Predicting movie success

by Xin Liu, Qing Hu, Linke Wang, Shuofei Chen (Team 8)

Overview

The purpose of this project is to build a model that we can use IMDB data to judge what will cause high or low ratings.

Solution

Dataset

We exact data from the IMDB Database API:

https://datasets.imdbws.com/

https://www.imdb.com/interfaces/

Data Collection

title.basics: 6685931 rows with 8 columns including titleType, primaryTitle, originalTitle, isAdult, startYear, endYear, runtimeMinutes, genres

title.principals:38604522 rows with 6 columns including tconst ordering, nconst, category, job, characters

title.ratings:1021240 rows with 3 columns including tconst, averageRating, numVotes

name.bascis:9993870 rows with 6 columns including nconst, primaryName, birthYear, deathYear, primaryProfession, knownForTitles

The original dataset contains multiple files, more than 10 million rows. So, we filter data in the following steps.

1. Load from title.basics, title.ratings, title.principals, and name.basics csv files, to select target columns

   Filter the rows that “titleType” == “movie” in the title.basics, to select only movie data. We also filter the rows that “startYear” from 2010 to 2020.
2. Group the title.principals by ‘category’, then make a new principal DataFrame with different categories.
3. Choose tconst (title ID) as the primary key of the movie. We use merge() to inner join title and rating tables first to avoid invalid calculation. After that, we get the target columns of average ratings and the number of votes. To Use the nested loop, iterate title ID to generate the table with specific categories including actor, director, producer… etc. with nconst (name ID).
4. Merge the tables from step 2 and step 3 and save the data.

> This part is in data_collection.ipynb. The output dataset is result2010–2020.csv.

Data Preprocessing

However, the dataset cannot be used in models directly. We need some preprocessing.

5. Drop tconst, which is unique numbers for each movie.
6. Check how many missing values in each category. If missing values > 40%, remove these columns.
7. Drop movies without rating and and numVotes. Then drop movies which contains > 1 categories missing values.
8. In genre, actor, actress and producer parts, we have multiple names in each cell. Only keep the first name.
9. Fill in NaN and /N data.
10. Make continuously data to numbers.
11. Encode categorical data.

Methods

- kNN: n_neighbors parameter = 28
- Logistic Regression
- Random Forest: Performance is often better than decision trees alone. n_estimators parameter = 220
- May use Support Vector Regression and Decision Tree to solve this problem, if available. However, the accuracy result does not seem as well as other models, so we remove them in the final codes.

Model evaluate and verification

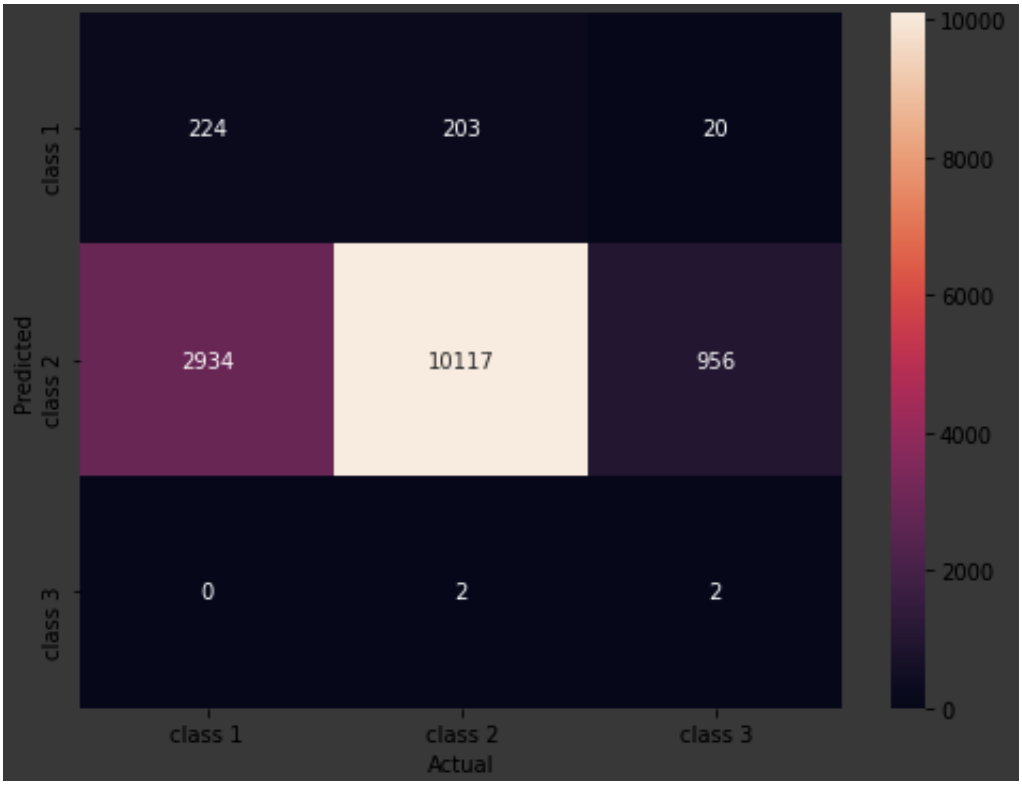
- Use accuracy and graphs to evaluate the model.
- Use the test dataset to check whether the model results fit the test cases very well.

Conclusion

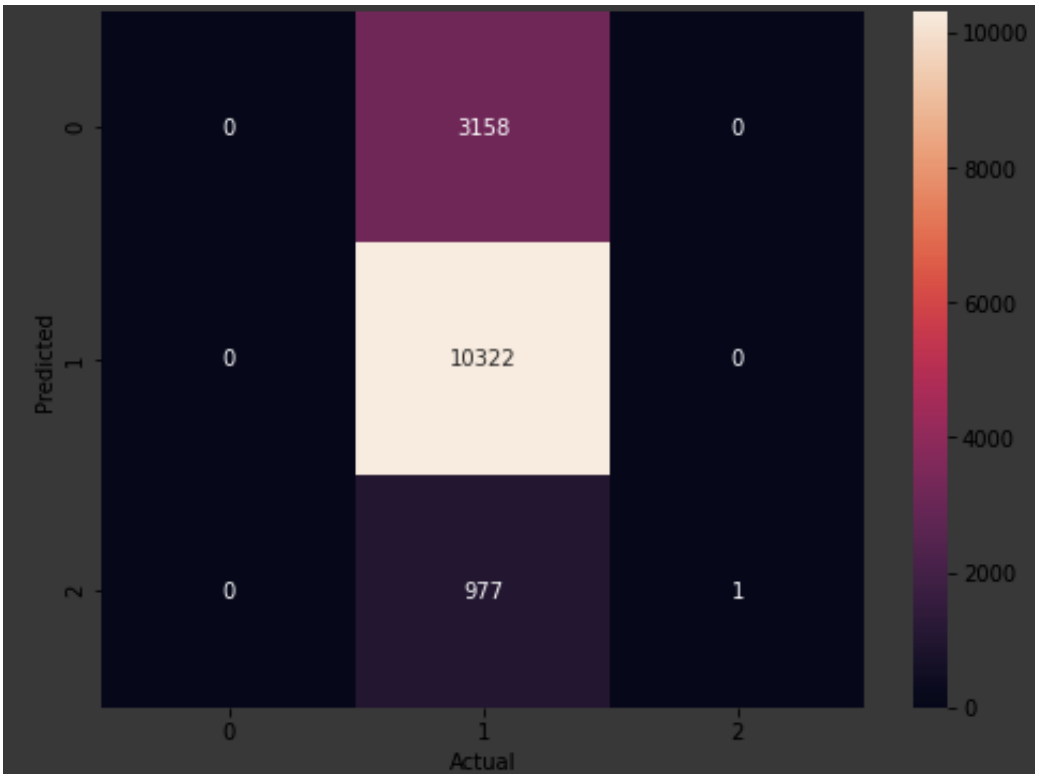
Accuracy

From the accuracy of each model, we find that these 3 models have similar accuracy. ***Random Forest*** *looks better.*

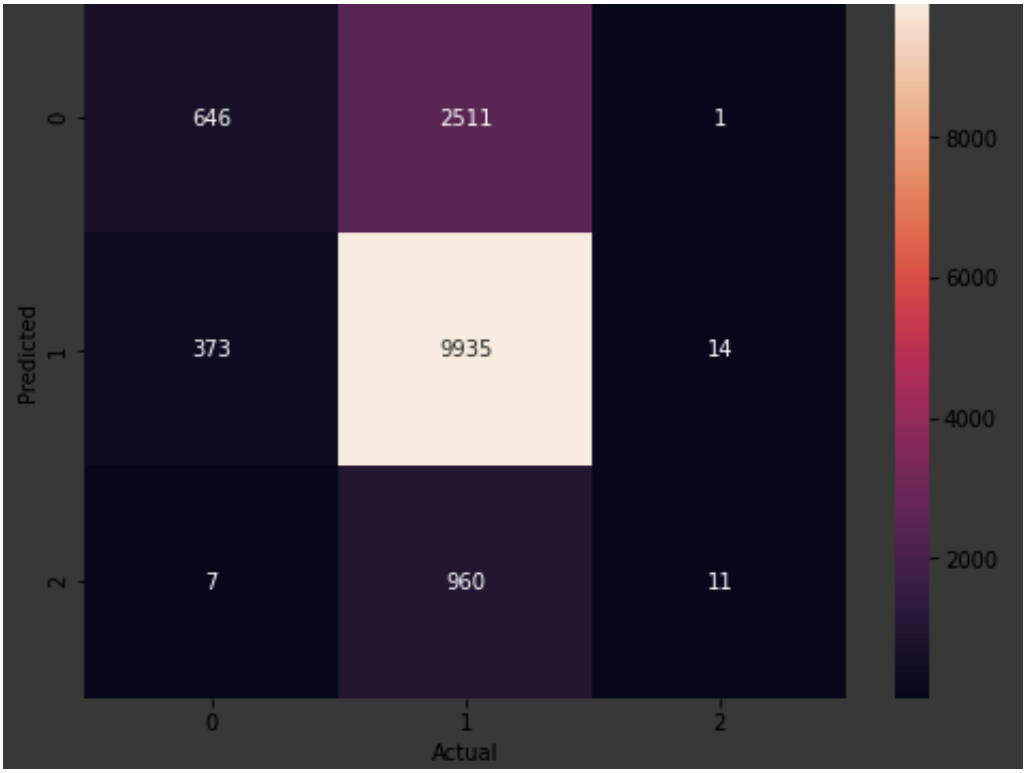
- KNN: 0.7153
    - recall: 0.9893
    - precision: 0.7171
    - F Score: 0.8315
    - confusion_matrix:



- Logistic Regression: 0.7140
  - confusion_matrix:



- Random forest: 0.7330
  - confusion_matrix:


646
2511
1
373
9935
14
7
960
11
0
1
2
Predicted
Actual
8000
6000
4000
2000