IMDb Score: Analysis and prediction

Team 13: Khaled Hechmi[1], Gian Carlo Milanese[2]

Abstract
IMDb, acronym for Internet Movie Database, is a website owned by Amazon.com where users can look for details about movies and TV shows: plot summaries, users reviews, genre, director and cast are just some of the attributes stored on IMDb.
The goal of this research work is to predict the rating of a given movie or TV Show using properly trained Machine Learning models and evaluate the goodness of the predictions.
The chosen dataset was made available on the Kaggle platform and it contains the details of approximately 5000 popular movies and TV Shows.

Keywords
Machine Learning — IMDb — Movies — TV Shows

[1] *Universita' degli Studi di Milano Bicocca, CdLM Data Science*
[2] *Universita' degli Studi di Milano Bicocca, CdLM Data Science*

Contents

Introduction

What makes a good movie? The first aspects that come to mind are a compelling plot, strong performances from its actors, a suitable and memorable soundtrack and good visuals. A more competent viewer will also be interested in the direction and photography, and some movies will need to satisfy criteria dictated by their genre (for instance, horror movies must be tense and frightening). Movies are also often judged for the message they want to convey, or for the relevance of their themes.

While this is an oversimplified view, at least at first glance we can see that most of the aspects that come into play when deciding whether a movie is good or not are *subjective* or *hard to quantify*. For example, a plot might be original and compelling for some, but boring and predictable for others, and it is not possible to measure how scary a movie is, or how strong is a performance.

On the other hand, objective and measurable features of a movie – like its length, budget or gross earnings – need not be directly tied with its quality.

Despite the inherent difficulty, it is still worth investigating whether it is possible to assign a score to a movie starting from some objective and measurable features. The dataset chosen for answering this research question is the *IMDb 5000 Movies Dataset*, available on the Kaggle Platform [5]. It is made of 5044 records each with the following 28 features :

- movie_title (categorical - nominal):
  Original movie title
- actor_1_name (categorical - nominal):
  Name of the main actor/actress
- actor_1_facebook_likes (numeric - ratio):
  Number of likes on the main actor's/actress's Facebook page
- actor_2_name (categorical - nominal):
  Name of the second main actor/actress
- actor_2_facebook_likes (numeric - ratio):
  Number of likes on the second main actor's/actress's Facebook page
- actor_3_name (categorical - nominal):
  Name of the third main actor/actress

- actor_3_facebook_likes (numeric - ratio):
  Number of likes on the third main actor's/actress's Facebook page
- director_name (categorical - nominal):
  Name of the movie director
- director_facebook_likes (numeric - ratio):
  Number of likes on the director's Facebook page
- cast_total_facebook_likes (numeric - ratio):
  Sum of likes on cast's Facebook pages
- movie_facebook_likes (numeric - ratio):
  Number of likes on the movie's official Facebook page
- num_voted_users (numeric - ratio):
  Number of users that have voted the movie
- num_critic_for_reviews (numeric - ratio):
  Number of critics that have written a review
- num_user_for_reviews (numeric - ratio):
  Number of users that have written a review
- budget (numeric - ratio):
  Money needed for recording the movie
- gross (numeric - ratio):
  Gross earnings of the movie
- genres (categorical - nominal):
  Genre(s) of the movie separated by '|' character.
- color (categorical - binary):
  Black and white or color
- duration (numeric - ratio):
  Movie length in minutes
- title_year (categorical - nominal):
  Year of movie release
- language (categorical - nominal):
  Original movie language
- country (categorical - nominal):
  Country where the movie was mainly recorded
- content_rating (categorical - ordinal):
  Rating of the movie's suitability for certain audiences based on its content
- plot_keywords (categorical - nominal):
  Keyword(s) that describe(s) the movie, separated by '|' character
- aspect_ratio (numeric - ratio):
  Proportional relationship between the width and the height of the movie image
- facenumber_in_poster (numeric - ratio):
  Number of faces displayed in the film poster
- movie_imdb_link (categorical - nominal):
  URL of the IMDb page about the movie
- imdb_score (numeric - ratio):
  Movie rating on IMDb

The goal of our analysis is to predict the rating for a given movie or TV show using machine learning tools and evaluate the goodness of the predictions.

This report is organized as follows:

1. Data exploration We examine features from the dataset, focusing on *imdb_score*.

2. Preprocessing We remove some columns from the dataset, transform some features and handle missing values in order to make the dataset more suitable for analysis.

3. Models We describe the different models used to predict the value of *imdb_score*.

4. Valuation We evaluate and compare the models described in the previous section.

1. Data exploration

While the dataset features information about movies produced between 1916 and 2016, roughly 73% of them have been released after the year 2000, and only around 4% of them are in black and white. Moreover, most movies available in the dateset are produced in the USA or in the UK (slightly less than 85%).

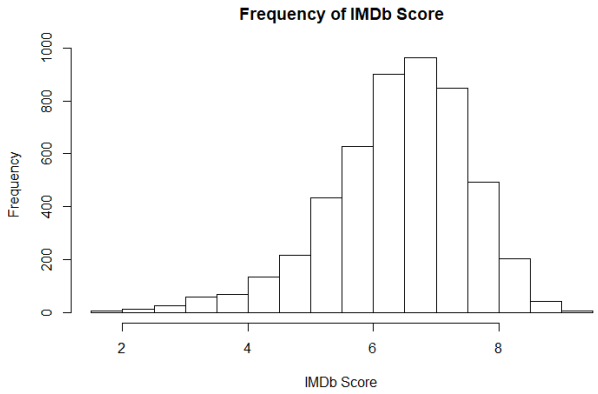

Figure 1. Histogram of IMDb Score Frequency

Since our goal is to predict the rating of a movie or a TV show, we consider *imdb_score* to be the dependent variable in our analysis. This variable represents the rating of a movie, as found on the IMDb website. The IMDb rating is calculated as a weighted average of all the individual ratings cast by IMDb registered users (the exact method used to generate the rating is not disclosed [4]) and can take values between 1 and 10 with precision up to one decimal point. We have thus treated *imdb_score* as a continuous attribute. The least value observed for *imdb_score* in our dataset is 1.6, while the greatest one is 9.6. The distribution of this feature is shown in Figure 1.

2. Preprocessing

In order to make the dataset more suitable for analysis we have implemented the following changes.

2.1 Feature selection

Out of all the 28 features we have discarded those that intuitively should have no effect on the dependent variable *imdb_score*, or those that are nominal and with too many values to be taken advantage of effectively. These include *aspect_ratio*, *plot_keywords* and all features consisting of the names of the director and cast members.

We have also removed the features *color* and *language* due to the very small frequency of movies that are not colored or not in the English language, and we have removed the columns *movie_imdb_link* and *movie_title* as their values univocally reference each row.

Lastly, we have discarded features that could cause multicollinearity: this circumstance should be avoided when training a multiple regression model, since it is required that the predictor variables are independent from one another. In order to achieve this we have computed the correlation between the available features, and then proceeded to filter out highly correlated variables through a *Correlation Filter* node, where we have set the Correlation Threshold to 0.8.

2.2 Feature transformation

For each movie in the dataset the column *genres* shows a list of relevant genres separated by the character '|'. We have added a new column to the dataset for each unique value of the feature *genres*, hence making each genre a binary feature taking value 1 if a movie falls under that genre, and value 0 otherwise. This was done because, otherwise, in the *genre* column there would have been 914 different values, which the predictive models would have treated as completely different, not exploiting their similarities (e.g. movies with "Action | Adventure" and "Action | Adventure | Fantasy" as *genres* values share the values "Action" and "Adventure"). A total of 26 new columns were added to the dataset and the original 'genre' column was deleted.

While the feature *country* can take 65 unique values, as we have mentioned the vast majority of movies in this dataset are produced in the UK or in the USA. We have thus decided to group all movies not produced in either of these two countries in a single category, so that *country* now has levels 'UK', 'USA' and 'Others'.

We have also performed a transformation for the column *content_rating*. In this case some ratings with a different name (for instance due to different naming conventions for movies and TV shows, like PG and TV-PG) indicate a similar intended audience. Over the years, the Motion Picture Association of America has changed their rating system [10] many times in order to address society and experts' requests for better clarity. We have thus combined similar ratings in five categories: *G* (general audiences), *PG* (parental guidance), *Adult* (adult audience), *Not Rated* (not submitted for a rating) and *Passed* (approved for screening).

Some features that we consider of importance in our analysis – namely *budget*, *gross* and *num_voted_users* – present a distribution that is heavily skewed to the right. We have transformed these features using a logarithm in order to make the relationships between each of these variables and the dependent variable more linear [2].

2.3 Handling of missing values

All but seven features in the *IMDb 5000 Movies* dataset have missing values: while there is no proper way to address this issue, because the reason why there are NAs is not known, these cases need to be handled carefully in order to avoid problems in the development of predictive models.

The features with the largest number of missing values are *gross* (17.5%) and *budget* (9.7%). We have decided to eliminate the rows having NAs in either of these two columns: no explanation was found online about their absence, so we applied this strategy in order to avoid distortions in the dataset due to wrong imputation.

Due to the relatively low number of missing values after this operation (the feature with the highest percentage of missing values is now *content_rating*, with 1.31% NAs) , the remaining missing values were handled as follows:

- Categorical Features: replace NAs with the most frequent value;
- Numerical Features: replace NAs with the median value.

2.4 Transformation of categorical variables

As previously described the IMDb dataset is composed by quantitative and qualitative features. The latter ones cannot be used in their original form for properly training the majority of predictive models. For this reason the transformation of these variables into binary ones was needed. The *one-hot encoding technique* was chosen: for each n unique values of a given feature, n columns were added to the dataset.

Compared to the *binary encoding* approach, the other most used technique for converting qualitative features, the increase in dimensionality is bigger (n new columns are added for n unique values, compared to $\lceil \log_2 n \rceil$ columns added using *binary encoding*). The main advantage of one-hot encoding is that it considers each unique value independent from the others, avoiding the addition of correlations not present in the original data.

This operation was done using the *One To Many* Knime node. The original categorical variables were deleted after performing the transformation.

After all the described operations are performed the dataset counts 3891 observations with 48 features.

3. Models

Different Machine Learning models were chosen for predicting the value of *imdb_score*. Here is the list of the Knime nodes:

- *Simple Linear Regression* (*SLR*), based on Weka 3.7, only uses one feature for predicting the target variable, namely the one that results in the lowest squared error;
- *Linear Regression* (*LR*), based on Weka 3.7, takes advantage of multiple features for predicting the target variable;
- *Simple Regression Tree Learner* (*SRTL*), which follows the algorithms described in CART [3];
- *Polynomial Linear Regression* (*POLY*), which also makes use of polynomial terms, and where the polynomials maximum degree to compute is set to 2.

Each of these algorithms was trained using three different approaches: Holdout, Cross Validation and Feature filtering.

3.1 Holdout

The first approach, known as Holdout, was implemented as follows: the entire dataset was split randomly in two parts, specifying the random seed so that different runs would lead to the same results. The first one, containing around 70% of records, was used for training the models. The remaining ones were not used for learning the relation between *imdb_score* and the independent variables. In this way it is possible to simulate a real production scenario, where the trained models will be queried with unseen data in order to obtain a valid prediction. All the features available in the dataset after the preprocessing phase were used.

3.2 Cross Validation

Each model was evaluated using cross validation. In particular 5-fold cross validation was used: the dataset was split in five subsets and the models were trained five times, using, at each iteration, four parts as training set and one part as test set.

This was done in order to have a clearer understanding of the model performances and to better compare the performances of all models. All the features contained in the dataset were used for training the models and scores such as R^2 and RMSE, obtained by each fold, were stored for analysis purposes.

3.3 Feature filtering

The last approach involved the filtering of features used for predicting *imdb_score*. One benefit of performing this activity is the reduction in the dataset's dimensionality. Another reason for reducing the number of features is the possibility of obtaining more understandable models, which is an important aspect for users in both industry and academia [6]. It will also be easier for domain experts to validate the models' results.

The Knime nodes *Feature Selection Loop Start (1:1)*, *Feature Selection Loop End* and *Feature Selection Filter* were used for implementing this operation: in particular it was chosen to use the backward elimination approach, using the R^2 as the metric to maximize. The algorithm starts using all the features, removing at each iteration the less relevant feature until the removal of a feature will lead to a significantly worse score. Among the results leading to the highest R^2 score, the one with less features was chosen for each model. The feature selection is performed using a 3-fold cross validation procedure: even though it is common practice to use 10-folds cross validation [9], this value is a compromise between computational resources and accuracy of results. The test set was not used for learning which features are less relevant so that bias introduction is avoided. The execution of this method takes around 15 minutes per model using a PC with 16 GB of RAM and an i5-7500 CPU: fine-tuning Knime performances is highly suggested [7]. The Feature filtering was not applied to the *Simple Linear Regression* model, because by design it already selects a single feature, and to the *Polynomial Linear Regression* algorithm because of performances issues: the latter's metanode is available in the workflow, under the "Feature Filtering" metanode, even tough it is not connected to any input or output ports.

4. Evaluation

4.1 Holdout

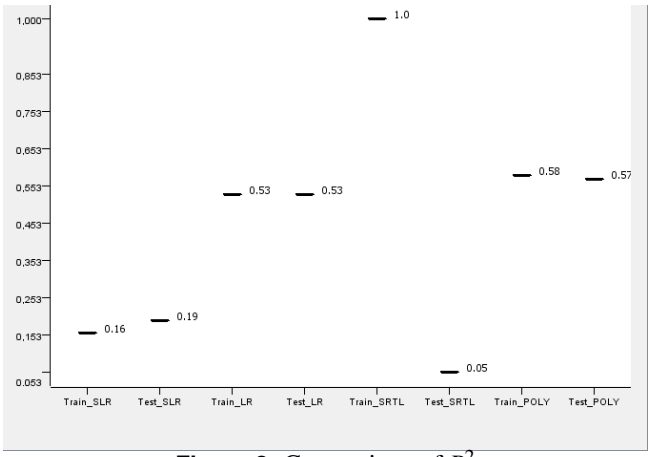

Figure 2. Comparison of R^2

The model with the best performances on train set is the *Simple Regression Tree Learner*. It has a R^2 score of 1.0, which is the maximum possible: the model predicts perfectly every single point in the train set.

Regarding the performances on test data, the best model is the *Polynomial Linear Regression*, with a R^2 score of 0.57. The score is not that different from the one obtained in the train set, suggesting that the model does not suffer from overfitting.

On the other hand, it is evident that the model that suffers from overfitting is the *Simple Regression Tree Learner*. The perfect score on the training set was by itself a huge indicator

of a mere memorization of training points instead of finding the potential relationship between independent variables and *imdb_score*. This was confirmed by the performances on the test set, where a R^2 score of 0.05 is achieved.

A similar situation is found as far as the *Root Mean Squared Error* (*RMSE*) for these models is concerned, as shown in Figure 3.

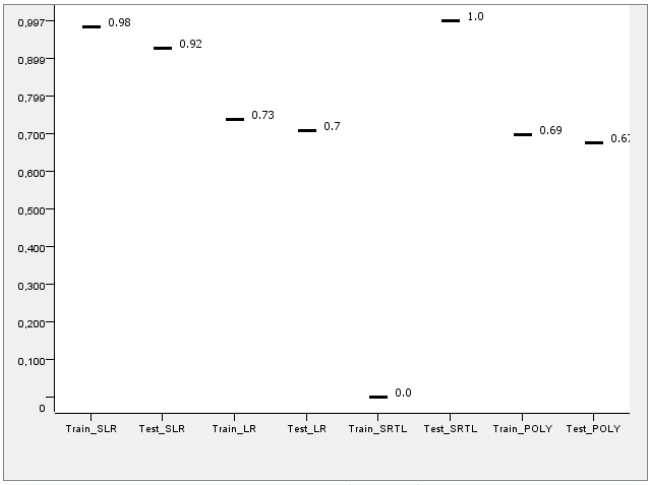

Figure 3. Comparison of *RMSE*

We also note that the Linear Regression model achieves scores that are very close to those of the Polynomial model, implying that the increment in complexity of the latter model does not translate incisively on its predictive capability.

Another important aspect to consider when evaluating the quality of a predictive model is the distribution of the residuals, which should be normally distributed. In order to verify whether this in fact holds for our models, we performed a Shapiro-Wilk Normality Test, as well as plotted the histogram of residuals and a quantile-quantile plot. We performed these operations for the *Linear Regression* and *Polynomial Linear Regression* models. Figure 4 and Figure 5 show the histogram of the residuals and the quantile-quantile plot for the latter model, respectively.

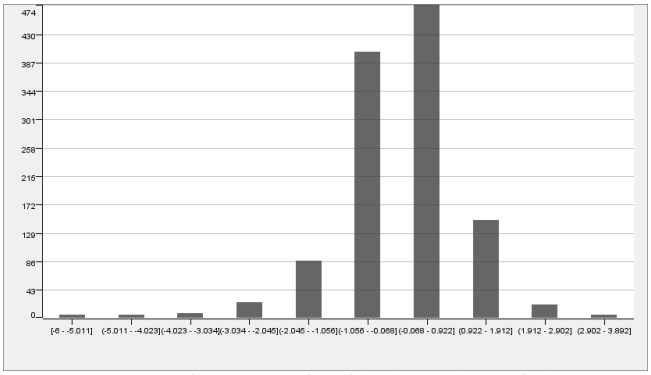

Figure 4. Histogram of residuals, Polynomial LR

While it is already heavily suggested by the figures the the distribution of residuals is not normal, the Shapiro-Wilk test

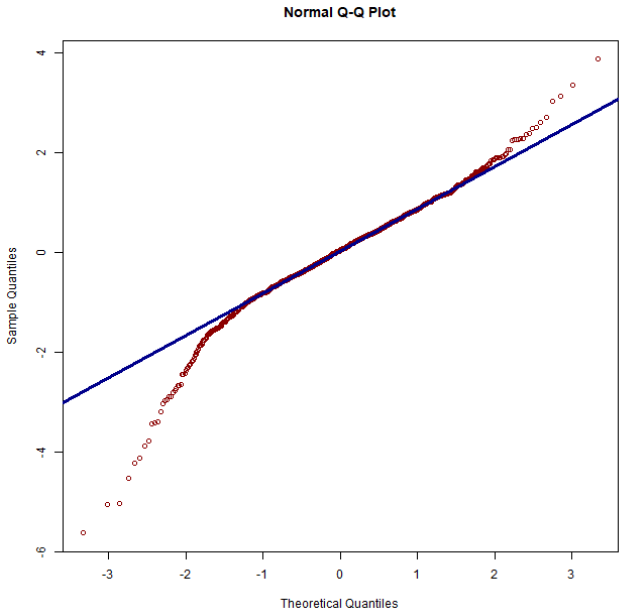

Figure 5. Q-Q plot of residuals, Polynomial LR

allows us to refuse this hypothesis at the significance level of 0.01 (p-value $= 2.22 \times 10^{-16}$).

Figure 6 shows the scatter plot comparing the real values of *imdb_score* versus the values predicted by the *Polynomial Linear Regression* model.

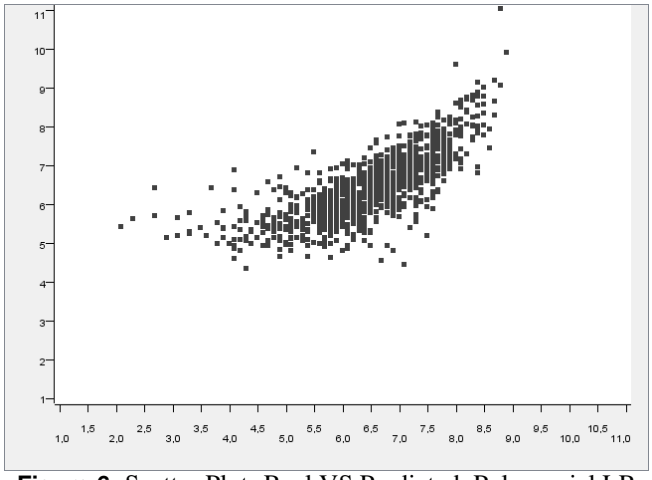

Figure 6. Scatter Plot: Real VS Predicted, Polynomial LR

Similar considerations can be developed with respect to the Linear Regression model, where the null hypothesis that the residuals are normally distributed can be rejected again at the significance level of 0.01 (p-value $= 1.11 \times 10^{-16}$). The relevant histogram of residuals, quantile-quantile plot and scatterplot of real versus predicted values can be found in the Knime workflow.

4.2 Cross Validation

Focusing on the cross-validated models, we have computed R^2 and *RMSE* on each fold of all the models. These scores can

be visualized through the box plots of Figure 7 and Figure 8.

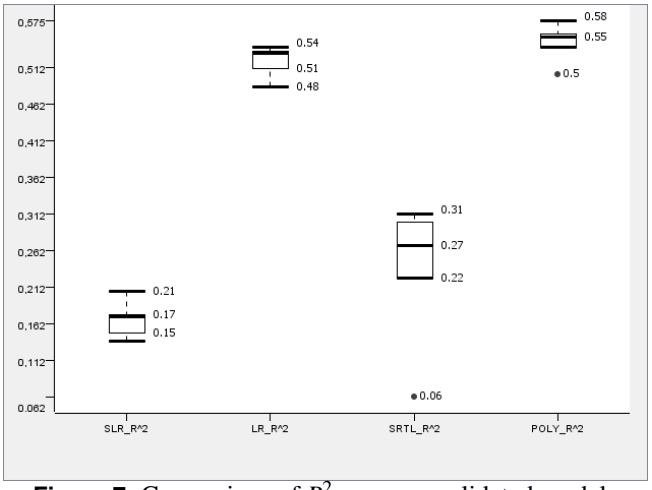

Figure 7. Comparison of R^2 on cross-validated models

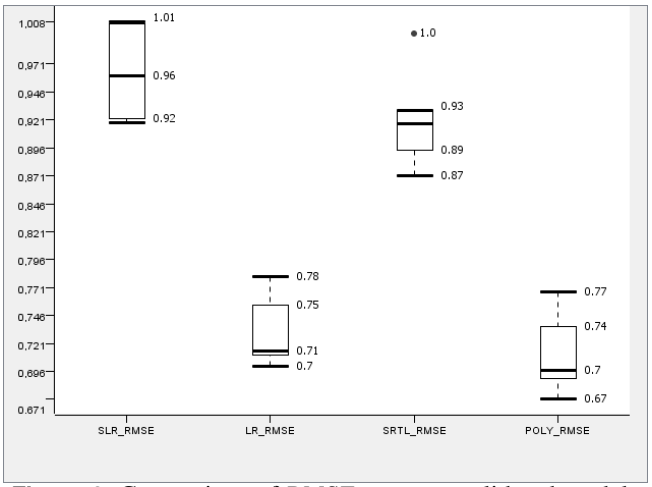

Figure 8. Comparison of *RMSE* on cross-validated models

These results support the impression that the *Linear Regression* and *Polynomial Linear Regression* models are better predictors of *imdb_score*, compared to the *Simple Linear Regression* and *Simple Regression Tree Learner* models. Besides the fact that the values of R^2 for the first two models are higher, we can also see that their variability is also lower.

The aforementioned models also benefit from smaller values of *RMSE*, similarly to what we saw in the previous section.

4.3 Feature filtering

After performing 3-fold cross validation for filtering the number of dataset features, the dataset changed as follows:

- *Linear Regression*: 21 features were kept. The R^2 score on the validation set is equal to 0.516;
- *Simple Regresion Tree Learner*: 23 features were kept. The R^2 score on the validation set is equal to 0.2: the best score was 0.202 with 28 features, but we preferred to use 5 less features instead of achieving a 0.002 better R^2.

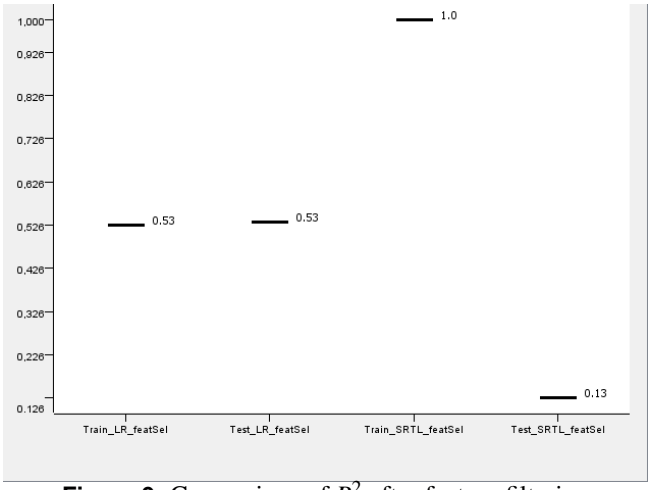

Figure 9. Comparison of R^2 after feature filtering

After training the models with the filtered datasets, the best perfomance on the train set is obtained by *Simple Regression Tree Learner* with a R^2 score equal to 1.0. Instead, on the test set the best performing model is *Linear Regression*, with a R^2 of 0.53. Based on these results it is evident that the *Simple Regression Tree Learner* model suffers from overfitting, as happened in the Holdout scenario. The *Linear Regression* model does not suffer from overfitting: its performances are the same on the train and on the test sets.

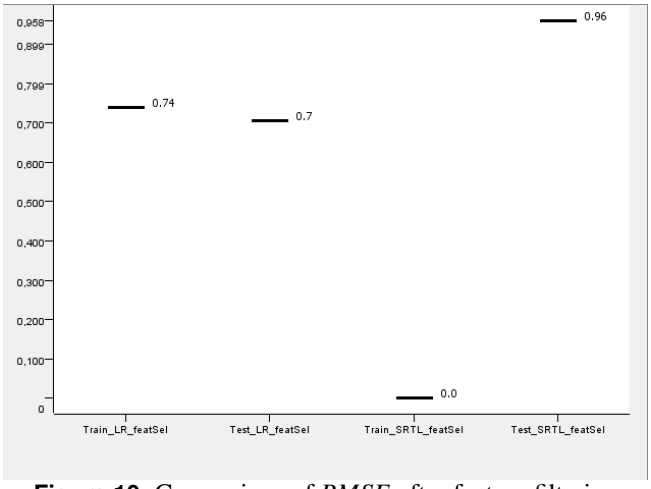

Figure 10. Comparison of *RMSE* after feature filtering

Using less features did not negatively impact the performances of the models, as shown in Figure 9 and in Figure 10. The scores are the same for the *Linear Regression* model, while *Simple Regression Tree Learner* has a better R^2 score on test set (0.13 vs 0.05). The gain on models interpretability and the decreased dimensionality are other two reasons for confirming the goodness of this approach.

Conclusion

As was argued in the introduction, predicting the rating of a movie or a TV show starting from some of its measurable quantitative features, which was the goal of our study, is not an easy task. This is reflected in the performance of the models we have analyzed, which seems to support the intuitive idea that the quality of a movie does not depend entirely on its budget, on the country where it was produced, or on its popularity (represented by a movie's gross earnings, by the number of likes on the Facebook pages of the director, actors and movie itself, and by the number of users and critics that have left a vote or a review on the IMDb page of a movie).

On the other hand, the impact of these features on the IMDb Score is not completely negligible, as witnessed by the R^2 and *RMSE* measures of, at least, the *Linear Regression* and *Polynomial Linear Regression* models: after the evaluation considerations these two models are the ones that obtained the most satisfying results.

Even though the results are encouraging, it is possible to improve them using common approaches such as adopting more complex models, like Artificial Neural Networks, or adding more features [8]. Another way to improve our analysis could consist in a different treatement of missing values. Discarding all rows presenting missing values in either the column *gross* or *budget* resulted in a significant reduction of the size of the dataset (a 22.85% decrease). Better approaches might involve *Maximum Likelihood Estimation* or *Multiple Imputation* [1]. Being able to impute the missing data would allow to keep all of the information in the original dataset, and possibly improve the predictive power of the models.

References

[1] Allison P. D. *Missing Data*. Retrieved from http://www.statisticalhorizons.com/wp-content/uploads/2012/01/Milsap-Allison.pdf

[2] Benoit, K. *Linear Regression Models with Logarithmic Transformations* [Course Notes]. Retrieved from: kenbenoit.net/assets/courses/ME104/logmodels2.pdf

[3] Breiman, L. (1984). Classification and Regression Trees. New York: Routledge.

[4] IMDB (January 2019). *FAQ for IMDb Ratings*. Retrieved from help.imdb.com/article/imdb/track-movies-tv/faq-for-imdb-ratings/G67Y87TFYYP6TWAV#

[5] Kaggle (January 2019). *IMDb 5000 Movies Dataset*. Retrieved from: www.kaggle.com/carolzhangdc/imdb-5000.

[6] KDnuggets (November 2018). *How Important is that Machine Learning Model be Understandable? We analyze poll results*. Retrieved from: kdnuggets.com/2018/11/machine-learning-model-understandable-poll-results.html

[7] Knime (January 2019). *Optimizing KNIME workflows for performance*. Retrieved from: www.knime.com/blog/optimizing-knime-workflows-for-performance

[8] Machine Learning Mastery (November 2016). *Machine Learning Performance Improvement Cheat Sheet*. Retrieved from: machinelearningmastery.com/machine-learning-performance-improvement-cheat-sheet/

[9] Wikipedia (January 2019). *K-fold cross validation*. Retrieved from: en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation

[10] Wikipedia (January 2019). *Motion Picture Association of America film rating system*. Retrieved from: en.wikipedia.org/wiki/Motion_Picture_Association_of_America_film_rating_system