HW2 : Progress report : Success Movie

Clément Bernard and Maxence Brugeres

December 19, 2019

Problem

What is our problem

It is hard for movie industry to know whether a movie will have success or not. Nevertheless, it seems that some companies have found a way to attract spectators. But there are still cases where high budget films are flopping and vice versa.
The first one is to determine what are the parameters that influence the most the popularity of a movie. We want to highlight how the gender, duration, cost or cast of a movie are important for its success.

What is there and the limitations

One way to know the success of a movie is the advertises made for it. The more the public sees and hears about the movie, the higher is the chance for them to go to see it. Nevertheless, the success isn't guaranteed. We can't rely only on that parameter.
Then, another way to judge the future success would be critical reviews or social media chatter. It will have an impact on the reputation of the movie and then on the number of viewers. But we can't rely only on these features too. There are too many cases where the critics aren't related to the success.
We can see that there are some parameters that come in mind when we are thinking and searching about movie success. Nonetheless, there are not really a magic tool that will say this specific movie will be appreciated. Our goal is to take into account all the data we can collect in order to give the best prediction for the success of a movie.

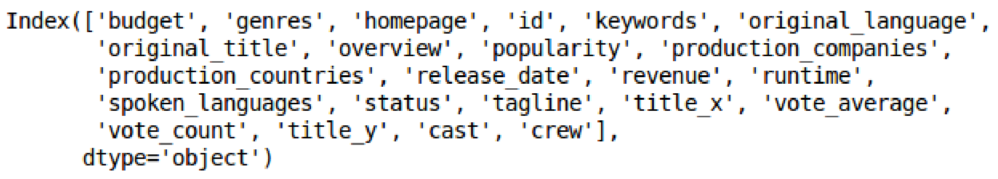
Preliminary results

Collecting data

We are using a dataset from Kaggle. This dataset comes from TMDb which is one of the most popular source for movie contents. It contains around 4800 different movies and 30 features.
These features are divided into 2 csv files. One for the general information about the movie (budget, gender, language, title, . . .) and the other one for more details (like the casting).
Our aim is to understand each feature in order to modify it (if necessary) to give as much information as it could. This is what we have done in the pre-processing part.
Then we have add another set of data, referencing all the nominees for an Oscar. This will lead to another column in our data set, referring to the popularity of the cast and the realisator / director.

Pre-processing

Here is the list of the features we have :

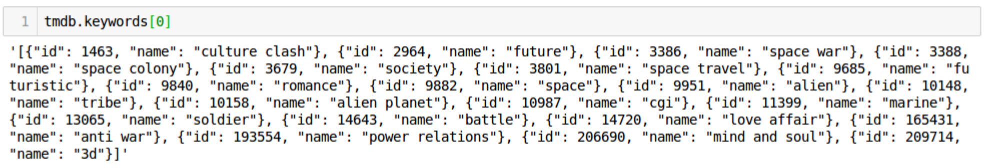


```
Index(['budget', 'genres', 'homepage', 'id', 'keywords', 'original_language',
       'original_title', 'overview', 'popularity', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title_x', 'vote_average',
       'vote_count', 'title_y', 'cast', 'crew'],
      dtype='object')
```

Figure 1: Columns of the dataset

First, we will consider two targets in this project standing for "the success of a film". The first one is the revenue of the film. The second one is the ratio, defined by $\frac{revenue}{budget}$. The third one is the vote attributed by the users of TMDB. Yet, here we come across our first problem, how can we rely on the vote since some films have received 100 votes and some others 10000. Then we will not consider the vote but the grade defined by vote $-\frac{10}{\sqrt{vote\_count}}$. The meaning of this formula is explained in the appendix.

One of our biggest problem was to deal with the JSON format of some of the features.
Here is an example for the keywords :



```
tmdb.keywords[0]
```

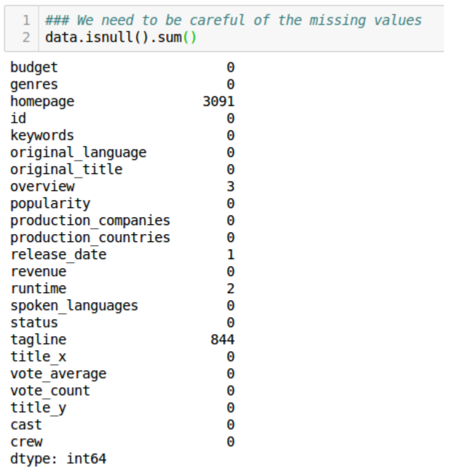
```
'[{"id": 1463, "name": "culture clash"}, {"id": 2964, "name": "future"}, {"id": 3386, "name": "space war"}, {"id": 3388,
"name": "space colony"}, {"id": 3679, "name": "society"}, {"id": 3801, "name": "space travel"}, {"id": 9685, "name": "fu
turistic"}, {"id": 9840, "name": "romance"}, {"id": 9882, "name": "space"}, {"id": 9951, "name": "alien"}, {"id": 10148,
"name": "tribe"}, {"id": 10158, "name": "alien planet"}, {"id": 10987, "name": "cgi"}, {"id": 11399, "name": "marine"},
{"id": 13065, "name": "soldier"}, {"id": 14643, "name": "battle"}, {"id": 14720, "name": "love affair"}, {"id": 165431,
"name": "anti war"}, {"id": 193554, "name": "power relations"}, {"id": 206690, "name": "mind and soul"}, {"id": 209714,
"name": "3d"}]'
```

Figure 2: Example of a json format

There are 5 features coded in this format : "*genres*","*keywords*","*production_companies*", "*production_countries*", "*cast*" and "*crew*". What we have done so far is to convert this json into a dictionary in order to get the values "*name*". More details will appear in the following description of our processing.

Deleting missing values or non relevant values

We've looked to the missing values. Here is the result :



```
### We need to be careful of the missing values
data.isnull().sum()
```

```
budget                     0
genres                     0
homepage                3091
id                         0
keywords                   0
original_language          0
original_title             0
overview                   3
popularity                 0
production_companies       0
production_countries       0
release_date               1
revenue                    0
runtime                    2
spoken_languages           0
status                     0
tagline                  844
title_x                    0
vote_average               0
vote_count                 0
title_y                    0
cast                       0
crew                       0
dtype: int64
```

Figure 3: Number of NaN instances

The two features that have important missing values are homepage and tagline. These features are strings and correspond to the website of the movie and a tagline that summarizes the movie. In this effect, we don't take it into account because we don't want to focus immediately on strings.
The vectorization of strings is a purpose for the next deadline (if we have time). Then, we have dropped the missing values for the other features.
Furthermore, we have deleted instances where the budget and revenue are null and where the runtime (duration of the movie) is null. Indeed, it corresponds to situation where we are missing important features for us.

Categorize our features

Budget For the budget, we assume this feature very relevant. In this case, we keep it like it is. Then, we can see a correlation between budget and revenue :

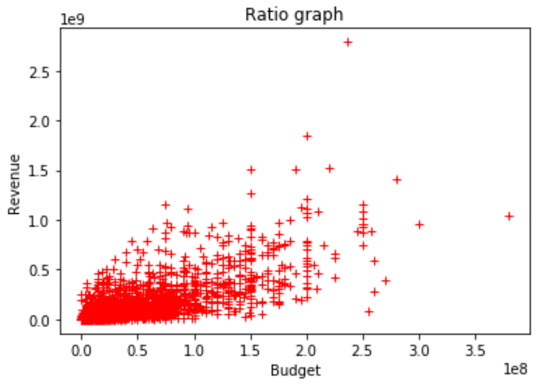


Figure 4: Ratio depending on the budget

It makes sense : the more budget, the more successful the movie tends to be. In this case we added a new feature, called “ratio” that is the revenue divided by the budget.

Genres This feature is coded as a json. In this effect, we created a function that iteratively convert the instance into a dictionary and then we collected the data. We counted for each genre the occurences in our dataset.
Here is the result :

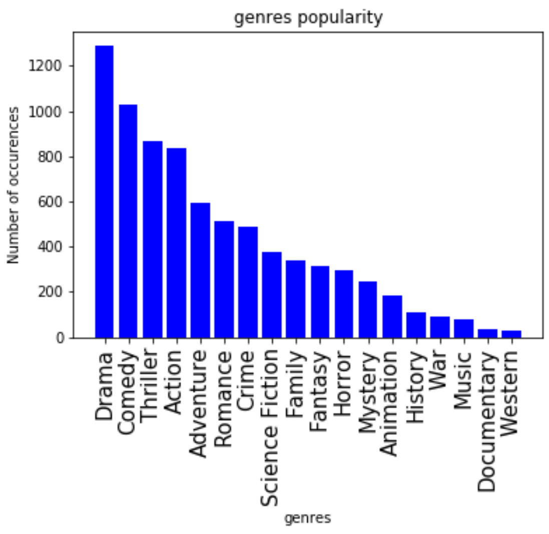


Figure 5: Occurences of genres

Then we used a 1-of-K encoding for the genres. We expect a different behavior for our classifier depending on the genre of the movie.
In this effect, we will for the moment keep the 1-of-K encoding for the gender, but we aim to create a different classifier for each genre.

Homepage This is a feature that contains the website of the movie. It doesn't seem to be relevant. In this effect, we dropped it from our database.

Id Just the id for the database. It has no impact on the model. We dropped it.

Keywords Here is another feature coded with json. To treat well, we wasn't able to categorize directly the instances. Each instance has few keywords associated.
In this effect, we decided to give a grade to each keyword. For each keyword, we saved the grade (new feature explained above, that corresponds to the label) and then compute the average among all the movies where the keyword belongs.
Here are the 25 best graded keywords :

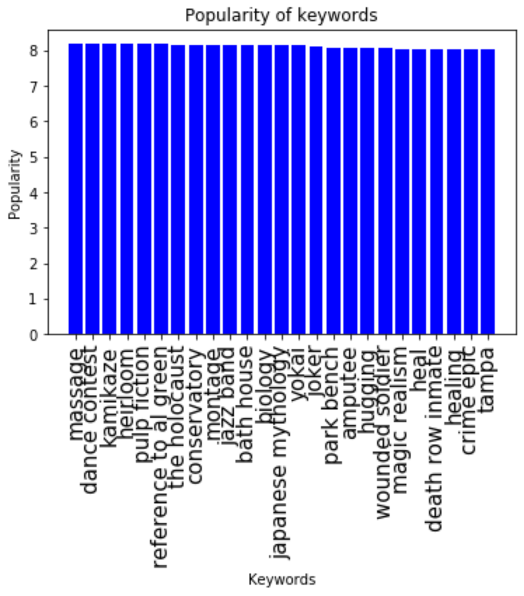


Figure 6: Grade of the keywords

Original_language It gives the language used for the recording. It has values like 'en' (english) or 'fr' (french). In this effect, we plotted the most used languages in the following chart :

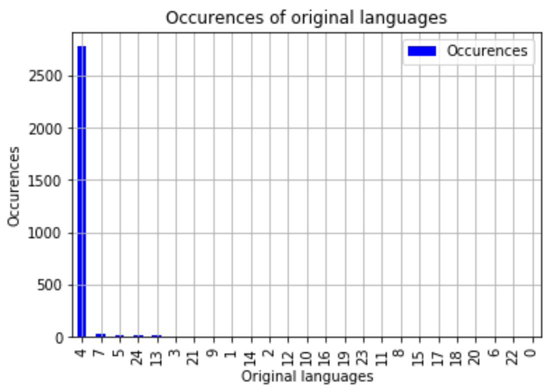


Figure 7: Occurences of the original language

We decided to keep the five first languages and make the rest into a \garbage" column. Then, we code it with the 1-of-K encoding.

Original_title / title_x / title_y Interesting to identify the film but won't be taken as a feature of our model.

Overview For now we drop it, it is a quick recap of the film so requires string analysis to be taken into account.

Popularity We decided to drop this column since this is a grading made by TMDB based on many criterias (vote per day, number of views, ...) which is not made for our study.

Production_companies This is coded as a json file. In this effect, we counted the occurrences of each production companies. Here is the result (we only plot the 20 first) :

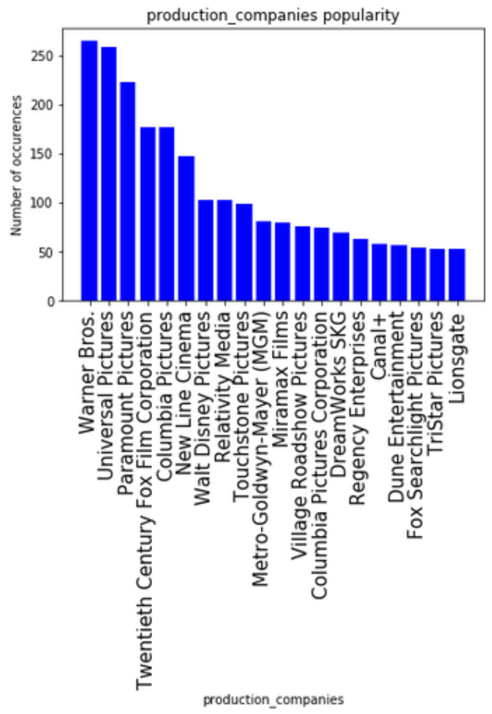


Figure 8: Occurences of production companies

We decided to split it into 3 categories : the big companies (where the occurrences are superior to 85), the middle companies (occurrences are between 20 and 85) and the independent (occurrences are inferior to 20).

Production_countries As the previous one, this is a json format. As before, we plotted the occurences :

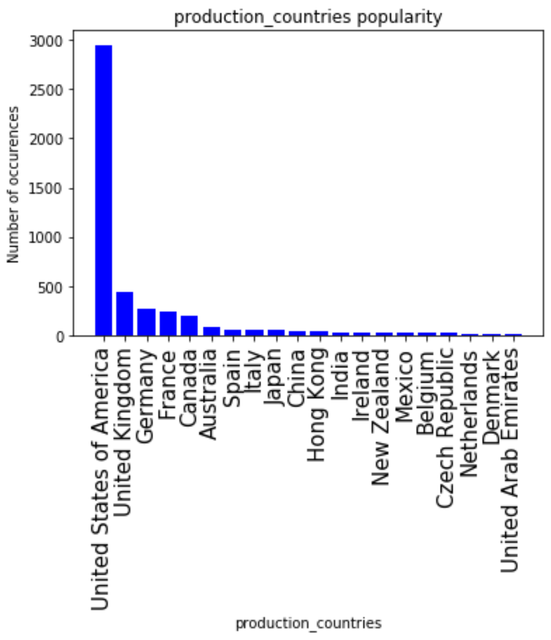


Figure 9: Occurences of production countries

We kept the 5 first ones and make the others as a unique feature. Then, we do 1-of-K encoding.

Release_date We dropped the movies that appeared before 1985 (around 300 instances) because the standard of movies before this year seems to us obsolete compared to now.
We then convert it as a time format (in panda).
Then, we created a year feature and 1-of-K encoding for the months.

Revenue We keep this value as it is. It corresponds to one of our output value.

Runtime We split the duration of the movie into 4 categories. The first one is the movies which last less than 60 minutes. The second one is between 60 minutes and 90 minutes. The third one is between 90 minutes and 120 minutes and the last one is after 120 minutes.

Spoken_language Describes all the languages spoken in the film, even if they are spoken only two minutes. Thus we consider this as irrelevant and dropped the column.

Status Describes whether the film is released or still in post-production. Yet, after dropping some rows for other criterias named in this report, all the status was "released", hence we dropped the column.

Tagline Tagline is small sentence whose purpose is to describe the film. As mentioned before, we don't take the tagline into account yet since it is string.

Vote_count The number of voters having graded a film.

Vote_average / Grade It is the mean of the vote given to a film by the users of TMDB. As explained in the appendix and the introduction of the part pre-processing, we won't take this indicator as its accuracy depends on the number of voters.
We have turned this into the "*grade*" column which is vote_average $-\frac{10}{\sqrt{vote\_count}}$.
Here is the result of the differences between grade and vote average :

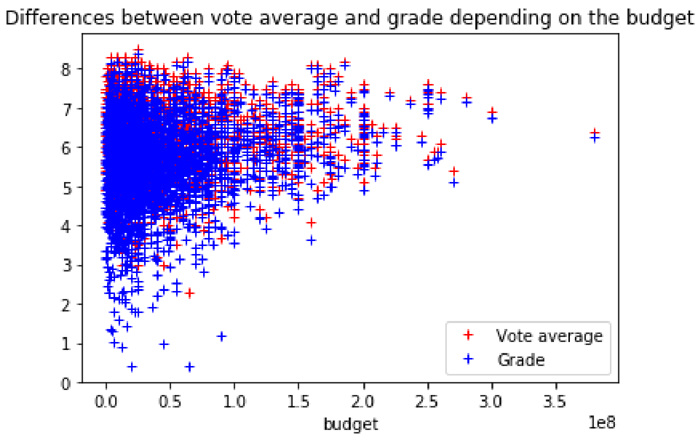


Figure 10: Plot of vote count and grade

Cast / Crew For each crew and cast we check how many people have already be nominated for an Oscar. Then the note we have attributed for crew and cast quality is $100 \times \sum_{i=0}^{n} (\frac{1}{2})^i$ with n the number of people among the cast/crew that have been nominated for an Oscar (or 0 if n = 0). Thus this grade is between 0 and 100. We then has dropped the cast and crew columns and replace them by the grade we had just found.
Nevertheless, the code is not yet implemented. We make the average of the grade instead for the moment.

What remains

Then the pre-processing is done, we are working on the conception of our classifier.

Appendix

Let's consider the chebyshev's inequality :

$$P(|X–E(X)| > \epsilon) < \frac{V(X)}{n\epsilon^2}$$

Let's apply this to Y = $\frac{1}{vote\_count}\sum_i^{vote\_count} X_i$ = vote_average where X_i is the vote of 1 user. We consider that the vote of one user is a random variable with expectation the real grade of a film.
All the X_i are i.i.d. Thus what we want to find is $\bar{X}$, the expectation of X (and Y). We get :

$$P(|\frac{1}{vote\_count}\sum_i^{vote\_count} X_i–E(X)| > \epsilon) < \frac{V(X)}{n\epsilon^2}$$

Let's find ϵ such as $P(|\frac{1}{vote\_count}\sum_i^{vote\_count} X_i–E(X)| > \epsilon)$ is smaller than 0.01.
We assume that the variance of X is 1 (sensible since X is between 0 and 10):

$$\frac{1}{n\epsilon^2} = 0.01 \Rightarrow \epsilon = \frac{10}{\sqrt{n}}$$

Then we have :

$$P(\frac{1}{vote\_count}\sum_i^{vote\_count} X_i–\frac{10}{\sqrt{n}} < E(X) < P(\frac{1}{vote\_count}\sum_i^{vote\_count} X_i + \frac{10}{\sqrt{n}}) > 0.99$$

We finally consider the lower bound, which implies that we are sure at more than 99% that the real grade of the film is bigger than $\frac{1}{vote\_count}\sum_i^{vote\_count} X_i–\frac{10}{\sqrt{n}}$ which is also vote_average - $\frac{10}{\sqrt{n}}$.
We name this new value "*grade*".