Report: TMDB Box Office Prediction with Deep Neural Network

Jorge Melo
Xavier Pinho

November 2019

1 Introduction

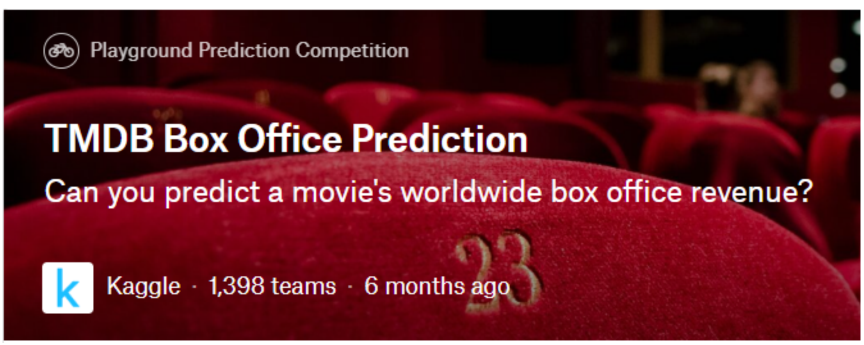

Figure 1: Kaggle competition header

The film industry is on the rise after breaking record revenue in 2018 with over $ 41 billion worldwide.[4] In the last 7 years, revenue has risen five times in a row. Disney, for example, accounted for about $ 7 billion in revenue, according to the same source.
But what makes a movie profitable and popular? What is the influence of the cast on the success of the movie? And the director? For some, the presence of a favorite actor will be enough for them to go to the movie theaters. For others, the quality of a trailer will influence decision making.
Obviously, there is a huge industry interest in predicting and detecting these patterns and this is where the challenge arises in Kaggle titled "TMDB Box Office Prediction". The ability to predict a movie's revenue before its release is important to avoid unnecessary financial risk, but forecasting is not easy due to the complex relationship between movie data and its revenue.
Artificial Neural Networks (ANNs) have proven their value in predictive systems in a wide range of fields, from social media analysis [13], [10] to medical imaging

[3] to recommendation systems [12] . Our goal with this work is to explore the potential of Supervised Learning (SL) and ANNs in predicting movie revenue, as well as the most determinant characteristics of the problem, using the database provided by the challenge. Subsequently, we explore the explainability of the generated model, which will support the discussion of the results. This work is divided into three parts: data processing, learning model and explainability.

2 State-of-the-art

Forecasts of film-generated revenues have been explored for quite long, recognized as a highly complex problem, considered even, for some, as unpredictable [5], several researchers have developed models for predicting financial success, initially using statistical bases for forecasts [14]. There are approaches that use post-film release data [11], but pre-release predictions are a way more challenging and valuable problem for the industry.

In our study, we explored the use of Neural Networks (NN) to predict the financial performance of films prior to release. This is a regression problem where revenue comes in USD.

2.1 Related Work

Previous research done in the area of predicting box office success have applied different techniques such as neural networks [14], [7], [17] and statistical Bayesian [6] and linear regression modeling techniques [8]. The most recent work about box office success prediction using ANNs is by Yao Zhou, et al [16], who exploited the potential of poster images' features combined with other movie-related data, using a multimodal deep neural model.

To obtain useful information from movie posters, they constructed a Convolutional Neural Network to extract representative features, which first pre-trained with movie posters as its input and movie box-office revenues as its output. Subsequently, the CNN was incorporated into the multimodal Deep Neural Network. In this study, they considered factors such as genre, duration, star value (number of "likes" on Facebook page of each actor/actress), social commentary (professional movie critics and viewer reviews), rate (number of votes and rating) and budget. All the data is represented as numeric data. Also, they adopted the original numerical and discretized form of movie box-office revenues as output and parameters of the multimodal DNN were updated according to the cost functions of these two outputs.

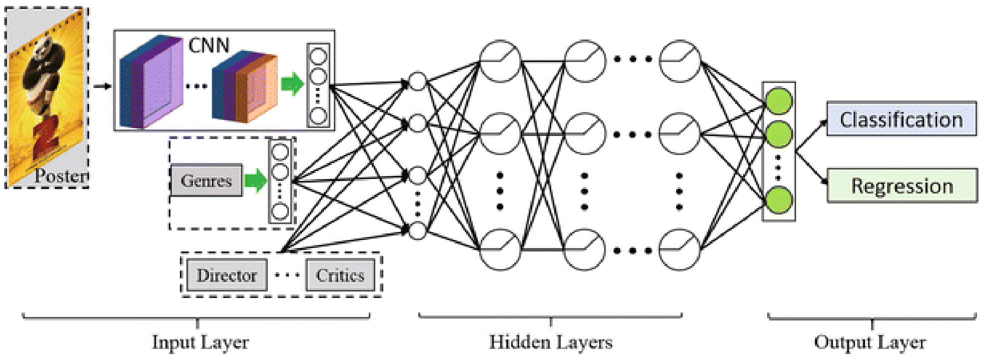

Figure 2: Yao Zhou, et al. model architecture

Yao Zhou was able to outperform previous models' performance with this architecture, showing that posters combined with movie-related data could increase the accuracy of models trying to solve movie box office revenue prediction.

3 Methods

The aim of this project was to predict movie box-office revenues given movie-related data. To achieve this, a deep neural network for movie box-office revenues prediction is built. As our main goal is to scrutinize the impact of different movie-related data on box-office revenues prediction, data processing is firstly addressed.

3.1 Data

This competition on Kaggle provided metadata on more than 7000 movies from The Movie Database [1]. Data samples include cast, crew, genre, plot keywords, budget, posters, release dates, languages, production companies, countries and more.

3.1.1 Check NaNs

First of all, we merged both train (known target value) and test (unknown target value) datasets and looked at the number of NaNs in the dataset. The results are shown in the figure 3. From here we knew we needed to do something about belongs_to_collection, homepage, tagline and keywords in order to deal with the high number of NaNs.

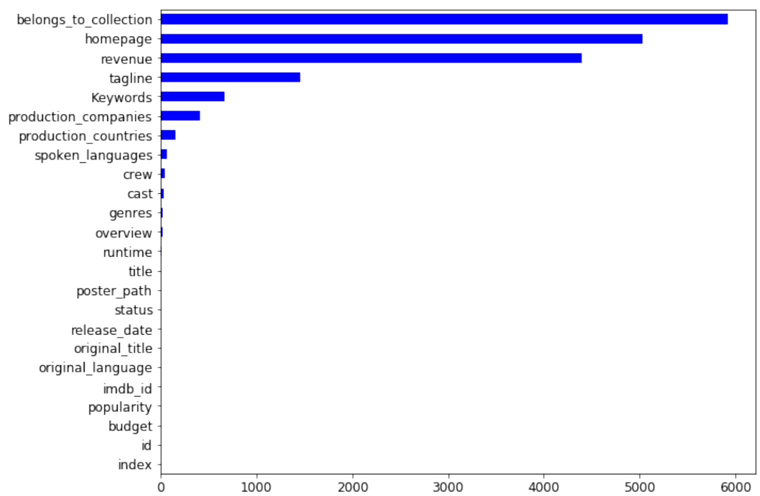

Figure 3: NaN values distribution

3.1.2 Feature Analysis and Processing

We analysed each feature individually by looking at the number of instances, it's mean, standard deviation, quartiles, minima, maxima, skewness and it's correlation with other features then we processed these features in order to get useful data for our model.

The results for the revenue before processing are shown in figure 4. We can see a minimum value of 1 for the revenue, which doesn't make much sense, since it's almost impossible for a movie to get 1$ in total box office revenue. In fact, this was a problem related by most of Kaggle competitors and there are several values for the revenue which are way too low. The best solution found for this problem was to assume that values under 100 were in millions (x 1000000) and left values under 1000 were in thousands (x 1000). Then we adapted the data into a logarithmic scale, in order to lower the skewness and amplify the relative differences between movies with diferent revenues. The analysis results are shown in figure 5.The same logic was applied for budget and the results are shown in figures 6, 7, 8 and 9.

```
***[revenue]***
describe :
count     3.000000e+03
mean      6.672585e+07
std       1.375323e+08
min       1.000000e+00
25%       2.379808e+06
50%       1.680707e+07
75%       6.891920e+07
max       1.519558e+09
Name: revenue, dtype: float64
skew :
4.542285301660683
*****[ corr : revenue ]*****
revenue       1.000000
budget        0.752965
popularity    0.461460
runtime       0.216380
index         0.000610
id            0.000610
Name: revenue, dtype: float64
*****[revenue]*****
```

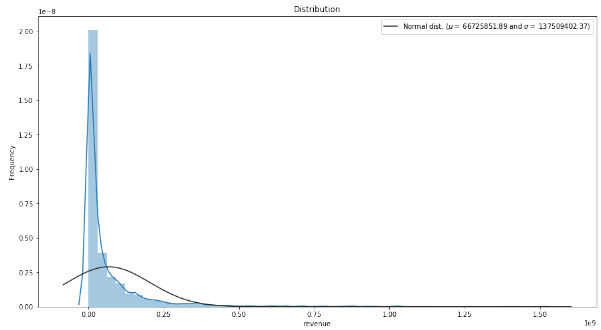

Figure 4: Revenue analysis results before processing

```
***[revenue]***
describe :
count    3000.000000
mean       16.270743
std         2.659430
min         7.247081
25%        14.815338
50%        16.693831
75%        18.133508
max        25.297977
Name: revenue, dtype: float64
skew :
-0.5581491025532077
*****[ corr : revenue ]*****
revenue       1.000000
budget        0.494160
popularity    0.281417
runtime       0.190119
index         0.003615
id            0.003615
Name: revenue, dtype: float64
*****[revenue]*****
```

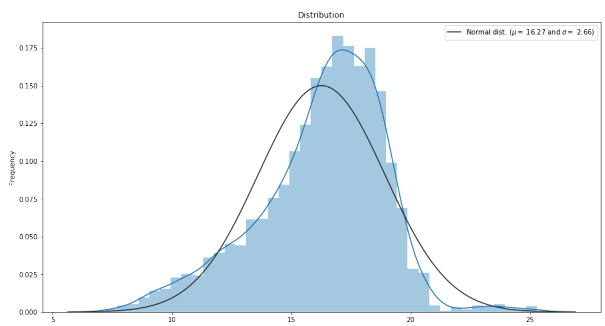

Figure 5: Revenue analysis results after processing.

```
***[budget]***
describe :
count     7.398000e+03
mean      2.260146e+07
std       3.694867e+07
min       0.000000e+00
25%       0.000000e+00
50%       7.500000e+06
75%       2.800000e+07
max       3.800000e+08
Name: budget, dtype: float64
skew :
2.892441747364616
*****[ corr : budget ]*****
budget        1.000000
revenue       0.494160
popularity    0.340145
runtime       0.216251
index         0.005024
id            0.003845
Name: budget, dtype: float64
*****[budget]*****
```

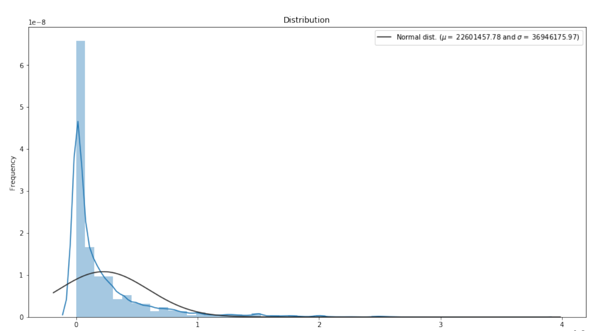

Figure 6: Budget analysis results

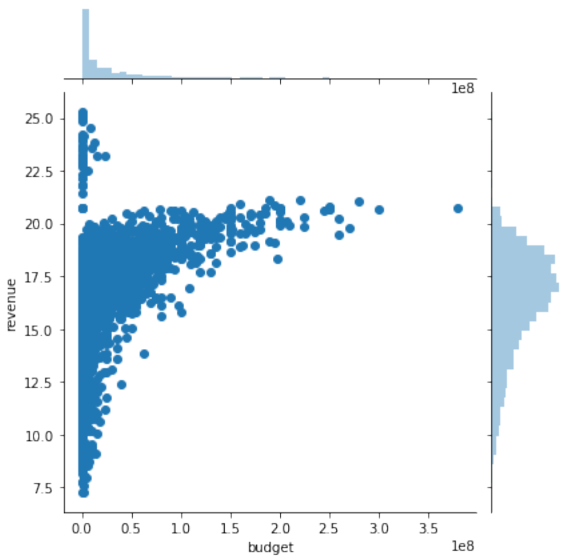

Figure 7: Revenue vs. Budget

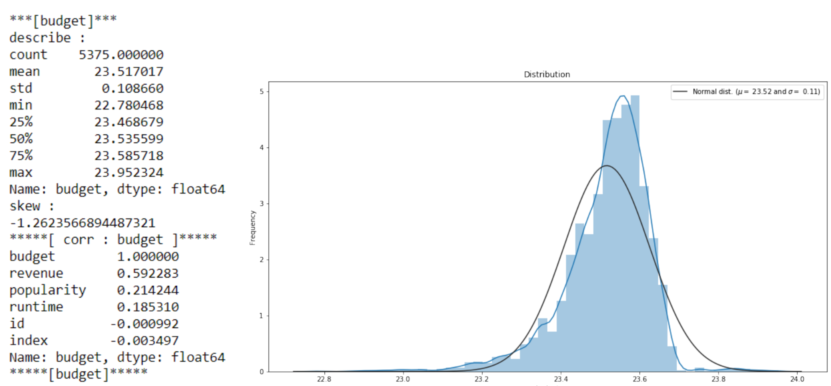

Figure 8: Processed budget analysis results

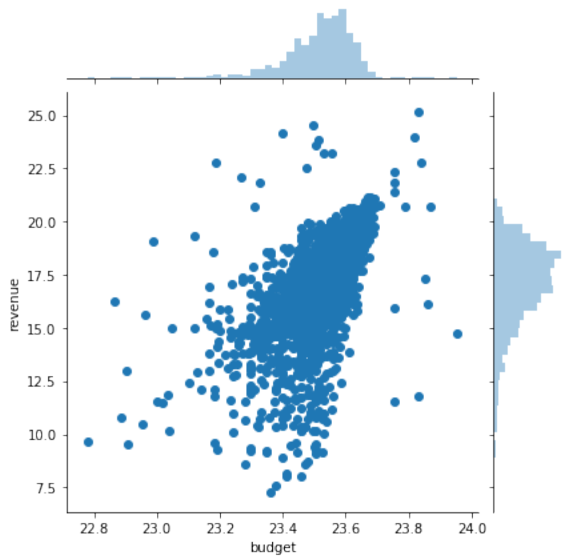

Figure 9: Revenue vs. processed budget

While processing the release_date feature, we added 2000 to values under 18 and 1900 to values above 18 and under 100, as suggested by other competitors, since there are no movies in this dataset prior to 1900 (as expected). Then we created two additional attributes: release_date_year and release_date_dayofweek, in order to have more frequency data and better express change patterns. The analysis of release_date_year and release_date_dayofweek is shown in figures 10, 11, 12 and 13. From here we can conclude that our dataset has a lot more movies from the most recent years and there isn't much correlation with revenue and the prefered week day to release movies is Friday, although it doesn't seem to correlate with revenue.

```
***[release_date_year]***
describe :
count    7398.000000
mean     1999.677210
std        15.369115
min      1918.000000
25%      1992.000000
50%      2004.000000
75%      2011.000000
max      2017.000000
Name: release_date_year, dtype: float64
value_counts :
2013    335
2014    320
2015    312
2011    311
2010    306
       ...
1922      1
1918      1
1921      1
1929      1
1924      1
Name: release_date_year, Length: 98, dtype: int64
```

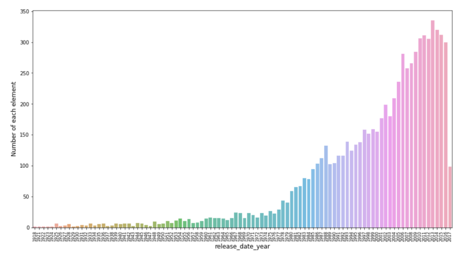

Figure 10: Release_date_year analysis results

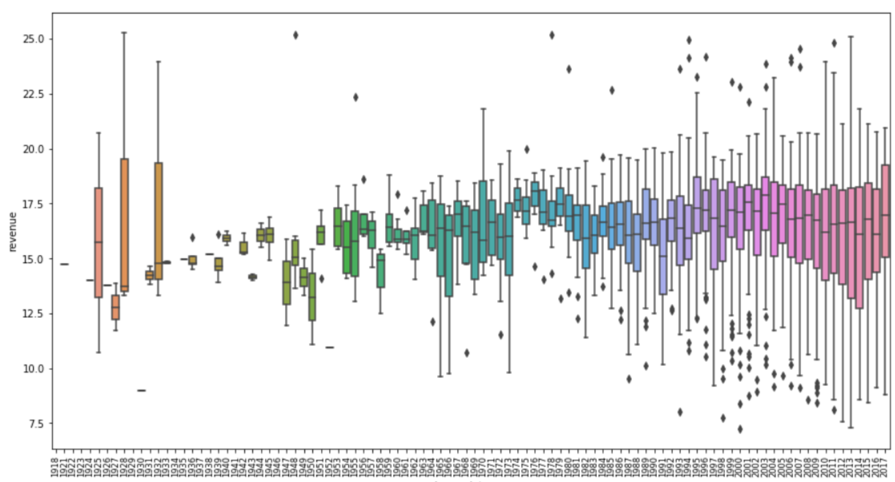

Figure 11: Revenue vs. release_date_year

```
***[release_date_dayofweek]***
describe :
count      7398
unique        7
top         Fri
freq       3244
Name: release_date_dayofweek, dtype: object
value_counts :
Fri    3244
Thr    1436
Wed    1234
Tue     470
Sat     400
Mon     317
Sun     297
Name: release_date_dayofweek, dtype: int64
```

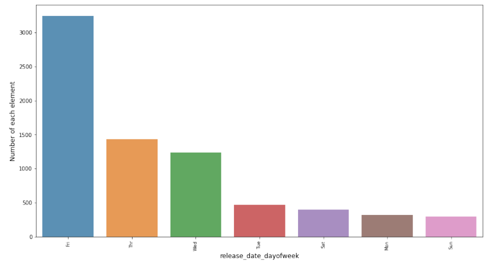

Figure 12: Release_date_weekofday analysis results

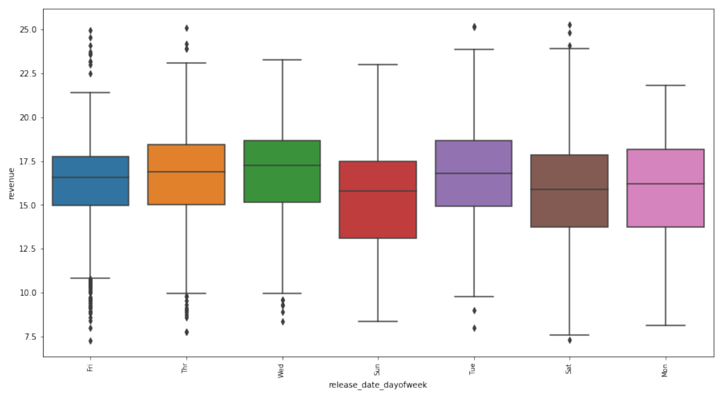

Figure 13: Revenue vs. release_date_dayofweek

In order to process the belongs_to_collection feature, we decided to replace it by a binary feature called inCollection, that is 1 if the movie belongs to any collection and 0 if it doesn't. The results are shown in figures 14 and 15. We can see that most movies don't belong to any collection and there seems to be some correlation with revenue, with movies that belong to some collection having more chances of having bigger revenues.

```
***[inCollection]***
describe :
count    7398.000000
mean        0.200189
std         0.400169
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000
Name: inCollection, dtype: float64
value_counts :
0    5917
1    1481
Name: inCollection, dtype: int64
```

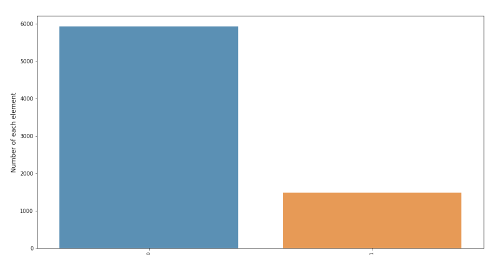

Figure 14: inCollection analysis results

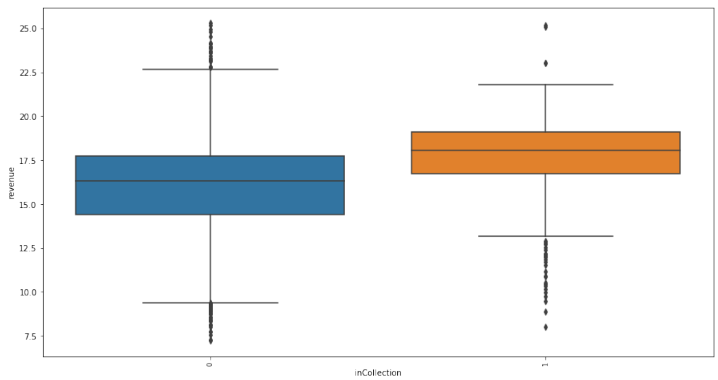

Figure 15: Revenue vs. inCollection

Next we analysed and processed the genre feature. Since movies have 0 to multiple genres associated and we have to feed our network with numerical data, we decided to replace this attribute with genre_len, that represents the number of different genres in each movie. The results are shown in figures 16 and 17. It seems the chances of bigger revenue are higher for movies with more genres.

```
***[genres_len]***
describe :
count    7398.000000
mean        2.494052
std         1.089433
min         1.000000
25%         2.000000
50%         2.000000
75%         3.000000
max         5.000000
Name: genres_len, dtype: float64
value_counts :
2    2384
3    2215
1    1497
4     969
5     333
Name: genres_len, dtype: int64
```

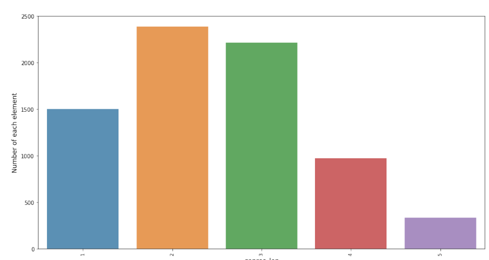

Figure 16: genre_len analysis results

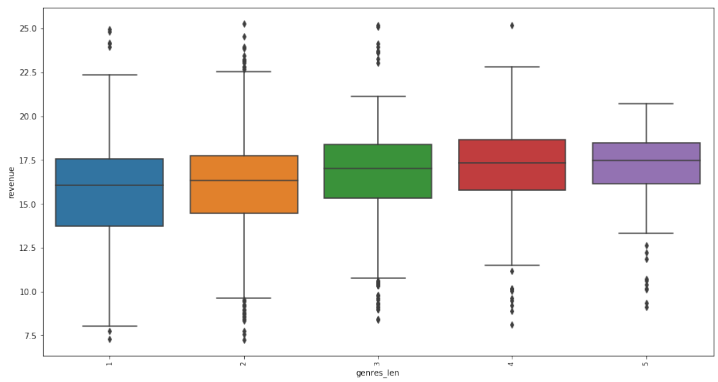

Figure 17: Revenue vs. genre_len

Another feature being processed and analysed is runtime, which represent the length of each movie in minutes. The results are shown in figures 18 and 19. There seems to be some movies with 0 value for runtime and we decided to replace it by the mean value and analysed the results after the change. Also, since we're interested in the differences between different movies with different runtimes and not the absolute value, the values were converted to a logarithmic scale. It's shown in figures 20 and 21.

```
***[runtime]***
describe :
count    7392.000000
mean      107.717262
std        21.480040
min         0.000000
25%        94.000000
50%       104.000000
75%       118.000000
max       338.000000
Name: runtime, dtype: float64
skew :
1.1189042635711173
*****[ corr : runtime ]*****
runtime              1.000000
budget               0.216251
revenue              0.190119
popularity           0.112664
genres_len           0.068846
index                0.006857
id                   0.000077
release_date_year   -0.035968
inCollection        -0.044143
Name: runtime, dtype: float64
*****[runtime]*****
```

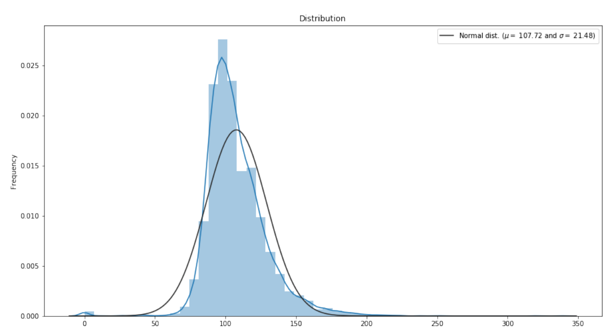

Figure 18: runtime analysis results

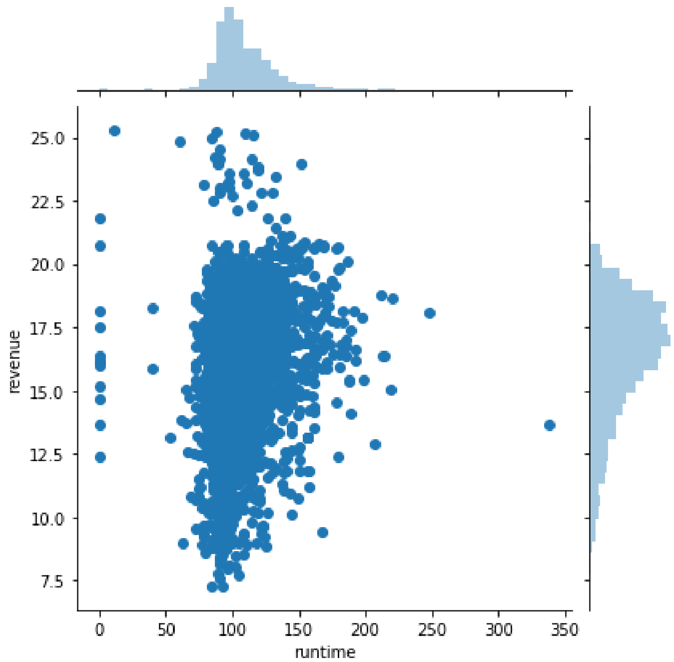

Figure 19: Revenue vs. runtime

```
***[runtime]***
describe :
count     7398.000000
mean         4.665702
std          0.180493
min          2.397895
25%          4.543295
50%          4.644391
75%          4.770685
max          5.823046
Name: runtime, dtype: float64
skew :
0.15575387140188032
*****[ corr : runtime ]*****
runtime              1.000000
budget               0.223337
revenue              0.203575
popularity           0.112406
genres_len           0.063690
index                0.008499
id                  -0.001732
release_date_year   -0.011247
inCollection        -0.048854
Name: runtime, dtype: float64
*****[runtime]*****
```

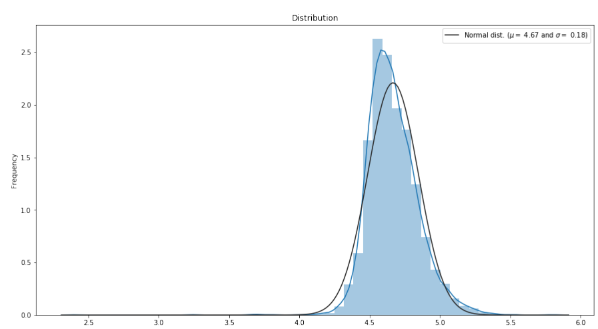

Figure 20: Runtime processed analysis results

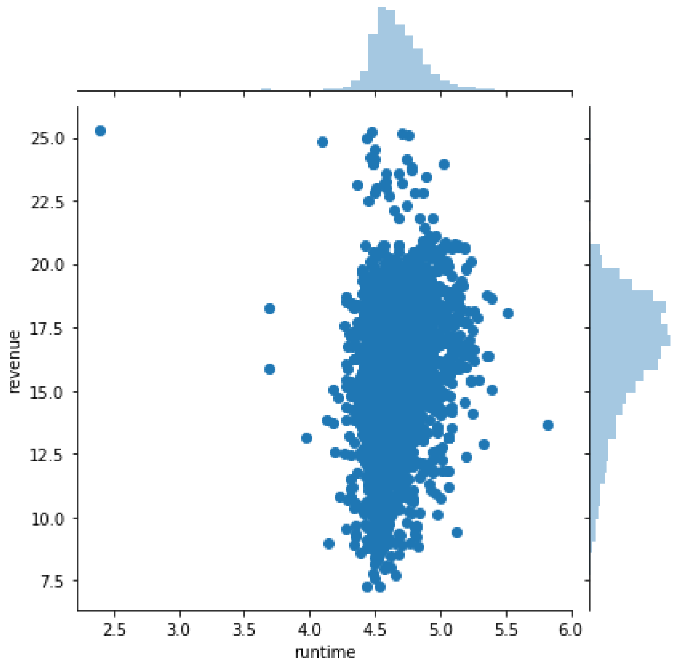

Figure 21: Revenue vs. runtime processed

Original_language gave place to a new binary feature is_eng that has the value 1 if the original language is English and 0 otherwise. The results are shown in figures 22 and 23. It's obvious that most of the movies are spoken mainly in English and there's some correlation between this feature and revenue.

```
***[is_eng]***
describe :
count    7398.000000
mean        0.858475
std         0.348586
min         0.000000
25%         1.000000
50%         1.000000
75%         1.000000
max         1.000000
Name: is_eng, dtype: float64
value_counts :
1    6351
0    1047
Name: is_eng, dtype: int64
```

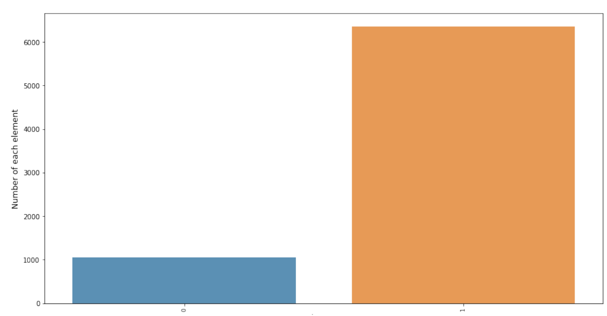

Figure 22: isEng processed analysis results

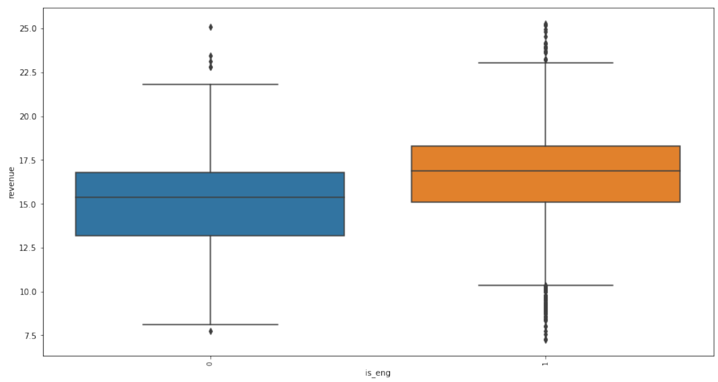

Figure 23: Revenue vs. isEng processed

In order to extract useful information from crew feature for our model, we created a new feature crew_num that represents the size of the crew for each movie and is represented in a logarithmic scale. The results are shown in figures 24 and 28. The same was done with cast information and the results are shown in figures 26 and 27.

```
***[crew_num]***
describe :
count    7398.000000
mean        2.649440
std         1.089911
min         0.000000
25%         2.079442
50%         2.639057
75%         3.332205
max         5.267858
Name: crew_num, dtype: float64
skew :
-0.18757957796201202
*****[ corr : crew_num ]*****
crew_num             1.000000
budget               0.406390
revenue              0.375356
popularity           0.292054
spoken_en            0.193591
is_eng               0.193591
runtime              0.176990
genres_len           0.159655
inCollection         0.110657
release_date_year    0.088226
index                0.009091
id                   0.002477
Name: crew_num, dtype: float64
*****[crew_num]*****
```

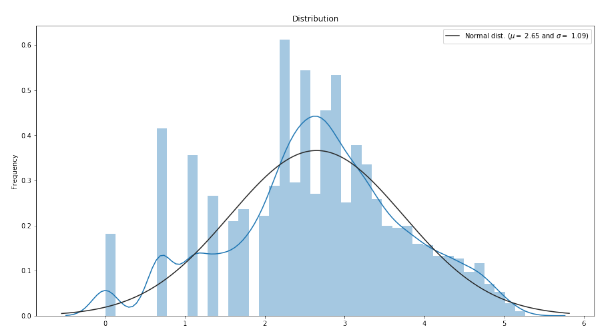

Figure 24: crew_num processed analysis results

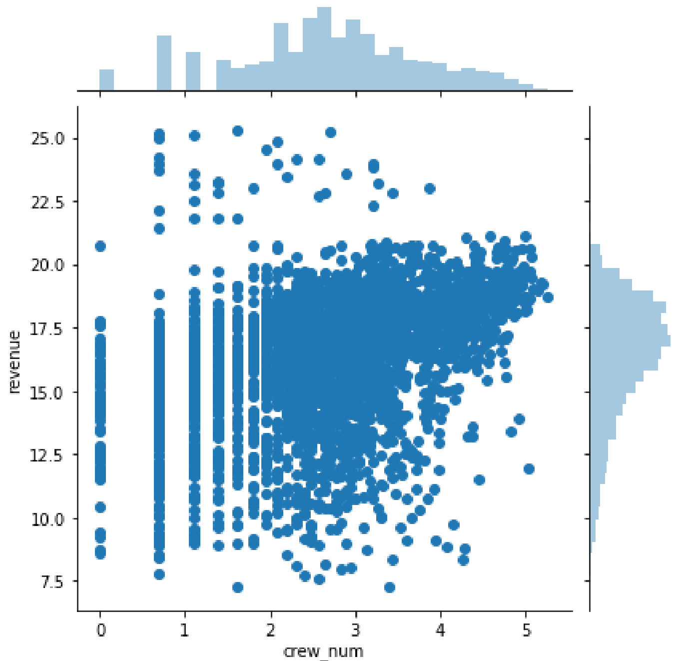

Figure 25: Revenue vs. crew_num processed

```
***[cast_num]***
describe :
count    7398.000000
mean        2.808945
std         0.690379
min         0.000000
25%         2.397895
50%         2.772589
75%         3.178054
max         5.105945
Name: cast_num, dtype: float64
skew :
-0.08338744731196232
*****[ corr : cast_num ]*****
cast_num             1.000000
crew_num             0.479969
revenue              0.342743
budget               0.339572
popularity           0.267683
runtime              0.253921
spoken_en            0.176705
is_eng               0.176705
inCollection         0.124424
genres_len           0.109470
release_date_year    0.014568
id                   0.013167
index                0.007910
Name: cast_num, dtype: float64
*****[cast_num]*****
```

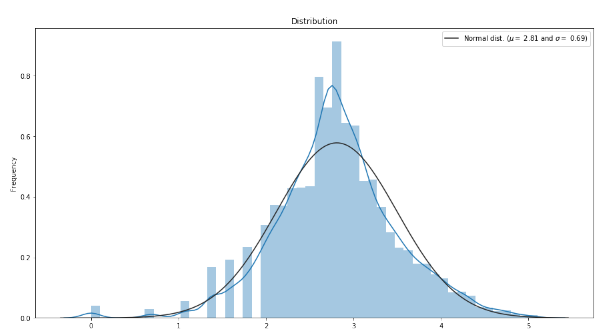

Figure 26: cast_num processed analysis results

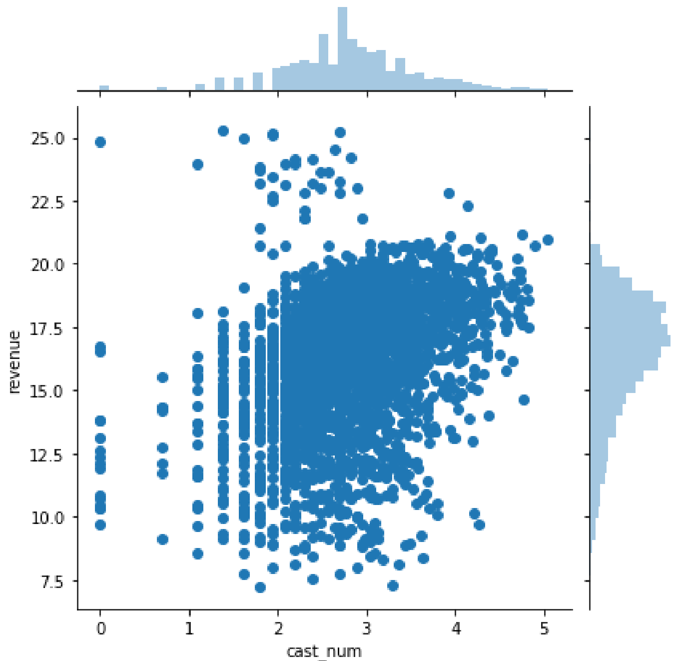

Figure 27: Revenue vs. cast_num processed

Finally, we converted features like production_countries and homepage into binary features prod_country_is_eng (1 if USA, 0 otherwise) and hasHomepage (1 if movie has homepage, 0 otherwise).

3.2 Model

In this study, we propose a predictive model composed by a linear stack of layers. We used 6 consecutive layers with ReLu activation function [15], with a different number of neurons, the input layer has 340 neurons, the second layer has 120, the third 80, the forth has 40 and the two last have 20. The last layer, the output layer, has 1 neuron with a linear activation function. We used the Adam optimization algorithm which is an extension to stochastic gradient descent [2], using a learning rate of 1e-5. A learning model tries to minimize the difference between the real value and the prediction during training. Since we work on a regression task, we used mean squared error Root Mean Squared Error (RMSE) as the loss function. The model was trained for 100 epochs using a batch size of 8 in order to update the weights of the network.

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1_input (InputLayer)   (None, 12)                0
_________________________________________________________________
dense_1 (Dense)              (None, 120)               1560
_________________________________________________________________
dense_2 (Dense)              (None, 80)                9680
_________________________________________________________________
dense_3 (Dense)              (None, 40)                3240
_________________________________________________________________
dense_4 (Dense)              (None, 20)                820
_________________________________________________________________
dense_5 (Dense)              (None, 20)                420
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 21
=================================================================
Total params: 15,741
Trainable params: 15,741
Non-trainable params: 0
```

Figure 28: Model architecture

4 Results

The evaluation metric used in this Kaggle competition was the Mean Squared Logarithmic Error (LMSE):

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N} (log(y_i + 1) - log(\hat{y_i} + 1))^2 \tag{1}$$

The loss is the mean over the seen data of the squared differences between the log-transformed true and predicted values. This metric is a value one when the target is a continuous vale and have a significant different order of magnitude. With this large errors will not be more penalizing for the model than the small ones.

After the submission, with our model, we got a score of 2.63. This score place us in the 1001th position of the Kaggle competition leader board.

5 Discussion

5.1 Results Discussion

One of the most relevant goals of this work was to only use pre-released movie data. In this approach, we strictly followed the competition rules, however after analyzing some of the top-leaders from this Kaggle competition we noticed that some of the "forbidden" features were used, like reviews. Besides that, they also used not only numerical data but also movie's poster data, using image processing and other kind of features.

5.2 Model Explainability

Machine and Deep Learning are at the core of many recent advances in technology and science. However the human understanding is not keeping up with these techniques. By this, it is vital to besides accuracy metrics, to determine trust in the model's predictions. Inspecting individual predictions and their explanations one of the possible solutions, for that we used LIME, [9] Lime stands for Local Interpretable Model-Agnostic Explanations, a novel explanation technique that explains the predictions of any classifier in an interpretable and faithful manner.

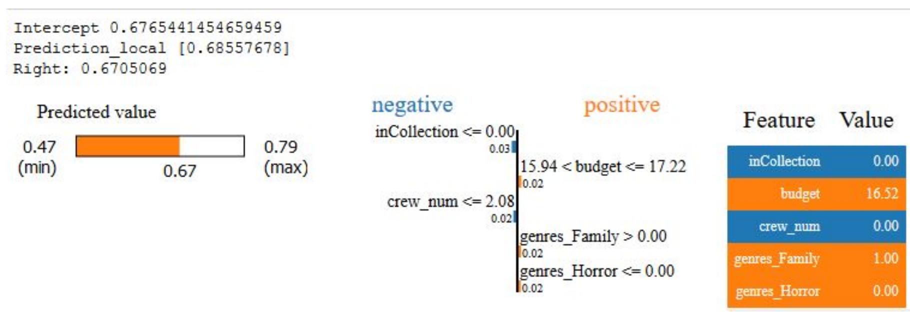

Figure 29: Explaining individual predictions (sample 23)

The LIME explainer for regression show us that the 23th test value's prediction is 0.67 (4.68 Millions USD) with the most important features for this prediction are budget and genres_Family while inCollection and crew_num providing negative valuation. We can see another examples of explanations of predictions in the following images.

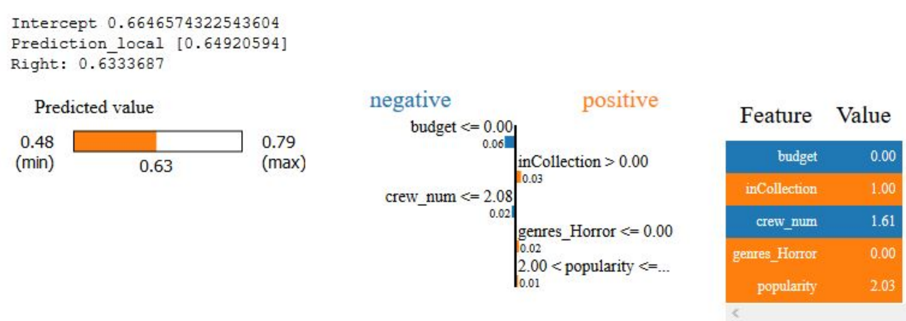

Figure 30: Explaining individual predictions (sample 112)

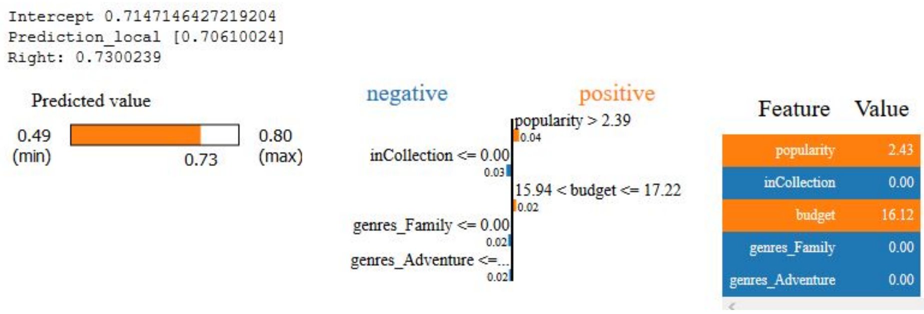

Figure 31: Explaining individual predictions (sample 106)

Many of the state of the art machine learning models are functionally black boxes, as it is nearly impossible to get a feeling for its inner workings. With this kind of tools, in the future there will be more white boxes and less black ones.

References

[1] The movie database. https://www.themoviedb.org/. Accessed: 2019-11-26.

[2] Jimmy Ba Diederik P. Kingma. Adam: A method for stochastic optimization, 2014.

[3] Carole Sudreb Lucas Fidona Dzhoshkun Shakira Guotai Wang Zach Eaton-Rosen Robert Grayc Tom Doel Yipeng Hu Tom Whyntie Parashkev Nachev Marc Modat Dean C. Barratt Sébastien Ourselin M. Jorge Cardoso Eli Gibson, Wenqi Li and Tom Vercauteren. Niftynet: a deep-learning platform for medical imaging, 2018.

[4] Mark Hughes. 2018 sets new box office record with enormous $41 billion worldwide. *Forbes*, Dec 2018.

[5] Barry R. Litman Linda S. Kohl. Predicting financial success of motion pictures: The '80s experience, 2009.

[6] Kyung Jae Lee and Woojin Chang. Bayesian belief network for box-office performance: A case study on korean movies, 2009.

[7] Jianhua Luo Li Zhang and SuyingYang. Forecasting box office revenue of movies with bp neural network, 2009.

[8] Pengda Liu. Machine learning on predicting gross box office, 2016.

[9] Carlos Guestrin Macro Tulio Ribeiro, Sammer Singh. 'why should i trust you?'explaining the predictions of any classifier, 2016.

[10] Morten Goodwin Mehdi Ben Lazreg and Ole-Christoffer Granmo. Deep learning for social media analysis in crises situations, 2016.

[11] Jehoshua Eliashberg Mohanbir S. Sawhney. A parsimonious model for forecasting gross box-office revenues of motion pictures, 1996.

[12] Jay Adams Paul Covington and Emre Sargin. Deep neural networks for youtube recommendations, 2016.

[13] Denise De Gaetano Rita Georgina Guimarães, Renata L. Rosa and Demóstenes Z. Rodríguez. Age groups classification in social network using deep learning, 2017.

[14] Ramesh Sharda and Dursun Delen. Predicting box-office success of motion pictures with neural networks, 2006.

[15] Geoffrey E. Hinton Vinod Nair. Rectified linear units improve restricted boltzmann machines, 2010.

[16] Lei Zhang Yao Zhou and Zhang Yi. Predicting movie box-office revenues using deep neural networks, 2017.

[17] Wenbin Zhang and Steven Skiena. Improving movie gross prediction through news analysis, 2009.