ORIE5750: Final Report

An Application of Machine Learning to Predict Rating of Movies

Applied Machine Learning
Cornell University - Cornell Tech

Xianglin Chen, Anlin Dai
ID: xc352; ad848

December 12, 2022

ABSTRACT

Nowadays, movies are becoming a more and more important entertainment in people's lives and the rating can be easily shared on the internet. Predicting movie user ratings automatically is specifically valuable for prediction box office gross in the cinema sector. Also, people often look at ratings to get an idea of how good movies are and to decide what movies to watch or which ones to prioritize. As a result, movie rating prediction has been a popular application area for machine learning researchers. Our project aims to predict movie ratings of new movies based on various features, such as budget, release year, genres, and plot. We build our three ML models, including Logistic Regression model, SVM model, and Random Forest model to predict which movies will get high ratings.

1 INTRODUCTION

What can we say about the success of a movie before it is released? Has any production company found the secret to high ratings? Given that major films costing over $100 million to produce can still flop, this question is more important to the industry. Film aficionados might have different interests. Can we predict which films will be highly rated, whether or not they are a commercial success?

We think that this is a great place to start digging into those questions, with data on the budget, revenues, genres, popularity, spoken languages, release date, actors, directors, production countries and production companies of several thousand films. Our project is an experiment which aims to predict the rating of upcoming movies through various machine learning models.

2 BACKGROUND

2.1 Dataset

Our project is mainly based on the Kaggle dataset TMDB 5000 Movie Dataset, which consists of 4803 movies information. We split our data into a 75% training set and 25% test set.

We selected our dataset due to its relatively large size, useful and well organized information, and accurate number. After investigating the dataset, we featured some of the major characteristics of the dataset, which will likely affect our choices of preprocessing techniques and machine learning.

2.2 Models based on Previous Work

Previous work done on movie rating prediction are usually based on machine learning models such as Logistic Regression, K-means clustering Algorithm, Stochastic Gradient Descent Algorithm and Neural Networks.

3 METHOD

3.1 Data Pre-processing

During the stage of preprocessing before training the models, we cleaned and decided on features (X) for both categorical and numeric variables. We dropped columns, like 'homepage', 'id', 'original_title', 'status', 'title', 'tagline'. For categorical variables, since one movie can have multiple genres and production companies, we applied one hot encoding method to these two columns and created a vector with 20 new dummy variable columns for genres and more than 3,000 columns for different production companies. Due to a large number of counties, we transferred the production countries column to if the movie is produced by the United States. We also count the number of spoken languages for each movie, which means making categorical variables into numeric variables. For numeric variables, we normalized budget

and revenue with a scale of one hundred million dollars. We also calculated the time difference between the release date and the current year (2022) to make numbers easier to classify with higher accuracy.

Finally, by observing the distribution of voting scores, we found that most of the voting scores (y) are in the range between 5 to 7 which may cause bias in the final evaluation of a movie. Therefore, we transferred the rating scores column to two classes to decide if a movie is good based on the average score which is 6.09. The movie with a voting score over 6.09 will be classified as 1(good movies) and otherwise will be 0 (bad movies).

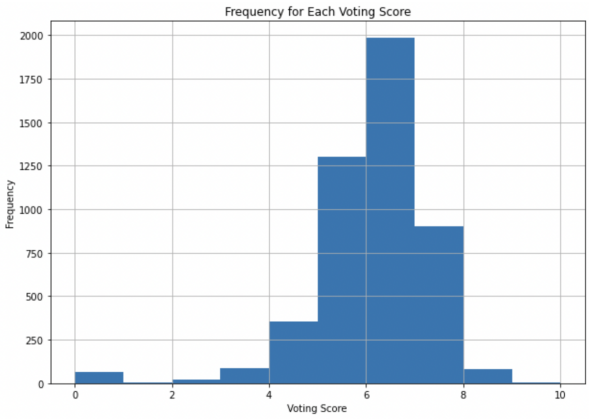


Figure 3.1.1 Frequency for Each Voting Score

Our baseline is the Logistic Regression Model when comparing the SVM model and Random Forest model to see which one is more suitable for the movie rating prediction. In this project, 25% of rows from cleaned data as a testing group for the three models. Through multiple tests with the algorithms, we dropped columns in the original data such as vote count, run time, and status which largely and negatively affected performance.

3.2 Baseline: Logistic Regression

Logistic regression is a supervised learning binary classification algorithm. It produces linear decision boundaries. Although regression is known to provide superior results when dealing with numerical data, it also allows for the prediction of discrete variables using the combined values of continuous and discrete predictors.

$$f_{\theta}(x) \ = \sigma(\theta^{T}x) \ = \frac{1}{1+exp(-\theta^{T}x)}$$

We framed the film rating prediction task as a classification problem. We discretized movie ratings uniformly into 2 classes, and chose a central value 60% for each interval to represent each class. We then applied a random forest classification model to the data using the same parameter settings as in regression. In the experimental analysis, we will try to turn the model with or without L2 regularization which in another word, Ridge Regression. Ridge Regression adds the “squared magnitude” of coefficients as a penalty term to the loss function.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

3.3 Benchmark: Support Vector Machine (SVM)

We also adopt the support vector machine (SVM) to compare with our classifiers. It is commonly used in many machine learning problems. SVM defines a margin from the hyperplane, where points inside the margin incur loss, while Support Vector Regression (SVR) defines a margin of ε-distance from the hyperplane such that data points inside the boundary are error-free. For our project, we used the radial basis function as the kernel.

$$f_\theta(x) = \sum_{j=0}^{p} \theta_p x^p = \theta^T \phi(x) \qquad \textit{Kernel}$$

$$K(x, y) = (x^T y) \qquad \textit{Linear kernel}$$

$$K(x, y) = (x^T y + c)^d \qquad \textit{Polynomial kernel}$$

$$K(x, y) = exp\left(\frac{||x-y||^2}{2\sigma^2}\right) \qquad \textit{Radial Basis Function}$$

3.4 Benchmark: Random Forest

As the SVM can only separate data linearly using a flat line or hyperplane, separating nonlinear data isn't directly possible. A nonlinear model could provide a better fit to our data, so we also experimented with decision trees and Random Forest, a supervised learning method for classification, regression, and other problems that works by constructing multiple decision trees when training. For classification problems, the Random Forest output is the class selected by the majority of trees. A decision tree contains internal nodes corresponding to input features, and leaves, which represent the output value following certain paths from the root. Random Forest utilizes a bagging strategy by splitting input features into a random subset while selecting the best feature for each node of a decision tree. It has strong stability and compensates for the decision trees' tendency to overfit their training set.

4 EXPERIMENTAL ANALYSIS

4.1 First Try

It is important to mention before processing the whole process of experimental analysis that we have tried to increase the number of training rows to avoid the models learning unexpected noise and extra information from the training dataset, or we can say, less overfitting problems.

	Accuracy	
%	Training	Testing
0.20	0.83	0.71
0.25	0.81	0.74
0.30	0.83	0.74
0.35	0.83	0.72
0.40	0.83	0.72
0.45	0.83	0.71

Tabel 4.1.1 Accuracy Scores with Different Percentage of Testing Dataset

Since our baseline model is Logistic Regression, we tested with the percentage between 0.20 to 0.40 with a gap of 0.05 to see the best amount of data to achieve the highest validation accuracy and the least

differences in training and validation scores. The result is shown in Table 4.1.1. In the end, as mentioned in 3.1, due to the trade-off between prediction validation accuracy and the overfitting problem, we decided to assign 25% of the dataset as the testing dataset and 75% as the training dataset. Since we need to control the baseline and benchmarks under the same condition to compare, we continued to use the same 25% of the original dataset as the testing group in the following benchmarks.

4.2 Baseline - Logistic Regression

We tested the Logistic Regression model with different parameters of "kernel", which means the regularization method of Logistic Regression, as none or L2 regularization. Figure 4.2.1 and Figure 4.2.2 in the appendix show that Logistic regression without any regularization achieved an accuracy rate of 0.81 for predicting training data and 0.74 for testing data. Similarly, the F1 scores for good movies in training (0.81) will be much higher than in the testing dataset (0.74). It means that the prediction using Logistic Regression is slightly overfitting for the training dataset. In addition, both classification reports show that the weighted precision and recall scores are nearly the same, which is about 0.81. But the Logistic model performs slightly better in good movies than in bad movies based on higher precision and recall scores than the bad movies.

By adding L2 regularization, the issue with overfitting didn't get better and there was a small improvement in overall performance for the model. Specifically, the weighted recall and weighted precision scores are slightly better (0.01) than the model with non-regularization. But the lLogistic Regression with L2 regularization has the same validation accuracy as the model without any regularization method. We then continue to use Logistic Regression with L2 regularization to compare with the rest of the benchmarks.

4.3 Benchmark: Support Vector Machine (SVM)

The Support Vector Machine model with Gaussian Kernel Radial Basis Function (RBF) kernel solves most of the overfitting problem when the accuracy scores and F1 scores for the training and testing dataset are close to each other. But it has a much lower validation score than the performance of Logistic Regression with L2 regularization – about 0.68 for validation accuracy and 0.69 for the F1 score shown in Figure 4.3.1. The polynomial kernel has a similar result to the RBF kernel. Its accuracy score and F1 score are much lower. Therefore, we were more focused on the linear kernel which seems to have better performance. The accuracy for the Support Vector Machine Model with a linear kernel has reached 0.92 for its training accuracy when the validation accuracy is 0.73 which is nearly the same as the baseline. But it also means that the model with a linear kernel learns too much noise and inaccurate data entities from the training dataset so the model is overfitting to the training dataset.

4.4 Benchmark: Random Forest Classifier

For the Random Forest classifier, we tried to adjust the number of trees and depth of trees to achieve the best performance from the model. To find the best combination of these two parameters, we set 25 combinations by using loops in a matrix of ([1, 5, 10, 50, 100]). For example, our testing model can be 1 tree with a depth of 1, 5, 10, 50, or 100, and so on. As shown in Figure 4.4.2, we found that the validation accuracy for depth in range 1 to 10 would exponentially increase and stay constant after. On the other hand, except for one tree, the increasing number of trees does not affect the performance.

By comparing the accuracy difference in Figure 4.4.3, we decided to analyze the Random Forest model with 100 trees and 50 max depth, and it has the highest validation accuracy and the least difference to deal with the overfitting problem. As the classification reports attached show, we surprisingly found that the performance for both Random Forest and Logistic Regression model (with L2 regularization) are nearly the same, but this model has a high possibility of overfitting the training data since it has an training

accuracy of 1.0. Since this model has a long running time due to its large number of trees and depth, we then chose to additionally analyze another combination of parameters – 50 trees and a depth of 5. Although this combination has a lower accuracy (0.71) than the previous combination, it largely reduced the running time and there's no overfitting problem with only a 0.05 difference in accuracy scores for training and testing. By analyzing the classification report, we also proved that this combination also has a good performance when it has close scores in precision and recall with the Logistic Regression model, which is about 0.71. It means that a Random Forest model with 50 trees and a depth of 5 has nearly the same performance as the Logistical Regression.

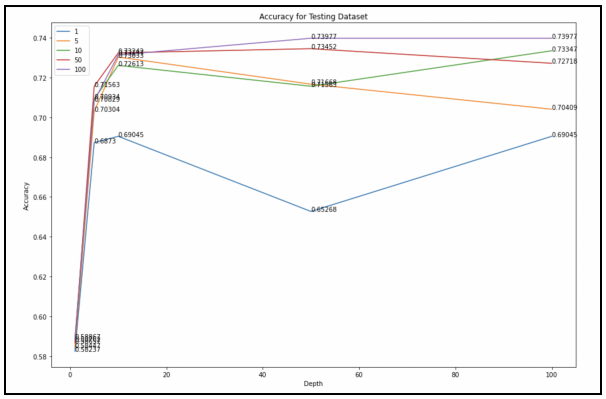


Figure 4.4.2 Accuracy for Testing Dataset

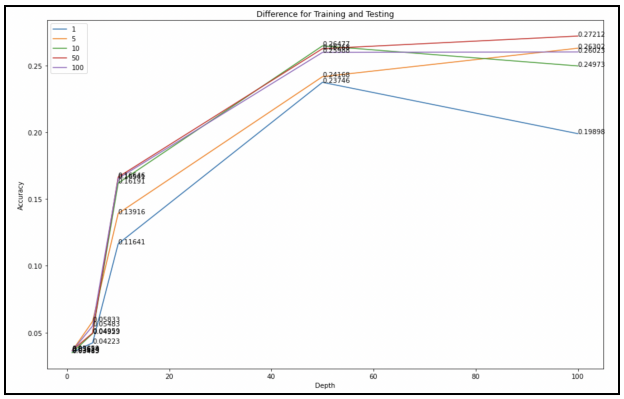


Figure 4.4.3 Difference for Training and Testing

5 DISCUSSION AND PRIOR WORK

In this project, we compared the three models that we tried – Logistic Regression, SVM and Random Forest models. Firstly, we focus on the test data. We compare the prediction accuracy of the three models and find that our baseline has the highest accuracy. Then, we check whether the models are overfitting or not, which means that we need to compare the difference between the *prediction_train* and the *predict_test*. The model is more likely to be overfitting if there is more difference between two values. To this point, the SVM model has the worst performance and thus we are not going to use this model for prediction. For the Random Forest model, we first choose the model with 100 trees and a depth of 5 as it performs better than the others. Since it also shows the huge difference between the prediction of the train and the test, we would not recommend using this model for rating prediction. However, we also found the Random Forest model with 50 trees and a depth of 5 has nearly the same performance and less running time, and no overfitting problem. Overall, by carefully evaluating the performance in accuracy, precision, recall, problem in overfitting, and running time, the Logistic Regression Model which is the baseline, will be the best choice for movies' rating prediction.

6 CONCLUSION AND FUTURE WORK

In the project, we applied three main machine-learning methods to the same dataset to classify the rating. With our baseline model which has the best performance, the industry could achieve an accuracy of 0.74 to predict the movie ratings. The results of this work can be used to recommend movies to users in web applications based on predicting the user's rating of a particular movie. It can also provide some business decisions for investors and sponsors, reducing the risk of loss costs. To improve, we should expand our dataset since currently we only have 4,803 rows which is a small scale to train. Then, we could have more analysis of running time and learning rate to improve the performance. Also, to try against our basic baseline method, we can try to tune more parameters in the models and choose more benchmarks such as deep learning models to compare.

7 SUPPLEMENTARY FIGURES AND REFERENCE

7.1 Reference

"TMDB 5000 Movie Dataset". Kaggle.

https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata?datasetId=138&sortBy=voteCount&searchQuery=regression.

Movie Rating Prediction Using the MovieLens Dataset, Yashodhan Karandikar, https://cseweb.ucsd.edu/classes/wi15/cse255-a/reports/wi15/Yashodhan_Karandikar.pdf.

A. Ö. Eren and M. Sert, "Movie rating prediction using ensemble learning and mixed type attributes," 2017 25th Signal Processing and Communications Applications Conference (SIU), 2017, pp. 1-4, doi: 10.1109/SIU.2017.7960604.

Movie Rating Prediction Using Ensemble Learning Algorithms, Zahabiya Mhowwala1, A. Razia Sulthana2, Sujala D. Shetty3, 8 Nov. 2020, https://thesai.org/Downloads/Volume11No8/Paper_49-Movie_Rating_Prediction.pdf.

Predicting IMDb Movie Ratings Using Supervised Machine Learning, Joe Cowell, 14 Oct. 2020, https://towardsdatascience.com/predicting-imdb-movie-ratings-using-supervised-machine-learning-f3b126ab2ddb.

7.2 Tables and Figures

	Accuracy	
%	Training	Testing
0.20	0.83	0.71
0.25	0.81	0.74
0.30	0.83	0.74
0.35	0.83	0.72
0.40	0.83	0.72
0.45	0.83	0.71

Tabel 4.1.1 Accuracy Scores with Different Percentage of Testing Dataset

	precision	recall	f1-score	support
0	0.79	0.77	0.78	1221
1	0.83	0.85	0.84	1637
accuracy			0.81	2858
macro avg	0.81	0.81	0.81	2858
weighted avg	0.81	0.81	0.81	2858

4.2.1 Classification report for Train Dataset using Logistic Regression without penalty

	precision	recall	f1-score	support
0	0.71	0.67	0.69	412
1	0.76	0.79	0.77	541
accuracy			0.74	953
macro avg	0.73	0.73	0.73	953
weighted avg	0.74	0.74	0.74	953

4.2.2 Classification report for Train Dataset using Logistic Regression without penalty

	precision	recall	f1-score	support
0	0.79	0.78	0.78	1221
1	0.84	0.85	0.84	1637
accuracy			0.82	2858
macro avg	0.81	0.81	0.81	2858
weighted avg	0.82	0.82	0.82	2858

4.2.3 Classification report for Train Dataset using Logistic Regression with penalty = L2

	precision	recall	f1-score	support
0	0.71	0.67	0.69	412
1	0.76	0.79	0.77	541
accuracy			0.74	953
macro avg	0.73	0.73	0.73	953
weighted avg	0.74	0.74	0.74	953

4.2.4 Classification report for Test Dataset using Logistic Regression with penalty = L2

	precision	recall	f1-score	support
0	0.62	0.66	0.64	1221
1	0.74	0.70	0.72	1637
accuracy			0.68	2858
macro avg	0.68	0.68	0.68	2858
weighted avg	0.69	0.68	0.69	2858

4.3.1 Classification report for Train Dataset using Support Vector Machine with kernel is default = "rbf"

	precision	recall	f1-score	support
0	0.59	0.63	0.61	412
1	0.70	0.67	0.68	541
accuracy			0.65	953
macro avg	0.65	0.65	0.65	953
weighted avg	0.65	0.65	0.65	953

4.3.2 Classification report for Test Dataset using Support Vector Machine with kernel is default = "rbf"

	precision	recall	f1-score	support
0	0.91	0.90	0.90	1221
1	0.92	0.94	0.93	1637
accuracy			0.92	2858
macro avg	0.92	0.92	0.92	2858
weighted avg	0.92	0.92	0.92	2858

4.3.3 Classification report for Train Dataset using Support Vector Machine with kernel is linear

	precision	recall	f1-score	support
0	0.69	0.66	0.68	412
1	0.75	0.78	0.76	541
accuracy			0.73	953
macro avg	0.72	0.72	0.72	953
weighted avg	0.73	0.73	0.73	953

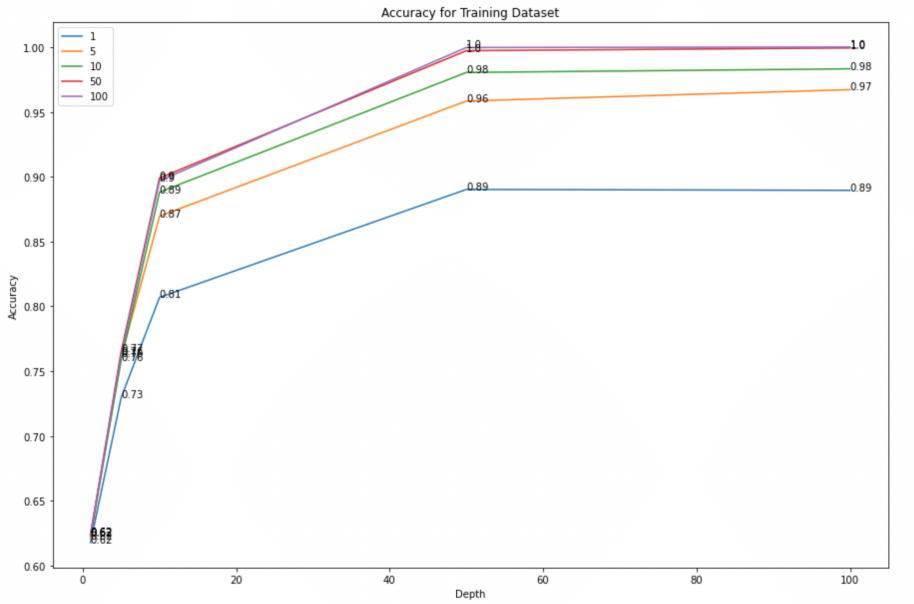
4.3.4 Classification report for Test Dataset using Support Vector Machine with kernel is linear

	precision	recall	f1-score	support
0	0.56	0.83	0.67	1221
1	0.80	0.51	0.63	1637
accuracy			0.65	2858
macro avg	0.68	0.67	0.65	2858
weighted avg	0.70	0.65	0.64	2858

4.3.5 Classification report for Train Dataset using Support Vector Machine with kernel is polynomial

	precision	recall	f1-score	support
0	0.54	0.79	0.65	412
1	0.76	0.49	0.60	541
accuracy			0.62	953
macro avg	0.65	0.64	0.62	953
weighted avg	0.67	0.62	0.62	953

4.3.6 Classification report for Test Dataset using Support Vector Machine with kernel is polynomial



4.4.1 Accuracy for Training Dataset

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1221
1	1.00	1.00	1.00	1637
accuracy			1.00	2858
macro avg	1.00	1.00	1.00	2858
weighted avg	1.00	1.00	1.00	2858

4.4.4 Classification report for Train Dataset using Random forest with number of tree = 100, max depth = 50

	precision	recall	f1-score	support
0	0.71	0.67	0.69	412
1	0.76	0.79	0.77	541
accuracy			0.74	953
macro avg	0.73	0.73	0.73	953
weighted avg	0.74	0.74	0.74	953

4.4.5 Classification report for Test Dataset using Random forest with number of tree = 100, max depth = 50

	precision	recall	f1-score	support
0	0.77	0.65	0.70	1221
1	0.76	0.85	0.81	1637
accuracy			0.76	2858
macro avg	0.76	0.75	0.75	2858
weighted avg	0.76	0.76	0.76	2858

4.4.6 Classification report for Train Dataset using Random forest with number of tree = 50, max depth = 5

	precision	recall	f1-score	support
0	0.70	0.58	0.63	412
1	0.72	0.81	0.76	541
accuracy			0.71	953
macro avg	0.71	0.69	0.70	953
weighted avg	0.71	0.71	0.71	953

4.4.7 Classification report for Test Dataset using Random forest with number of tree = 50, max depth = 5