LELEC2870 - Project 2022

Predicting movie's revenues with machine learning models

Rousseau Mathieu, 67001800

UCLouvain
Belgium
23/12/2022

1 Introduction

Starting with a dataset containing data from *IMDb* [1] as well as *BoxOfficeMojo*, the goal of this project was to predict a movie's revenue in the USA with the help of machine learning models.

2 Exploratory data analysis and preprocessing

The first part of this project was to perform some data analysis to understand better our dataset and then preprocess it in order to enhance prediction results.

Our dataset does not contain that much features at a first glance. However, the principal difficultie was to manage the so-called **embeddings** that consist, for every observation, in large vectors of features (represented as float numbers). With *img_ embeddings* feature being a vector of size 2048 and *text_ embeddings* a vector of size 768. Keeping all these features would give us a very high dimensional dataset with a number of features greater than the number of observations. This situation should be avoided as it is prone to overfitting [2].

The first thing was to remove obvious unnecessary features like *title* which contains only distincts features or *img_ url* and *description* that we judged unnecessary as we already have the embeddings. We also noticed there were duplicated observations and removed them.

We then took a look at the missing values. There were only 4 values missing for the column *genre* consisting in only a tiny portion both datasets (*X1.csv* and *X2.csv*) therefore we simply dropped the observations with missing genres. However, there were respectively 264 and 106 values missing for the column *runtime* in *X1.csv* and *X2.csv*. These counting for more than 5% of our dataset, we did not take the risk of removing the observations containing these to avoid introducing a bias in our datasets. Looking at the dataset, it seemed like these values were missing randomly. We noticed that the shape of the distribution of the *runtime* feature is not too far from a Normal. As a consequence, we decided to impute the missing values with the mean of this feature.

Looking at the distributions of the other variables, we noticed that *n_ votes* and *revenues* were heavily right skewed. Because of that, simply removing outliers outside $1.5 * IQR$ [3] could lead to removing a lot of datas. To fix the skewness, we took the log of these variables resulting in a much more homogenous distribution. We also noticed that the variable *is_ mature* only contained a unique value so we deleted this column.

The *genres* feature consists for every observation in a list of the maximum 3 genres that most accurately represent the given movie. Because there were not that much unique genres, we decided to **one-hot encode** them. However, the *studio* feature consist in more than 300 unique studios so we did not one-hot encode them to avoid exploding the dimension of our dataset. We choosed to **count-encode** them. The idea behind count-encoding is to replace each value by the number of times it appears in the column.

To manage the embeddings, we decided to run a PCA against them and to keep a certain amount of variance. For example, keeping 80% reduced the dimension of the *img_ embeddings* from 2048 to 125 features and of the *text_ embeddings* from 768 to 3. We also tried to not to run the PCA but the result of our machine learning models were worse because of overfitting.

Finally, we standardized our dataset to put every features on the same scale and to have better results with the linear regression and multi-layer perceptron models (MLP). For example, in the MLP, it allows the gradient descent to converge more quickly. We ensured to only compute the transformation on the training set ...

1. International Movie Database.
2. Situation where the model learned results of training data instead of generalizing and therefore cannot predict any new data.
3. Interquartile range.

3 Feature selection

Even if we run a PCA on the embeddings, we still have a lot of features in our dataset. We performed feature selection in order to decrease the number of variables in our dataset.

3.1 Correlation and mutual information

Since we do not only deal with linear models we cannot only rely on the correlation matrix to select features because it cannot detect any nonlinear relationships between variables. We still decided to consider it because it can be useful for the linear regression.

The mutual information can detect nonlinear relationship between variables and therefore is a filter of choice for nonlinear models.

Since we had to deal with hundred of features (even after running a PCA), we did not investigate the correlation matrix and the mutual information matrix in details. Instead, we choosed to keep a certain percentage of most significant features.

We also decided to implement a **recursive feature elimination** (RFE) method even though the biggest downside of it is that it is computationally intensive. The idea behind RFE is to start with all the features and then successively removing one feature at at time until reaching a desired number of features.

4 Model selection

We tried severall models : linear regression, K-nearest neightbors (KNN), multi-layer perceptron and a random forest. For each model except the linear regression, we choosed to tweak severall well chosen hyperparameters and used a searching algorithm along with cross-validation to find the best hyperparameters for each model. A common searching algorithm is *Grid Search* that will test every possible combination of hyperparameters. However, it can be computationaly intensive if the search space is wide which is often the case since we don't want to miss the optimal parameters. Another approach is to use *Random Search*. With this approach, we can also draw hyperparameter configurations from distributions besides discrete sets and the random search will randomly select a combination of hyperparameters among the ones defined instead of doing an exhaustive search. It allows to explore a wide range of hyperparameter configurations with time-efficiency. Looking at the result of the random search, we can identify the area where the results are promising, tighten our ranges and perform a random search again. Although this algorithm being less computationaly intensive, it can be time consuming of running and running again the random search to find the best set of hyperparameters. A third approach and the one we tried in this project is to perform a *Bayesian Search* which is basically similar to random search but...

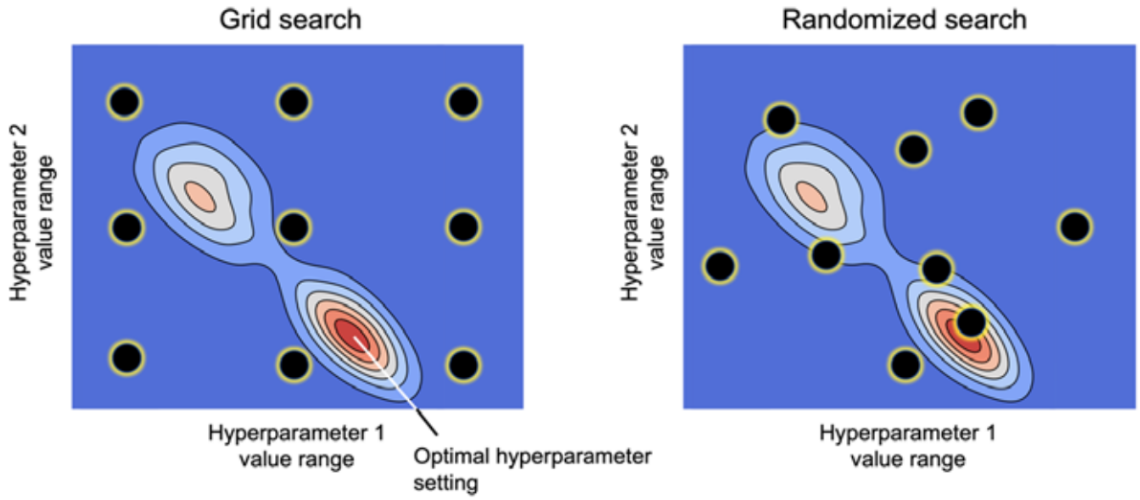

Figure 1 – *Comparison of grid search and randomized for sampling* 9 *different hyperparameter configurations*

Appendix