A General Overview on Movie Data with a Look at Oscar Nominees, the Building of a Recommendation System and an Analysis on the Relationship Between Genre and Plot

Sergio Gentilini
Institutional ID: 481089
Email:
sergio.gentilini01@universitadipavia.it

Gianluca Caronte
Institutional ID: 479637
Email:
gianluca.caronte01@universitadipavia.it

Michele Inchingolo
Institutional ID: 482748
Email:
michele.inchingolo01@universitadipavia.it

Bill Mono
Institutional ID: 479379
Email:
bill.mono01@universitadipavia.it

I. Introduction

Movies are an important piece of culture in our society. The best films are remembered for generations and can make an unknown actor's career, while bad movies can have the opposite effect. The revenue aspect is also crucial, since production companies will reward an excellent film with sequels and a well-performing actor with more castings. Our analysis focuses exactly on this: we examine a movie dataset to learn about patterns and what makes a film a "good" one.

The movies have been analysed through four datasets: the most comprehensive one includes metadata such as the title, the genre, and the average vote given by reviewers to about 45000 movies; the second dataset is a reduced version of the former, with only a third of the observations, and provides the movie plots; The third dataset contains the ratings manually inserted by users in the *MovieLens* platform; the last dataset has been collected from the *Academy Awards Database* and provides a list of all the Oscar Award winners from 1927 till 2020 with relevant metadata.

II. Data

A. Dataset A - Movie metadata

The main dataset. It includes information about movies, with the earliest being released in 1874/12/09 and the latest planned for 2020/12/16, which are collected in the following format:

- The movie ID;
- The movie budget;
- The genres the movie is categorised as;
- The original title;
- The international title;
- An overview, which briefly describes the movie;
- The popularity score, which takes into account the page views on the IMDB website;
- A list of the production companies involved in making the movie;
- The date of release;
- The revenue;
- The runtime in minutes;
- The status of the movie, such as "released", "post-production" or "planned";
- A tagline;
- The average of the votes given by reviewers;
- The number of votes that have been given to the movie by reviewers (on *IMDB*).

B. Dataset B - Movie Plot Synopsis with Tags (MPST)

The MPST dataset contains information about the movie plots. The main fields are:

- The title of the movie;
- The plot of the movie;
- A list of genres and tags to which the film belongs;
- A *split* field that indicates whether this observation will be part of the training set or the test set.

C. Dataset C - MovieLens ratings

This dataset contains a collection of ratings collected from the *MovieLens* website. The ratings are composed by:

- A user ID;
- The movie ID (related to *MovieLens*);
- A rating (from 0 to 5);
- The timestamp.

D. Dataset D - The Oscar Award 1927-2020

This dataset contains information about all the nominees and the winners of the Oscar Award from 1927 to 2020. The main fields are:

- The year of release of the film;
- The year of ceremony;
- The name of the film;

Data Science and Big Data Analytics Course – 2019/2020. Report of group MIC. Submitted on 11/01/2021
Supervisor: Prof. Antonino Nocera

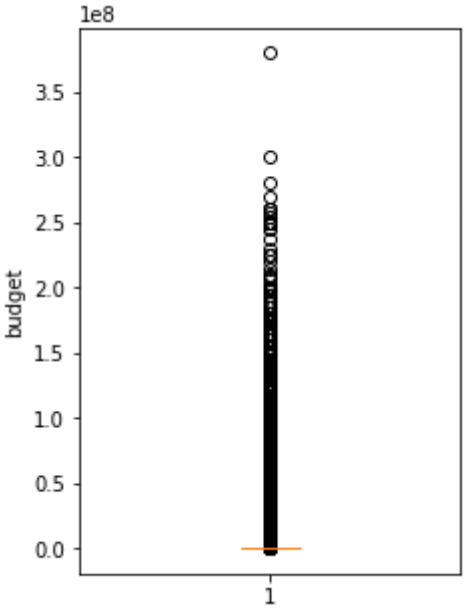
- The name of the person, or production team, who has been nominated;
- A boolean *winner* field which indicates whether the person won the Oscar or not.

III. Exploratory Analysis

A look at movie metadata

The main dataset on which the hypotheses have been verified is dataset A. It includes about 45000 different observations that are mostly clean. Missing data generally relates to taglines (more than half are missing) and production companies (the companies for 11000 movies are unlisted). Some categories, such as *budget*, replace missing values with a "0", which should be taken into account.

By analysing the relevant numerical features through histograms and boxplots, it's easy to see that only one of them follows a somewhat normal distribution: *vote_average*, whose mean sits between 6 and 7. The boxplots show that many of them are mostly composed of outliers; in these cases, the box part is centred around 0 and barely visible because of the default value for missing data (the *budget* problem explained above).



The budget *boxplot is "squashed" because of missing data.*

To increase readability, the first data transformation procedure focussed on removing the outliers. The 2.5% quantiles, on both sides, have been taken off for all the numerical features: the improvement is clearly visible on the histograms. Among them, *runtime* assumes a normal distribution after removing niche movies with an exceedingly-long duration (sometimes even above 20 hours). At times, however, outlier removal makes little sense, such as in the case of *vote_average*.

After these observations, it was decided instead to remove the outliers from *budget*, *popularity*, *revenue*, *runtime* and *vote_count*, then all the values equal to 0 in *budget* and *revenue* were substituted with the mean. This leads to a smaller, but more reliable, dataset: around 41000 observations.

Correlation was checked on this transformed dataset. It shows that *revenue* is positively-correlated with *budget*, *popularity* and *vote_count*, possibly meaning that highly-grossing movies require a high budget, but lead to a large number of reviews and page views. The average vote doesn't seem to be correlated with anything, which means that, with the available data, it may be hard to guess whether a movie will be well-received or not.

As a last pre-processing measure, the data have been standardised.

Actor analysis

Instead of concentrating on the whole dataset, this analysis focussed on a single actor. The aim was to provide a visual representation of an actor's career by plotting their movies chronologically against certain quality measures: *vote_average*, *popularity* and *revenue*. The plot might give insight about whether an actor starts starring in high-budget movies once their career takes off, whether movie popularity depends on the involved actors and so on.

Given the name, all the movies the actor has starred in were extracted and ordered by opening date. The *revenue* null values were substituted with the median value over the rest of the movies the actor had participated in (rather than with the mean over all the movies in the dataset, as explained earlier) for a more reliable interpretation of the plot.

The correlation analysis is largely inconclusive, as it depends on the chosen actor. However, if the whole dataset is considered instead, there is visible correlation (0.5) between *revenue* and *popularity*, which may reasonably indicate that fame leads to bigger earnings.

Next, the movies were plotted against their *vote_average*, *popularity* and *revenue* both separately and, for ease of interpretation, in a single plot (post standardisation). This visualisation makes it possible to observe the trends and events of an actor's career, as well as the relationship between the three target variables.

In general, movies with high *revenue* have a high *popularity* score as well, which matches the findings obtained via correlation. However, it's hard to make solid observations on a career and its evolution in time, since the findings may be different depending on the chosen actor. Usually, it's impossible to see any sort of trend.

Production companies analysis

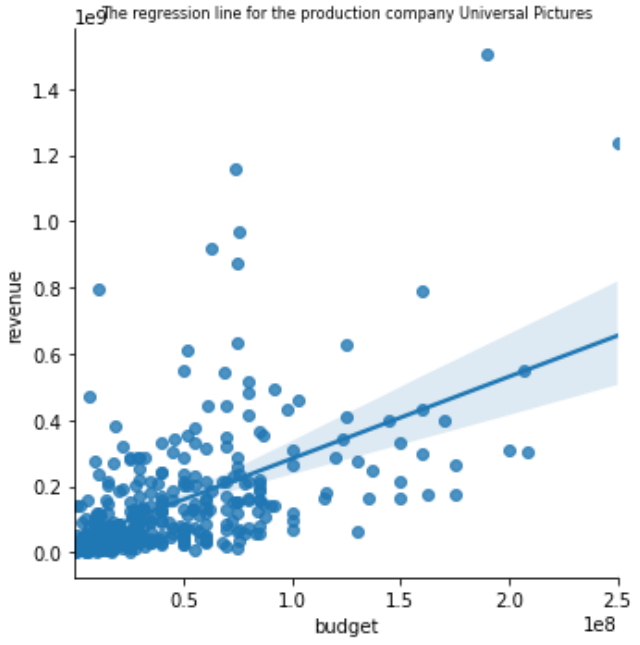
The goal of this analysis was to see if, for the different production companies, there is a relation between the budget spent on movies and the relative income.

We selected the production companies that had made the most films (*Warner Bros* and *Universal Pictures*), then we measured the co-relationship between *budget* and *revenue*; however, this highly depends on the production company. In the case of *Warner Bros*, for example, it was higher than 0.7, while for *Universal Pictures* it stood at 0.55.

To understand if there is a linear relationship, we created a model through linear regression, putting *budget* as a feature. From the graphs we obtained what seems to be a linear relationship, even if, for all the production companies examined, the number of films was not enough: in particular, more high-budget movies would be required for a

more precise evaluation. The R^2 value for *Warner Bros* was around 0.5 in training and 0.4 in test, which is significant even if not very high; a worse performance was found in the case of *Universal Pictures*, where, as we had seen before, the correlation between budget and revenue was lower.

To improve the model, we included the number of votes obtained for each film as a feature in addition to the budget, since it had a high correlation with revenue for both production companies. In doing so we obtained an R^2 of 0.7 in the training for Warner Bros and 0.62 in the test, so a much better result. For *Universal Pictures*, however, even if there was an increase in the value of R^2, it was not of much significance.



Regression line for Universal Pictures.

Regression line for Warner Bros.

Analysis on plots and movie genres/tags

This analysis was performed on the *MPST* dataset. Since both tags and plots are available, the idea of examining the words for each genre is almost consequential. In particular, this analysis consists in isolating the words that are the most frequent in well-received movies, divided by genre. The results might give insight into potential hypotheses and further analyses.

First of all, the genres were extracted and plotted in decreasing order of occurrence: *murder* and *violence* are the most frequent (5800 and 4400 occurrences, respectively), while all the others appear less than 3000 times. Despite there being 71 tags, 87% of them occur less than 1000 times each: indeed, many are too specific (such as *blaxploitation* and *philosophical*) or vague (like *boring*, *cruelty* or *sentimental*) and end up being underutilised.

After extracting the average votes given to the movies from *dataset A*, the *MPST* dataset was transformed to only include "good" films, which in this case includes those with an average vote equal to or greater than 7.5/10. The words used in the movie plots were then counted and separated based on the genre/tag, after dealing with the *stopwords* in order to exclude common terms. The 25 most-occurring words were then plotted in decreasing order by genre.

From the plots, it's evident that *murder*, *violence*, *flashback* and *revenge* are tags that define very similar movies, as the most-frequent words are largely the same: they include *police*, *home* and *father*. Many critically-acclaimed action movies fall into these categories and have a plot that can be broadly summarised by those words, so it's not a surprise.

Romantic deals with *love*, *man* and *life*; these are very generic terms that might apply to many romantic films, whether they are popular or not, so this might perhaps imply that you don't need a revolutionary plot to tell a convincing love story.

The *comedy* tag is interesting in that the fourth most-occurring word is *marty*. It probably relates to British actor Marty Feldman, one of the most influential comedians of the 60's and 70's; if this is the case, then this would mean that his presence in comedy movies is sure to lead to high votes by reviewers.

Most of the less-frequent tags only cover a niche. This is a natural conclusion when looking at the plotted words: *suspenseful*, *neo noir*, *insanity*, *mystery, dark* and *realism* talk about *batman*; *alternate history* concerns the *Back to the Future* movies, as evinced by the words *doc* and *delorean*; and so on. Many tags have been underutilised, as can be seen from the most-frequent word for *western* films, which is *django* (an obvious reference to the Tarantino movie), with no references to, for instance, *spaghetti-western* movies. Furthermore many tags include a copious amount of proper nouns, which could either mean that certain movies belonging to the same series (with the same characters or actors) are all evaluated positively, or, more probably, that the tag was applied to a very restricted number of movies.

In order to have a more intuitive visual approach, the most-occurring words by genre and tag were presented through a *word cloud* representation.

Word cloud representation of murder.

By contrast, the specificity of realism.

After examining the dataset thoroughly and pondering on the results, the following hypotheses were chosen:

A. *Hypothesis A: "Movies belonging to the 'historical' genre have their mean vote influenced by the costume designer(s)."*

There should be a visible relationship between the mean vote of historical movies and the costumers that worked in them. Given a costume designer, the movies they have worked in shouldn't be valued below a certain threshold.

B. *Hypothesis B: "There is a relationship between the genre of a movie and its plot such that it should be possible to predict the former from the latter."*

Movies belonging to a certain genre use similar plot devices and situations, so it should be possible to predict the genre from the plot. The exploratory analysis gave valid results for certain genres, making this a promising hypothesis.

C. *Hypothesis C: "Users that have shown similar tastes in the past will show similar tastes in the future."*

A user keeps showing the same tastes: if user A and B share similar review ratings on different movies but they didn't rate all the same movies, then an unknown movie rating of user A should be similar to the one given by B (and the other way around).

D. *Hypothesis D: "An actor who has fulfilled more roles has participated in more films with high marks."*

Through this hypothesis, we want to see if there is a relationship between the number of roles a person has played in their entire career, such as *actor* and *director*, and the ability to participate in more films that have obtained high marks from reviewers, hence having a better career.

E. *Hypothesis E: "The best movies have the best actors."*

This hypothesis is quite self explanatory, here the difficulty lies in the definition of the "best" metric for both movies and actors, since this adjective can be very ambiguous and subjective.

IV. Methodology

A. *Metric A: "Average votes."*

The average of all the votes given by reviewers to a specific movie.

B. *Metric B: "Model accuracy."*

Measures such as accuracy and precision were used to quantify the goodness of fit for hypothesis *B*.

C. *Metric C: "Users ratings."*

To verify hypothesis *C* we have divided the ratings randomly in two sets that represent the previous experiences of the users and the future ones.
Statistical metrics of goodness RMSE and R^2 were used for evaluation.

D. *Metric D: "Ratio between the number of actors with a good career and the total number of actors."*

To verify hypothesis *D* we divided people into groups based on the number of jobs performed and, for each of them, we calculated how many people have had a good career compared to the total number of the people belonging to the group.

E. *Metric E: "Average votes of films in which an actor participated."*

This metric has been defined in order to demonstrate hypothesis *E* and, in particular, has been used to define an objective way of measuring the "skill" of an actor. It was calculated by averaging the marks of all the movies in which an actor has participated.

1) Metric E.1: "Only movies released before the observed one."

By considering an observation composed of a film with its vote and its cast, we found out that there was a strong correlation between the vote of the movie and the skill vote of its cast, calculated by averaging the skill vote of each actor. So, in order to decorrelate these two measures, we decided to take into account only the movies released before the currently-observed film.

F. *Metric F: "At least one actor who has won at least one Oscar."*

This metric has been used to demonstrate hypothesis *E* and it was obtained by applying a flag to a movie indicating the presence in the cast of at least one actor that won at least one Oscar in his whole career. Hence, we looked for how this separation in the movie dataset affects the distribution of the votes.

V. Analysis Design

A. *Database*

To solve scalability issues, we decided to use MongoDB, a NoSQL database, as our data storage tool. This is because MongoDB has got numerous advantages over traditional relational databases in a big data context. First of all, it can be scaled horizontally, which is an invaluable characteristic when dealing with large amounts of data. Another advantage is granted by the organisation of data in schemaless documents, which makes it able to answer queries much more quickly than relational databases.

MongoDB brings a few issues for people coming from relational databases: in fact, the well-known SQL can't be used to build queries; they must be built through a JSON-like structure that defines the series of operations to be applied to the data.

Unlike RDBMSs, the execution of queries, both to interrogate and to update the database, are not blocking for other users: this means that even a query with a very long time of execution will not influence other operations on the same collections.

B. *Dimensioning the Problem*

When dealing with big data, the architecture should always reflect the problems at hand. In our analyses we examined datasets concerning movies, actors and Oscar winners/nominees; the datasets are provided beforehand and do not require any kind of real-time streaming, so the system architecture needs to reflect this situation.

Dealing with big data means dealing with high-volume information, which can be dealt with efficiently by configuring a multiple-nodes architecture on *OpenStack*. The idea is to first define the problems at hand, then assign nodes to each analysis or set of operations based on how computationally-intensive it is.

The analysis tasks are generally quick and easy to perform, so there is no need for a large number of nodes in most cases:

- The preliminary exploratory analysis should only be performed once the data are provided and every time the dataset is updated. Assuming monthly or yearly updates, the number of nodes dedicated to this task can be very small.
- The analysis of an actor's career doesn't require much computing power even when dealing with world-famous actors, however it is likely to be executed on several individuals, thus more often. Even in this case, a small number of nodes is enough.
- Analysing the plots and movie genres can be more resource-intensive than the previous tasks depending on the number of genres, the visualisation techniques (plots, word clouds…) and, of course, the amount of data. Since it's an essential part of the project and it can be built upon with further explorations, elaborations and studies, a larger number of nodes should be dedicated to this exploratory analysis.

Proving (or disproving) the hypotheses, however, often requires more computing power. Machine learning techniques, such as logistic regression, are especially demanding:

- Hypothesis *A* is perhaps the quickest of them all to test: it only involves a small subset of the data and the computations are not particularly complex. Still, having a sizeable number of nodes could be convenient if further data on historical movies were to be provided.
- Hypothesis *B* undoubtedly requires considerable computing power. The operations involved, such as word count, are very resource-intensive when carried out on the large dataset; moreover, there are multiple classification tasks to be performed for each sub-case or variation that might come up. For a fine tuning of all the possible parameters, the validation steps could end up being executed tens or hundreds of times. Depending on the computing power of each machine, several nodes may need to be allocated to test this hypothesis thoroughly.
- Hypothesis *C* is very labour-intensive and demanding; not only should the task itself be computed in a reasonable amount of time, it also needs to be executed frequently, since new movie reviews tend to get written every day. As such, it should be allocated a large number of nodes.
- Hypothesis *D*, much like hypothesis *A*, only involves a small subset of the data and is quick to execute on its own. As such, even a single node could be enough.
- Hypothesis *E* is relatively quick to execute up until *metric E.1* needs to be calculated. For this reason, adding a few nodes can be beneficial and speed up the process.

Since there is no crucial need for real-time processing and strict response times, all the operations could be performed on a cluster of everyday computers through Hadoop: the inexpensive hardware is an unarguable upside, and the high computing time has been said not to be an issue. Spark can be used as well when the operations to be performed are too complex to be translated into Hadoop *MapReduce* jobs, such as machine learning tasks. The upside coming from the quicker computing time (given enough RAM) may not be strictly necessary, but is a welcome addition nonetheless.

C. *Hadoop/Hive*

We decided to use Hadoop to perform a MapReduce join job, as we had a collection, *Oscar Award*, that wasn't integrated with the other collections present in our dataset.

The join was implemented at the *reduce* side of the MapReduce task by exploiting the *sort* phase, during which all the intermediate key-value pairs produced by the *map* side are reordered and grouped by same key; the join was automatically performed by using the intrinsic properties of

MapReduce. The join key was created during the *map* phase according to the most suitable information available in the joined collections.

Since our data was stored on a Mongo database and a Hadoop MapReduce task must be executed on its filesystem, it became necessary to use a specific library that allowed us to connect these two environments: MongoDB connector for Hadoop[1].

Oscar Awards lacks unique identifiers that might map the Oscar data to the documents present in the other collections. Each Oscar nomination is mainly characterised by a person, or organization, and a film name, so the join operation had to add information from documents concerning movies and people: two joins were performed using two different sets of information, one containing only the *Metadata* collection and the other one containing the *People*, *Cast*, *Crew* and *Production companies* collections.

1. Adding movie identifiers:
    - *Map phase*: data from MongoDB have been properly formatted and then used to create the join key, using the movie name and its release year. The output consists of a key-value pair , where the key is the join key and the value is a dictionary with all the other info contained in the document.
    - *Reduce phase*: in this phase, a projection on the information that is relevant to our ends was performed. The movie identifier was added to the Oscar documents and all the remaining information was discarded. The output is the same Oscar document as before with the addition of movie IDs.
2. Adding the person, or company, identifier:
    - *Map phase*: just like before, the data have been gathered from MongoDB and properly formatted. This time the join key was created by using the name of the nominated person, or company, and the movie identifier. The use of a natural language processing library, *Spacy*[2], was also invaluable: it was used to extrapolate the name of the candidates in the corresponding field of the Oscar collection, since the provided data wasn't always well-formatted and included multiple names as well, along with other less-useful information.
      The output is a key-value pair structured like in the previous join task.
    - *Reduce phase*: as before, uninteresting data were discarded and the identifier of the nominated person or company was added.

The final output consists in a collection of Oscar documents with the same structure as the original dataset, but with the addition of movie and person/company identifiers.

D. Spark

Spark is a unified analytics engine for large-scale data processing[3]. It can work on top of Hadoop and other computing frameworks and can access different sources of distributed data, like the HDFS (that can be connected to other sources, such as *MongoDB* in our case*)*. On top of Spark there are different libraries that allow for interrogating, computing or even training machine learning models on top of the data.

The main advantage of using Spark is that an interrogation, model training or other request can be designed to work on a little amount of data and be transparently used to process a huge amount, with the only constraints being based on the usage of the Spark interface. This can be accomplished by installing the Spark engine on a cluster of machines.

The Spark SQL library is designed to be simple to use: it can even allow for the interrogation of data by using SQL, which will be translated into a scalable combination of MapReduce jobs.

In our project it can be particularly convenient when working on the ALS algorithm for matrix factorisation, as there is supposed to be a large quantity of ratings in continuous growth that requires recurring processing in order to supply a valid model.

We also used it to pre-process plots and build classification models to predict the movie genres. In this use case, unlike in the previous situation, we do not expect a big growth of data, but they are still of considerable dimensions; being already on Spark, the preparation will save time once a hybrid model needs to be built.

A hybrid model is built by combining different models that return final predictions. It could consist in a simple weighted sum of the predictions of the different models. In this case, the weights could be estimated through linear regression, or the pre-filtering (or weighting) process can be based on user personal information (like age or sex) and movie charateristics.
i.e. We can exclude, or give a penalty to, horror *movies if the target is a nine-years-old girl.*

VI. Analysis description and Results

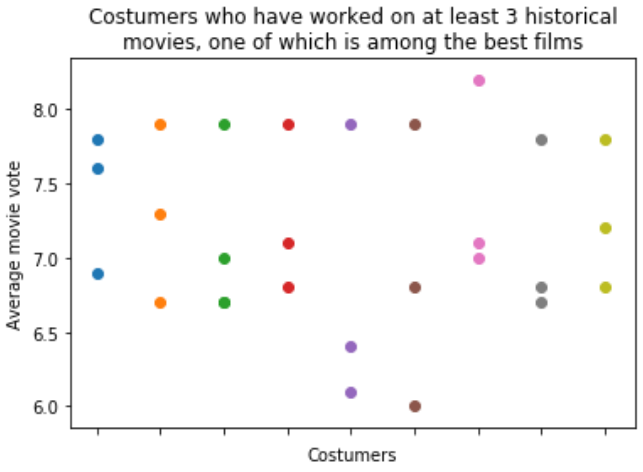
Historical movies and their relationship with costumers
Since data about movie crews and genres are available, some interesting hypotheses can be devised to predict the popular appreciation of a movie. Movies, in general, may be considered better or worse depending on several factors, such as the cast or the plot, but historical movies (as in, belonging to the *historical* genre) intrinsically depend on costumes more deeply than other kinds of films. Thus, the composition of the costume crew could be a good asset when predicting the average vote of a historical movie.

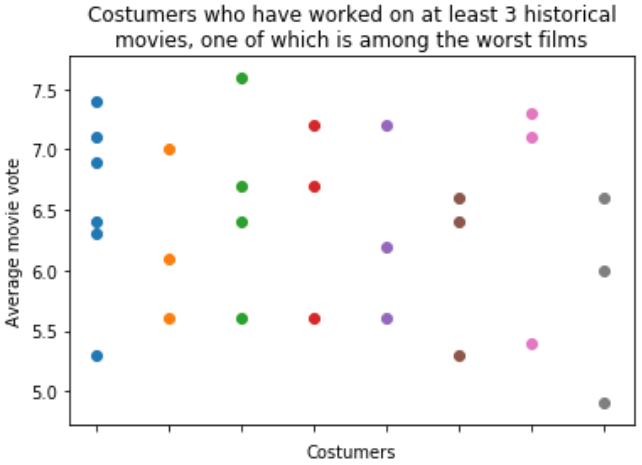
The *movie metadata* dataset was filtered to only provide historical movies for which the crew composition was recorded and with a number of written reviews equal to or greater than 10. This returns around 800 observations; a rather small dataset, but still usable for an analysis of this kind. Given the size, a "good" movie was defined as having a vote of 7 or more, while a "bad" movie possesses a vote below 6; the values in the middle were simply not assigned a label.

The first analysis was performed on the most prolific costumers: the costume designers were sorted on the basis of the number of historical movies they had worked in and the first 10 were compared with the average votes given to their movies. The resulting graphs showed no relationship between costumer and votes: all of the considered crew members had worked on both "good" and "bad" movies and no one kept strictly over a visible threshold that could define a clear separation between "good" and "bad" films. This revelation wasn't very surprising, since a costumer who has worked on several movies isn't necessarily "good".

Therefore, it was decided to take a look at the 20 best historical movies and view how their respective costumers performed in all the films they had participated in. However, the results are rather inconclusive, as almost all of the observed costumers only worked on a single film. The same observations can be made when taking a look at the worst historical movies.

In order to obviate the problem, the same analysis was repeated while only considering costume designers who have worked on 3 historical movies or more, with at least one of them being among the best 50 films. The reasoning, as before, is that a costumer who has worked on a well-received historical movie might have been involved in other good films. From the graph which takes into account the first 9 costumers, the idea seems promising: no movie has been given a "bad" mark. However, if the analysis is repeated on the 50 worst-rated movies, it's easy to see that there's no clear relationship between the average vote given by reviewers and the costume crew: even if a costumer has worked on a "bad" movie, it's quite common to see a collaboration on at least one "good" film as well. In fact, in the selected dataset no costumer has worked on more than one "bad" film.





In conclusion, there is no visible relationship between the reception of a historical movie and the involved costumers. Clearly, the costume crew is either irrelevant or not enough to judge a film, even when considering this specific niche of a genre. While more data might give a better understanding of the situation, the limited dataset that has been used for this analysis is enough to determine that the hypothesis is false under the examined conditions.

Relationship between the number of roles performed and an actor's career

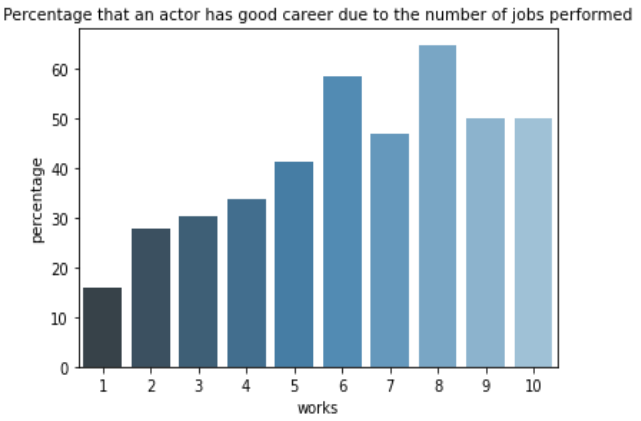
While analysing our datasets, we could see that some people had worked more jobs over the course of their entire career: for example, people who, in addition to the role of actor, had held the role of writer and sound engineer in another film, or a lighting or production worker in other movies. So we thought that an actor specialising in multiple roles related to the film industry could have a benefit in terms of career: there could be a correlation between the number of extra roles that an actor has fulfilled and the possibility of being able to participate in more films that will be assigned a positive rating by reviewers (i.e. "good" movies).

To verify the hypothesis, we collected the data relative to all the people who had worked in at least one movie (about 200'000 observations), then we separated them according to the number of roles, starting from *actor*, played in their entire career (with *role* we mean positions such as *writing*, *production*, *editing*, *sound*, *light*, *photography*...). From this subdivision 10 categories were created, which are divided by number of roles: they start from 1, which includes people who have only played an actor role, up to 10, which involves people who have played 10 different roles throughout their career.

For each person, we calculated the total number of films and highly-rated films they had participated in (where, in our case, "highly-rated" stands for movies that have an average rating greater than 7.5). Then, for each of the 10 categories we selected the people whose career consisted of highly-rated movies by at least 10%, as we considered that the 10% threshold was sufficient to regard a career as "good". For each category, we made a ratio between the number of people belonging to that category and the people (still in that category) who had worked in movies with a rating higher than 7.5 for at least 10% of their career: this

way, it is possible to see how much playing multiple roles can actually affect a person's career.

The results were represented in a graph, where it's possible to see that there is an almost perfectly-growing trend starting from the category with only one role played (*actor*), in which 15% of the people have had a "good" career, up to the category comprising 8 different roles, where 64% of the people have had a good career, therefore bringing a much greater value than those who have worked fewer roles. However, in categories 9 and 10 there is a drop compared to 8: this is most likely due to having too few people in those categories, which makes it hard to have a good understanding of the trend.



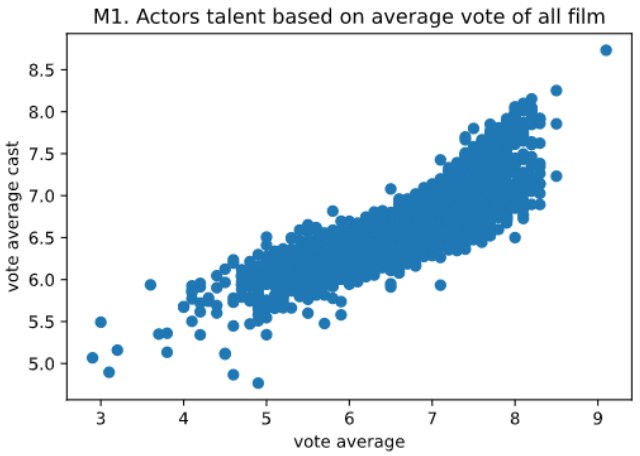
Percentage of good careers compared to the number of roles held.

Best films have best actors

This assumption might come naturally, as it's intuitive that better movies correspond to better actors, but defining a measure about how good a film or an actor is might not be that obvious.

Our dataset already provides a score for the movies, obtained by averaging the votes of each reviewer, so we decided to use this measure in order to define a measurable "best": a film is better the higher its score. When dealing with actors this task turned out to be not so trivial.

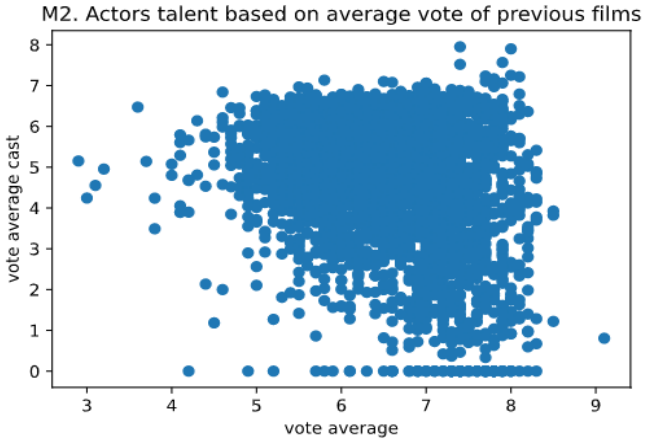
As a first hypothesis, we decided to assign a skill score calculated by averaging all the movie scores in which the actor, or actress, has been involved. So, taking into account this definition, a measurement for "best actors" is nothing more than the average of the actors' skill scores for a certain film. This way, we found the two quantities from which we expect to find some kind of relationship.



Metric 1: Scatter plot between movie vote and cast vote

The graph above suggests that there is a strong correlation between the two measures. This happens because of the incorrect usage of our metric for an actor's skill: if we consider the score of all movies when calculating it, we end up including the vote of the film itself for a certain observation, as well. This way we erroneously introduce correlation between the two measures, which is clearly visible in the graph.

In order to decorrelate these two quantities, we considered the idea of modifying the wrong metric such that, given an actor and a film, we would consider only the movies that precede the current one. The actors' skill scores, then, were recalculated by following this logic. This way, we consider a more realistic assumption by taking into account only the previous career of an actor: indeed, when choosing the cast, the production (or film director) would evaluate an actor for what they have already done without knowing *a priori* their contribution for the success of the movie.

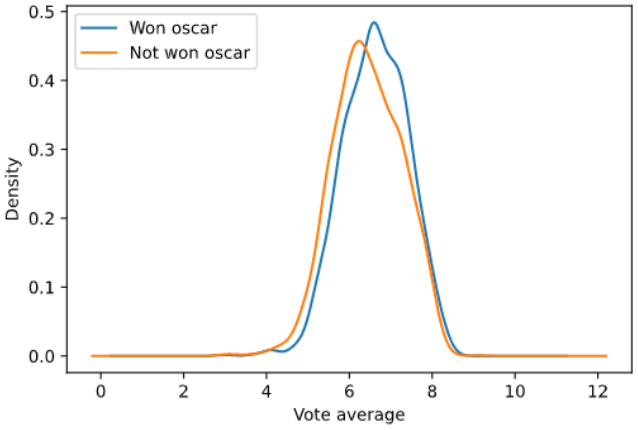


Metric 2: Scatter plot between movie vote and cast vote when considering previous films

The scatter plot now shows clearly that, with this change in metric definition, we cannot assume the existence of any relationship between the movie score and the cast score, the latter of which was obtained, as usual, by averaging the actors' (new) skill score.

One further step was made by also including the *Oscar Awards* collection to introduce a binary variable in our dataset. This variable indicates whether a film has got at

least one actor that has won at least one Oscar throughout their entire career; hence, the movie dataset was split into two categories according to the presence or absence of this kind of actors.



Since the difference between the two scatter plots, after the division, was not so significant, we decided to show the probability density function of the movie votes instead. From the density function we can calculate the probability for a certain range of votes. This way, we found out that sufficient films (movie score greater than 6) are more likely to belong to the Oscar-winning category, rather than the other. The percentages are probabilities of about 77% against 68% to be in a range of grades between 6 and 10, having 10 as the best possible score.

According to these results we can conclude that, for a film, having at least one Oscar-winning actor in the cast will lead to being more appreciated by the reviewers of the platform and will result in a higher score.

Spark Analysis

Deducing the genre of a movie by its plot
Movies are often inspired by their predecessors. If the focus is shifted toward a particular genre, influences from other films in the same group can be easy to identify. This observation begets a question: if movies belonging to the same genre really are so similar to each other, shouldn't it be possible to predict the genre by the plot?

The reduced *MPST* dataset was used to test this hypothesis, which amounts to around 15000 observations, part of which will be lost because of missing genres. A preliminary analysis was performed by counting the number of movies belonging to each genre/tag and plotting the occurrences in decreasing order, as explained in SECTION III. As a reminder, the most-numerous films belong to *murder* (~5800) and *violence* (~4400), while the vast majority of the other genres include less than 1000 movies each. Since the tags are 71 and many of them are vague and underutilised (such as *clever*, *thought-provoking* and *avant garde*), this behaviour is not unexpected. This has led to three different tests.

General case
For this analysis, all the available genres/tags have been considered when building the classifier(s). Since movies in the dataset can belong to many genres, only the first one (the most relevant) has been considered.

After translating the genres into numbered categories and removing the *stopwords* from the plots, a quick analysis was performed by counting the word occurrences on the whole *plot* dataset. Despite the cleaning operations, the most frequent words are generic: in order, the first ones are *tells*, *man*, *house* and *time*. This is regular behaviour. What cannot remain unnoticed, however, is that there's a considerable jump between the first two words: *tells* occurs ~20000 times, while *man* just ~12500. This may be due to the common structure of a written movie plot in which the description reports that the movie "talks" about this, "tells" this kind of story and so on, or perhaps it derives from the impossibility of reporting direct speech, although from this preliminary analysis it's hard to tell.

After getting familiar with the most-occurring words, the next step was to define the features. A simple bag-of-words method was applied to count the words belonging to the plot of each observation, with a vocabulary comprising the 10000 most-frequent terms. A few classifiers were tested, but the choice ultimately fell down on multinomial naive Bayes, as it's quick to execute, gives satisfactory results even if its independence assumptions are not fully respected[4] and has been proved to work well with bag-of-words features in many cases, such as spam filtering[5].
The results are underwhelming: the model accuracy is 19.1%, a very low result, and a quick analysis shows that the best-classified genres, *romantic* and *sci-fi*, are only guessed correctly 55.4% and 53.8% of the time, respectively. Furthermore, the confusion matrix reveals that several genres are completely missed by the classifier, as there isn't enough data for training.

However, the analysis was not for nought: it revealed that the main problem is the abundance of vague, unusable tags which label only a very small subgroup of the dataset. Thus, the analysis was repeated from a different point of view.

Binary case
Among the available genres in the *MPST* dataset, two were chosen based on their frequency and ease of distinction: *murder* and *romantic*. In particular, only movies that belonged to only one of the two genres were selected, which led to ~7000 observations.

As in the previous case, a preliminary analysis on words (barring *stopwords*) was performed and, once again, the most frequent word is *tells* by almost 5000 occurrences (~11100 against ~6700 for *man*); the rest of the word-occurrence values are much closer to each other. Like in the previous case, the most frequent terms include generic words such as *time*, *home* and *room*, but now it's also possible to see some terms that are undoubtedly linked to the selected genres, like *police*, *killed* and *love*. Being so frequent, they are going to play an important role in the classification process.

This time, the bag-of-words approach was replicated with just 1000 terms in the vocabulary; this choice was made after several attempts with different values, which showed that 1000 is enough for both a quick and reliable

classification. In fact, this binary case was dealt with through three classifiers: naive Bayes for a rough and fast result; logistic regression, as it's a standard for binary classification; SVM to perhaps gain an edge by using non-linear boundaries. The dataset was split between training and test, then the classifiers were trained.

Naive Bayes is very fast and reaches an accuracy of 83.0%. As mentioned before, the bag-of-words features representation considers all the words independently from each other, which allows the model to provide reliable results. There is class imbalance, as *murder* movies are 2.3 times the number of *romantic* movies, yet recall is around 80% for both genres. Overall, the result is satisfactory, but it could be improved with additional data.

Logistic regression provides 85.9% accuracy. As a whole, it's better than naive Bayes and it leads to fewer false negatives; however, in percentage, more *romantic* movies are mistakenly classified as *murder* movies (70% recall), which could be due to class imbalance or data scarcity. Overall the classification is shifted towards *murder* movies, which means they are correctly classified more often, but at the same time *romantic* movies are guessed wrong more frequently.

The SVM classifier provides the highest accuracy, equal to 87.4%. Classification imbalance is still more prevalent than in the naive Bayes case, but recall on *romantic* movies is slightly improved with respect to the previous model. The result is satisfying and leaves little improvement when working with the analysed dataset.

The conclusion is that working on genres that are clearly identifiable can lead to valid results. Thus, the next step was to use the same approach on a larger dataset with a limited number of genres.

The genre-overview case on *movie metadata*

For the third case, the analysis was performed on a bigger dataset. Given the available data, the most-promising option was to consider the *movie metadata* dataset. Compared to *MPST*, it provides different features and classes: the genres are now 32 and the long plot descriptions were substituted with shorter movie overviews.

This time, because of the different dataset, the preliminary analysis on the *stopwords*-processed overviews gave different results: the words follow a more regular distribution, with the most frequent being, in order, *life*, *young* and *new*. They are what one could expect from a brief description or summary of a new movie: the terms describe characters (*young*, *old*, *woman*, *father*, *girl*…) and the setting or subject (*story*, *war*, *town*, *school*, *love*...).

The features were represented, as in the previous cases, through bag-of-words; the number of features per observation was set to 1000. At this point, the training-test split yields ~38500 observations for training and ~4300 for the test, which is a much larger amount than with the previous dataset. Naive Bayes, logistic regression and SVM (the latter two via *one vs rest*) were all trained on the available data.

Naive Bayes is quick and easy to train. The resulting accuracy is 42.2%, which is much better than the full *tag-plot* case, but not nearly enough to be considered reliable. Logistic regression and SVM behave similarly and return an accuracy of 42.4% and 43.8%, respectively. Given the comparable performances, only the first one was analysed more thoroughly, as it's simpler.

The confusion matrix reveals that among the genres with the most observations in the test set there are *drama*, *action* and *comedy*. The classifier, however, gets mistaken easily, and the result is that the genre that has been predicted the most accurately is *documentary* (71.6%), followed by *drama* (53.8%) and *comedy* (51.3%); all the others have got an accuracy below 50%. The good prediction capability on two of the largest genres is a welcome result, however the low accuracy values reveal that some improvement is necessary. Even on a large dataset, the high number of genres and the presence of ubiquitous words make classification a hard task.

The conclusion is that this operation requires a clear separation of genres in order to obtain valid results. Even with a large dataset, the frequency of common terms and the murky distinction between one genre and the other makes classification difficult. It's not a case that the best results were had in the binary case between two very different movie genres.

Collaborative filtering

Real-world usable applications of movie analysis vary from an impact study on the market by a production house before actualising the movie to a personalised movie suggestion by a distributor; we focus on the last one.

Movie recommendations could be trivial when knowing, besides their past movie preferences, people characteristics like their age, nationality, education level and such. Public datasets don't offer this kind of information, but even when they are built from a public source deanonymisation is often possible[6], allowing analysts to extract personal information from social media. That is not our case, since the user ratings coming from MovieLens are private, so we work only on past preferences. According to hypothesis *D*, the ratings prediction is purely based on the ratings of other users; this technique is known as *collaborative filtering*.

The first, non-trivial approach applied to building a recommendation system is *slope one*[7]. As expected, it did not outperform a trivial model and returned a negative *R-squared* value (it fits worse than a model which predicts the mean value). This can be due to high data sparsity, which leads to similarity-computation fallacies and/or a non-context-efficient similarity function.

Following these inconclusive results, a promising improvement, based on the *page rank* algorithm, was proposed[8], but *slope one* (or similar algorithms) doesn't show good scalability opportunities; consequently, no action was undertaken.

Another interesting approach, which was demonstrated to be successful in the famous *Netflix Prize challenge,* is the matrix factorisation technique[9]. Roughly speaking, it

consists in the decomposition of a sparse matrix (which includes the *user-movie_rating* pairs) into two rectangular matrices of lower dimensions that represent, respectively, the latent vectors of user and movies; then, a dense matrix is built through the multiplication of the former two, thereby filling the missing ratings.

A latent vector is the result of mapping a high-dimensional space that is directly observed to a lower one, of which the meaning of the single variables is unknown; however, it preserves a significant content of information since, as can be guessed, the quantity of the preserved information depends on the chosen number of latent variables.

$$R = \overset{m}{\rightarrow} \begin{bmatrix} 1.5 & 3.0 & \dots & 5.0 & n.a. & 2.0 & 1.5 \\ 2.0 & n.a. & \dots & 2.5 & 2.0 & 1.0 & 3.0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 1.0 & 3.0 & \dots & 5.0 & 1.0 & n.a. & 1.5 \\ n.a. & 4.5 & \dots & 2.5 & 2 & 1.5 & 3.0 \end{bmatrix} \downarrow u$$

$$U_{lat} = \overset{l}{\rightarrow} \begin{bmatrix} 0.2 & \dots & 2.3 \\ 1.2 & \dots & 7.8 \\ \vdots & \ddots & \vdots \\ 4.1 & \dots & 3.3 \\ 0.8 & \dots & 6.2 \end{bmatrix} \downarrow u \qquad M_{lat} = \overset{m}{\rightarrow} \begin{bmatrix} 0.2 & 2.3 & \dots & 1.2 & 2.3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 4.4 & 2.3 & \dots & 6.2 & 3.2 \\ 2.1 & 1.3 & \dots & 5.6 & 3.7 \end{bmatrix} \downarrow l$$

$$R \approx U_{lat} \cdot M_{lat}$$

R *is the sparse matrix of the ratings,* m *and* u *are, the number of movies and users, respectively;* l *is the dimension of the latent space;* U_{lat} *and* M_{lat} *are the results of the decomposition that contains the latent representation of users and movies.*

Different algorithms can be exploited to apply this approach; we opted for *alternating least squares*, which is already available in the Spark ecosystem[10]. This implementation does not require us to explicitly build the sparse matrix, it provides good scalability and, given the huge amount of data we need to process, it meets our requirements.

The dataset is split into training and test by sampling ratings randomly, keeping in the training set the 80% percent of the samples, and excluding the users that were not sampled for the training dataset. The following results are obtained on the test set.

A post-prediction binarisation of the ratings was applied to predict the *like/dislike* pattern. Every rating over 2.5 out of 5 (the mean rating over the test set) was considered a *like*.

As reference, a model predicting the mean rating of movies, computed on the training dataset, is reported.

Model	**RMSE**	**R^2**	**AUC**
Ref. model	0.9335	0.1828	0.5496
ALS	0.8984	0.3756	0.6732

Conclusion

Throughout our analyses we have examined three large categories: actors, movies and reviewers. While not all of our experiments turned out to be reliable, most of the hypotheses were proven to be either true or false under specific conditions.

Actors:
We have seen that there is correlation between the number of extra roles played by an actor in their career and the possibility of participating in more films that end up being rated positively. People who have fulfilled multiple roles in their career are more likely to have a good career, rather than actors who have only had one or few positions.

From our analysis, we found out that it is not so easy to define a suitable metric to express quantitatively how good an actor is; changing the metric led to different variations of the scatter plots even if we were considering the same dataset. Furthermore, the addition of information about Oscar-winning actors allowed us to distinguish our observations and, this way, we were able to say that a film having *this* kind of actor will be more appreciated by reviewers and so forth, although, compared to the former case, the difference is not so significant. In conclusion we can say that it's very difficult to find a metric for an actor's skill, since it could depend on several features that in most cases, even if using a measurable metric, rely on subjective tastes or opinions.

Movies:
While historical movies place great emphasis on costumes due to their nature, the ability of the costume designer doesn't affect the vote given by reviewers. Evidently, historical movies that are badly-written, badly-acted or badly-received for some other reason can only be improved so much by apt costume choices.

Classifying movies by genre based on their plots can be tricky, as it depends on several factors. The number of genres, the amount of available data and the length of the provided plots can all affect the final accuracy of the classifier. The best results were had with fewer, non-niche movie genres: with a larger quantity, even if the available data is plenty, the classifier ends up making too many mistakes due to class imbalance and the absence of a clear separation among genres.

Reviewers:
As can be easily seen, the ALS model shows good performance. Generally speaking, the results are not enough to estimate the actual rating given by the user, but they are sufficient for a practical usage: recommending a movie based on the preferences previously expressed by the user. This could be improved even more by mixing these predictions with models built upon the previous findings about movie characteristics.

References

[1] https://github.com/mongodb/mongo-hadoop. This library was updated by us for compatibility with the most recent version of MongoDB.
[2] spaCy · Industrial-strength Natural Language Processing in Python.
[3] spark.apache.org.
[4] I. Rish, *An empirical study of the naive bayes classifier*, in IJCAI 2001 workshop on empirical methods in artificial intelligence, pp. 41–46, 2001.

[5] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, *A bayesian approach to filtering junk e-mail*, in Learning for Text Categorization: Papers from the 1998 workshop, vol. 62, pp. 98–105, 1998.

[6] Arvind Narayanan and Vitaly Shmatikov, *How To Break Anonymity of the Netflix Prize Dataset.* arXiv:cs/0610105 [cs.CR], 2007.

[7] Daniel Lemire and Anna Maclachlan, *Slope One Predictors for Online Rating-Based Collaborative Filtering.* arXiv:cs/0702144 [cs.DB], 2007.

[8] Jiang F., Wang Z. (2010) Pagerank-Based Collaborative Filtering Recommendation. In: Zhu R., Zhang Y., Liu B., Liu C. (eds) Information Computing and Applications. ICICA 2010. Lecture Notes in Computer Science, vol 6377. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-16167-4_76

[9] Simon funk. https://sifter.org/~simon/journal/20061211.html, 2006.

[10] https://spark.apache.org/docs/2.4.1/ml-collaborative-filtering.html