Predicting the movie revenues with Machine Learning

Meriç DEMİRÖRS
Artificial Intelligence Engineering Department
TOBB ETU
Ankara, Turkey
mdemirors@etu.edu.tr

Emre BELİKIRIK
Artificial Intelligence Engineering Department
TOBB ETU
Ankara, Turkey
ebelikırık@etu.edu.tr

***Abstract*—This research paper is written by Meriç Demirörs and Emre Belikırık(both from TOBB ETU) to predict revenues of movies between 1990 and 2022. Some classic and deep machine learning models are used and best models selected. The observed problem is sometimes a movies revenue is incalculable, sometimes it is not calculated or sometimes it is not published openly. Our purpose is to create an AI model for movies in this case, which can estimate the expected revenue close enough to unknown reality by learning other movies data. There are papers in internet whichs topic is same but has some differences in detail. Our papers biggest significant difference is that we used vote numbers of each movie in out dataset. Data sources and usage are explained clearly in the paper.**

***Index Terms*—SVM, SVR, NN, RFR, MSE, RMSE, MAE**

I. Introduction

Predicting movie revenues can be a valuable task for a variety of reasons. For one, understanding the financial performance of a film can help studios and production companies make informed decisions about which projects to pursue. By training an AI model to predict movie revenues, studios can potentially identify which films are likely to be more financially successful, allowing them to allocate resources more efficiently and potentially increase profits.

Another reason to train an AI model to predict movie revenues is to help with marketing efforts. By understanding which films are likely to be more popular, studios can target their marketing efforts towards the right audience, potentially increasing ticket sales and revenue. Additionally, knowing which films are likely to be more successful can help studios decide how to allocate their marketing budgets, potentially leading to a better return on investment.

There are also potential applications for predicting movie revenues outside of the film industry. For example, investors and financial analysts may be interested in understanding the financial performance of a film as a way to make investment decisions. Similarly, advertisers may want to know which films are likely to be more popular in order to decide which films to advertise with.

In summary, training an AI model to predict movie revenues can be valuable for a variety of reasons. It can help studios and production companies make informed decisions about which projects to pursue, aid in marketing efforts, and potentially have applications outside of the film industry.

But our main problem is trying to realistically calculate the revenue of a movie by looking at the watch number prediction charts of a newly released movie and/or some other data. To solve this, we scraped our data from various websites(IMDb, Wikipedia, The Numbers, Boxofficemojo) and on this data we trained some Machine Learning models which will be listed later on the paper.

To be more precise, the problem is sometimes a movies revenue is incalculable, sometimes it is not calculated or sometimes it is not published openly. Our goal is to create some solution, an AI model, for movies in this case which can estimate the expected revenue close enough to unknown reality by learning other movies data.

Our dataset includes title and release year of the movie, its IMDb rating and its vote count on IMDb and ofcourse our target WorldwideBox Office revenues.

II. Related work

As searching through to internet and looking up papers in ”Movie Revenue Prediction with Machine Learning” topic we can see that there is a notable amount of papers and studies which may and will inspire further works on topic. But, since we do not have time to take a look at all of this available information right now we picked four papers(which are referenced bottom of the document) and we read them.

To summarize the common features for all four papers they all have some special datasets for themselves and unique methods to use, they all used IMDb data(some of them uses more than that[such as OMDB which is the second most popular data site in this four paper]). These papers demonstrate the various approaches that can be taken to predict movie revenues using machine learning, and they provide inspiration for further research in this area. At some points each one has a difference side from others and these differences are listed below(some attributes are same but since they are not used in all four papers they are included):

• Uses ABD and China spesific data only[1]
• Uses gross domestic product (GDP) and the number of movie screens (NMS) as feature[1]
• Trains the model in the LSTM way[1]
• Looked up Rotten Tomatoes, Box Office Mojo and Metacritic for data[2]
• Calculates thing called "Star Power" for Actor/Actress/Director which indicates the "persons impact on movies popularity/revenue"[2]
• Calles attention to revenue spike at spesific months such as festival times[2]
• Uses "budget of the movie" as a feature, and emphasized that is important[2]
• Says that some movies have a good revenue despite their quality just because they are a sequal to another good movie[2]
• Uses A graph representation of actor-actor, actor-movie and movie-movie relations[3]
• Takes data from ShowBiz Data(a private movie database website).[3]
• Uses 771 English movies released in the USA in 2010-2015[4]
• Uses MPAA rating, and indicates its importance[4]
• Movies production stage is not very useful to predict revenue unlike the first week of screening[4]
• Uses film critic and the academy award nominations data[4]
• Uses number of screening in the first weekend[4]
• Uses budget, star power, festival times and sequel data as well as second paper[4]
• Says that sequel movies has bigger revenue[4]
• removes release date feature since they think its unimportant[4]

Mentioned Machine Learning models are NN, SVM, NLP, Random Forest, Regression models, ARIMA and some other economic regression/prediction models which are out of our scope for now. Since all four papers data are seperate from each other, as expected best performed models are seperate also(SVM for [1], NN for [2], two layered NN for [3], Multiple Linear Regression Model(2 seperate model for 2 different prediction type:1.Pre-production: predicts revenue during the movies production state and 2.Post-Release: predicts revenue after movies first week screening) for [4])

Upon reviewing the papers on the topic of "Movie Revenue Prediction with Machine Learning," we noticed that our data scraping source is more limited in scope compared to some of the other papers. However, our data does contain more features, such as ratings, release year, total votes, and number of votes at each point, than any of the other data sources used in the papers. Despite having a narrower data scraping source, the additional features in our data may provide us with a more comprehensive understanding of movie revenues and enable us to build more accurate predictive models.

III. Method

Models we used are listed below:
-Linear regression
-Support vector regression
-Random forest regression
-Neural network

First we scrape our data from IMDb, Wikipedia, The Numbers and Boxofficemojo. We downloaded ready IMDb datasets to select movies by year and construct their urls. After that ve decide to use movies since 1990 because older the movie less useful to our models goal. After sorting we are left with 426.000 movie, and scraping all movie vote numbers take around 1-2 weeks with two computers. After that around 8.000 movie passed our 10.000 vote threshold(to eliminate noise and outliers/non trainable movies beforehand) so we joined that dataset with movie revenue datasets which we acquire from The Numbers and Boxofficemojo on movie name and year. After that there were movies without revenue data or with unreliable revenue data. So we seperated this movies and scrape their revenue information from Wikipedia and The Numbers. There were around 3.000 movies and it take 1 day with 1 computer to take data. And we manualy switch nearly 100 indian movies revenue to USD. At the end, we ended up with 7.526 movies data. Final dataset features were: title id(IMDb's unique id for each movie),Year(release year of movie), Rating(IMDb rating of movie), Votes(Total number of votes for movie), Title(title of movie), rating link(movies statistic pages url for movie), 1, 2, 3, 4, 5, 6, 7, 8, 9, 10(number of votes at each vote point for movie), Title-Year(unique key to join movie dataset with WorldwideBox Office dataset), WorldwideBox Office(Worldwide revenue of movie[in dollar]), imdb_url(movies main page at IMBd).

Then we clear the data from unusable/ noninformative features and normalize the numeric values(with log2 and other normalizations), then our models are trained on this data. Then succesful models are used embedded.

To prepare our features, we pass them more than one normalizations to set them in a trainable range. We replace each year with that years mean value of revenues which indicates in a way which gives a understandable reasoning for each years movies revenue(like a number which represent

how big is that years movies revenue). After that we divide them with 1e8 to scale minimum and maximum mean revenue of years to a close range to 0-1. For ratings we just divide them with 10 to set them in 0-1 range. Then we take the log2 of other columns and pass with MinMaxScaler since their histograms are have a expolential function sort of manner. After that we cut the outliers on WorldwideBox Office features. After we realized, to models there is no difference between predicting 30 as 31 and 10 as 11 we create a different dataset, we pass WorldwideBox Office to log2 function but while training models we pass a sample weight which represents how important that revenue to learn, for example lower number means lower revenues and we can tolerate to be one or two number off from real prediction but for higher numbers such as 30, we can't tolerate models to predict 32, or even 31 because that is 1 billion difference in numbers.. And for another dataset we didn't use log2 for WorldwideBox Office column, instead of that we divide each revenue with the smallest one after cutting outliers.

We then trained our models with logged dataset, sample weighted dataset and dataset without log. Predictions were better than models which are trained without sample weight as expected. But we were still loosing the importance of big revenues. Yes models were more likely to learn bigger revenued movies and less focused to lower ones but as we mentioned before predicting 30 as 31 is a bigger error than predicting 10 as 11, we can't train models saying that ones importance is 10 and others is 30, there is thousand times more deviation in numbers. Using log2 function was causing us to lose importance of bigger revenues and applying a basic sample weight wasn't patching up. So instead of giving a bigger weights we train models with dataset without log2 function. After that models did learn way better because now our data is not capsulated with log2 function and it preserves its numbers linearity. And for lastly, for experiment we tried training models with non-touched revenue data, whichs range is 100.000 to 1.000.000.000. And models did not perform well because of the range of numbers, because predictions were of with thousands and backpropagations were causing derivations to converge 0 or 1 which prevents learning.

Best models are selected with parameter optimizations of GridSearchCV() from sklearn library and search()/RandomSearch()/BayesianSearch() from KerasTuner library. Our classic models are trained around 2 days(biggest are: Random Forest Regressor: 32 hour grid search, Suppor Vector Regression: 8 hour grid search) and NN models take 1-2 week to search and find the right parameter space to search. NN models parameter optimization take around a week to finish search(), RandomSearch() and quarter of the BayesianSearch() with much shallow layers. All trainings done by one computer.

IV. Experimental Results

Our goal in this project is to develop numeric estimates of movie revenues, and we will measure the accuracy of our predictions by comparing them to the actual revenues. To evaluate the performance of our models, we will use error calculation functions such as MSE and RMSE. Below, we present the results of our best models, including an ensemble model, along with a summary of their performance. We will discuss the results in terms of how closely our revenue estimates match the real revenues, and we will use the error calculation functions to provide a more detailed analysis of the models' accuracy.

Below models are the one we choose to use at our deployment since classical models did not learn as good as them they are all Neural Network models. First four one were selected after around hundred Neural Network tries to find sweet spot to do hyperparameter search. Next 3 were best performed models in hyperparameter search with search()/RandomSearch()/BayesianSearch() from KerasTuner library. Last one is the embedded models predictions evaluation scores. Our search()/RandomSearch()/BayesianSearch() spaces are like this:

A. search() and RandomSearch()

At search() and RandomSearch() parameters since it doesn't take too long(RandomSearch is limited by a max trial number of 1000 trials in our case) we look for a bigger space and both of them did run until the end.

Default search space size: 12
hidden layer1 (Int) 'default': None, 'conditions': [], 'min': 32, 'max': 64, 'step': 32, 'sampling': None
hidden layer2 (Int) 'default': None, 'conditions': [], 'min': 32, 'max': 128, 'step': 32, 'sampling': None
hidden layer3 (Int) 'default': None, 'conditions': [], 'min': 64, 'max': 128, 'step': 32, 'sampling': None
dropout layer1 (Float) 'default': 0.0, 'conditions': [], 'min': 0.0, 'max': 0.3, 'step': 0.1, 'sampling': None
hidden layer4 (Int) 'default': None, 'conditions': [], 'min': 64, 'max': 256, 'step': 32, 'sampling': None
hidden layer5 (Int) 'default': None, 'conditions': [], 'min': 128, 'max': 265, 'step': 32, 'sampling': None
hidden layer6 (Int) 'default': None, 'conditions': [], 'min': 128, 'max': 256, 'step': 32, 'sampling': None
dropout layer2 (Float) 'default': 0.0, 'conditions': [], 'min': 0.0, 'max': 0.3, 'step': 0.1, 'sampling': None
hidden layer7 (Int) 'default': None, 'conditions': [], 'min': 64, 'max': 128, 'step': 32, 'sampling': None
hidden layer8 (Int) 'default': None, 'conditions': [], 'min': 32, 'max': 128, 'step': 32, 'sampling': None
hidden layer9 (Int) 'default': None, 'conditions': [], 'min': 16, 'max': 64, 'step': 16, 'sampling': None
learning_rate (Choice) 'default': 0.001, 'conditions': [], 'values': [0.001, 0.01], 'ordered': True

B. BayesianSearch()

But at BayesianSearch() since all parameter combination are trained fully we have to cut it at some point since there is 4032 combinations and with learning rate included 8064. Each model were running 100 epoch which will take around 224 hours(nine to ten days) to complete, so we end it at 700th trial.

Default search space size: 7
hidden layer1 (Int) 'default': None, 'conditions': [], 'min': 32, 'max': 64, 'step': 32, 'sampling': None
hidden layer2 (Int) 'default': None, 'conditions': [], 'min': 32, 'max': 128, 'step': 32, 'sampling': None
dropout layer1 (Float) 'default': 0.0, 'conditions': [], 'min': 0.0, 'max': 0.3, 'step': 0.1, 'sampling': None
hidden layer4 (Int) 'default': None, 'conditions': [], 'min': 64, 'max': 256, 'step': 32, 'sampling': None
hidden layer5 (Int) 'default': None, 'conditions': [], 'min': 64, 'max': 128, 'step': 32, 'sampling': None
hidden layer6 (Int) 'default': None, 'conditions': [], 'min': 16, 'max': 64, 'step': 16, 'sampling': None
learning_rate (Choice) 'default': 0.001, 'conditions': [], 'values': [0.001, 0.01], 'ordered': True

C. Best models after Neutal Network tries and Hyperparameter searchs

As mentioned before NN14, NN16, NN23 and NN24 were Neural Network tries to find relevant hyperparameter search space. And kerasNNtuner models are best performing models of hyperparameter searchs(001 means learning rate and 24 means batch size. This parameters were different in first 4 Neural Networks).

model:NN14 without log

Layer (type) Output Shape Parameter number
=================================
dense (Dense) (None, 32) 768
dropout (Dropout) (None, 32) 0
dense 1 (Dense) (None, 64) 2112
dropout 1 (Dropout) (None, 64) 0
dense 2 (Dense) (None, 64) 4160
dense 3 (Dense) (None, 32) 2080
dropout 2 (Dropout) (None, 32) 0
dense 4 (Dense) (None, 13) 429
dense 5 (Dense) (None, 1) 14
=================================
Total params: 9,563
Trainable params: 9,563
Non-trainable params: 0
MSE: 858479.6420716271
RMSE: 926.5417648825265
MAE: 470.5805976898069
R2: 0.609300343896483
AdjustedR2: 0.5969860280880627

model:NN16 without log

Layer (type) Output Shape Parameter number
=================================
dense 6 (Dense) (None, 32) 768
dropout 3 (Dropout) (None, 32) 0
dense 7 (Dense) (None, 64) 2112
dropout 4 (Dropout) (None, 64) 0
dense 8 (Dense) (None, 64) 4160
dense 9 (Dense) (None, 32) 2080
dropout 5 (Dropout) (None, 32) 0
dense 10 (Dense) (None, 13) 429
dense 11 (Dense) (None, 1) 14
=================================
Total params: 9,563
Trainable params: 9,563
Non-trainable params: 0
MSE: 851082.588205637
RMSE: 922.5413747933677
MAE: 481.30658805166445
R2: 0.612666791113155
AdjustedR2: 0.6004585810909335

model:NN23 without log

Layer (type) Output Shape Parameter number
=================================
dense 12 (Dense) (None, 32) 768
dropout 6 (Dropout) (None, 32) 0
dense 13 (Dense) (None, 64) 2112
dense 14 (Dense) (None, 32) 2080
dropout 7 (Dropout) (None, 32) 0
dense 15 (Dense) (None, 13) 429
dense 16 (Dense) (None, 1) 14
=================================
Total params: 5,403
Trainable params: 5,403
Non-trainable params: 0
MSE: 807743.6071700169
RMSE: 898.7455742144252
MAE: 451.8298872868648
R2: 0.6323906426253856
AdjustedR2: 0.6208041012754693

model:NN24 without log

Layer (type) Output Shape Parameter number
==========
dense 17 (Dense) (None, 32) 768
dense 18 (Dense) (None, 64) 2112
dropout 8 (Dropout) (None, 64) 0
dense 19 (Dense) (None, 64) 4160
dense 20 (Dense) (None, 64) 4160
dropout 9 (Dropout) (None, 64) 0
dense 21 (Dense) (None, 32) 2080
dense 22 (Dense) (None, 13) 429
dense 23 (Dense) (None, 1) 14
==========
Total params: 13,723
Trainable params: 13,723
Non-trainable params: 0
MSE: 841217.1934451301
RMSE: 917.178932076577
MAE: 445.3226490337043
R2: 0.6171565962889123
AdjustedR2: 0.6050898987507405

model:kerasNNtuner without log-001-24.h5

Layer (type) Output Shape Parameter number
==========
dense 24 (Dense) (None, 32) 768
dense 25 (Dense) (None, 96) 3168
dropout 10 (Dropout) (None, 96) 0
dense 26 (Dense) (None, 128) 12416
dense 27 (Dense) (None, 192) 24768
dropout 11 (Dropout) (None, 192) 0
dense 28 (Dense) (None, 128) 24704
dense 29 (Dense) (None, 64) 8256
dense 30 (Dense) (None, 32) 2080
dense 31 (Dense) (None, 1) 33
==========
Total params: 76,193
Trainable params: 76,193
Non-trainable params: 0
MSE: 810993.9251145053
RMSE: 900.5520113322191
MAE: 452.8431457304746
R2: 0.6309114018363153
AdjustedR2: 0.6192782368512135

model:kerasNNtuner2 without log-001-24.h5

Layer (type) Output Shape Parameter number
==========
dense 32 (Dense) (None, 32) 768
dense 33 (Dense) (None, 96) 3168
dense 34 (Dense) (None, 96) 9312
dropout 12 (Dropout) (None, 96) 0
dense 35 (Dense) (None, 128) 12416
dense 36 (Dense) (None, 128) 16512
dense 37 (Dense) (None, 160) 20640
dropout 13 (Dropout) (None, 160) 0

dense 38 (Dense) (None, 64) 10304
dense 39 (Dense) (None, 96) 6240
dense 40 (Dense) (None, 32) 3104
dense 41 (Dense) (None, 1) 33
==========
Total params: 82,497
Trainable params: 82,497
Non-trainable params: 0
MSE: 793314.2178322267
RMSE: 890.6818836331109
MAE: 440.09037967056236
R2: 0.6389575513507388
AdjustedR2: 0.6275779899319942

model:kerasNNtuner3 without log-001-24.h5

Layer (type) Output Shape Parameter number
==========
dense 42 (Dense) (None, 64) 1536
dense 43 (Dense) (None, 32) 2080
dropout 14 (Dropout) (None, 32) 0
dense 44 (Dense) (None, 224) 7392
dense 45 (Dense) (None, 64) 14400
dense 46 (Dense) (None, 48) 3120
dense 47 (Dense) (None, 1) 49
==========
Total params: 28,577
Trainable params: 28,577
Non-trainable params: 0
MSE: 839638.6222608767
RMSE: 916.3179700632727
MAE: 461.2935493576889
R2: 0.6178750142788071
AdjustedR2: 0.6058309602875946

model:embedded 7 NN validation

MSE: 807660.4701058877
RMSE: 898.699321300449
MAE: 451.9713867762668
R2: 0.6324284788427783
AdjustedR2: 0.620843130038396

model:embedded 7 NN test

MSE: 581046.4163246541
RMSE: 762.264006971767
MAE: 426.70861703874084
R2: 0.6513887430810024
AdjustedR2: 0.6404009957282546

D. Classical Models

And our classical model results:

model:LR without log

MSE: 1246815.5069980384
RMSE: 1116.6089319891894
MAE: 673.035218261938
R2: 0.43256617171124245
AdjustedR2: 0.42294864919787367

model:SVR without log

MSE: 880231.8669605369
RMSE: 938.206729330235
MAE: 426.4701384347858
R2: 0.5994007651911859
AdjustedR2: 0.5926109476520535

model:RF with log

MSE: 3.493796331056587
RMSE: 1.869169957777138
MAE: 1.436990225296124
R2: 0.614211322064746
AdjustedR2: 0.6076725309133011

It is hard to estimate why classic models are worse than Neural Networks just by looking at these numbers but by comparing prediction graphs and calculating smallest and biggest revenue estimation gaps show that classical models are not suitable for this task.

MSE and RMSE both penalize large errors more heavily than smaller ones. It means you are more concerned with getting the larger errors right. On the other hand, MAE treats all errors equally, it will not be influenced as heavily by a few large errors. R-squared is the proportion of the variance in the dependent variable that is explained by the model, while adjusted R-squared takes into account the number of independent variables in the model and adjusts the R-squared value accordingly. First 2 evaluation metric is used to evaluate the models without log function and last 2 metric are used to evaluate the models with log.

V. Deployment

First we deployed it on heroku with the streamlit library, then when heroku started asking for money for use, we tried many different deployment methods such as fly, railway, render, huggingfaces, skops, spaces, gradio. However, in the end, we decided that the most convenient and easy way for us was to deploy over streamlit, and we deployed it only through streamlit without using any other platform.

VI. Discussion

Even if our data seems bigger than other ones our predictions are way off then the acceptable interval because our revenue range is between 100.000 to 1.000.000.000 but we only have thousands of train data. Yes, we can generalize the data, and yes we can find lines/curves to fit data but being precise is the hard part. Our study show that some basic information such as votes and rating is not enough for precise model since most of the movies are effected by their related peoples fame, the timeline they release and advertisements. We can merge our dataset with other research's dataset but since all of them have a different movie archives and models for themselves it would need a lot work to done. Another way can be generating new data with Bootstrap method or to get movies from bigger year range or lower the vote threshold(these can be result in poor predictions also because models will learn non related movie since our year range goes to far)

In the upcoming researchs using both our vote information and other researchs spesific features may and will certainly lead to better results and more precise predictions. After creating a decent dataset that includes all of this information, a model that predicts well is possible.

VII. Conclusion

Our primary challenge is attempting to accurately estimate the income of a film by examining the watch number prediction charts and/or other data. In order to address this, we collected data from a variety of websites, trained some Machine Learning models on this data. To be more exact, the issue is that a movie's revenue is occasionally impossible to quantify, occasionally not estimated, or occasionally not disclosed in the public domain. Our objective is to develop a model that can predict predicted income for movies with sufficient accuracies based on data. We trained some classical models and variety of NNs with all different parameters and data types. Classical models did not perform well, and at their finest they predict some movies renevues negative. Neural Networks were better than classical models but they

still do badly predictions. At the end, we saw that even the most complex model of ours is predicting poorly. To our conclusion, our data is not big enough and does have less features than needed. As we said in the discussion, “In the future there may be a model to solve this problem with a larger and more comprehensive dataset.”.

VIII. DATA WEB PAGES

- https://www.imdb.com/interfaces/
- https://datasets.imdbws.com/
- https://www.the-numbers.com/box-office-records/worldwide/all-movies/cumulative/all-time
- https://www.boxofficemojo.com/year/world/

REFERENCES AND READINGS

- [1] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiIiNHi6ND6AhVsVfEDHcB6BREQFnoECBMQAQ&url=https%3A%2F%2Fwww.mdpi.com%2F1099-4300%2F24%2F5%2F711%2Fpdf&usg=AOvVaw0dyDg4d1gy5S2UwFZ6RWGj

- [2] https://www.researchgate.net/profile/Dipankar-Chaki/publication/322138608_A_Machine_Learning_Approach_to_Predict_Movie_Box-Office_Success/links/5a7e42934585154d57d4f318/A-Machine-Learning-Approach-to-Predict-Movie-Box-Office-Success.pdf?origin=publication_detail

- [3] http://snap.stanford.edu/class/cs224w-2015/projects_2015/Predicting_Box_Office_Revenue_for_Movies.pdf

- [4] https://arxiv.org/pdf/1804.03565.pdf

DEPLOYMENT

https://emre-bl-movie-revenue-prediction-deploy-po0x7e.streamlit.app/