TMDB MOVIE REVENUE MODELING

Key insights from movie metadata and earnings

Abhinav Chanana
Faculty of Mathematics
University of Waterloo
Waterloo, Canada
a2chanan@uwaterloo.ca

Danish Farid
Faculty of Mathematics
University of Waterloo
Waterloo, Canada
d2farid@uwaterloo.ca

Ishjot Singh Suri
Faculty of Mathematics
University of Waterloo
Waterloo, Canada
i2suri@uwaterloo.ca

ABSTRACT

With this project, we aim to analyze the Box Office Movie dataset (TMDB) and then use the data to model how much revenue a movie will make, given its features. Our dataset is a popular one that consists of thousands of films and their metadata from across several domains including data for the entire cast and crew as well as frequently used keywords during marketing.

While the quality of the movie itself is an obvious indicator of how much money it will make, other indicators can have a big impact as well. This includes the actors, the budget, premarketing, locations of filming, release time, etc.

Our approach to learning more about the dataset was twofold, involving a deep dive into looking at correlation between factors and modelling the data after cleaning it and leaving more usable information.

With regards to our first task, we saw some interesting links between several factors including one that highlighted how, often lower budget films ended up with higher user ratings. We will elaborate on this further over the course of this report.

For modelling revenue, we used Simple Linear Regression, Random Forest Regression and the boosting algorithm such as XG Boost, out of which XG Boost performed the best, achieving 80% accuracy on test data.

KEY WORDS

Linear Regression, Random Forest Regression, XGBoost, Pyspark, Python, Classification, Predictions, Data Science, Box Office, Movies

INTRODUCTION

The entertainment industry is growing at an exponential rate every year. According to IMDb 11,912 were released in the year 2018 which was an investment of 20.8 Billion USD. The scale of the investment makes it an interesting proposition to be able to predict movie revenues based on parameters like cast, directors, native language, etc.

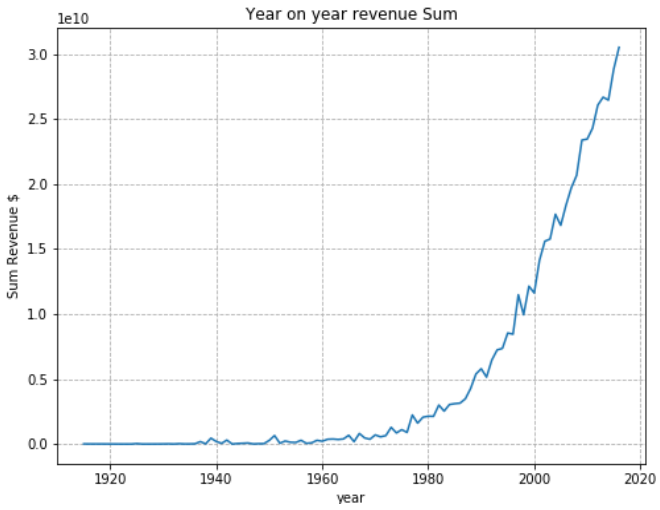

Figure 1.1: **Yearly growth of movie revenue collection**

This proposition would be even more exciting if we could predict revenues **before** a movie was to launch. And while, to some degree, one can imagine that a movie with a large budget for marketing, top end cast and capable director should do well with audiences, how much does each factor impact a movie's success exactly? And what are some of the hidden variables that can contribute to this success without being entirely obvious?

Our dataset (about 1GB in total, all metadata included) is ripe with features that include both the obvious predictors and, we hope, the more subtle ones. We begin our analysis with the PySpark library in mind. Given that our dataset is large and spread over several files, we feel that the PySpark library with its ability to scale across multiple machines can help speed up key elements of our preliminary analysis. Alongside we will also use the Pandas data science toolkit.

Scalability is important given that our dataset is one that is dynamic and will only get larger at a much faster yearly rate given the industry, we'd like to conduct analysis that can scale.

Given that revenue is a continuous data (the amount of money a movie makes both at the box office and in sales online and in stores) we have to treat the prediction side of our project as a regression task, which can be more open ended and more difficult to tune. We continued and worked with this fact in mind, however, such a task can be converted to other kinds such as classification (binary or multi-class) using mathematical transformations.

Following is a histogram for how our data is. Currently, not a lot of movies are made with over a 100 million USD budget and so there is skew. Using this data directly for classification would have a highly skewed dataset, not to mention having to bin it manually.

As a proof of concept, we adjusted the data using a logarithmic function and ended up with the result that follows below.

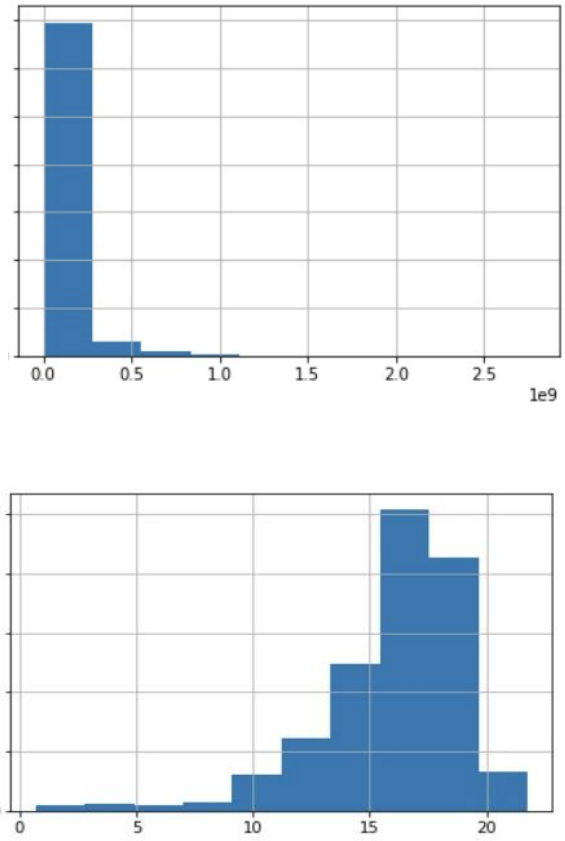

Figure 1.2: **Before and After the log function to create classes for classification from continuous data**

The data naturally formed into bins itself and assumed a scale that forms "classes" itself, and so we could approach this also as a classification task. An even simpler approach would be to see which movies 'broke even' (earned an amount equivalent to their initial budget in revenue) 2 or 3 times over and define that as a metric for "success", with anything less being "failure". For our current work however, we chose to continue with a regression problem.

Methodology

1.1 Data Collection and Preprocessing

We have used the TMBD Box Office dataset from Kaggle [1]. This is a combination of smaller divided files. Specifically, they contain movie meta-data, cast data, complete crew data, movie keywords data, as well as movie rating data.

Alas, as with any data science project, the target 'revenue' variable was missing from a large number of rows in our movies metadata file. To bypass this and additionally supplement our data with more varied information for user ratings, we wrote several scrappers to scrape box office site (www.boxofficemojo.com) for the missing revenue data and used an IMDB API to access rotten tomatoes user ratings.

Once all data had been locally acquired and indexed by IMDB movie IDs, we used PySpark dataframes to merge and combine all data into a single large dataframe, in a distributed fashion. PySpark features RDD's and DataFrames, both of which distribute automatically making the "scaled" processing simpler.

1.2 Data Cleaning and Augmentation

Following our final data merge, irrelevant columns such as "groups" and "adult" were removed, since they served no purpose to our analysis and mostly contained "NaN" values.

Since we wanted to use regression, the data had to be in numerical format. Things like "release date", "keywords", "overview" and string dictionaries for "cast" and "crew" would not suffice. Following is how we tackled some of this unworkable data and turned it into something a model could learn from.

1.2.1 Text Fields. Dates were converted into months, and then turned into one-hot-vectors which were merged into our dataframe. We also did the same with the movie genre fields, creating vectors of 1's and 0's and merging them into each record.

1.2.2 Dictionaries in Strings. Much data was contained in a logical dictionary/JSON format but was read as a string. Pandas literal_eval function to evaluate strings 'literally' as to what they contain was helpful here, and we were then able to use lambda functions after, to extract only the names of the cast and the names of directors.
The lists of production companies and countries were also in such a format. However, they were converted into counts (how many countries/companies involved) and we dropped any text-based information altogether.

1.2.3 Data Scraping. Some columns of information for the movies were not available in our dataset but were available on some public websites. This included our target variable 'revenue', as well as others including some user ratings. A few scrappers were developed with the 'beautiful soup' library for fetching the data from online adding it to the dataset. Thanks to this, 'gaps' in our working dataset were minimized.

1.2.4 Data Augmentation. To get useful information out of our cast variables, we created a count for every cast member, for each film they appeared in. Each film then had it's "cast popularity" field populated with the sum of each of the cast's film count, thus givings us its popularity. We believe an actor appearing more often than others will have more 'star power'. We saw Tom Hanks very close to the top of the list when ordering this data, which makes sense.

1.2.5 Other data. Rows with NaN's in key data such as 'cast' or 'revenue' (those that remained) were removed. In addition, absurd values were filtered out, such as a movie having a budget of 1 USD.

Thus, our final list of variables was as follows (some condensed):

budget	original_language	popularity	production_companies
vote_average	revenue	runtime	production_countries
vote_count	cast	release_month	genre
spoken_languages			

A full explanation for each row is given in the Appendix

1.3 Data Visualization

After having cleaned the data, we began our first phase, which was to explore the data and see what interesting links were present. Our first few plots are simpler than those to follow.

1.3.1 Budget and Revenue

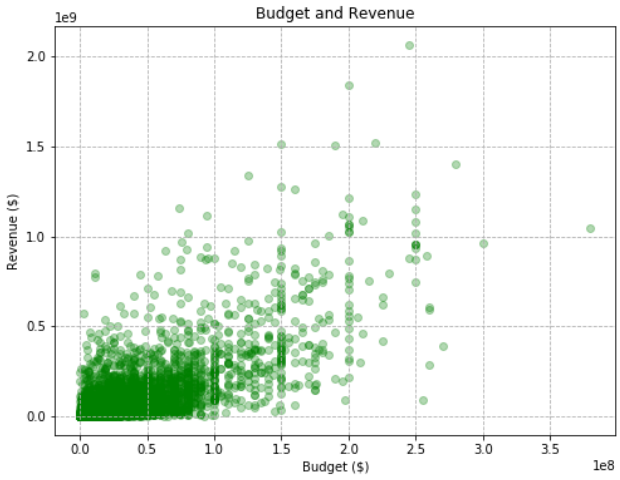

Figure 3.1: Graph showing plot between Revenue and Budget

Budget and Revenue have a very easy to understand relation. A greater budget leads to better production quality and a better product for the audience. It also allows a film to have more star power and a more lavish marketing rollout, bringing more people to the theaters. In addition, the biggest budget films in recent years have typically been crowd-pleasers from studios like Marvel [2], which rake in an extremely large amount of revenue. This relation supports that.

1.3.2 Run Time and Revenue The run time for any movie has a significant impact on the revenue. While this can

vary from genre to genre, we wanted to get an overview of what the link between the two is given our large and varied dataset.

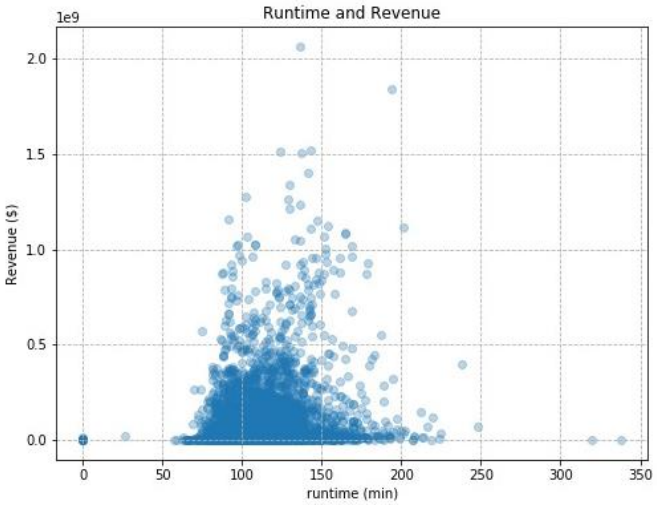

Figure 3.2: Graph showing plot between Revenue and Runtime

Given that the meat of the points is between 100 and 150 minute runtime, many movies having 100 mins have a high chance of grossing over 500 million USD . Movies having time < 75 mins or more than 175 minutes gross less. This is in line with the lengths of recent blockbusters at the box office as well [3].

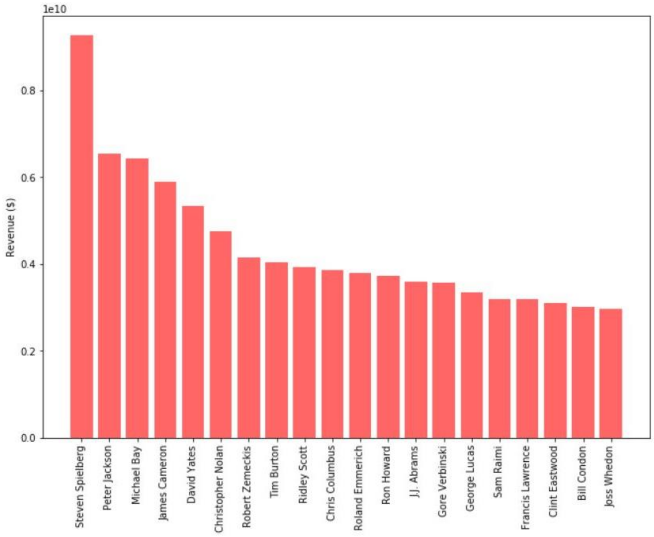

Figure 3.4: Top 20 combined grossing directors for films

1.3.3 Top director (Revenue) In terms of revenue purely, again, bringing people to the cinemas and having the most sales in the physical and streaming market, Steven Spielberg leads the pack here. Followed closely by Peter Jackson, Michael Bay and James Cameron. This graph was nearly a sanity check for our ability to group data correctly, the names that appear are those that we would expect to see.

1.3.4 Top director (User Rating)

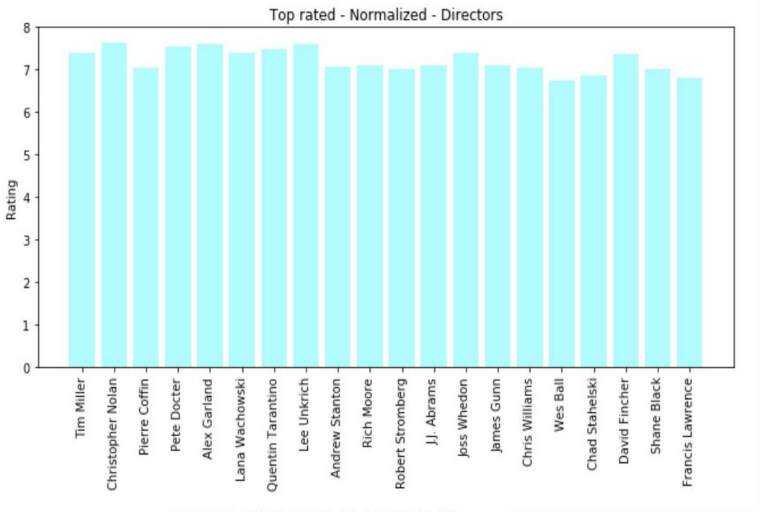

Figure 3.5: Top 20 user rated (avg) directors for films.

As per graph it can be seen the best director is Christopher Nolan, in terms of the user ratings on average for their films. This is not surprising given the nature of his films is always out of the box. J.J Abrams is present here, after having directed several of the new 'Star Wars' films which, thanks to a strong fan base, are often received very well.

This rating is a combination of those from IMDB from users and those from Rotten tomatoes, normalized.

1.3.5 Budget and User Ratings

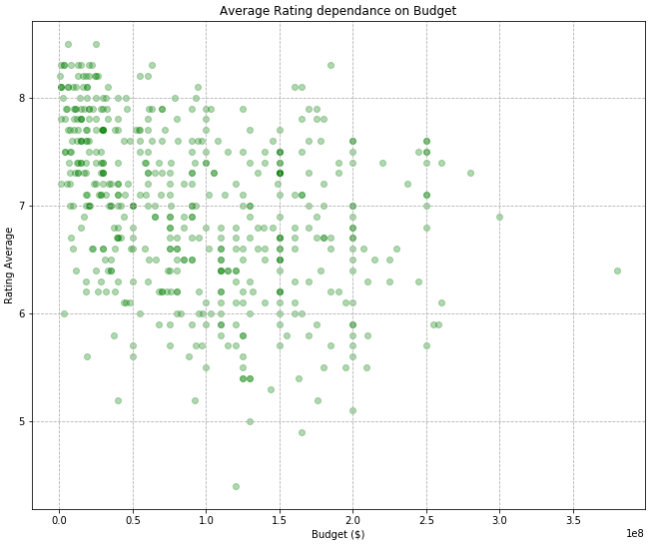

Figure 3.6:Plotted relationship between budget and user ratings

Upon observing user ratings plotted against movie budget we can see that movies with budgets less than 50 million USD seem to have a higher user rating average, around 7.5 while, high budget movies have a lower average and a more widely spread average. It seems as the budget increases the average vote goes down. This may be due to the fact that less budget films concentrate on the script and content delivery while big banner movies tend to invest in the star factor and marketing techniques.

One look at the Top 10 list on IMDB confirms this [4]. The highest rated film had a budget of about 40 million USD, adjusted for inflation.

1.3.6 Language and Movie Origin

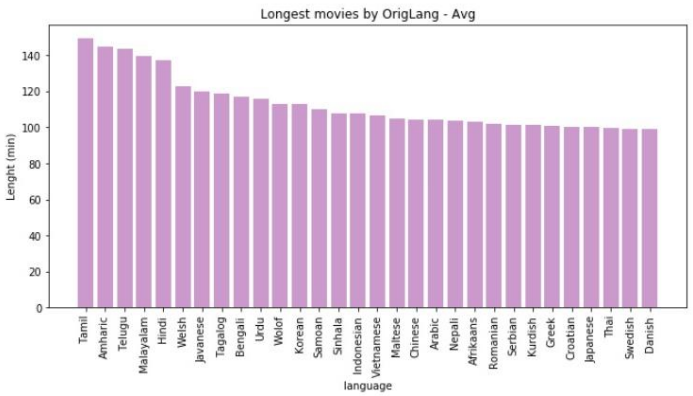

Figure 3.7: Plot between language and movie origin

Grouping data by language of origin shows a general trend in data overall. Movies originally filmed in Asian or European languages appear to be the longest. In fact, English comes a fair bit lower in this ordering, in terms of avg length. This was not exactly surprising, given that Hindi movies from 'Bollywood' are typically much longer than their western counterparts, and that they would influence nearby regions to enjoy and create similarly long film media.

1.4 Predicting Revenue

Post analysis, we've seen several connections in the data, some more obvious than others. We will now attempt to build several models to try and predict movie revenue data.

Before we attempt to model however, we have to bear in mind that all the data is currently unscaled. For learning models, it is a good practice to scale data before training. This gives the model more idea of what the min and max possible values are for any given variable for better establishing range. We used the min max scaling to achieve this for all continuous data columns, including revenue. The same scaler can be used to inverse scale predicted data back to normal.

This is where any augmented data was also handled properly. Every film's 'cast_popularity' value was calculated as the sum of each of its cast members film appearances in total, as specified before. Some films thus had a much larger value for cast_popularity that would make it hard to determine an upper bound, but min max scaling fixes that.

We used 4 Machine Learning models to try and predict revenue. We split the data 80/20 and used the 20% as our test set to measure final accuracy on.

1.4.1. Simple Linear Regression:

1. We followed a **linear** approach to modeling the relationship between revenue and all the other features in our dataset namely budget, vote count, star power etc.
2. We used the PySpark library's function, **org.apache.spark.ml.regression.LinearRegression** implementation of the linear regression in spark.
3. Our regression model predicted revenues with **74.5%** accuracy. The results of the same can be found in the graph below.

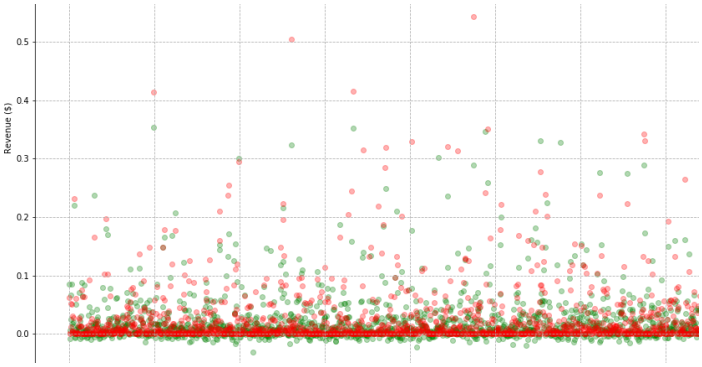

Figure 3.8: Prediction (orange) and Ground Truth (Green)

In figure 3.8, for every index on the x-axis we can see that the dots are not very far, justifying the ~75% accuracy of the model. This is a trend that mostly followed throughout later on.

1.4.2 Random Forest Regressor:

1. A supervised learning algorithm which uses ensemble learning method for classification and regression.
2. As random forest is a meta-estimator which fits over part of the training data and averages the results from all the models. This leads for the algorithm to have better efficiency.
3. We were able to **76.4%** accuracy using the algorithm.

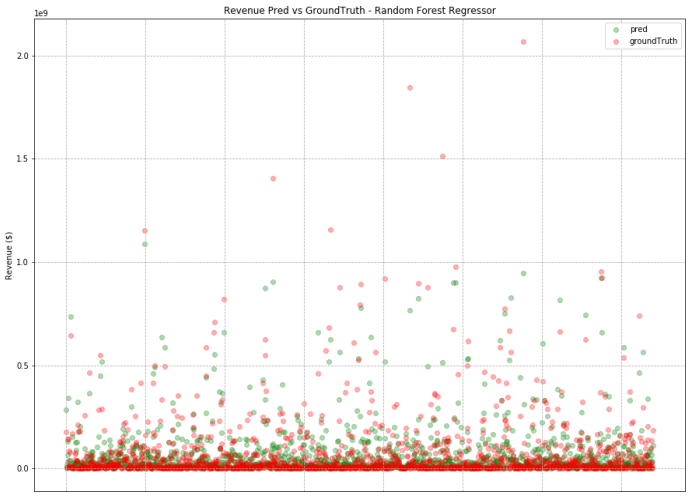

Figure 3.9: Prediction (orange) and Ground Truth (Green) RF Regressor

1.4.3 K Neighbor Regressor:

1. K Neighbor Regressor uses the concept of feature similarity to predict values based on training data.
2. This model gave us the least accuracy as it requires a large amount of similar data and it doesn't predict accurately for outliers.
3. We achieved an accuracy of **45%** which was the least across all our models.

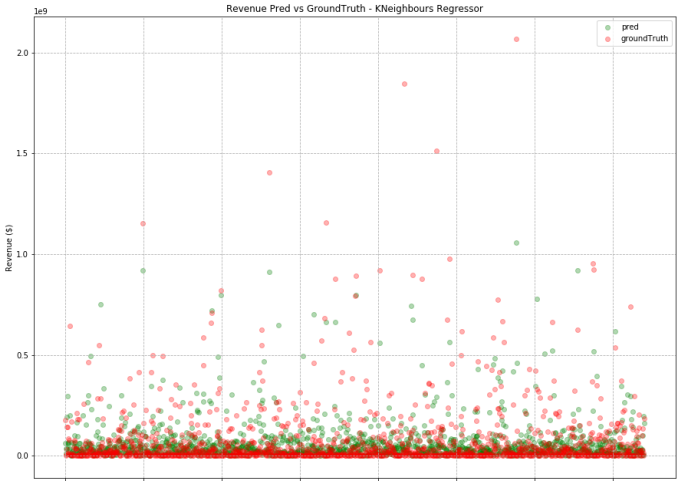

Figure 3.10: Prediction (orange) and Ground Truth (Green) - Greater distance between predictions and GT

1.4.4 XG Boosting Algorithm:

1. It is an implementation of gradient boosting machines created by Tianqi Chen, it is one of the boosting algorithms used commonly on Kaggle competitions to get high efficiencies[7].

2. XG Boosting gave back the best accuracy, beating out other models by up to about 5%

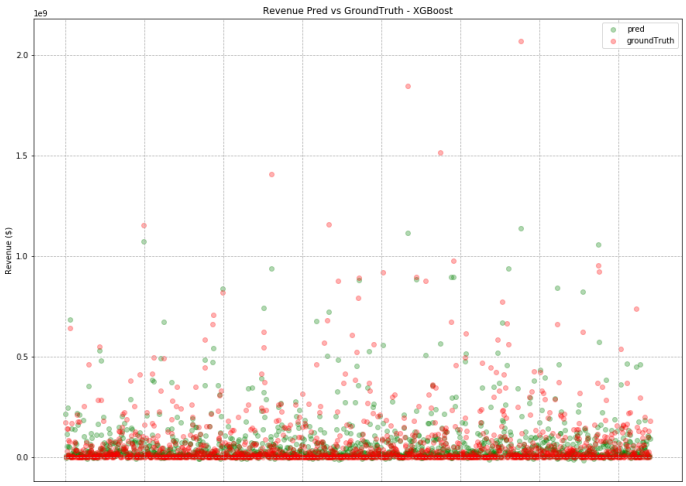

Figure 3.11: Prediction (orange) and Ground Truth (Green) - Dots closer due to increase accuracy

We can see dots nearby in all of the plots, except for in the KNN one. XGBoost gave us the best results and we found that in some ways we were able to verify this visually as well.

Results and Conclusion

With regards to our primary focus, the ML models, we were able to train 3 with high accuracy. And the advantage of using linear models is that we are able to "open them" to check the coefficients assigned for each variable.

We passed a total of 38 scaled and one hot encoded variables out of which our best performing model identified 14 variables and useful in a linear combination to predict revenue.

We will divide these variables into three categories:

1. High Impact: **Budget**, **Ratings_Count** (the number of user reviews)
2. Medium Impact: **film popularity** (rotten tomatoes), **cast popularity** (our augmented data variable) and a mix of one hot encoded **'Adventure'**, **'Family'** and **'Fantasy'** genres.
3. Low Impact: **Production Companies** (count), **runtime**, **spoken_lanuages** (count), **one hot Encoded Romance Genre**, and one hot encoded **Apr, Jun, May releases times**

Given this insight into what variables provide what kind of impact on the revenue generated, this model can have potential in predicting future information.

In addition, the dataset itself also provided a other insights such as the different nature of films in different parts of the world, longer in Asia and Europe, especially those in Tamil. Another such insight involves the max revenue generating movies and their runtimes being the median runtime of the total. Perhaps this is why most movies are around the same length.

Appendix

Full List of cleaned and final Variables:

Variable	Description
Budget	Films original budget
Original_language	The language it was shot in (indicative of where it was made)
popularity	Taken from Rotten Tomatoes
production companies	Number of Production companies involved
production countries	Number of countries where the film was produced
release_date	12 one hot encoded month-wise columns (1's and 0's)
runtime	Film duration in minutes
spoken_languages	Number of languages spoken in film
user_avg_votes	The avg user rating online
user_vote_count	Number of users having rated film online
Genre_Encoding	20 one hot encoded genre columns (1's and 0's), each for a different genre

REFERENCES

[1] https://www.kaggle.com/rounakbanik/the-movies-dataset

[2]https://towardsdatascience.com/random-forest-and-its-implementation-71824ced454f

[3]https://www.businessinsider.com/2021-movie-release-schedule-could-break-records-marvel-dc-avatar-2019-11

[4]https://www.businessinsider.com/are-movies-getting-longer-2016-6

[5]https://www.imdb.com/list/ls003992425/

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

[7] http://delivery.acm.org/10.1145/2940000/2939785/p785-chen.pdf?ip=207.107.159.98&id=2939785&acc=CHORUS&key=4D4702B0C3E38B35%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35%2E6D218144511F3437&__acm__=1575467634_a15cb75c74470c85750419741386143f