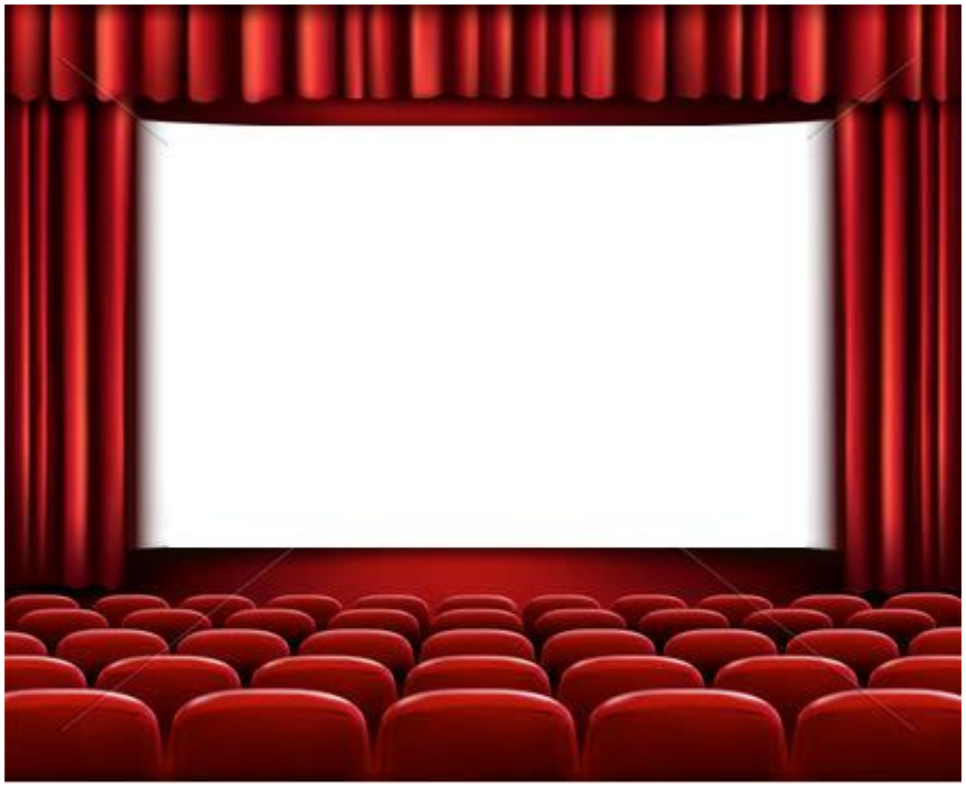
Predicting movies' U.S. box office



Guillaume Drugeot - Adrian Lemaigre - Vasileios Papanikolau

IEOR 242 - Paul Grigas

Motivation

For each upcoming movie release, movie theaters have to plan which rooms and for how long they will project it. Optimizing these parameters enable those establishments to create the most value with higher earnings but also for the customers with less crowded rooms. However, some long-awaited movies turn out to be disappointing and after the first days, people read the ratings and decide not to see it, which creates empty rooms for the theater. On the other hand, some movies may turn out to be unexpectedly well received by the public and it will become very difficult for everyone to see them.
As a result, we thought that the problem of predicting the U.S. box office of movies was an effective way of addressing this problem. Indeed, it is related to the number of tickets sold and a direct indicator of the movie's success.

Data collection

For our purposes, we decided to scrape data from the RottenTomatoes website, as they have several details for each movie. It was also an opportunity to practice acquiring raw data and having to process it rather than just downloading the IMDB database from Kaggle. To do that, we used a Python framework called Scrapy which enabled us to get metadata for movies from 2000 to 2018.

Our methodology was:

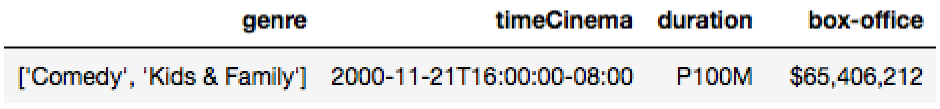
- scrape Wikipedia for the titles of movies released during that period
- use these titles to generate urls in a RottenTomatoes format
- use Scrapy to get the HTML code of the corresponding pages
- look up the fields of interest in the HTML code.

We get the following raw features:

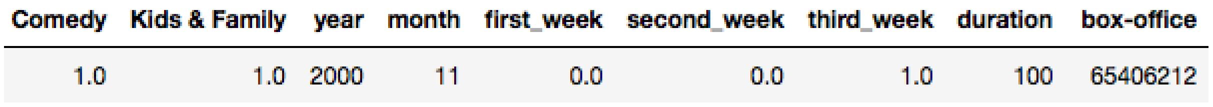
Title	Studio
Genre	Duration
Date of first projection in the U.S.	Synopsis
Content rating	Rating by critics
U.S. box office	

Data cleaning & Feature engineering

Since our features were obtained through parsing of a HTML code, some of them were in a format difficult to analyze. The following figures depict the first row of our dataset before and after the cleaning of these features.

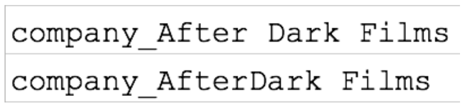


genre	timeCinema	duration	box-office
['Comedy', 'Kids & Family']	2000-11-21T16:00:00-08:00	P100M	$65,406,212



Comedy	Kids & Family	year	month	first_week	second_week	third_week	duration	box-office
1.0	1.0	2000	11	0.0	0.0	1.0	100	65406212

We also had the "studio" feature, with 517 unique values. We first notice that often the same company was spelled differently. Solving that issue and regrouping the different studios by their 'parent' studio allowed us to get 290 unique values.



company_After Dark Films
company_AfterDark Films

This was still too much (it would require 290 binary features), we decided to look up the biggest studios and create two binary features: "Major" and "Mini-Major" studios as per a classification found on Wikipedia.

Additionally, we wanted to get exogenous variables such as the weather and economic indicators. We gathered data about the average temperature and precipitation in the U.S. from the National Climatic Data Center at the National Oceanic and Atmospheric administration. We also downloaded unemployment time series from the Bureau of Labor Statistics.

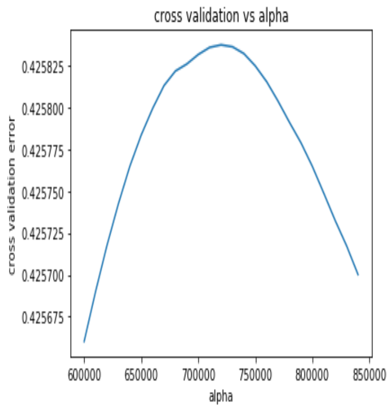
Then with the modified dataset, we could try different models to understand the relationship between our independent variables and the box-office.

We also noted that some movies actually had a release date before 2000, had incoherent box office values or had missing values for studios. These rows were dropped from the dataset which resulted in a reduction from 3,573 rows to 2,888 rows.

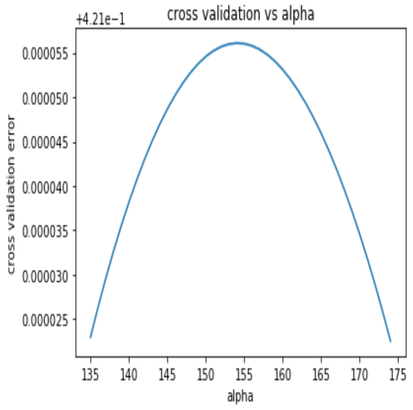
Predicting box-office using linear models

As a first approach to predict box-office, we used linear models. More specifically, we used linear regression, lasso, ridge, and elastic net. We performed hyperparameter tuning by performing grid search and 5-fold cross validation over the basic parameters. We visualized our results with the following figures:

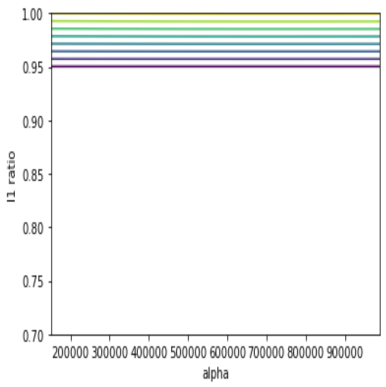
Ridge Regression



Lasso Regression



Elastic Net

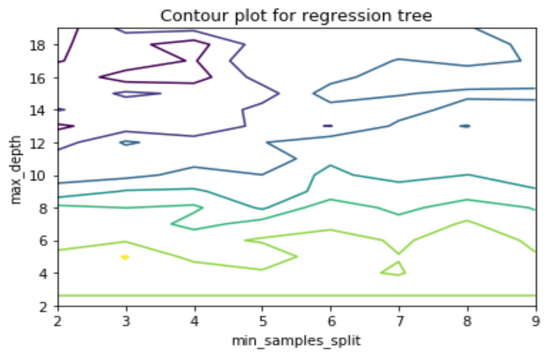


We can see that best regularization parameter for ridge is 154 with OSR^2 = 0.42, for lasso it is 72 * 10^5 with OSR^2 = 0.41 while elastic net basically reduces to lasso.

Predicting box-office using tree-based models

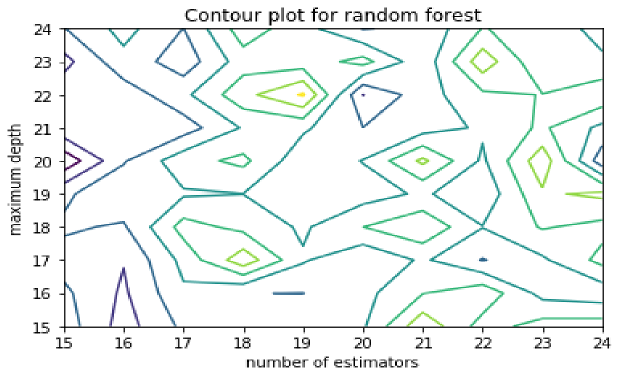
As a second approach to predict box-office, we used tree based models. More specifically, we used basic tree regressors, random forest and AdaBoost with a regression tree as a base estimator. We performed performed hyperparameter tuning by performing grid search and 5-fold cross validation over the basic parameters. We visualized our results with the following figure:

Regression Tree



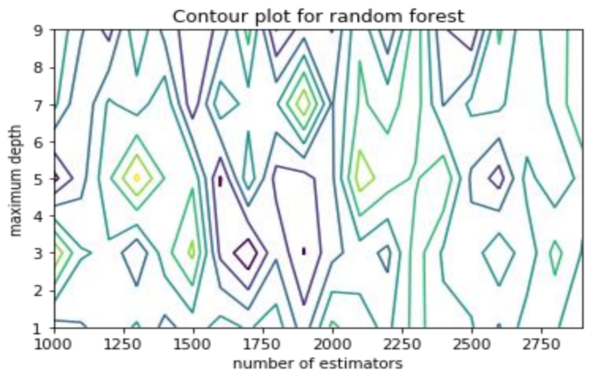
For regression tree we performed grid search over the max depth of the tree as well as the minimum number of data points in a bucket to perform split. We conclude that the best max_depth and min_bucket parameters are 5 and 3 respectively with OSR^2 = 0.36. Tree graph is included in the appendix.

Random Forest



A word on the hyperparameters we chose to grid over. For our tree we chose to generate 200 tree in each iteration with minimum samples on each leaf (equivalent to minimum in bucket) = 5 and minimum samples split = 10. We chose to iterate on maximum tree depth and maximum features chosen. The criterion we chose was MSE and we did not include minimum impurity gain. The best combination of hyperparameters thus were (maximum depth, number of estimators) = (22, 19) with OSR^2 = 0.497. Feature importance graph is included in the appendix.

Ada-Boost



Regarding the hyperparameter tuning for AdaBoost we have the following: first, we have to choose your base estimator, which is going to be a regression tree for us, and thus we have to chose all the parameters of the tree. Then, we have to choose the number of trees that we sequentially are going to train, the loss function that is going to define the weights and the learning rate.. We set the parameters of tree to be the predefined besides the maximum depth. Also, we set the parameters of the boosting algorithm as the default ones except the shrinkage that we set it to 0.01 and the number of trees to be boosted. Then performing hyperparameter tuning, we ended up with the best model of max_depth = 5 number of trees = 1500 with an OSR^2 = 0.45

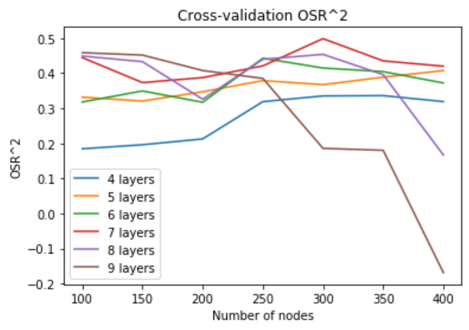
Predicting box-office using deep-learning

In order to get a more reliable prediction of the movies' box-office, we decided to implement more complex algorithms such as Neural Networks, at first with all the features and then reducing the dimensionality of the dataset to prevent overfitting. In both cases, the input data is scaled before being fed to the Neural Network (and after applying PCA).

Neural Network with all the input features

In order to obtain the best results possible, we used 3-fold cross-validation to optimize the number of hidden layers (from 7 layers to 9) as well as the number of nodes per layer (between 200 and 400 nodes). The number of folds is relatively low so that the computation does not take too much time.

The cross-validation yielded the following results:

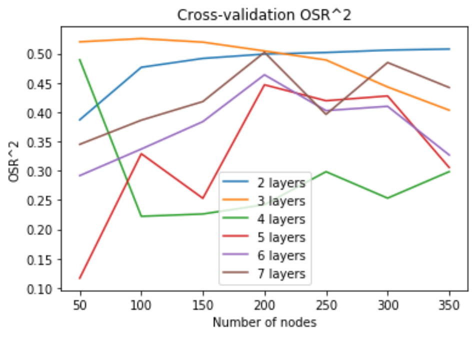


As we can see, the maximum is reached for 7 layers and 300 nodes. The OSR^2 of the best model on the test dataset is **0.495**, the MAE is 27,751,705 and the RMSE is 48,493,358.

Preprocessing the dataset using PCA

As we can see Neural Network have a similar performance on the test dataset than Random Forest, therefore we tried to use PCA to preprocess the data in order to increase the OSR^2 on the test dataset.

First, we computed the singular values of the covariance matrix of the training dataset in order to have an idea of how many components to use. The result was less than 5 components. Therefore, we used cross-validation to determine the number of hidden layers, number of nodes and number of components (between 10 and 50).



We find that the best Neural Network has the following parameters:

- Number of components: 40
- Number of hidden layer: 3
- Number of nodes per layer: 100

This plot represents the cross-validation OSR^2 of cross-validation (ie the mean from each fold). The performance of this model on the test dataset is

$OSR^2 = 0.490$, MAE = 28818310 and RMSE = 48720018. This is similar to the result we got for Neural Network without PCA.

Conclusion

In order to help movie theaters and studios in predicting the success of a movie, we have scraped movie data from Rotten Tomatoes for films from 2000 to 2018. For these movies, we engineered multiple features and added the external information such as the rain level and the unemployment rate at the time of the release.

With this data, we trained multiple Machine Learning algorithms, from simple ones such as Linear Regression and LASSO Regression to more sophisticated ones such as Random Forest, AdaBoost and Neural Networks. In the end the ones that perform the best where Random Forest and Neural Network using all features.

The best OSR^2 on the test dataset were around 0.49 for the two best models. Thus we can hypothesize that the current features can only explain half the variance of the dataset. Taking into account other features such as the director and the lead actors may help get a more precise prediction.

References

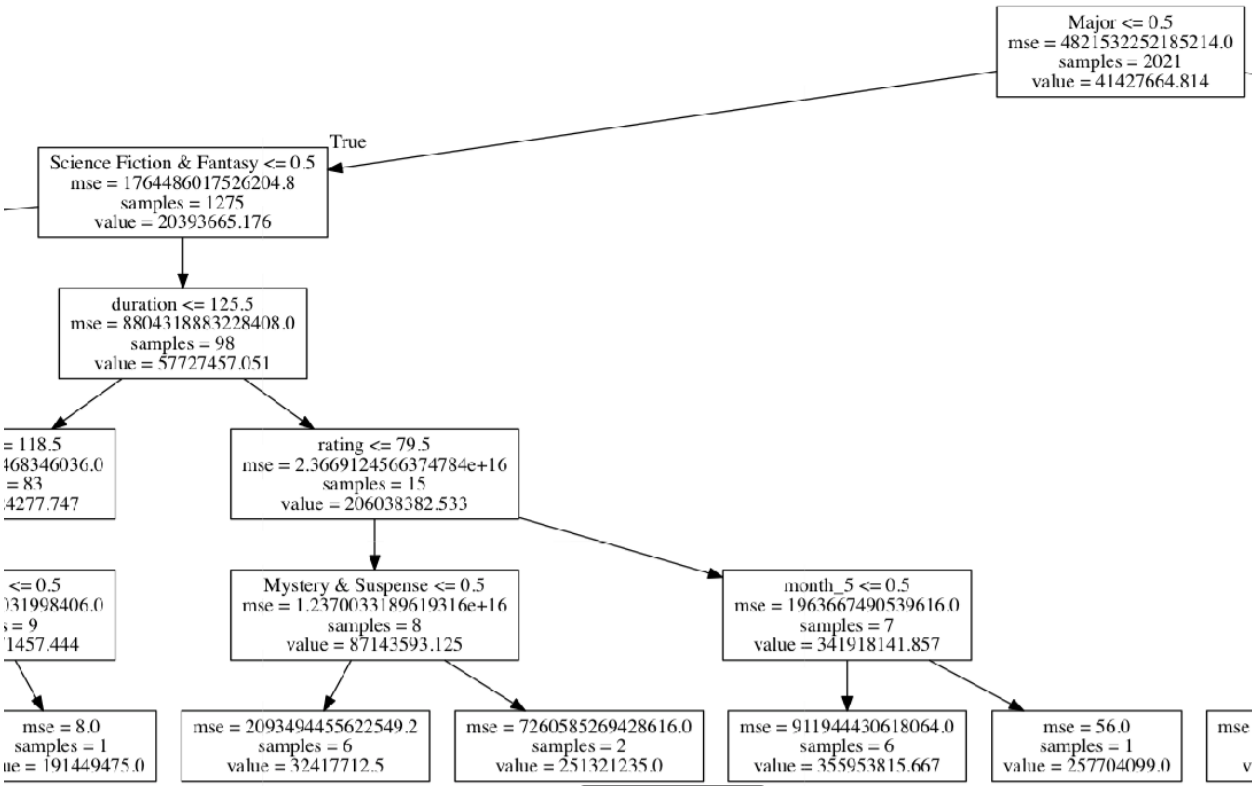
- Movies' list for 2000 - 2018: https://en.wikipedia.org/wiki/2000_in_film (for every year)
- Information about studios: https://en.wikipedia.org/wiki/Major_film_studio
- Unemployment data: https://data.bls.gov/timeseries/LNS14000000
- Weather data: https://www.ncdc.noaa.gov/cag/national/time-series

APPENDIX

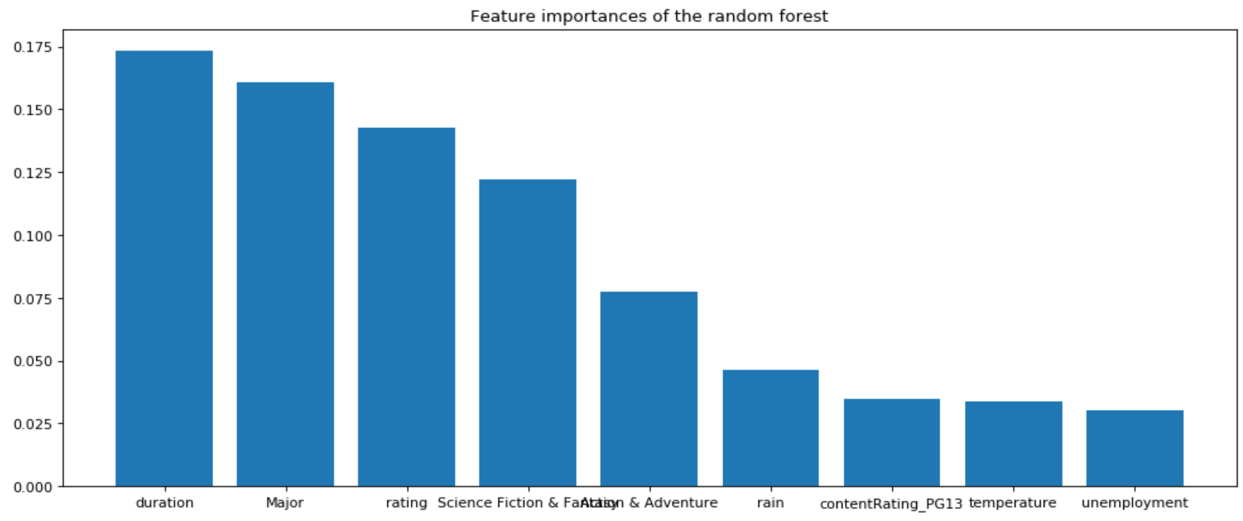
Complete list of features

'name', 'duration', 'synopsis', 'rating', 'box-office', 'first_week',
'second_week', 'third_week', 'Comedy', 'Kids & Family',
'Action & Adventure', 'Mystery & Suspense', 'Science Fiction & Fantasy',
'Drama', 'Television', 'Horror', 'Animation', 'Anime & Manga',
'Art House & International', 'Romance', 'Sports & Fitness',
'Musical & Performing Arts', 'Western', 'Documentary', 'Classics',
'Special Interest', 'Cult Movies', 'Faith & Spirituality',
'Gay & Lesbian', 'temperature', 'unemployment', 'rain',
'contentRating_NC17', 'contentRating_NR', 'contentRating_PG',
'contentRating_PG13', 'contentRating_R', 'year_2001', 'year_2002',
'year_2003', 'year_2004', 'year_2005', 'year_2006', 'year_2007',
'year_2008', 'year_2009', 'year_2010', 'year_2011', 'year_2012',
'year_2013', 'year_2014', 'year_2015', 'year_2016', 'year_2017',
'year_2018', 'month_2', 'month_3', 'month_4', 'month_5', 'month_6',
'month_7', 'month_8', 'month_9', 'month_10', 'month_11', 'month_12',
'Mini-Major', 'Major'

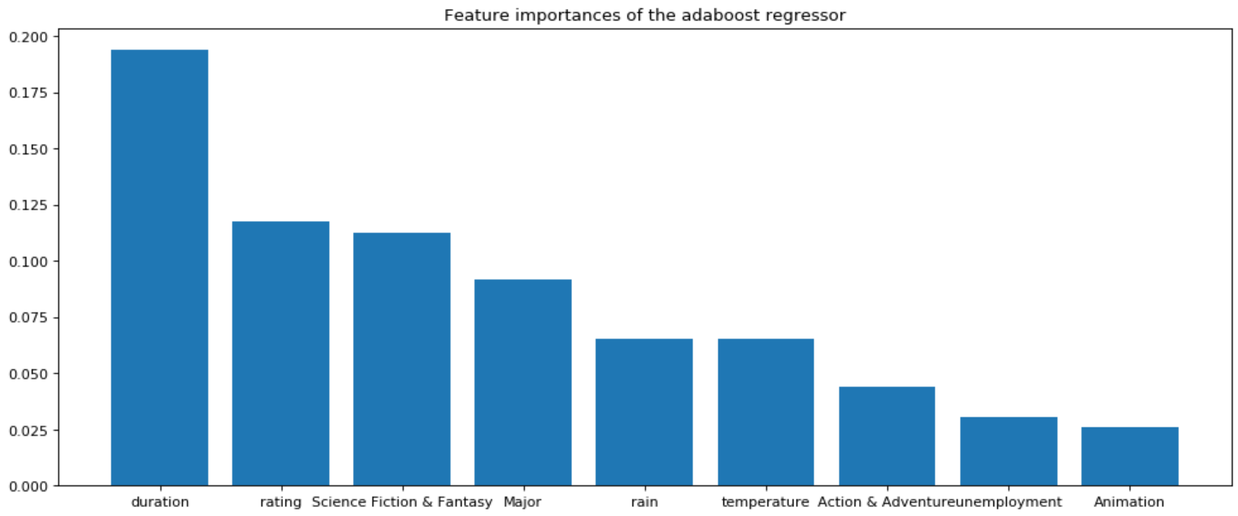
Best Tree (partial view, complete tree in the file)



Features importances - Random Forest



Feature importances - Adaboost


Feature importances of the adaboost regressor
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
duration
rating
Science Fiction & Fantasy
Major
rain
temperature
Action & Adventure
unemployment
Animation