Machine Learning Engineer Nano-Degree

Capstone Project

Srinivas C Reddy

August 10, 2019

I. Definition

Project Overview

We live in a movie obsessed world, where movies made an estimated $41.7 billion in 2018, if we include home entertainment revenue global film industry is worth $136 billion. (IBISWorld, 2018).[1] The film industry is more popular than ever. But what movies make the most money at the box office? How much does a director matter? Or the budget? For some movies, it's "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." (Kaggle Competition, 2019)[2]

For the capstone project, I chose to work on Kaggle movie revenue prediction competition problem; since it gave me a good machine learning problem to work on with appropriate open dataset to train along with it. The problem also gave me access to rich community of participants, to review state of the art algorithms, variety of techniques used and diverse approaches as I embark my ML Engineer practitioner journey.

Objective of this project is to predict a movie revenue based on historic data about movie revenues and performance at the global box office. Such a prediction would be useful for optimization in many areas during various stages of planning and production of movies, for instance, selection of actors, crew, location, production spend, marketing spend, logistics and so on. Predicting revenue potential would enable movie production enterprises make wise investment decisions, come up with movies with plots

1 IBIS World 2018 https://www.ibisworld.com/industry-trends/global-industry-reports/other-community-social-personal-service-activities/movie-production-distribution.html

2 Kaggle Competition 2019 https://www.kaggle.com/c/tmdb-box-office-prediction

relevant to society, higher entertainment satisfaction and ultimately greater good of all the involved parties.

In the given dataset, I am provided with 7398 movies and a variety of metadata obtained from The Movie Database (TMDB). Movies are labeled with id. Data points include cast, crew, plot keywords, budget, posters, release dates, languages, production companies, and countries. I am expected to predict the worldwide revenue for 4398 movies in the test file.

Note - many movies are remade over the years, therefore it may seem like multiple instance of a movie may appear in the data, however they are different and should be considered separate movies. In addition, some movies may share a title, but be entirely unrelated.

E.g. The Karate Kid (id: 5266) was released in 1986, while a clearly (or maybe just subjectively) inferior remake (id: 1987) was released in 2010. Also, while the Frozen (id: 5295) released by Disney in 2013 may be the household name, don't forget about the less-popular Frozen (id: 139) released three years earlier about skiers who are stranded on a chairlift.

This dataset has been collected from TMDB. The movie details, credits and keywords have been collected from the TMDB Open API. This competition uses the TMDB API but is not endorsed or certified by TMDB. Their API also provides access to data on many additional movies, actors and actresses, crew members, and TV shows.

Problem Statement

In this public competition hosted by Kaggle, I am presented with metadata on past films from The Movie Database, and the challenge is to predict their overall worldwide box office revenue. Data points provided include cast, crew, plot keywords, budget, posters, release dates, languages, production companies, and countries. I can also collect other publicly available data to use in your model predictions, but in the spirit of this competition, I am allowed to use only the data that would have been available before a movie's release. I plan to use additional features dataset with ratings and number of votes for the movies in training dataset. The goal of this capstone project is to predict the overall worldwide box office revenue given data about movies.

Solution Statement

The solution will be worldwide box office revenue prediction, which is continuous. Because of nature of solution, I will be able to quantify in math or logical terms. Goal of this ML exercise is to minimize the error term on the prediction, meaning the smaller the error term, better the prediction matches with the

real world actual revenues. I plan to perform thorough data Exploration activities, understand data well, perform data profiling, visualize distributions for each attribute, perform feature engineering, encode ordinal and nominal features. I intend to use DecisionTreeRegressor [3]with default values as a simple benchmark model and compare other model performances to the benchmark Model. In particular, I intend to use LightGBM[4], XGBoost[5] and CatBoost[6] regression techniques to perform modeling activities. The reasoning behind my selection of these modeling algorithms is that they are widely used for similar regression problems, perform better than simple tree algorithms, and they fit well within the constraints of supervised learning category. Of course it is important to mention that there are wide range of other algorithmic choices for regression problems, I narrowed down on high yield models based on quick survey of latest trends in ML research.

I will evaluate predictions with the validation and test datasets to calculate error measurement Root-Mean-Squared-Logarithmic-Error (RMSLE), logs are taken to not overweight blockbuster movies. Goal of this ML exercise is to minimize the error term on the prediction.

Metrics

The main evaluation metric used will be Root-Mean-Squared-Logarithmic-Error (RMSLE), which is given by[7] Figure 1:

$$\begin{aligned}\text{RMSLE} &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\log(y_i+1)-\log(\hat{y}_i+1)\right)^2} = \\ &= RMSE\left(\log(y_i+1),\log(\hat{y}_i+1)\right) = \\ &= \sqrt{MSE\left(\log(y_i+1),\log(\hat{y}_i+1)\right)}\end{aligned}$$

Figure 1

3 DesisionTreeRegressor - https://scikit-learn.org/stable/modules/tree.html#regression

4 LightGBM - https://lightgbm.readthedocs.io/en/latest/

5 XGBoost - https://xgboost.ai/

6 CatBoost - https://catboost.ai/

7 RMSLE - How to select right evaluation metric for Machine Learning Models – Regression https://towardsdatascience.com/how-to-select-the-right-evaluation-metric-for-machine-learning-models-part-2-regression-metrics-d4a1a9ba3d74

RMSLE is Root Mean Square Error calculated in logarithmic scale. The targets are usually non-negative, but if target is equal to 0, logarithm of 0 is not defined, that is the reason for adding constant 1 before taking logarithm. I will perform hyperparameter tuning based on how model performs in minimizing RMSLE. The RMSLE serves to aggregate the magnitudes of the errors in predictions for various times into a single measure of predictive power. RMSLE is a measure of accuracy to compare forecasting errors of different models for a particular dataset and not between datasets, as it is scale-dependent.[8]

The reason I use RMSLE for this problem is that a blockbuster movie revenue could skew prediction accuracy. Using logarithmic scale smooths out the effect of blockbuster movies. Optimization objective is here to minimize the RMSLE, which are effectively error residuals. The closer the error to zero, the better the prediction accuracy.

II. Analysis

Data Exploration

The competition problem comes with training dataset, initial look at data indicates it is in decent form; however, has embedded JSON text columns, missing values, dates. There are 4 numerical columns excluding id, in particular budget and revenue can have large range of values for blockbuster movies, which would necessitate use of appropriate techniques to transform.

Dataset Sampling:

Training data :

A sample of first 5 rows of data is as indicated in Figure 2

8 RMSD/RMSLE details - https://en.wikipedia.org/wiki/Root-mean-square_deviation

	id	belongs_to_collection	budget	genres	homepage	imdb_id	original_language	original_title	overview	popularity	...	release_date	runtime	spoken_languages	status	tagline	title	Keywords	cast	crew	revenue
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	93.0	[{'iso_639_1': 'en', 'name': 'English'}]	Released	The Laws of Space and Time are About to be Vio...	Hot Tub Time Machine 2	[{'id': 4379, 'name': 'time travel'}, {'id': 9...	[{'cast_id': 4, 'character': 'Lou', 'credit_id...	[{'credit_id': '59ac067c92514107af02c8c8', 'de...	12314651
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	113.0	[{'iso_639_1': 'en', 'name': 'English'}]	Released	It can take a lifetime to find true love; she'...	The Princess Diaries 2: Royal Engagement	[{'id': 2505, 'name': 'coronation'}, {'id': 42...	[{'cast_id': 1, 'character': 'Mia Thermopolis'...	[{'credit_id': '52fe43fe9251416c7502563d', 'de...	95149435
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	105.0	[{'iso_639_1': 'en', 'name': 'English'}]	Released	The road to greatness can take you to the edge.	Whiplash	[{'id': 1416, 'name': 'jazz'}, {'id': 1523, 'n...	[{'cast_id': 5, 'character': 'Andrew Neimann',...	[{'credit_id': '54d5356ec3a3683ba0000039', 'de...	13092000
3	4	NaN	1200000	[{'id': 53, 'name': 'Thriller'}, {'id': 18, 'n...	http://kahaanifilm.com/	tt1821480	hi	Kahaani	Vidya Bagchi (Vidya Balan) arrives in Kolkata ...	3.174936	...	3/9/12	122.0	[{'iso_639_1': 'en', 'name': 'English'}, {'iso...	Released	NaN	Kahaani	[{'id': 10092, 'name': 'mystery'}, {'id': 1054...	[{'cast_id': 1, 'character': 'Vidya Bagchi', '...	[{'credit_id': '52fe48779251416c9108d6eb', 'de...	16000000
4	5	NaN	0	[{'id': 28, 'name': 'Action'}, {'id': 53, 'nam...	NaN	tt1380152	ko	마린보이	Marine Boy is the story of a former nationals...	1.148070	...	2/5/09	118.0	[{'iso_639_1': 'ko', 'name': '한국어/조선말'}]	Released	NaN	Marine Boy	NaN	[{'cast_id': 3, 'character': 'Chun-soo', 'cred...	[{'credit_id': '52fe464b9251416c75073b43', 'de...	39239700

Figure 2

Additional Features: A sample of 5 rows from additional feature dataset is as indicated in Figure 3

	imdb_id	popularity2	rating	totalVotes
0	tt0169547	16.217	8.0	6016.0
1	tt0119116	26.326	7.4	5862.0
2	tt0325980	28.244	7.7	11546.0
3	tt0266697	18.202	7.9	8638.0
4	tt0418763	9.653	6.6	1201.0

Figure 3

Missing value analysis:

A quick review of missing value counts indicates 80% of movies don't belong to collection, 68% of them don't have a homepage, 20% of them don't have a tagline. Rest of the missing values are smaller portions of less than 10%, giving an early indication of types of cleansing/transformations that could be useful.

I decided to use pandas.profile_report package, as it gives variety of general purpose descriptive statistics in a neat report format, as indicated in Figure 4 through 6.

Dataset info: Figure 4

Number of variables	23
Number of observations	3000
Missing cells	5601 (8.1%)
Duplicate rows	0 (0.0%)
Total size in memory	539.1 KiB

Variables types: Figure 5

Numeric	5
Categorical	17
Boolean	0
Date	0
URL	0
Text (Unique)	1
Rejected	0
Unsupported	0

Warnings: Figure 6

belongs_to_collection has a high cardinality: 423 distinct values	Warning
belongs_to_collection has 2396 (79.9%) missing values	Missing
budget has 812 (27.1%) zeros	Zeros
cast has a high cardinality: 2976 distinct values	Warning
crew has a high cardinality: 2985 distinct values	Warning
genres has a high cardinality: 873 distinct values	Warning
homepage has a high cardinality: 942 distinct values	Warning
homepage has 2054 (68.5%) missing values	Missing
Keywords has a high cardinality: 2649 distinct values	Warning
Keywords has 276 (9.2%) missing values	Missing
original_title has a high cardinality: 2975 distinct values	Warning
overview has a high cardinality: 2993 distinct values	Warning
poster_path has a high cardinality: 3000 distinct values	Warning
production_companies has a high cardinality: 2384 distinct values	Warning
production_companies has 156 (5.2%) missing values	Missing
production_countries has a high cardinality: 322 distinct values	Warning

production_countries has 55 (< 0.1%) missing values	Missing
release_date has a high cardinality: 2398 distinct values	Warning
spoken_languages has a high cardinality: 402 distinct values	Warning
tagline has a high cardinality: 2401 distinct values	Warning
tagline has 597 (19.9%) missing values	Missing
title has a high cardinality: 2969 distinct values	Warning

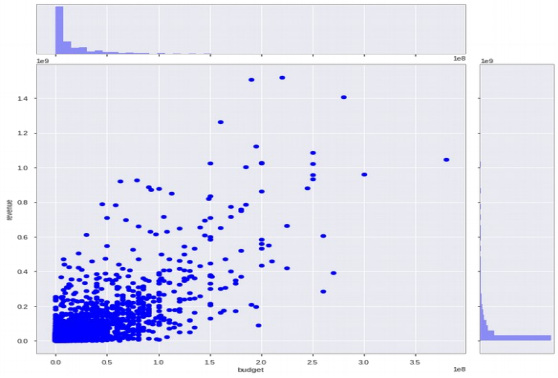
Looking at cardinality warnings with distinct values and missing counts of missing values from profile_report, my task of data exploration pretty much becomes streamlined, and I can draw strategies on how to handle data elements in my training dataset.

Exploratory Visualization

During this phase of the project, I focused on visualizing attributes from dataset, making sense out of their relationships, correlations, and any transformations to make them more useful as I built a map of their significance to the problem at hand.

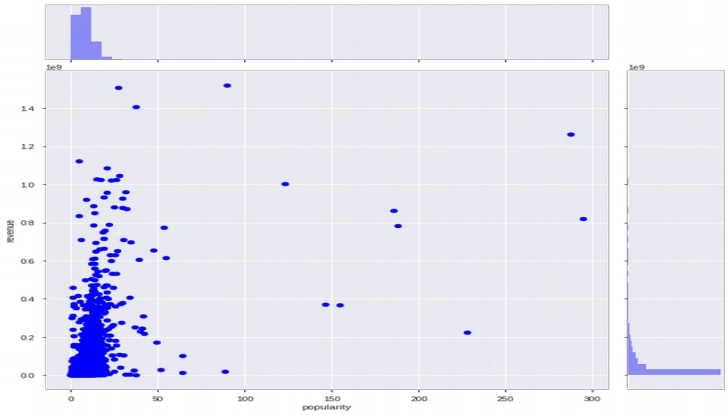
1. Exploring relationship between **Budget and Revenue** scatter plot gives general correlation that budget of movie has influence on revenues, and most of the data is clustered in lower quadrant, blockbuster hits and huge budget movies are relatively infrequent as indicated by following graph in figure 7

Figure 7



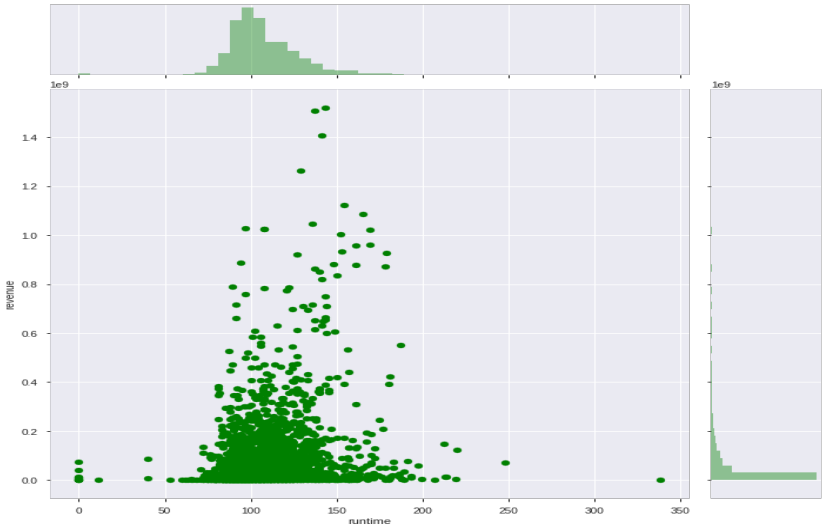
2. Exploring relationship between **Popularity and Revenue** gives similar that highly popular movies earn higher revenues as indicated by following graph in Figure 8.

Figure 8



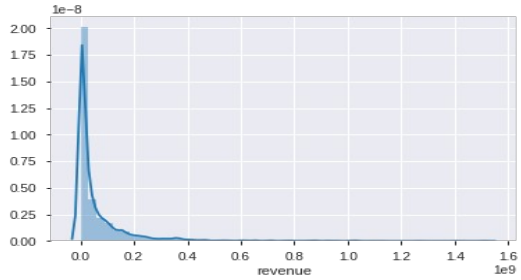
3. Exploring relationship between **Runtime and Revenue** indicates that pretty much normal distribution of population around 100 minutes per following graph in Figure 9.

Figure 9



4. **Revenue distribution analysis** indicates the data is skewed, with large number of movies closer to left, with a long tail of very small number of movies with large revenues, typical of very few blockbuster movies as indicated in Figure 10.

Figure 10



5. **Release date** needs data cleansing and manipulation as it has only last two digits of the year, manipulated to add correct first two digits based on current year we are in (2019). After substitution, explored with plots of Release Year counts, Month counts, Day counts, Day of the week counts, Quarter counts to get a feel of seasonality and overall trends of releases over years.

Note: For brevity of report, rest of exploratory visualization graphs are not included in project report, they can be referenced in notebook associated with the project

6. Exploring relationship of **Release Year vs Revenue** shows general uptick trend as movies became popular with a spike in 1970s and a noticeable dip in 1984 and 1992, it is interesting to mention that this relationship follows overall economic indicators, great inflation of 1970s and recessions in 80s and 90s.
7. Exploring relationship between **release month vs Revenue** clearly indicate summer peak, a fall in fall season and uptick during Holiday season. Exploring revenue by the day of week released indicates a peak on Tuesday releases, interesting contrast as most movies are release on Thursdays.
8. Exploring **mean runtime by the year** seem to indicate it oscillates around little less than 2 hours, though longer movie lengths were common in 1960s averaging around 140 minutes in 60s. For the current project, it is insignificant.
9. Exploring **mean popularity by year** indicate general upward trend in movie popularity, occasional dips and peaks over years potentially tied to something outside like economy, culture and other forms of entertainment. However, it is encouraging to movie industry seeing overall upward trend.
10. Exploring **mean budget by year**, it indicates exponential growth over years; however, we should be cautious to use this as feature, would have to compensate for inflation over years using appropriate methods.
11. After exploring numerical values, focused on categorical features. **Genre of movies** is significant, as invariably Genre is primary criterion used by both industry as well as movie goers as common proxy to narrow down of movies of interest to make or watch. With due respect that a given movie can be classified to multiple Genres, it is interesting to see “Drama” rank first, as movie goers love to see Drama! It is followed by “Comedy”, “Thriller”, “Action” and “Romance”. Rest of the genres are under 1/6th of total population in training set.

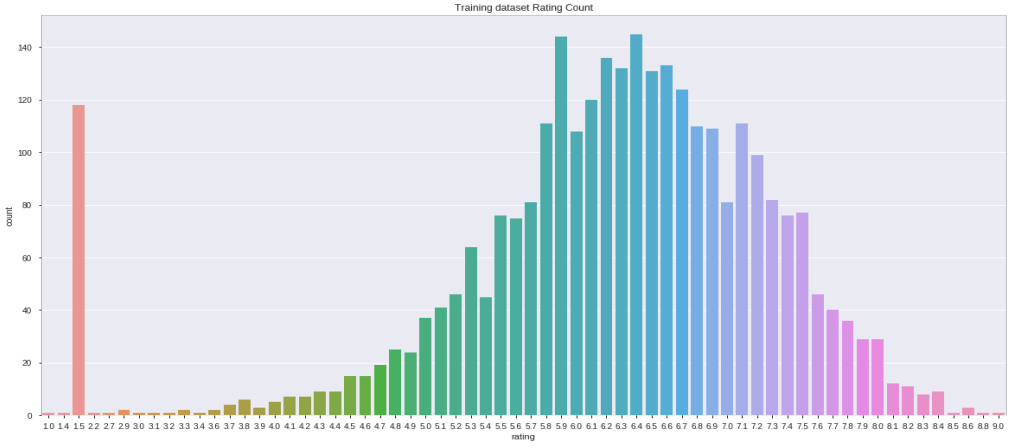
12. Exploring **original language counts**, the training dataset seems totally biased on English, followed by insignificant count of French and other languages. This kind of makes sense due to other considerations outside of scope of the problem, overall English movies makeup bulk of contribution to revenues compared to other languages.
13. Exploring **status**, it is interesting to see there were 4 movies in training dataset and 7 in test dataset which were not released, but they did have some revenue with them. During feature engineering time, need to account for effect of these outliers, though effect should be insignificant as shown by numbers.
14. Exploring **Homepage existence**, roughly 2/3rds of movies in training set don't have a home page, a indicator for homepage existence would be useful? Did a correlation analysis to explore relationship, which does indicate the movies with homepages had significantly higher revenues.
15. Exploring **tagline impact on revenue**, shows similar results, movies with tagline had significantly higher revenues. Similarly, exploring "Original Language" on Revenue showed significant correlation, which could be attributed to dataset bias and in general reality of English as dominant language for movies.

Additional Dataset

I did explore an additional dataset with **rating and votes** for movies, to augment features to be used.

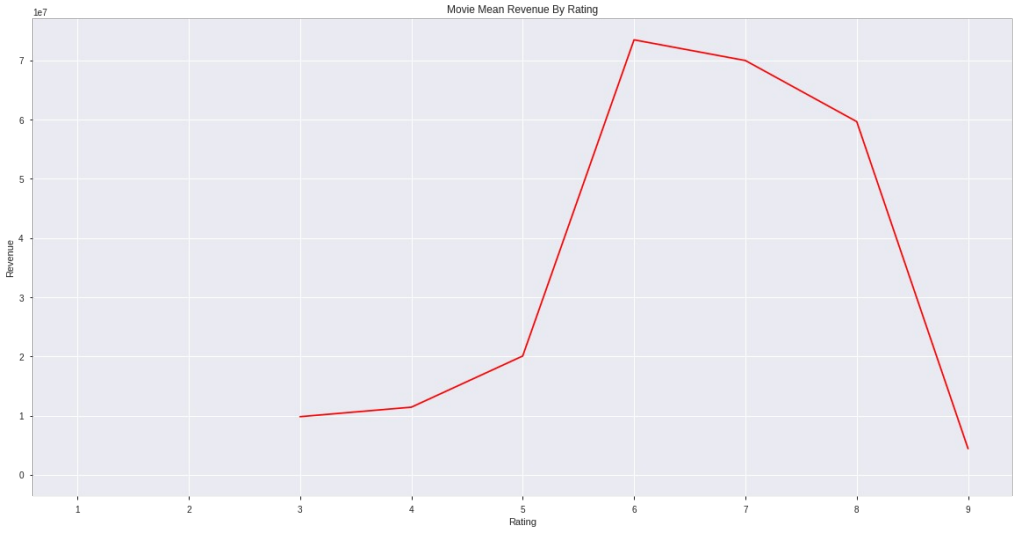
1. First thing I would have to consider when using additional dataset is about if meaningful joins can be established, and missing value analysis. Around 118 observations in additional training dataset have missing values, which could be handled using an appropriate impute strategy. For "**Rating**", decided to use a default of 1.5, which is clearly highlighted in following visualization as anomaly. Looking at rest of the graph in Figure 11, we see roughly Gaussian distribution.

Figure 11



2. Similarly, exploring **Mean Revenue by Rating**, it appears that movies in the range of 5 to 8 amount for significant revenues as indicated by the following graph in Figure 12

Figure 12



3. Exploring **mean revenue by total votes** indicate couple of peaks in mid 1500s and mean total votes by release year roughly indicate correlation potentially to something outside of the problem domain like economy.

4. Exploring relationship between **Mean Total Votes and Rating** indicates the obvious, movies having ratings 6 to 8 have votes in excess of 700, with a peak of 8 rating above 750 votes and sharply dropping beyond rating of 8.

After exploring individual features and relationships of couple at a time, I visualized using heatmap (as depicted in Figure 13) among following features.

['budget','rating','totalVotes','popularity','runtime','release_year','release_month','release_dayofweek','revenue']

Figure 13

As expected, total votes and revenue show very high correlation at 0.77, as highly voted movies perform well at box office, followed by budget and revenue at 0.75, which makes sense as well as highly popular movies which collect large revenues typically have huge budgets. Popularity takes up next spot at 0.46 correlation to revenue, runtime at 0.22. It is surprising to see rating is only at 0.17. Release year and release day of week and release month show very weak to very weak negative correlation.

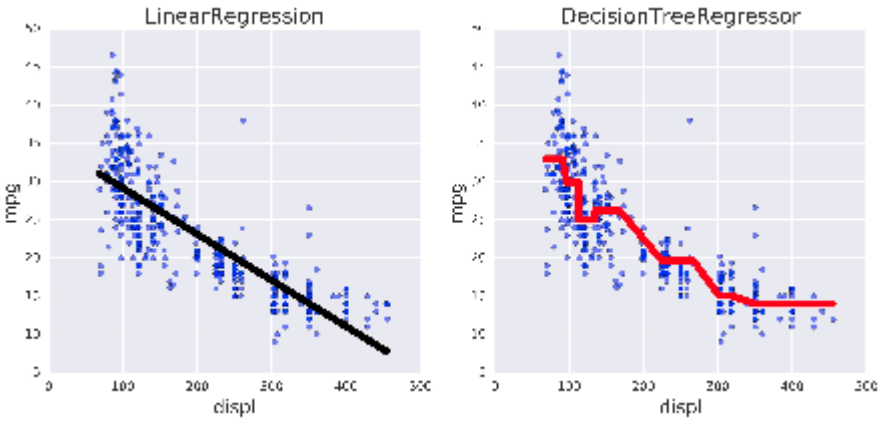
Algorithms and Techniques

The solution will be worldwide box office revenue prediction, which is continuous target prediction problem (regression problem). Because of nature of solution, I will be able to quantify in math or logical terms. As I intend to use tree based algorithms, here is a brief overview of such algorithms, followed by more detailed description of algorithms I chose to use.

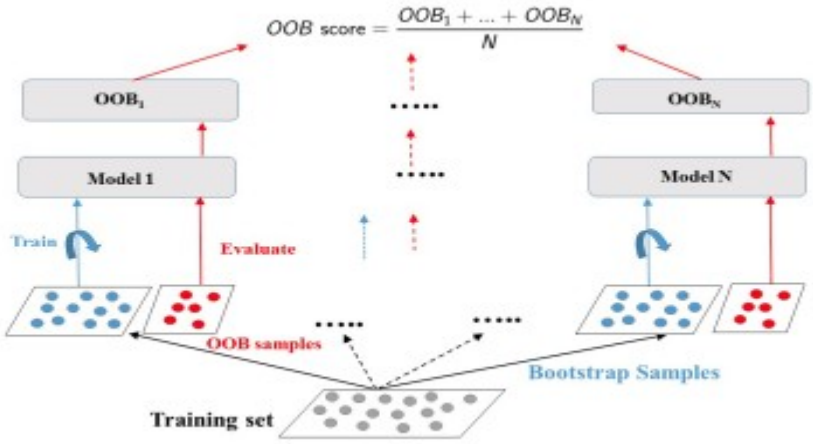
Decision trees are supervised learning models used for problems involving classification and regression. Tree models present a high flexibility that comes at a price: on one hand, trees are able to capture complex non-linear relationships; on the other hand, they are prone to memorizing the noise present in a dataset. By aggregating the predictions of trees that are trained differently, ensemble methods take advantage of the flexibility of trees while reducing their tendency to memorize noise[9].

9 Tree based models in Python - https://www.datacamp.com/community/blog/course-machine-learning-with-tree-based-models-in-python

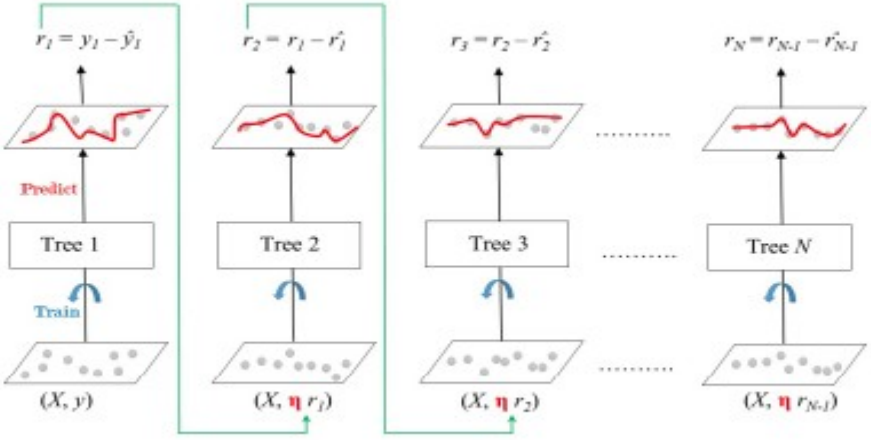
Visual for Linear vs Tree based regression – As we can see from visualization, Tree based algorithms fit better to data, but are more complex, overfit.



Visual for bagging and Random Forests – Bagging is ensemble method involving using same algorithm many times sing different subsets sampled from training data.



Visual for Boosting – Boostingrefers to ensemble method in which several models are trained sequentially with each model learning from the errors of its predecessors. In particular Gradient boosting methods are more popular as equations for optimization are differentiable.



Goal of this ML exercise is to minimize the error term on the prediction. Here are list of techniques[10] and algorithms[11] I intend to try:

1. Data Exploration, feature engineering
   a) Data profiling with pandas-profiling[12] - a fantastic package which profiles tabular data in pandas dataframe, and produces a nice HTML-based report, indicating dataset information, variable types, warnings, distinct values, cardinality etc., along with distribution for numeric variables, as detailed in exploration section.
   b) Visualize distributions for each feature (using pandas-profiling)
   c) Transform and/or normalize, scale features if required
   d) Find outliers and determine if they need to be removed
   e) Encode the ordinal or nominal features using one hot encoding and label encoders.
2. Establish benchmark model metrics (using DecisionTreeRegressor[13]) - Decision trees use a set of binary rules to calculate target value. Each individual tree is fairly simple model that has branches, nodes and leaves. A decision tree arrives at an estimate by asking a series of questions to the data, each question narrowing our possible values until the model gets confident enough to make a single prediction. As a supervised machine learning model, a decision tree learns to map data to outputs in training phase of model building.
3. Supervised model selection using grid-search/k-folds – All 3 of these algorithms fall under gradient boosting decision trees, they are well known to solve Machine Learning problems at scale, especially well suited for regression problem or predicting continuous variable like revenue in our case, they are widely used for Kaggle competitions:

   a) XGBoostRegressor [14]– XGBoost is an optimized distributed gradient boosting library. It implements algorithm under the Gradient Boosting[15] framework. The main strengths of this algorithm is that it provides parallel tree boosting, that can solve data science problems in a fast and accurate way, runs on major distributed environments like Hadoop, MPI and can solve problems beyond billions fo examples. XGBoost provides options to penalize complex modes with

10 Data Visualization using Python for ML and Data Science - https://towardsdatascience.com/data-visualization-for-machine-learning-and-data-science-a45178970be7

11 Choosing the Right Machine Learning Algorithm - https://hackernoon.com/choosing-the-right-machine-learning-algorithm-68126944ce1f

12 A better EDA with Pandas-profiling - https://towardsdatascience.com/a-better-eda-with-pandas-profiling-e842a00e1136

13 Decision Tree Regressor explained in depth - https://gdcoder.com/decision-tree-regressor-explained-in-depth/

14 About XGBoost - https://xgboost.ai/about

15 Gradient Boosting - https://en.wikipedia.org/wiki/Gradient_boosting

L1 and L2 regularization, can handle sparse data very well, can make use of multiple cores, it is cache aware and provides out-of-core computing.

b) CatBoostRegressor[16] - CatBoost falls under similar capabilities compared to XGBoost in that it does gradient boosting on decision trees, it allows ability to run on distributed environments, and provides robust set of hyperparameters for users.

c) LightGBM[17] - LightGBM uses histogram based algorithms, which bicket continuous feature values into discrete bins as compared to many boosting tools which use pre-sort-based algorithms. Binning speeds up training and reduces memory usage, thereby reducing cost of calculating gain for each split, using histogram subtraction for further speedup, bins provide for memory usage reduction (assuming bins are smaller in number) and reduce communication cost for parallel running.

Benchmark

I used **DecisionTreeRegressor** with default arguments as the benchmark model. Other models' performances will be compared to that of DecisionTreeRegressor, as all models are fitted with the same train and validation datasets.

III. Methodology

Data Pre-processing

With thorough understanding of data used from exploratory data analysis, including additional features, data per-processing and transformation steps needed become pretty clear and obvious. We could do this modern ML Engineering way with breaking down into modules, building unit tests etc.; however, I chose to do this within notebook to stay focused on project and finishing it on time, instead of getting side tracked with beautification and production grade hardening of ML Feature engineering pipeline. It should be clear that any production grade implementation of ML solution should take care of building data pipelines, testing, performance evaluation and scaling solution to meet user expectations.
Here are the details of the given dataset:

1. **Id** - Integer unique id of each movie

16 About CatBoost - https://catboost.ai/docs/concepts/about.html

17 Features of LightGBM - https://lightgbm.readthedocs.io/en/latest/Features.html

2. **belongs_to_collection** - Contains the TMDB Id, Name, Movie Poster and Backdrop URL of a movie in JSON format. You can see the Poster and Backdrop Image like this: Example https://image.tmdb.org/t/p/original//iEhb00TGPucF0b4joM1ieyY026U.jpg
3. **budget -** Budget of a movie in dollars. 0 values mean unknown.
4. **genres** : Contains all the Genres Name & TMDB Id in JSON Format
5. **homepage** - Contains the official homepage URL of a movie.
6. **imdb_id** - IMDB id of a movie
7. **original_language** - Two digit code of the original language, in which the movie was made. Like: en = English, fr = french.
8. **original_title** - The original title of a movie. Title & Original title may differ, if the original title is not in English.
9. **overview** - Brief description of the movie.
10. **popularity** - Popularity of the movie in float.
11. **poster_path** - Poster path of a movie.
12. **production_companies** - All production company name and TMDB id in JSON format of a movie.
13. **production_countries** - Two digit code and full name of the production company in JSON format.
14. **release_date** - Release date of a movie in mm/dd/yy format.
15. **runtime** - Total runtime of a movie in minutes (Integer).
16. **spoken_languages** - Two digit code and full name of the spoken language.
17. **status** - Is the movie released or rumored?
18. **tagline** - Tagline of a movie
19. **title** - English title of a movie
20. **Keywords** - TMDB Id and name of all the keywords in JSON format.
21. **cast** - All cast TMDB id, name, character name, gender (1 = Female, 2 = Male) in JSON format
22. **crew** - Name, TMDB id, profile path of various kind of crew members job like Director, Writer, Art, Sound etc.
23. **revenue** - Total revenue earned by a movie in dollars.

For data preparation work, I had to take care of handling JSON, with potentially repeated key value pairs. So, wrote couple of helper function to get dictionary for a given JSON string, and get counts of each key in dictionary, so that I can make a determination to eliminate any low frequency keys and biased data. So far as the data preparation, the function speaks for itself, here are few highlights:

1. **Release_date:** Release date comes in as a string from input dataset, and it has 2 digit year. Took care of adding first two digits of year appropriately, after checking range of data in year. After cleansing individual parts of date, derived day of the week and quarter as features.
2. **Rating and Voting:** These two attributes are obtained from additional features, so, had to merge with main data set, and since merge mismatches are significant, came up with strategies for filling missing values, be defaulting rating to 1.5 and votes to 5, ensuring this defaulting is not going to impact overall model training. Also, derived weighted rating using combination of rating and total votes.
3. **Budget:** Our dataset spans on 100 years. Dollar value goes through inflation, and assuming value is same across that span is naive. At the same time, didn't want to make it super complicated with looking up actual inflation tables, used simplistic approximation of 2.1% inflation per year.
4. **Runtime:** Some movies had Runtime of zero, data quality error most probably. Replaced such zeros with average runtime.
5. **Counts:** Derived counts of genders, number of keywords, number of cast members, etc.
6. **Indicators:** For attributes like homepage, belongs_to_collection, tagline etc., applied transformations and converted them to binary values.
7. **Ratios:** Derived few ratios with budget and runtimes, popularity with release year, rating with popularity, budget with rating etc., as they would make meaningful features as we explore models.
8. **String attributes:** For remaining string attributes like Title, overview, Tagline etc, computed lengths of strings, letter and word counts to see if they could become useful features.
9. **One Hot encoding:** For values found in JSON strings like Genres, Production_countries etc., used get_dummies to perform one hot encoding.
10. **Label encoding:** For collection_name used label encoder to convert it to categorical values.

Implementation

For implementing model training, started out reading training and test datasets along with additional feature datasets, merged additional datasets with left joins on imdb_id. Ensured consistency between test and training datasets, shapes and values before applying feature engineering functions. Cleared out test dataset's revenue by setting it to NaN. Applied log transformation to revenue column for the reasons cited earlier with data distribution skewness of very few large numbers for blockbuster hits compared to majority of low revenue movies. Split-ted train and test datasets after feature engineering and commenced training model.

As planned, started training process with establishing benchmark model, DecisionTreeRegressor with default arguments. Objective of this benchmark model is to compare performance of models with the benchmark. For training benchmark model, used simplistic approach with training and test split of 80/20 within training data RMSE stands at 3.27 and RMSLE stands at 0.33 for benchmark model.

For initial model training, I started off with default parameters, here is version of each library I used, along with default parameters tried out:

1. LightGBM 2.2.3 Training RMSE was at 1.19 and Validation was at 1.89

```
boosting_type='gbdt', num_leaves=31, max_depth=-1, learning_rate=0.1, n_estimators=100,
subsample_for_bin=200000, objective=None, class_weight=None,
min_split_gain=0.0, min_child_weight=0.001, min_child_samples=20,
subsample=1.0, subsample_freq=0, colsample_bytree=1.0, reg_alpha=0.0, reg_lambda=0.0,
random_state=None, n_jobs=-1, silent=True, importance_type='split'
eval_metric='rmse', verbose=1000, early_stopping_rounds=5
```

2. CatBoost 0.16 Training RMSE was at 2.35 and validation was at 2.3

```
'learning_rate': 1,'depth': 2,'allow_writing_files': False, iterations=5, verbose=True
```

3. XGBoost 0.90 Training RMSE was at 1.45 and validation was at 2.05

```
'eta': 0.3, 'objective': 'reg:linear','max_depth': 6,'subsample': 1,'colsample_bytree':
1,'eval_metric': 'rmse','seed': 0,'silent': True, num_boost_round=10,
early_stopping_rounds=20, verbose_eval=500
```

It was interesting to see LightGBM beat both CatBoost and XGBoost with initial attempt, kind of expected with small dataset I am training on. Didn't really into any coding complications; however, comprehending so many parameters, their meaning, and applicability was an amazing learning experience. I spent couple of weekends just reading, tweaking and adjusting hyperparamters before I landed on the on optimal final hyperparameters. I am sure there are many great ways of navigating through hyperparamter search space, and that is an area I intend to learn and improve my skills as I practice.

For actual model training and visualization of features of importance, I used feature importance outputs from Light GBM (as depicted in Figure 14):

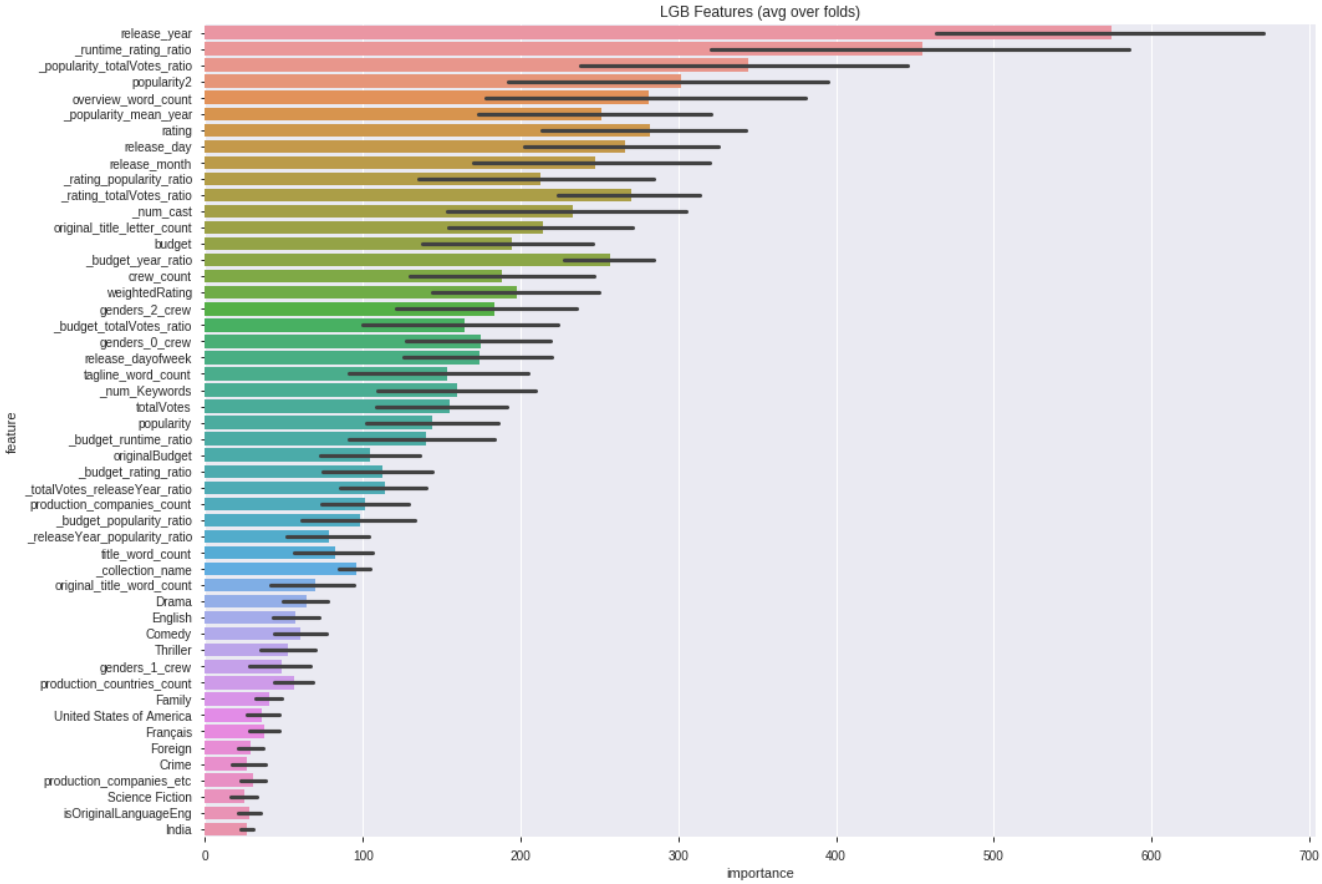


Figure 14

Reviewed feature weights using eli5(Figure 15):

WeightFeature
0.2258 _budget_year_ratio
0.1081 _rating_totalVotes_ratio
0.0690 budget
0.0670 release_year
0.0435 _popularity_mean_year
0.0366 totalVotes
0.0365 _totalVotes_releaseYear_ratio
0.0365 popularity2
0.0253 _runtime_rating_ratio
0.0249 _popularity_totalVotes_ratio
0.0173 rating
0.0164 release_day
0.0153 popularity
0.0151 originalBudget
0.0140 _num_cast
0.0139 weightedRating

WeightFeature
0.0127 overview_word_count
0.0122 _budget_totalVotes_ratio
0.0120 _rating_popularity_ratio
0.0119 release_dayofweek
… *184 more* …

Figure 15

Used shap summary plots as well to visualize feature importance and impact on model outputs(as in figure 16):

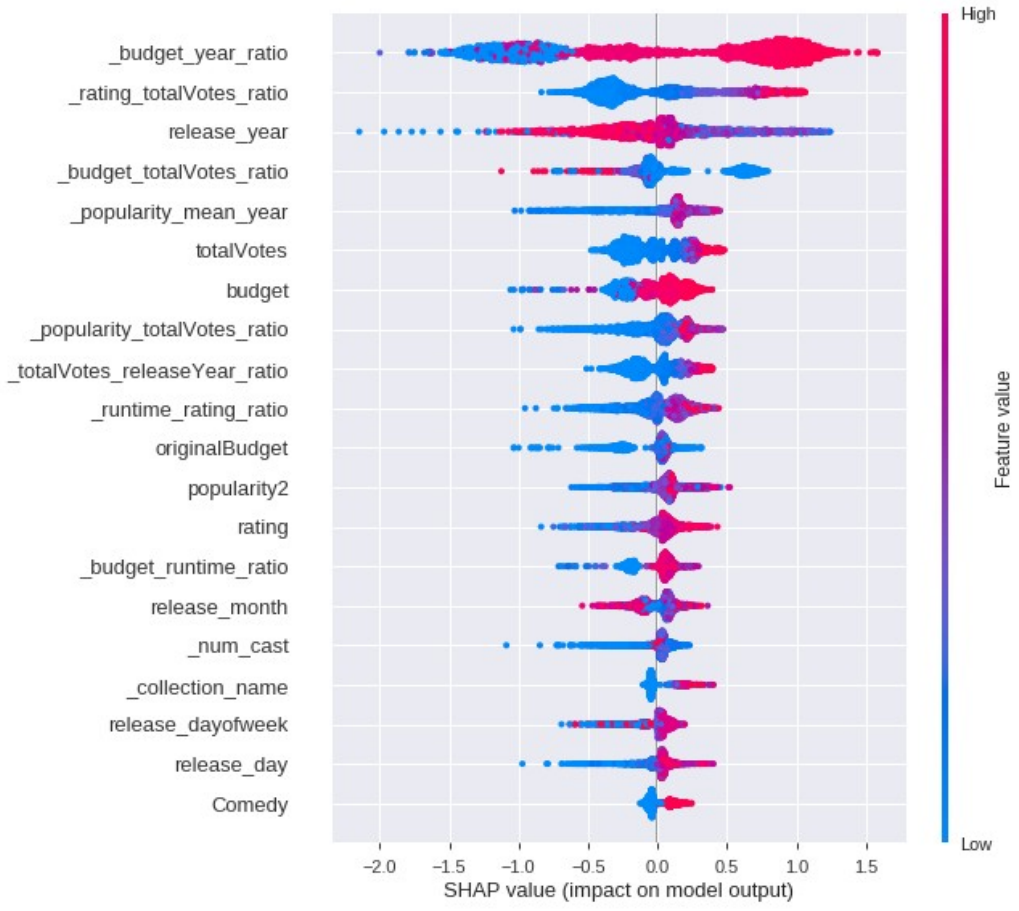


Figure 16

For actual model training, I decided to use stacking and blending techniques[18] on three chosen algorithms (LightGBM, CatBoost and XGBoost), in addition to individual model predictions. Since I was working with pretty small dataset of 3K movies, using K-fold cross validation would be a good choice. Tried with few values of K, got good results with 6 folds, and chose that for final training.

18 Tips for stacking and Blending - https://www.kaggle.com/zaochenye/tips-for-stacking-and-blending

Refinement

For each of the models, tried at least 3 combinations of hyperparamters and chose optimal by using GridSearchCV and varying parameter grids for few combinations, this is an area I intend to learn and improve my skills as I practice further. The goal of this modeling exercise was to minimize RMSLE, which would mean that predictions are closer to actual revenues.

Here are the final hyperparameters I settled on:

LightGBM 2.2.3 – tuned for num_leaves, max_depth and learning_rate hyperparameters

```
objective':'regression','num_leaves' : 30,'min_data_in_leaf' : 20,'max_depth' : 9, 'learning_rate':
0.004,'min_child_samples':100,'feature_fraction':0.9,"bagging_freq": 1,"bagging_fraction":
0.9,'lambda_l1': 0.2,"bagging_seed": 200,"metric": 'rmse','subsample':.8,
'colsample_bytree':.9, "random_state" : 200, "verbosity": -1, n_estimators = 20000, nthread = 4
```

CatBoost 0.16 - tuned for learning rate, depth, l2_leaf_reg hyperparamters

```
'learning_rate': 0.004,'depth': 5,'l2_leaf_reg': 10,'colsample_bylevel': 0.8,'bagging_temperature':
0.2,'od_type': 'Iter','od_wait'
```

XGBoost 0.90 - tuned for eta, subsample, colsample_bytree, num_boost_round hyperparamters

```
'eta': 0.01, 'objective': 'reg:linear','max_depth': 6,'subsample': 0.6,'colsample_bytree':
0.7,'eval_metric': 'rmse','seed': 25,'silent': True, num_boost_round=20000, early_stopping_rounds=200,
verbose_eval=500
```

Following table I summarizes initial modeling attempt with default RMSE/RMSLE, compared to final Hyperparameter tuned RMSE/RMSLE. Reviewing values, it appears LightGBM was overfitting with limited validation data. CatBoost model training detected overfitting when it reached 1.78 on training set and dropped out. Comparing performances of each of the model to benchmark model, it did beat benchmark models performance consistently.

Algorithm	Initial RMSE(default Parms)		Hyperparameter Tuned RMSE	
	Train	Validation	Train	Test
LightGBM v2.2.3	1.19	1.89	1.03	2.03
CatBoost v0.16	2.35	2.3	1.78	2.01
XGBoost v0.90	1.45	2.05	1.01	2.05
	Initial RMSLE(default Parms)		Hyperparameter Tuned RMSLE	
	Train	Validation	Train	Test
LightGBM v2.2.3	0.08	0.28	0.01	0.31
CatBoost v0.16	0.37	0.36	0.25	0.30
XGBoost v0.90	0.16	0.31	0.00	0.31
Benchmark – DecisionTreeRegressor	RSME=3.18		RSMLE=0.33	

Table I

Ideal value would be a zero for RMSLE, which would indicate perfect fit for given data, but that would overfit model to the given training data. An important point to mention is that hyperparameter tuning takes significant time and effort, potentially using automated solutions to explore hyperparameter space in automated fashion would have been better, I clearly lack skills in that area, and intent to pursue acquiring skills in that area.

IV. Results

Model Evaluation and Validation

The final model I am using for this problem is blend of all 3 models, based on the heatmap of correlations and scatter matrix plots that follow. It is inherent problem in choosing a competition problem for the project as actual values for test dataset is not known, all models developed RMSE at or around 2, compared to simple benchmark model at 3.18.

In particular, for model evaluation and validation I chose cross validation with 10 folds for chosen model, cross validation scores are listed as follows:

Fold 0 started at Tue Aug 27 20:01:29 2019

```
Training until validation scores don't improve for 200 rounds.
Early stopping, best iteration is:
[402]   training's rmse: 1.95096        valid_1's rmse: 1.91741
Fold 1 started at Tue Aug 27 20:01:29 2019
Training until validation scores don't improve for 200 rounds.
Early stopping, best iteration is:
[412]   training's rmse: 1.94475        valid_1's rmse: 1.93522
Fold 2 started at Tue Aug 27 20:01:30 2019
Training until validation scores don't improve for 200 rounds.
```

```
Early stopping, best iteration is:
[364]   training's rmse: 1.92848        valid_1's rmse: 2.14938
Fold 3 started at Tue Aug 27 20:01:30 2019
Training until validation scores don't improve for 200 rounds.
Early stopping, best iteration is:
[276]   training's rmse: 1.96896        valid_1's rmse: 1.94637
Fold 4 started at Tue Aug 27 20:01:31 2019
Training until validation scores don't improve for 200 rounds.
[1000]  training's rmse: 1.872  valid_1's rmse: 2.28419
Early stopping, best iteration is:
[865]   training's rmse: 1.87848        valid_1's rmse: 2.28051
Fold 5 started at Tue Aug 27 20:01:31 2019
Training until validation scores don't improve for 200 rounds.
Early stopping, best iteration is:
[289]   training's rmse: 1.95733        valid_1's rmse: 2.0107
Fold 6 started at Tue Aug 27 20:01:32 2019
Training until validation scores don't improve for 200 rounds.
Early stopping, best iteration is:
[420]   training's rmse: 1.95664        valid_1's rmse: 1.88253
Fold 7 started at Tue Aug 27 20:01:32 2019
Training until validation scores don't improve for 200 rounds.
Early stopping, best iteration is:
[518]   training's rmse: 1.92253        valid_1's rmse: 2.10808
Fold 8 started at Tue Aug 27 20:01:33 2019
Training until validation scores don't improve for 200 rounds.
Early stopping, best iteration is:
[475]   training's rmse: 1.93791        valid_1's rmse: 1.94623
Fold 9 started at Tue Aug 27 20:01:34 2019
Training until validation scores don't improve for 200 rounds.
Early stopping, best iteration is:
[206]   training's rmse: 1.98601        valid_1's rmse: 2.10761
CV mean score: 2.0284, std: 0.1212.
```

RSMLE of chosen model is at 0.30, reviewing results across folds, the validation performance across each individual validation fold is stable around 2, and doesnt fluctuate much; we can be confident that model is robust against small perturbations in the training data.

Visualizing the heatmap (figure 17) of predictions by candidate models, it is evident that “pred_blend_all3” has perfect correlation with XGB and “blend_xgb_lgb” and is at 0.99 for rest 3 models. Drawing that intuition from heatmap, and reviewing scatter matrices, which also support “pred_blend_all3” is most appropriate for this prediction problem. From scatter plots (figure 18) it is evident that for higher values of revenues we see higher variation, expected in prediction problems with skewed data.

Figure 17

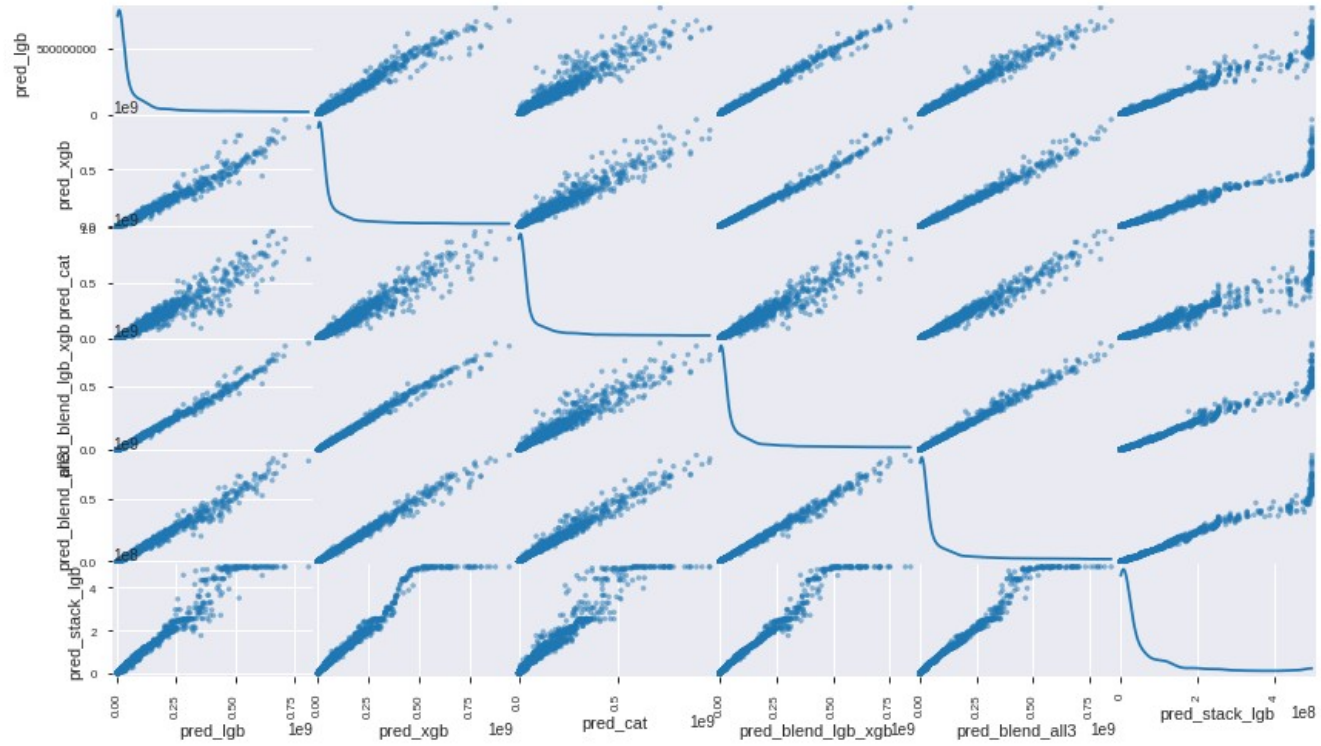


Figure 18

The model has been tested with test dataset given for competition, it does generalize well and meets objective of minimizing RMSE error. I believe the model results can be trusted for reasonable accuracy; however, the problem domain of predicting movie revenues is inherently multi-dimensional and can use

help from many other ML techniques. For purposes of this exercise, I am immensely pleased with the outcome and objective of error optimization.

V. Conclusion

Free-Form Visualization

Stacking models was a very interesting topic to me, significance of each model stacked to overall needs exploration. From following visual (figure 19), we can see that catboost model contributed the most and lightgbm the least. The stacking methods[19] combine multiple regression models via a meta-regressor. The base level models are trained on complete training dataset, then the meta-model is trained on the output of the base level models as features.

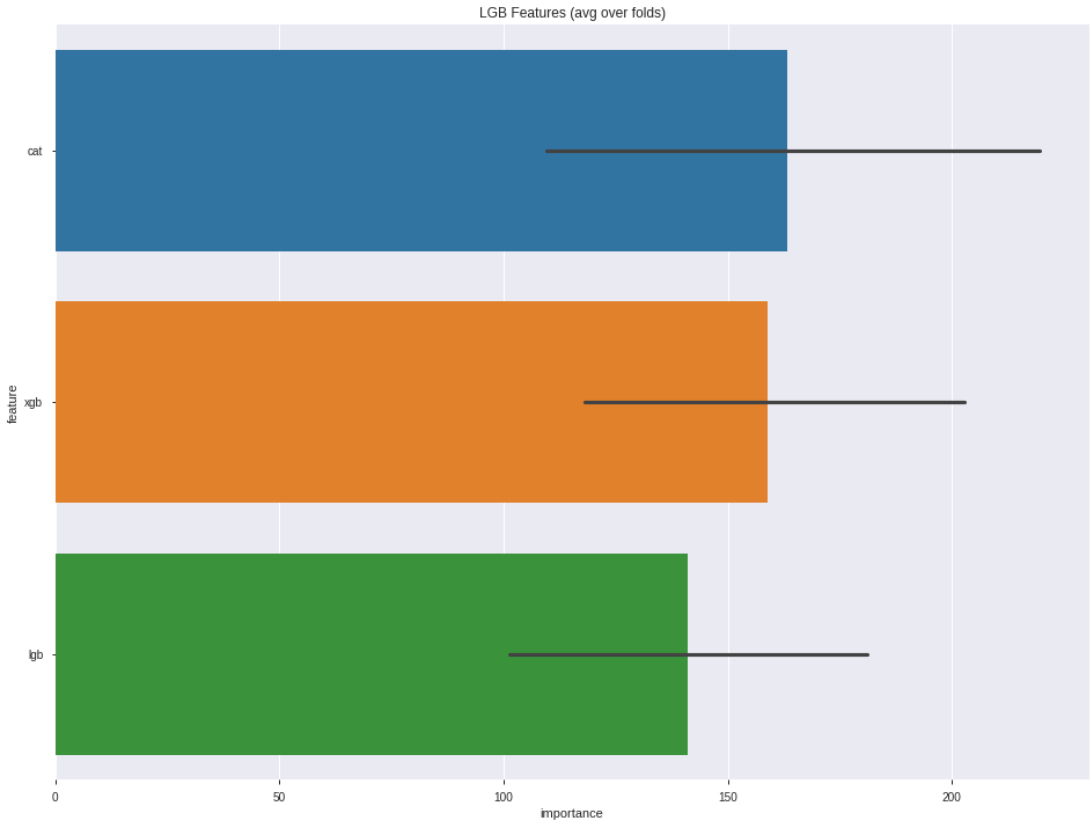


Figure 19

Reflection

As part of model training and selection, I did try quite a few techniques as documented in project proposal, which was primarily based on my limited knowledge of one of the earlier project where model

19 Ensemble Learning to Improve Machine Learning Results - https://blog.statsbot.co/ensemble-learning-d1dcd548e936

accuracy and runtime was compared across many modeling techniques. As I was exploring results and various ways other ML practitioners approach this problem, especially for competitions[20] (I referred to many of these notebooks, experimented with packages I had never heard of like eli5, shap etc., I realize this is common practice in ML to review other practitioners' work and learn from each other; however, work on this project though might have been influenced by various ideas, it is my own. I appreicate such vibrant community of Kaggle competitors and give due credit to the community) came to realization that my naive approaches would fall short. So, decided to explore stacking and blending would be appropriate for the given problem, which produced much better results, minimized chosen metric. During the process, my ML knowledge improved significantly, learned in detail how XGBoost, CatBoost, and LightGBM work, their hyperparameters and techniques to adopt them to given problem.

Before wrapping up this project, I would like to mention that I could have chosen to go with other routes with algorithms and techniques used. I could have used plain simple linear regression methods or deep learning methods. Probably they would have performed better than tree based models, or might not be based on limited dataset given for the problem. Normally deep learning needs lots of data, lots of compute power, so, I didn't go that route. I could have gone in adopting public cloud route and tried this with SageMaker on AWS, could have explored various text, Natural Language Processing (NLP) based algorithms, though I should admit that would take long time, as I am yet to learn NLP techniques. I learned just enough to visualize word clouds. I could have chosen to explore lot more data sources to augment features of interest. However, One thing pretty clear to me in developing ML solutions is that there are many, many ways to solve same problem; we do have limited resources like time to complete project, compute resources, and skillset of ML Engineer working on the problem.

For me, pretty much everything in the project was interesting, this project gave me a perfect opportunity to take what I learned over the course of six months in this Nano Degree course and apply my skills to a practical problem. Of course, I had my difficulties as being naive ML practitioner adopting various techniques for ML. It is probably worth mentioning that my day job focuses on other domains of ML Engineering like Infrastructure, model deployments, model monitoring, statistical analysis on model performance etc., not real hands on model development. This project gave me ample opportunities to learn how a typical ML model is developed, pitfalls to watch out for, importance of feature engineering and making sure data transformations are correct. This project also gave me opportunities to improve upon my base knowledge of Pandas and NumPy as applied to practical problem, in addition to various tree based ensemble techniques I learned. During the process of working on this project, I have

20 Kaggle TMDB Box office Prediction Kernels - https://www.kaggle.com/c/tmdb-box-office-prediction/notebooks

developed enough skills, intuition and appreciation for ML model development. I feel confident to take these skills as I interact with many ML practitioners developing state of the art models in my organization.

I feel that techniques used for this modeling effort are generic enough to use it similar linear regression problems. I am an avid supporter of AutoML techniques, potentially approaches used in this project can be generalized to solve any linear regression problem in future.

Improvements

The implementation can further be improved to further optimize and reduce prediction error function. For instance, the hyperparameter space for stacked ensemble can be searched further for optimal solution, which would be computationally intensive. Since there are few free form text attributes, adopting Natural Language Processing techniques, figuring out semantic meaning and extracting features out of such text columns would make feature engineering much more robust! NLP techniques are beyond my current skillset. As mentioned little earlier, deep learning opens up wide array of options to further improvements, for instance we could learn more about plots by using image/video analysis on climax or important sequences of movies. Could be extended to audio/tone analysis, to figure out what works well with moviegoers and how it changes over time. Possibilities are endless for improvements, ML and AI is ever-growing with new algorithms and techniques. I would definitely think a better solution exists if my solution is used as new benchmark and further improved upon. However, under given constraints of educational setting and hard timelines, I am pretty happy with the results I got. I should take this opportunity to thank Udacity staff and reviewers for being so prompt and supportive.