2019

COMP4211 Project – TMDB Box Office Prediction

SHAYAN IMRAN

20316792

Contents

Introduction

This project presents experimentation using multiple types of machine learning techniques on the TMDB Box Office Prediction Kaggle Competition (https://www.kaggle.com/c/tmdb-box-office-prediction/overview). The aim of the competition is to predict the revenue for movies given pre movie release metadata for over 7000 past movies.

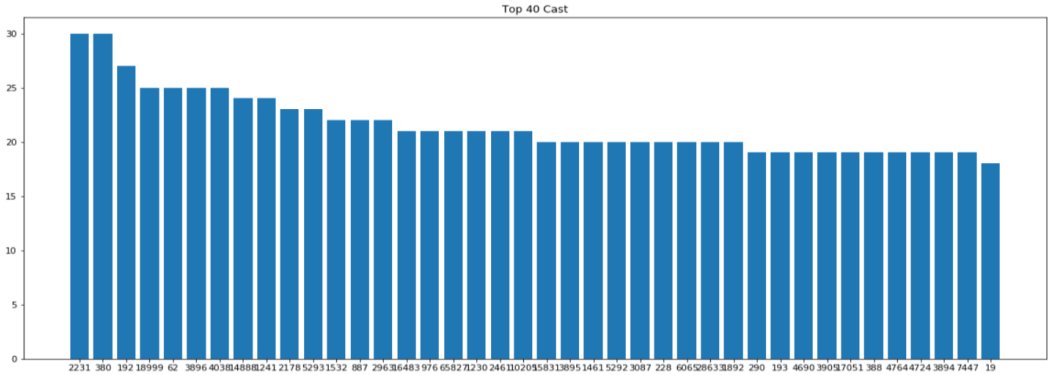
Dataset and Preprocessing

The dataset provided in this competition consists of 22 columns of metadata for 7398 movies from the TMDB movie database. Only 3000 of the movies in this list contain a 23rd column containing the revenue earned by the movie, predicting which is the target of this competition. These 3000 rows are therefore used to train the machine learning models that are used.

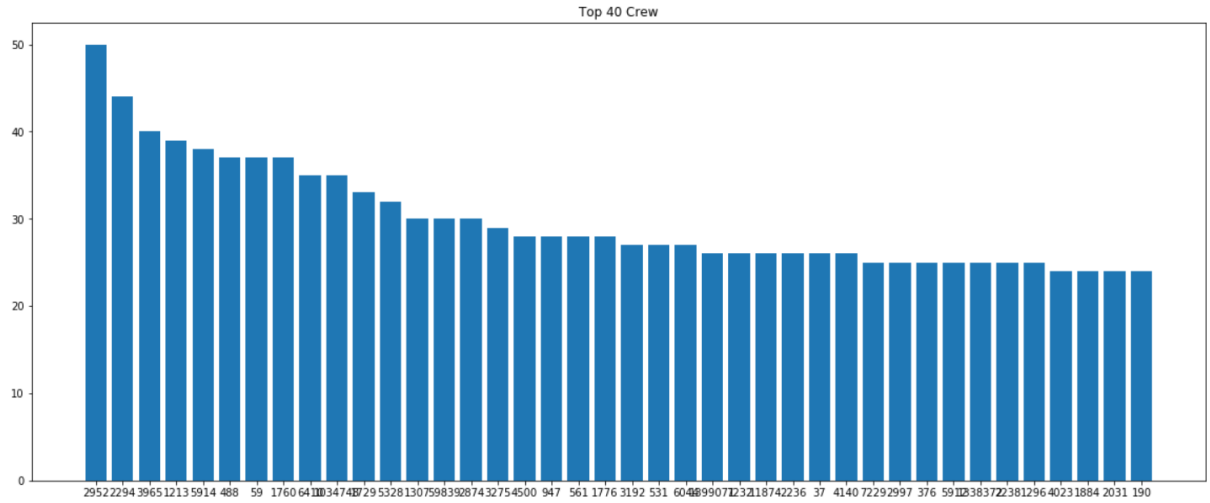
Dropped Columns

The following columns were dropped as they are believed to be irrelevant and do not contribute much to the machine learning task:

- id
- imdb_id
- original_title
  The original title of the movie in its original language
- overview
  A few sentences introducing the movie
- poster_path
  A web location for the poster in the TMDB website
- status
  Whether the movie has been released or not
- spoken_languages
  The languages spoken in the movie eg. [{'iso_639_1': 'ar', 'name': 'العربية'}, {'iso_639_1': 'en', 'name': 'English'}, …]
- tagline
- title
  The title of the movie in English
- Cast
  A list of the cast in the movie eg. [{'cast_id': 4, 'character': 'Lou', 'credit_id': '52fe4ee7c3a36847f82afae7', 'gender': 2, 'id': 52997, …} …]. The data contains 38,760 unique cast names, but each of these have very low occurrences, due to which this feature is dropped.



- Crew
  A list of the crew in the movie eg. [{'credit_id': '59ac067c92514107af02c8c8', 'department': 'Directing', 'gender': 0, 'id': 1449071, 'job': 'First Assistant Director', 'name': 'Kelly Cantley', 'profile_path': None}, ...]. The data contains 38,897 unique crew members, but the occurrences of the crew members is low, with the top 40 crew occurring between 20 and 50 times in the data.



One Hot Encoding

original_language

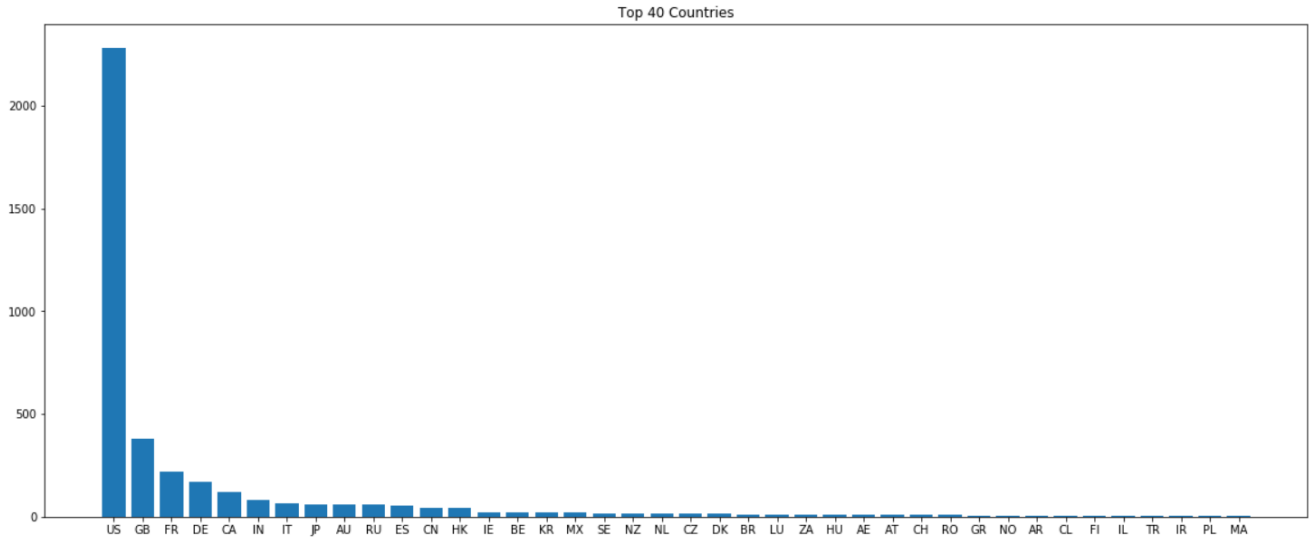
This is the original language of the movie, the original data consists of around 40 different languages, but the occurrence of these languages is quite skewed; 85% of the movies were English movies, while around 25 of the other languages occurred less than 10 times in the data. In order to get good data representation, while minimizing the number of columns required for one hot encoding, only languages that occurred in at least 1% of the data were chosen. Altogether, this represented 94.1% of the entire dataset. The languages chosen are English, Spanish, French, Hindi, Japanese and Russian.

genres

A list of genres this movie belongs to eg. [{'id': 28, 'name': 'Action'}, {'id': 35, 'name': 'Comedy'}, …]. There are a total of 20 unique genres, so all of these are one hot encoded.

production_countries

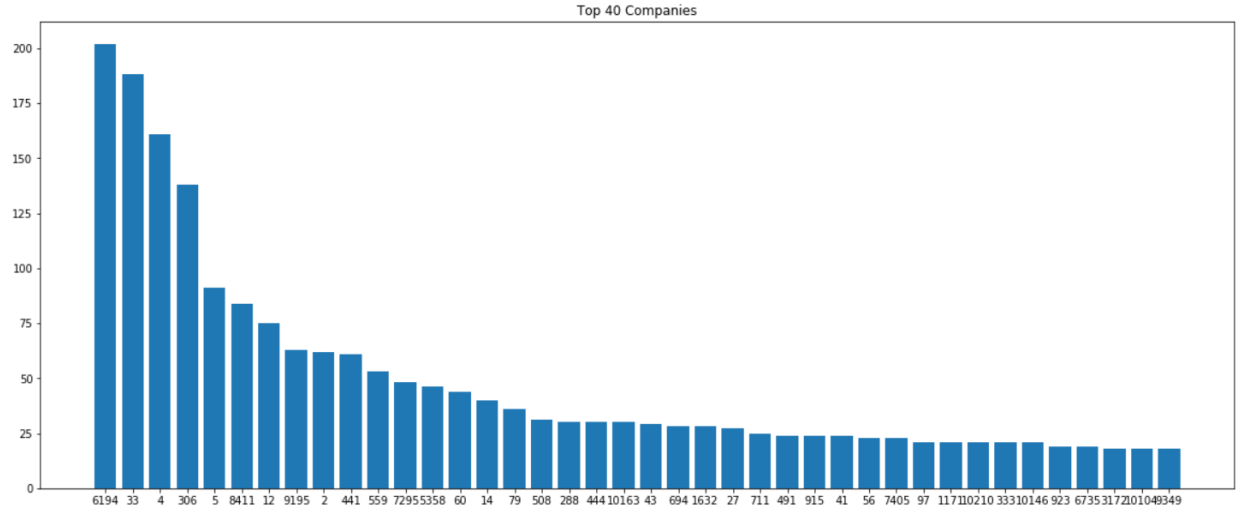
A list of the countries where the movie was produced eg. [{'iso_3166_1': 'US', 'name': 'United States of America'}, {'iso_3166_1': 'CA', 'name': 'Canada'}, …]. There are 74 unique countries, but the distribution of these countries is quite skewed as shown by the following table:



54 out of the 74 top occurring countries are chosen and one hot encoded.

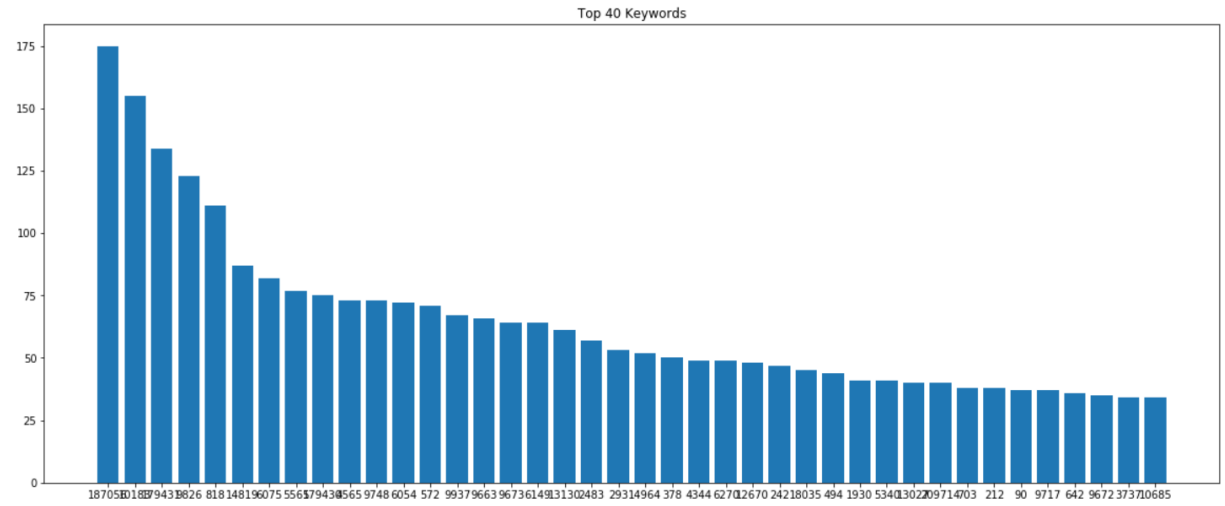
production_companies

A list of the companies involved in the production of the movie eg. [{'name': 'Paramount Pictures', 'id': 4}, {'name': 'United Artists', 'id': 60}, …]. There are 3712 unique companies, out of which the top occurring 1034 are selected for one hot encoding and only the id is kept. This is because the occurrences of companies rapidly drop off, as show below, so keeping values that occur very few times in the entire dataset may result in overfitting.



Keywords

Keywords for the movie eg. [{'id': 4379, 'name': 'time travel'}, {'id': 9663, 'name': 'sequel'}, …]. There are 7400 unique keywords in the dataset out of which only half are used, and only the id of each keyword is kept. The distribution of Keyword occurrences follows a similar trend to the previous two features, as shown below.



Binary Transformation

belongs_to_collection

The name of the series this movie belongs to, if it belongs to one eg. a Transformers movie belongs to the Transformers movie collection. Since this column is sparse and the collection names are mostly unique, the column was transformed to a binary column with a value of 1 if the movie belongs to a collection, otherwise 0.

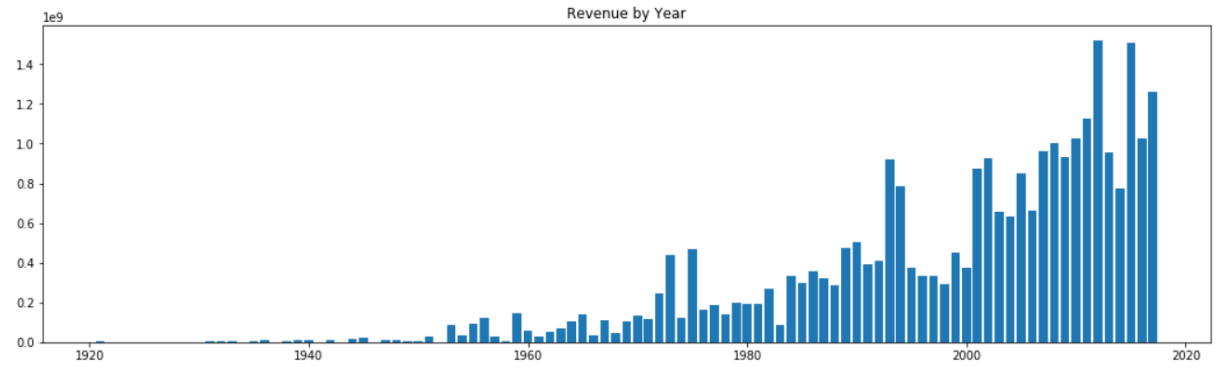
homepage

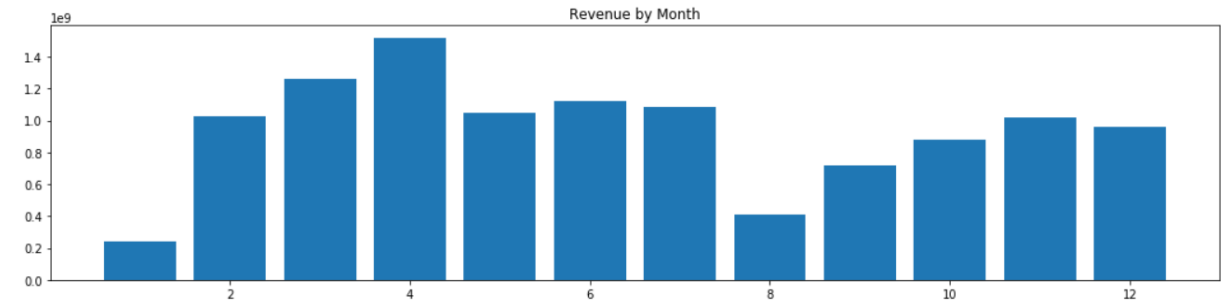
A link to the website of the movie and it too is very sparse, so the values are transformed to 1 if there is a homepage, otherwise 0.

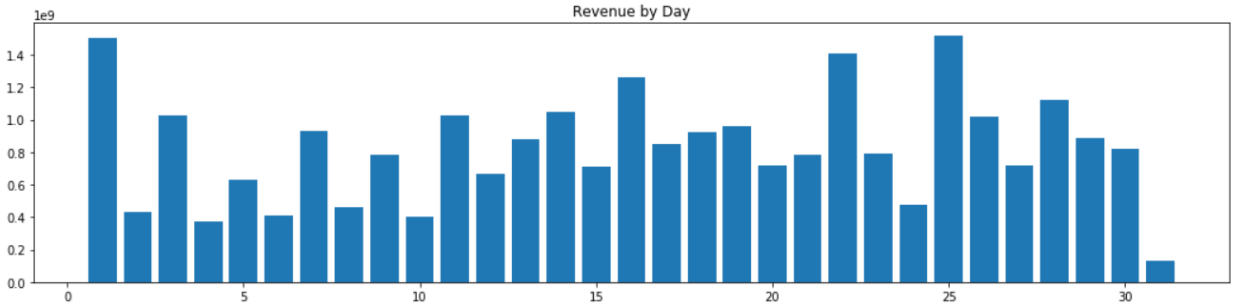
Release Date Transformation

The release date is given as a string of the form MM/DD/YY, this string is separated into 3 columns for day, month and year. The data presents some ambiguity as it does not specify whether the year given is in the 1900s or the 2000s. Therefore, it is assumed that all values for year between 00 and 19 are in the 2000s, while anything less than 00 and over 19 is in the 1900s.

The date presents some interesting trends as shown below, so it can be assumed that this will be an important feature in predicting the revenue. This is especially clear in the revenue by year, which trends upwards over time. The revenue obtained by year, month and day are presented below.



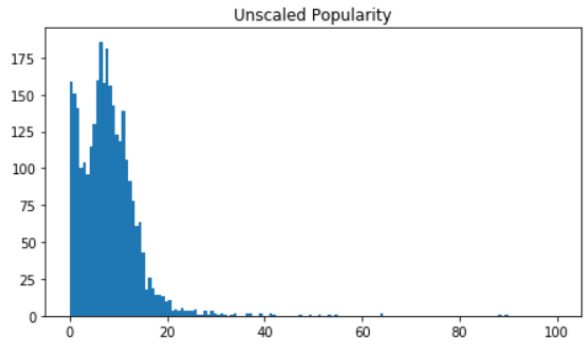


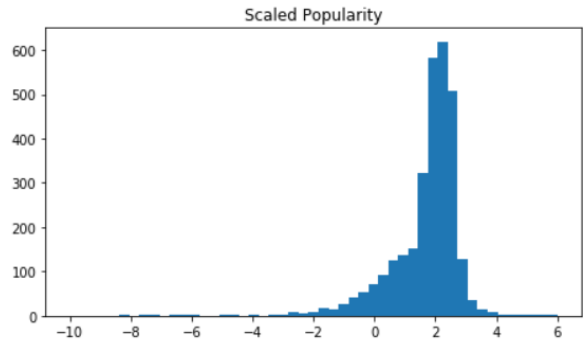


Feature Scaling

popularity

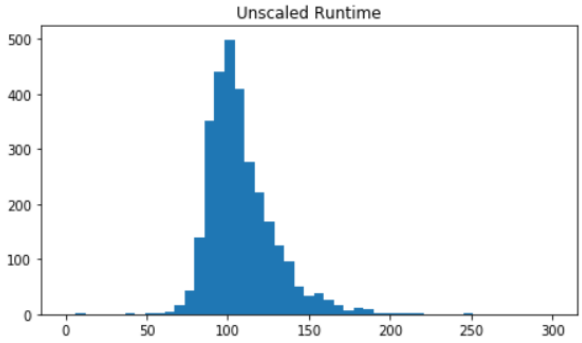
A real number which seems to be higher the more popular the movie is, but it is not explained how this is obtained. This is scaled using sklearn's StandardScaler which standardizes features by removing the mean and scaling to unit variance. The popularity is skewed as there are a very small number of popularities which are extremely high, which could affect the regression as these large values may overshadow the small ones, which is why it must be scaled.

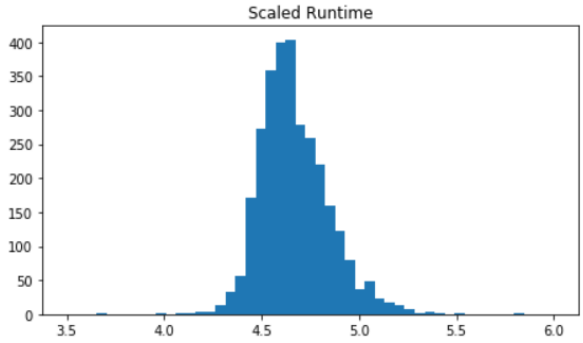




runtime

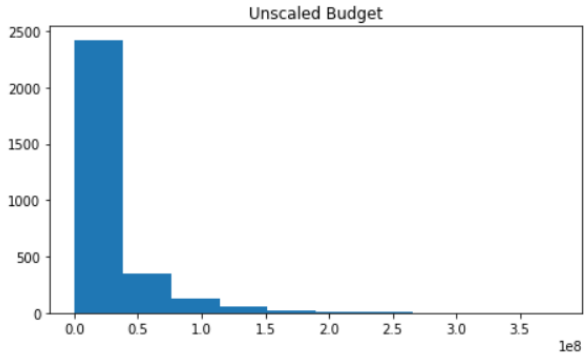
The runtime of the movie in minutes, this data is scaled using a StandardScaler which standardizes features by removing the mean and scaling to unit variance.

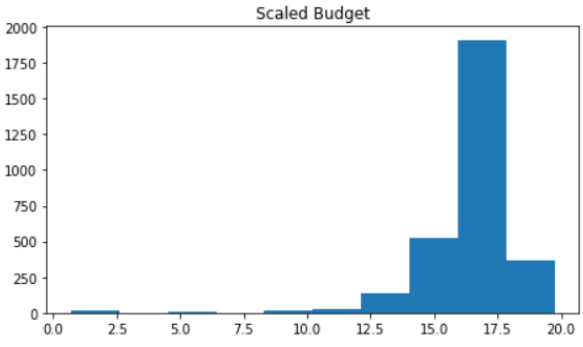




budget

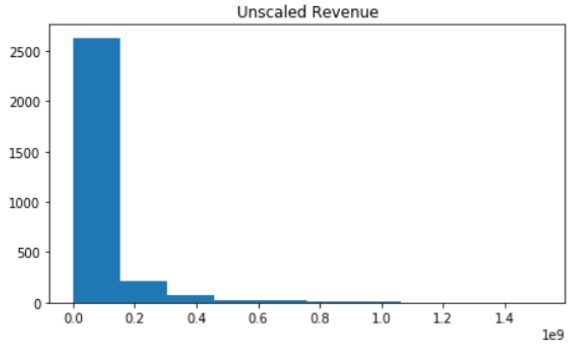
The budget is very skewed to lower values and there is a large difference between the smallest and largest values. Therefore, the budgets are transformed to log values using numpy.log1p.

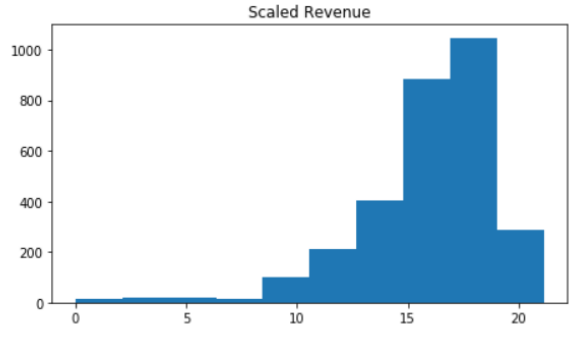




revenue

The revenue is the target feature that is present only in the training dataset, and it is very skewed towards lower values with a large difference in the smallest and largest ones. This could greatly affect the error statistic as the prediction results for blockbuster movies with large revenues could overshadow the error for smaller ones, therefore not training the models well. Therefore, the revenue is transformed to log values using numpy.log1p.





Machine Learning Task

The Kaggle challenge involves using the given metadata to predict movie revenues on the test dataset, using the training data. Since revenue is a continuous variable, this is a regression task

In order to perform the machine learning task, the loss function that is minimized is the Root Mean Squared Log Error:

$$L(w;S) = \sqrt[2]{\frac{1}{N}\sum_{l=1}^{N}(f(\log(x^{(l)};w)) - \log(y^{(l)}))^2}$$

This is done in the code using the following function:

```
def rmsle(y, y_pred):
    assert (y > 0).all() and (y_pred > 0).all(), "y or y_pred have a non-positive value"
    out = np.square(np.log(np.expm1(y)) - np.log(np.expm1(y_pred)))
    out = np.sqrt(np.mean(out))
    return out
```

Note: In the code the predicted and actual values are transformed using np.expm1 first as the actual revenue values were transformed using the log function in the preprocessing step, so the numpy.expm1 function reverses this process to give the actual revenue.

A Regressor class is created in the code for convenience, to run the regression and to report the output on user provided inputs.

The models used for the regression task are:

- Baseline: LASSO
- Ridge Regression
- Decision Tree Regression
- Random Forest Regression
- XGBoost Regression

Additionally, the model parameters are validated using a 50:30:20 data split, 50% of the data is used for training and 30% is used for validation. After validating and selecting the best parameters, the model is retrained using all the data in the 80% that was used for training and validation, and then tested against the remaining 20% that was unused before.

In addition to the above steps, the performance of two different cross validation techniques was also evaluated as a secondary task; sklearn.model_selection.RandomizedSearchCV and skopt.BayesSearchCV. RandomizedSearchCV tests parameters randomly drawn from a user-specified distribution for each parameter. BayesSearchCV works in a similar way, it utilizes Bayesian Optimization where a predictive model is used to model the search space and is utilized to arrive at good parameter values combination as soon as possible. It also samples from a user specified distribution for each parameter.

Machine Learning Methods

All the methods were run using 42 as the random_state to allow reproducibility.
The reported results are the loss scores obtained after running the model on the test set using the default library settings, Bayes Search CV result settings and Randomized Search CV result settings. The validation is performed for 50 iterations.

Baseline: LASSO

LASSO regression is a good model to use as a baseline as the input data contains many features after preprocessing, but as there are not a lot of examples of data there is a danger of overfitting. LASSO is meant to reduce over-fitting as during training it may lead to "zero coefficients" which can function as feature selection in a way. Hence, if the other models used can perform better than LASSO then there is a high chance that they can avoid overfitting to the training data.

Note: Bayes Search CV could not be run using this model as it resulted in some negative revenue predictions which cannot be evaluated using root mean squared log error.

Results

Validation performed:
alpha: uniform distribution between 0 and 30

	Default	Bayes Search CV	Randomized Search CV
Validation Result	1		alpha: 0.61753
Root Mean Squared Log Error	2.52961		2.522926

Ridge Regression

Ridge regression is similar to linear regression, except it adds a regularization term, this helps to avoid overfitting since the data dimensionality is high while the number of datasets are small.

Note: Bayes Search CV could not be run using this model as it resulted in some negative revenue predictions which cannot be evaluated using root mean squared log error.

Results

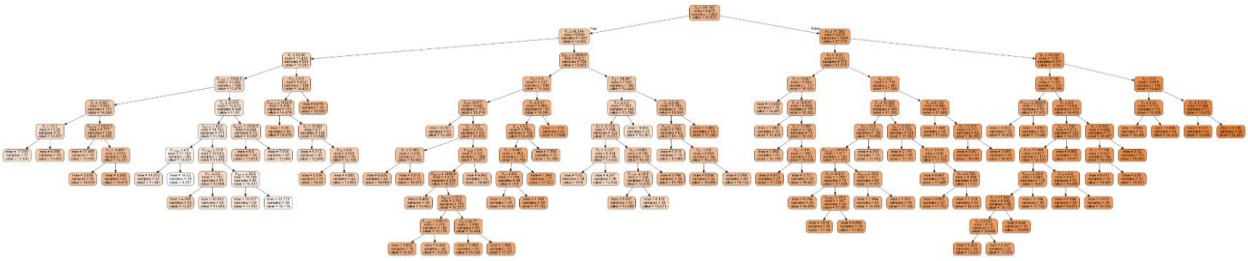
Validation performed:
alpha: uniform distribution between 0 and 20

	Default	Bayes Search CV	Randomized Search CV
Validation Result	1		alpha: 19.3981
Root Mean Squared Log Error	2.726012		2.154292

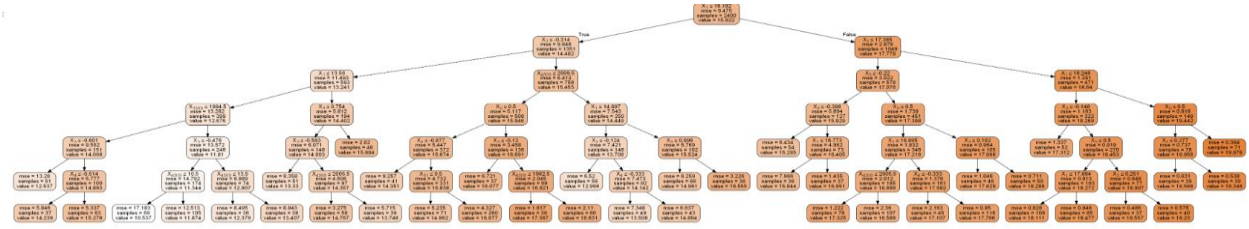
Decision Tree Regression

Default Settings Decision Tree

Bayes Search CV Decision Tree



Randomized Search CV Decision Tree



From the resulting decision tree diagrams we can see that the default settings result in a very large tree with a large number of branches, which implies that there is a very large possibility of overfitting on the training set, especially because of the small number of training samples.

The trees after validation are much smaller and have a much smaller number of branches which implies that they generalize more than the tree with the default settings. This becomes clear after looking at the results below.

Randomized Search CV performs marginally better than Bayes Search CV in this model.

Results

Validation performed:
max_depth: randint(4, 10)
min_samples_leaf: randint(2, 50)
min_samples_split: randint(2, 50)

	Default	Bayes Search CV	Randomized Search CV
Validation Result	max_depth: expands till all leaves pure min_samples_leaf: 1 min_samples_split: 2	max_depth: 10 min_samples_leaf: 19 min_samples_split: 49	max_depth: 6 min_samples_leaf: 36 min_samples_split: 34
Root Mean Squared Log Error	2.553579	2.198857	2.190488

Random Forest Regression

Random forest regression builds an ensemble of decision trees so running it on tabular data produces better result than a single decision tree.

Bayes Search CV performs better than Randomized Search CV in this model.

Results

Validation performed:
n_estimators: randint(10, 50)
max_depth: randint(1, 20)
min_samples_leaf: randint(1, 300)

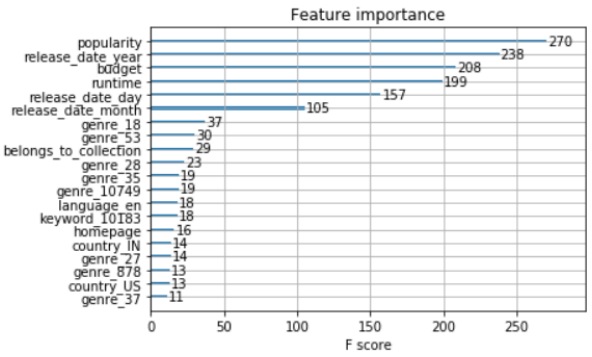
	Default	Bayes Search CV	Randomized Search CV
Validation Result	n_estimators: 10 max_depth: expands till all leaves pure min_samples_leaf: 1	n_estimators: 34 max_depth: 10 min_samples_leaf: 7	n_estimators: 33 max_depth: 7 min_samples_leaf: 9
Root Mean Squared Log Error	2.150786	2.074492	2.096104

XGBoost Regression

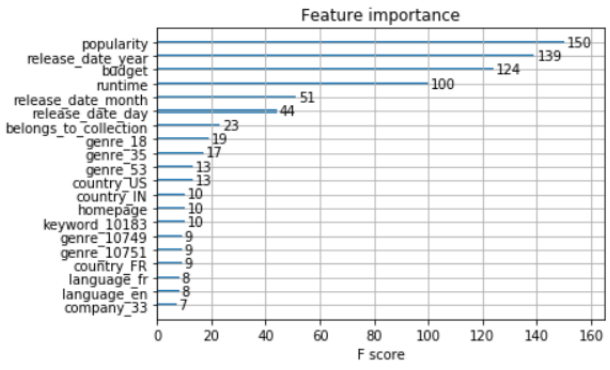
XGBoost is a very popular model on Kaggle as it produces very good results, in fact it has produced the best loss results amongst all the models tested for this project too. The model works by combining multiple weak learners in order to come up with a result. It makes sense for it to work well on this project as the data has a very large number of columns allowing XGBoost to produce several weak learners for the column.

Both of the validation methods work similarly as they produce similar feature importance orderings. The F-score produced using Bayes Search CV is, however, higher.

Bayes Search CV



Randomized Search CV



Bayes Search CV produces a better error score in this model.

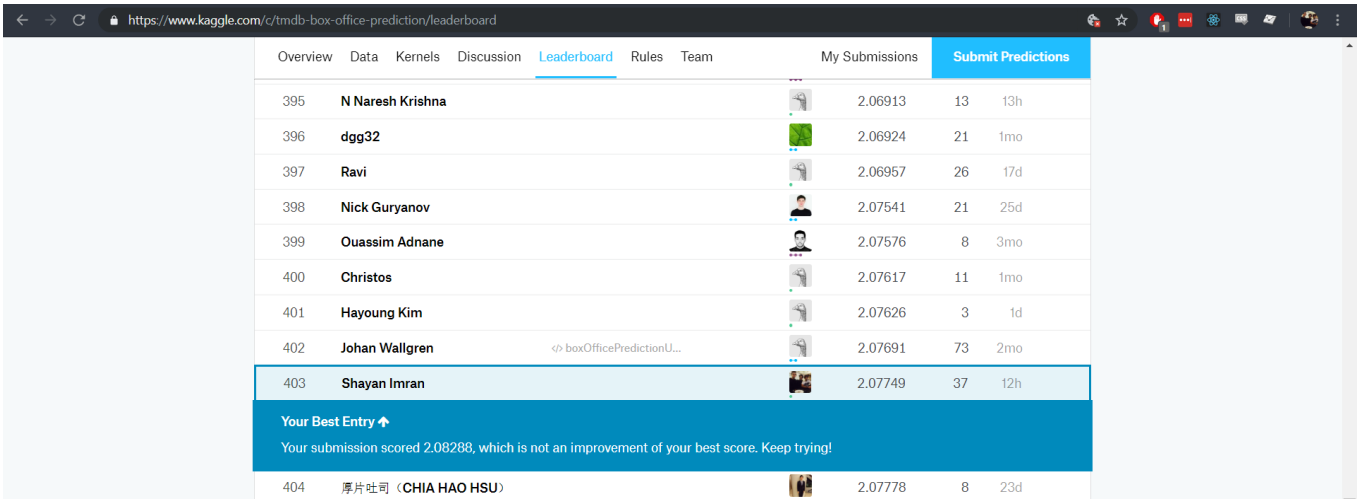
Results

Validation performed:
n_estimators: randint(10, 100)
max_depth: randint(10, 100)
min_child_weight: randint(1, 200)

	Default	Bayes Search CV	Randomized Search CV
Validation Result	n_estimators: 100 max_depth: 3 min_child_weight: 1	n_estimators: 16 max_depth: 8 min_child_weight: 67	n_estimators: 73 max_depth: 5 min_child_weight: 55
Root Mean Squared Log Error	2.122015	2.023341	2.03813

Kaggle Competition Leaderboard

Position: 403/1007
The test output resulting in the highest score used XGBoost with Bayes Search Validation.



Discussion

Conclusion

From the table below it can be seen that, after validation, all of the models outperformed the baseline.

Model	Default Loss	Bayes Search CV Loss	Randomized Search CV Loss
LASSO	2.52961		2.522926
Ridge Regression	2.726012		2.154292
Decision Tree Regression	2.553579	2.198857	2.190488
Random Forest Regression	2.150786	2.074492	2.096104
XGBoost	2.122015	2.023341	2.03813

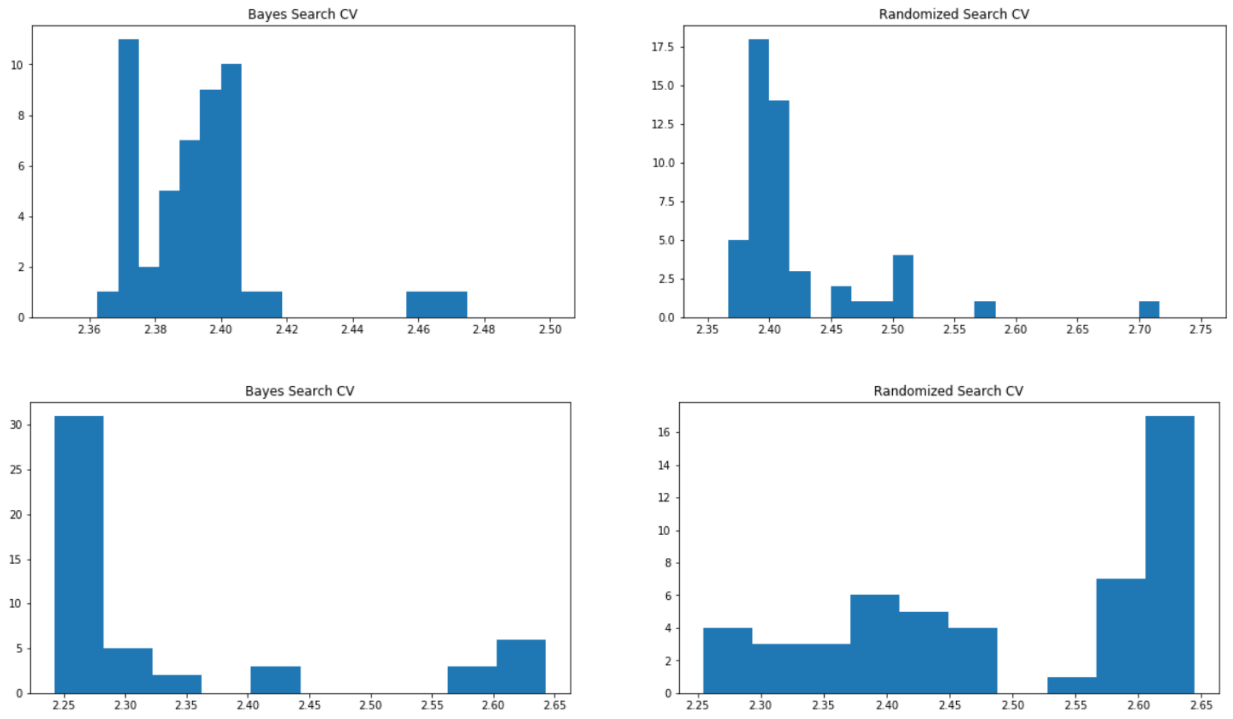
This also shows that tree-based methods generally work well in situations where there is a highly tabular dataset, especially if ensemble methods are used to combine multiple models in order to produce performant and generalized predictions.

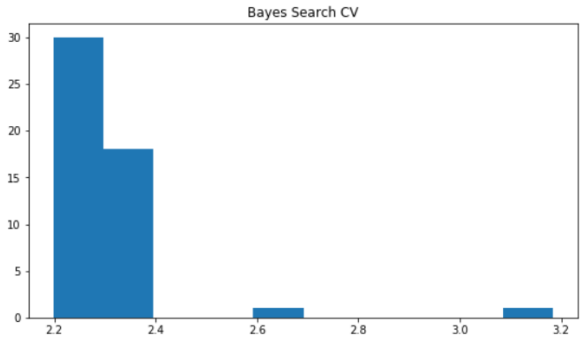
Additionally, an important feature transformation in the preprocessing process that may easily go unnoticed is using the log values of the budget and revenues. Before doing this, the error results that were obtained were around 3.5, while after this the drastic decrease above was seen. This is because the revenue has very large values compared to the other columns, therefore the differences in the predicted output and the actual output could be very large. This results in the models being unable to find good features to optimize, as a feature could produce bad results for just one type of input while working well for the rest, however this one input could have a very large revenue resulting in a very large error. Hence, it was very important for this project to spend a large amount of time experimenting with feature selection and preprocessing.

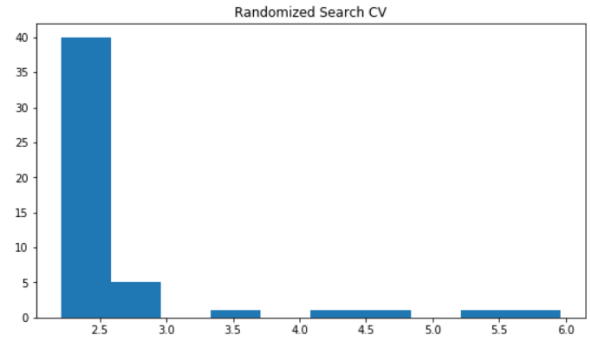
Bayes Search CV vs Randomized Search CV

As is seen in the results above, Bayes Search CV performs better when using random forest regression and XGBoost, while it is marginally outperformed by decision trees. This shows that in situations such as these Bayes Search performs better as it tries to approach a better score by taking previous error values into account, whereas Randomized Search CV randomly tries parameters with no regard for what kind of parameters worked better before.

The histograms below show mean error values for the cross validation runs for decision trees, random forest regressor and XGBoost respectively.







From the histograms it can be seen that the two methods have similar values in a lot of their runs, for Bayes Search CV, however, it seems that there are a greater number of values which have lower errors, which makes sense as it used previous data to try parameters similar to those with previous lower scores. This shows that it may be better at performing informed cross validation, however, it is possible that it loses out in terms of exploration since it may use parameters similar to each other which are a "local minimum" while Randomized Search may be better at getting out of a local minimum since it can result in drastically different values being tried.

Future Parameters

Some interesting parameters that can be tried if more data is available are:

Cast member pairs or trios that occur frequently, as movies made with a combination of good actors and directors generally tend to do well

Day of week of release, as consumers may make their decision to watch a movie based on what day of the week it is since consumers may be more willing to watch movies on the weekend.

These features were not used in this project as there is not a large amount of data available to test the behavior of these features, as they would probably work better when more data is available.

Computing Environment

OS: Windows 10 Home 64 Bit
OS Version: 17134
CPU: Intel Core i7-4720HQ
GPU: Nvidia GeForce GTX 960M
RAM: 16 GB
Programming Language: Python 3.7.1
Programming Environment: JupyterLab 0.35.3