Project Report

Group Members: Abhimanyu Swaroop (as6434), Karveandhan Palaisamy (kp2941), Wanru Hu(wh2483), Xiaorui Qin (xq2209), Binghong Yu (by2325)

1. Data Sources

Data comes from https://www.kaggle.com/c/tmdb-box-office-prediction/overview

The training dataset contains 7398 movies with 22 features some of which are cast, crew, plot keywords, budget, posters, release dates, languages, production companies, and countries. Some movies share the same titles while having unique ID thus should be treated as separate movies. This happens due to different movies having the same name or old movies having a remake. The dataset contains few missing values, and text data which require appropriate pre processing explained in the next section.

2. Data Preprocessing

To extract the most valuable information, we drop some columns, i.e., "id", "belongs_to_collection", "homepage", "tagline", "poster_path", "imdb_id", "original_title", and "title", in our dataset. Then we will introduce how we preprocess other columns to build the models.

2.1 Numerical Columns

For numerical columns "revenue" and "budget", we apply log transformation to them to make our models more robust to outliers. Missing values for "budget" and "runtime" are filled with the average value for movies.

2.2 Text Columns

Our data contains several columns with text data, e.g., "overview", "tagline", and "title". Most of them only provide a phrase with around 3 words for each movie, while the column "overview" contains long sentences with valuable information of descriptions for movies. We decided to choose this column as one feature in our models.

To preprocess the data in the column named "overview", we consider TF-IDF first. However, the vocabulary size is too large. We will get a lot of noise with tens of thousands of new columns of TF-IDF. Then we decided to use the Word2vec model. First, we construct a corpus with all of the sentences in the column named "overview" of the training set and the testing set. Then we use Word2vec to train word embeddings for all of the words in this corpus. Then we construct the "sentence embedding" for each overview of each movie. For each movie, we compute the average embedding of all words in the sentence of "overview" as the "overview embedding". In our experiment, we choose the dimension of embeddings as 20, and add 20 new columns to represent all of the dimensions of the overview embedding for movies.

2.3 Categorical Columns

In our dataset, there are many dictionary columns. For each dictionary of each movie, we convert it into a list of unique elements, and process these lists as categorical variables. The information of weekday, day, month, year are extracted from the column "release_date".

Up to now, we still have to process 10 categorical columns, i.e., “original_language”, “status”, “genres”, “production_countries”, “spoken_languages”, “Keywords”, “cast”, “crew”, “character”, and “production_companies”. We divide them into two groups.

First, for “Keywords”, “cast”, “crew”, “character”, and “production_companies”, “production_countries”, and “spoken_languages”, the numbers of unique elements are too large. For example, “Keywords” contains 11930 unique elements. If we simply use one-hot encoding and add one new column to represent the presence or absence of each unique element, there will be a lot of noise. To extract the most valuable information from each column, we choose the top 20 most frequent unique elements and use 20 new columns to represent the presence or absence of these 20 elements for all movies.

Second, for “original_language”, “status”, and “genres”, the numbers of unique elements are decent. For example, “genres” contains 20 unique elements. Except for in the case of LightGBM and CATBoost, we use one-hot encoding and add new columns whose size is equal to the number of unique elements to represent the presence or absence of each element.

2.4 Data Splitting

After encoding all of the categorical features, we split the dataset into training (60%), validation (20%), and test (20%) sets. Note that our file “test.csv” does not contain the target variable, i.e., “revenue”, we use data from the file “train.csv” to split data for model training and evaluation, and the column “revenue” constructs the target matrix y. Then we standardize the columns in the feature matrices X_train, X_val, and X_test to have zero mean and unit variance.

3. Models

Performance Overview for all models:

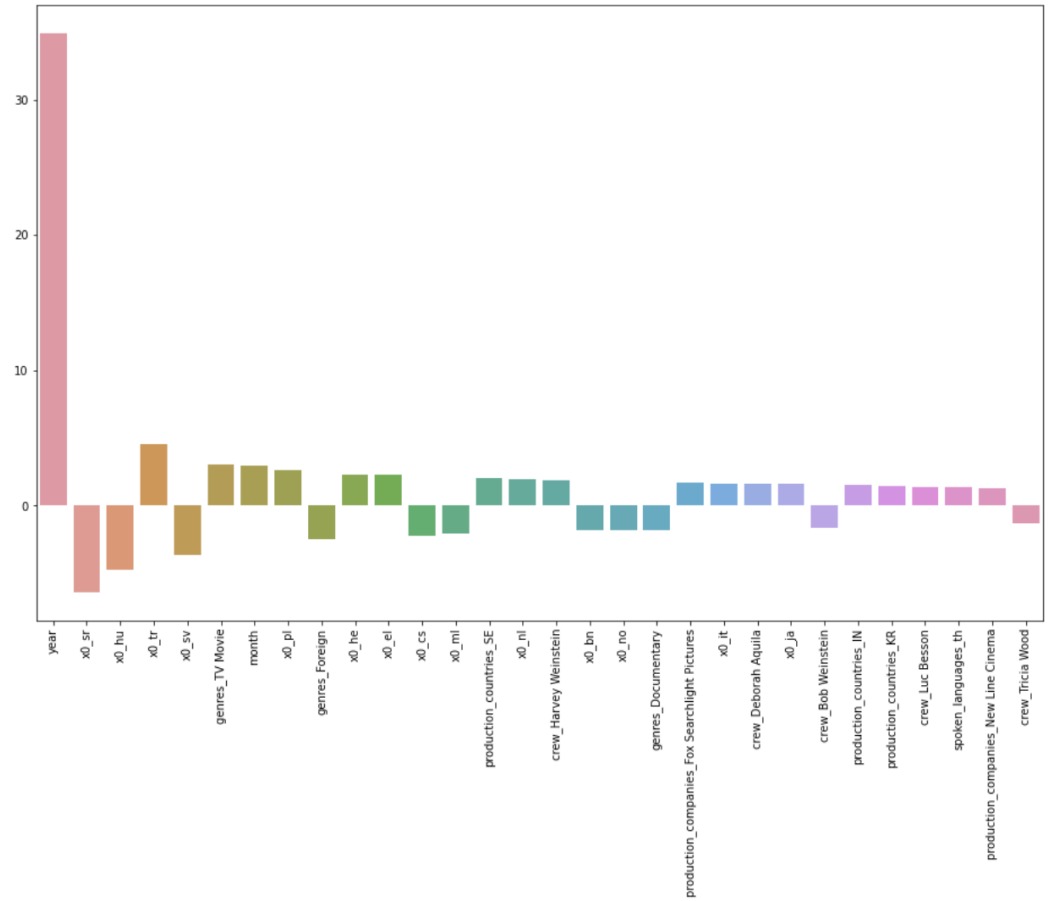
The scores that have been mentioned below are the log_rmse scores (RMSE calculated after taking log of the target variable).

Model	Train	Validation	Test (through kaggle)
Linear regression	2.1868411604703804	2.342614536999423	2.72587
Random forest	0.9884039588085319	2.0409173889641266	2.05486
Gradient boosting	1.2630344278884205	2.0550509145015794	2.09395
Hist Gradient boosting	1.0825357405711262	2.049200806590121	1.99825
XGboost	1.4440987790091433	2.0529415937843125	2.0837
Lgbm (all features)	1.005784057468253	2.127682790366256	2.04484
Lgbm (first 36 features)	1.0080675861900885	2.1251874072707118	2.10413
Cat (all features)	1.144144135795002	2.0831320679871856	2.05839

Cat (first 36)	1.2016415242630871	3.7860941879158836	2.10603
NN	2.763757030782901	2.768222144866415	2.79334

3.1 Linear regression

Through the second step of visualization with scatter plots, we check the linearity (the relationship between X and the mean of Y is linear), homoscedasticity (the variance of residual is the same for any value of X), independence (observations are independent of each other) and normality (for any fixed value of X, Y is normally distributed). Linear regression is our base model. The feature importance is plotted under. Here we didn't perform scaler on the data. Since after scaling, the rmse becomes really large (even goes to infinity). So here linear regression is not a suitable model.



3.2 Tree-based model

We also implement tree-based models including random forest, Gradient Boosting, Histogram Gradient boosting, XGBoost, LightGBM, and Catboost.

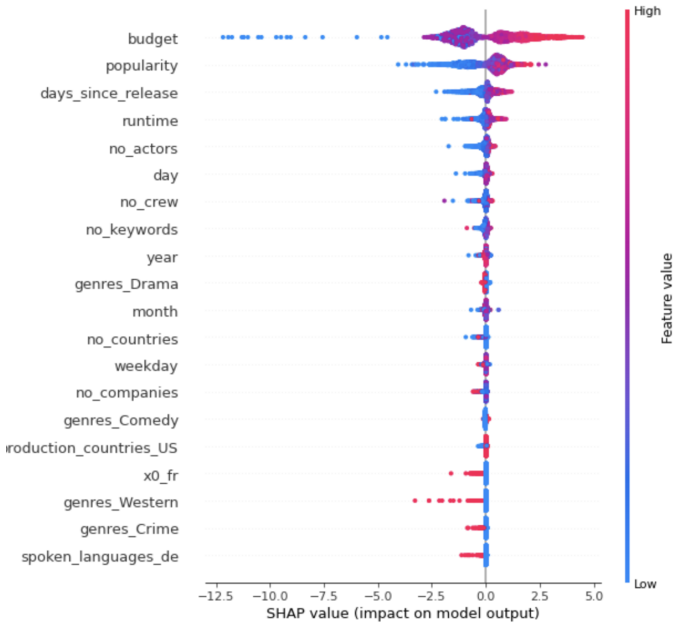
Even so, the performance of tree-based regression might not be as good as dealing with classification problems, as the way of splitting a node is through a loop through all possible node values.

We have two pipelines for tree based models. One of them is through loading data after one-hot encoding, and it is for those models who could not preprocess category data. While the other pipeline is through loading data without one-hot encoding. The second pipeline is mostly used on Catboost and lightGBM as they could handle the category encoding by themselves.

We did not use traditional grid search and cross validation methods. Instead, we set the k fold =10, and shuffle the index for 10 times. Then we select the train val data based on the index list. After this step, we fit the model with the shuffled dataset and append the result to list. This enabled us to create 10 models on different subsets of the dataset. The final prediction was made using the average of these 10 models to reduce the chances of overfitting in the dataset. The evaluation metrics we used in mse as we are dealing with regression problems. Test result score is based on submission to kaggle.

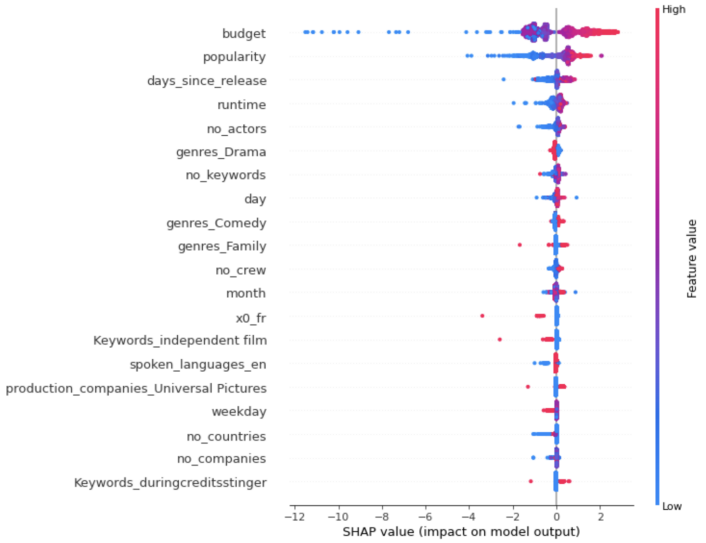
3.2.1 Random forest

We chose a random forest as the starting point of tree-based models. We performed one-hot encoding to the text data and then proceeded with model training. From the plot, we could see that budget, popularity are the most important features related to revenue. The model was trained with 100 different regression trees. From the result, the model performed very well with the training data set. While the model tested on the testing set, it produced a very poor score as there was overfitting. Keeping this model as a baseline for tree based models we tried working further on other models.



3.2.2 Gradient boosting

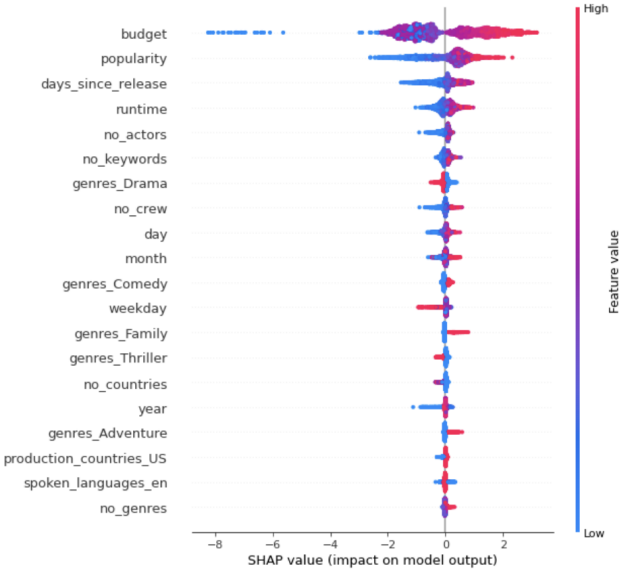
Gradient boosting did not give better performance than random forest. From the plot, we could see that budget, popularity are the most important features related to revenue. Unlike Random Forest, Gradient Boosting suggests that genre Family also results in more revenue and less importance is given to no.of crew members. Upon testing, the results in the kaggle were not as efficient as expected.



3.2.3 Histogram Gradient boosting

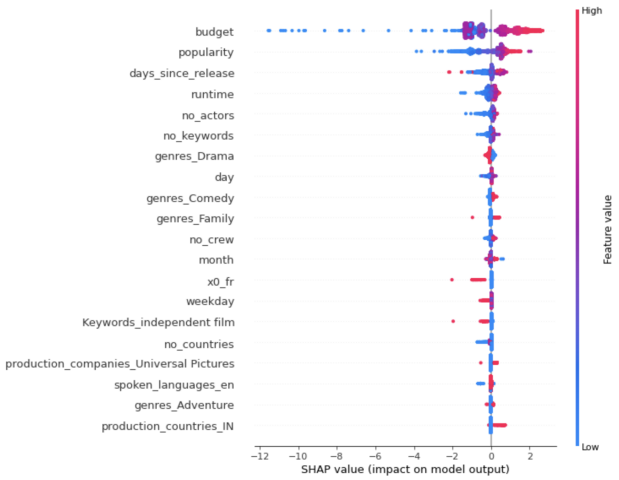
Histogram Gradient boosting performance and training speed is better than other models. And it has the best performance on the test set according to the Kaggle submission. Thus Histogram Gradient Boosting is the final model for this prediction.

In Histogram Gradient Boosting model, budget, popularity, days since release and runtime all have huge positive influence on the revenue. Also from the shap plot, it shows that audiences prefer Comedy, Family and Adventure movies more than Drama and Thriller movies.



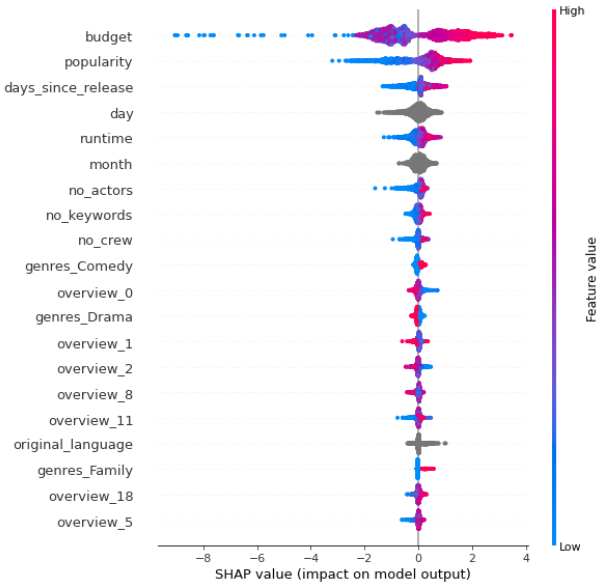
3.2.4 XGBoost

XGBoost also has decent performance on the test set. In XGBoost, the shap plot looks similar to the Histogram Gradient boost's shap plot. But different from Hist, original language - France value more and has a negative influence on the revenue.



3.2.5 LightGBM

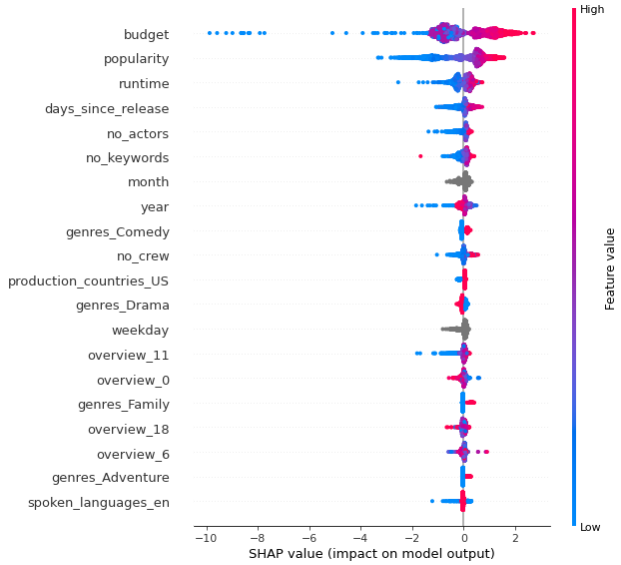
For the LightGBM we did not conduct one hot encoding to the categorical variables, instead set the variables to be "category" type and then used categorical variable hyperparameter which automatically preprocesses the categorical variables. It should be noted that day and month are grey in color because they are categorical variables.



3.2.6 Catboost

Similar to what was done for LightGBM, we did not conduct one hot encoding here. Instead, we created a list that specifies which columns are categorical and passed it into the CATBoost model which takes care of it on its on and is known to outperform simple one hot encoding. The color for month is grey

because it was a categorical variable.



The shap plot look quite similar for all tree-based models where budget, popularity, runtime, days_since_release are the most important features for predicting revenue.

3.3 Neural network

For the Neural Network, model performance is not as good as other models. The reason behind this is lacking of data. Neural network needs large amount of data for training, but train data set only contains 3000 records. So to avoid overfitting, here we only applied two hidden layers. The summary of Neural Network is shown as follows.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense (Dense)                (None, 64)                14912

 dense_1 (Dense)              (None, 32)                2080

 dense_2 (Dense)              (None, 1)                 33

=================================================================
Total params: 17,025
Trainable params: 17,025
Non-trainable params: 0
_________________________________________________________________
```

4. Conclusion

From the modeling attempts and SHAP analysis conducted above we can conclude that certain features such as budget has a significant impact on the prediction. We can also notice that a higher budget generally results in a higher revenue generated. It should also be noted that from the feature importance analysis, we can see that models more or less result in the same feature being important. In addition to the feature impotence being the same, the features that come out to be the most important make logical sense as well. For example a higher budget, days_since_release, popularity etc. are important features as one would intuitively think. High budgets are often associated with better scripts, screenplays, more famous

directors and famous actors, etc. On the other hand, less budgets generally tends to result in a lower revenue since the production quality and the quality of actors are affected. Days since release is also an important factor. For example, movie released last month might tend to have a higher revenue than movie released last month as audiences have more access to the movie released earlier. One other assumption is movie ticket is more expensive than ticket in the past as cost of maintaining a cinema increases.(but there is no evidence to support this.. Another assumption is that season also matters. For instance, during holidays such as Christmas, movie released during that time period tend to have higher revenue.