What does matter in the total revenue of a movie?

Data Science 3: Big Data Management Systems and Tools

2020
Group 15

Ambagaspitiye Gedara, Chitranee
Jimenez, Sergio
Lee, Hye Lim
McCluskey, Tyler
Sun, Oscar
Velagala, Priyanka

What is this all about? Objectives

In the current situation the motion picture industry is in a deep crisis in a scenario unknown in its history. Movie theaters are closed in most of the countries and where (or when) they aren't, the public affluence is drastically reduced because of health protocols and fear of getting covid, which set up a scenario where most of the studios are struggling to get movies with cost effective results. In this scenario, every single movie is a very expensive bullet and studios cannot miss the shot of what to produce, when to release their movies and where to release with a hypercompetitive streaming market even without covid restrictions.
Our study is aimed to detect what factors we need to consider in order to predict the total revenue of a movie. It's not just to look for which movies get higher revenue, but the different relationships of movies, intrinsic factors (such as budget and runtime) and popularity factors (likeability and unanimity in the preferences) which could explain possible revenue scenarios for future movies in order to choose whether to produce them, or define which distribution circuit to choose.

Hypothesis or exploration? How do we better predict the revenue of a movie

Our work is exploratory because we are trying to understand what makes a movie generate more revenue. We are using most of the numerical variables in our dataset such as budget adjusted, revenue adjusted, popularity, runtime, vote count and vote average. Our idea is to play with different approaches to get a better prediction of movie revenues:

- Is it always the same factors with the same weight that share a similar relationship with the revenue? In this case we should be able to get a good prediction using a linear regression.
- Or are there some factors with different weights which need to be considered in order to get a better and more accurate distribution of the power of these variables in the model? In this case, we will get a better prediction using Random Forest. We have tried two different approaches:
    - Is there a normalized feature for the movies such as having a website a good predictor for the revenue of a movie
    - What other features beyond popularity, budget and public votes could get a better prediction for the movie.

Depending on the results we should be able to know what are the more important features to predict the revenue.This should be useful to define the motion pictures industry in the close future. If there are just intrinsic factors such as budget and runtime they should focus on a diversified portfolio equilibrating both components in order to decide where to release movies. If there are also popularity factors, studies should invest in marketing and communication and depending on the power of popularity and average votes, decide if it is better to have an expansive approach reaching a lot of people, or an aimed strategy trying to get high values in a specific segment of the public.

Our dataset: TMDb Dataset

In order to do our study we have used the dataset TMDb Movies Dataset, a set of more than 10000 movies in the IMDB with the following 21 columns:

- 9 string data
    - IMDB Id
    - Original Title: title of the movie
    - Cast: Main cast of the movie
    - Director: director of the movie
    - Tagline of the movie: tagline of the movie if it exists. If it doesn't it will be empty.
    - Keywords: keywords associated to the movie
    - Overview: short description of the movie
    - Genres: list of genres to catalogue the movie
    - Production: Producing companies
- 5 integer data
    - Budget: Budget of the movie at the release date
    - Revenue: revenue of the movie at the release date
    - Runtime: duration of the movie in minutes
    - Vote count: number of votes at the IMDb
    - Release year: year of release
- 4 decimal
    - Popularity of the film: Score calculated by IMDb
    - vote average: Average of the votes in IMDb
    - budget adjusted to the current date because the inflation has an impact in a set which has movies for all the ages.
    - revenue adjusted same as the budget
- 3 of other kind
    - id: id of the movie in the IMDb
    - homepage: homepage of the movie, if there is one. If not, it would be empty
    - release date: Release date of the movie

What have we done with data? Data preparation

There are some missing values, mostly in string variables such as tagline, production companies, total vote and very few missing cases for budget, revenue and their adjusted versions and a few more in runtime, but less than 10%. Also, we have dropped all the movies with 0 values in adjusted revenue or adjusted budget because they're out of our research's scope.

For comparison terms we have taken adjusted budget and revenue to compare the performance of movies from different years which ranges from 1959 to 2015

Analysis

For this study we are using four different Spark notebooks, one for the linear regression model, one for each random forest model and another for XGBoost Regression to find if a boarder model gets better predictions. All of them have required a specific treatment of featuring the dataset, creating pipelines and training to get the results.

1. Linear Regression

1. 1 Data preparation

To analyze the adjusted revenue using linear regression, categorical values are omitted for this analysis in the raw dataset.

	0	1	2	3	4
summary	count	mean	stddev	min	max
popularity	3855	1.1914630350194564	1.4750267558885362	0.0	32.99
budget	3855	3.720182829156939E7	4.220290791494386E7	1	425000000
revenue	3855	1.0765893266977951E8	1.7652480956117198E8	2	2781505847
runtime	3855	109.21582360570687	19.922165844758034	15	338
vote_count	3855	527.6119325551232	879.8683744573902	10	9767
vote_average	3855	6.167859922178991	0.7950397346338067	2.2	8.4
release_year	3855	2001.2632944228276	11.281989129282543	1960	2015
budget_adj	3855	4.4236299177868016E7	4.480402733045216E7	0.97	4.25E8
revenue_adj	3855	1.3702938610299385E8	2.16094430035931E8	2.37	2.82712375E9

Figure 1.2.1. Descriptive values

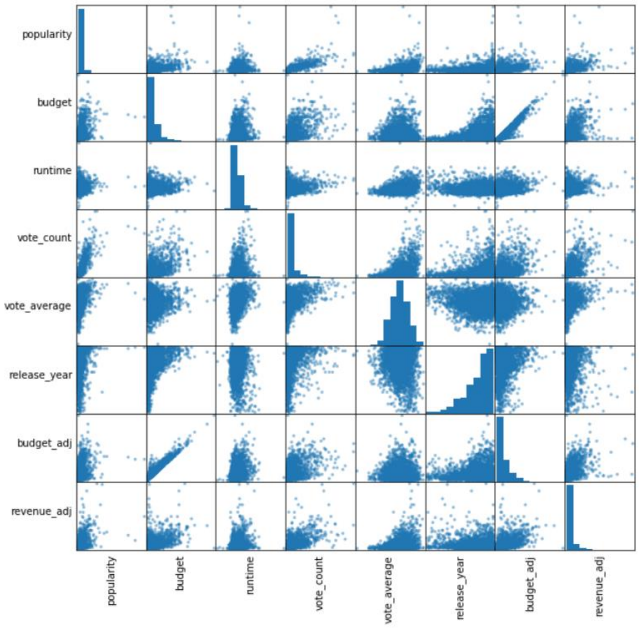


Figure 1.2.2. Scatter matrix

By having descriptive values from the raw datafile, it is possible to compare values from the regression model to validate if the model performs well easily. Furthermore, by checking the scatter matrix, if the adjusted revenue has a linear correlation with other independent variables roughly.

```
Correlation to revenue for  popularity 0.5469596708153259
Correlation to revenue for  budget 0.5334967276540608
Correlation to revenue for  revenue 0.9008921777149419
Correlation to revenue for  runtime 0.2806043799608099
Correlation to revenue for  vote_count 0.6547128315332141
Correlation to revenue for  vote_average 0.26699608475255326
Correlation to revenue for  release_year -0.1032602504356872
Correlation to revenue for  budget_adj 0.5704661221226086
Correlation to revenue for  revenue_adj 1.0
```

Figure 1.2.3. Correlation values

If correlation coefficient is close to 1, it means that there is a strong positive correlation. When the coefficient is close to –1, it means that there is a strong negative correlation. For this analysis, except the popularity, budget, vote_count, and budget_adj, other variables do not have values more than 0.5. Since the budget and budget_adj have the same meaning, only the budget_adj is used.

1.3 Data analysis

In this analysis, Apache Spark's spark.ml linear regression is used in the databricks site. To use the ml linear regression, a vector assembler is conducted to make the variables usable. Additionally, the training and testing data portion is set to 70% and 30% respectively.

summary	revenue_adj
count	2638
mean	1.3735791056167182E8
stddev	2.1232510310269985E8
min	2.37
max	2.506405735E9

Figure 1.3.1 Descriptive values from the training dataset

Since correlation coefficients are not close to 1, low R-Squared value is achieved which is 0.497437 from the training dataset and 0.535711 from the testing dataset. RMSE is 150492315.282752 from the training and 1.5266e+08 from the testing dataset respectively. RMSE measures the differences between predicted values by the model and the actual values. Interestingly, it is found the descriptive values from the raw dataset and training dataset are not much different. The variable that has the highest difference is the max value.

In conclusion, a linear regression model is not a suitable model to predict the adjusted revenue using the numerical variables that are given.

2. Random Forest Classifier

2.1 Homepage-Analysis

Having a homepage gives opportunities to attract new audiences.Looking at this data set you may think homepage is not very important to movies but doing this analysis will bring with it a higher revenue interest to have homepage.

2.1.1 Data preparation

This dataset has categorical and numerical columns (21 total). Homepage and runtime provide some information as below.

```
%sql
select runtime,homepage,count(*)from movie_analysis group by runtime,homepage
```

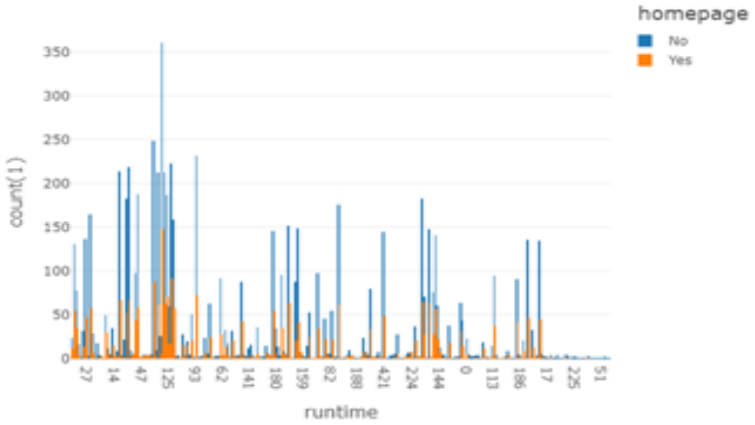


Figure 2-1 runtime - homepage counts

DataFrame Pipeline (feature transform:Transformer 1, Transformer 2, Transformer N etc.)Transformers use in OneHotEncorder and VectorAssembler to data preprocessing.Each Transformer stays within stages

My target column homepage turns to value as Yes=1 No= 0 then assigns it to stages. Then fit in to train and test data to evaluate the model. In this case No value is high that means imbalance data.

2.1.2 Data Analysis

LogisticRegression and RandomForestClassifier : check the precision in and recall compared with both models.

```
Accuracy: 0.9998592936541438
FPR: 0.0003825798917157537
TPR: 0.9998592936541438
F-measure: 0.9998592820111872
Precision: 0.9998592936541438
Recall: 0.9998592936541438
AreaUnderROC: 0.9999564935229732
```

Figure 2-2 - LogisticRegression

```
display(results.describe())
```

	summary	raw_prediction	revenue_adj	Precision
1	count	3032	3032	1363
2	mean	0.2763852242744063	5.0417462313275665E7	0.004425964665219787
3	stddev	0.44728337295671905	1.4244865725013855E8	0.11401964234587109
4	min	0.0	0.0	0.0
5	max	1.0	1.90272313E9	3.5097166667365567

Figure 2-3 -RandomForestClassifier

In conclusion, homepage analysis is ok to go with the above models. Revenue has a relationship with the homepage.

2.2 Random Forest Regression: using numerical variables

The random forest regression model was built by leveraging all available numeric data as input. This approach was considered as columns that had non-numeric or non-boolean data types often contained multiple string values concatenated with delimiters. It would have presented a significant challenge to analyze and transform data in such columns into meaningful input without incorporation of external datasets. For example one such column was *cast* which has a list of all actors and actresses in a film. As it's possible that cast members can have the same names, without a unique identifier and additional information regarding each cast member, such as popularity, experience and other attributes that may entice an audience to watch a film featuring the cast member, columns with such data were deliberately omitted from this model.

2.2.1. Data preparation

Nonetheless, as this data was readily available instead of using raw data, new features were developed and incorporated as input variables. See below for list:

1. Feature #1: Number of actors - count of cast members in a film
2. Feature #2: Number of genres - count of genres associated with a film
3. Feature #3: Has a tagline? - boolean value to indicate whether the film has a tagline

The rationale behind the above developed features are as follows. For feature one, number of actors, we hypothesized that having a larger crew has a positive correlation with revenue, as a larger cast means the film appeals to a larger fan base. For feature two, the same hypothesis holds as the more genres a film is categorized as, for example, on paid streaming service, that would imply this film has more visibility appearing across multiple genres thus increasing the odds that an user pays for viewership, further generating revenue. And finally feature three, has a tagline, checks for existence of a non-blank tagline value in the input dataset. The underlying hypothesis for this feature was that a tagline would provide additional information about the nature of a film which may convert people with exposure to the advertising content for the film to a paying customer.

2.2.2 Data Analysis

The base attributes used to train the model are - popularity, runtime, votes, votes_average and budget_adjusted. After training the model on them, the above stated features were incorporated into the model using a stepwise approach. The regression model was run with the following parameters, where maxDepth = [3,5,8] and numTrees = [5,10,15]. There was limited control over these two variables due to limitations in computing capacity of the

underlying infrastructure available with the community edition of databricks. Hence, more efforts were focused on improving the model through altering features from the dataset.

The table below lists R-squared and root mean squared error values that were used to evaluate the model performance on test data while altering input features:

Model	Input Features	R-squared	Root Mean Squared Error (RMSE)	numTrees	maxDepth
1	Base attributes = popularity, runtime, votes, votes_average, budget_adjusted	0.88386	64550939	15	8
2	Base attributes + feature 1	0.90713	54221949	15	8
3	Base attributes + feature 1 + feature 2	0.91147	53410535	15	8
4	Base attributes + feature 1 + feature 3	0.88326	67084100	10	8

Table 2-2-1 - Random Forest Regressor Model Evaluation

For each set of models, the best performing model always had 15 trees with maximum node depth of 8 which are the upper limit for both the numTrees and maxdepth parameter, which could be indicative of further potential for optimization. However the best performing model from the result set was model 3 which used all base attributes along with features, count of cast members and count of genres which yielded an R-squared value of 0.91147 and a root mean squared error of 53,410,535. These results also imply the original hypotheses for feature #3 which checks for existence of a tagline and its correlation to adjusted revenue isn't as strong as we believed it to be as it resulted in a reduced R-squared value.

3. XGBoost Regression

The homepage variable is considered for the XGBoost Regression model. That is, whether the movie has a dedicated homepage or not. The big box office movies usually have their own homepages. Next, the data is filtered to keep only movies with non-zero, positive inflation-adjusted revenue and budget.

3.1 Data preparation

Currently, there is not a lot of content on feature selection and importance for XGBoost using Pyspark. We decided to simply use the following numeric and indicator variables as features, as it would make sense to include from the business point of view:

- Popularity
- Homepage (Yes or N/A)
- Runtime (in Minutes)
- Vote count
- Vote average
- Release year
- Inflation-adjusted budget

A further ~400 records are dropped due to null values in Runtime and Vote count. Including null values causes error in feature vector assembly.

Next, the Vector Assembler is performed. A new features vector column is created in the data frame. Please see below snapshot for examples of the dataset to be used for model building.

	revenue_adj	popularity	homepage	runtime	vote_count	vote_average	release_year	budget_adj	features
1	1392445893	32.985763	1	124	5562	6.5	2015	137999939	▸ {"vectorType": "dense", "length": 7, "values": [32.985763, 1, 124, 5562, 6.5, 2015, 137999939]}
2	348161293	28.419936	1	120	6185	7.1	2015	137999939	▸ {"vectorType": "dense", "length": 7, "values": [28.419936, 1, 120, 6185, 7.1, 2015, 137999939]}
3	271619025	13.112507	1	119	2480	6.3	2015	101199956	▸ {"vectorType": "dense", "length": 7, "values": [13.112507, 1, 119, 2480, 6.3, 2015, 101199956]}
4	1902723130	11.173104	1	136	5292	7.5	2015	183999919	▸ {"vectorType": "dense", "length": 7, "values": [11.173104, 1, 136, 5292, 7.5, 2015, 183999919]}

Figure 3-1 - Dataset tu use in XG Boost

3.2 Data Analysis

The data is then split into 70% for train and 30% for test. The XGBoost Regressor model is staged with a maximum iteration of 10, to avoid overfitting. The pipeline is used to automate the processes of model training. The model is now being trained on 70% data. This step only takes a few seconds.

As the model is built and the 30% testing data has been held out, the performance of the model can be tested using this partition. The revenues for these movies are predicted and compared with the actual revenue_adj. See below:

	popularity	homepage	runtime	vote_count	vote_average	release_year	budget_adj	revenue_adj	prediction
921	0.372231	0	144	74	6.1	1974	28744845	117337343	174065153.42440897
922	1.107167	1	101	546	6.7	2004	32324471	118164853	108174086.19818868
923	1.180341	1	105	549	6.6	2007	21033372	123844633	102325349.87821427
924	0.815044	0	136	177	6.3	2004	115444540	124799531	139422806.22191894
925	1.3747	1	102	187	6.3	2007	78875144	125148993	107586943.89726658

Figure 3-2 - Revenue prediction

Two main metrics are used: RMSE and R squared. They can be evaluated respectively, see below.

```
1  evaluator = RegressionEvaluator(
2      labelCol="revenue_adj", predictionCol="prediction", metricName="rmse")
3  rmse = evaluator.evaluate(predictions)
4  print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
5
6  evaluator = RegressionEvaluator(predictionCol="prediction", \
7                  labelCol="revenue_adj",metricName="r2")
8  print("R Squared (R2) on test data = %g" % evaluator.evaluate(predictions))
9
10 gbtModel = model.stages[1]
11 print(gbtModel)
```

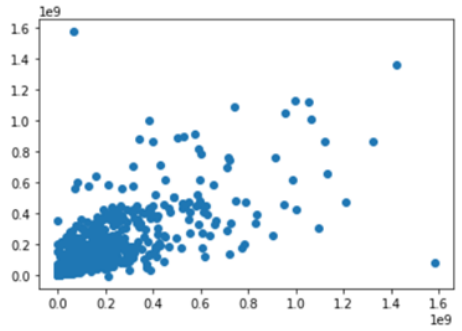
```
▸ (2) Spark Jobs
Root Mean Squared Error (RMSE) on test data = 1.35876e+08
R Squared (R2) on test data = 0.504984
GBTRegressionModel: uid=GBTRegressor_44b803fae0f2, numTrees=10, numFeatures=7
```

Figure 3-3 - Performance metrics

Lastly, despite a lot of visualisation functionalities not being available in PySpark at this moment, it is still interesting to us to see how our predictions correlate with the actual revenue in a chart. The data frame containing the predicted revenue and actual revenue is converted to a panda data frame for graphing the scatter plot. Again, this is only based on the 30% unseen data. See below.

```
1  plt.scatter(test_panda['revenue_adj'],test_panda['prediction'])
```

```
Out[56]:
```




```
<matplotlib.collections.PathCollection at 0x7f56446f8590>
```

Figure 3.3 - Scatterplot revenue-adf vs prediction in XGBoost

Conclusions

What does work to predict the revenue of a movie and what does not work? This is our first question and it requires an answer in two different levels.
First of all, we should talk about the technical approaches in order to get our predictions. We have used three different approaches to solve our problem: linear regression, random forest and XG Boost. But in order to understand the performance of these different approaches, we must compare what we have done with the data.
Our dataset was pretty neat with not a big number of missing values. We have dropped those which affect our main purpose: the ones with no data or 0 in budget and revenue.
Beyond that, we have pivoted some common data to predict the revenues, and that we can call our "core" data:

- Budget adjusted
- Popularity
- Vote count.

These have a very predictable relationship with the revenue. There are other features we have used in some of the approaches, like:

- Vote average (random forest 1 and 2 and XG boost)
- Runtime (random forest 1 and 2 and XG boost)
- Release year (random forest 1 and XG boost)

And there are two additional data columns which are specific or of special relevance for their respectives approaches:

- Homepage (random forest 1, but also used in XG boost)
- Cast, genres and tagline (random forest 2).

This gives us a different panorama to compare different predictions and models.
First of all, we have the Linear Regression approach: it uses our core data to make a prediction. The result is not high with an R2 of 0.497. Maybe it's the model itself or maybe the lack of additional data beyond these three. Anyway, it should be a good baseline for further comparisons. We know that budget, popularity and vote count, at least in a linear regression this doesn't give a good prediction, so we should go further.
As the linear regression model doesn't get a good prediction we are going to use random forest but in two different ways.
First of all, we try to find out if the existence of an official website of a movie has any relationship with its final revenue. The result is positive, there is some kind of relationship and it is consistent when we use a logistic regression to compare both elements. This is an interesting find but with a limited impact in the total prediction and strategy of the film. So, we have used another approach for the random forest related with the total number of our core data but using also some others related with the product design of the movie which may improve the likeability of the movie for a larger part of the public. In this case, we first compared the core values in our random forest approach (with runtime and vote average) and we got a far better result than the linear regression model, reaching up to 0.88 as our R2 value. This is telling us that, at least in this case, the random forest is a better fit for our dataset using this data. On the other hand, we have added additional features as actors, genres and tagline. A higher diversity of genres or a larger number of actors can diversify the appeal of the public in

the movie and, therefore, get a better prediction. In this case, we have observed that the number of genres and the number of actors act as good predictors altogether with our core data. Tagline seems to not improve the results that the previous two features get us.
Finally we have analysed XG Boost as a solution which could improve results of linear regression and, maybe even that of Random Forest. In this case our data included homepage, runtime, vote average and release year. In this case, using more data and with a different model than the linear regression we get a pretty similar result with R2 equal to 0.50 which is worse than random forest even including data that we have used in both models.

We can conclude that the revenue of the movie is something we can predict better using random forest. Not just because of the minimal use of data in the linear regression, but also with a similar approach of features used in XG boost. Actually, using XG boost with more features doesn't get a far better result than a classical linear regression approach.

About the random forest what we can see is that with the simple "numerical" values we get a good result close to 0.9 R2. Anyway, using non numerical data features, we get a better prediction, and this is when we use factors associated with getting a broader public such as different genres and actors.

So, we began this document talking about how we can predict if a movie will get good revenue. The first idea is pretty simple and basic: we need a big budget, a large number of people who watch it and that engage them, at least enough to vote for it positively or negatively. Also we know that the higher the mean, the more likely it's to generate good revenue and also whether it has a website. But also we have found that a movie with different insights such as a large number of actors or a diverse mix of genres can reach a larger quota of public and so, a better revenue. And we know this using a classification model such as the random forest because the diversity or variety of movies needs classification in order to get a better prediction than a simple and classical linear regression.

Appendix

Notebooks:
Linear regression
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/2360406909121662166/1894993239916400/511646904879043/latest.html

RandomForestClassifier homepage Analysis https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3502107196020110/1336852583605608/5483278593984256/latest.html

Random Forest non numerical values
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/1813937161281675/2430590801042781/3514329913855704/latest.html

XGBoost Regression
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/5725144014341294/3456254292081874/7398960806603811/latest.html