LELEC2870 Machine Learning Project: Predicting a film's gross revenue

Robin Paquet - 02192001
Hugo Vandermosten - 25741900

December 23, 2022

1 Introduction

In this project, we have been asked to apply machine learning in the context of revenue prediction for a movie in the USA. Therefore, we had access to a certain amount of data allowing us to train the prediction model that would be the best for our application. These data were collected from the International Movie Database (IMDb) and the BoxOfficeMojo website.

We present in this report our results as well as the different steps we followed in order to achieve this goal.

2 Our data preprocessing methods

Data preprocessing is an important step since it aims to have a resulting high quality features in a suitable format for our analysis. We therefore used multiple preprocessing methods that are presented in this section.

2.1 Unused features

After simply looking a the different features of the dataset, we noticed that the dataset included a feature called `Unnamed: 0` which did not seem useful in this project since it was not describing anything in particular.

We also realized that the features standing for the image and description of the movies appeared twice in different forms :

- Concerning the images, one feature listed the urls of each movie image and an other listed those images processed by a ResNet-50 pretrained on the ImageNet dataset. And the output is an embedding with a float vector of size 2048.
- Regarding the description of the movies, a first feature contains the description and another feature contains this same description in the form of embedding with a float vector of size 768. This vector is the output of the BERT language model pretrained on translation tasks.

We thus decided to remove the features named `img_url` and `description` and to only keep the embedded features[1].

We then wondered if it was interesting to leave or remove the feature `is_adult` as a categorical one. But, since its value is always equal to 0 whether in the training set `X1` or in the dataset `X2`, we decided removing it too since it could not have beneficial effects on our results.

[1] Using other preprocessing methods to, for an example, take the number of words of the film description as a new feature could also have been done. However, as you will be able to see in next sections, the film descriptions and images did not seem to have a huge impact on the target values regarding the embeddings at least.

Finally, the `title` feature also remained unused because we didn't did not get time to create embeddings ourselves. Furthermore, we also believed that looking at basic metrics such as the size of the title could not really improve our results.

2.2 Directly usable features

In our dataset, there were some features that we realized did not really have any preprocessing needed. The `ratings` feature is one of them because it is a numerical feature that shouldn't need normalization or scaling since all its values are between 1 and 10 for the rating of each movie.

In a similar way but with higher numbers, the feature `n_votes` is left as such because it is a quantitative feature with no real missing values. We tried using scaling and normalizing function for this feature but, since it was not improving our result, we decided to not keep it.

2.3 Other preprocessing methods

2.3.1 `production_year` and `release_year` features

The `production_year` and `release_year` features are pretty straightforward and might not need any preprocessing since it lists dates between 1916 and 2019. However we have thought of some methods for a possible more efficient optimization. Both features are processed in the same way and therefore under the same name of variable in the project. We defined multiple different ways to deal with these features :

- The first one is to leave the years as they are by using the `no_period` tag for our preprocessing.
- The second one is produced by using the `per_quantile` tag which results in preprocessing the two features by creating categorical features corresponding to some time periods. These periods are chosen so that the number of sample in each period is similar.
- The last tag `per_period_length` is similar to the previous one but creates the period categories by computing periods of nearly equal lengths.

The two last possibilities were chosen because we believed the popularity of films and even of the cinema overall could be more relied to some periods. The number of periods to compute is chosen by the `n year period` parameter in our notebook.

2.3.2 `runtime` feature

The `runtime` feature causes the problem that 7% of the values in the dataset are missing and have the symbol "\\N". We thus implemented sevaral ways to deal with these missing values:

- The first way we deal with it is by replacing all these symbols by the value 0 even if it doesn't seem intuitive. This could induce a bias in the data because the missing values should all be different from 0. This is done by using the tag `zero` for the `runtime replace type` option in our code.
- The two other possible tags are `mean` and `median` which replace the values either by the average or by the median of the `runtime` feature. This would seem to be more appropriate to avoid introducing bias in our dataset but this is not always the case as we will see in the Result section.

2.3.3 `studio` feature

For the preprocessing of the `studio` feature, we had to face the problem that a large amount of studios only created a very small amount of films. Creating a qualitative feature for each studio would thus have surely led to over fitting. The solution we found to deal with this problem is to create a categorical feature corresponding the the studios who created a small amount of films. The threshold for the minimal number of films that a studio has to have produced in order to be considered as a studio in its own right has arbitrarily been set to 5 in our implementation.

2.3.4 `genres` feature

Regarding the different genres of films, these were written as a "list" of coma-separated genres for each film. The solution we found to deal with this feature is to simply create a categorical feature for each possible film genre since it was not bringing big outliers such as for the `studio` feature.

Moreover, some of the `genre` values were not defined, we thus created a category called `Others` to classify them in a more organized way.

2.3.5 `text_embeddings` and `img_embeddings` features

The `text_embeddings` and `img_embeddings` features, which translate the movies descriptions and images into embeddings, are written as float vectors with sizes of 768 and 2048. This number of features being way too high, it was mandatory to reduce the dimensions in order to not impact too much the performances of our models in terms of over fitting. We therefore decided to apply PCA to these embeddings to reduce the dimensions while preserving as much information about the data as possible.

The output dimension of these PCA reductions are chosen using the `text embedding PCA dim` and `img embedding PCA dim` options.

2.4 Applying a scaling function to the target value

Since the target value was very far from having a well distributed form and films even had revenues of quite different orders of magnitude, we also thought that maybe trying to scale de target value could be a good idea for that.

We thus tried predicting the log of the films revenues too but the results were very bad compared to the results when using no scaling and we therefore did not use it in the end.

3 Model selection

3.1 Using randomized search for model parameters selection

Selecting the best parameters for a given regression method and preprocessed dataset can be done using multiple searching algorithms. Since the `sklearn` library provides two built-in searching functions able to tune these parameters, we decided to use one of them. The two functions use cross validation to compute their score, it thus also ensures a certain reliability.

We therefore had the choice between using either a grid search algorithm or a randomized search algorithm. We opted for the use of the randomized search cross validation algorithm in the end since the most computationally expensive algorithm we tried tuning were also the ones presenting the biggest amount of impacting parameters. Since sklearn does not provide any way to use a GPU in the calculations, combining it with our local search algorithm (explained in next section) for the preprocessing parameters was a little too expensive in terms of time/calculations. Furthermore, It is sometimes quite complicated to know what parameters really have the most impact on a regression model, using a localsearch algorithm instead of a full grid search algorithm most of the time ensures a right parameter selection for the most impactful ones without having to check every single possible combinaison.

3.2 Using local search for preprocessing parameters selection

As explained in the "data preprocessing" section, we thought of multiple ways to preprocess the given dataset. Which of our method is the best one to use is very hard to determine theoretically. The best preprocessing method for some of the features surely even depends on the regression model we apply on it.

We therefore decided to determine experimentally what is the best preprocessing we could apply on our dataset. This is done in our project by using a basic local search algorithm over the preprocessing parameters. Local search algorithms are known to be extremely efficient to find optimum or nearly optimum states for problems whose objective function do not present much local minimums. Since we believe that the preprocessing we apply on each feature should not highly depend on the other features preprocessings, we thought using local search might be very efficient in terms of quality of the solution found without it asking for too much computational power.

3.3 Combining the two searching methods

The method we finally used to choose an as good combinaison of preprocessing and model parameters as possible was to combine the two just presented searching algorithms. Indeed, our final method works in two layers.

The main algorithm is the local search algorithm we implemented. This one looks for the best preprocessing parameters among some feature preprocessing options we give to the algorithm at initialisation. Each state in this algorithm is thus represented by a combination of preprocessing parameters. The function this "first layer algorithm" tries to minimize is the RMSE of the best regressor parameters found by the 5-fold cross validation randomized search on the dataset preprocessed according to the state parameters. The different regressor parameters options also have to be given as input to the main algorithm.

4 Results

4.1 Disclaimer

Running our code will surely not create the exact same graphs that we present in this "Result" section since multiple parts in each layer of the algorithm rely on some randomness. However, the final results and the overall plots shapes should always be very similar. Every computed result in this section also has been produced with the same parameters possibilities as the preset ones of the notebook.

4.2 Used regression methods

As asked in the project statement, we tried optimizing an `OLS`, a `KNN` and a `MLP` regressor. The other non-linear regression method we decided to use is the `RandomForestRegressor`. This method has been chosen since it has interesting strengths for our application :

- It can smooth out the decision boundary of the decision tree and thus make more robust predictions, which can be extremely useful when randomness is involved in the target feature and this is surely the case when analysing film's gross revenues.
- It deals very well with sparse data and since we use multiple categorical features, it is important to deal with it as well as possible.
- It is computationally-wise rather inexpensive for a non-linear regression method, which is useful in our case since we have to fit it multiple times with our experimental parameters tuning method.

Since the `OLS` regressor gave quite good results (as presented in next section) and linear methods are very inexpensive methods, we also tried some other linear methods. The one showing the best results among them was the `Ridge` regressor, we thus also kept it for our final analysis. Since over fitting is a huge deal in this project and the `Ridge` regressor introduces a regularization term that penalizes the size of the coefficients in order to prevent it, better results were indeed expected.

Because the MLP regressor requires much more calculations in order to be trained than the other regression methods and since `sklearn` does not provide a way to work with a GPU, we also decided to not train it using our algorithm described above. Further informations about it stand in section 4.4.

4.3 Our main algorithm results

4.3.1 Optimal parameters and resulting RMSE

The final results we obtained when running our complete parameters-tuning algorithm on the chosen regressors[2] are the followings :

	production year style	n year period	runtime replace type	text embedding PCA dim	img embedding PCA dim	mean RMSE	RMSE variance
OLS	per period length	3	zero	100	1	5.620e7	5.006e6
Ridge	per period length	3	zero	100	1	5.539e7	5.9221e6
KNN	per quantile	3	zero	20	1	6.099e7	5.432e6
RF	no period	/	zero	1	1	5.290e7	5.831e6

Regarding the final scores over the 5-fold cross validation of each of the final models, we can easily select the `RandomForest` regressor as the most promising one. Indeed, it has the lowest cross validation mean RMSE as well as a RMSE variance very comparable to the other regressors. This thus means the model should be as robust as the other ones while giving better results. The fact that this regressor shows the best results is obviously linked to the reasons why we chose to use it (presented in last section).

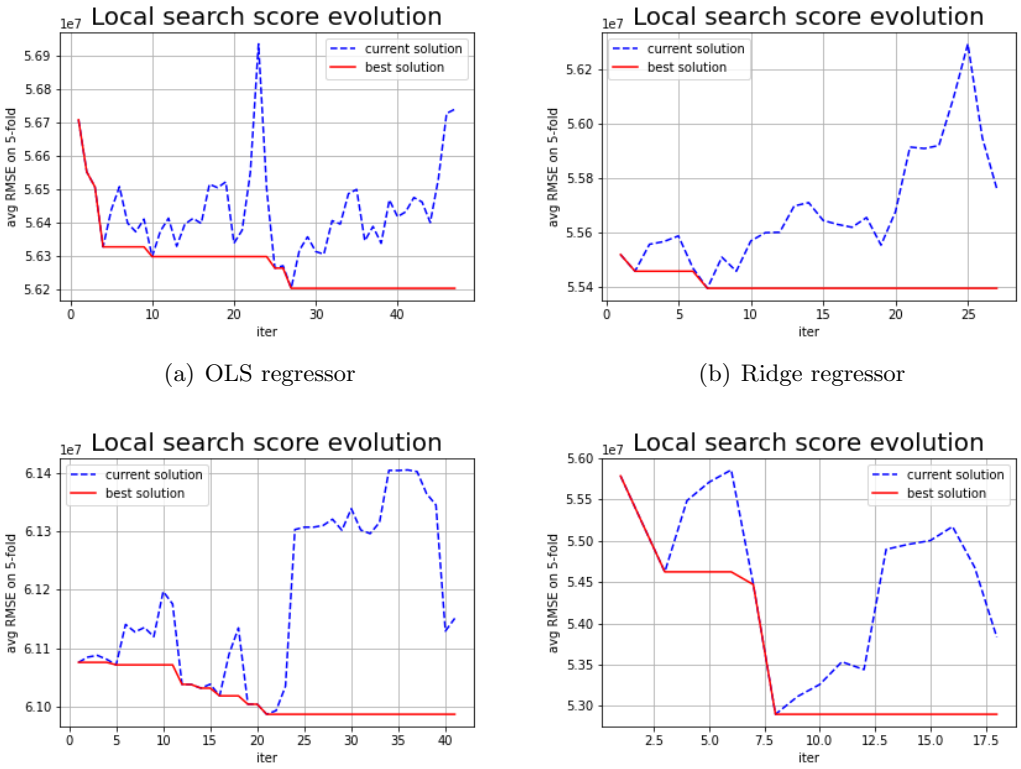
An analysis we can make about these results is also that, whatever the regression method we used is, the found optimal way to deal with the `runtime` feature is to replace the missing values by 0, which could be quite counterintuitive, using an experimental method to tune the parameters might thus have been a good idea regarding that. It is also a good example to show that not making any assumptions is sometimes the best way to work in data science. Furthermore, the KNN regressor shows the worst results among the selected regressors. Since it firstly is a classification algorithm, this means that the films revenues do not seem to be related to specific film clusters or classes. We can also observe that each regression method ends up getting better results with a dimension of 1 for the `img_embedding` PCA reductions. That means that this feature might be quite irrelevant in the films revenue predictions. A good improvement we could have added to our project would thus maybe have been to add the possibility to remove this features completely and check whether it leads to better performances. On the other hand, nearly each regression method ended up working well with different ways of dealing with the `production year` and `text embedding` features. It is thus very hard to make any other analysis than guessing these might be "less important" features for our application.

Tables containing the optimal regressor parameters found for these optimal preprocessings also stand in appendix. However, since most of the optimal parameters finally were (close to) the default ones, we decided to not make a full analysis about it.

4.3.2 RMSE evolution with local search

The following images (on Figure 1) show the evolution of the mean RMSE over the 5-fold cross validation of the best models parameters with specific preprocessing parameters with respect to the iterations of our local search algorithm :

[2] except MLP.



(a) OLS regressor

(b) Ridge regressor

(c) KNN regressor

(d) Random Forest regressor

Figure 1: RMSE evolution with preprocessing parameters selection

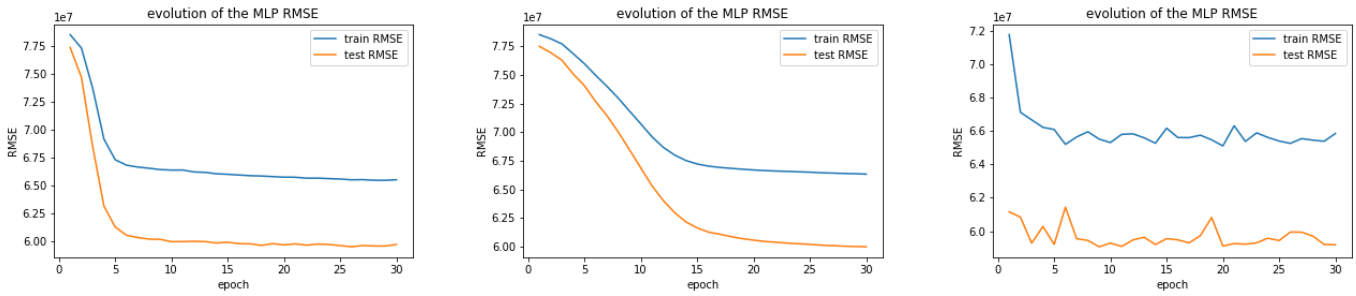
An observation we can make here is that the preprocessing parameters seem to have more impact on the non-linear regression methods since the range of reached RMSE are much larger for these ones. Another observation of the RMSE evolution but for the randomized search over the best preprocessing found for each regressor can be found in appendix and confirms this observation.

4.4 The exception : MLP

As explained before, using the `sklearn` implementation of MLP regressor in our tuning algorithm would have been too computationally expensive compared to the other ones. Our decision has thus been to not work on the exact same way as for the other regressors. Indeed, since the best preprocessing parameters for the other methods were always very similar to each other, we assumed that it meant that the corresponding preprocessing methods were the best ones to get as much relevant informations for our problem as possible.

We then created our own MLP regressor using torch and trained it on the so-preprocessed datasets. We then manually tried multiple ways of optimizing this MLP training by modifying the number of hidden layer sizes as well as the number of hidden layers. However, the more hidden layers we used and the higher their dimension was, the faster our training and testing RMSE was reducing but the final RMSE was not really getting improved. Trying alternative activation functions or trying different values for the Adams algorithm parameters did not get better results either.

Plots of the evolution of the training and testing RMSE over the epochs for some MLP instances trained on the same dataset as the optimal one for Random Forest stand on the following Figure :



(a) `lr` = 1e-3, `beta` =(0.9, 0.999), `n hidden layers` = 2, `hidden layers dim` = 100 (b) `lr` = 3e-3, `beta` = (0.8, 0.99), `n hidden layers` = 1, `hidden layers dim` = 100 (c) `lr` = 1e-2, `beta` = (0.7, 0.95), `n hidden layers` = 2, `hidden layers dim` = 50

Figure 2: RMSE evolution with the MLP training epochs

As you can observe on the above graphs, none of the trained MLP models was even close to reach the efficiency of our Random Forest regressor since the results were not going much bellow the value of 6e7 for the testing RMSE for any of the trained models. We therefore decided not to do any further optimizations on our MLP.

5 Conclusion

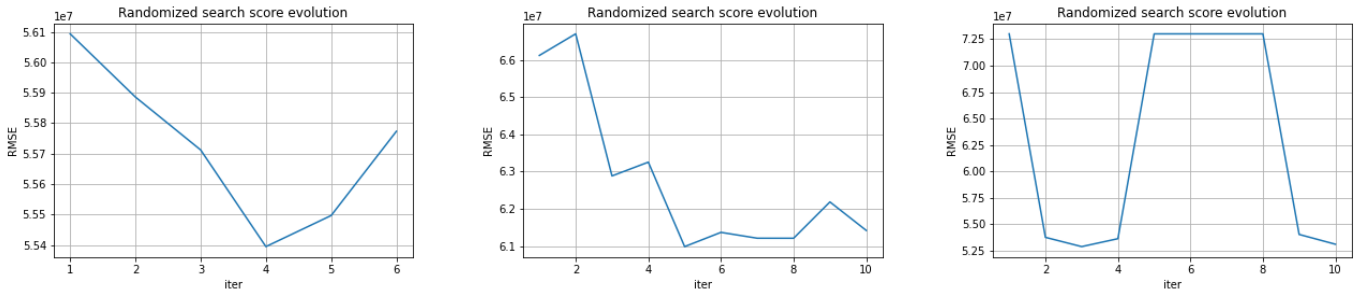
During the different steps of the project, we were able to try several techniques to optimize the results of our machine learning models. And in our case, the one using RandomForest was the most efficient in predicting the revenue of a movie. We also learned that each step of this project was important and that we should not ignore them at the risk of losing accuracy in the prediction of the final revenue. In the end, we are aware that we could not explore all the possibilities and that this could lead to better results.

6 Appendices

6.1 Optimal regression parameters

OLS	Ridge	KNN		RF		
/	alpha	weights	n neighbors	n estimators	min samples split	max features
/	10	uniform	10	100	3	sqrt

6.2 RMSE evolution with randomized search over the regressors parameters



(a) Ridge regressor (b) KNN regressor (c) Random Forest regressor

Figure 3: RMSE evolution with regressor parameters selection on the best preprocessing parameters found