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.

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 from X1.

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 distribution and 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 $1.5 * IQR$ [2] 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. We did not removed outliers for the other features as the result were not convincing (even slightly worse) without them.

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 suffer of high-cardinality. Indeed, this feature consists in more than 300 unique studios so to avoid exploding the dimension of our dataset (and therefore avoid the curse of dimensionality), we replaced the less representative studios (the ones that represent less than 1% of the dataset) by a special name : "other".

To manage the embeddings, we tried to run a PCA and Kernel PCA (with a radial based function kernel) against them to keep the most significant features. The later has the advantage of recognizing non-linear pattern and gave slightly better results for our model. At first, we wanted to keep a certain amount of variance but we noticed we got better results for our model by being more drastic. We then chose to keep only the five most important principal components for each of the embeddings.

Finally, we split our dataset (*X1*) into a training and a testing part and we standardized it 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

1. International Movie Database.
2. Interquartile range.

quickly. We ensured to only compute the transformation on the training set and applying it on the testing set to avoid data leakage. We ended up with a dataset of 62 features.

3 Feature selection

Even if we ran a (K)PCA on the embeddings, we still have a lot of features in our dataset and not enough data in order for our models to generalize well. We performed feature selection in order to decrease the number of variables in our dataset. We investigated multiple technique to select features : correlation, mutual information and recursive feature elimination. The problem with the first one is that it is only relevant for linear models since a low correlation does not mean the absence of relationship for non-linear models. We could have selected features highly correlated with the target for the linear regression but we found that even for this model we had slightly better results with mutual information. The last one on the other hand is a wrapper method and therefore use a model to compute a score and remove unnecessary features accordingly. The "problem" is that it is computationnaly intensive and very dependent on the chosen model.

3.1 Mutual information

We chose to use the mutual information to select the features. Mutual information has the advantage over correlation to detect nonlinear relationships between variables and therefore is a filter of choice for nonlinear models. Our strategy was to minimize the redundancy between inputs variables and maximize relevancy between inputs variables and the target variable (*minimum redundancy, maximum relevance*). The idea is that even if 2 features are highly relevant, we shouldn't add both of them to our subset of features if they are highly correlated since it would increase the model complexity and could cause overfitting.

To minimize the redundancy, we computed a normalized mutual information matrix (fig. 5) and decided to remove for each group of redundant features, the feature that share the less information with the target. As you can see on the heatmap (fig. 5) in the appendix, *release_year* shares a lot information with *production_year* and *img_feature1* shares also a significant amount of information with *text_feature0*. Looking at the mutual information of each feature with the target variable (fig. 6), we decided to remove *production_year* and *text_feature0* since they share less information with the target.

We did not chose a specific set of features afterwards. Instead, we decided to train our different models with multiple subsets of features. Each subset is a selection of the k best feature based on the mutual information of the explanatory variables of the training set with the target. More about that in the next section.

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 (5 folds) to find the best hyperparameters for each model. We used the well known **Grid Search** for most of the models. There exists other searching algorithms like **Random Search** and **Bayesian Search**. These last two have lower computational cost due to the fact they make less iterations (the number of iterations is chosen by the user) while finding a good model. However we prefered to rely on them only for very computational intensive model and/or model that required searching over a huge hyperparameters space like Random Forest.

For each model, we ran a 5-fold cross-validation to avoid overfitting and we computed the validation score as well as the test score that's been computed on a training part of *X1* that has obviously not

been used to train any of our models.

4.1 Linear regression

The linear regression has no hyperparameters to tune so we just ran a cross-validation for each set of features we decided to keep. The results show us that the best linear model is the one with the **35** best features selected.

	scores	
#features	validation rmse ($)	test rmse ($)
5	1.629e08	1.154e08
10	9.760e07	1.029e08
15	8.146e07	9.603e07
20	6.848e07	8.768e07
25	6.936e07	8.719e07
30	6.844e07	8.576e07
35	**6.869e07**	**8.564e07**
40	7.006e07	8.672e07

Table 1 – *Cross-validation results of linear regression model*

4.2 K-Nearest Neighbors

For the K-Nearest Neighbors model, we need to take care of adjusting the number of neighbors hyperparameter. Indeed, a value too low will induce overfitting and a value too high will induce underfitting. Moreover, because this algorithm suffers a lot from the curse of dimensionnality, we ran it with a smaller subset of features : $[3, 5, 7, 9, 11, 13, 15]$. We grid searched for a number of neighbors between 1 and 50 first but as you can see on the figure below, this model overfits with a number of neighbors approximatively less than 7. Then, its performance slightly decrease as this number increases.

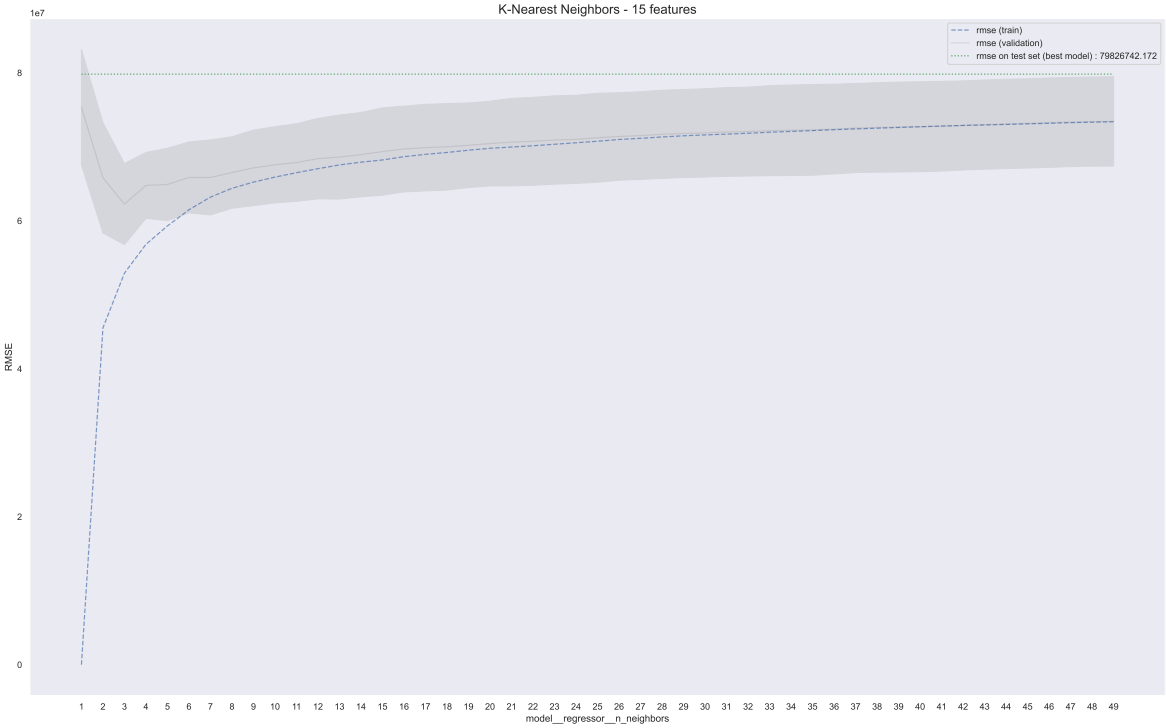

Figure 1 – *RMSE score with respect to the number neighbors for the best K-Nearest Neighbors model*

We chose to run this algorithm again but this time starting from 7 neighbors up to 50

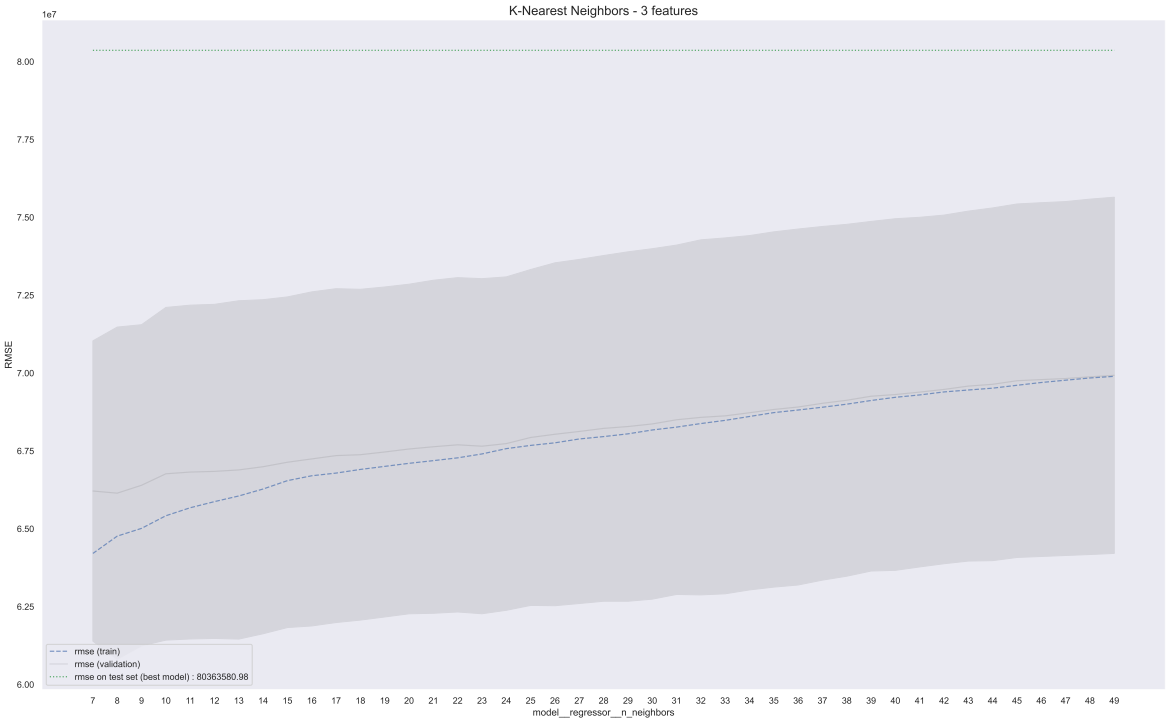

Figure 2 – *RMSE score with respect to the number neighbors for the best K-Nearest Neighbors model*

You can see the results of this second pass on the following table,

	scores	
#features	validation rmse ($)	test rmse ($)
3	**6.614e07**	**8.036e07**
5	6.642e07	8.166e07
7	6.556e07	8.103e07
9	6.572e07	8.176e07
11	6.584e07	8.292e07
13	6.542e07	9.271e07
15	6.586e07	8.408e07

Table 2 – *Cross-validation results of K-Nearest Neighbors model*

The best model is the one with the **3** best features and has 8 neighbors.

4.3 Multi-Layer Perceptron

Since the multi-layer perceptron has the ability to set weights to zero, we can make the hypothesis that the feature selection will have less influence. Therefore, we wanted to test that model on a greater number of features. We also tested with all the features. We tested different number of layers and neurons per layers as well as different activation functions. We can observe that two layers gives the best results and adding more layers to the network tends to decrease the model performance.

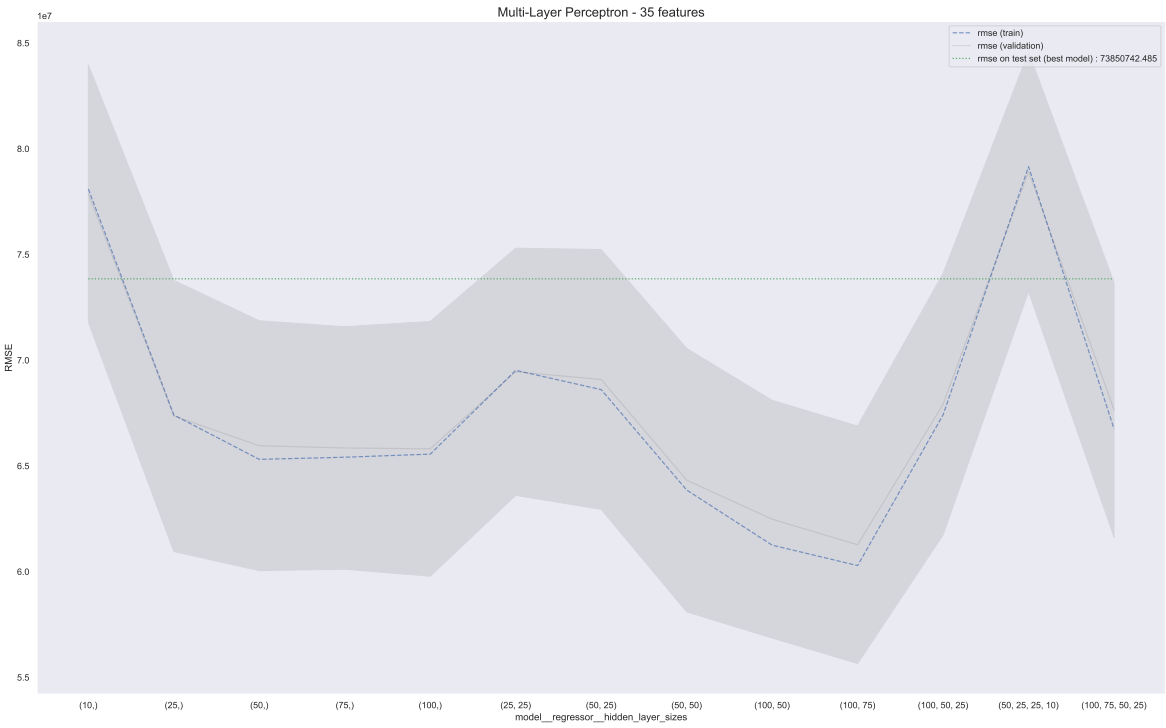

Figure 3 – *RMSE score with respect to the number of hidden layers and their sizes for the best multi-layer perceptron model*

The results are summarised in the following table,

	scores	
#features	validation rmse ($)	test rmse ($)
5	6.649e07	8.101e07
15	6.254e07	7.469e07
25	6.124e07	7.491e07
35	**6.147e07**	**7.385e07**
all	6.041e07	8.124e07

Table 3 – *Cross-validation results of Multi-Layer perceptron model*

The best model is the one with the **35** best features and has its first layer with 100 neurons and its second layer with 75 neurons. It uses a $\tanh(\cdot)$ activation function and has a starting learning rate of 0.001.

4.4 Random Forest

A random forest is an ensemble technique that combines multiple decision trees. As a consequence, it has a better generalization performance than a single decision tree due to randomness which decrease the model's variance. Another thing to note is that it is little sensitive to outliers. For this model, we performed a **Bayesian Search** with 30 iterations. Maybe we didn't find the best model but it still should be a good model. Moreover, considering the number of hyperparameters combinations we wanted to try, it allowed use to win much computational time.

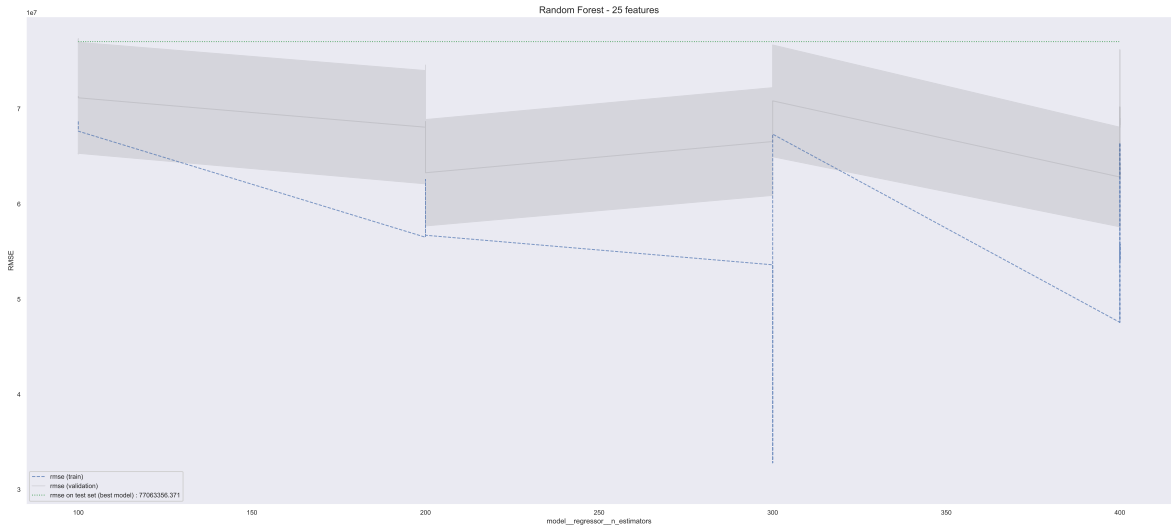

Figure 4 – *RMSE score with respect to the number trees for the best random forest model*

	scores	
#features	validation rmse ($)	test rmse ($)
5	$6.465e07$	$7.943e07$
10	$6.342e07$	$7.813e07$
15	$6.316e07$	$7.743e07$
20	$6.254e07$	$7.711e07$
25	**6.283e07**	**7.706e07**

Table 4 – *Cross-validation results of Random Forest model*

The best model is the one with the **25** best features. It uses 400 trees.

5 Conclusion

The RMSE is pretty big for all our models but it was pretty much expected since our target variable is distributed over a large range of values and we lack of features that could explain more the target variable (features like budget for example). The main challenge was to deal with the embeddings and there are probably clever ways to manage team and obtain better results as a consequence. Our personnal computer have limited computational resources and therefore we did not have the time to fully explore the hyperparameter space of every model.

Anyway, if we summarize the results of all our models, the best model that we keep to predict the revenues on the X2 dataset is the Multi-Layer Perceptron with 35 features selected according to mutual information. Using the score obtained on the testing set, we expect an RMSE of $7.385e07$\$ on $Y2$.

Appendix

Mutual information matrix

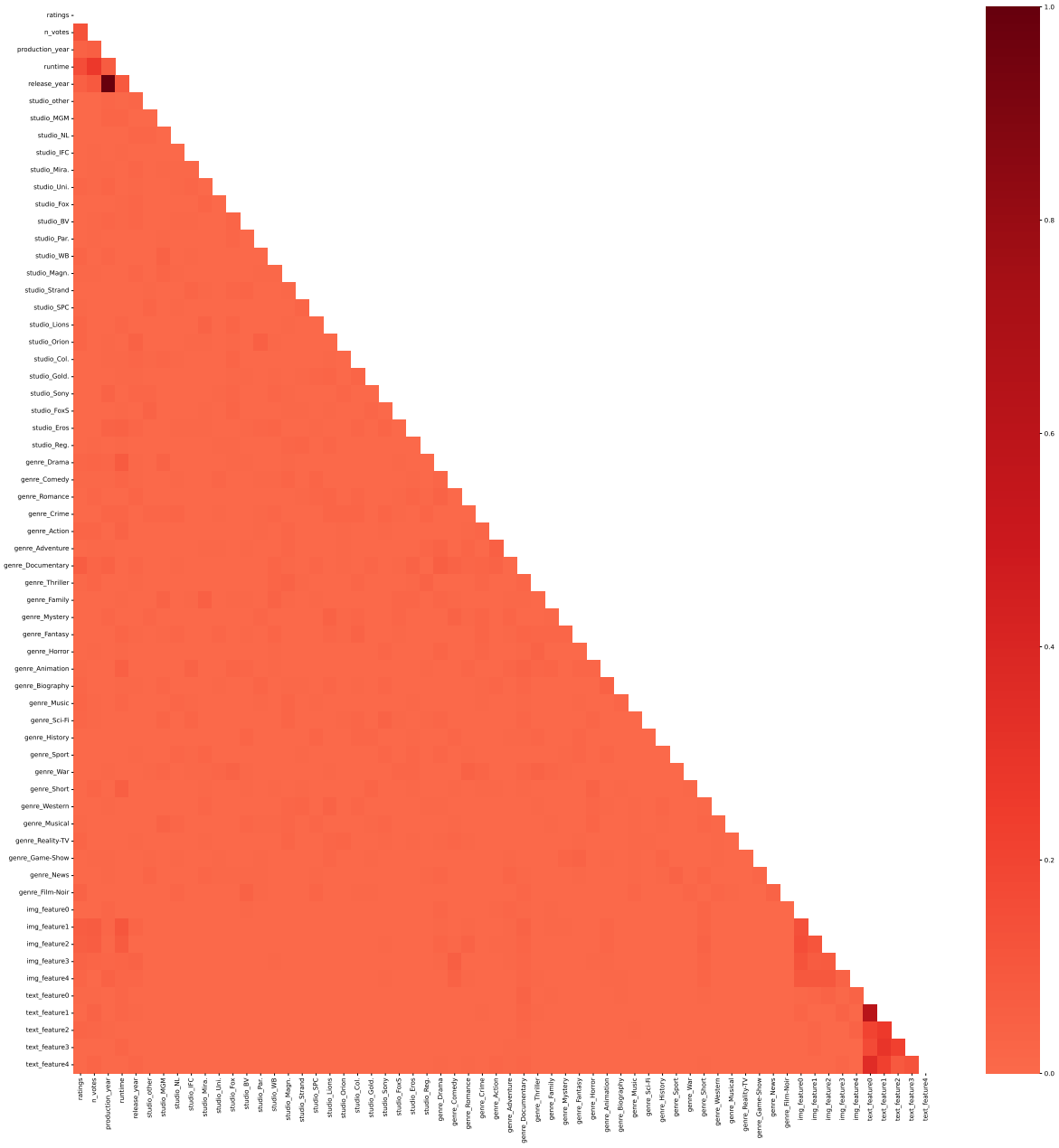

Figure 5 – *Mutual information matrix*

Mutual information with the target variable

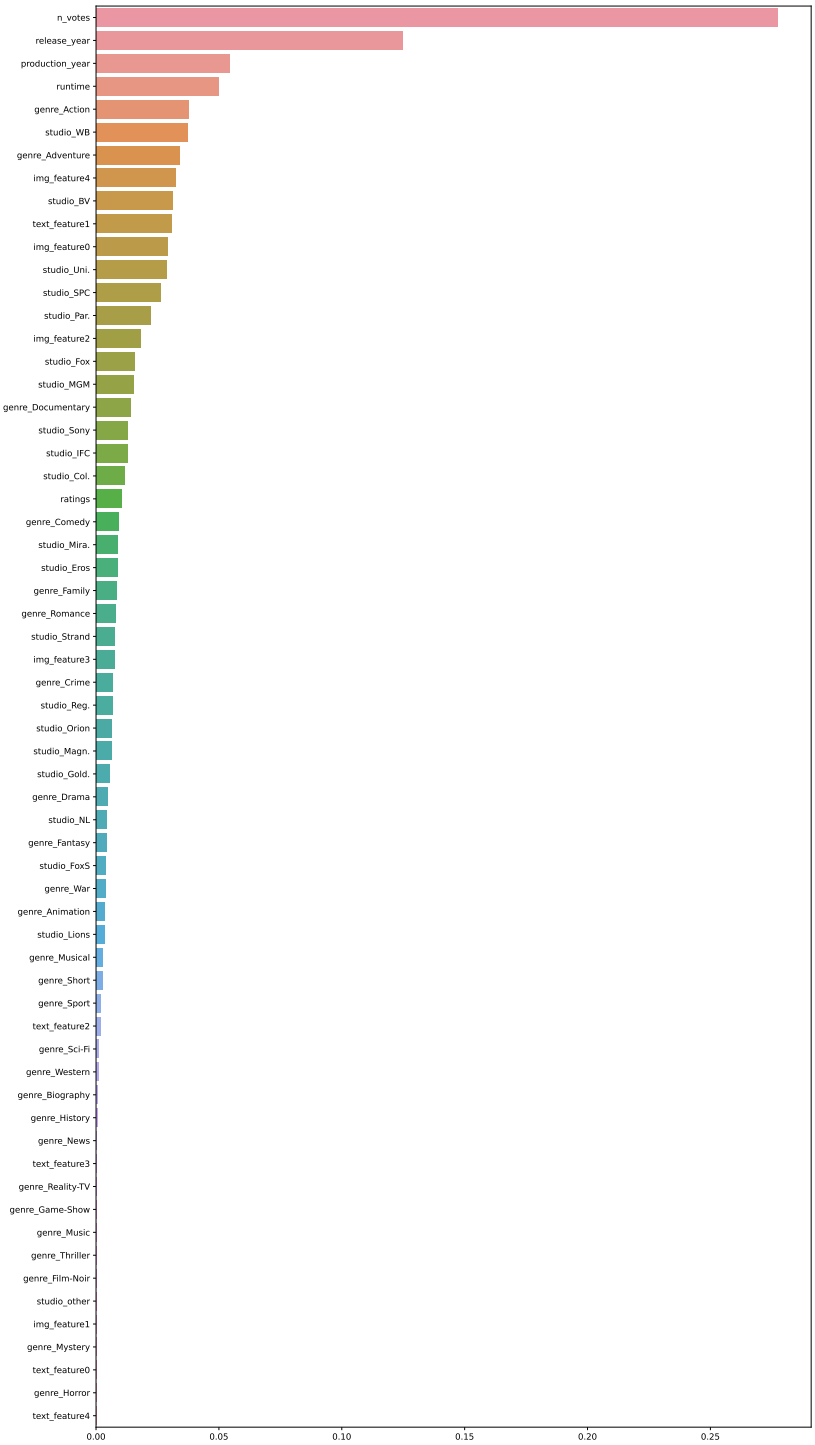

Figure 6 – *Mutual information of each feature with the target variable (revenues)*