Machine Learning Engineer Nanodegree

Capstone Project

TMDB Box Office Prediction

Can you predict a movie's worldwide box office revenue?

Sougata Ghosh

March 21, 2019.

I. Definition

Project Overview

In a world where movies made an estimated $41.7 billion in 2018, the film industry is more popular than ever. It is at the forefront of creating artistic and cultural trends that impact billions of viewers worldwide. Films can bring to us stories that need to be told and help shape public opinion through their unique blend of entertainment and widespread reach. For the industry to continue to thrive and make meaningful contributions to society it is important for it also to be profitable and for films to make money. But what movies make the most money at the box office? How much does a director matter? Or the budget? For some movies, it is "You had me at 'Hello.'" For others, the trailer falls short of expectations and you think "What we have here is a failure to communicate."

In this Udacity Machine Learning Engineer Nanodegree capstone project, which is based on this Kaggle Competition , we are presented with metadata on over 7,000 past films from The Movie Database to try and predict their overall worldwide box office revenue.

We are provided with two csv files (train and test) containing names of 7398 movies and a variety of metadata obtained from The Movie Database (TMDB). Movies are labelled with id. Data points include cast, crew, plot keywords, overview, budget, posters, release dates, languages, production companies and countries, the genres each movie belongs to, its popularity and runtime, etc in both datasets.

The project objective is to predict the worldwide revenues for 4398 movies in the test file. The test data has 22 columns, one less than the train dataset as the revenue information for the test set movies is not given.

This dataset has been collected from TMDB. The movie details, credits and keywords have been collected from the TMDB Open API as part of the Kaggle competition. The train and

test data along with a sample submission file can be obtained here. They are also included in the capstone project github repository.

Problem Statement

Using the data from The Movie Database train and test datasets, our aim is to predict the worldwide revenues for films in the test data set. Data points in both datasets provide a variety of information on each movie. The models trained use the information given above as well as new features created from them. Since the target variable (revenue) is continuous, this is essentially a **regression** task that is addressed with one of several different machine learning algorithms we are going to use.

The strategy for solving the problem involves the following steps:

1. Exploratory Data Analysis of raw datasets including outlier and missing value detection and creation of visualizations to explore relationships of features with target variable.
2. Data preprocessing including handling of outliers and missing data, modification of target variable (revenue) into logarithmic form, feature creation and engineering.
3. Using XGBoost, Light GBM and CatBoost (from Yandex), all tree-based algorithms popular with Kaggle competitions, to fit the data and make predictions.
4. Using k-fold cross-validation and hyperparameter tuning for each algorithm to improve predictions over baseline model.
5. Averaging the predictions of three models at each stage to make final predictions and Kaggle submissions to track public leaderboard scores.

Metrics

Since this is a Kaggle competition we have already been provided with a suitable evaluation metric on which the Kaggle submissions are evaluated. Submissions are evaluated on Root-Mean-Squared-Logarithmic-Error (RMSLE) between the predicted value and the actual revenue. The root mean square error is a common evaluation metric for regression tasks and will therefore be appropriate for the current project. In the present case logs are taken to not overweight blockbuster revenue movies.

II. Analysis

Data Exploration

- The train dataset has information on the actual box office revenues for 3000 movies to train our machine learning algorithms on. The train set has 3000 records and 23 columns including the target variable ‘revenue’.

- The project objective is to predict the worldwide revenues for 4398 movies in the test file. The test data has 22 columns, one less than the train dataset as the revenue information for the test set movies is not given.
- The target variable is skewed and a logarithmic transformation is required to normalize the data. There are some outliers (mainly abnormally small values).
- The predictor features can be grouped as follow:
    - **Numeric** – Budget, Popularity, Runtime. Over 800 hundred records have zero budget indicating information not available and suitable imputation is required. The movie runtime feature also has some zero values which requires to be addressed.
    - **Date** – Movie release date in mm/dd/yy format. It is required to process this into mm/dd/yyyy format to extract important features like release year, release month, etc.
    - **JSON Format columns** – Collections movie belongs to, Genre, Production Companies and Countries, Spoken Languages, Keywords, Cast, Crew. These variables need to be processed into dictionaries and appropriate measures extracted from them as detailed later.
    - **Other** – Homepage, Original Language, Original Title, Overview, Tagline, Status, Title.

Data Sample (train)

	id	belongs_to_collection	budget	genres	homepage	imdb_id	original_language	original_title	overview	popularity	...	release_date
0	1	[{'id': 313576, 'name': 'Hot Tub Time Machine ...	14000000	[{'id': 35, 'name': 'Comedy'}]	NaN	tt2637294	en	Hot Tub Time Machine 2	When Lou, who has become the "father of the In...	6.575393	...	2/20/15
1	2	[{'id': 107674, 'name': 'The Princess Diaries ...	40000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	tt0368933	en	The Princess Diaries 2: Royal Engagement	Mia Thermopolis is now a college graduate and ...	8.248895	...	8/6/04
2	3	NaN	3300000	[{'id': 18, 'name': 'Drama'}]	http://sonyclassics.com/whiplash/	tt2582802	en	Whiplash	Under the direction of a ruthless instructor, ...	64.299990	...	10/10/14
3	4	NaN	1200000	[{'id': 53, 'name': 'Thriller'}, {'id': 18, 'n...	http://kahaanithefilm.com/	tt1821480	hi	Kahaani	Vidya Bagchi (Vidya Balan) arrives in Kolkata ...	3.174936	...	3/9/12
4	5	NaN	0	[{'id': 28, 'name': 'Action'}, {'id': 53, 'nam...	NaN	tt1380152	ko	마린보이	Marine Boy is the story of a former national s...	1.148070	...	2/5/09

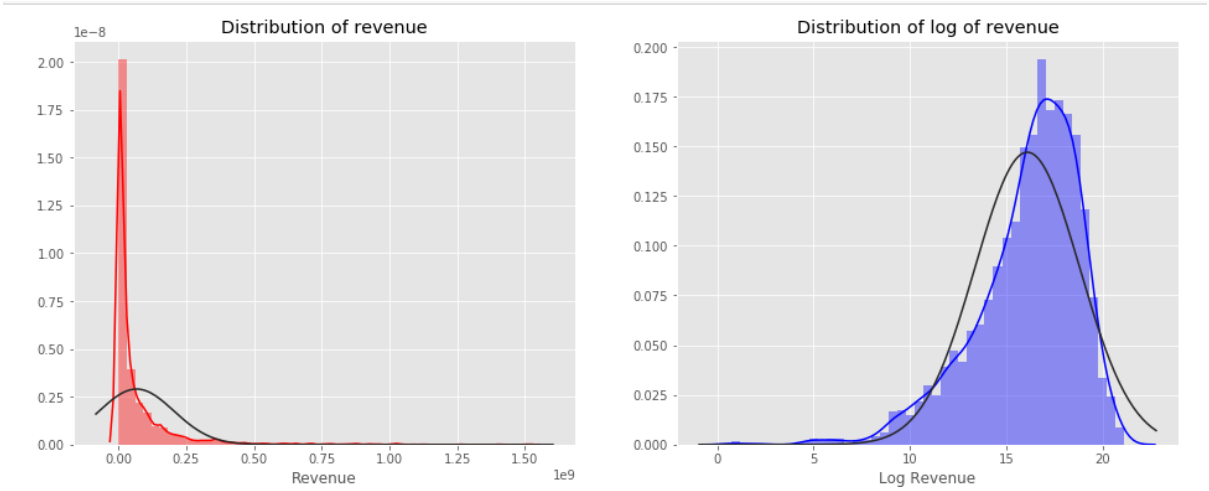
5 rows × 23 columns

Key Statistics (train)

	id	budget	popularity	runtime	revenue
count	3000.000000	3.000000e+03	3000.000000	2998.000000	3.000000e+03
mean	1500.500000	2.253133e+07	8.463274	107.856571	6.672585e+07
std	866.169729	3.702609e+07	12.104000	22.086434	1.375323e+08
min	1.000000	0.000000e+00	0.000001	0.000000	1.000000e+00
25%	750.750000	0.000000e+00	4.018053	94.000000	2.379808e+06
50%	1500.500000	8.000000e+06	7.374861	104.000000	1.680707e+07
75%	2250.250000	2.900000e+07	10.890983	118.000000	6.891920e+07
max	3000.000000	3.800000e+08	294.337037	338.000000	1.519558e+09

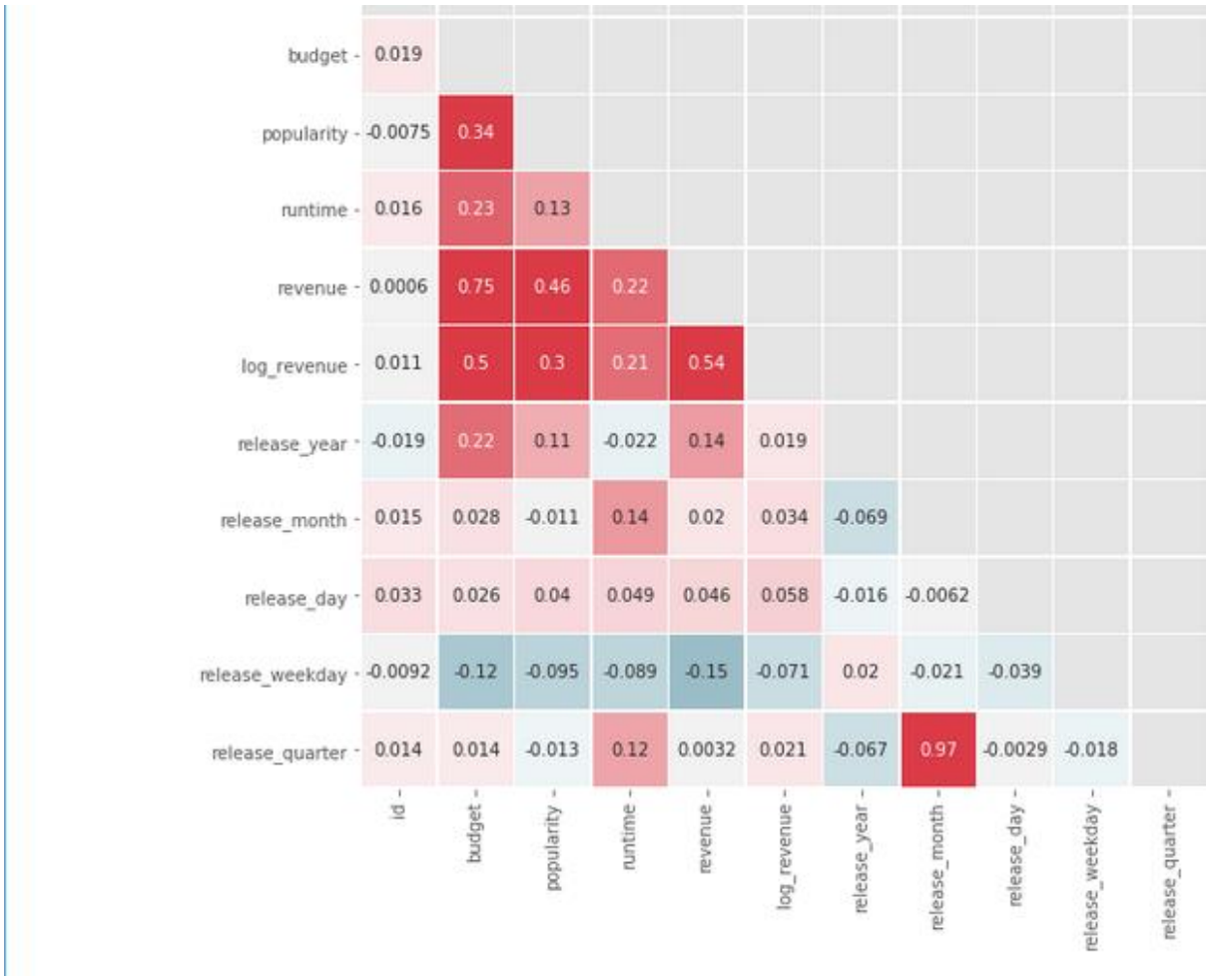
Exploratory Visualizations

1. Target Variable



We can see that the distribution of revenue is quite skewed. The distribution of log revenues is more normalized and will be used as the target variable for modelling.

2. Correlation with Target Variable

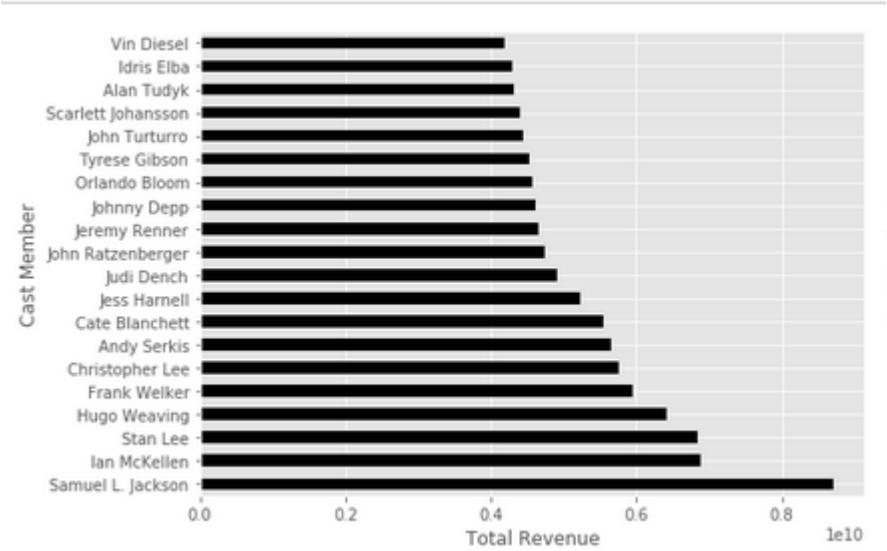


We can see that budget, popularity, runtime and release year are positively correlated with revenue.

3. Popular Movie Genres

Drama, comedy, action and thriller are the popular movie genres.

4. Important Actors



We can see which actors are in films that contribute the highest total revenues.

For a complete list of all visualizations done please refer to the accompanying Jupyter Notebook.

Algorithms and Techniques

The solution proceeds by training several gradient boosting decision tree (GBDT) algorithms, namely XGBoost, LightGBM and CatBoost. Such algorithms are currently the best techniques for building predictive models from structured data. The best decision tree packages can train on large datasets with sublinear time complexity, parallel over many cpus distributed amongst a cluster of computers and can extract most of the information from a dataset under a minute.

Ever since its introduction in 2014, **XGBoost** has been lauded as the holy grail of machine learning competitions. It is usually the first algorithm of choice in any machine learning competition and we will use its powerful features here. XGBoost introduces two techniques to improve performance. First, the idea of Weighted Quantile Sketch, which is an approximation algorithm for determining how to make splits in a decision tree (candidate splits). The second is Sparsity-aware split finding which works on sparse data, data with many missing cells. Apart from this it has other useful features such as L1 and L2 regularization and faster computing through using multiple cores on CPU.

LightGBM from Microsoft is often considered an improvement on XGBoost because of its faster training speed and higher efficiency, lower memory usage, higher accuracy through using a leaf-wise split approach and compatibility with large datasets.

CatBoost is a recently open-sourced machine learning algorithm from Yandex that provides state of the art results and is competitive with any leading machine learning algorithm on the performance front. Its advantage over LightGBM is that it handles categorical features better. We can use CatBoost without any explicit pre-processing to convert categories into numbers. CatBoost converts categorical values into numbers using various statistics on combinations of categorical features and combinations of categorical and numerical features. It reduces the need for extensive hyper-parameter tuning and lower the chances of overfitting also which leads to more generalized models.

Next, we perform **GridSearch –** a simple hyperparameter optimization method where exhaustive searching of manually specified subset of algorithm hyperparameters is performed, guided by cross-validation score of the training set.

The final step in obtaining the optimal solution is to stack these models to create an ensemble of strong yet diverse models. This is especially popular in data science competitions. We intend to use a simple stacking approach that averages the base models.

Benchmark

- Since this is a competition, the Kaggle leaderboard where one can see one's score with respect to other competitors will provide a good benchmark.

- Additionally, for each of the three models above, K-fold cross-validated average score on the training set without additional feature engineering and hyperparameter tuning

as well as the Kaggle submission score after averaging will be the baseline benchmark to judge successive performance on.

Benchmark	XGBoost	Light GBM	CatBoost
Average RMSE	1.7098	1.7427	1.7358

Benchmark Kaggle Score: 2.1135

III. Methodology

Data Preprocessing

1. Outlier, missing value processing

- The abnormally low revenue figures (< 100) were scaled up by multiplying by a constant 100000
- The zero values for budget in train and test data (indicating lack of information) were imputed with median values for the variable in each set.
- Additionally, single digit budget figures (possibly budget in millions of dollars) were scaled up by multiplying by 1000000.
- Zero runtime values were fixed as above with median runtime values in train and test sets.

2. Target Variable

As seen in EDA above the target variable ('revenue') is skewed. We applied a logarithmic transformation to normalize it and used the variable 'log-revenue' as target.

3. Date processing and feature extraction

Since release date for movies was in mm/dd/yy format we changed it to mm//dd/yyyy format by adding '20' before years with values less than 20 and by adding '19' before years with values greater than or equal to 20. Thereafter, features like release year, month, day and weekday were extracted from release date.

4. JSON Column processing and feature extraction

- All JSON format columns as mentioned above were converted into dictionaries.
- Since there are relatively few values in 'belongs_to_collection' we created a simple binary (0,1) column to indicate if a movie belongs to any collection or not.
- Since each movie can belong to more than one genre we extracted an additional variable – num_genres to show the number of genres associated with a movie. Then

we examined the total and average revenues associated with each individual genre. For the top 10 revenue-generating genres we created additional one-hot-encoded features. The JSON column ‘genres’ was then dropped.

- From the variable ‘production_companies’ we similarly created a variable to indicate the number of companies involved in a film and for the top 10 revenue generating companies we created one-hot-encoded features before dropping the column.
- From the variable ‘production_countries’ we created a variable indicating number of countries involved in producing a film. Since US-produced films accounted for the highest revenue by far, we added another variable to indicate whether the production country for a film is US or not.
- From the variable ‘spoken_languages’ we extracted a variable to show the number of spoken languages in a film. Since English is by far the dominant language in the dataset we added another variable to indicate if spoken language is English or not.
- From ‘Keywords’ we extracted the number of keywords attached to a film and whether the top 10 most common keywords are attached to a film or not.
- From ‘cast’ we extracted the number of cast members in a film, whether one of the top 10 revenue generating actors are in a film or not and the number of cast members of different genders in a film.
- From ‘crew’ we extracted the number of crew members in a film, whether one of the top 10 revenue generating crew members are in a film or not and whether one of the top 10 most common crew departments and crew jobs are there in a film or not.

5. Processing and feature extraction of other columns

- From ‘homepage’ we extracted a feature indicating whether a movie has a homepage or not.
- From ‘original language’ we extracted a feature indicating whether a movie’s original language is English or not.
- From ‘original title’ we extracted a column indicating the number of words in the original title.
- From ‘overview’ we extracted the number of words in the film overview.
- From ‘tagline’ we extracted the number of words in the tagline and whether a film has a tagline or not.
- The variable ‘status’ mostly showed ‘Released’ and would not be informative. It was dropped.
- From ‘title’ we extracted a variable indicating word count in each title and whether a film has the same title as original title or not.

All remaining columns with missing values (word counts of title, tagline and overview) were imputed with zero for missing values.

Implementation

The implementation proceeded in three stages.

Stage 1

The XGBoost model was trained with the xgboost train method and K-fold cross-validation with the following parameters: {'learning_rate': 0.01, 'objective': 'reg:linear', 'max_depth': 5, 'gamma': 0, 'min_child_weight': 3, 'subsample': 0.8, 'colsample_bytree': 0.8, 'eval_metric': 'rmse', 'seed': 11, 'silent': True}.

The training set (after the data preprocessing detailed above) was split into training and validation sets for each of 5 folds, converted to xgboost DMatrix format and trained. The average Root Mean Squared Error for validation set across folds was reported. Finally, a prediction on the entire test set was made with the trained model.

The Light GBM model was trained using the sklearn LGBMRegressor wrapper with the following parameters : {'num_leaves': 30, 'min_data_in_leaf': 20, 'objective': 'regression', 'max_depth': 5, 'learning_rate': 0.01,"boosting": "gbdt","bagging_freq": 1, "bagging_fraction": 0.9, "bagging_seed": 11,"metric": 'rmse'}.

The training set (after the data preprocessing detailed above) was split into training and validation sets for each of 5 folds and trained. The average Root Mean Squared Error for validation set across folds was reported. Finally, a prediction on the entire test set was made with the trained model. Additionally, the feature importance figures were extracted.

The Cat Boost model was trained using the sklearn CatBoostRegressor wrapper with the following parameters : {'learning_rate': 0.02, 'depth': 5, 'l2_leaf_reg': 10, 'bootstrap_type': 'Bernoulli', 'od_type': 'Iter', 'od_wait': 50, 'random_seed': 11, 'allow_writing_files': False}.

An empty list was given for the cat_features parameter to indicate that all features are numeric.

The training set (after the data preprocessing detailed above) was split into training and validation sets for each of 5 folds and trained. The average Root Mean Squared Error for validation set across folds was reported. Finally, a prediction on the entire test set was made with the trained model.

Finally, a simple average of predictions from all three models was used for Kaggle submission.

Stage 2

In this stage additional features incorporating interactions among the most important predictors as found in Stage 1 were added. Details of these features are provided in the Refinement section. Thereafter each model was trained as before with the same parameters as before and k-fold cross validation. Average validation scores across folds was reported for each model and prediction on the full test set performed. The average of predictions across models was used for Kaggle submission.

Stage 3

In this stage hyperparameter tuning for each model was performed. Details of these tunings are provided in the Refinement section. Thereafter each model was trained as before with the tuned parameters and k-fold cross validation. Average validation scores across folds was reported for each model and prediction on the full test set performed. The average of predictions across models was used for Kaggle submission.

Refinement

Additional Features

After Stage 1 as detailed in Implementation section above we extracted the important features and found the most important to be (across models): budget, popularity, release year and runtime. In order to capture interactions among these features better we added the following derived features to the data:

1. Budget-runtime-ratio = budget/runtime
2. Budget-popularity-ratio = budget/popularity
3. Budget-year-ratio = budget / (release_yearXrelease_year)
4. Release_year-popularity ratio = release_year / popularity
5. Release_year-popularity ratio_2 = popularity/release_year
6. Year-to-log_budget = release_year / log(budget)
7. Year-to-log-popularity = release_year / log(popularity)
8. Runtime-to-mean-year = runtime / mean of runtime grouped by release year
9. Popularity-to-mean-year = Popularity/ mean of popularity grouped by release year
10. Budget-to-mean-year = Budget/mean of popularity grouped by release year

HyperParameter Tuning

In Stage 3 of implementation we tuned the most important hyperparameters for each model.

The choice of parameters for the XGBoost model was informed by this article and was as follows:

max_depth : range(3,10,2)

min_child_weight : range(1,6,2)

The choice of parameters for the LightGBM model was informed by this article and was as follows:

max_depth : [4,5,6]

num_leaves : [15,30,40,50]

The choice of parameters for the CatBoost model was informed by this article and was as follows:

depth : [4,5,6,7]

l2_leaf_reg : [5,10]

IV. Results

Model Evaluation and Validation

The process of arriving at the models at each stage has been outlined above. We give below the cross-validated score at each stage for each model.

Stage 1 results:

	XGBoost	Light GBM	CatBoost
CV Mean RMSE	1.7098	1.7427	1.7358
CV RMSE std dev	0.1053	0.1073	0.1252

Stage 2 results:

	XGBoost	Light GBM	CatBoost
CV Mean RMSE	1.7204	1.7534	1.7348
CV RMSE std dev	0.1113	0.1120	0.1251

Stage 3 results:

	XGBoost	Light GBM	CatBoost
CV Mean RMSE	1.7175	1.7566	1.7264
CV RMSE std dev	0.1065	0.1107	0.1232

The final model chosen has been the averaged predictions of Stage 3 models. Each of the Stage 3 models have low standard deviation indicating that they generalize well and are robust to changes in data. The Stage 3 models capture more of the interactions among data and have also been tuned with Grid Search. The final reason for choosing the Stage 3 models is the improvement in Kaggle submission score with blended predictions as shown below which aligns well with solution expectations.

Justification

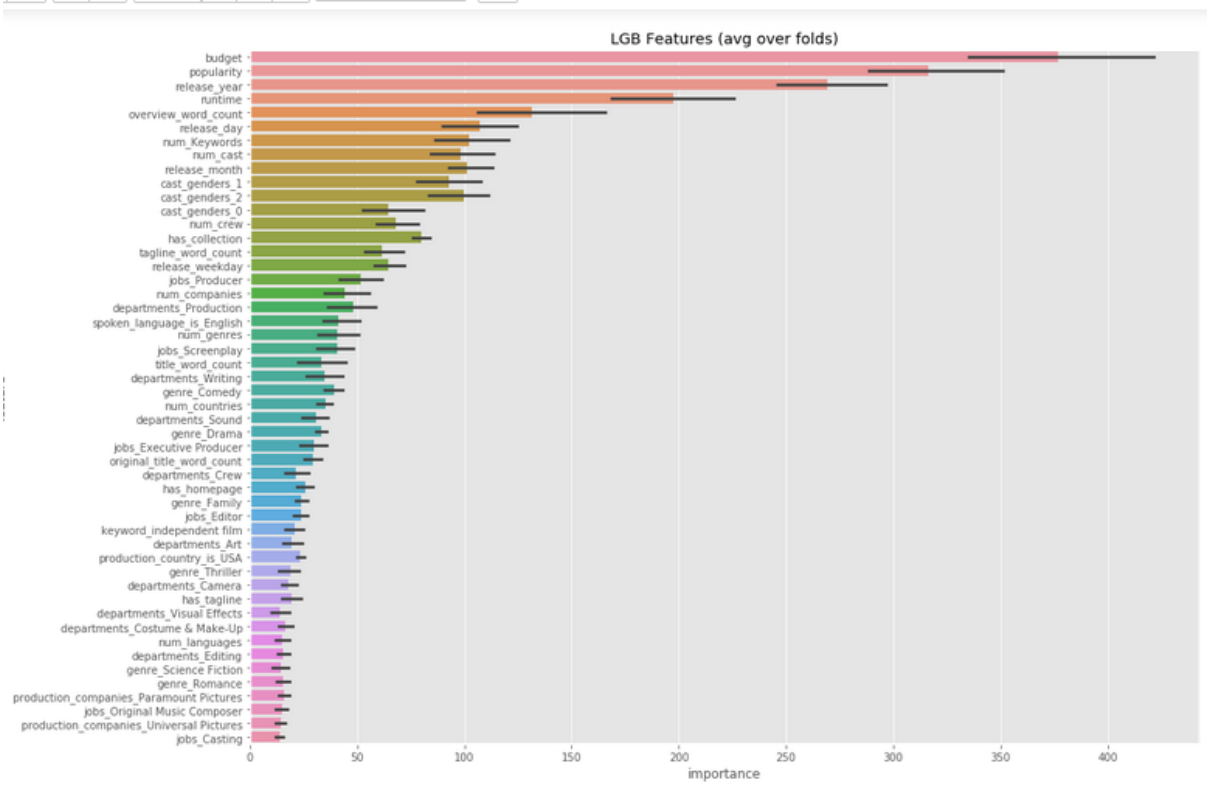
Blended Prediction	Kaggle Submission Score
Stage 1 Models	2.1135
Stage 2 Models	2.09696
Stage 3 Models	2.09397

As we can see the final model score outperforms the benchmark score by close to 1 %. This is expected as the final solution incorporates additional features, parameter tuning. Also stacking the models has had its own benefit. While there are better scores in the competition, this indicates that we have progressed in the right direction in our analysis and with further improvements (some of which are outlined below) there is scope to better our scores further above the benchmark.

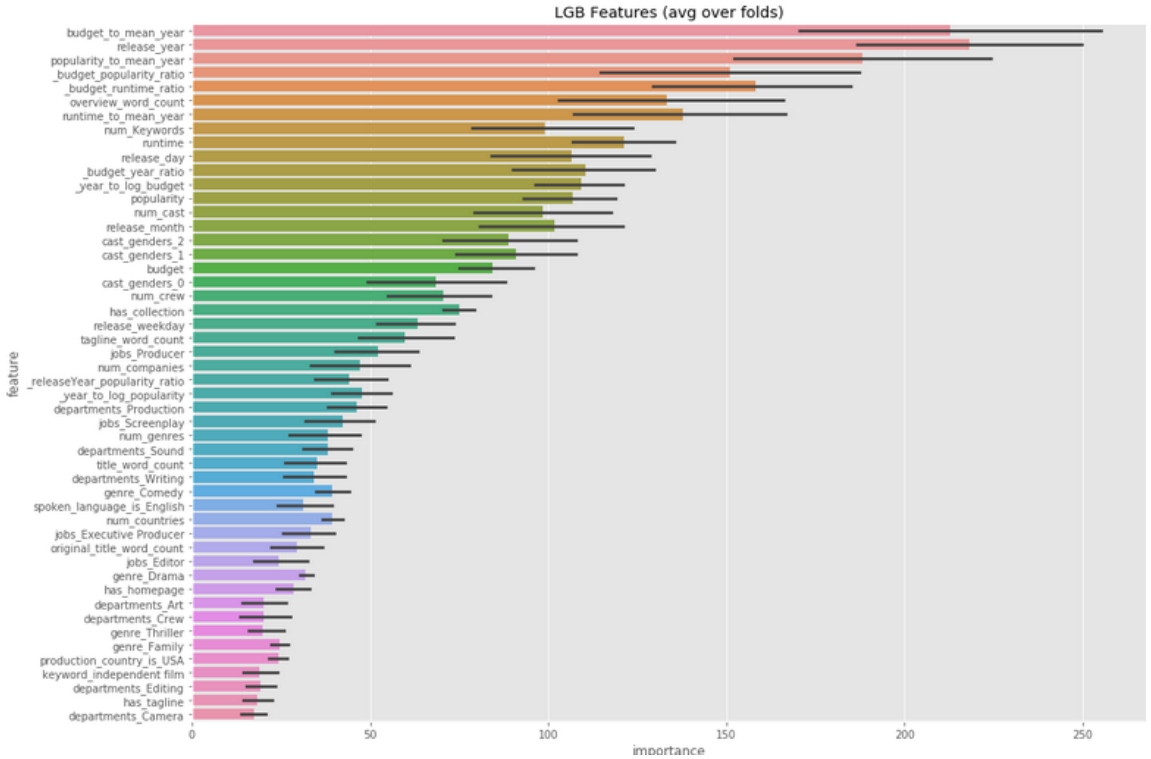
V. Results

Free-Form Visualization

Feature Importance Stage 1 Light GBM



Feature Importance Stage 2 Light GBM



The figures above show the feature importance with the base model and after additional features incorporating the interactions among the top predictors. Many of the derived features in the second model have emerged as more important than the features in the base model. This indicates the benefits of feature engineering in this type of problems.

Reflection

This was challenging project, not least because it was a Kaggle competition where one's position on the leaderboard was a learning experience. The project involved end-to-end use of machine-learning techniques learned in the Nanodegree program: data cleaning and preprocessing, feature extraction and engineering, model selection and hyperparameter tuning. It was really interesting to work with state-of-the-art machine learning algorithms like XGBoost, Light GBM and CatBoost and get to know a lot about the strengths of each. It was also interesting to see the benefits of model stacking in improving one's scores.

I found the data preprocessing a challenging aspect of the project, especially extracting meaningful features from the often messy JSON format columns. Another difficult part was hyperparameter tuning. Although I worked with relatively modest number and range of parameters and even though I ran my Jupyter Notebook on Amazon EC2, the process of tuning each model was quite lengthy.

Given the improvement in Kaggle submission scores at each step, I think proceeding methodically in the way done was in the right direction and the choice of algorithms and techniques used is suitable for solving this type of problems.

Improvement

There are several improvements that can be looked at:

More features – Even though we did quite a bit of feature engineering, there are many other features that can be extracted/derived which can make our predictions better.

More models – The power of stacking models has come out in this project. Using and stacking more models and using more sophisticated stacking techniques can I think also enhance our predictions further.

More parameter tuning- We have experimented with a relatively small subset of the huge parameter set that is available with 3 models being considered. Further tuning of hyperparameters would I feel also positively impact our predictions further.

Thus, even though we have demonstrated improvement over the benchmark performance set at the beginning of the project, further improvement in scores and competition performance is surely possible with some of the improvements suggested above.