.

The most frequent terms and the distribution of plots by number of words after this phase are represented below:

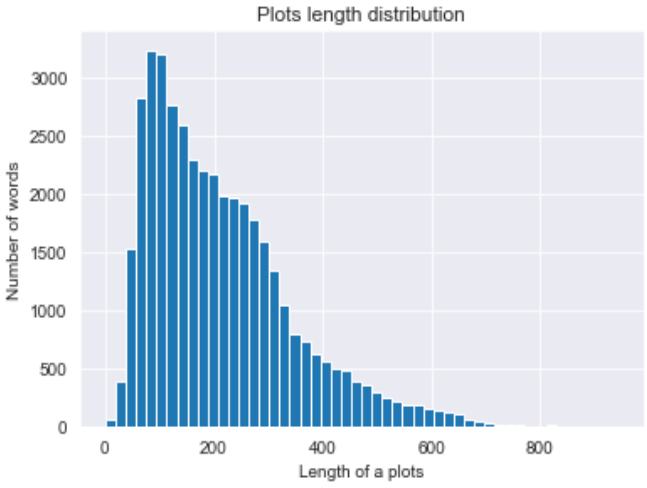


Figure 7

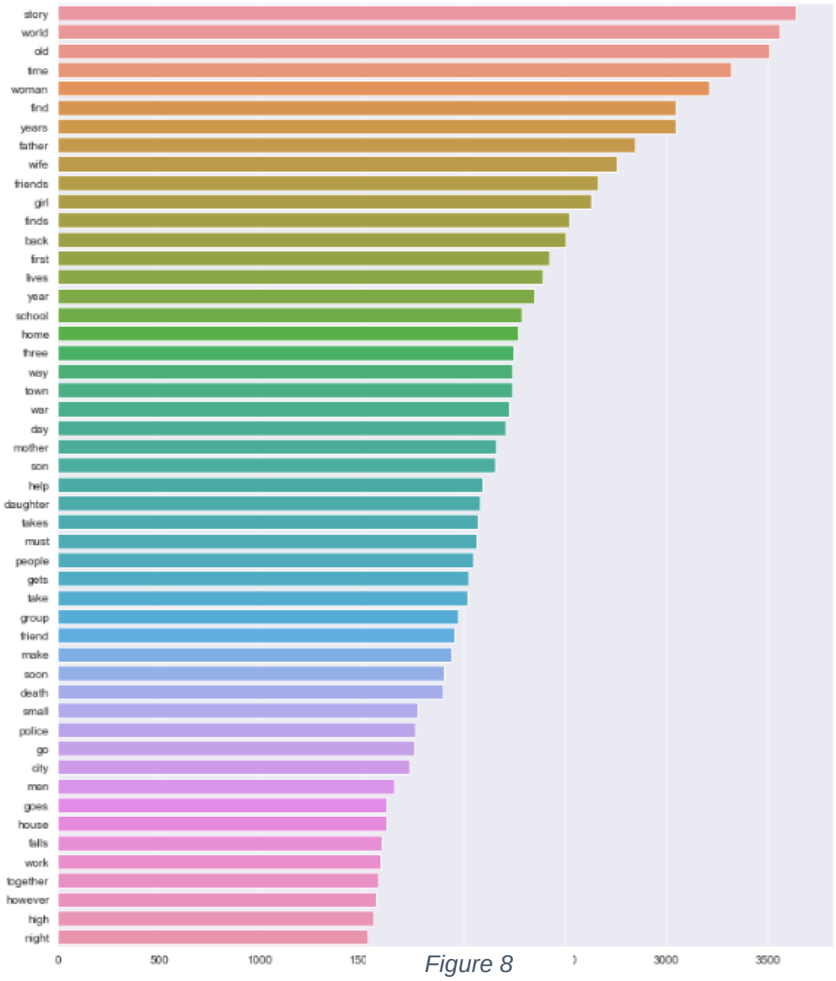


Figure 8

It was also necessary to define a stemming method to reduce each token to its base or root form, we use the stemming algorithm present in python nlkt library for English language.

To classify text, it is necessary to transform it into a numerical vector, to do this we used the *TfidfVectorizer* method provide in Scikit-Learn library, that convert a collection of text documents to a matrix of token counts and transform it to a normalized tf-idf representation.

Feature selection

For testing the different algorithms we also applied a feature selection technique for search through all possible combinations of features in the data, a subset of attributes that works better for prediction. We use the *SelectKBest* method, implemented in scikit-learn library, for extracting the k highest scoring features, testing with different scoring functions: *chi2* and *f_classif*.

Rebalancing

The dataset analyzed is not balanced so we tried to apply three different techniques to approach the imbalanced dataset:

- Oversampling: it consists in duplicating samples in the minority classes by picking them with replacement randomly.
- Undersampling: it undersample the majority classes by randomly picking samples without replacement.
- SMOTE: in generates synthetic example from the existing sample of the minority classes.

In this case the performances of the models are lower than those without rebalancing.

Balancing every movie's genre will give to each genre a probability of being classified *1/n_genres*, so classifier will forget about actual distribution of plots in the original sample. Instead without rebalancing frequent cases will get fewer misclassifications, in this case balancing the dataset means loose information about appearance frequencies, and it would mean train the model on a different distribution compared to what will appear in the distribution of new samples to classify.

Models' evaluation

The following classifiers were tested for genre's classification:

- LinearSVC
- MultinomialNB
- LogisticRegression
- RandomForestClassifier
- KNeighborsClassifier
- BernoulliNB

Initially we evaluated the performance of the previous algorithms without applying feature selection techniques, while in a second step we evaluate a second time the algorithms applying feature selection techniques using different score functions and a different number of selected features to find the best parameters. To perform these evaluations, we used the GridSearchCV method implemented in scikit-learn that allow us to specify different parameters and find the ones that get a higher score.

For each evaluated algorithm we compared the best result obtained. In the following table are presented with different scoring metrics and the computational time needed for training the model.

Algorithm	Accuracy	Precision	Recall	F1_score	Time to build the model
BernoulliNB	0.582	0.586	0.582	0.579	0.017
KNeighborsClassifier	0.445	0.449	0.445	0.440	0.009
LinearSVC	0.594	0.594	0.594	0.589	0.992
LogisticRegression	0.593	0.602	0.593	0.583	10.399
MultinomialNB	0.480	0.589	0.480	0.430	0.033
RandomForestClassifier	0.383	0.597	0.383	0.274	0.874

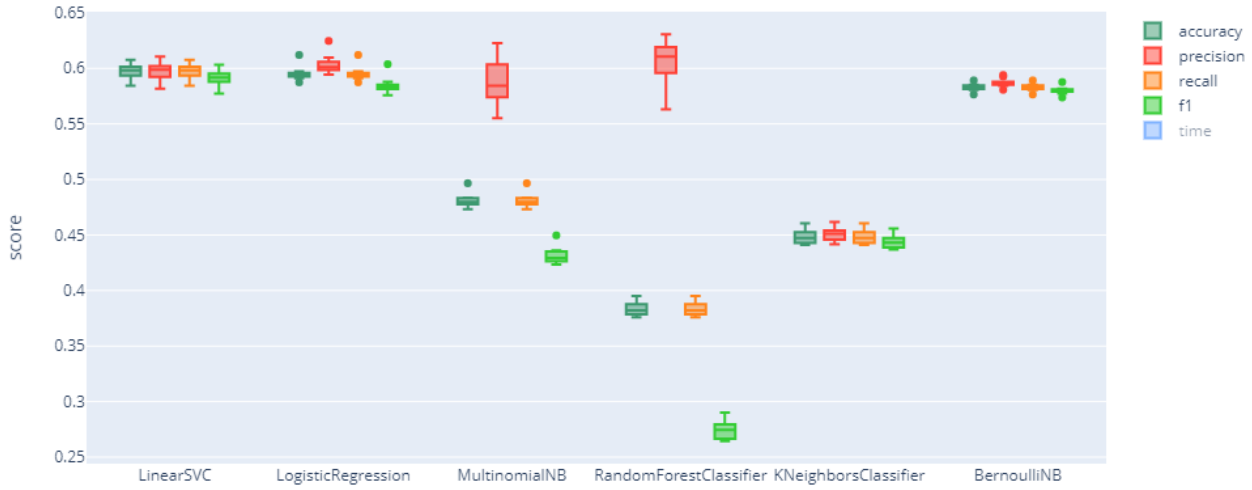


Figure 9

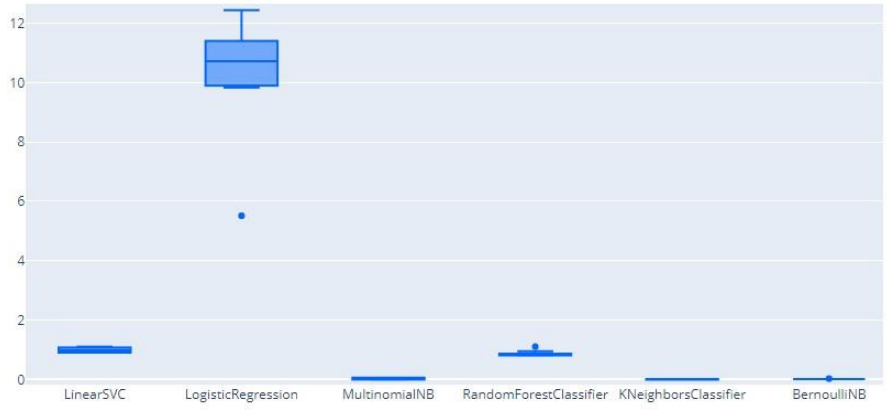


Figure 10

T-Test (F1_score) **p-value**	**LogisticRegression**	**BernoulliNB**
LinearSVC	0.1015	0.0013
LogisticRegression		0.0777

From the T-test table we can see that there is a statistical difference only between LinearSVC and BernoulliNB.

We chose to implement in the application the model created with LinearSVC, because compared to LogisticRegression has a lower computational training time and compared to other models although in some cases it has a lower precision, we can attribute this difference to the fact that the other model predict very well the majority class but not the other in the same way (we can see it in the confusion matrix).

Below, is shown the confusion matrix for each method evaluated with the best parameters:

LinearSVC:

The best result is obtained applying features selection technique with chi2 scoring function and limit the number of features to 20000.

F1-score: 0.596

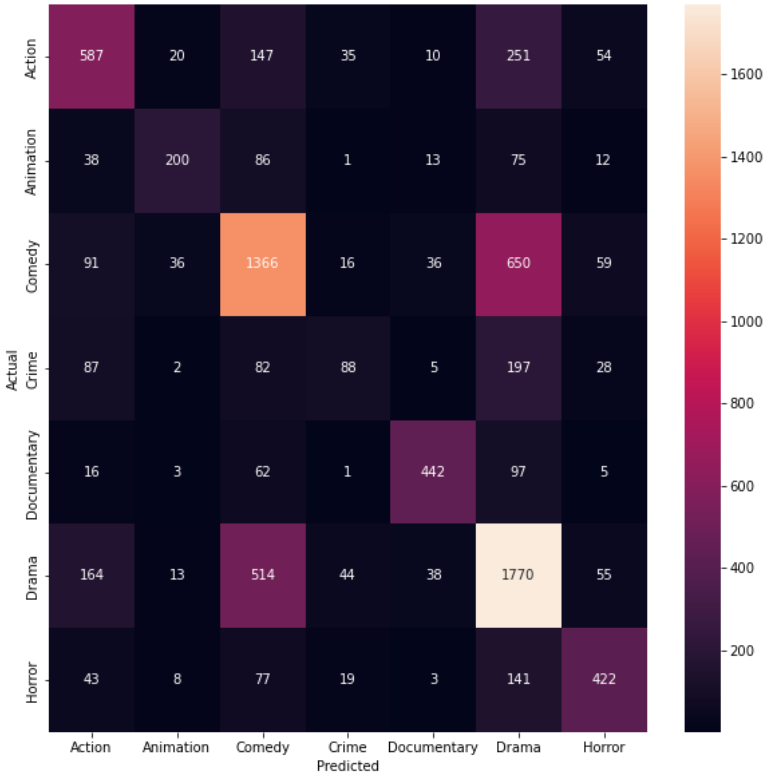


Figure 11

MultinomialNB:

The best result is obtained applying features selection technique with f_classif scoring function and limit the number of features to 5000.

F1-score: 0.481

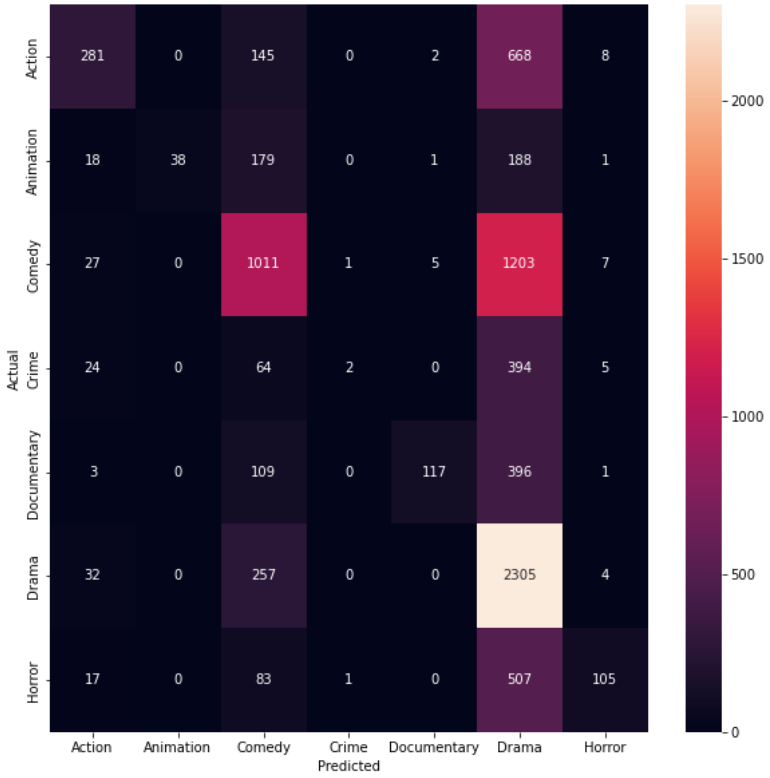


Figure 12

LogisticRegression:

The best result is obtained without applying features selection.

F1-score: 0.594

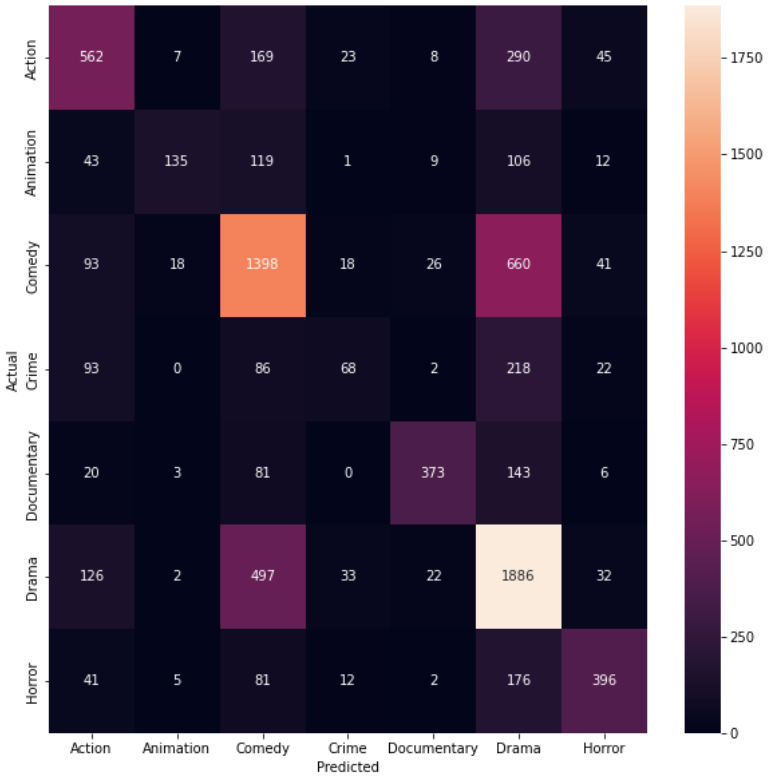


Figure 13

RandomForestClassifier:

The best result is obtained applying features selection technique with chi2 scoring function and limit the number of features to 500.

F1-score: 0.382

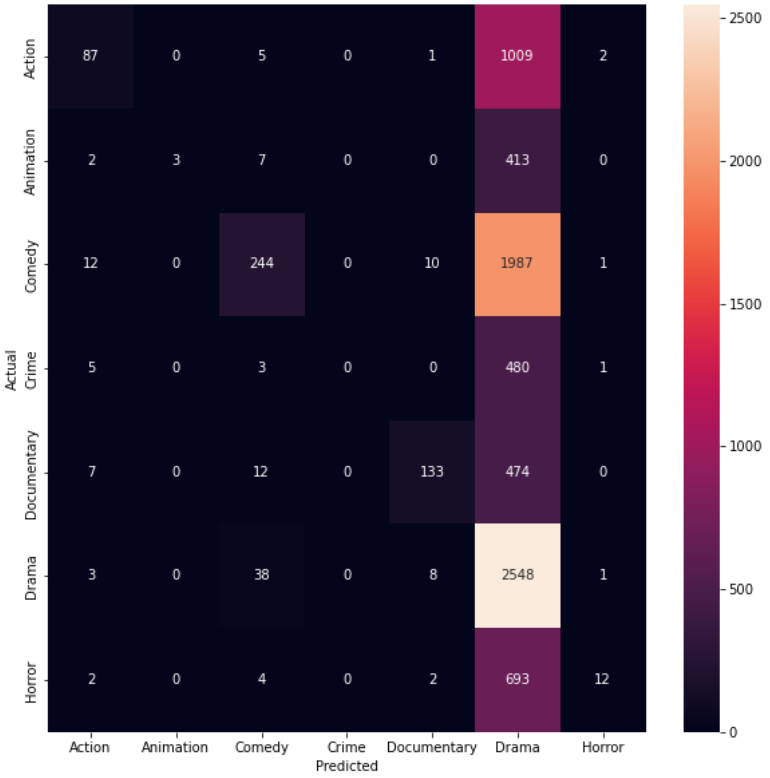


Figure 14

KNeighborsClassifier:

The best result is obtained without applying features selection.

F1-score: 0.447

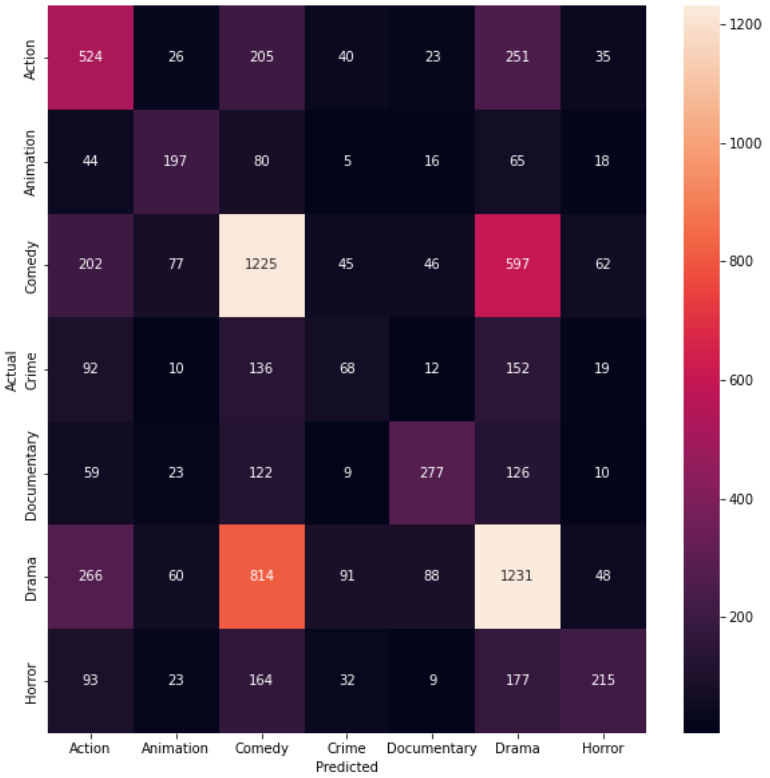


Figure 15

BernulliNB:

The best result is obtained applying features selection technique with f_classif scoring function and limit the number of features to 10000.

F1-score: 0.581

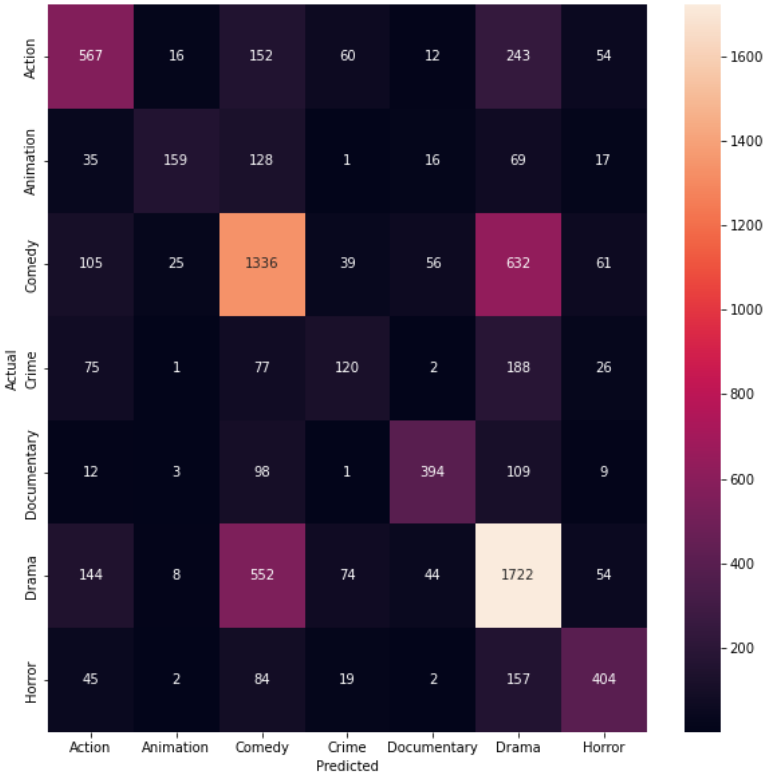


Figure 16

Architectural Design

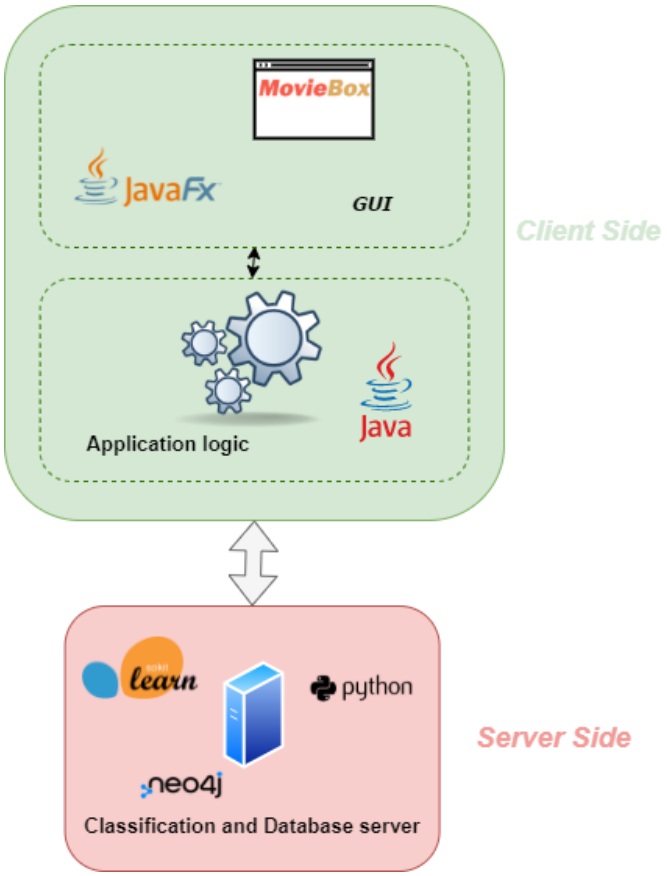
The application was implemented as client-server architecture. The client will request to query the database or to classify the film genre to the server.

The client side is composed by:

- Front-end, the graphical interface developed in JavaFX, user can use the GUI to interact with the application.
- Middleware: handle the connections and communications with classifier server and Neo4j database.

The server side is implemented in Python and consist of two nodes:

- Neo4j database, for managing the data.
- Classification server, that receives the plot to classify and send back the result to the client.



Server application

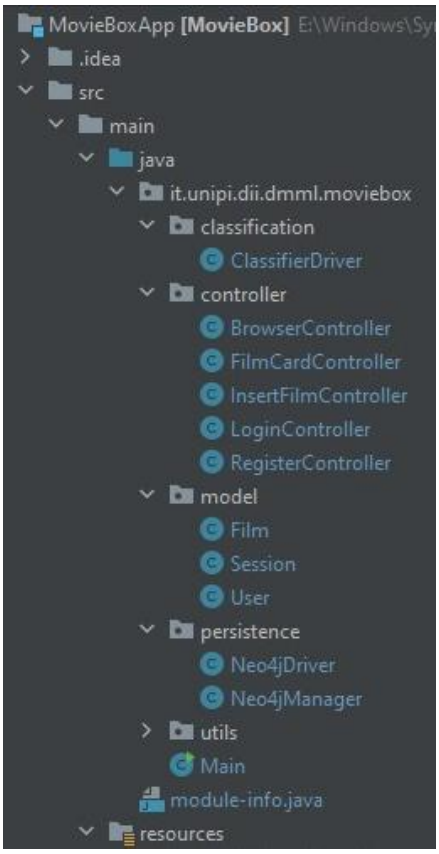
- *classifier.py*, script that create and export the machine learning model.
- *getMovies.py*, script that downloads the movies from TMDB and saves in .csv format.

- *server.py*, script that start the server: load the implemented classifier and responds to movie classification requests.
- *getUsers.py*, script that returns the complete users dataset in. json format.
- *main.py*, script that accepts server commands:
    - startServer: start classification server.
    - retriveMovies: download the dataset.
    - exportClassifier: export classification model.
    - initDB: initializes the database with movies, users and likes.
    - exit: close the application.

Java Application Package Structure

Main Packages and Classes

The MovieBox application is composed of the following packages and classes.



it.unipi.dii.dmml.moviebox.model

This package contains the classes required for the model. These classes are the java bean for our application.
Classes:

- Film: The class contains movie information.
- User: The class contains user information.
- Session: The page contains information about the logged user.

it.unipi.dii.dmml.moviebox.persistence

This package contains the classes to interface with databases.
Classes:

- Neo4jDriver: Implements the methods to manage the connection with Neo4j.
- Neo4jManager: In this class are implemented all the queries to interact with Neo4j.

it.unipi.dii.dmml.moviebox.controller

This package implements the controller part of Model-view-controller of the application. Each controller is related to a different page that is shown to the user.

Classes:

- BrowserController: this class manages the homepage and allows the user to navigate to the other pages of application and to carry out search operation and show results.
- LoginController: this class manages the login page of application.
- RegisterController: this class manages the register page of application.
- FilmCardController: this class manages the view of film card.
- InsertFilmController: this class manages the administrator page for classifying, insert, update and delete movies.

it.unipi.dii.dmml.moviebox.utils

This package contains utility functions.
Classes:

- Utils: this class contains functions for manage the configuration parameters necessary for the application and the scene change for pages.

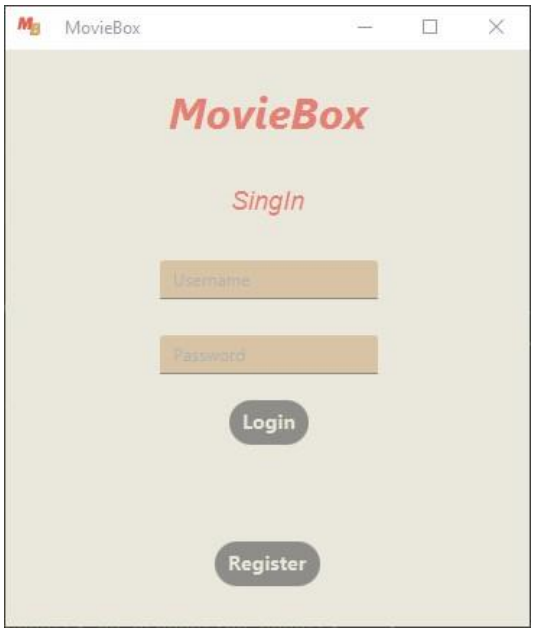
it.unipi.dii.dmml.moviebox.classification

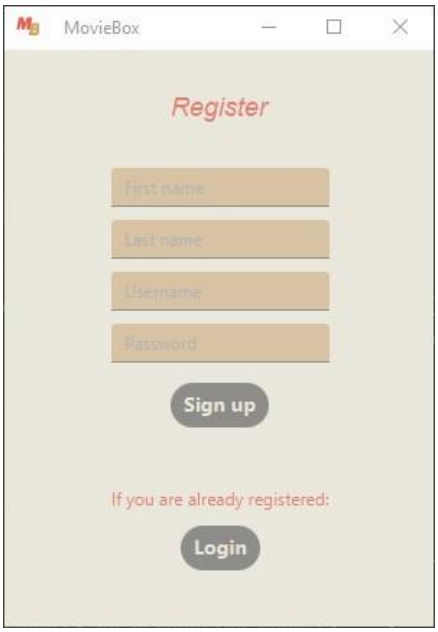
This package contains the class to interface with classification server.
Classes:

- ClassifierDriver: Implements the method to manage the connection and to interact with the sever.

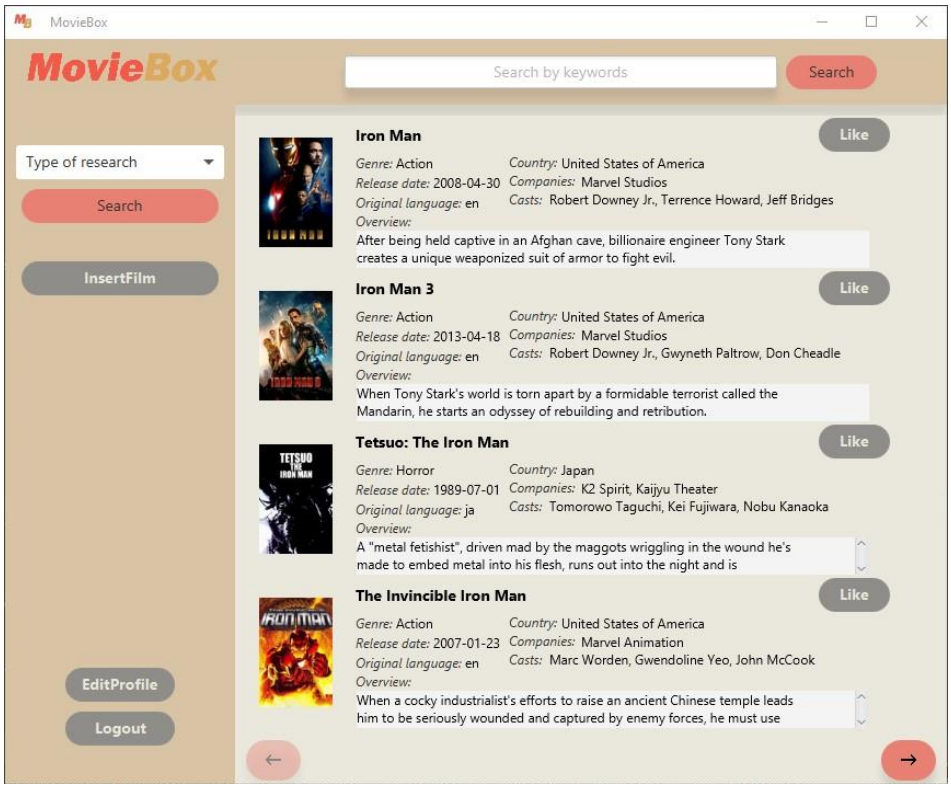
User Manual

- The first page which a user will see is the **Login** page, where a user can insert his own credentials or go to the **Register** page to create a new account.





- After the login the user is redirected to the **Browser**, in this page users can search by title the movies loaded in the database. In the left bar there is also the possibility to view liked movies and to see some suggestions. The user can navigate through his results with two buttons on the bottom. There is also the possibility to change user information and to logout from the application.



- If the user is an Administrator, he will have the button “InsertFilm” in the browser, that load the **InsertPage**. In this page is possible to insert a new movie in the database and automatically classify its genre. It is also possible to remove a film by Id.