Movie Success Prediction Using Machine Learning Algorithms

Author: Guillermo San
Advisors: James Papademas,
José Adrián Rodríguez
July, 2020

This thesis is dedicated to my parents, for providing me with all the means to succeed in life and teaching me the value of effort.

To the rest of my family, for always pushing me to strive for greatness.

Abstract

In a fast-paced evolving world, everything is connected. Sooner than later, our most basic daily routines will be examined. Grocery and drug stores will keep track of all our purchases, smartphones will register how much we walk, run and bike a day and our headphones will keep track of the number of decibels we place our ears under. Despite sounding far-fetched and frankly, a bit scary, all the information collected can be used for our own benefit. There is no denying that the three types of data aforementioned can give us a valuable insight on our health, study possible hazards of our lifestyle and even contribute to finding possible diseases. All that is done through Machine Learning.

The term Machine Learning (ML) was coined in 1959 by Arthur Samuel, an American IBMer and pioneer in the field of computer gaming and Artificial Intelligence (AI). Being nowadays on its prime, most experts define ML as the study of computer algorithms that improve automatically through experience. And don't be fooled, by 'experience' they mean data. Machine Learning requires huge amounts of data on which to build their algorithms to later be able to predict uncertain outcomes. Actually, some experts even point out that the amount – and quality – of data available is far more important than the learning algorithm itself.

This thesis studies several ML algorithms and uses them to successfully predict the revenue and rating of movies by means of a Kaggle movies dataset. Linear Regression (LR), Decision Trees (DT) and Random Forests (RF) algorithms will be discussed and their performance will be evaluated in terms of accuracy.

Resum

En un món en constant canvi, tot està connectat. Tard o d'hora, els elements més bàsics de la nostra rutina diària seran analitzats. Els mercats i les farmàcies tindran constància de tot allò que hem comprat, els nostres telèfons mòbils registraran quant caminem, correm i anem en bici durant el dia i els nostres auriculars detectaran a quants decibels hem sotmès les nostres oïdes. Tot i sonar surrealista i, francament, una mica angoixant, tota aquesta informació pot ser utilitzada en el nostre benefici. És innegable que els tres tipus de dades prèviament esmentats poden proporcionar-nos una valuosa informació sobre la nostra salut, estudiar els possibles riscos del nostre estil de vida o inclús contribuir a detectar possibles malalties. Tot això es duu a terme mitjançant el Machine Learning.

El terme Machine Learning (ML) va ser encunyat el 1959 per Arthur Samuel, un Americà de IBM, pioner en el camp dels jocs d'ordinador i Intel·ligència Artificial (AI). Estant actualment en ple auge, la majoria d'experts defineixen el ML com l'estudi d'algoritmes d'ordinador que milloren automàticament mitjançant l'experiència. I no us deixeu enganyar, al dir 'experiència' es refereixen a dades. el Machine Learning requereix grans quantitats de dades sobre les quals desenvolupar els algoritmes que s'encarreguen de predir resultats incerts. En realitat, alguns experts asseguren que la quantitat – i qualitat – de dades disponible és com a mínim igual d'important que l'algoritme d'aprenentatge en sí.

Aquesta tesi estudia diversos algoritmes de ML i els utilitza per predir de forma exitosa els ingressos i la valoració de pel·lícules mitjançant una base de dades de pel·lícules de Kaggle. Els algoritmes de Linear Regression (LR), Decision Trees (DT) i Random Forests (RF) son analitzats i el seu rendiment és comparat en termes de precisió.

Resumen

En un mundo en constante cambio, todo está conectado. Tarde o temprano, los elementos más básicos de nuestra rutina diaria serán analizados. Los supermercados y las farmacias tendrán constancia de todo aquello que hemos comprado, nuestros teléfonos móviles registrarán cuanto andamos, corremos y pedaleamos durante el día y nuestros auriculares detectarán a cuantos decibelios hemos sometido nuestros oídos. Pese a sonar surrealista y, francamente, una poco agobiante, toda esta información puede ser utilizada en nuestro beneficio. Es innegable que los tres tipos de datos previamente mencionados pueden proporcionarnos una valiosa información sobre nuestra salud, estudiar los posibles riesgos de nuestro estilo de vida o incluso contribuir a detectar posibles enfermedades. Todo esto se lleva a cabo mediante el Machine Learning.

El término Machine Learning (ML) fue acuñado en 1959 por Arthur Samuel, un americano de IBM, pionero en el campo de los juegos de ordenador e Inteligencia Artificial (AI). Estando actualmente en pleno auge, la mayoría de expertos definen el ML como el estudio de algoritmos de ordenador que mejoran automáticamente mediante la experiencia. Y no os dejéis engañar, al decir 'experiencia' se refieren a datos. El Machine Learning requiere grandes cantidades de datos sobre los cuales construir los algoritmos que se encargan de predecir resultados inciertos. En realidad, algunos expertos aseguran que la cantidad – y calidad – de datos disponible es al menos igual de determinante que el algoritmo de aprendizaje en sí.

Esta tesis estudia diversos algoritmos de ML y los utiliza para predecir de forma exitosa los ingresos y la valoración de películas mediante una base de datos de películas de Kaggle. Los algoritmos de Linear Regression (LR), Decision Trees (DT) y Random Forests (RF) son analizados y su rendimiento es comparado en términos de precisión.

Acknowledgments

I would like to thank my tutors James Papademas and José Adrian Rodríguez for the priceless support and advice provided during this thesis development.

List of Acronyms

AI Artificial Intelligence

B&W Black & White

DT Decision Trees

DL Deep Learning

EDA Exploratory Data Analysis

HBO Home Box Office

IBM International Business Machines

IMDB Internet Movie Database

LR Linear Regression

LGR Logistic Regression

ML Machine Learning

MSE Mean Squared Error

MLR Multiple Linear Regression

NLP Natural Language Processing

NaN Not a Number

PR Polynomial Regression

RF Random Forests

RNNs Recurrent Neural Networks

RL Reinforcement Learning

SLR Simple Linear Regression

SL Supervised Learning

SVMs Support Vector Machines

UL Unsupervised Learning

List of Symbols

$\mathbf{X}$ Matrix of examples

$\mathbf{X}_{train}$ Matrix of examples (training set)

$\mathbf{X}_{test}$ Matrix of examples (test set)

$\mathbf{Y}$ Matrix of outputs

$\mathbf{Y}_{train}$ Matrix of outputs (training set)

$\mathbf{Y}_{test}$ Matrix of outputs (test set)

$\mathbf{x}^{(i)}$ i_{th} row of matrix $\mathbf{X}$, corresponding to i_{th} example

$\mathbf{x}_j$ j_{th} column of matrix $\mathbf{X}$, corresponding to the j_{th} feature

$\mathbf{m}$ Number of input/training examples, number of rows of X_{train}

$\mathbf{n}$ Number of features for every example, number of columns of X_{train}

List of Figures

Contents

1 Introduction

1.1 History

In 1952, Arthur Samuel joined IBM's Poughkeepsie Laboratory and began working on what were later known as the very first machine learning programs, first creating programs that played checkers [1].
In 1960's, Bayesian methods were introduced for probabilistic inference in machine learning.
After the decade of 1960, known as 'AI Winter' due to the growing pessimism about its effectiveness, Machine Learning experienced a resurgence thanks to the rediscovery of Backpropagation.
The 1990's brought its final explosion. Work on Machine Learning shifted from a knowledge-driven approach to a data-driven approach. Scientists began creating programs for computers to analyze large amounts of data and draw conclusions – or "learned" – from the results. Support Vector Machines (SVMs) and Recurrent Neural Networks (RNNs) became popular.

In 2014, Facebook researchers published their work on DeepFace, a system using Neural Networks that identified faces with 97.35% accuracy. The results were an improvement of more than 27% over previous systems and rivaled human performance [2].

According to Forbes magazine, same-day shipping from Amazon is available thanks to Machine Learning. In fact, their current ML algorithm has decreased the 'click-to-ship' time by 225%.

1.2 Project outline

This thesis reviews the basics of Machine Learning and implements three of its most well-known algorithms. In a summary, the project goals are the following:

- Review basic concepts of Machine Learning and its applications
- Perform a thorough Data Analysis on a dataset
- Implement a total of three Machine Learning algorithms and compare their performance

1.3 Organization

The development of this thesis has been organized in 4 work packages, which are shown in Figure 1.

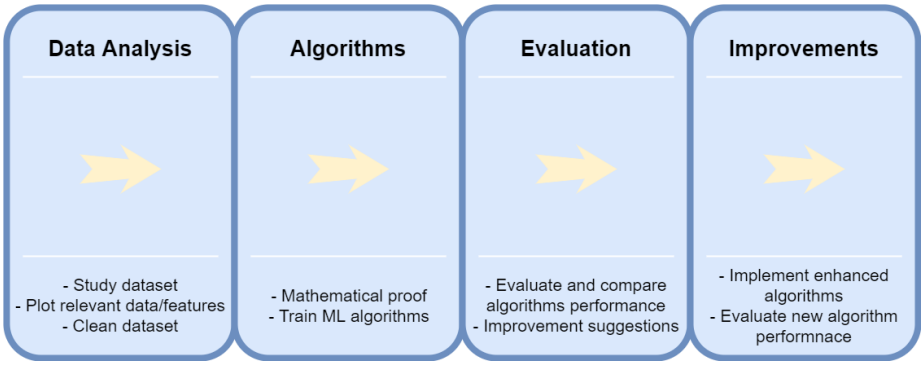


Figure 1: Work packages distribution

- Data Analysis: Data science techniques will be used to extract valuable information from data. Incomplete data will be dealt with.
- Algorithms: The development of several Linear Regression models will be carried out, together with their mathematical proof.
- Evaluation: The aforementioned algorithm's performance will be evaluated in terms of accuracy.
- Improvements: Decision Trees and Random Forests algorithms will be implemented, and their results will be compared with those obtained through the LR models.

This thesis has implied a total time of 6 months. The time distribution of the different tasks undertaken can be seen in Figure 2.

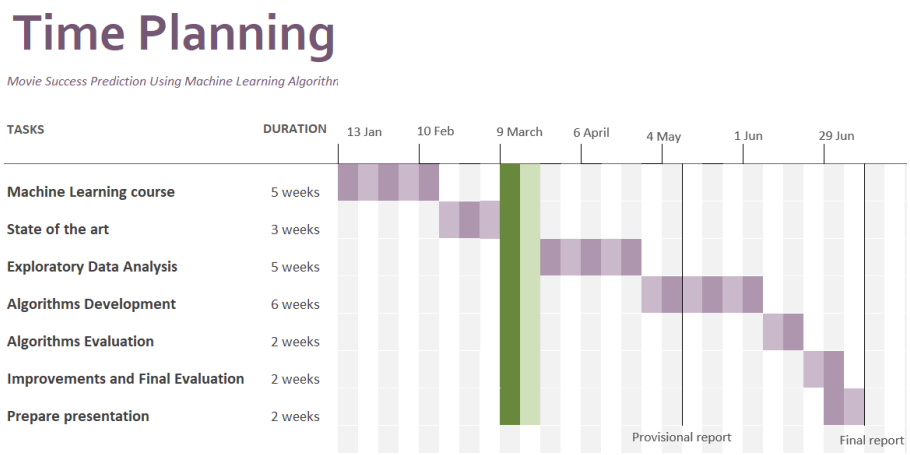


Figure 2: Project Time Planning

The Machine Learning course took 5 weeks and then the research phase to study the State of the Art and project goals definition took 3 weeks.
The Exploratory Data Analysis (EDA) took 5 more weeks.
Later, 6 weeks were used to develop the algorithms and their mathematical proof.
After that, 2 weeks were spent on the algorithms performance evaluation and the study of further improvements.
Finally, 2 weeks were used for developing the other algorithms and performing the final evaluation.

2 State of art

2.1 Introduction

What is Machine Learning?
Despite being pretty abstract concept, most ML developers would agree that Machine Learning is a branch of Artificial Intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimum human intervention. Although a controversial classification, most experts agree that the main areas of Artificial Intelligence are:

- Machine Learning
- Deep Learning
- Natural Language Processing (NLP)
- Expert Systems
- Computer Vision
- Robotics

In [3], Machine Learning is defined as the programming of computers to optimize a performance criterion using example data or past experience.

Deep Learning can be utilized for addressing some important problems in Big Data Analytics, including extracting complex patterns from massive volumes of data, semantic indexing, data tagging, fast information retrieval, and simplifying discrimination tasks [4].

Authors in [5] mention speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics as the main applications for Deep Learning.

NLP is defined as the function of software or hardware components in a computer system which analyze or synthesize spoken or written language [6], where the term 'natural' refers to human language as opposed to other languages such as mathematics notation or programming languages. NLP is usually used for automatic question answering, sentiment analysis or even toxic comments detection.

2.2 Types of Machine Learning

Machine Learning algorithms are divided into two main categories[1]:

- Supervised Learning: Includes the vast majority of Machine Learning algorithms. They are called supervised because they partially require the solution to the problem they are trying to solve. In Supervised Learning (SL) we can find input variables x and the output variable Y, being:

  $$Y = f(x)$$

  The goal in SL is to approximate the mapping function f so well that, when prompted with new data x, the algorithm can perfectly predict its corresponding output Y. Supervised Learning algorithms and grouped into two major categories:

  - Classification algorithms: the output variable is a category, such as "cat"/"dog" or "disease"/"no disease".
  - Regression algorithms: the output variable is a real value, such as "dollars" or "weight"

- Unsupervised Learning: Refers to problems where only the input data x is available and no corresponding output variables Y. The goal for Unsupervised Learning (UL) is to model the underlying structure or distribution in the data, in order to learn more about it.
  An example of UL could be an algorithm that, given a database of animals' pictures, identifies how many different animals there are and creates groups with all the pictures of every animal. In that case, the algorithm would put all dog pictures together, but would have no idea on what label to put on that group. In other words, the algorithm would not know what animal it is.

- Reinforcement Learning: In this type of ML, software gets rewards for winning and punishment for losing. Reinforcement Learning (RL) algorithms are typically applied to games and forgery detectors.

[1]Sometimes Reinforcement Learning is also considered the third type of Machine Learning algorithms, but that is out of the scope of this work.

In Figure 3, an scheme showing the two major ML algorithms sub-types is shown:

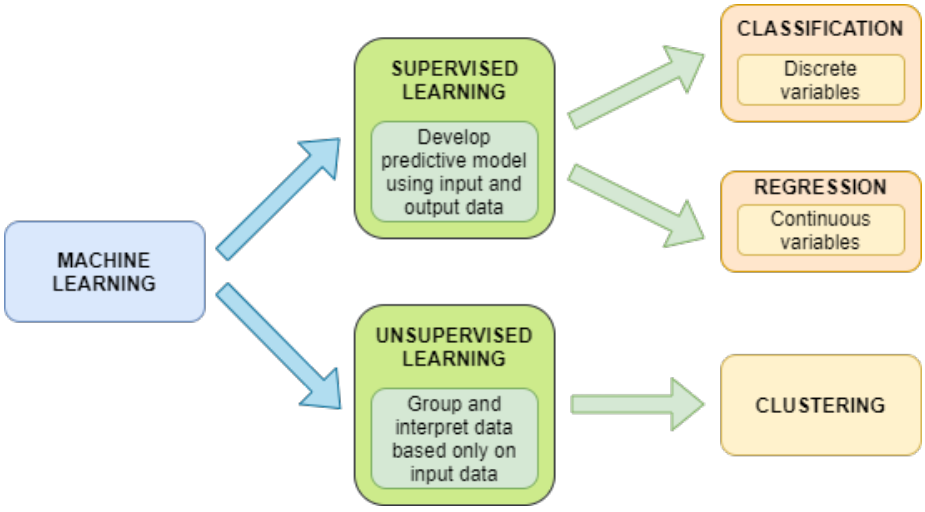


Figure 3: Machine Learning algorithms scheme

Then again, there are many ML SL algorithms that implement Classification and many others that implement Regression. Despite the list being endless, the most relevant ones up to date could be summarized in:

- Classification
  - Support Vector Machines (SVM)
  - Decision Trees (DT)
  - Random Forests (RF)
- Regression
  - Linear Regression (LR)
  - Logistic Regression (LGR)

Similarly, the most common ML UL algorithms are:

- Unsupervised Learning
  - K-Means Clustering

2.3 Applications

Machine Learning is growing at an extremely fast pace. Both the reason and the consequence of that phenomenon is the large amount of interdisciplinary applications that are discovered every year for this technology. From home gadgets and computer games to health and cybersecurity, Machine Learning is present in almost every aspect of our lives.

In [7], authors use protein biomarkers and microarray data to successfully detect cancer. In big companies like Facebook, nearly all aspects of user experience make use of Machine Learning. Ranking posts for news feed, speech and text translations and real-time video are just some of the examples [8].

Ads in all cutting-edge websites leverage ML to determine what content to display to a given user. Ads models are trained to learn user traits and context, previous interactions and interests and many more. Thanks to that, ML algorithms can predict the likelihood for a user to click on an ad, visit a website, and/or purchase a product [9].

In [10], authors discuss the challenges derived from both Face Detection (properly detecting a human face in a picture) and Face Recognition (determining who the face corresponds to). In this work, the object-detection algorithm proposed by Viola and Jones is used due to its high accuracy and speed.

In [11], authors discuss the applications of ML in pattern recognition. In that field, ML could be used to process handwritten characters, such as zip codes on envelopes or amounts on checks.

Authors in [12] use Machine Learning techniques for forecasting short-term loads and prices as well as very short-term wind power prediction in power systems.

3 Exploratory Data Analysis

3.1 Introduction

In this section, the *IMBD Movies Dataset* from Kaggle will be studied which can be found on:

https://www.kaggle.com/kevalm/movie-imbd-dataset

Relevant data statistics will be displayed in order to have a general overview of the data distribution and meaningful, hidden information present on the dataset will be shown.

3.2 The dataset

The dataset used in this work is formed by 5043 rows (movies) and 28 columns (features).

The columns on this dataset are:

- color: either B&W or Color
- director_name: name of the movie's director
- num_critic_for_reviews: number of official newspapers/critics that have reviewed the movie on IMDB
- duration: length of the movie in minutes
- director_facebook_likes: number of likes the director's profile has on Facebook
- actor_3_facebook_likes: number of likes the 3rd character's profile has on Facebook
- actor_2_name: name of the 2nd character
- actor_1_facebook_likes: number of likes the 1st character's profile has on Facebook
- gross: gross revenue generated by the movie
- genres: all the genres present in the movie
- actor_1_name: name of the 1st character
- movie_title: title of the movie

- num_voted_users: number of users that gave the movie a rating on IMDB
- cast_total_facebook_likes: sum of all cast Facebook profiles' likes
- actor_3_name: name of the 3rd character
- facenumber_in_poster: number of faces that appear in the movie poster
- plot_keywords: list of key words on the plot description
- movie_imdb_link: link to the movie on IMDB site
- num_user_for_reviews: number of users that gave the movie a review on the IMDB platform
- language: original language of the movie
- country: country of origin of the movie
- content_rating: type of public the movie is intended for
- budget: amount of money allocated for the movie
- title_year: year the movie was released
- actor_2_facebook_likes: number of likes the 2nd character's profile has on Facebook
- imdb_score: rating over 10 on the IMDB site
- aspect_ratio: dimensions of the screen used
- movie_facebook_likes: number of likes the movie has on Facebook

On Figure 4, a snapshot of the first 10 rows of the dataset can be seen, showing the (a priori) most relevant features of the movies.

	movie_title	color	director_name	genres	duration	title_year	budget	gross	imdb_score
0	Avatar	Color	James Cameron	Action|Adventure|Fantasy|Sci-Fi	178.0	2009.0	237000000.0	760505847.0	7.9
1	Pirates of the Caribbean: At World's End	Color	Gore Verbinski	Action|Adventure|Fantasy	169.0	2007.0	300000000.0	309404152.0	7.1
2	Spectre	Color	Sam Mendes	Action|Adventure|Thriller	148.0	2015.0	245000000.0	200074175.0	6.8
3	The Dark Knight Rises	Color	Christopher Nolan	Action|Thriller	164.0	2012.0	250000000.0	448130642.0	8.5
4	Star Wars: Episode VII - The Force Awakens ...	NaN	Doug Walker	Documentary	NaN	NaN	NaN	NaN	7.1
5	John Carter	Color	Andrew Stanton	Action|Adventure|Sci-Fi	132.0	2012.0	263700000.0	73058679.0	6.6
6	Spider-Man 3	Color	Sam Raimi	Action|Adventure|Romance	156.0	2007.0	258000000.0	336530303.0	6.2
7	Tangled	Color	Nathan Greno	Adventure|Animation|Comedy|Family|Fantasy|Musi...	100.0	2010.0	260000000.0	200807262.0	7.8
8	Avengers: Age of Ultron	Color	Joss Whedon	Action|Adventure|Sci-Fi	141.0	2015.0	250000000.0	458991599.0	7.5
9	Harry Potter and the Half-Blood Prince	Color	David Yates	Adventure|Family|Fantasy|Mystery	153.0	2009.0	250000000.0	301956980.0	7.5

Figure 4: First 10 rows of the dataset

It's worth noting that, even on the first 10 rows, there are already some values missing, in this case, row number 4. Star Wars movie value for some fields is Not a Number (NaN), a special value that will be dealt with later on.
In relation to that, the feature with more missing values is the gross revenue, with 4159 out of 5043 values, or just 82.47% of them.

On Figure 5, some stats are shown for the numeric features of the dataset, the most significant ones (a priori) being the budget and gross revenue, as well as the IMDB score.

	budget	gross	duration	actor_1_facebook_likes	actor_2_facebook_likes	actor_3_facebook_likes	movie_facebook_likes	imdb_score
count	4.551000e+03	4.159000e+03	5028.000000	5036.000000	5030.000000	5020.000000	5043.000000	5043.000000
mean	3.975262e+07	4.846841e+07	107.201074	6560.047061	1651.754473	645.009761	7525.964505	6.442138
std	2.061149e+08	6.845299e+07	25.197441	15020.759120	4042.438863	1665.041728	19320.445110	1.125116
min	2.180000e+02	1.620000e+02	7.000000	0.000000	0.000000	0.000000	0.000000	1.600000
25%	6.000000e+06	5.340988e+06	93.000000	614.000000	281.000000	133.000000	0.000000	5.800000
50%	2.000000e+07	2.551750e+07	103.000000	988.000000	595.000000	371.500000	166.000000	6.600000
75%	4.500000e+07	6.230944e+07	118.000000	11000.000000	918.000000	636.000000	3000.000000	7.200000
max	1.221550e+10	7.605058e+08	511.000000	640000.000000	137000.000000	23000.000000	349000.000000	9.500000

Figure 5: Some stats for dataset's numeric features

For example, the average value for the movies budget is $39,752,620 and the mean gross income is $48,468,410. That being said, we can conclude the average profit for a movie from this dataset is approximately $8,715,790.
Similarly, the number of values (count), the standard deviation (std), together with the min, max and quartiles values are shown.

3.3 Data distribution

Although Figure 5 shows a lot of useful metrics, it is difficult to get an exact idea of the information it conveys. Do all the movies share more or less the same duration, budget and gross revenue? How much spread are their scores? Were they released around the same year?
To answer all these questions, let's now take a look to the data in a more graphical approach.

How old are movies in the dataset? To answer that question, the year title of the movies is shown in Figure 6. Movies from this dataset were released as early as 1916 and as late as 2016. Despite this 100-year span, just 15% of the movies from this dataset were released before 1995 (79 years), and 85% from 1995 to 2016 (32 years). For that reason, the average year of release for our movies is the year 2002.

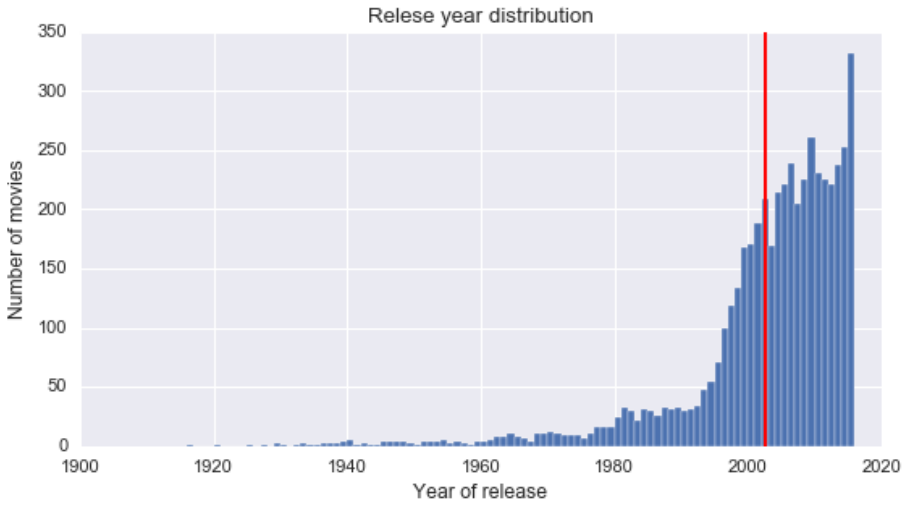


Figure 6: Distribution of movies release year

Another feature to look at is the movies color. That is, whether they are Black & White movies or Color movies. In Figure 7, a pie chart shows this distribution, with 4815 Color movies (95.84%) and 209 B&W ones (4.16%).

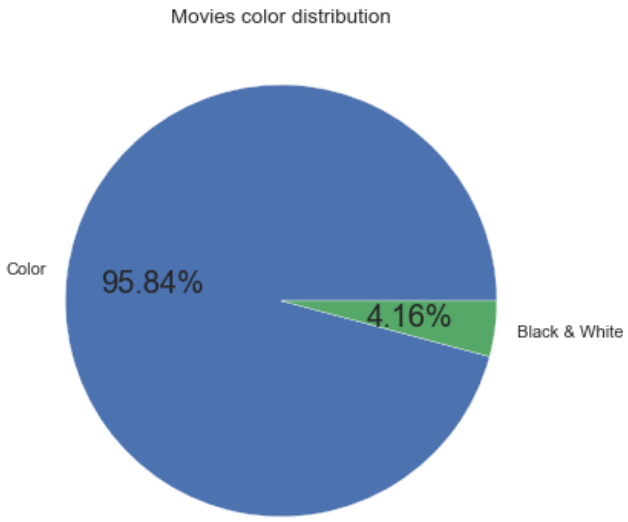


Figure 7: Distribution of Color and Black & White movies

In Figure 8, the distribution for the duration of the movies (in minutes) is shown. It can be observed that most of the movies (88%) have a duration between 80 and 140 minutes, what could be considered as standard movie duration times.

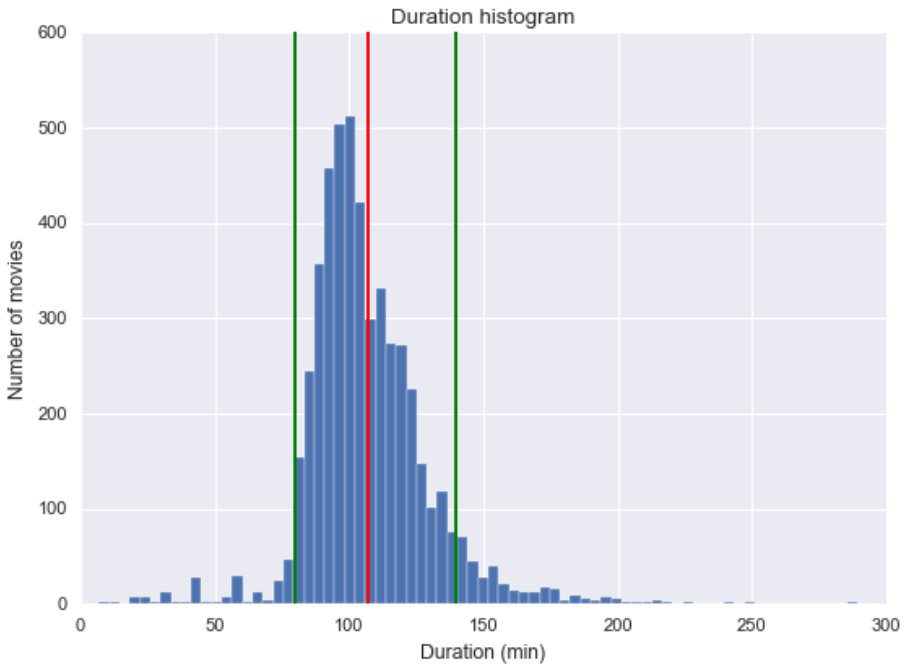


Figure 8: Distribution of movies duration in minutes

In Figure 9, the distribution for the budget of 4538 out of the 5043 movies (90%) can be seen. There are 492 values missing (9.75%) on the dataset and 13 of them (0.25%) have been left our of the histogram for being outliers (having a budget greater than $390M), far from the rest of the data.

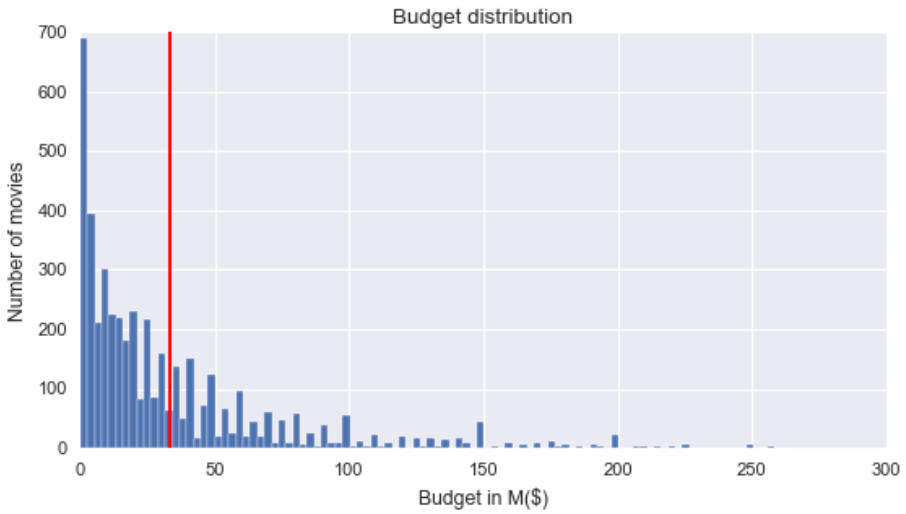


Figure 9: Distribution of movies budget in M($)

To put the effect of this 13 outliers in perspective:

- The average budget for the movies shown on the histogram is $33.38M and the maximum value is $267.3M.
- The average budget for the overall dataset is $39.75M and the maximum value is $12,215.5M, what is 307 times the dataset mean or simply more than 12 billion dollars.

All in all, it is clear that the presence of outliers, despite representing less than 1% of the data, constitute an issue that should be taken into account, as all the metrics and, consequently, all the relations in the data, are susceptible of being deeply affected by them.

In Figure 10, the distribution for the gross revenue of 4124 out of the 5043 movies (81.78%) can be seen. There are 884 values missing (17.53%) on the dataset and 35 of them (0.69%) have been left our of the histogram for being outliers (having a gross revenue greater than $350M), far from the rest of the data.

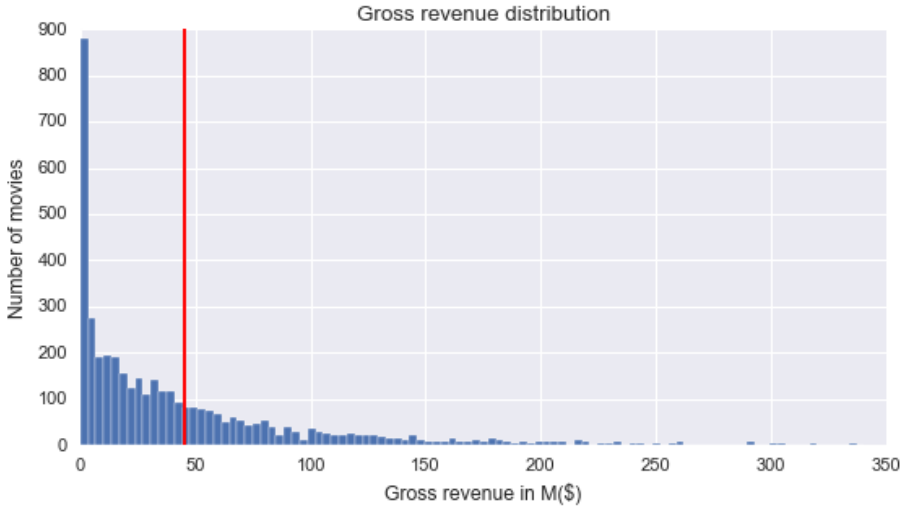


Figure 10: Distribution of movies gross revenue in M($)

It's interesting to note the immense peak right next to the origin, that represents a great deal of movies with a poor gross revenue. As revealed on the stats in Figure 5, the 25-percentile of the gross revenue is $5.34M. That means that 25% of this dataset movies had an income of $5.34M or lower.

In Figure 11, the distribution for the score obtained on IMDB by the movies from the dataset is shown.

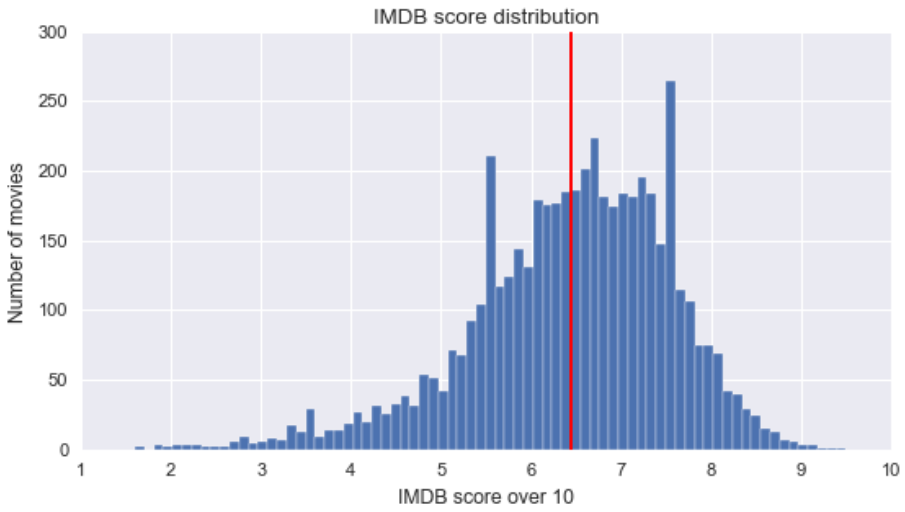


Figure 11: Distribution of movies score

The minimum score obtained is 1.6/10 and the maximum is 9.5/10. The average score is 6.44/10 and most films (85.43%) fall in the range between 5/10 and 8/10, which could be considered acceptable movies.

There is other data on the dataset can be analyzed and discarded without the need to plot it.
In Figure 5, the number of movie likes' 50-percentile is 166, while the maximum is 349,000. This means most movies don't have a strong presence on Facebook and, consequently, that is not a relevant metric to consider.
That same pattern is appreciated in the actor_1_facebook_likes feature (50-percentile is 988 and maximum is 640,000), and actor_2_facebook_likes (50-percentile is 595 and maximum is 137,000).

Other data that may seem irrelevant may end up playing a crucial row on trying to predict the revenue and rating of a movie. In Figure 12, the distribution of the number of users that reviewed a movie on IDMB is shown for 4970 movies over 5043 (98.57%), as 21 values were missing (0.43%) and 52 values (1.05%) were left out as outliers.

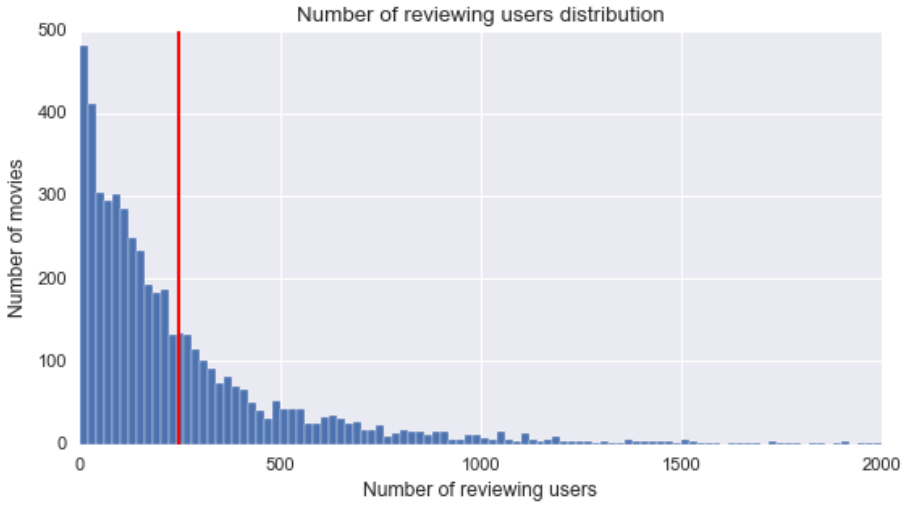


Figure 12: Distribution of the number of reviewing users for a movie

As seen on the image, the average number of users that review a movie from our dataset on IMDB is 248.

3.4 Stats and information

There is no doubt these columns (or features) provide valuable information on the movies under study. It seems logical to think that a movie with a well-known director, a higher budget or a famous actor is more likely to generate a higher revenue and/or obtain a better rating on the IMDB platform. But, is that so? To answer that question, let's take a look at the correlation between the different features.

For that purpose, the correlation matrix [2] in Figure 13 provides a bird perspective of these relations for the numeric features.[3].

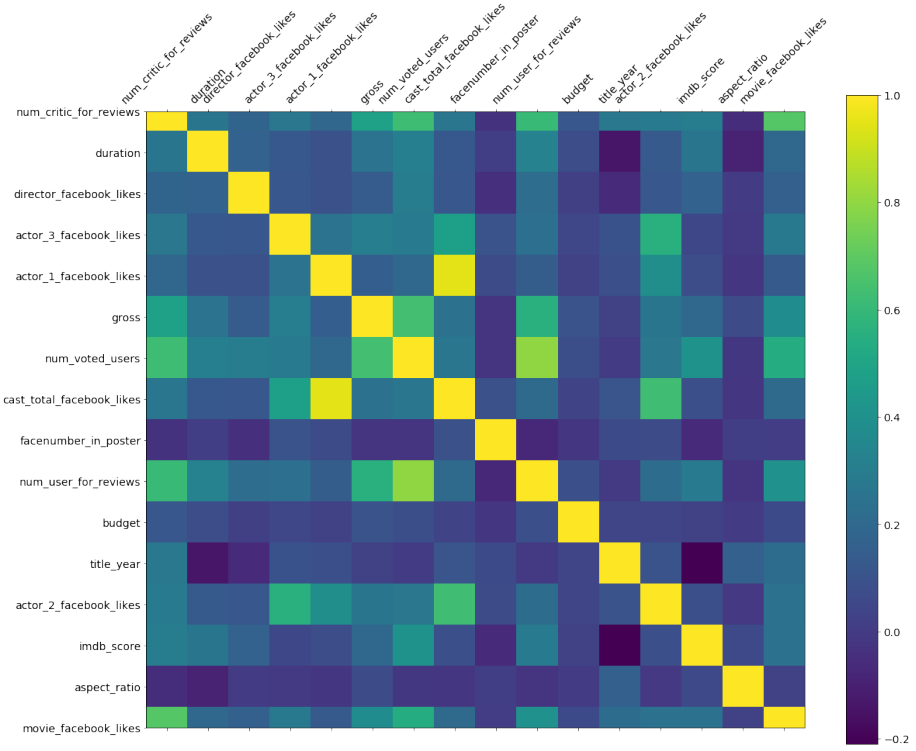


Figure 13: Correlation matrix

[2] Although the numeric version is much more precise, this visual one is included for easier interpretation. The numeric correlation matrix can be found on Annex 1.

[3] In this visual correlation matrix, yellow cells indicate positive correlations and purple cells indicate negative (or inverse) correlations. In both cases, the darker the color, the stronger the correlation in absolute value, as the legend shows.

Being all the coefficients of the diagonal equal to 1 (maximum correlation factor for a feature with itself), let's now analyze other correlations that are considerably high (yellow-ish).
In that sense, it can be observed that there is a high correlation betwen the feature actor_1_facebook_likes and cast_total_facebook_likes (specifically a correlation factor of 0.95), which seems obvious, as the total cast likes include the main actor's ones.

Although not as elevated, some correlations that do convey some information are, for example, the inverse correlation between the duration and the title year (dark blue). This negative correlation implies that, the newer a movie is ('higher' year), the shorter it is on average.

Figure 14 shows the evolution of the movies' duration (and the greater amount of movies) over time for both Color and B&W movies.

Figure 14: Duration of movies in minutes over the years

Then, are movies granted a higher budget over time? In Figure 15, the evolution of the allocated budget for movies over time us shown.

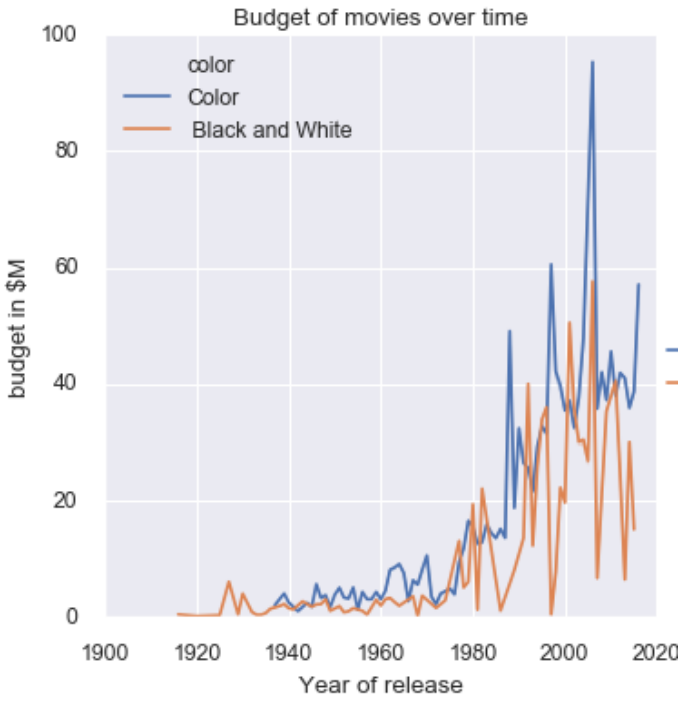


Figure 15: Evolution of allocated budget over the years

From the graphic, it can be seen that Color movies usually have a higher budget, and that both Color and B&W movies' budget has increased over time, especially from the 1980's onward.

In Figure 16, the evolution of the gross revenue of movies over time can be seen for both Color and B&W movies.

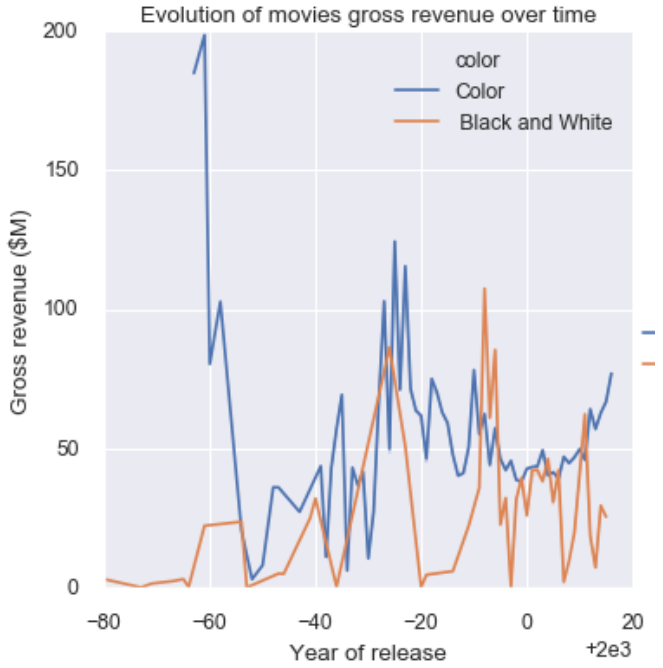


Figure 16: Evolution of gross revenue over the years

Again, Color movies generated a higher income than B&W ones consistently over time, although somewhere around the 2000's this gap started closing, probably because the few movies that are currently recorded in B&W have actually the same resources as Color ones do.

In Figure 17, the profitability factor over time is shown for both Color and B&W movies[4].

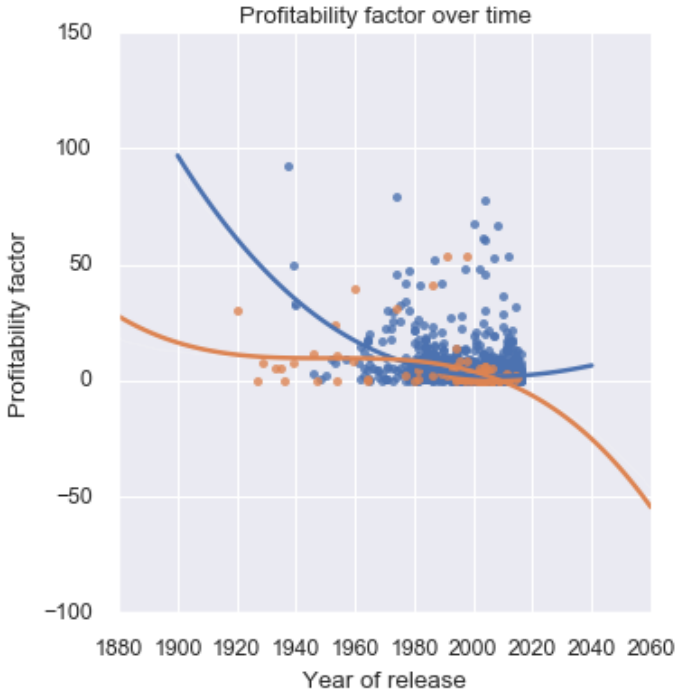


Figure 17: Evolution of movies profitability over times

As shown in the image, movies from this dataset have became less profitable over time. The main reason for that phenomena is probably the fact that movies' budget has tremendously increased over the last few years, whereas their gross revenue (although higher), has not been able to keep with that pace, probably due to the increasing amount of movies released each year and the appearance of streaming platforms like Netflix or HBO, which hinder movies revenue in cinemas.

[4]The profitability factor is not a default metric of the dataset. The profitability factor is defined in this work as the relation between the gross revenue of a movie and its budget. This means, if higher than 1, the movie was profitable and, if lower than 1, the movie had losses.

In Figure 18, the evolution of the score obtained on the IMDB platform by movies over time is shown.

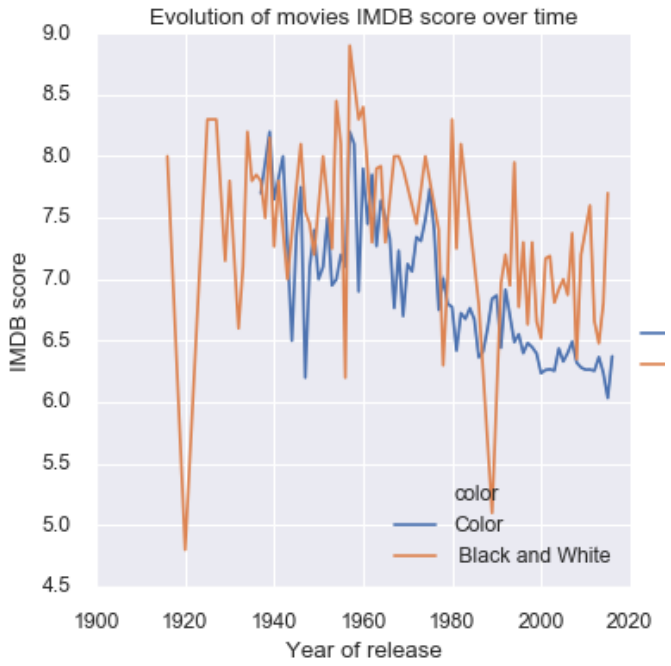


Figure 18: Evolution of IMDB score over the years

It's worth noting that, whereas the B&W movies score has being somewhat regular over time (despite several spikes), the Color movies started with decent scores (7.5/10) around the 1940's and have being doing worse over time, with an average score of 6/10 in the last years.
From the 1990's onward, B&W movies from this dataset have a higher average score than Color movies, again probably due to their matching resources and their uniqueness.

And then, is there a relation between the movies' duration and the gross revenue they generate or the rating they get on IMDB?
Back to Figure 13, we can see there is in fact a positive-valued correlation factor between movie duration and gross (precisely 0.25), as well as between movie duration and IMDB score (0.20). Let's now take a look at these relations.

First of all, Figure 19 shows the relation between the duration of the movies and the gross revenue they obtained.

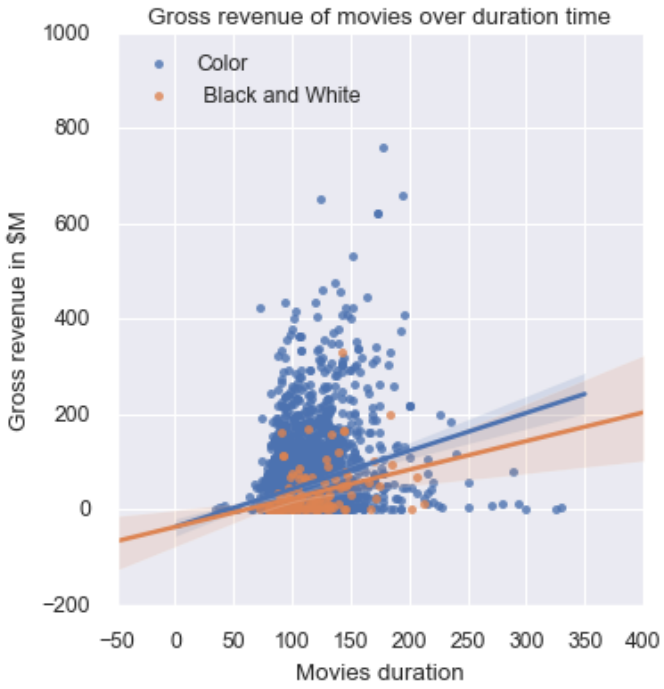


Figure 19: Gross revenue as a function of movies' duration

As seen on the graphic, long-lasting movies tend to do better both for B&W and Color movies. Moreover, that relation is specially important on the Color movies, as the linear equation hereby used to model that relation presents a higher slope. However, the dataset under study has way more examples of Color movies than it has of B&W ones, so it would be interesting to compare this linear relation with a greater amount of B&W movies.

Similarly, Figure 20 shows the relation between the duration of the movies and the score they obtained on the IMDB platform.

Figure 20: IMDB score as a function of movies duration

In this case, a second order polynomial has been used to estimate the relation between the two features, resulting in Color movies having a greater score for longer movies up to 250 minutes and falling then onward. B&W movies, on the other hand, present a constant increase on their rating with respect to their duration, although again, a greater amount of data could affect that tendency.

Finally, let's analyze the two features we will later try to predict using Machine Learning algorithms: IMDB score and Gross revenue. How tied are they?
The correlation matrix on Figure 13 says not so much, with a (obviously positive) but small correlation coefficient of 0.198.
But is this relation linear or polynomical?
In Figure 21, the gross revenue of movie is represented as a function of their IMDB score, together with their linear relation.

Figure 21: Gross income as IMDB score (linear relation)

In this case, both Color and B&W have a higher revenue if their score is higher (and vice versa), especially Color movies.

In Figure 22, that same relation is shown, this time estimating it with a polynomial of 2_{nd} order.

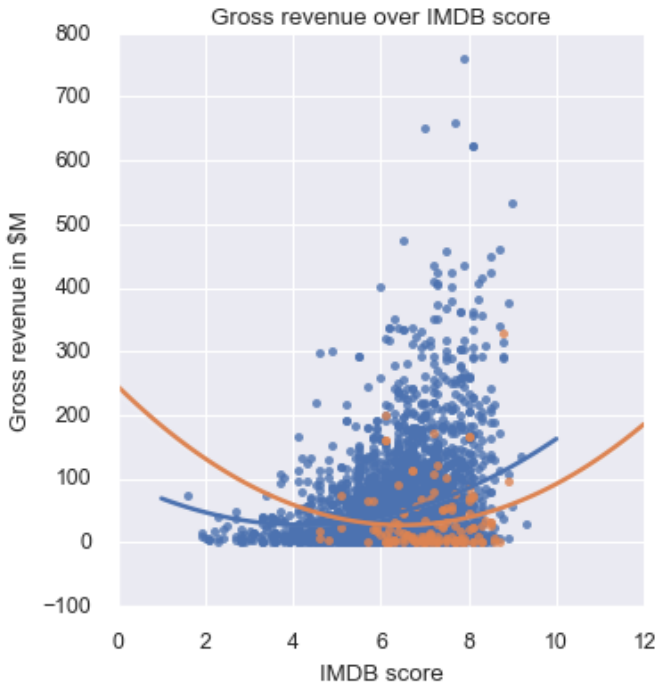


Figure 22: Gross income as IMDB score (polynomial relation)

In this case, again both Color and B&W movies associate a greater revenue stream with a higher IMDB rating and again this relation is stronger in the case of Color movies.

4 Methodology and project development

4.1 Introduction

In this section, the different algorithms under study will be discussed and examined from a mathematical point of view.

Nevertheless, prior to using any ML algorithm, the dataset has to be properly managed. The original dataset will be divided into two sets: training and testing. Usually, ML algorithms use 80% of their data set examples for training the algorithm and 20% for testing it.
Following that rule, our dataset (5043 examples) will be divided into a training set of **m**=4034 examples[5] to train the ML algorithm and 1009 examples to actually test its precision[6].

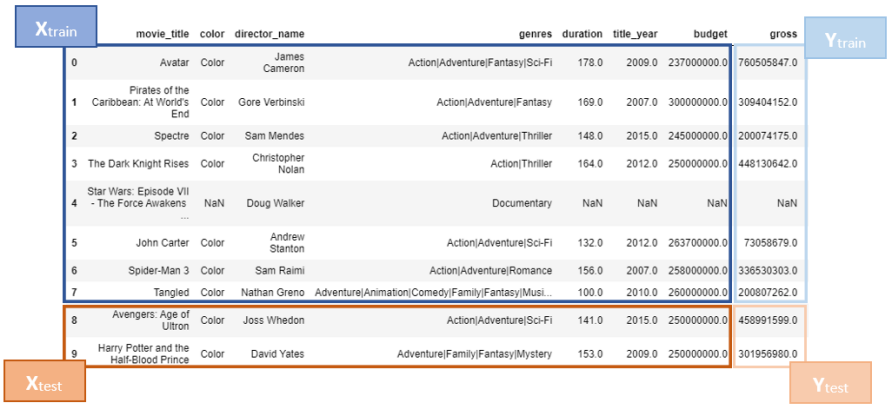
Later, both training and testing sets are divided into two groups: the input features (information available) and the output features (prediction).
For that reason, the whole original dataset is finally divided into:

$$\mathbf{X}_{train}, \quad \mathbf{Y}_{train}, \quad \mathbf{X}_{test}, \quad \mathbf{Y}_{test}$$

[5]We will later see that is not exactly true, as NaN values cannot be used.

[6]In this case, the term 'precision' is used in a generic way. The different metrics to evaluate a ML algorithm's performance will be further discussed on section Annex 2: Metrics.

In Figure 23, an example of this partition is shown for a database of **m**=10 examples and **n**=7 features.



	movie_title	color	director_name	genres	duration	title_year	budget	gross
0	Avatar	Color	James Cameron	Action|Adventure|Fantasy|Sci-Fi	178.0	2009.0	237000000.0	760505847.0
1	Pirates of the Caribbean: At World's End	Color	Gore Verbinski	Action|Adventure|Fantasy	169.0	2007.0	300000000.0	309404152.0
2	Spectre	Color	Sam Mendes	Action|Adventure|Thriller	148.0	2015.0	245000000.0	200074175.0
3	The Dark Knight Rises	Color	Christopher Nolan	Action|Thriller	164.0	2012.0	250000000.0	448130642.0
4	Star Wars: Episode VII - The Force Awakens ...	NaN	Doug Walker	Documentary	NaN	NaN	NaN	NaN
5	John Carter	Color	Andrew Stanton	Action|Adventure|Sci-Fi	132.0	2012.0	263700000.0	73058679.0
6	Spider-Man 3	Color	Sam Raimi	Action|Adventure|Romance	156.0	2007.0	258000000.0	336530303.0
7	Tangled	Color	Nathan Greno	Adventure|Animation|Comedy|Family|Fantasy|Musi...	100.0	2010.0	260000000.0	200807262.0
8	Avengers: Age of Ultron	Color	Joss Whedon	Action|Adventure|Sci-Fi	141.0	2015.0	250000000.0	458991599.0
9	Harry Potter and the Half-Blood Prince	Color	David Yates	Adventure|Family|Fantasy|Mystery	153.0	2009.0	250000000.0	301956980.0

Figure 23: Training and testing split example

In the dataset under study, $\mathbf{X}_{train}$ is composed of **m** examples with **n** features each and $\mathbf{Y}_{train}$ would be the output feature (i.e. gross revenue or IMDB score) of those examples. Then $\mathbf{X}_{test}$ would be the set of movies for which we want to predict the output feature (to test the built model) containing their corresponding input features and finally, $\mathbf{Y}_{test}$ would be the actual output feature of those testing examples

What Machine Learning Regression algorithms do is, based on the examples from $\mathbf{X}_{train}$ and their available 'solution' $\mathbf{Y}_{train}$, build a model which is later used to predict the output of $\mathbf{X}_{test}$ (usually referred to as $\mathbf{Y}_{pred}$) and compare them to the actual solutions $\mathbf{Y}_{test}$ to check the performance of that algorithm.
However, the reasoning and the mathematical process behind every algorithm to achieve this is completely different. In this section, the behind-the-scenes of these algorithms is shown.

4.2 Linear Regression

Linear Regression (LR) is one of the most used Machine Learning algorithms. LR is a Supervised Learning algorithm, more specifically a Regression algorithm. This means LR tries to model the relationship between the dependent (or output) variable and one (or several) independent (or input) variables. The case of one unique input variable (or feature) is known as **Simple Linear Regression (SLR)**, whereas the case of several input variables (or features) is called **Multiple Linear Regression (MLR)**.

What LR does is come up with the linear equation that minimizes the Mean Squared Error (MSE), also known as the average squared distance of all training examples to that line.
Then, for any new example, the algorithms estimates the output it could have by mapping it using the built equation.

But how does the algorithm come up with that estimator line?
The model builds a hypothesis h_θ based on all the pairs of training examples and their outcomes:

$$(x^{(i)}, y^{(i)}) \tag{1}$$

Being $x^{(i)}$ the i_{th} training example and $y^{(i)}$ its corresponding output feature. In a Linear Regression case, the hypothesis[7] has the form:

$$h_\theta(x) = \theta_0 + \theta_1 \mathbf{x} \tag{2}$$

Where θ_0 is often referred to as the **intercept** and θ_1 is known as the **coefficient**. The goal is that the modeled hypothesis maps the training examples (in vector $\mathbf{x}$) to their output value as similarly as possible to their actual value $\mathbf{y}$. In mathematical terms, the optimization problem is expressed:

$$min_{\theta_0,\theta_1} \ (\mathbf{h}_\theta(x) - \mathbf{y})^2 \tag{3}$$

[7]For simplicity, the hypothesis $h_\theta()$ is usually written down as simply h(). For that same principle, in this work the sub-index θ will eventually fade into the dark abyss.

Where vector $\mathbf{y}$ contains the output value of all the training examples. This minimization problem is usually expressed as:

$$min_{\theta_0,\theta_1} \ \frac{1}{2m}(\mathbf{h}_\theta(x) - \mathbf{y})^2 \tag{4}$$

The $\frac{1}{2m}$ factor[8] does not alter the solution of the minimization (optimization of a constant) and is introduced to match the definition of Mean Squared Error (MSE).

The function we are trying to minimize is usually often referred to as the cost function, $J(\theta_0, \theta_1)$:

$$J(\theta_0, \theta_1) = \frac{1}{2m}(\mathbf{h}_\theta(x) - \mathbf{y})^2 \tag{5}$$

In the above equation, both $\mathbf{h}_\theta(x)$ and $\mathbf{y}$ are vectors. If we develop the cost function in a non-vector formulation:

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{6}$$

Which basically depicts the goal to minimize the difference between the mapping of all the $\mathbf{m}$ examples $h(x^{(i)})$ and their actual value $y^{(i)}$.
This minimization problem is usually solved using the Gradient Descent algorithm. The mathematical expression of such algorithm is, for every θ parameter on the hypothesis, simultaneously update:

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0, \theta_1) \qquad for\ j = 0\ and\ j = 1 \tag{7}$$

Combining this parameter update with the expression of $J(\theta_0, \theta_1)$ from equation (6), we obtain:

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2\ x_j^{(i)} \tag{8}$$

Being $x_0 = 1$, as it multiplies the intercept θ_0.

[8]Friendly reminder: m is the number of examples user for training the algorithm. For that reason, dividing the total error by m provides the average error for every observation, also known as MSE.

Gradient descent simultaneously updates the value of all θ_j until convergence ($J(\theta_0, \theta_1)$ reaching its minima).

For the sake of simplicity, let's take the case of just one θ parameter. Figure 24 shows the evolution of an example cost function $J(\theta)$ as a function of a unique θ parameter.

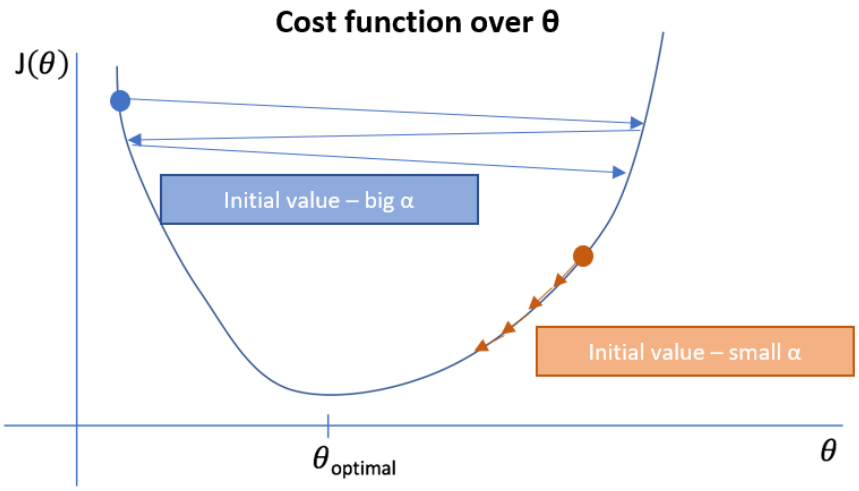


Figure 24: Cost function and learning rate analysis

The model first randomly selects an initial θ in order to minimize the cost function. This θ could be the blue spot or the orange one or basically any point on the $J(\theta)$ curve from Figure 24.

From then, if the partial derivative of the cost function is negative (blue case), θ updates to a higher value (remember negative sign in the equation). On the other hand, if the partial derivative of the cost function for the current theta is positive (orange case), θ updates to a smaller value.

However, note that the partial derivative of the update function is also multiplied by the α parameter. The term α is known as the **learning rate** and determines the speed at which the Gradient Descent converges to its minima (optimal solution).

Learning rate
Back to Figure 24, we can see that a big learning rate (blue case) would make every jump of the θ parameter considerably big, making the cost function go from side to side of its global minima, taking a long time to converge. Moreover, if the initial θ is close to the minima, a really big α could even make the θ parameter run away from the cost function minima.

On the other hand, and despite being a safe option, a small learning rate (orange case), would require many θ updates to reach the cost function minima, making the convergence process very slow.

Like everything in life, the key is in the balance. Normal values for the α parameter fall in the range between 10^{-3} and 1. It's interesting to note, though, that even with a fixed value of α, the steps (or jumps) taken by the Gradient Descent when approaching the global minima are smaller. This is because, when approaching this minima, the slope of the cost function curve (its partial derivative) is progressively smaller.

Finally, the way this convergence algorithm knows that it has reached a decent solution is basically getting to a point of the cost function of $slope = 0$ (null partial derivative). As this scenario (exactly 0) is seldom reached, Gradient Descent usually stops when the step taken is sufficiently small (i.e. 10^{-3}) or after a certain number of iterations.

Multiple Linear Regression
As mentioned before, Simple Linear Regression is an extremely simple ML algorithm. Nevertheless, it does not take full advantage of the wide range of features available for the examples on many datasets. For that reason, Multiple Linear Regression is usually a promising alternative to consider.

So, what are the main differences between SLR and MLR? In MLR, the hypothesis has the form:

$$h_\theta(x) = \theta_0 + \theta_1 \mathbf{x}_1 + \theta_2 \mathbf{x}_2 + ... + \theta_n \mathbf{x}_n \tag{9}$$

Where $\mathbf{x}_1$ corresponds to the value of feature 1 for all the training examples. Basically, there are $\mathbf{n}$ vectors of examples (one for every input feature), each associated to a different θ parameter.

It is obvious that SLR is just a specific case of MLR in which only one feature is considered. For that reason, in a more generic case, the optimization problem of Linear Regression could be written as:

$$min_\theta \;\; J(\theta)$$
$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2 \tag{10}$$

Where the parameter θ represents the vector of all θ parameters $(\theta_0, \theta_1, ..., \theta_n)$.

Again, the Gradient Descent algorithm solves this minimization problem by updating the θ parameters, that is:

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta) \qquad for\ j = 0, 1, ..., n \tag{11}$$

Which again, using equation (6), can be re-written as:

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2\, x_j^{(i)} \tag{12}$$

Although the number of features a MLR can use is considerably high, let's take a look at the visual representation of the cost function for a hypothesis with two θ parameters in Figure 25.

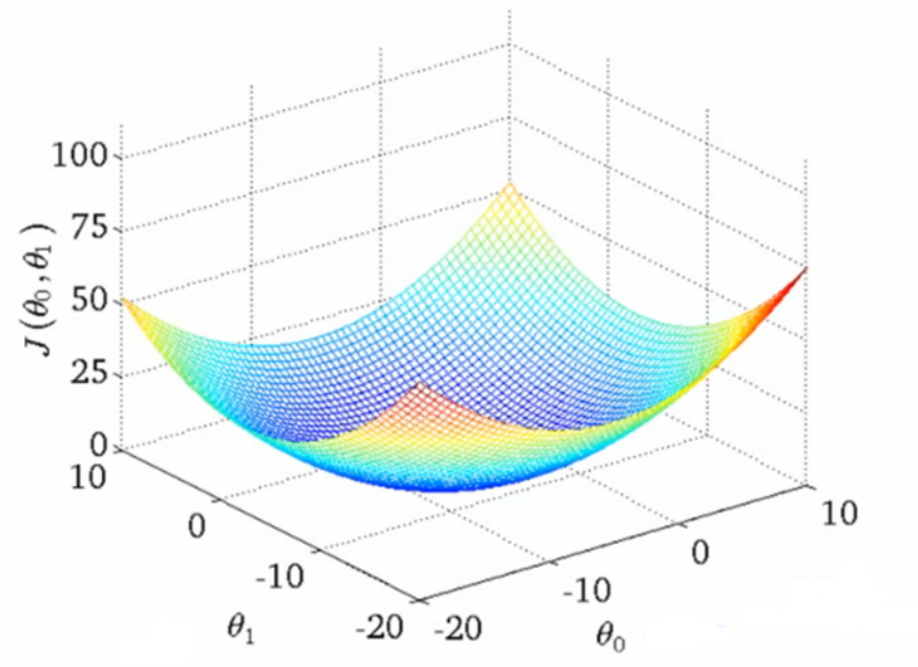


Figure 25: Cost function J(θ) for two θ parameters

Just as in the previous case, the minimization problem is solved using the Gradient Descent algorithm, which, after a couple of iterations and using the appropriate α parameter, is able to find the global minima of the cost function and then uses its corresponding parameters θ_0 and θ_1 to build the linear model.

Nevertheless, not all features are equally weighted. If we were trying to predict the price of a house by using the number of bedrooms (i.e. 1-10) and the squared footage of the house (i.e. 1,000-4,000) as features, the disparity in their range would cause the squared footage feature to have an unfair, much higher weight in the model prediction[9].

[9]On a similar note, a really small-ranged feature ($< 10^{-2}$) such as the concentration of O_2 in the air would be in huge disadvantage against features in the aforementioned ranges

For that reason, **feature scaling** is an important aspect to consider when using several features.

Feature Scaling
Feature Scaling is a technique to constrain and standardize the independent features present in the data to a fixed range, usually (-1, 1). The main reasons behind feature scaling are to avoid the model being unfairly balanced towards high metrics and also to help the Gradient Descent algorithm converge faster.

Mathematically, feature scaling can be expressed as:

$$x_j^{(i)} := \frac{x_j^{(i)} - \mu_j}{max_j - min_j} \tag{13}$$

Which basically means that, for every i_{th} example of the j_{th} feature, its value should be adjusted by subtracting the mean of the feature and divided by the feature range[10].

Another issue to consider when dealing with a large number of features is how to deal with non-numeric features. This is usually done by means of One-hot-encoding.

[10] Although sometimes its variance is used instead.

One-hot-encoding
One-hot-encoding is a technique to express non-numeric ML features as groups of bits where the legal combinations of values are only those with a single high bit and all the others low.

For example, back to the houses data set example, common numeric features would be the number of bedrooms, their square footage and their age, whereas a non-numeric feature requiring one-hot-encoding would be the name of the neighborhood. If the neighborhoods were, for example, Lincoln Park, Lakeview, Chinatown and Ravenswood, the one-hot-encoding would substitute the original feature by four separate ones, one for each neighborhood, with their names as labels. Then, houses from Lincoln Park would show a 1 in the Lincoln Park feature and 0s in the rest of them, and so on.

By means of one-hot-encoding, and despite the binary sequence associated to each tag being usually meaningless, non-numeric features can be used to build ML models.

Polynomial Regression

Until now, we have seen the Linear Regression case for one or multiple features, but always with a model with the expression:

$$h_\theta(x) = \theta_0 + \theta_1 \mathbf{x}_1 + \theta_2 \mathbf{x}_2 + ... + \theta_n \mathbf{x}_n \tag{14}$$

Which, despite elapsing several dimensions, is still a linear expression. The problem is, usually the feature we are trying to predict does not relate linearly to the feature (or features) used as input.

Let's take the example depicted in Figure 26. The training data (in blue) is modeled first with a linear equation (in green), but is obvious that a polynomial model (in red) better fits the data distribution.

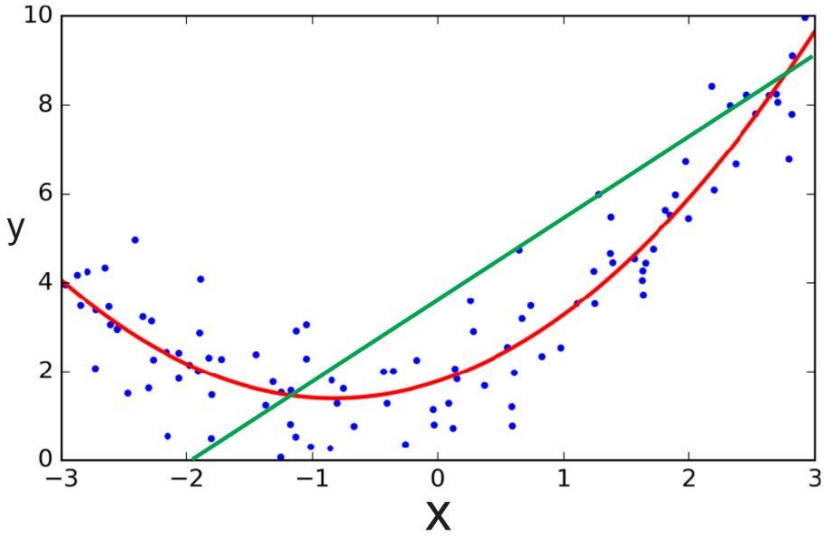


Figure 26: Linear Regression vs Polynomial Regression

Thus, the model of a Multiple Polynomial Regression could have the form[11]:

$$h_\theta(x) = \theta_0 + \theta_1 \mathbf{x}_1^2 + \theta_2 \mathbf{x}_2 + ... + \theta_n \mathbf{x}_n^3 \tag{15}$$

Where some features (i.e. x_2) are still linearly related to the output feature, while others (i.e. x_1, x_n) follow quadratic or cubic relations.

[11] Note there is absolutely no relation between the sub-index of the feature vector and the degree of that feature in the polynomial.

The problem with using a high degree polynomial to model the relation between the output feature and input feature (or features) is that said model could easily incur in overfitting.

Overfitting
Overfitting is a modeling error that occurs when the chosen function is too closely fit to a limited set of data points. Overfitting the model generally takes the form of making an overly complex model to explain idiosyncrasies in the data under study.
When this happens, it is said that the model presents a low bias, but a high variance.
The opposite case would be a model that is too simple to fit the data distribution. That phenomena is known as underfitting, also known as a high bias, low variance model.

Showing overfitting with an example, Figure 27 shows some data distribution (in blue), together with a two[12] θ parameters' linear model (red), a three θ parameters model (blue) and also a 301 θ parameters model (green).

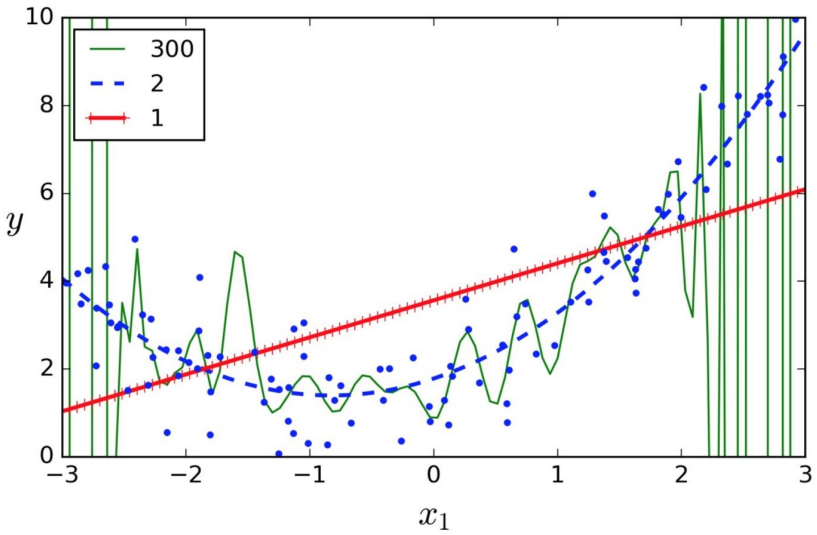


Figure 27: Overfitting example

[12]Note that the degree of the polynomial is always equal to the number of θ parameters used minus one, as θ_0 represents the intercept, and is not assigned to any feature.

It can be seen that, whereas the red model does not perfectly map the data to their correct outputs (underfitting), and the green model is way more complicated than what the data suggests (overfitting).

For that reason, if we checked the accuracy of the green model with the testing set, it would predict an example with $x_1 = -1.5$ should have an output feature $y = 5$, although taking a quick look at the distribution of the data, it is obvious that a more accurate output prediction would be approximately $y = 2$.

In other words, the green model, despite almost perfectly mapping all the training examples with a very small error, would be far from properly mapping the new examples from the test set, consequently providing a huge test set error, which is the one we are trying to reduce.

What can be done to address overfitting?

- Reduce the number of features: Manually select only the most relevant ones
- Use regularization

Regularization
Regularization consists on keeping all the available features, but reducing the magnitude/values of the θ parameters. Regularization works especially well when the data set has a lot of features, each contributing a bit to predict the outcome y.

To regularize the Linear Regression model, we need to ensure that the θ parameters are kept small so that, despite using a high order polynomial as a model, it resembles a low degree one while trying to map the data.

In mathematical terms, this condition is expressed as:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \tag{16}$$

Where λ is known as the regularization parameter. The bigger the lambda, the more the minimization problem tries to keep the θ parameters low.

In Figure 28, we can see two polynomial regressions, both using 5 θ parameters (degree = 5).

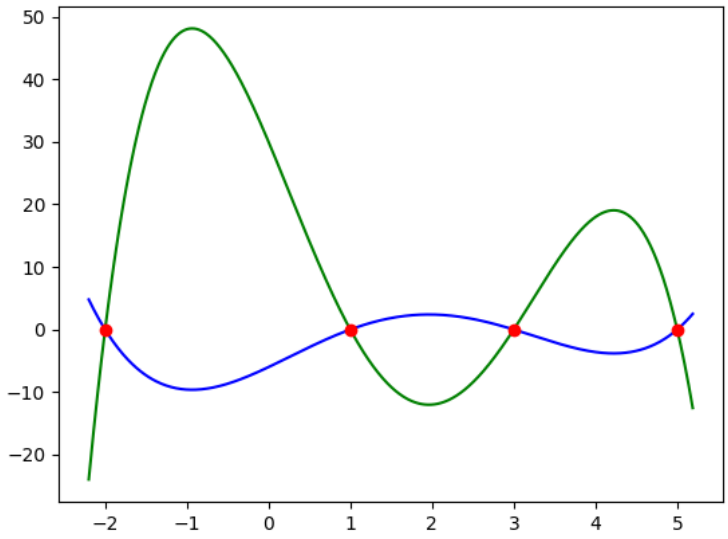


Figure 28: Regularization example

Despite both regression models fitting the training data perfectly, the green one (no regularization) has a much higher variance than the blue one (regularization) and is definitely overfitting the training data. Although this is an over-simplified example, it can be conveyed that the blue regression model is able to better map new testing examples thanks to regularization.

As a summary, the different aspects to consider in Linear Regression are:

- Number of input features n
- Dimensions/degree of the model used
- Learning rate α
- Feature scaling
- One-hot encoding for non-numerical features
- Regularization parameter λ

In Section 5.1, Linear Regression algorithms will be used to predict relevant features of the data set.

4.3 Decision Trees

Decision Trees (DT) are a predictive modeling approach used in statistics and ML. It is one of the most popular ML algorithms due do its simplicity and intuitive approach.
DT use a tree structure as a predictive model to go from observations about an item (in the branches) to the conclusion about that item's target value (in the leaves).
DT where the target variable (output) can take only a discrete set of values (genre, name of the director, etc.) are called Classification Trees. On the other hand, DT where the target variable can take continuous values (gross revenue, IMDB score, movie duration, etc.) are called Regression Trees.
Decision Trees are specially useful when accommodating higher dimension datasets, like the one under study.

In Figure 29, an example of a DT is shown for the case of predicting the *target* output using only the variable *data* as input feature.

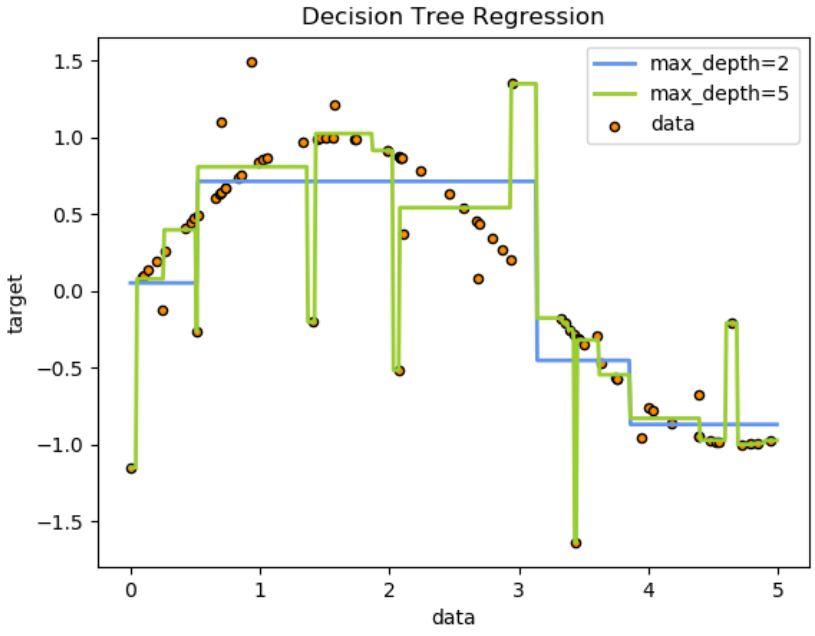


Figure 29: Decision Tree example - data

In the previous Figure, two different Decision Trees were used: one with $depth = 2$ and one with $depth = 5$ (this parameter will be later discussed).

In a more intuitive way, Figure 30 shows the leaves of he Decision Tree of $depth = 2$ from the previous Figure.

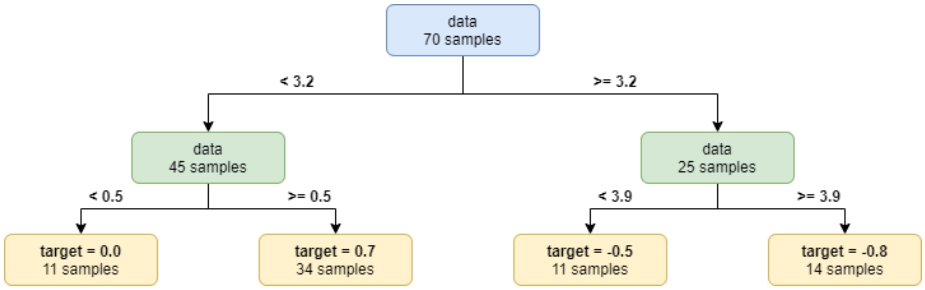


Figure 30: Decision Tree example - thresholds

The functioning of the tree is very simple. Once the depth[13] is fixed to 2, the algorithm distributes the 70 samples in two groups, depending if their value *data* is higher or lower than a certain threshold (data = 3.2 in this case). Later, each of these subgroups is divided into two, again, depending on whether the value of *data* is higher or lower than certain thresholds ($data = 0.5$ and $data = 3.9$ in this case).

Then, all the samples assigned to each subgroup will be predicted an output value equal to the mean of the values from that subgroup. That is, a test set value falling on the 2_{nd} subgroup ($0.5 \leq$ data ≤ 3.9) will be assigned target = 0.7.

However, if we were to analyze the case of $depth = 5$, we would have many more leaves in the tree. But would that be a better solution? The answer is maybe. When building a Decision Tree, several parameters need to be tuned to reach the best result possible.

[13] A DT has $depth + 1$ levels and 2^{depth} leaves.

Depth
Indicates how deep the decision tree can be or, in other words, how many levels are allowed. The deeper the tree, the more splits it has ($splits = depth - 1$) and the more information it captures about the data. When the depth is too high ($depth = 5$ in Figure 29), the model may perfectly predicts the training data (0 error), but then fail to generalize to new examples. As mentioned in the LR model, this is an example of overfitting.

Minimum samples split
Specifies the minimum number of samples required to split an internal node. It can be specified with either an integer value (number of samples) or a decimal one (fraction of samples from training set). In Figure 30, if the minimum samples for split allowed was 30, the split on the right branch would not be allowed, but still the tree would reach its maximum depth allowed through the left branch.

Minimum samples leaf
Denotes the minimum number of samples that a leaf (base of the tree) may have. If, after splitting a node, the leaf node has less samples than the minimum samples leaf number, the splitting is undone (regardless of if it was allowed by the minimum samples split parameter) and the previous node becomes the leaf node. In Figure 30, if the minimum samples per leaf allowed was 15, neither of the second level splits would be undertaken, as they both generate a leaf with 11 samples.

All in all, for a split to be allowed, it needs to be in accordance with the maximum depth, the minimum samples per split and the minimum samples per leaf hyperparameters.

Maximum features
Indicates the maximum number of features to consider when looking for the best split. If specified with an integer value, it denotes the total number of features allowed. If specified by a decimal number, it denotes the fraction of features allowed over the total available. This parameter plays an important role in preventing overfitting. In the example in Figure 30, only one feature is being used ($data$), but more sophisticated Regression Trees models use a higher number of features.

As a summary, the hyper parameters to consider when building a Regression Tree model are mainly:

- max_depth
- min_samples_split
- min_samples_leaf
- max_features

Once all these parameters have been established, what the algorithm does is decide where to place the thresholds. To do that, the algorithm basically finds the one that minimizes the MSE of all the samples to the average of their subgroup. So, in a sense, it could be seen as a LR per parts (although the linear equation is always $y = constant$, where the constant is the average output of the elements of that subgroup).

In section 5, the Regression Tree model will be built and its results will be compared to the ones obtained with all the Linear Regression models.

4.4 Random Forests

Like DT, Random Forests or Random Decision Forests are an ensemble learning method for classification, regression and other tasks. Random Decision Forests combine the simplicity of Decision Trees with flexibility, and also correct for Decision Trees' habit of overfitting the training set, all in all resulting in a vast increase in accuracy. The name Random Forests is indeed a representation of the way they are built:

As mentioned in the previous section, Decision Trees are built using all the examples available for training on the dataset and use all the features provided to the algorithm. Instead, Random Forests start by randomly selecting **m** examples[14] from the dataset, constituting what is called the **bootstrapped dataset**. This random selection occurs with replacement, meaning the same example can be selected twice.

The examples that are not selected for the bootstrapped dataset are usually referred to as **Out-Of-Bag dataset**, and constitute the test set of a Random Forest model.

Later, a Decision Tree is built using only the examples from the bootstrapped dataset as training examples and some of the dataset features, which are randomly selected from the complete list of features. The number of features that are used to build this tree is known as **max_features**, and it is a hyperparameter of RF.

Then, another DT is built using a different subset of features from the complete list. This process is repeated **n_estimators** times, where n_estimators is the number of trees that will be built, and should fulfill:

$$1 \leq n\_estimators \leq P(n, max\_features) \tag{17}$$

Where $P(n, max\_features)$ denotes the permutations of a subset of size $max\_features$ in the set of n features, and can be expressed as:

$$P(n, max\_features) = \frac{n!}{(n - max\_features)!} \tag{18}$$

[14] Size of the training set previously established.

Then, the first example from the test set (or Out-Of-Bag dataset) is passed into every DT that was just built, obtaining $n_{estimators}$ different outputs (either a tag if it's a classification problem or a value if it's a regression problem), on for every tree built.

Finally, the predicted value for the first example of the test set is the average of the output of all the DTs built (or the most outputed label in the case of a classification problem).

This technique of Bootstrapping the data and using the AGGregate to make a decision is called **Bagging**.

As a summary, the hyperparameters to consider when building a RF are:

- max_depth: Like in a DT, defines the depth of the DT that will be built.
- max_features: Maximum number of features to be used in every DT that will be built. Being **n** the total number of features available, a typical start value for max_features is $\sqrt{n}$ and then some values above and below are also studied.
- min_samples_split: Just like in DT models, this parameter defines the minimum number of samples a node needs to have for a split to be allowed.
- min_samples_leaf: Like in the DT model, establishes the minimum number of samples a leaf of the tree may have.
- n_estimators: Maximum number of DTs to be built. The higher, the more precise the algorithm, but also the more computationally expensive.

5 System evaluation and numerical comparison

In this section, the performance of the different algorithms developed will be compared in terms of the accuracy[15] achieved when predicting both the rating of the movie and its revenue. The different algorithms under evaluation are:

- Linear Regression
- Decision Trees
- Random Forests

5.1 Linear Regression

5.1.1 Simple Linear Regression

Simple Linear Regression tries to predict a numeric, continuous feature using just one input feature. Truth be told, this algorithm does not fully take advantage of the wide range of features available for each movie, but is very simple to implement.
We will first try to predict the movies' gross income and then their IMDB score, using the most appropriate input feature on each case.

Predict Gross income
To predict the gross income of a movie, we should find out what feature presents the highest correlation with it. Taking a look at the correlation matrix on Figure 13 we can see the most correlated feature is num_voted_users[16] with a coefficient of 0.637 (see numerical correlation matrix on Annex 1).
The correlation between these two features implies that the greater engagement (could be extrapolated to acceptance) generated by a movie, the more money it generates. This relation makes sense, as the more people watching a movie, the more money generated by it and also the higher the number of reviews for that movie will be.

[15] The discussion on the metrics used can be found on Annex 2.
[16] Just as a reminder, this feature shows the number of users that gave a rating to the movie on the IMDB platform.

In Figure 31, the training set (in blue) has been used to come up with a LR model (in green), using just two θ parameters (polynomial of 1_{st} degree).

The green points from the LR model show where the training points would be matched if they were used in the test set. For example, the example with the highest number of voting users (more than 1.5M) is perfectly represented by the model, whereas examples around 1M votes fall far from the model.

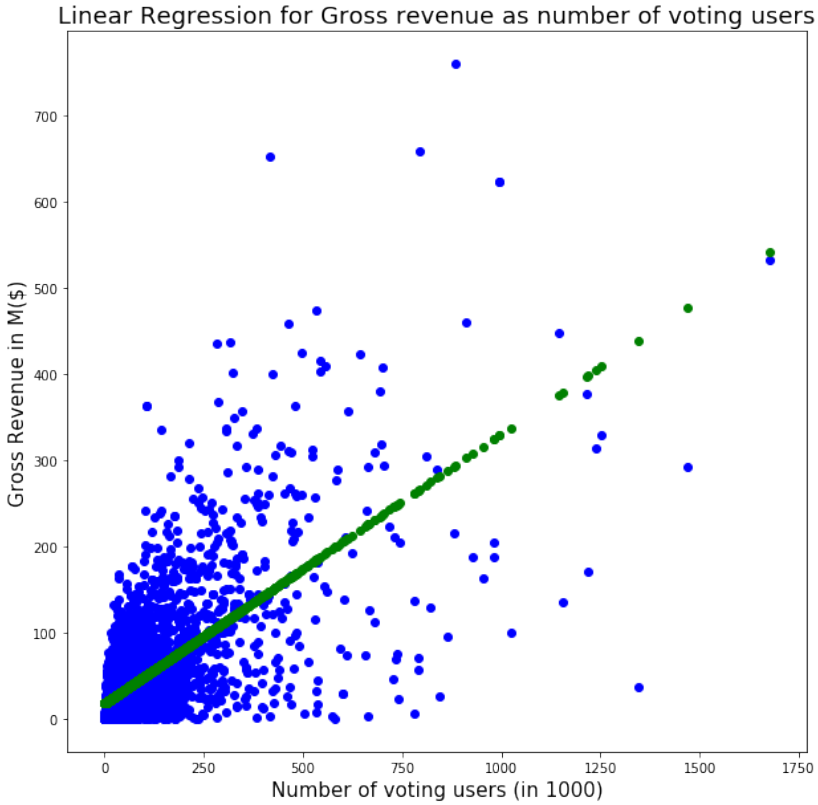


Figure 31: Linear Regression to predict Gross revenue - Training set

From this plot it can already be seen that most of the data is pretty scattered and does not exactly follow a certain distribution. For that reason, we can expect the test error to be considerably high.

In Figure 32, the built model (in green) mapping the test examples predictions is shown. However, the red points constitute our actual test set with their real mapping value. Although the mapping is decent for small values, some examples with a high number of voting users fall extremely far from the built model, especially the example with 1340 votes.

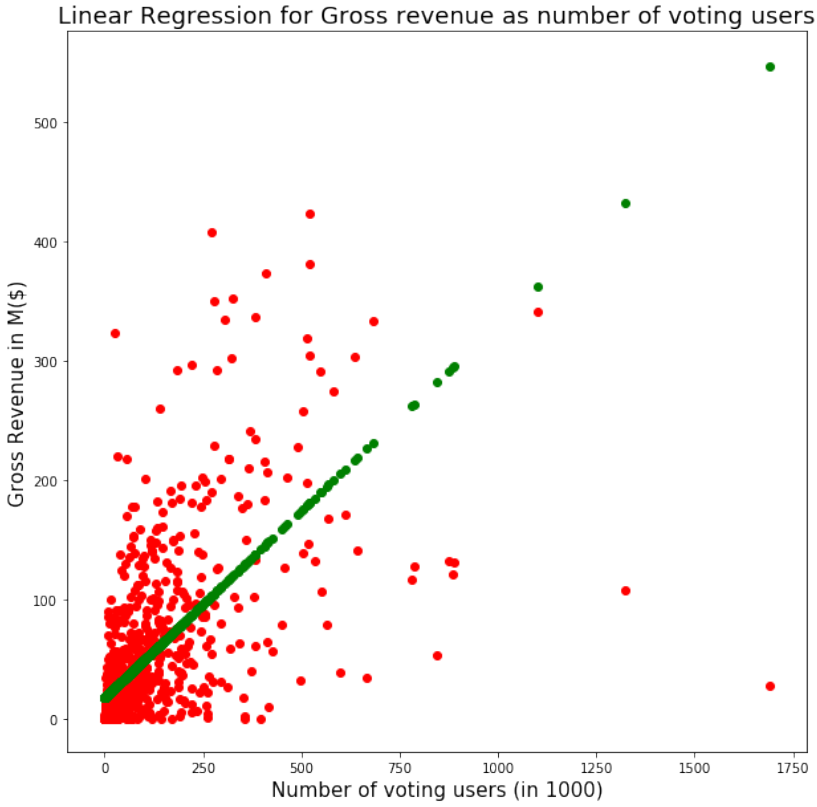


Figure 32: Linear Regression to predict Gross revenue - Test set

The indicative metrics[17] for this simple LR model predicting the gross revenue[18] are:

Mean Absolute Error (MAE):	35.489
Mean Squared Error (MSE):	3330.051
Root Mean Squared Error (RMSE):	57.707
R^2:	0.278

[17] Later, more robust metrics will be calculated by means of several realizations for all algorithms and they will be compared in a visual format.

[18] These metrics are obtained using a single realization, achieved using the first 80% of the examples from the sorted dataset as training and the last 20% as test set.

Predict IMDB score

To predict the IMDB rating a movie can achieve, we should find out what feature presents the highest correlation with it. Taking a look at the correlation matrix on Figure 13 we can see the most correlated feature is again num_voted_users, this time with a coefficient[19] of 0.411.

In Figure 33, the distribution of the training data (in blue) is shown, together with the resulting LR model (in green), again with just two θ parameters (1_{st} degree polynomial).

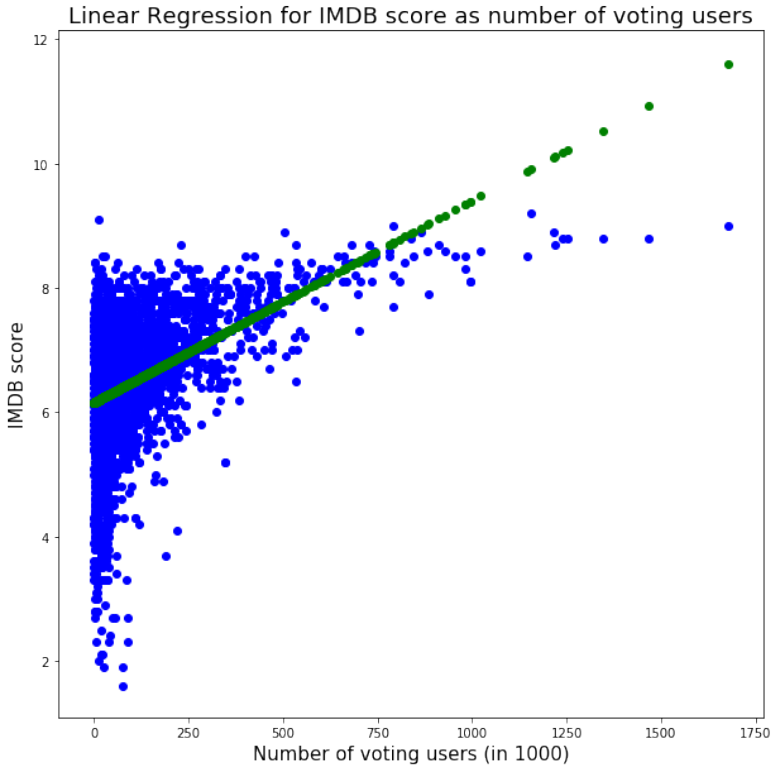


Figure 33: Linear Regression to predict IMDB score - Training set

In this case, the data is extremely scattered[20]. for low number of voting users (ratings from 1.6 to 9.5), whereas for high number of voters (more than 750,000), scores are between 8 and 9.5.

[19] See numerical correlation matrix on Annex 1.

[20] Moreover, this model even predicts scores higher than 10 for movies with more than 1.2M voters, another proof that 100% of the time, ML algorithms have no idea what they are actually doing.

In Figure 34, the regression model (in green) mapping the test set examples is shown together with the actual testing set and their mapping (in red). In this case, both low voted movies and highly voted movies are poorly predicted, whereas the score for movies with an average number of voters (0.5-1M) is predicted reasonably well.

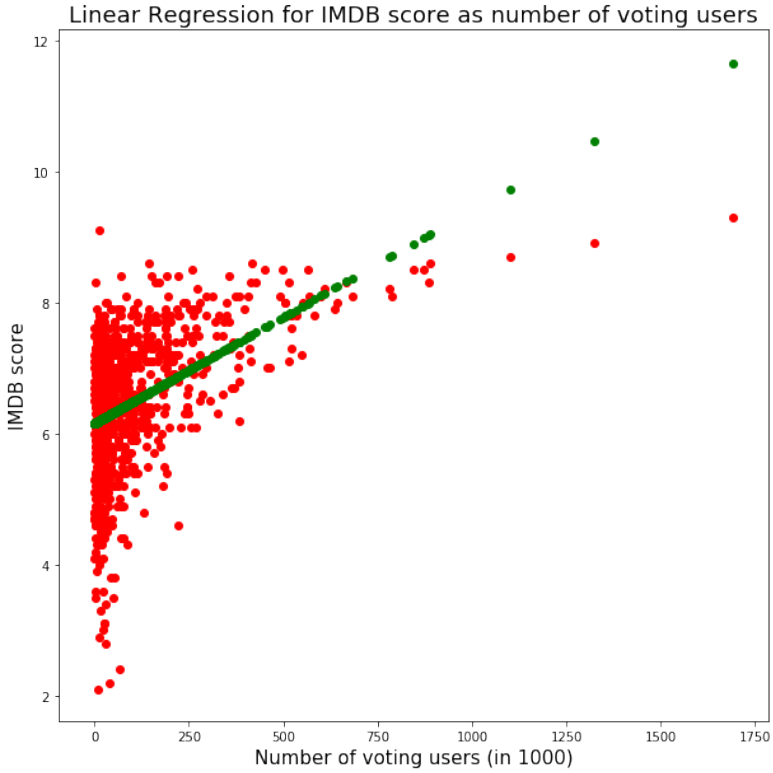


Figure 34: Linear Regression to predict IMDB score - Test set

The indicative metrics for this simple LR model predicting the IMDB score are:

Mean Absolute Error (MAE):	0.733
Mean Squared Error (MSE):	0.889
Root Mean Squared Error (RMSE):	0.943
R^2:	0.226

5.1.2 Polynomial Regression

It has been shown that a simple LR does not fully represent the complex data distribution, either for gross revenue or IMDB score. For that reason, we will now try to model these features (again using the number of voting users as input) by means of Polynomial Regression (PR).

Predict Gross revenue

In Figure 35, the distribution of the training data (in blue) is shown, together with the resulting PR model (in green), this time with three θ parameters (2_{nd} degree polynomial).

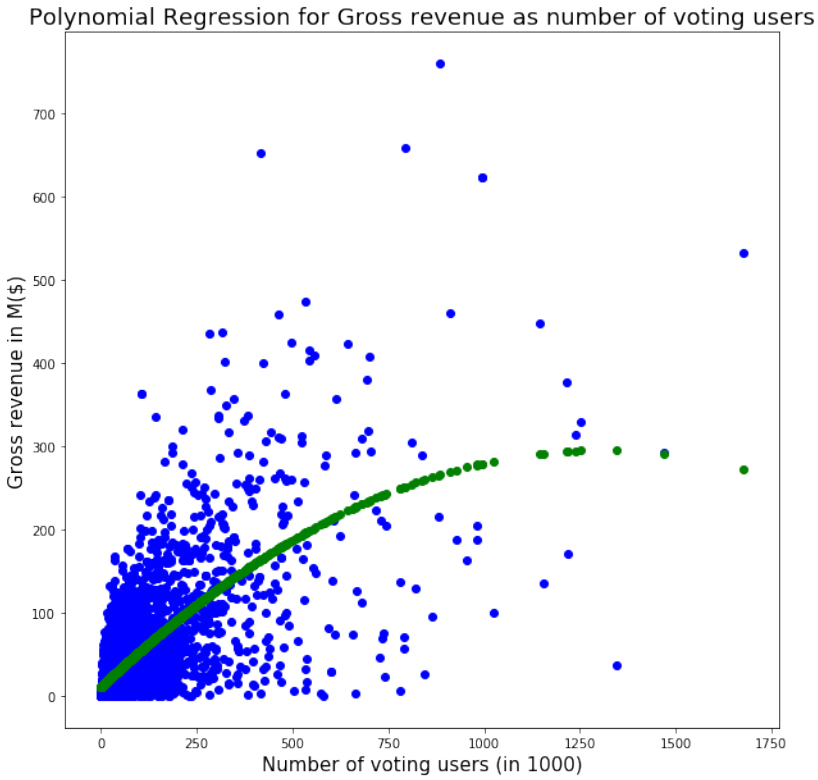


Figure 35: Polynomial Regression (2_{nd} degree) to predict Gross revenue - Training set

In this case, the model predicts reasonably well the gross revenue for low-voted movies (considering the scattered data), but does not generalize well for highly voted movies, again due to the extremely scattered data available.

In Figure 36, the regression model (in green) mapping the test set examples is shown together with the actual testing set and their mapping (in red). In this case, the gross revenue for movies with a reduced number of voters is reasonably well predicted, but the model starts to considerably diverge from the actual output for more than 250,000 voters due to the highly scattered data.

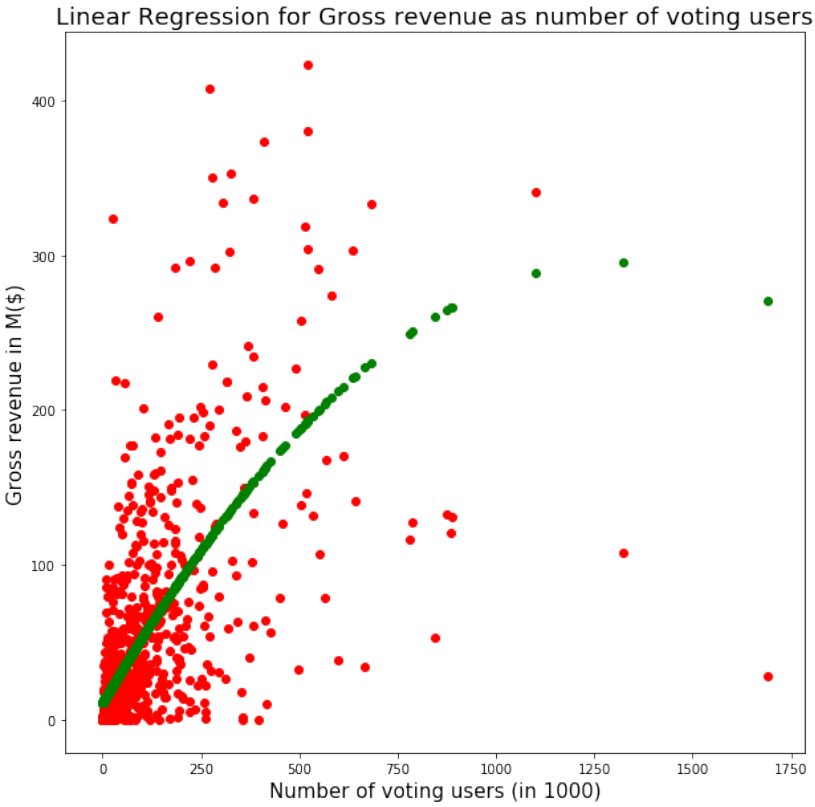


Figure 36: Polynomial Regression (2_{nd} degree) to predict Gross revenue - Test set

The indicative metrics for this PR model predicting the gross revenue are:

Mean Absolute Error (MAE):	33.810
Mean Squared Error (MSE):	2877.047
Root Mean Squared Error (RMSE):	53.638
R^2:	0.376

From these metrics it can be seen that all four indicators show a much better performance for the polynomial of 2_{nd} degree with respect to the simple LR case. This shows that a simple linear equation was a too simple model to explain the data distribution we have.

In Figure 37, the test data is shown (in red) together with a PR model using six θ parameters (polynomial of 5_{th} degree).

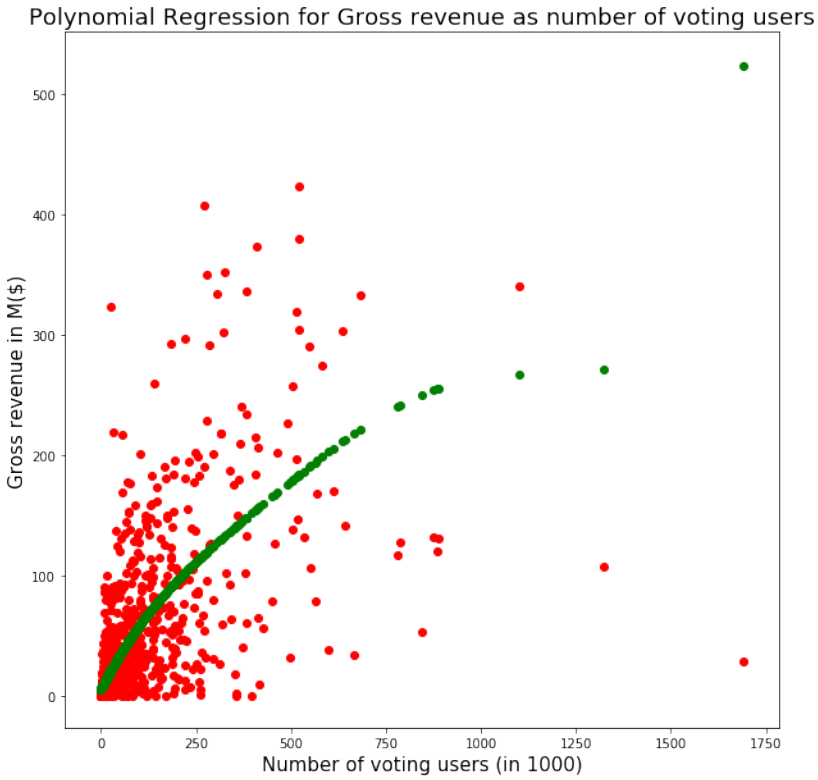


Figure 37: Polynomial Regression (5_{th} degree) to predict Gross revenue - Test set

The indicative metrics for this PR5 model predicting the Gross revenue are:

Mean Absolute Error (MAE):	33.593
Mean Squared Error (MSE):	3061.483
Root Mean Squared Error (RMSE):	55.331
R^2:	0.336

Taking a look at the metrics, we can see that all four of them are either equal or worse than those achieved with the polynomial of 2_{nd} degree. This means using a polynomial of 5_{th} degree overfits the data.

Predict IMDB score

In Figure 38, the distribution of the training data (in blue) is shown, together with the resulting PR model (in green), this time with three θ parameters (second degree polynomial).

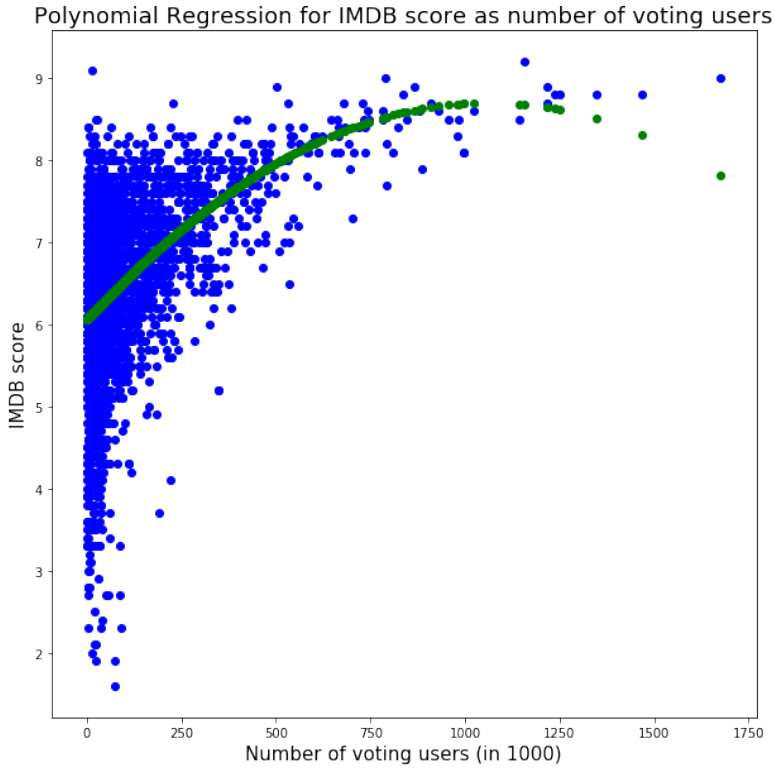


Figure 38: Polynomial Regression (2_{nd} degree) to predict IMDB score - Training set

In this case, the IMDB score for highly voted movies can be accurately predicted (with the exception of the most voted movie), whereas low voted movies are poorly predicted due to the highly scattered data.

In Figure 39, the regression model (in green) mapping the test set examples is shown together with the actual testing set and their mapping (in red). As we could expect, movies with low number of voters are not accurately predicted, but from 500,000 voters onward, the model has a low error.

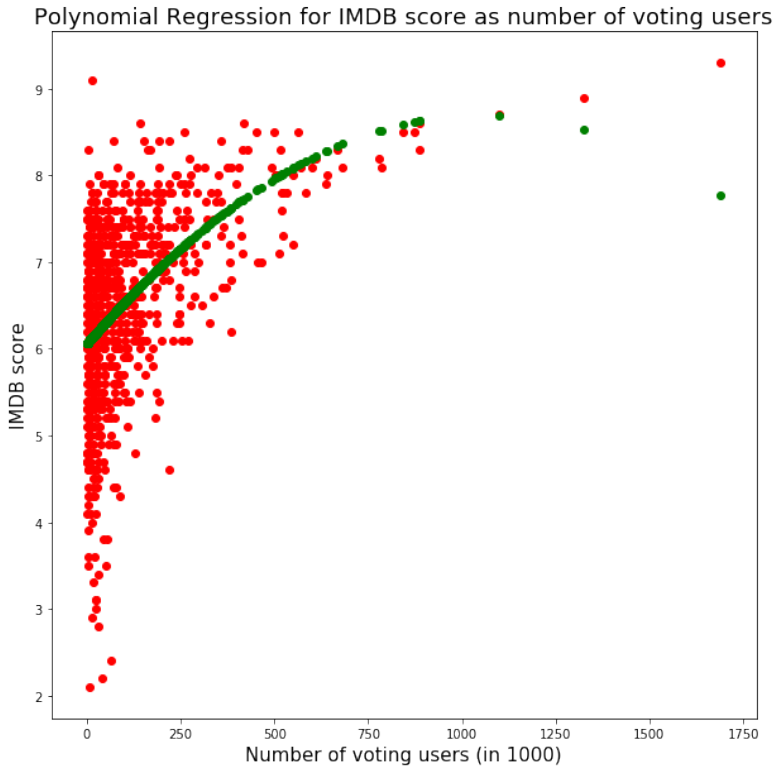


Figure 39: Polynomial Regression (2_{nd} degree) to predict IMDB score - Test set

The indicative metrics for this PR2 model predicting the IMDB score are:

Mean Absolute Error (MAE):	0.719
Mean Squared Error (MSE):	0.860
Root Mean Squared Error (RMSE):	0.928
R^2:	0.251

These numbers show that all four metrics are better for a 2_{nd} degree polynomial than for a simple LR equation. Again, this conveys that the data idiosyncrasies cannot be explained with a simple linear equation.

In Figure 40, the test data is shown (in red) together with a PR model using six θ parameters (polynomial of 5_{th} degree).

Figure 40: Polynomial Regression (5_{th} degree) to predict IMDB score - Test set

The indicative metrics for this PR5 model predicting the IMDB score are:

Mean Absolute Error (MAE):	0.713
Mean Squared Error (MSE):	0.851
Root Mean Squared Error (RMSE):	0.922
R^2:	0.259

Taking a look at the metrics, we can see that using a polynomial of 5_{th} degree slightly improves all four metrics with respect to the polynomial of 2_{nd} degree. Although using this high degree polynomial does not cause overfitting, we should ask ourselves whether or not is worth to increase the degree of the polynomial to achieve such a minor improvement.

5.1.3 Multiple Regression

As mentioned before, building a successful Machine Learning model to fit the data with a simple LR is a risky task, even when using a high degree polynomial. Luckily, this data set has a wide range of features full of information ready to be exploited.

We already discussed that, for both the gross revenue and the IMDB score of our movies, the most correlated feature was the number of users that voted the movie on the IMDB website, or num_voted_users (0.64 and 0.41 correlation coefficients with gross revenue and IMDB score, respectively). However, there are other numeric features with a pretty decent correlation[21] with these features, like:

- num_user_for_reviews (0.56 and 0.29 correlation coefficients, respectively)
- num_critic_for_reviews (0.48 and 0.31 idem)
- movie_facebook_likes (0.38 and 0.25 idem)

Similarly, there are some non-numeric features that may possess hidden information and could be key to improve our model. In this case, we will use:

- color
- director_name
- genres
- actor_1_name
- plot_keywords
- country
- content_rating

This non-numerical features will be hot-encoded to display information in a numeric way and therefore be used together with the previous features to come up with a more accurate model.

[21] Right next to each feature, their correlation with the gross revenue and the IMDB score are provided, in that order.

Although there is no way to display the graphical results for the Multiple Regression model due to the fact that each feature would correspond to a different dimension, the same metrics that have been used for the previous models are provided below.

Multiple Regression with the aforementioned 11 features[22] and a polynomial of 1_{st} degree will be used used.

Predict Gross revenue
The indicative metrics for this MR model predicting the Gross score are:

Mean Absolute Error (MAE):	35.794
Mean Squared Error (MSE):	2966.711
Root Mean Squared Error (RMSE):	54.468
R^2:	0.454

All four aforementioned metrics are either equal or better than the ones obtained on the simple Linear Regression model. However, all four metrics are either equal or worse than the ones obtained on the Polynomial Regression of 2_{nd} degree.

Predict IMDB score
The indicative metrics for this MR model predicting the IMDB score are:

Mean Absolute Error (MAE):	0.698
Mean Squared Error (MSE):	0.885
Root Mean Squared Error (RMSE):	0.941
R^2:	0.223

Comparing these metrics with the ones obtained in the simple LR case, all four of them are pretty close. Similarly, when comparing them to the ones obtained through the PR of 2_{nd} degree, no significant differences can be observed. However, on further results analysis, more realizations will be taken into consideration, thus being the error metrics more representative.

[22]The original, most correlated one (num_voted_users) and the complementary 10 discussed.

5.1.4 Regression Results

In this section, the aforementioned regression models will be compared in terms of R^2 and RMSE. Using one table for each metric[23], the models from left to right correspond to:

- Linear Regression (LR)
- Polynomial Regression of 2_{nd} degree (PR2)
- Polynomial Regression of 5_{th} degree (PR5)
- Multivariate Linear Regression (MLR)

R^2	LR	PR2	PR5	MLR
Gross revenue	0.403	0.443	0.432	0.451
IMDB score	0.207	0.226	0.230	0.248

$RMSE$	LR	PR2	PR5	MLR
Gross revenue	53.23	51.50	52.03	50.46
IMDB score	0.935	0.923	0.921	0.911

[23] All results are obtained averaging 25 realizations. Every realization comes from a different shuffling of the original data to randomly distribute training and testing examples on every iteration.

In a more visual way, Figure 41 shows the value of R^2 for the four aforementioned models trying to predict gross revenue.

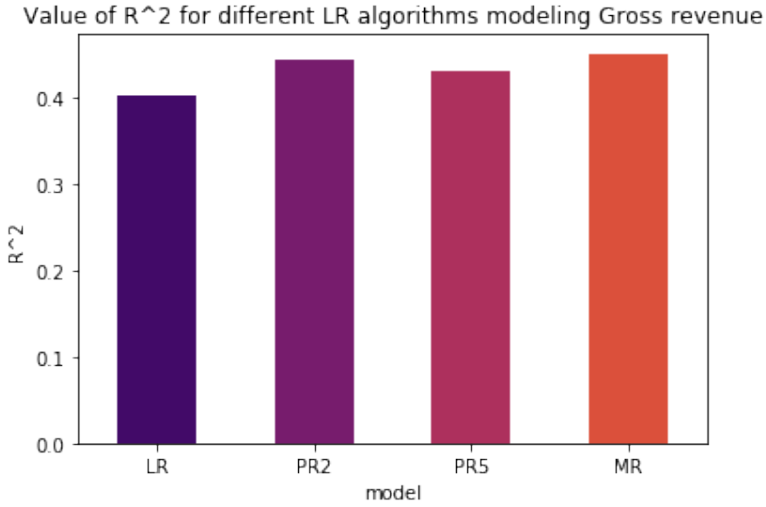


Figure 41: R^2 results for Gross revenue (LR)

The graphic shows that a Polynomial Regression of 2_{nd} degree ($R^2 = 0.443$) better models the data than a simple Linear Regression model (0.403). However, using a Polynomial Regression model of 5_{th} degree (0.431) is a too complicated model for such data distribution, consequently overfitting the training data. Finally, Multiple Regression presents the highest R^2 coefficient (0.451), meaning it better represents changes in the output feature as a result of input features changes or, in other words, better fits the data.

Similarly, Figure 42 shows the value of RMSE for the same four models when predicting Gross revenue.

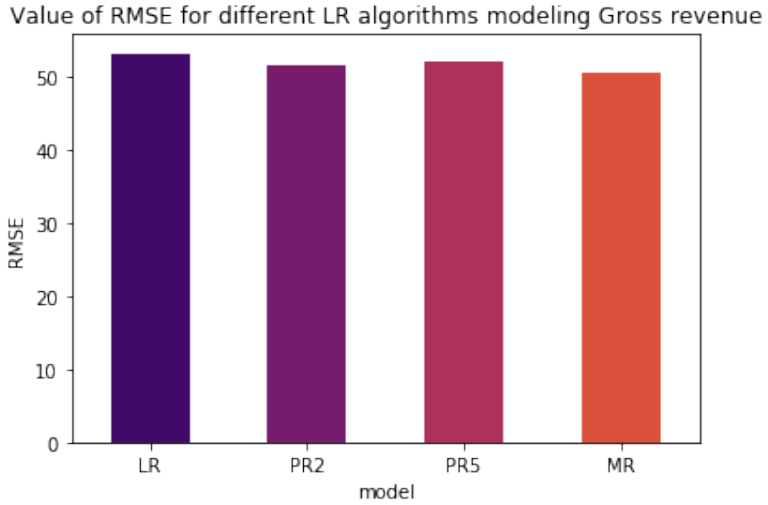


Figure 42: RMSE results for Gross revenue (LR)

In this scenario, we can also observe the aforementioned tendency. Simple Linear Regression (RMSE = 53.230) presents a high error. The Polynomial Regression of 2_{nd} degree has a lower error (51.50), but the Polynomial Regression of 5_{th} degree (52.034) overfits the data. Finally, the Multiple Regression model presents the lowest RMSE (50.458) and is, therefore, the most suitable model for this data distribution when it comes to predicting the gross revenue of new films.

On the other hand, Figure 43 shows the value of R^2 for the four models when predicting the IMDB score.

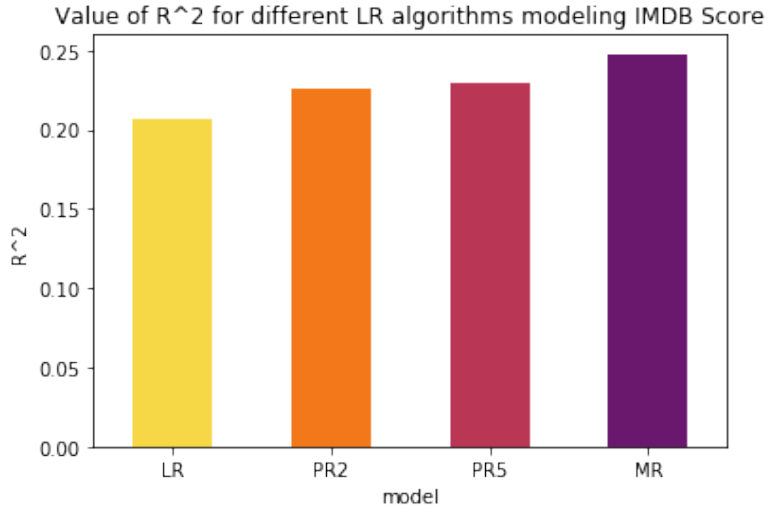


Figure 43: R^2 results for IMDB score (LR)

This graph shows a strong dependency between the complexity of the model and the value of R^2. The Simple Linear Regression algorithm (R^2 = 0.207) models the data slightly worse than the Polynomial Regression of 2_{nd} degree (0.226). The Polynomial Regression of 5_{th} degree (0.23) is a little bit better then the 2_{nd} degree one, what means there is no overfitting in this case[24]. Finally, the Multiple Regression model presents the highest R^2 value (0.248).

[24]However, it could be argued that complicating the model from a 2_{nd} to a 5_{th} degree Polynomial Regression to achieve such a small improvement on the R^2 is not worth it.

Similarly, Figure 44 shows the value of RMSE for the same four models when predicting the IMDB score.

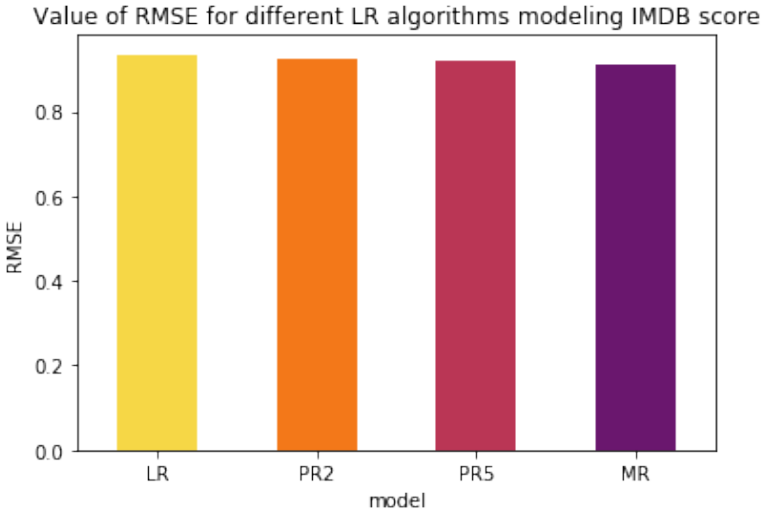


Figure 44: RMSE results for IMDB score (LR)

This graph also shows a strong dependency between the complexity of the model and the error associated to it. More complicated models present a lower RMSE. The Simple Linear Regression algorithm (RMSE = 0.935) has a higer error then the 2_{nd} order Polynomial Regression (0.923). Then the 5_{th} order Polynomial degree presents a slightly lower error[25] (0.921). Finally, the Multiple Regression algorithm obtains the lowest error (0.248).

[25] Again, it could be discussed whether or not such a minor improvement is worth the increase in complexity.

All in all, the R^2 coefficients for the models predicting the IMDB score are much lower than those obtained when predicting the gross revenue. This information, together with the constant growing of the metric in most sophisticated models tells us that the data can give us even more information. In other words, there is no overfitting and more complicated models could be used to achieve a better performance.

On the other hand, the RMSE obtained on the two problems cannot be compared in such a simple way, as the gross revenue case ($162-$760,000,000) deals with way higher numbers than the IMDB score does (1.6-9.5). However, it is interesting to note that, the constant decrease in the RMSE metric seen in the IMDB score case does not show in the gross revenue one. Being more specific, using a 5_{th} degree PR is better than using a 2_{nd} degree one when predicting the IMDB score, but not when guessing the gross revenue.

5.2 Decision Trees

5.2.1 Simple Regression Tree

Simple Regression Trees use only one feature to predict the output variable. As mentioned in previous sections, the most correlated feature with both the gross revenue and the IMDB score is the number of voting users. For that reason, this is the feature that will be used in this first one-feature-only model.

Predict gross Revenue[26]

Mean Absolute Error (MAE):	32.724
Mean Squared Error (MSE):	2706.906
Root Mean Squared Error (RMSE):	52.028
R^2:	0.413

Despite this being the most simple Regression Tree model, all metrics (except for R^2) are better than the ones obtained in the best performing Linear Regression model (Multiple Regression: 35.794, 2966.711, 54.468 and 0.454 respectively) and way better than those obtained on the Simple Linear Regression model (35.489, 3330.051, 57.707 and 0.278 respectively). This means that the Regression Tree model using just one feature is better than the Multiple Regression using 11 features.

Predict IMDB score[27]

Mean Absolute Error (MAE):	0.795
Mean Squared Error (MSE):	1.111
Root Mean Squared Error (RMSE):	1.054
R^2:	0.189

In this case, all four metrics are worse than the ones obtained in the case of Simple Linear Regression (0.733, 0.889, 0.943 and 0.226 respectively) and, of course, worse than those from the Multiple Regression (0.698, 0.885, 0.941, 0.223 respectively), but when analyzing them using more realizations, different results may appear.

[26] Hyperparameters for SDT predicting Gross revenue are: max_depth = 6, minimum_samples_leaf = 0.02

[27] Hyperparameters for SDT predicting IMDB score are: max_depth = 5, minimum_samples_leaf = 0.01

The question here is if, despite the most simple DT model being worse than the most basic LR model, whether or not its 11-features version offers a better performance. In other words, whether or not the 11-features Multiple Regression Tree is better than the 11-features Multiple Linear Regression model or not.

5.2.2 Multiple Regression Tree

Multiple Regression Trees use several features to predict the output variable. Just like in the Multiple Linear Regression section, the features used will be:

- num_voted_users (0.64 and 0.41 correlation coefficients with gross revenue and IMDB score)
- num_user_for_reviews (0.56 and 0.29 idem)
- num_critic_for_reviews (0.48 and 0.31 idem)
- movie_facebook_likes (0.38 and 0.25 idem)

Moreover, the same non-numeric features will be used, which were:

- color
- director_name
- genres
- actor_1_name
- plot_keywords
- country
- content_rating

Using all these features, the metrics obtained with the Multiple Linear Regression Tree model are now:

Predict gross Revenue[28]

Mean Absolute Error (MAE):	28.098
Mean Squared Error (MSE):	2276.831
Root Mean Squared Error (RMSE):	47.716
R^2 :	0.581

In this case of Multiple Regression Tree, all four metrics are better than the ones obtained in the case of Multiple Regression (35.794, 2966.711, 54.468 and 0.454 respectively). This answers the question previously formulated: Despite the Regression Tree's most basic model being worse than the LR's most simple one, DT offer a higher degree of complexity through its hyperparameters tuning that provides a better performance in this particular dataset.

Predict IMDB score[29]

Mean Absolute Error (MAE):	0.686
Mean Squared Error (MSE):	0.828
Root Mean Squared Error (RMSE):	0.910
R^2 :	0.273

Again, the Multiple Linear Regression Tree used in this section gives better results than the Multiple Regression model in previous sections (0.698, 0.885, 0.941 and 0.223 respectively), although the improvement is not as remarkable as the one observed when predicting the gross revenue.

[28]Hyperparameters for MDT predicting Gross revenue are: max_depth = 8, minimum_samples_leaf = 0.02, max_features = 9174

[29]Hyperparameters for MDT predicting IMDB score are: max_depth = 10, minimum_samples_leaf = 0.02, max_features = 9174

5.2.3 Decision Tree results

In this section, the aforementioned DT models will be compared in terms of R^2 and RMSE. Using one table for each metric[30], the models from left to right correspond to:

- Simple Decision Tree
- Multiple Decision Tree

R^2	SDT	MDT
Gross revenue	0.433	0.564
IMDB score	0.214	0.283

RMSE	SDT	MDT
Gross revenue	51.906	44.741
IMDB score	0.925	0.901

[30] Again all results are obtained averaging 25 realizations. Every realization comes from a different shuffling of the original data to randomly distribute training and testing examples on every iteration.

In a more graphical way, Figure 45 shows the value of R^2 for both the Simple Decision Tree and the Multiple Decision Tree when predicting the gross revenue of new movies.

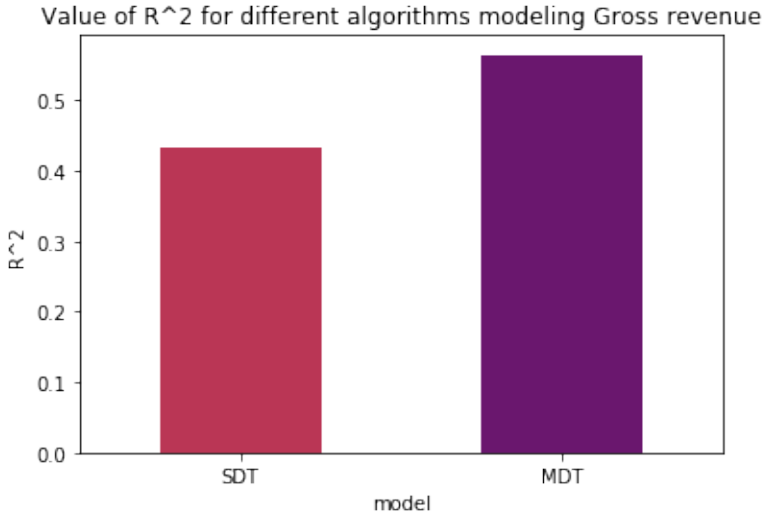


Figure 45: R^2 results for Gross revenue (DT)

As mentioned before, the Multiple Decision Tree (purple) offers a considerably better modeling of the data than the Simple Regression Tree (garnet), as their R^2 value shows (0.564 vs 0.433 respectively).

Later in this work, these two values will be compared with those obtained in all the LR models as well as in the Random Forest model.

In Figure 46, the RMSE value for the SDT and MDT models when predicting the gross revenue are shown.

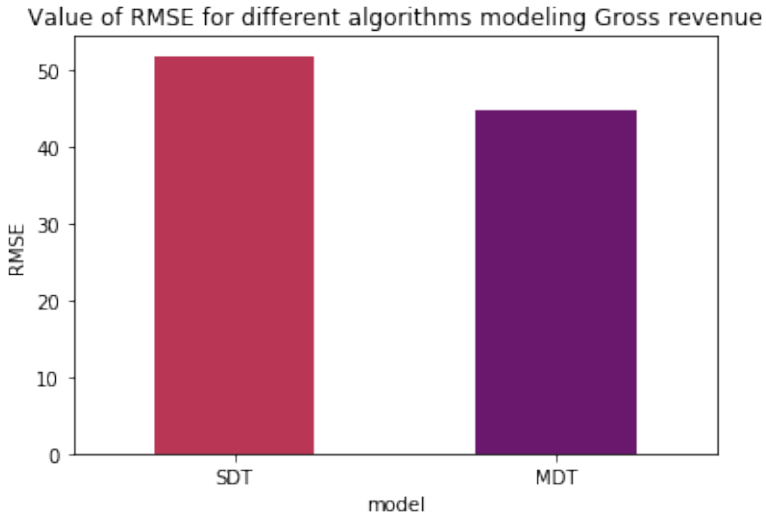


Figure 46: RMSE results for Gross revenue (DT)

Again, MDT (purple) presents a lower error (44.741) than SDT (garnet, 51.906), although the difference is not as significant as in the R^2 metric.

In Figure 47, the value of R^2 is shown for SDT and MDT when predicting the IMDB score for new movies.

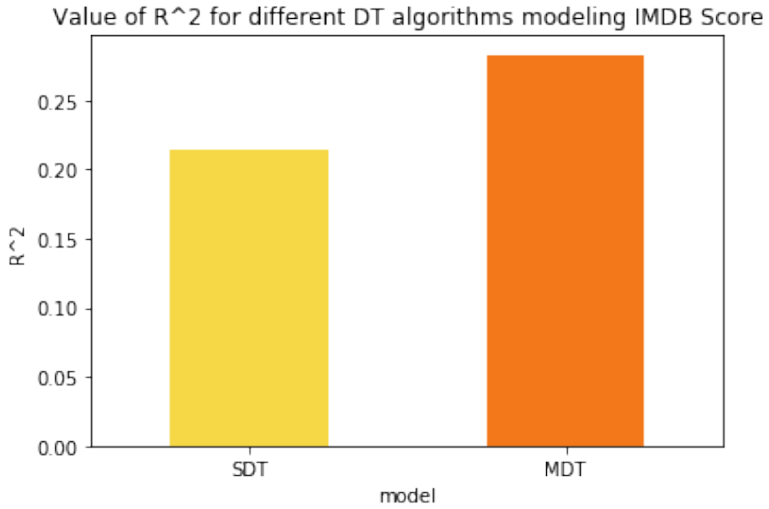


Figure 47: R^2 results for IMDB Score (DT)

As expected, the R^2 value for SDT (yellow) is very low (0.214) as compared to the MDT value (orange, 0.283), which is not very high either.

Finally, Figure 48 shows the RMSE obtained through SDT and MDT when predicting IMDB score.

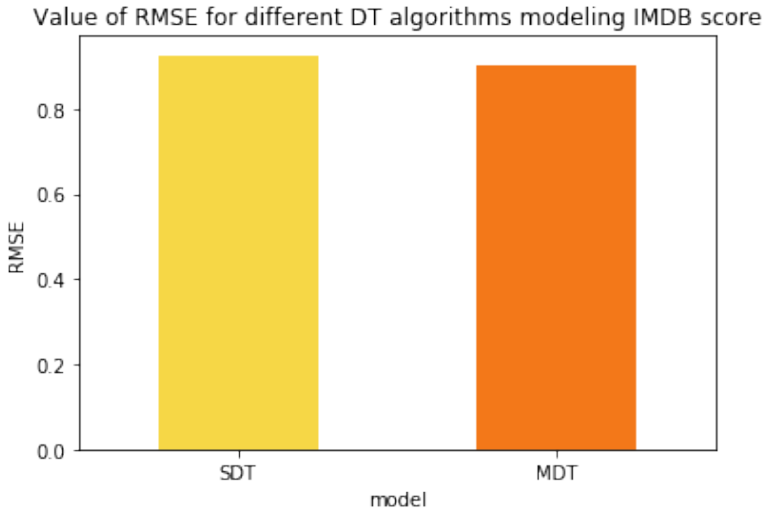


Figure 48: RMSE results for IMDB Score (DT)

As expected, the RMSE for the SDT model (yellow) is slightly higher (0.925) than the error obtained through the MDT model (orange, 0.901), although the difference in this case is small enough not to be considered a 'better' model per se[31].

[31] Using a bigger dataset or even just some more iterations (and consequently different training-testing distributions) could make the difference even smaller or even reverse it.

5.3 Random Forests

Using all these features, the indicative metrics obtained with the Random Forests model are now:

Predict gross Revenue[32]

Mean Absolute Error (MAE):	27.237
Mean Squared Error (MSE):	2143.827
Root Mean Squared Error (RMSE):	46.301
R^2 :	0.605

When comparing these metrics with the ones achieved when using the DT model, it can be observed that all four metrics are better than the ones obtained with MDT. Needless to say, all metrics are way better than the LR ones.

Predict IMDB score[33]

Mean Absolute Error (MAE):	0.640
Mean Squared Error (MSE):	0.729
Root Mean Squared Error (RMSE):	0.854
R^2 :	0.360

Regarding the prediction of the IMDB score for new movies, again all four metrics are better than the ones obtained when using the MDT model and, of course, any other model we have studied so far.

After analyzing the model's metrics using 25 realizations, the results are:

R^2	RF
Gross revenue	0.611
IMDB score	0.335

RMSE	RF
Gross revenue	42.541
IMDB score	0.859

[32]Hyperparameters for RF predicting gross revenue are: max_depth = 10, minimum_samples_leaf = 0.01, n_estimators = 200

[33]Hyperparameters for RF predicting IMDB score are: max_depth = 100, minimum_samples_leaf = 0.01, n_estimators = 75

5.4 Overall results

In this section, the performance of all the models under study will be compared in a visual way. On the below tables, the R^2 and the RMSE value for all the algorithms is shown when predicting gross revenue and IMDB score:

R^2	LR	PR2	PR5	MLR	SDT	MDT	RF
Gross revenue	0.403	0.443	0.432	0.451	0.433	0.564	0.611
IMDB score	0.207	0.226	0.230	0.248	0.214	0.283	0.335

$RMSE$	LR	PR2	PR5	MLR	SDT	MDT	RF
Gross revenue	53.23	51.50	52.03	50.46	51.906	44.741	42.541
IMDB score	0.935	0.923	0.921	0.911	0.925	0.901	0.859

In Figure 49, the value of R^2 in the case of predicting the gross revenue is shown for all the algorithms.

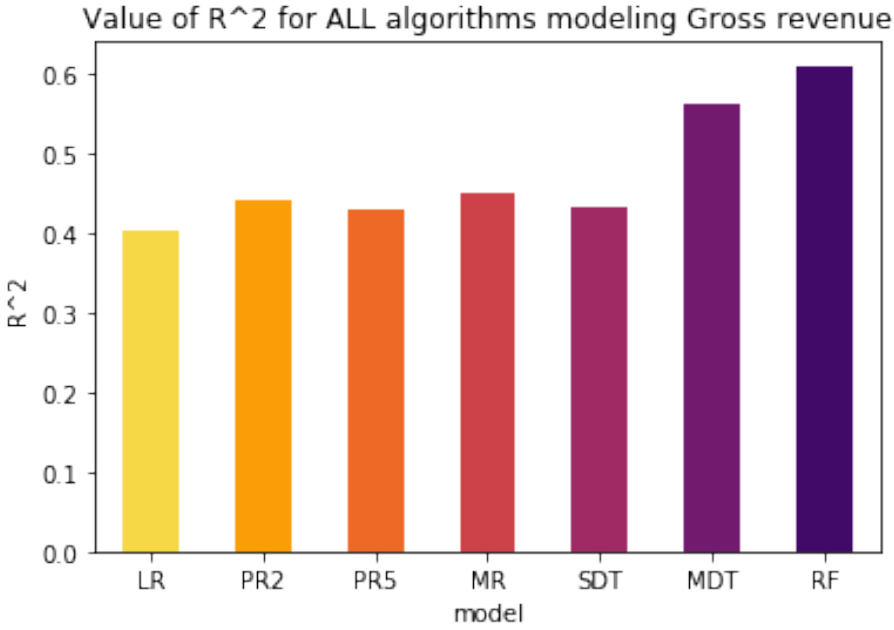


Figure 49: R^2 results for Gross revenue (all algorithms)

This graphic shows the tendency that has already been mentioned before in this work. Linear Regression models, although being fast and easy to implement, cannot offer the same modeling of the data as other more sophisticated algorithms do. The best performing LR model (Multiple Regression) presents a value of $R^2 = 0.451$, whereas the Random Forest model reaches a much higher $R^2 = 0.611$.

Nevertheless, the biggest jump is observed between the Simple Decision Tree ($R^2 = 0.433$) and the Multiple Decision Tree ($R^2 = 0.564$), conveying that, in the case of this dataset, using a high number of features allows for a much more precise mapping of new data, especially under the right model.

In Figure 50, the value of RMSE in the case of predicting the gross revenue is shown for all the algorithms.

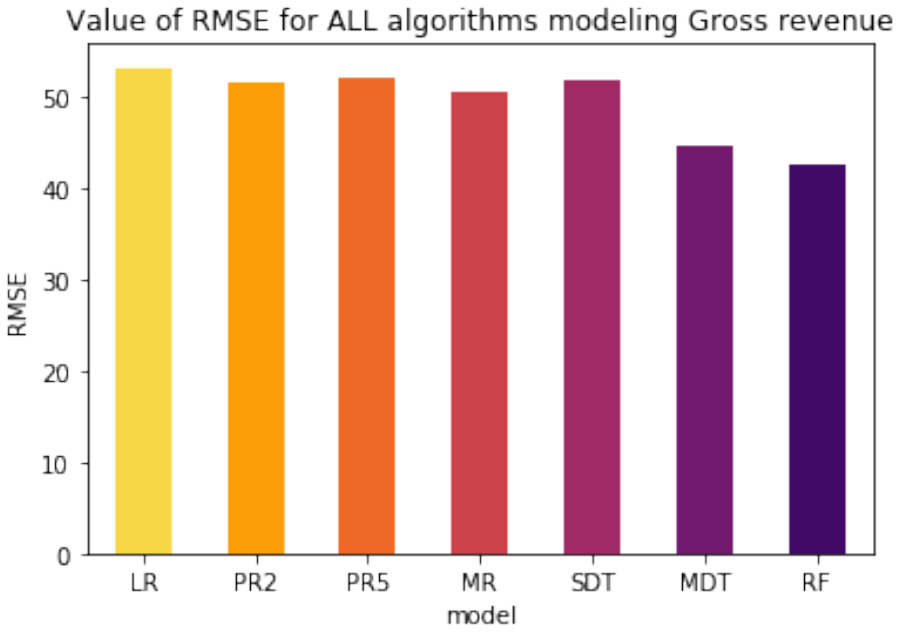


Figure 50: RMSE results for Gross revenue (all algorithms)

As it could be expected, the last models discussed offer a much lower RMSE value. LR models go as low as $RMSE = 50.460$, whereas RF reach $RMSE = 42.541$.

Again, the biggest difference appreciated is between the SDT and the MDT models, which present an error or $RMSE = 51.906$ and $RMSE = 44.741$, respectively.

To put these values in context, our RF model could predict the amount of money a new movie would generate with an average error of \$44M. This error may not seem much for high-revenue movies, but it becomes a problem for movies in the range of $\$0 - \$100M$ gross. For that reason, Section 6 discusses possible algorithm enhancements and new development options to reduce the predicting error and more accurately predict the output features under study.

In Figure 51, the value of R^2 in the case of predicting the IMDB score is shown for all the algorithms.

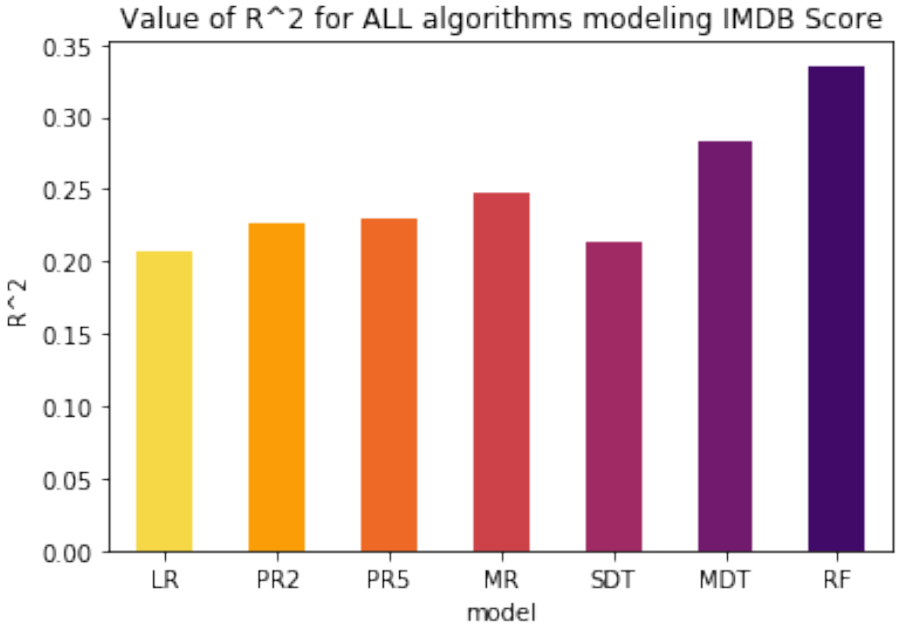


Figure 51: R^2 results for IMDB score (all algorithms)

In this case, the value of R^2 grows even more for the last models, being $R^2 = 0.207$ for the most simple LR model and $R^2 = 0.335$ for the RF model, almost twice as good a modeler for new data.

Once again, there is a considerable improvement in the DT models between the SDT ($R^2 = 0.214$) and the MDT model ($R^2 = 0.283$)

Finally, Figure 52 shows the value of RMSE in the case of predicting the IMDB score for all the algorithms.

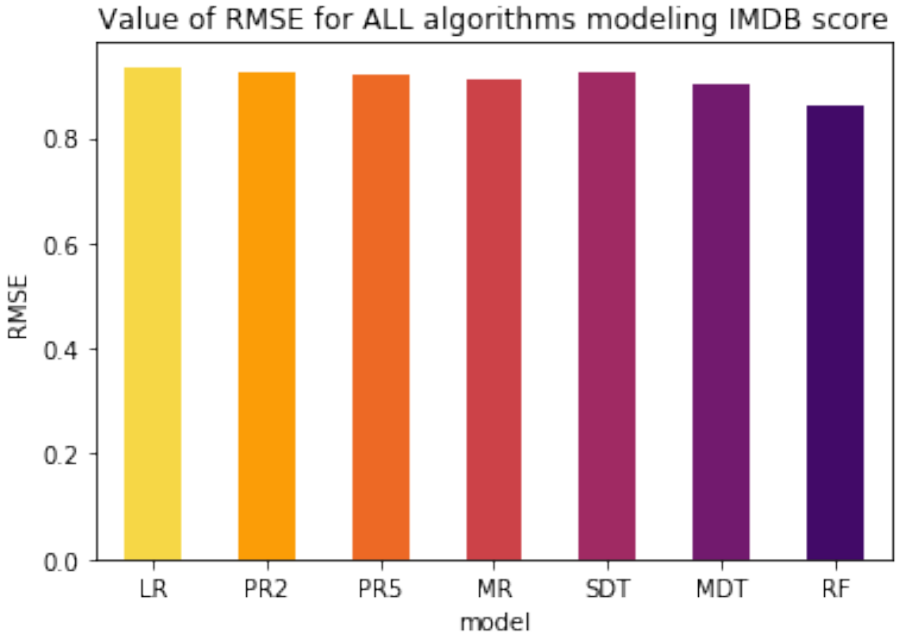


Figure 52: RMSE results for IMDB score (all algorithms)

In this case, the RMSE value is pretty stable for all algorithms, but only because the output range is much more limited (scores may go from 0 to 10).

The simplest LR model could predict the IMDB score for a new movie with an average error of $RMSE = 0.935$, whereas the RF model could do that with $RMSE = 0.859$.

6 Future Work

Once this work is finished and its ML algorithms have been studied, several future enhancements come around.

First and foremost, we have seen that more complicated algorithms better model the dataset under study, probably due to its highly scattered characteristics. With that in mind, it should be studied whether or not a Deep Learning algorithm would be appropriate for this dataset or, on the contrary, would be too sophisticated of a model.

Moreover, more data could be gathered to improve the results. The dataset under study was published 2 years ago, being its most recent movies from the year 2016. Being nowadays a highly marketed sector, it should not be complicated to collect thousands of examples of new movies with the features present on this dataset and even new ones.

Moreover, additional features could be used. With today's NLP techniques, the number of words per minute of movies could be counted, as well as how much their characters swear, the Spotify and Apple Music downloads for every song on their soundtrack, the number of news regarding the movie filming and many more.
Despite some of these features striking as seemingly irrelevant, they might later be proven to hold a considerable correlation with the movie's success, both in terms of revenue and IMDB score.

Finally, a lot of work could be done on the already existing features. Is a movie with a gross revenue of $50M in 2020 equivalent to a $50M one from 1990? Of course not. Money-related features could be modified to take into account the year's inflation, or the average gross for movies on that time. The profitability factor mentioned in this work could also be exploited, and the effect of streaming platforms like Netflix or HBO could be studied in relation to movies' gross.

In a nutshell, there are many things that could be done in a ML project like this, and that is probably the uniqueness of this field of study: the possibility of tackling a specific area of development from an infinite range of options.

7 Conclusions

In this piece of work, the foundations of Machine Learning have been discussed, together with some of the most promising applications for this cutting-edge technology.

Later, a movies' dataset from Kaggle has been studied using data science techniques, displaying hidden patterns in the data and gaining a bird perspective of the complexity of the information it contains.

Later, several ML algorithms have been discussed in detail and then used to predict the Gross revenue and the IMDB score for new movies.

In the last section, the performance of al the ML algorithms has been compared, reaching some important conclusions:

- Always the more data the better: datasets usually contain null values, outliers and even repeated values. Having a huge amount of data allows us to use sophisticated algorithms to study the idiosyncrasies in the data without incurring in overfitting.
- Not always the more features the better: Using too many features when trying to map new data may sometimes generate overfitting, consequently increasing the prediction error.
- Choose ML algorithm that best fits data: Not all datasets are equal. Some may contain a clear data distribution or have input/output features highly correlated. Others may not. Finding the most suitable algorithm for every dataset is a key aspect of Machine Learning.
- Parameter tuning: Once an appropriate ML algorithm has been chosen, its complexity and performance is deeply affected by the hyperparameters associated to it. These parameters are different for every algorithm, and it's an essential duty to properly tune them to get the most out of every algorithm depending on the data under study.

All in all, Machine Learning is a powerful and accessible, decision-making tool that, if used properly, will lead the coming technological revolution and, hopefully, make our lives a little easier.

References

[1] J. McCarthy and E. A. Feigenbaum, "In memoriam: Arthur samuel: Pioneer in machine learning," *AI Magazine*, vol. 11, no. 3, pp. 10–10, 1990.

[2] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[3] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.

[4] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, no. 1, p. 1, 2015.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[6] P. Jackson and I. Moulinier, *Natural language processing for online applications: Text retrieval, extraction and categorization*. John Benjamins Publishing, 2007, vol. 5.

[7] J. A. Cruz and D. S. Wishart, "Applications of machine learning in cancer prediction and prognosis," *Cancer informatics*, vol. 2, p. 117693510600200030, 2006.

[8] K. Hazelwood, S. Bird, D. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, M. Fawzy, B. Jia, Y. Jia, A. Kalro *et al.*, "Applied machine learning at facebook: A datacenter infrastructure perspective," pp. 620–629, 2018.

[9] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers *et al.*, "Practical lessons from predicting clicks on ads at facebook," in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, 2014, pp. 1–9.

[10] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 acm sigsac conference on computer and communications security*, 2016, pp. 1528–1540.

[11] V. Vovk, A. Gammerman, and C. Saunders, "Machine-learning applications of algorithmic randomness," 1999.

[12] M. Negnevitsky, P. Mandal, and A. K. Srivastava, "Machine learning applications for load, price and wind power prediction in power systems," in *2009 15th International Conference on Intelligent System Applications to Power Systems*, 2009, pp. 1–6.

8 Annexes

8.1 Numeric Correlation Matrix

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes	gross	num_voted_users	cast_total_facebook_likes
num_critic_for_reviews	1	0.258486	0.180674	0.271646	0.190016	0.480601	0.624943	0.263203
duration	0.258486	1	0.173296	0.123558	0.088449	0.250298	0.314765	0.123074
director_facebook_likes	0.180674	0.173296	1	0.120199	0.090723	0.144945	0.297057	0.119549
actor_3_facebook_likes	0.271646	0.123558	0.120199	1	0.249927	0.308026	0.287239	0.47392
actor_1_facebook_likes	0.190016	0.088449	0.090723	0.249927	1	0.154468	0.192804	0.951661
gross	0.480601	0.250298	0.144945	0.308026	0.154468	1	0.637271	0.2474
num_voted_users	0.624943	0.314765	0.297057	0.287239	0.192804	0.637271	1	0.265911
cast_total_facebook_likes	0.263203	0.123074	0.119549	0.47392	0.951661	0.2474	0.265911	1
facenumber_in_poster	-0.033897	0.013469	-0.041268	0.099368	0.072257	-0.027755	-0.026998	0.091475
num_user_for_reviews	0.609387	0.328403	0.22189	0.230189	0.145461	0.559958	0.798406	0.206923
budget	0.119994	0.074276	0.02109	0.047451	0.022639	0.102179	0.079621	0.036557
title_year	0.275707	-0.135038	-0.06382	0.096137	0.086873	0.030886	0.007397	0.109971
actor_2_facebook_likes	0.282306	0.131673	0.119601	0.559662	0.390487	0.262768	0.27079	0.628404
imdb_score	0.305303	0.261662	0.170802	0.052633	0.076099	0.198021	0.410965	0.085787
aspect_ratio	-0.049786	-0.090071	0.001642	-0.003366	-0.020049	0.069346	-0.014761	-0.017885
movie_facebook_likes	0.683176	0.196605	0.162048	0.278844	0.135348	0.378082	0.537924	0.209786

	facenumber_in_poster	num_user_for_reviews	budget	title_year	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes
num_critic_for_reviews	-0.033897	0.609387	0.119994	0.275707	0.282306	0.305303	-0.049786	0.683176
duration	0.013469	0.328403	0.074276	-0.135038	0.131673	0.261662	-0.090071	0.196605
director_facebook_likes	-0.041268	0.22189	0.02109	-0.06382	0.119601	0.170802	0.001642	0.162048
actor_3_facebook_likes	0.099368	0.230189	0.047451	0.096137	0.559662	0.052633	-0.003366	0.278844
actor_1_facebook_likes	0.072257	0.145461	0.022639	0.086873	0.390487	0.076099	-0.020049	0.135348
gross	-0.027755	0.559958	0.102179	0.030886	0.262768	0.198021	0.069346	0.378082
num_voted_users	-0.026998	0.798406	0.079621	0.007397	0.27079	0.410965	-0.014761	0.537924
cast_total_facebook_likes	0.091475	0.206923	0.036557	0.109971	0.628404	0.085787	-0.017885	0.209786
facenumber_in_poster	1	-0.069018	-0.019559	0.061504	0.071228	-0.062958	0.013713	0.008918
num_user_for_reviews	-0.069018	1	0.084292	-0.003147	0.219496	0.292475	-0.024719	0.400594
budget	-0.019559	0.084292	1	0.045726	0.044236	0.030688	0.006598	0.062039
title_year	0.061504	-0.003147	0.045726	1	0.10189	-0.209167	0.159973	0.218678
actor_2_facebook_likes	0.071228	0.219496	0.044236	0.10189	1	0.083808	-0.007783	0.243487
imdb_score	-0.062958	0.292475	0.030688	-0.209167	0.083808	1	0.059445	0.247049
aspect_ratio	0.013713	-0.024719	0.006598	0.159973	-0.007783	0.059445	1	0.025737
movie_facebook_likes	0.008918	0.400594	0.062039	0.218678	0.243487	0.247049	0.025737	1

Figure 53: Numerical version of the Correlation Matrix

8.2 Metrics

Mean Absolute Error (MAE)
MAE is used to express the difference between the predicted output of the test set examples and their actual output, weighing all differences equally.
In mathematical terms:

$$MAE = \frac{1}{K}\sum_{k=1}^{K}|y_k - y_{pred,k}| \tag{19}$$

Where K is the size of the test set, y_k is the actual output of the k_{th} test exampleand $y_{pred,k}$ is its predicted output.

Mean Squared Error (MSE)
MSE is used to express the squared difference between the predicted output of the test set examples and their actual output, weighing a large difference higher than several small ones.
In mathematical terms:

$$MSE = \frac{1}{K}\sum_{k=1}^{K}|y_k - y_{pred,k}|^2 \tag{20}$$

Root Mean Squared Error (RMSE)
RMSE is simply the squared root of MSE. In mathematical terms:

$$RMSE = \sqrt{\frac{1}{K}\sum_{k=1}^{K}|y_k - y_{pred,k}|^2} \tag{21}$$

Coefficient of determination (R^2)
This metric expresses the sensitivity of the dependent variable (y) in relation to the variance on the independent variable (x). The higher the R^2 is, the better the dependent variable adapts to (or models) the independent variable.
In mathematical terms:

$$R^2 = 1 - \frac{SS_{res}}{SS_{mean}} = 1 - \frac{\frac{1}{K}\sum_{k=1}^{K}(y_k - y_{pred,k})^2}{\frac{1}{K}\sum_{k=1}^{K}(y_k - \overline{y})^2} \tag{22}$$

Where $\overline{y}$ is the average of vector y.