ORIE 5741 Final Project Report

Yihan Zhang (yz476), Chuntian Chi (cc2234), Shuyan Huang(sh2429)

Introduction

Movies are one of the most important forms of entertainment worldwide, and the market size is predicted to grow even further in the near future. However, whether a movie will be successful or not depends on a number of factors. Our group would like to dive into this field and try to predict whether a movie would be a commercial success given data on cast, crew, budget and so on by predicting the revenue a movie would make. We believe that this is important because a movie production usually costs hundreds of millions of dollars and a long period of time, and it would largely help the business company to determine if a movie would be a success and how much money it would possibly bring.

Dataset and Description

https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata

We intend to use the TMDb 5000 Movie Dataset which contains exhaustive information about films, including not only direct information about a movie like its genre and title, but also information about casts, crews, popularity, budget, production company and historical information of the production team. The wide variety of types of information guarantees the comprehension of our analysis.

Algorithm and Implementation

1. Data Processing

   Clear Empty Data

   Since we are predicting the overall revenue that a movie can make, we need to make sure that our dataset has all the necessary information about revenue and budget. If a row/movie does not have the information of its budget or revenue, we don't want it to interfere with our prediction, so we need to delete them. We did this by dropping all rows with budget = 0 or revenue = 0. After dealing with the empty data, our dataset still has about 3500 rows, and we think this would be enough for us to continue our modeling and predicting.

   Splitting into Train/Test Set

To make our model able to make predictions on unseen data, we split the dataset into the training set and the testing set. We chose the proportion of 8:2, so that our training set contains 2500+ rows while the testing set contains ~800 rows.

Boolean Data

For the column “homepage”, we found it good to transform the data into boolean value, since a movie’s homepage link itself does not seem to affect the popularity of a movie, while it is possible that a more successful movie could contain a homepage while some non-famous movie would not bother to have a homepage. We first developed a helper function that transformed the input string into a boolean value, and then we used the map function to apply the helper function to every row in the dataset.

Real-number Data

For the column “tagline”, “budget”, “popularity”, “runtime'”, and “vote_average”, we decided to transform them into real-number data.

For “tagline”, since there are thousands of thousands different taglines in total and the repetition rate (that a movie will contains a same tag as the other) is small, we think it will not be able to correctly reflect reliable information about the movie if we turns it into many-hot data, so we just evaluate the length of it. Other selected columns are themselves real-value data, and we just need to convert string into float. Here we do the same thing as we did in boolean data and map the transformation helper function to each row.

One-hot Data

For the “original language” column, since every movie would only have one of the languages, we decide to use one-hot representation. We use the set of original languages present in the training set and transform the data into a one-hot array based on that. It’s possible that in test data we may encounter a movie that has the original language that is different from any languages present in the training set. In this case, the one-hot array of it would just be an array of 0.

Many-hot Data

For the columns like “production company”, “production countries” and “spoken languages”, we decided to turn these data into many-hot arrays since every movie may have many of each feature. Meanwhile, for production companies, since there are a lot of companies that could make the many-hot array extremely long, we decided to only account for companies that occur in the training set more than 3 times, and remove all others.

Offset

Finally, we combine all the existing features into a large n-d array, and then we add offset, which is a column of one's to the features array to build the final feature set.

Discrete Target

The original target revenue is continuous data, so we have to use a regression model. In order to use a classification model, we converted the revenue into discrete data by changing a number to its number of power of 10. For example, 1000 into 3 and 10000 into 5.

2. Data Visualization

   We first explored correlations between each pair of features. The result can be seen in Fig 1.

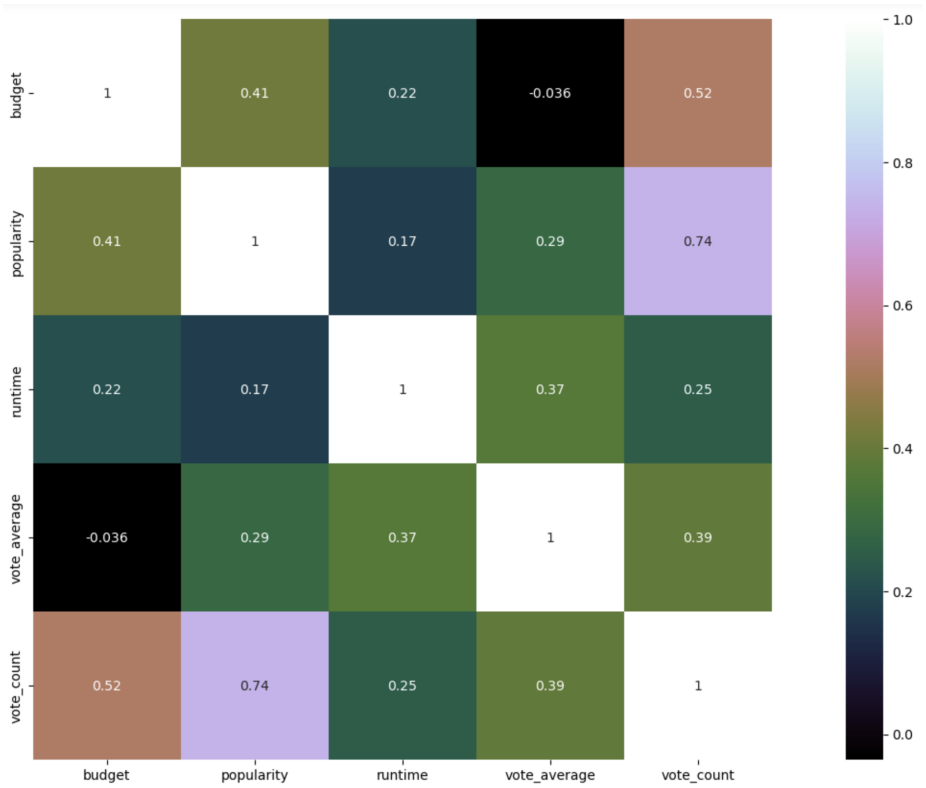

Fig1. Correlation Visualization

There is a strong positive correlation between popularity and vote_count. So, we took a closer look at these two data as in Fig 2.

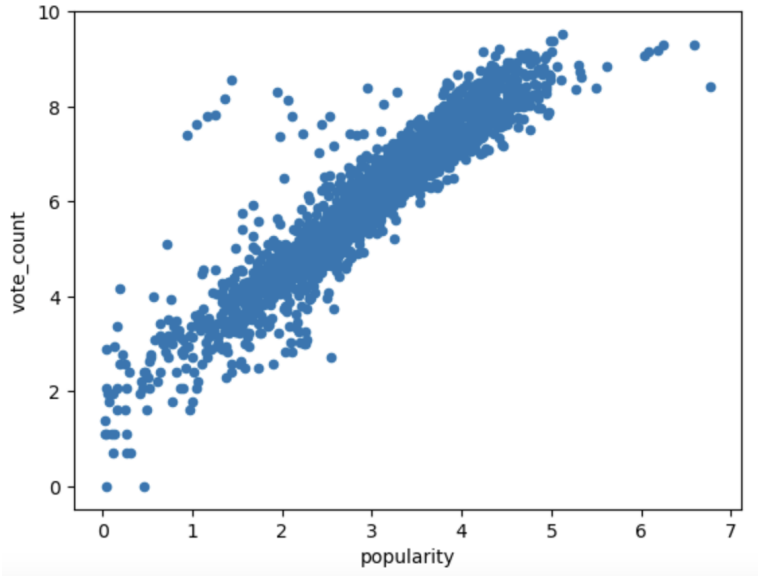

Fig 2. Vote_count vs. Popularity

There is indeed a positive correlation between these two features. It seems that vote_count could be a good indicator of movie popularity, which could be directly related to its revenue.

PCA

We tried to extract a subset of essential components in features that best explains variance in the data, so we tried the PCA method.

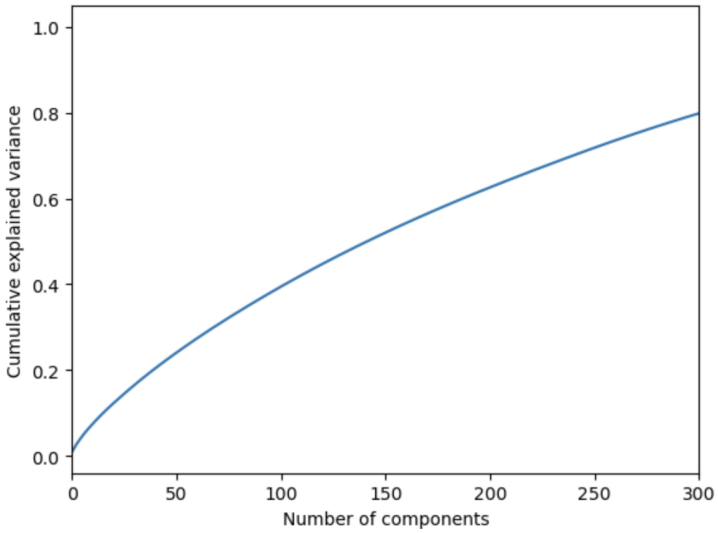

Fig 3. PCA Result

However, as shown in Fig 3., we need around 300 components to explain 70% - 80% of the data, which is not an ideal choice. Hence, we trained our model using all features.

3. Training and Testing

We tried different models using our pre-processed data and compared their results:

Linear Model

The first and most simple model we tried is the linear model. We used the same function as homework 3, which runs OLS and bypasses any SVD convergence errors by refitting the model. We also used MSE to calculate the mean squared error.

Neural Network

We used MLPClassifier and MLPRegressor respectively. When using MLPRegressor, we used the original target, which contains continuous values. When using MLPClassifier, we used the target converted into discrete values. To avoid overfitting, we restricted the number of hidden layers to (20, 20).

SVM

We used a linear SVM model with a stochastic gradient descent algorithm.

KNN

We tried the KNN classifier in order to obtain high accuracy, and maybe capture some underlying nonlinear trends of the data.

Decision Tree

We tried the decision tree classifier as it handles both numerical and categorical data.

Random Forest

Random forest, which is based on decision trees, is used. It balances the error rooted from the imbalance of the dataset, and works well on nonlinear data, which is what we believed to be the underlying trends of our data. Grid search is used to find the best hyper parameters used for the model (min_sample_leaf = 1, min_sample_split = 2, estimators = 40).

Ridge Regression

We then used ridge regression, which is an extension to linear regression to minimize model complexity because the nonlinear and more complex models we tried all seem to result in suboptimal results.

Results and Discussion

We got different results for different models we tried, the best model is ridge regression, which achieved 0.80 training accuracy and 0.76 testing accuracy.

Linear Model

Linear models didn't give us good results. The MSE it produced was extremely high: about 10e15. Therefore, we're not confident about this model. We believe that this is because our dataset is not linearly separable, and therefore should use some other nonlinear models.

Neural Network

The MLP model gave us much better results than the linear one. To be more specific, the MLPClassifier reached 0.49 training accuracy and 0.46 testing accuracy. And the MLPRegressor also reached 0.48 training accuracy and 0.54 testing accuracy. So, we're not very confident about this model. Even though it's already better than linear, this model still isn't ideal and thus we need to try some other methods.

SVM

The SVM model gave really bad results with a training accuracy of -0.125 and testing accuracy of 0.129. This means that this model does not follow the trends of our data at all.

KNN

The KNN model resulted in comparable performance to the MLPRegressor but has a higher training accuracy of 0.72. The testing accuracy is 0.57, which is basically the same as the MLPRegressor. More complex model is used to improve the accuracy.

Decision Tree

The decision tree model has poorer performance than that of the KNN and MLPRegressor models. Its training accuracy is 0.37 and its testing accuracy is 0.45, which is similar to the MLPClassifier. This seems like our goal is more suitable as a regression problem than a categorical classification problem.

Random Forest

The random forest model overfits the data, which in hindsight is understandable given our parameter choice. But trying different combinations of parameters did not resolve the problem. The training accuracy is 0.83, which is by far the best in our models. However, the testing accuracy is -0.11. It overfits really bad.

Ridge Regression

Surprisingly, ridge regression results in the best performance. It has a training accuracy of 0.81 and a testing accuracy of 0.76.

The residual graph for each model is shown in Fig 4.

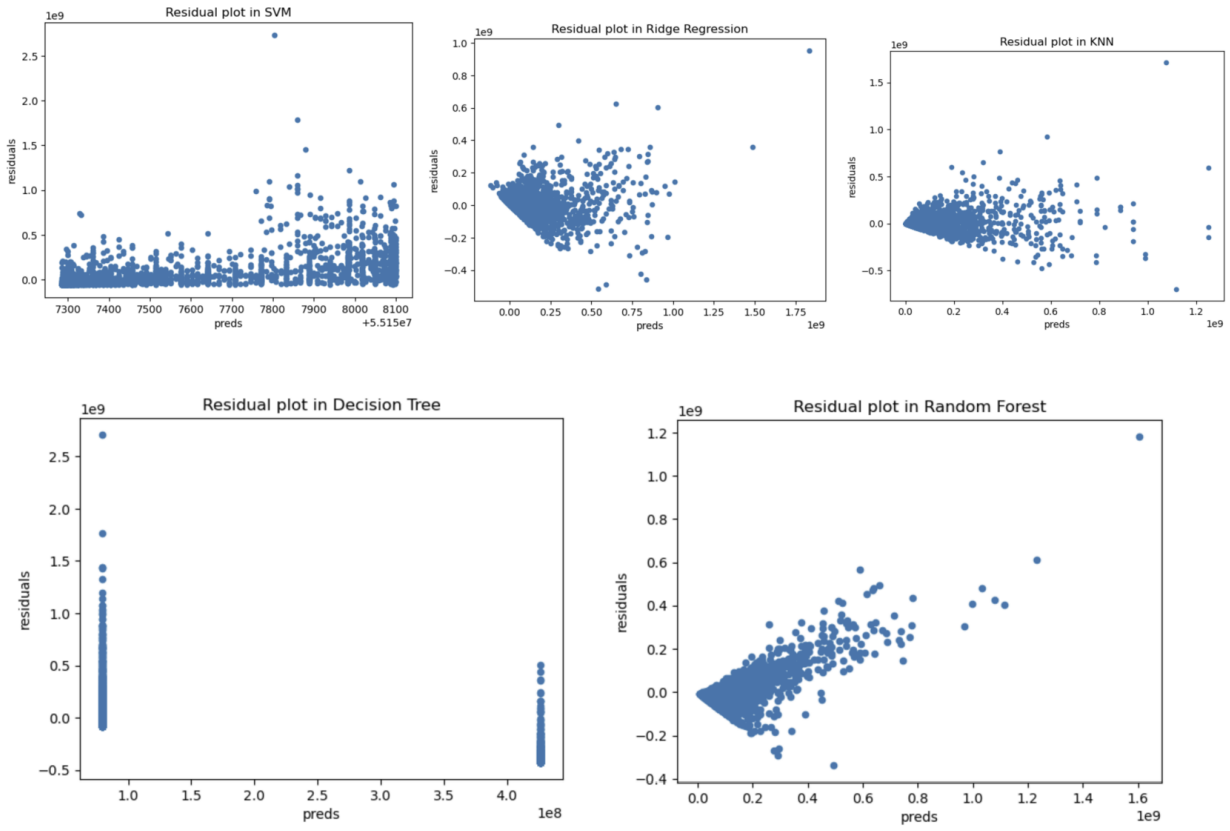

Fig 4. Residual Graphs

In conclusion, we are confident about the performance of the ridge regression model, and to some extent also the performance of the KNN model. We believe that the SVM and random forest model should not be trusted in making predictions about movie revenues.

Discussion

We would be very willing to use the data and model we generated in production to change how our company makes decisions, because our prediction is generally accurate and could reflect to some extent what kinds of movies the market likes. However, there are still plenty of other influencing factors that we shall need to take into account, and this model would just be a good reference but should not impact our decision as the main factor.

Our model will not produce a Weapon of Math Destruction, but it's true that we need to take into consideration fairness. Our dataset only contains the data from a particular dataset and it might be limited or biased. For example, the movie dataset has a more focus on US audience, and also it does not contain the data of more recent movies.

Conclusion

To conclude, we used movie data to predict its revenue by using different techniques taught in class. For example, one-hot and many-hot in data pre-processing, SVD data convergence, ridge regression, and random forest. We reached a final accuracy of about 0.76 and we are confident about our result.

Work Contribution

For this project, Yihan was mainly responsible for data processing including feature extraction and transformation. Shuyan aided Yihan for data processing, and implemented the linear model and MLP neural network model. Chuntian was responsible for other training techniques such as KNN, random forest, SVM. The report is also equally divided among the three people, with each person writing the corresponding section.