Movie Predictor

Mini-project for Introduction to Data Science Course in University of Helsinki

Ville Tuulio, Andreas Andra, Jussi-Pekka Martikainen

28th of October 2017

1 Introduction

Movie Predictor is a topic for our mini-project done in Introduction to Data Science course in autumn 2017. The project group consisted of three students: Ville Tuulio, Andreas Andra and Jussi-Pekka Martikainen. Our support person from the course was TA Johanna Bouri. Purpose of the mini-project was to show in practice what the students had learned during the course.

2 Ways of work

Our mini-project was a group effort as it was supposed to be. The workload was shared quite intuitively but happened still to work pretty evenly. Most of the work was done individually on personal laptop computers. The use the Internet to find out which tools to use and how to use them. For communication we relied heavily on “WhatsUp!” to publish topics which each were working on at the time, publishing progress, asking for advice and to agree on issues. We also had face to face meeting every week.

For storing and sharing the source codes amongst ourselves we used GitHub. Our repository can be found from: https://github.com/viltuul/moviepredictor

We used python 2.7 as programming language. This was mainly due to the fact that the open source IMDB library (IMDbPy - http://imdbpy.sourceforge.net/) that we used was available for python 2.7.

We also used machine learning library scikit-learn in our prediction algorithms, namely Linear Regression and Random Forest for predictions.

3 Mini-project topic – Movie Predictor

Our topic for the mini-project was Movie Predictor. Below you can find the original hand-in for the mini-project describing what the groups was about to do before any work had taken place.

> “Movie predictor. For given person (actor, director or producer name) the program downloads dataset from IMDB and shows his or her movie history. From the output one can see persons most favorable movie genre, successfulness of the person e.t.c. Then with this dataset the program predicts the next movie where this person is taking part. Prediction includes movie title, plot keywords, when the movie is coming out and the successfulness of the movie.”

But as you will find out by reading forward that some predictions are not 100% accurate. So wasn't ours. Here is a shortlist of the implemented functionalities:

- predict persons next 5 movies rating and budget, using 2 machine learning algorithm from data sets retrieved from IMDB
- download movie history with such related data sets that were available from IMDB for a chosen person and chosen datasets
- since downloaded data is stored locally in files and further wrangled to csv-format, person's movie history and dataset can be viewed
- keyword based text analysis for the top ranking movies for the person

Ideally our initial aim was provide a movie producer with information of a profitable movie based on given inputs given to our solution. Quite a niche but luxorious market we hoped. But as our understanding about the data and what we can do with it grew when going forward we came to a conclusion that this is mainly for amusement.

4 Path to Glory

Our high-level steps to get over the finish line was setting up the environment, data gathering, data wrangling and machine learning.

Caring for the environment

Setting-up a working environment for the development was an effort of it's own worth to mention. Python was chosen since everyone in the group was already familiar with it to some extend. Some of us had to familiarize oneself to python package and environment management.

One of the hurdles was to realize that our choice of IMDB API library, IMDbPy, was only implemented for python 2.7. This caused some wondering around the issue since the library really was obviously not getting installed on python 3.6. Also the documentation on IMDbPy was not clearly stating to which python version it was for.

Gathering the data

As mentioned already before we used IMDbPy library to source the data from IMDB. There is some documentation available for the library in http://imdbpy.sourceforge.net/support.html but we found the documentation to be not so clear and not user friendly. So it took quite a lot of our time to understand how to use the library and to figure out get data.

We used http-method from the IMDbPy library to source the data. With the library there is also a possibility to start to store the data locally on database and then use SQL for fetching the data. But since it was not possible to download all the IMDB data to a local store we opted to use the http-method to fetch the data we required.

With the library it is possible to do search on objects like movies, people, characters and companies based on search string e.g. movie name or person name. Then it is also possible to do more defined fetch with an id belonging to a single movie, person, character or company object. Along with the defined fetch one specify different data sets to be included in the result. The data set are different for different objects.

The hardest part for us was to understand what data there is and how it can be fetched. The use of data sets as explained above was not very clearly stated in the documentation of IMDbPy library. The library includes methods to discover which data sets are possibly available for an object. And by stating possibly we mean that clearly not all the objects have data in all data sets related to them. Also further details of which fields belong to a data set was unknown to us.

Together with discovering the use of data sets we needed to compare what the IMDB web pages for an object hold and do a full search on the same object to understand what fields and data can a certain data set hold. This is where we learned which data is in which data set (or at least that where the data that interests us is at) so we could only fetch those ones.

Another discovery was that the it took a lot of time to get requested data with the IMDbPy library from IMDB. The time it took was clearly depending on the number movies the chosen person had in different roles, since we chose to fetch for a person in several roles. Also the number and size of chose data sets to be included in a movie had it’s effect on the fetch time.

Our implementation of the data fetching methods suited to our needs are implemented in imdb_get_data.py. In the implementation we are utilizing search_person, get_person and get_movie methods from the IMDbPy library. And we only used specified data sets for Movie objects to be returned.

We use search_person method with person name as a search string. The method returns a list of Person objects the same or close to same name. We simply pick the first Person object from the returned list. From the chosen Person object we use PersonID attribute to fetch the filmography for that person in ‘producer’, ‘writer’, ‘actor’, ‘actress’ and in ‘director’ roles with get_person method. The filmography contains a list of Movie objects containing some narrow set of data included. Hence obtaining the filmography we used movieID from from each Movie object to fetch predetermined set of data sets for each movie, one movie at the time. Our weapons of choice were ‘main’, ‘business’, ‘vote details’, ‘keywords’, ‘taglines’, ‘trivia’ and ‘release dates’ data sets.

Wrangling with the data

After the desired data was fetched with the use of IMDpPy-library it needed to be stored and modified so that it would be easier to read and use for different kind of predictions. This was done in two parts. First part was done by csv_write.py which would select the most interesting and useful information fetched from the person and write it onto a csv file. The data fetched by IMDpPy comes in .json format but since the data was full of empty values we decided that it was better to write in .csv format so that we could actually get some clue what did we find from a person. Idea was that the filename was “firstname lastname.csv” and it contained the filmography of the person. One row in the file was one movie. After the .csv file was created it still needed some reconstruction and that’s where csv_data_parser.py comes in.

The csv_data_parser.py reads the raw_data file and creates a new .csv file where many columns are parsed so that they are easier to read and usable for machine learning and plotting. For instance runtimes columns in raw data had multiple runtime values. Apparently the movie runtime might’ve been different in different countries or directors cut. Also budget column (and many other columns) included words or sentences which needed to be parsed out since only the number value was useful here. Finally csv_data_parser.py created new columns for every possible movie genre. In example action and horror had own columns. These columns got value of 1 if the movie genre was same as the column. Otherwise it got value 0. These was later used in the machine learning functionalities. Even after many hours spent in creating of csv_write.py and csv_data_parser.py they include some bugs. This was mostly due to the fragmented data from imdb. Sadly at this point we were pretty much committed to the IMDbPy library and the project idea.

After the data was stored and parsed it was time to show and visualize it. Idea was to start off with simple movie tops and flops and worked iteratively to more complex data visualization methods as the deadline of the project was getting closer and closer. So first step was to show movies with highest and lowest budget and rating. Second step was to utilize plotly.py library and make an interactive plotting from the movies. Axis were rating and budget and also rating and release year of the movie. From these we calculated also the correlation if the actor / director created better or worse movies according to the budget or release year. Next step would’ve been to do some regression analysis for the movies but the time was up. ”If only we had one more week.”

Learning from the data

Original goal was to predict multiple attributes for person’s next movie (rating, budget, box office income, etc) , but because of issues with data mining and IMDB’s lack business data we ended up getting rating and budget predictions to work. However movie’s budget prediction aren’t as accurate as they could have been. This is mainly because of the lack of business data at IMDB site, general rule is: less known the movie is, bigger chance that the data is lacking. Solution to this issue could be achieved by data mining the business data from other sites than IMDB (for example boxofficemojo.com, but there just wasn’t enough time available to do that. Also having some sort of current popularity metric for a person would have helped with the prediction accuracy. This could be for example: amount of followers person’s professional Facebook page or Twitter account has.

The machine learning techniques that were used on predictions were Linear Regression and Random Forest Classifier. Those algorithms were mainly picked, because it was easy enough to understand the functionality behind them and to make our data fit. Also scientific articles written on topic indicated that the prediction accuracy of those machine learning algorithms is quite good when trying to predict movie’s critical and financial successfulness. We didn’t had the time to implement those algorithms ourselves so instead scikit-learn libraries were used. These algorithms were also the core functionality of the two predictor programs that were implemented during the project. Both of those programs can be found at the root of project’s github repository. Open source library called Pandas was used for handling the data in both those programs.

First program is called predictor.py and it should work in any environment with python 2.7 installed on. The program has 1 required argument which is a path to .csv datafile that should contain person’s parsed IMDB data. This file can be created by csv_data_parser.py mentioned in previous section. The program has also one optional argument, which is used when a user wants to predict ratings and budget for specific movie genre. In this case only data from specified genre will be used. For example if we want to predict Roger Corman’s next action movie and our Roger Coreman’s IMDB data is in location ‘parsed_data/Roger Corman.csv’ the program is started by typing following command: *python predictor.py “parsed_data/Roger Corman.csv” Action*. Output of the program will be predictions for person’s next five movies ratings and budgets computed by both linear regression and random forest classifier. The program’s output also includes Random Forest feature importance. Features which are used for predicting rating are: year the movie came out, number of votes the movie has at IMDB, movie’s length, movies budget, person's role in movie (director, actor etc), kind (movie, tv show etc). For budget predictions rating and data switched places. Original plan was to use way more attributes that this for the predictions, but there was problems with lacking data and difficulties to fit some of the attributes for the prediction models. Before algorithms can be run on the data, missing numerical values must be filled with means and categorical values must be turned into numbers. Data is splitted into training and test sets (training set was 80% and test set 80%). After this training set are used to train the models and the predictions are made by using the test set.

Second program is called predict_n_write.py and it is also implemented using python 2.7. This program also requires a parsed IMDB datafile to work, but the functionally is a bit different. Program’s goal is to fill the missing rating and budget data in reverse order (starting from the oldest data). It does it by using scikit-learn’s linear regression library. Other missing values than rating and budget and categorical values are dealt as they were dealt with in predictor program mentioned earlier. Only rows with non-missing rating and budget data are used to form training and test sets. Output of the program should be .csv file where every missing rating and budget is predicted one by one based on data that was already at IMDB and on previous predictions. Rating is predicted first because it has less values missing than budget for pretty much every person at IMDB.

We also tried to do some kind of text analysing. Basic idea was to recognize frequently used words for movies of person and associate those words with the rating class of the movies. This would have resulted in know which word are used to describe top rated movies for person. From those words one could recognize what kind of movie would be rated good (or any rate) for that person if a movie to be would containing those themes from the words. For this we tried to use the rating medians and keywords which were readily available in the data sets. The downside was the either the data or our implementation resulted in counting all the keywords as commonly used. But we basically ran out of time to analyze further what was the problem with our text analysis.

5 Self-assassment

Jussi-Pekka

The main thing that I learned with this mini-project was the eyeballing of the data. To try to understand that what this data really is that you are looking at and how to use it. Also to try to understand how the data set (in sense) evolves when changing the input parameters to retrieve it. And that it could really take a lot of time to understand it. What I hope I could have been more involved in was the actual “machine learning” part in our work and the visualizations of the data and results. For me plotting is still rather difficult. For throughout the mini-project I think that as group we worked pretty well together and we all put a decent amount of effort in. In retrospect the one thing that we should put more focus on was the 3-5 minute pitching. As what comes to the mini-project assignment in general I think it is fair to say that it came as a surprise and personally to me it was a bit inconvenient surprise. It would have been more fair if the mini-project assignment would have been there from the beginning. But I will not go more in it in here because this the mini-project report. But I am glad that it is now done and I am happy what we were able to finalize the mini-project decently as a team.

Ville

After Jussi-Pekka had learned (and taught us) how the IMDbPy works and created the first search methods I started off with the csv_writer.py and csv_data_parser.py. Csv_writer.py was pretty straightforward to write but testing it was pain in the ass because the slowness of the IMDbPy. Luckily I found an actress named Anne Sellors who had only one movie entry in imdb and downloading the info from her took only about 10 seconds. Csv_data_parser.py was then the tricky one. Raw data was filled with empty values and weird or inconsistent data so there was many hours and sleepless nights when tried to figure how to parse the data to desired form. Finally I was responsible of the data presentation and visualization. Since it was the last day before the deadline I couldn’t manage to create the visualization I wanted but luckily plotly was an quick and easy tool to get something for the eyes of the fellow students. All in all I would’ve needed couple of more days for the project to show more sense for the end user but still I was happy to get something out from the bits and pieces of movie data.

Andreas

I like to be always prepared, but for this mini-project I was not prepared at all. This should have been announced at the start of the group. As a 32 year old, I just can’t rearrange my schedule on such short notice and neither could other members of our group. So only option was to work remotely on different time schedules. Good software gets built as team that is working at same location, that is a fact. Of course there are exceptions, but these usually happen when all involved parties have long experience in the field. Data Science was and is completely new subject to me. Problem with newbies working long distance is that no one really knows what to do next and people don’t want to step on other people’s toes. My biggest regret was that my personal workflow, because of my other commitments (that I was aware and committed to well before this mini-project was announced) was very tail ended. First half I couldn’t be much use for the group, but I made up for it as best I could at the second half. My main focus of the project was the machine learning part and I learnt surprisingly lot about it in those couple of weeks. I was kind of forced to read those boring articles and documentation on subject. Another thing that improved quite a lot was my programming skills with Python and Pandas. Overall I'm quite pleased with the end result and performance of every group member, only thing that could have been better was the three minute long “presentation”.