Université catholique de Louvain

Project report

LELEC2870 Machine Learning

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

Martin, Florian (15001800)
Group 107

09/12/2022

1 Introduction

The goal of this project is to predict the revenue of a film based on many features provided. This is a regression problem and many regressors have to be used. This project will provide the results of a fully connected neural network, a knn regressor, a random forest regressor and a decision tree regresor. A first baseline will be provided with the well-known linear regressor. This report investigates what is the regressor that performs the best, how to use preprocessing to increase the overall performances, how to deal with the different features present in the dataset. Different type of models are tried but also different dataset, including more or less features and using reduction techniques to keep predictions quite fast enough. At the very end, a convolutional neural network is specifically used to address the problem of the embeddings and how to deal and extract information from such features.

2 Data Visualization

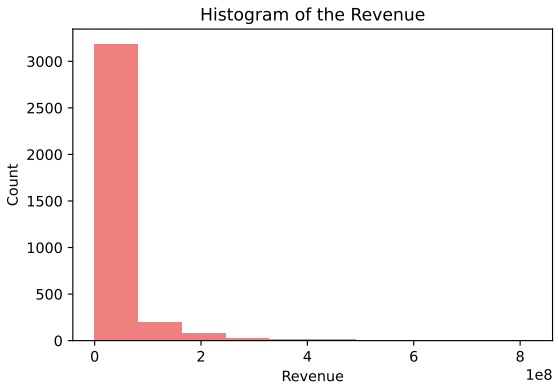

Figure 1: Histogram of the target "Revenue"

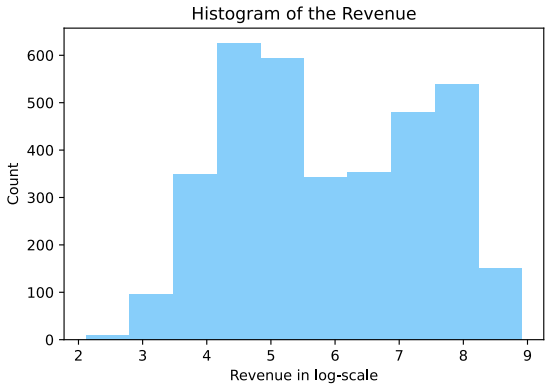

Figure 2: Histogram of the target "Revenue" with log-scale.

Figure 1 shows that the target "revenue" follows a logarithm distribution. Figure 2 highlights the target with a log scale, it shows that revenue do not follow exactly a normal distribution but a bimodal distribution. Actually, the base of the logarithm can be discussed. I tried log in base 10, 2 and **e**. Results are not much more affected except for the CNN architecture (detailled later on) where the log in base **e** seems to perform better.

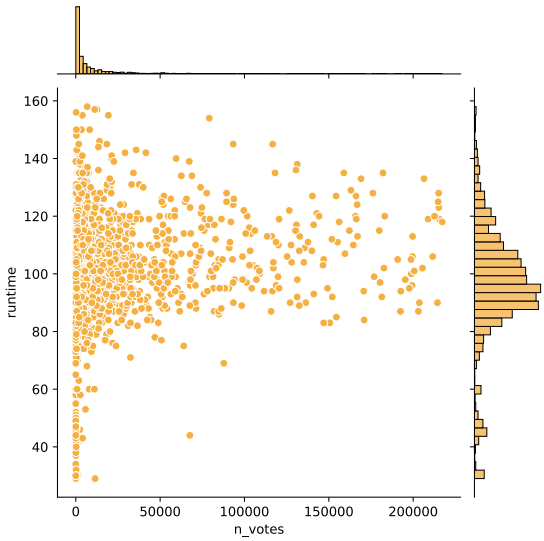

Figure 3: Before applying Power Transform

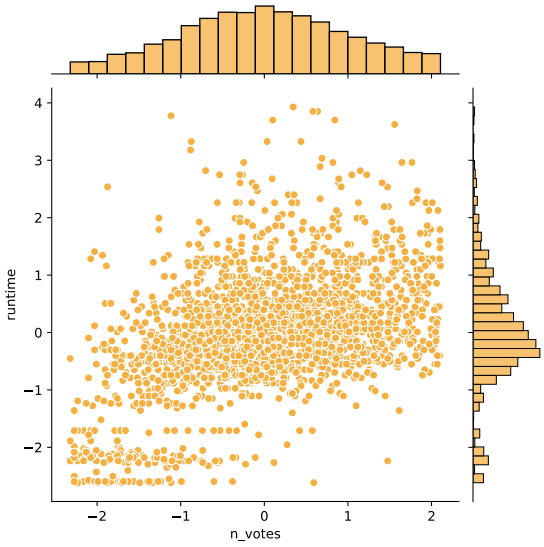

Figure 4: After applying Power Transform

Figure 3 shows that the distribution of the number of votes is following a logarithm distribution (as the target). In this way, transforming the feature into a log-scale can be helpful to unstack the samples. Figure 4 shows how power transform tries to fit the original distribution to a gaussian distribution but for such a feature (following a log-distribution), performing a simple logarithm operation provide the same result as it will also provide a gaussian distribution. Getting closer of a normal distribution can be helpful for the model to understand how the data is shaped altogether. Others jointplots highlight the same behavior and can be found in the annex whereas the runtime already follow naturally a gaussian distribution (kind of logical). Be careful that even if the individual distributions are gaussian, this doesn't mean that the joint distributions follow such a shape and it does not provides any information on it. Figure 5 that no relevant correlation between data is found. Actually, both jointplots and confusion matrix do not show any particular pattern between two features.

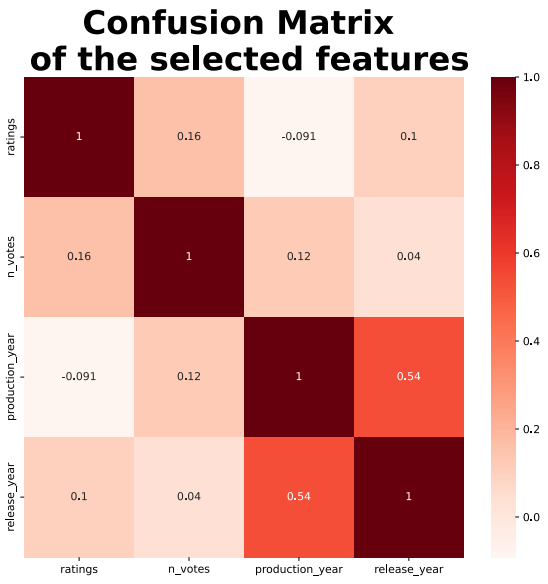

Figure 5: Correlation between features.

3 Preprocessing

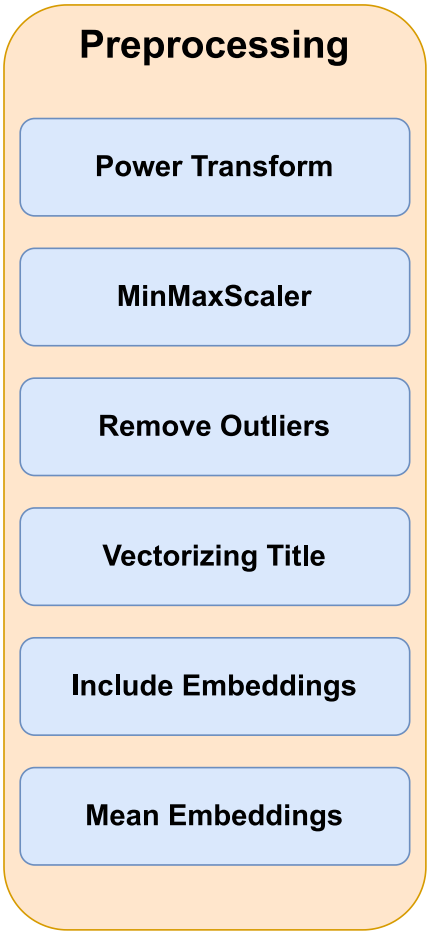

Figure 6: Steps available as preprocessing.

- **Power Transform** is a transformation of the dataset that try to fit the distribution of a feature to a normal distribution, as close as possible. This is one of the key point of my preprocessing such that it permits to obtain better results.
- **MinMaxScaler** is a scaling process that fits the scale of a feature between 0 and 1. This is the only scaler that will be used.
- **Remove Outliers** discard values in the dataset that do not fit in the well-known 3-sigma rule where 99.7% of the data is located.
- **Vectorizing Title** is turning text features into numerical features (same principle for "Include Description").
- **Include Embeddings** provides shaping of the two embeddings. First, it transforms the data from string to list and then add each value of the list to a column of the dataset.
- **Mean Embeddings** is also a shaping of the two embeddings features. It also transforms from string to list and then only add one column to the dataset which is the mean of the list.

4 Predictions Comparisons

This section is dedicated to show the result obtained on test set. The different techniques mentionned before will be compared between them through this section.

4.1 First Prediction

A function called "quick_preds" is used to make quick predictions without using K-Fold, just to give a first overview of the achievable predictions. The baseline that will be mine is to keep all the numerical features, discard NaN, remove the embeddings and non numerical features and to create a fetaure for each genre found in the initial feature : "genres". The revenue is set in log-scale. Multiple occurences of a film are discarded also.

Table 1 shows the different RMSE for each model. Of course, the RMSE is quite high because the revenue of a film can go close of 900 000 000 dollars of revenue. Apart the linear regressor, all the others regressor perform correctly and the Random Forest Regressor is the better regressor for the moment. However, the problem is that the RMSE value is way too high to get accurate predictions because its order is the same than the revenue. In this way, this is needed to find another relevant information in the dataset to help models for predictions.

Model	**RMSE**
Linear Reg.	2 577 966 377
Decision Tree	53 653 817
Random Forest	39 305 861
K-Neighbors	54 653 580
Multi Layer P.	63 125 773
Support Vector	54 063 742
Gradient Boosting	41 490 237

Table 1: First predictions with different models.

4.2 K-Fold and GridSearch

4.2.1 Predictions with K-Fold

Model	**K-FOLD 1**	**K-FOLD 2**	**K-FOLD 3**	**K-FOLD 4**	**K-FOLD 5**	**MEAN**	**STD**
Decision Tree	72 502 946	45 828 104	62 666 198	57 086 479	56 909 863	58 998 718	8 686 767
Random Forest	59 489 010	49 080 782	48 224 880	47 498 549	46 395 113	50 137 667	4 758 046
K-Neighbors	71 885 139	58 540 906	63 697 183	63 972 034	56 572 443	62 933 541	5 320 742
Multi Layer P.	79 268 867	65 061 856	72 393 652	72 716 965	66 247 175	71 137 703	5 118 993

Table 2: 5-Folds with different models.

Why the RandomForest Regressor performs the best ? Actually Random Forest is an ensemble technique that use multiple Decision Tree, it will fit many simple decision trees and averaging the result at the end. It will decrease in a way the overfitting because it tries to generalize with mutiple sub-models. The main problem is that the target is so much spread and that the features given are not relevant (contrary to the budget or the revenue after one week in the box office as the litterature exposed) such that it is difficult to identify pattern into the data that permits to provide good predictions (see section on data visualization). For example, KNeighbors Regressor won't perform well because it provides a local interpolation of the targets and because of the spread, it will make a lot of errors.

One way to verify more accurately the results is to use KFOLD. Here, a 5-fold is used with only the model shown in Table 2 (All the values are RMSE). One can notice that the Random Forest Regressor still outperforms the other regressors. All the regressors perform less better than before. This is mainly due to the lack of data. Each fold have less samples to train on it and the model underperforms compared to before because of that. Figure 7 shows the boxplot of the RMSE and R2 score for each model (except the MLP). The RMSE and R2 score is indeed better for the Random Forest Regressor and even the spread of these are better for this model. The two others have a large RMSE spread or a large spread in the R2-score. All this results permit to conclude that the Random Forest is the best regresor for the moment. As far as now, the models used were the models with parameters initialised by default. In order to improve the results, it is a good idea to search the best parameters using GridSearch and then checking the result with kfold using the parameters found for each model. However, after performing kfold and finding the best parameters the results are close to the one observed before and no benefits are observed.

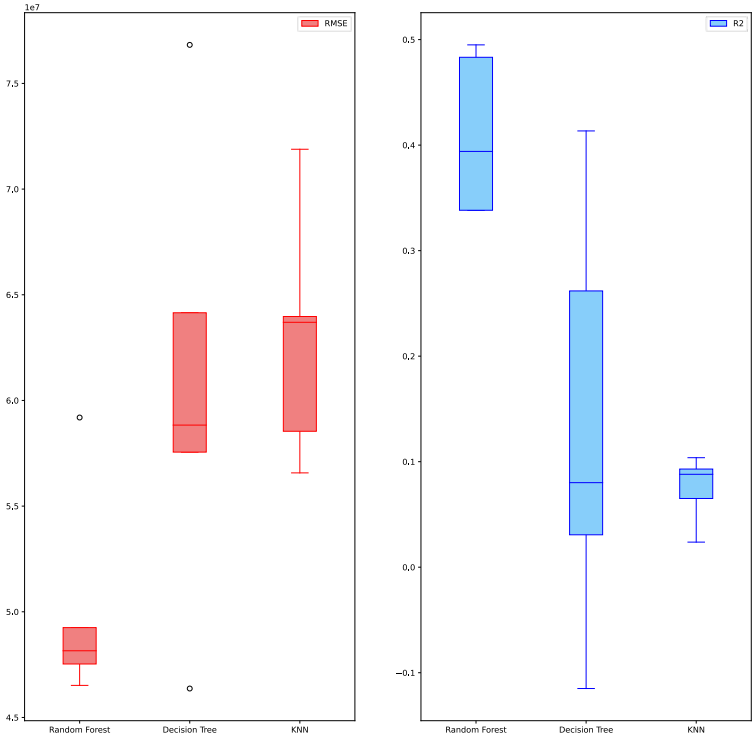

Figure 7: Boxplot of RMSE and R2-score for different models.

4.3 Predictions with feature reduction techniques

As mentionned in the section 3, one can transform a lot of non numerical feature to include it in the dataset and analyse the impact of them. In this way, vectorizing title and include embeddings are used to create a huge dataset, containing a lot of features (bigger than the number of samples). Actually, additional information is added when embeddings and titles are added to the dataset but it makes the dataset with too much features to be computed by a model, a lack of samples will be faced and curse of dimensionality is faced. Figure 8 shows the evolution with the number of principal components, of the cumulative explained variance of the principal component of PCA. A point is set on the curve highlighting a percentage of 80% of the explained variance ratio. This number is 141 and then the PCA will be performed with 141 features to keep and to feed into the ML models. This is quite high but still better than using the entire 7000+ features that were present before.

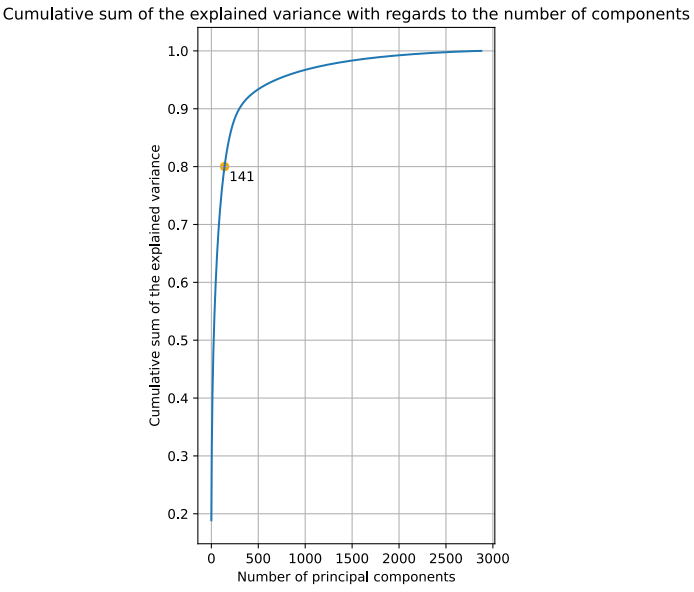

Figure 8: Cumulative explained variance of the principal components.

Figure 9 shows the boxplot of the RMSE and R2-score for each model with input features reduced by a PCA and Figure 10 shows the boxplot of the RMSE and R2-score for an AutoEncoder used to reduce the number of features to 20. Both highlight that the results are worse than before. The problem with the autoencoder is that it is built as a fully connected autoencoder and not a convolutional one and then quite often this does not provide low error while compressing-decompressing. Actually, reducing the number of features is mandatory but doing it means also loosing information and that is why the results are getting worse. One advantage shown on Figure 9 is that the standard deviation is lower such that the boxplots are less spread for the R2-score but not for the RMSE, in this way, the models without reduction techniques remain better in all points. This is why another method has to be used to include the information by the embeddings (see section 4.4).

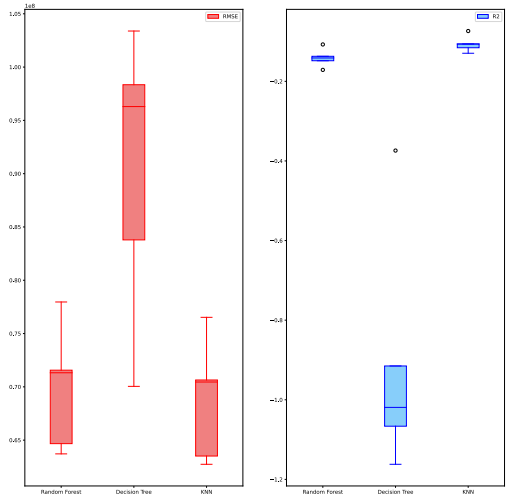

Figure 9: Boxplot of RMSE and R2-score for different models with PCA as feature reduction technique.

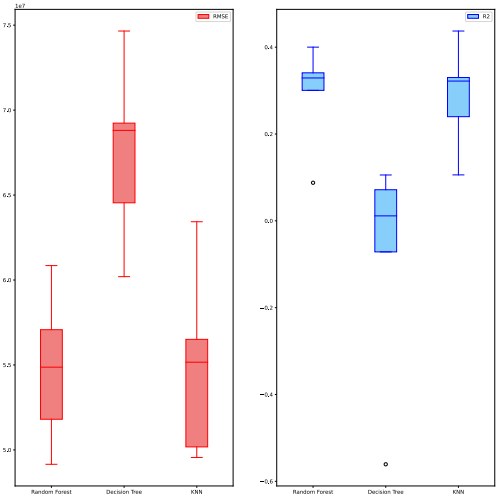

Figure 10: Boxplot of RMSE and R2-score for different models with autoencoder as feature reduction technique.

4.4 Predictions using embeddings

As seen in the section 3, **Include Embeddings** takes the text and image embeddings features to make an entire dataset. This dataset will be used to provide image-like to a convolutionnal neural network (CNN). The image embeddings contains 2048 samples per row and the text one contains 768 samples per row, resulting in a vector made of 2816 samples per row. In this way, each vector is reshape into an image like of shape 64 per 44 (arbitrarily chosen).

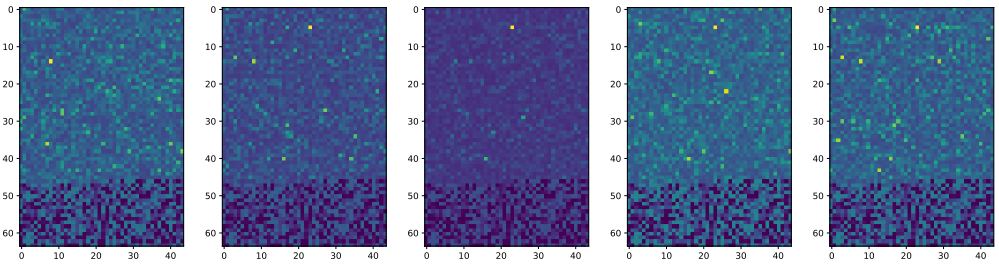

Figure 11: Embeddings represented in the form of a 2D-vector shown in an image style.

Figure 11 shows what such a 2D vector look like and one can notice that each image vector has an horizontal line which highlights the image embeddings shaped (above) and the text embeddings shaped (below). For a human, such

vectors do not provide any information but a CNN can detect underlying patterns and then performs a regression between these images and the revenue of a film. Once the model is trained, it is saved in order to perform prediction later on. Figure 12 depicts the architecture of the CNN. The training takes as criterion the R2-score. A callback, setup by the library keras, is used to keep and save the best model achieved during training because when the R2-score reaches values over 0.9, it is difficult to hope only increasing results.

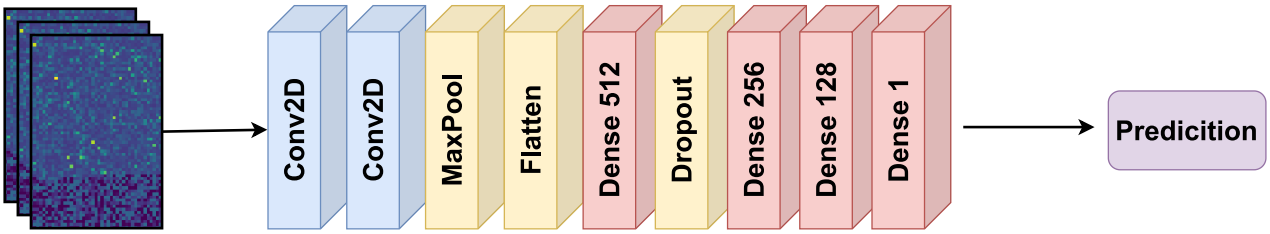

Figure 12: Convolutional Neural Network Architecture with embeddings as input vector.

The training time of CNN to get correct results (similar than the ones observed in the section 4.1) is up to few hours. Since the training is time demanding, there is no K-Fold that will be performed to obtain the performances and compare them with section 4.2.1. The RMSE obtained by the CNN architecture model is 61,275,737. Actually, this is not performing as I wish because it does not provide any clue that ensemble technique could bring better information. However, it is promising and it will be tried in the next section.

4.5 Predictions with Ensemble techniques

In order to combine the results obtained in section 4.2.1 with the RandomForest regressor that performs the best overall and the results obtained in section 4.4 with the CNN architecture, the predictions given by the two models is combined with a proportional factor α :

$$y_{predicted} = \alpha\, y_{CNN,predicted} + (1-\alpha)\, y_{RFR,predicted}$$

the proportional factor α can be tuned in order to give more weights to one or the other predictions from RFR or CNN. If $\alpha = \frac{1}{2}$, the two predictions have the same influence on the final prediction. Table 3 shows the results for the RFR, the CNN and the combining ensemble technique. This is the technique that will be used for the final predictions.

Model	RMSE
RFR	39 443 250
CNN	61 476 970
Ensemble	46 898 242

Table 3: Ensemble technique with RFR and CNN.

5 Conclusion

This report has shown multiple potential solutions in order to provide predictions about the potential future revenues of a film. However, this tasks is still very challenging even for the literature because no one has found accurate result for the predictions with the features that we had at our disposal. The best regressor remains the Random Forest one with around 39 000 000 of RMSE which is still huge compared to the revenue of the overall films but still a bit better than returning always the mean that would provide around 60 000 000 of RMSE. This report proposed way to quantify the results through KFOLD, ways of improving the results through many preprocessing steps, reduction techniques to decrease the complexity, and even a CNN architecture to catch patterns into the embeddings. However, as quite often, the simplest solution is the good one and using simply a Random Forest with scaling and Power Transform from sklearn is largely enough to provide the results obtained. Further analyses could have been done, for example, improving the autoencoder, trying deeper architecture for the CNN, getting bigger datasets that can be easily found on the internet, modifying the ResNet model that generates the embeddings to get more informations or to use it directly as a regressor by replacing the last dense layer of 2048 with a dense layer of 1 for example. Probably that other shape of the embeddings can be found and would provide better feature extraction.

6 Annexe

6.1 Jointplots before and after Power Transform