IEOR 242 Project Report - TMDB Movie Revenue Prediction

Anqi Xu[1], Yan Li[2], Boyu Yang[3], and and Charlotte Jin[4]

[1]anqi_xu@berkeley.edu
[2]yan_cc@berkeley.edu
[3]boyu_yang@berkeley.edu
[4]charlotte_jin@berkeley.edu

December 11, 2022

1 Motivation & Imapct

With the gradual popularity of the film industry, film directors and producers invest an increasing amount of money on movie production and later advertising promotion. Given that major films costing over $100 million to produce can still flop, how to predict the success of a movie before it is released is more important than ever to the industry. Can we predict which films will be highly rated, or earn high revenue? Can we find some consistent formulas to recommend to filmmakers to ensure commercial success after the release? These are the main problems we want to solve in our project.
By doing this project, multiple machine learning models can be built to accurately predict the box office revenue, which greatly informs the tough investment decision-making process when facing such a high-risk and high-yield opportunity in the film market. In addition, our prediction is highly important for advertisement companies that seek to embed their ads in popular movies. By mastering the key formula for film success, we can help film producers produce films that cater to the market and capital. Our prediction can also assist cinemas in scheduling movies and help people choose movies to watch.

2 Introduction to the dataset

Our project uses the Movies Daily Update Dataset generated from TMDB Dataset. The Movie Database (TMDB) is a popular, user-editable database for movies and TV shows. The Movie Details, Credits, and Keywords have been collected from the TMDB Open API. The dataset updates daily to ensure an updated movie dataset. By 11/21/2022, this dataset had 575748 rows, each representing information about one movie. Data columns include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages, reviews, and recommendations.

3 Data preprocessing and feature engineering

Data preprocessing is a step in the data mining and data analysis process that takes raw data and transforms it into a format that can be understood and analyzed by computers and machine learning.

3.1 Data cleansing

We found 90 duplicated1 rows, all of which were deleted to keep the latest dated row. We found many missing values in the columns genres, runtime, production-companies, tagline, etc. To ensure the integrity of the data, all rows containing missing values are deleted.
We found that there are rows with budget and revenue less than 0, which is not in line with common sense, so they are removed.

Removing outliers is an effective way to reduce the noise in data. For the three columns of runtime, popularity, and vote-count, we drew box plots for each of them, found the outliers, and removed them.

3.2 Feature Engineering

We found that the magnitudes of budget and revenue are too large compared to the other columns. To prevent the magnitudes from affecting the accuracy of the prediction model, we performed log transformations on these two columns. Logarithm naturally reduces the dynamic range of a variable so the differences are preserved while the scale is not that dramatically skewed.
We found that for the original language column, 3655 of all the 4291 movies are English. To explore the impact of language is English on film revenue, we created a binary column to represent whether the language of the movie is English

4 Modeling

4.1 Model Structure

In our project, the goal is to predict the movie revenue when giving information about the movie's budget, popularity, genres and credits related information. As a result, the dependent variable for our project is Log-revenue. After processing our row data and applying text analysis methods, we get 402 independent variables in total.

4.2 Baseline Model

Since every machine learning task is unique, it is always good to start small and then build on initial findings. A baseline model is essentially a simple model that acts as a reference in a machine learning project. Its main function is to contextualize the results of trained models. We treat the baseline model as a benchmark for other trained models that we propose.
In this case, we use the simple multi-variable linear regression as our baseline model. After training on the training set, the baseline model can achieve an OSR square of 0.785.

```
from sklearn.linear_model import LinearRegression
model1 = LinearRegression().fit(X_train, Y_train)
Y_pred1=model1.predict(X_test)

osr2_1=OSR2(Y_train,Y_test,Y_pred1)
print('The OSR^2 is: ',osr2_1)
```

```
The OSR^2 is:  0.7851254325758611
```

4.3 CART Regression

In order to increase the predictability, we use CART decision tree to train our model. CART is a predictive algorithm used in Machine learning and it explains how the dependent variable's values can be predicted based on other independent variables. In the decision tree, nodes are split into sub-nodes on the basis of a threshold value of an attribute. This continues till the last pure sub-set is found in the tree or the maximum number of leaves possible in that growing tree.
In order to get the optimal value for the hyper-parameter ccp-alpha in CART model, we conduct 5-fold cross validation. We test on ccp-alpha from value 0 to 0.2 and use the R square as the choosing score. The result is shown in figure 1 and the optimal value for ccp-alpha is 0.012.

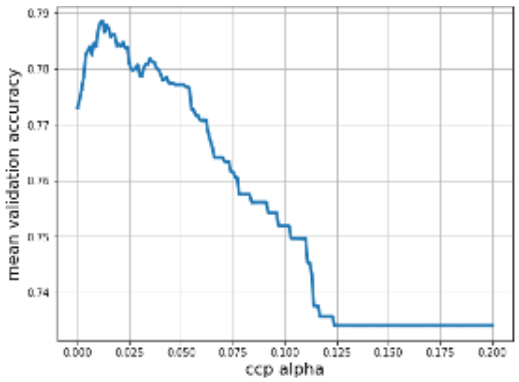

Figure 1: 5-fold cross validation for ccp-alpha

Using the optimal hyper-parameter, the CART regression model can achieve an OSR square of 0.785, which is the same as our baseline model.

```
from sklearn.tree import DecisionTreeRegressor
model2=DecisionTreeRegressor(ccp_alpha=0.012,
                             min_samples_leaf=5,
                             min_samples_split=20,
                             max_depth=30,
                             random_state=88)
model2.fit(X_train,Y_train)
Y_pred2=model2.predict(X_test)

osr2_2=OSR2(Y_train,Y_test,Y_pred2)
print('The OSR^2 is: ',osr2_2)
```

```
The OSR^2 is:  0.7849346935257493
```

4.4 Random Forest Model

In order to further increase the predictability of our model, we choose random forest model for this prediction problem. Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression, which combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. A Random Forest operates by constructing several shallow decision trees with different independent variables during training time and outputting the mean of the classes as the prediction of all trees.
Similarly, we apply 5-fold cross validation to choose the hyper-parameter max-feature from 0 to 402 to decide its optimal value. Using R square as the choosing score, the result is shown in figure 2. The optimal max-feature is around 130, which is just the turning point in the figure.

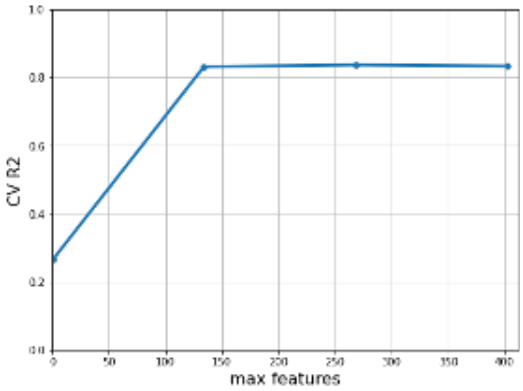

Figure 2: 5-fold cross validation for max-feature

Using the optimal hyper-parameters, we can achieve an OSR square of 0.832, which is a 4.7% increase in model performance.

```
from sklearn.ensemble import RandomForestClassifier
model3 = RandomForestRegressor(max_features=130,
                               min_samples_leaf=5,
                               n_estimators=500,
                               random_state=88)
model3.fit(X_train, Y_train)
Y_pred3=model3.predict(X_test)

osr2_3=OSR2(Y_train,Y_test,Y_pred3)
print('The OSR^2 is: ',osr2_3)
```

```
The OSR^2 is:  0.8316347860610183
```

4.5 Boosting Regression Model

Lastly, we apply boosting regression model to our prediction problem. "Boosting" in machine learning is a way of combining multiple simple models into a single composite model. Decision trees are used as the weak learners in gradient boosting. Gradient boosting Regression calculates the difference between the current prediction and the known correct independent variable value, which is the residual. Then, gradient boosting regression trains a weak model that to predict that residual. This residual predicted by a weak model is added to the existing model input and thus this process leads the model's prediction towards the true value. Repeating this step again and again improves the overall model prediction.
In our project, we choose a reasonably large hyper-parameter n-estimator to train the model. The OSR square can reach 0.823, which is similar to the random forest model.

```
model4_2 = GradientBoostingRegressor(n_estimators=10000, learning_rate= 0.001, max_leaf_nodes=3,
                                     max_depth=5, min_samples_leaf=10, random_state=88, verbose=1)
model4_2.fit(X_train, Y_train)
Y_pred4_2=model4_2.predict(X_test)
```

```
osr2_4_2=OSR2(Y_train,Y_test,Y_pred4_2)
print('The OSR^2 is: ',osr2_4_2)
```

```
The OSR^2 is:  0.8232500790503152
```

4.6 Feature Importance Analysis

Using the result from the random forest model, we analyze the feature importance. The top important features are shown in figure 3.

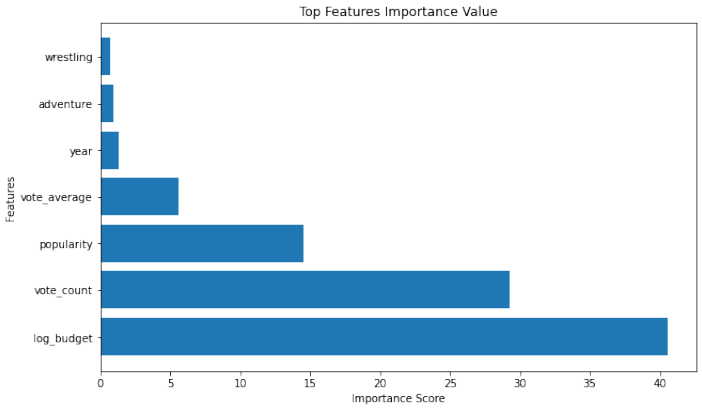

Figure 3: top feature importance analysis

From the above analysis result, we can find some very interesting conclusions. First, the movie budget has a really important effect on the final movie revenue. Besides, audients' voting rate and number also play very important roles in predicting the movie revenue. Last but not least, some genres, such as adventure and wrestling, have stronger effect on the final revenue.

5 Bootstrap Validation & Confidence Interval

Form the modeling and evaluation part, it's obvious that the random forest model is the best performing one. Therefore, we choose this model to carry out further validation through bootstrap method. By randomly select samples from the original test set with duplication for 5000 times, we get 5000 samples with the same size as the original test set and these would be used for our validation.
After calculating the OSR^2 on these 5000 samples, we fit the distribution of the 5000 OSR^2 we get and calculated the corresponding 95% confidence interval, which is $[0.77988502, 0.8716448]$. This interval implies that out model is good and reliable on the test set.

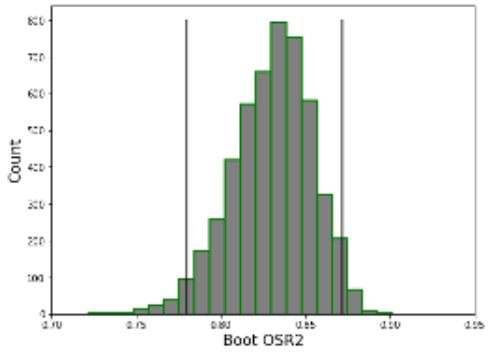

Figure 4: Distribution for Bootstrap OSR^2

Then with the similar method, we get the 95% confidence interval of the difference between the mean of the Bootstrap OSR^2 and the OSR^2 we have from the original test set. The interval is [-0.05174977, 0.04001001], implying that there is no significant difference between these two values. Additionally, because the interval is relatively symmetric, we can say that our selection of the test set is not biased.

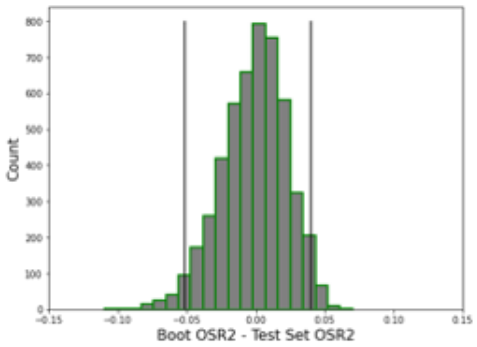

Figure 5: Distribution for Bootstrap OSR^2 – Test Set OSR^2

6 Summary & Suggestions

6.1 Summary

From all our modeling and validation process, we find that the Random Forest Model performs best on this data set and is suggested to be used for the movie revenue prediction. It performs better over the OLR, CART, and GBM models, and have the highest OSR^2 score, 0.832. Based on our bootstrap validation, the 95% confidence interval for the OSR^2 of this model is [0.77988502, 0.8716448]. This interval implies that out model is good and reliable on the test set.

6.2 Suggestions

From the feature importance score we can tell that the most important features for predicting the revenue of a movie are: budget, number of votes, popularity, vote average, and release year. It's interesting to find that these features can be obviously divided into 3 groups, which are: Investment (Budget), Audience Satisfaction Level (Number of votes, popularity, vote average), and Timing (Release year). Therefore, based on our research, we suggest that if a movie company wants to make a best seller, they should make sure that they have enough investments which is the most important factor. And before releasing the movie, they could carry out some sample preview to see the audience satisfaction level and decide whether the choose to out this film to a broader screening.