Predictive Model for Movie Revenue

Final Project DSC 148 WI23

Vibha Sastry
University of California, San Diego
vsastry@ucsd.edu

Anika Garg
University of California, San Diego
agarg@ucsd.edu

Introduction

Movie revenue is commonly viewed as a measure of a movie's success. A movie's profitability coincides with its popularity and longevity in the film industry. Predicting a film's revenue is essential for making releasing decisions, casting choices, and more. This project attempts to assess a movie's profit through use of other movie information.

1 Dataset

1.1 Identify Dataset

The dataset we obtained from Kaggle is called 'The Movies Dataset', and it contains information on movies and movie ratings from viewers. The dataset is composed of multiple files in csv formats with information on different aspects of the movie, including IMDb information, keywords about the movie's plot, and metadata on how the movie was made.
For this project, we utilized two of these files: 'movies_metadata.csv', in which each row is a movie object and each column is a different feature of movies, such as the production_company, and 'ratings.csv', in which each row is a user object, with the columns containing the movie the user reviewed and the user's rating. The 'movies_metadata' file contains information on over 45,000 movies, and the 'ratings' data contains 26 million ratings from over 270,000 different users. From this data, our goal is to determine how different aspects of a movie can be used to predict the amount of revenue that it generates.

1.2 Data Cleaning

After narrowing down on our goal, our first step was to merge the 'movies_metadata' and 'ratings' files into one dataframe, using the movie 'id' as the key to bring the two tables together. We then determined which features of the data were irrelevant to predicting revenue generated. Columns containing identifying features, including the title of the movie, the user_id of users who rated it, and links to the movie's poster and page on IMDb were dropped, because identifying features do not play a role in finding trends that are generalizable to all movies in the dataset. We also excluded the column 'adult' from our dataset because almost every single value in the column is 'True', and 'belongs_to_collection' because nearly 50% of the data in the column was missing.
Furthermore, we dropped several rows that contained values that were irrelevant to the dataset, including gifs and random numbers and strings. There were only a few such rows, and most of the values in them were unusable, so we determined that dropping these rows would not affect our overall data. We also removed rows where the budget of the film or the revenue of the film was 0, as these rows mostly contained null values in the rest of the columns of the dataset and are likely produced by data entry error.

Some numeric features of our dataset were stored as strings, so we converted these features to integers and floats depending on what was most appropriate. All of the numeric features of the dataset, such as 'vote_count', 'rating', and 'popularity', had several missing values, typically between 3-10 values for each column. After basic data exploration, we determined that the best method of imputation for the numeric features of our dataset is median imputation, because many of the distributions of the numeric variables were skewed in one direction and very few values were missing, so using the median would not change the data significantly.

Several of our features, including 'genre', 'production_companies', and 'production_countries' were stored as a list of dictionaries in a string. We used custom functions and regex to parse these features and convert them to strings that could be encoded in our predictive models. We derived the feature 'release_month' from the original feature 'release_date', which was in the format "Month DD, YYYY" to show how the month/time of year a movie is released has an impact on the revenue it generates, regardless of the year released.

Finally, after plotting the distribution of our target variable 'revenue' and doing the exploratory data analysis below, we determined that a log transformation was necessary for all of our numeric variables to normalize the distributions and put less emphasis on the outliers in our dataset when training the predictive models. We observed that the revenue data is right-skewed, making it less suitable for use with our evaluation metric. Log-transforming all the data will reduce skewness to make the data as valid as possible.

1.2 Exploratory Data Analysis

For the remaining features in our table, we examined the relationship between each variable and revenue. For the purposes of better understanding our visualizations and showing trends in our data, we plotted the variables (except revenue) before log transformation.

Rating. The 'rating' feature is the number of stars, from 1-5, that a user rated a given movie. From the boxplot we can see that there is a difference in the distribution of revenue among each rating, with higher rated movies tending to have a distribution that is skewed in the direction of larger revenue.

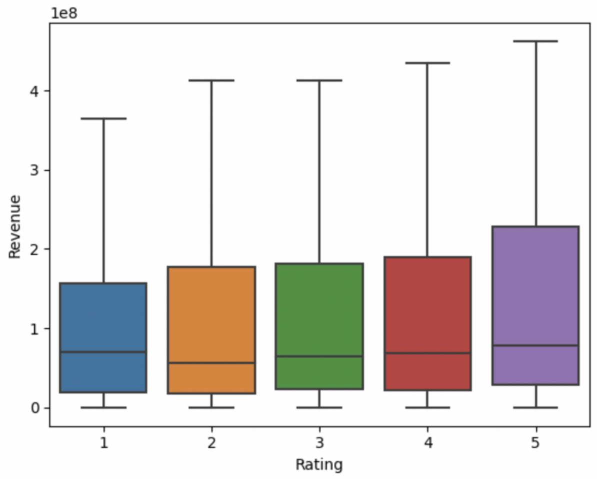
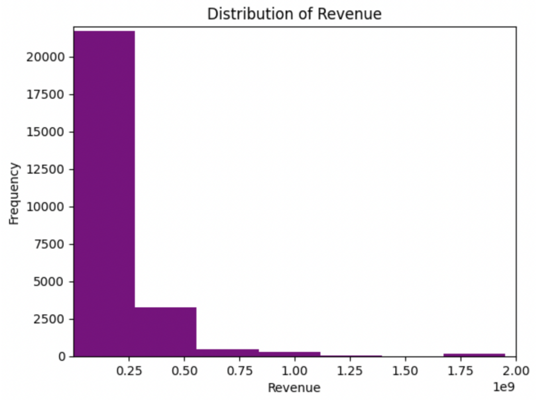


Figure 1: **Boxplot showing distribution of revenue for each star rating (excluding outliers in the dataset).**

Revenue. The 'revenue' feature of the dataset is a float indicating how much revenue the movie generated. Before log transformation, we can see that the distribution of revenue is heavily skewed right, with many outliers. After log transformation, the distribution of revenue follows a much more normalized shape and noise is reduced significantly.



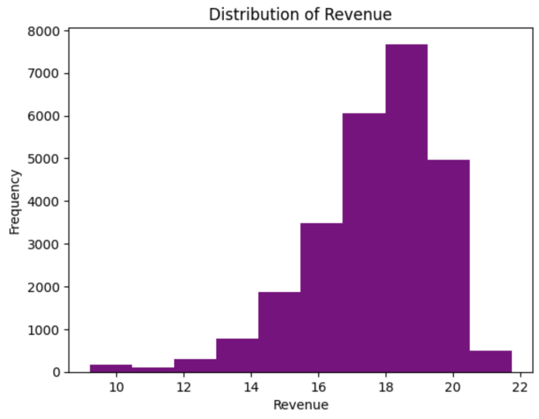


Figure 2: **Distribution of revenue, excluding large outliers (top) and distribution of revenue after log transformation (bottom).**

Budget. The 'budget' feature of the dataset is a float indicating how much money was spent creating the movie. In the scatterplot below, each point represents the budget of a movie and the revenue it generates. Using simple linear regression to form a line of best fit, we can see that there is a general positive correlation between the two variables, and that the majority of movies have a budget under $150,000,000.

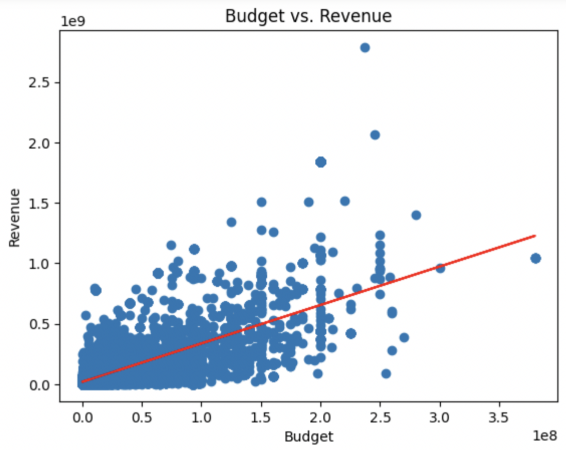


Figure 3: **Scatter plot showing relationship between budget and revenue of each film.**

Release month. The 'release_month' feature is a number from 1-12 showing which month of the year the movie was released. From the bar chart, it is clear that movies released in certain months tend to generate more revenue, particularly movies released at the end of the year in November and December.

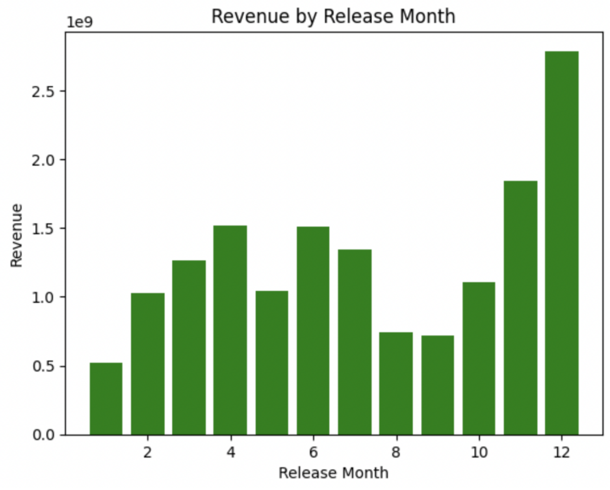


Figure 4: **Barplot showing average revenue generated for movies released each month.**

Vote count. The 'vote_count' feature of the dataset is an integer representing the number of people who voted on a given movie's IMDb page, where each vote contributes to the 'vote_average' feature. The scatterplot below shows a generally positive correlation between the number of votes on a movie's IMDb page and the amount of revenue that movie generated.

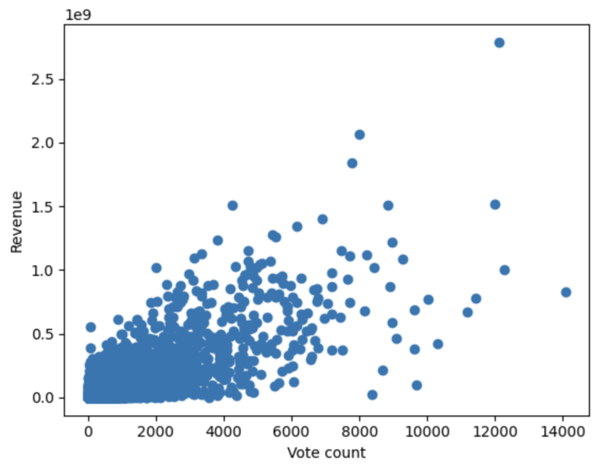


Figure 5: **Scatter plot of vote count vs. revenue.**

Original language. The 'original_language' feature of the dataset is a string with the language code of the language the movie is made in (i.e. 'zh', which represents Chinese). To examine how language plays a role in the revenue generated, we created a barplot showing the total revenue of movies for each language and found that movies in Chinese, English, and Korean are the top three languages with the highest total revenue.

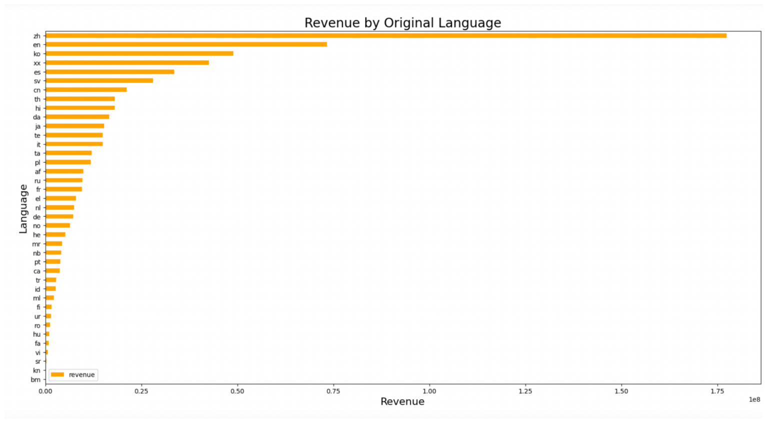


Figure 6: **Total revenue for movies in each original language, in descending order.**

The 'runtime' feature of the dataset is a float representing how many minutes long a given movie is. The scatterplot below shows the relationship between the runtime of a movie and how much revenue it generated, and we can see that movies are typically between 60 to 200 minutes, with movies longer and shorter than this runtime generating less revenue.

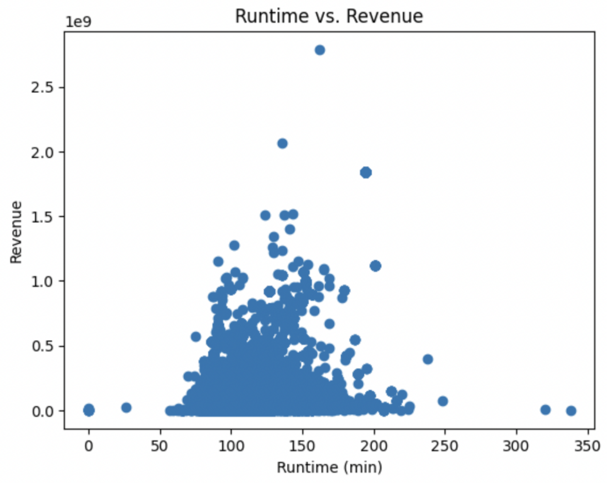


Figure 7: **Scatter plot of movie runtime vs. revenue.**

2 Predictive Task

After basic cleaning and engineering, **given the metadata and rating data of each movie, predict the revenue of the given movie.** This is a regression prediction problem.

2.1 Evaluation

Our evaluation metric is RMSE (Root Mean Square Deviation). The RMSE will be calculated on the original unit of measurement (revenue). Since the revenue data is right-skewed, we will be performing our regression on the log-transformed revenue data.

RMSE is a common evaluation metric which ranges from 0 to infinity, the lower the value the better. It is the square root of the average of squared differences between predicted and observed values. RMSE is useful because it can be calculated efficiently, it penalizes large errors, and it is more likely to detect predictions that stray further from the center.

2.2 Baseline

Data: merged (contains movie metadata and ratings merged)
Features involved: Only numeric variables are used in the baseline models, as after basic cleaning and feature engineering, it was apparent that many numeric variables had a linear relationship with revenue, and therefore would be a good fit for our baseline models. In addition, we would like to use linear regression as a baseline model, which restricts us to using only numeric variables in the baseline.
Models: We are using 4 baseline models: Linear regression, Random Forest Regressor, XGBoost Regressor, and LightGBM Regressor.

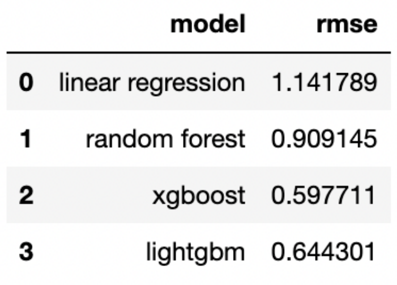
Linear regression. Makes predictions assuming linear relationship between target and prediction variables. This model will be appropriate as we saw in our EDA that many numeric variables have a fairly strong linear relationship with revenue. In addition it is a simpler model that provides a good basis for comparison with more advanced models we are using.

Random Forest Regressor. Makes predictions using multiple randomly selected decision trees and averaging results to create most accurate predictions. This model will be appropriate for our data as it can make use of non-linear relationships between the variables to produce more accurate predictions.

XGBoost Regressor. Makes predictions using optimized implementation of Gradient Boosting. Uses ensemble learning methods to train models on top of each other and produce the most accurate final outcome. Model is appropriate for the data as it will use decision trees to find the best path to target variables in an efficient manner.

LightGBM Regressor. Another gradient boosting ensemble model that works similarly to XGBoost Regressor, however performs better on larger datasets through use of histogram-based optimization. Model is appropriate as our dataset is very large and therefore may produce better predictions with the LightGBM model.

All baseline models are trained on the same training set created using sklearn.train_test_split with the same (numeric) features and performance is evaluated with RMSE calculated from predictions made with test data. RMSE was calculated using sklearnmetrics.mean_squared_error. Hyperparameters are left on default settings.



	model	rmse
0	linear regression	1.141789
1	random forest	0.909145
2	xgboost	0.597711
3	lightgbm	0.644301

Figure 8: **Table of RMSE score of baseline models.**

Baseline results. Linear Regression and Random Forest Regressor performed about the same (average RMSE < 1.5). LightGBMRegressor and XGBoostRegressor also performed about the same (average RMSE < 0.7). LightGBM and XGBoost Regressors performed significantly better than linear regression and RandomForest and are therefore selected to be further optimized as our final models.

3 Model

3.1 Feature Engineering

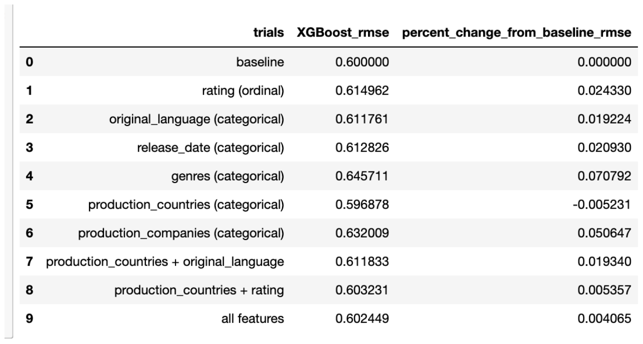
We applied log transformation to each of the numeric features as part of the data cleaning and EDA process of this dataset, and used these numeric features in our baseline models. To further improve the performance of each model, we encoded every non-numeric feature and added them to our baseline model one at a time, comparing the RMSE between the baseline and the baseline with a single feature added, to determine which features are improving our model.

To encode the additional features to input in the models, we used the sklearn library ColumnTransformer() function. We chose to use OrdinalEncoder() to encode the feature ‘rating’ because it is an ordered scale of numbers between one and five. Furthermore, we used ‘OneHotEncoder’ to encode our categorical features "original_language", "release_date", "production_companies", "production_countries", and "genres".

To optimize our regressors, we add each feature individually and compare the performance of the model to the baseline features. Based on these results, we then try different combinations of different features that improve performance individually.

XGBoost Regressor
Feature one-hot encoding production_countries slightly improves performance of the model (percent_change RMSE = -0.5231%). No other features or combination of features improve performance of the model. Therefore, ‘production_countries’ is the only

categorical feature added to this model. Results are shown in the table below.

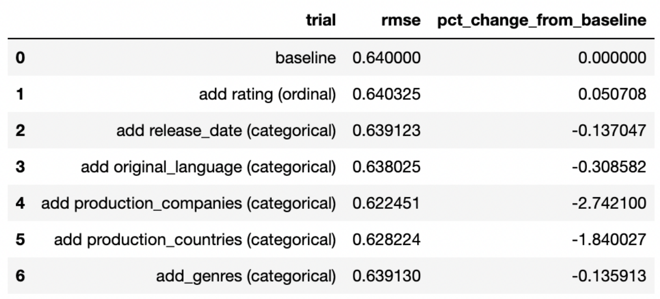


	trials	XGBoost_rmse	percent_change_from_baseline_rmse
0	baseline	0.600000	0.000000
1	rating (ordinal)	0.614962	0.024330
2	original_language (categorical)	0.611761	0.019224
3	release_date (categorical)	0.612826	0.020930
4	genres (categorical)	0.645711	0.070792
5	production_countries (categorical)	0.596878	-0.005231
6	production_companies (categorical)	0.632009	0.050647
7	production_countries + original_language	0.611833	0.019340
8	production_countries + rating	0.603231	0.005357
9	all features	0.602449	0.004065

Figure 9: **Comparing RMSE score of XGBoost baseline model with a single added features and feature combinations.**

LightGBM Regressor

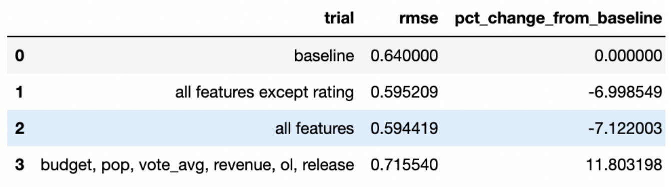
We can see from the table below that each additional feature added, barring 'ratings', improved the RMSE of the baseline model, indicating that they are all features that should be included in the final model.



	trial	rmse	pct_change_from_baseline
0	baseline	0.640000	0.000000
1	add rating (ordinal)	0.640325	0.050708
2	add release_date (categorical)	0.639123	-0.137047
3	add original_language (categorical)	0.638025	-0.308582
4	add production_companies (categorical)	0.622451	-2.742100
5	add production_countries (categorical)	0.628224	-1.840027
6	add_genres (categorical)	0.639130	-0.135913

Figure 10: **Comparing RMSE score of baseline model and baseline model with a single added feature.**

To further examine how each feature plays a role in improving predictions of revenue for a given movie, we input different combinations of features into the LightGBM regressor and calculate the RMSE. Some of these combinations are shown below:



	trial	rmse	pct_change_from_baseline
0	baseline	0.640000	0.000000
1	all features except rating	0.595209	-6.998549
2	all features	0.594419	-7.122003
3	budget, pop, vote_avg, revenue, ol, release	0.715540	11.803198

Figure 11: **Comparing RMSE score of baseline model and models with different combinations of features.**

We found that the LightGBM regression model with the best RMSE included every single numeric, ordinal, and categorical feature present in our cleaned dataset.

4 Literature

Many researchers have attempted to predict movie revenue, we will discuss a few which inspired our project.

4.1 **IMDb Box Office Prediction Using Machine Learning Algorithms** by Mohini Gore, Aishwarya Sheth, Samrudhi Abbad, Paryul Jain, Prof. Pooja Mishra. https://www.ijraset.com/research-paper/imdb-box-office-prediction-using-ml-algorithms

In this study, Gore and collaborators attempt to predict movie revenue from related factors. The authors used a public Kaggle dataset containing movie metadata, much like the dataset used in our analysis. The columns they used included information about budget, vote average, vote count, runtime, genres, spoken language, production companies, release date and cast, which are all the same attributes we used aside from cast. For their models they used linear regression, polynomial regression, SVR, as well as K-means clustering. They also used RMSE as their model evaluation metric, and found that K-means clustering performed the best for predicting revenue. Gore and colleagues' report showed us what attributes of movie metadata can be useful for predicting revenue and how our produced models can be evaluated. Our project uses somewhat more advanced models, and therefore may produce more accurate predictions.

Movie Reviews and Revenues: An Experiment in Text Regression by Mahesh Joshi, Dipanjan Das, Kevin Gimpel, and Noah A. Smith.
https://aclanthology.org/N10-1038.pdf

Joshi and colleagues take a new approach to movie revenue prediction by combining critics' reviews with metadata. They obtained metadata of many movies by scraping sites such as metacritic.com and the-numbers.com, and movie review data from various news sites. They make use of a linear regression model and compare the results of their model before and after adding movie review features. The movie metadata columns they used contained many attributes similar to features used in our project, such as country of origin and budget. The evaluation metrics used in this predictive task are MAE, mean absolute error and Pearsons' correlation coefficient. Through their analysis, Joshi and colleagues found that combining movie review data along with movie metadata greatly improves revenue predictions. Joshi and colleagues' report showed us an innovative approach to predicting revenue. As this report is from 2010, the models used are not as advanced as the models we used in our project, but seeing that adding review data improved their results so greatly, we may be able to produce even more accurate results by combining our advanced models with their review feature implementation.

5 Results

5.1 Comparison

Both our XGBoost Regressor and LightGBMRegressor outperformed our baseline models. After testing individual feature addition and combination feature addition we have improved our RMSEs by about 16% for the LightGBMRegressor and 2% for the XGBoostRegressor. The LightGBM model improved substantially more than the XGBoostRegressor, but our final model of the XGBoostRegressor still outperforms all of our baseline models. After much trial and error, we found that in general, the most useful numeric features for predicting are budget, vote count, and runtime, and the most beneficial categorical features are rating, production_countries, and genres. Carefully observing our models' performance with different feature variables helped us hand-pick features that optimize our predictions and lead to improved results.

5.2 Effectiveness of features

For our XGBoostRegressor, only some of our features improved performance, and for our LightGBMRegressor all of our features improved performance. XGBoostRegressor seems to be more sensitive in using categorical features, while LightGBMRegressor greatly improved from adding categorical features. Our baseline model of the XGBoostRegressor (with numeric features only) performed fairly well already, and seemed to only slightly change with the addition of any features.

5.3 Hyperparameters

Our optimal hyperparameters were obtained by testing multiple sets of possible hyperparameters and narrowing down on the values with RandomizedGridSearchCV. We then manually tuned some of the parameters that required closer examination, and kept tuning until we achieved lower and lower RMSEs. Overall hyperparameter tuning lessens overfitting and improves model performance.

5.4 Major takeaways

1. Feature engineering can be made more efficient by engineering similar features in the same way. Skewness can be dealt with by normalizing the data.
2. Numeric variables with strong association with revenue are essential for the model, while categorical variables that exhibit strong association with revenue can either improve/worsen data depending on how sensitive the model is. We may have achieved different results if we included categorical variables in our baseline models, but seeing that numeric features have a stronger association with revenue, we decided that they are the best basis to start with.
3. The features we chose all improved LightGBMRegressor, indicating that they can all be helpful in predicting revenue, however one must be careful with this application as models such as XGBRegressor may be more sensitive to the addition of categorical features.
4. Keeping the data and models as controlled as possible when producing new RMSEs is essential for achieving impactful results. Whether during feature selection or hyperparameter tuning, all other variables should be kept the same to assure accurate results.
5. Movie_metadata and ratings are extremely useful for predicting revenue, and looking at the data in different ways, such as by genre, production country, etc. may improve results. There are many ways we could improve our model by incorporating movie reviews, movie history, cast, and more in our analysis.

6 Notebook

Our code is linked below:

https://github.com/vibhas1/dsc148_wi23_finalproject

REFERENCES

[1] Mohini Gore, Aishwarya Sheth, Samrudhi Abbad, Paryul Jain, and Prof. Pooja Mishra. 2022. Ijraset Journal For Research in Applied Science and Engineering Technology. (May 2022). Retrieved March 19, 2023 from https://www.ijraset.com/research-paper/imdb-box-office-prediction-using-ml-algorithms

[2] Mahesh Joshi, Dipanjan Das, Kevin Gimpel, and Noah A. Smith. 2010. Movie Reviews and Revenues: An Experiment in Text Regression. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 293–296, Los Angeles, California. Association for Computational Linguistics.