COMP30027 - Assignment 2 Report

Anonymous

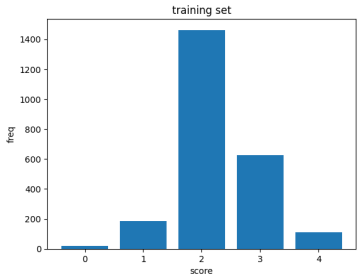
1 Introduction

In this modern era, watching movies is always a top choice for entertainment. However, with millions of movies available, selecting which one to watch is usually difficult. To determine which films are worth watching, numerous review websites provide scores for them, and IMDb (Internet Movie Database) is one of the most reliable sites.
In this study, we will use a dataset collected from the IMDB 5000 Movie Dataset[1] and build several models to predict scores for 752 movies, based on various movie features.

2 Methodology

2.1 Train-Test Splitting

Because the labels are not included in the test set, we will use the hold-out method with an 80/20 split on the training dataset to report the accuracy of the models.



[1] https://www.kaggle.com/datasets/carolzhangdc/imdb-5000-movie-dataset

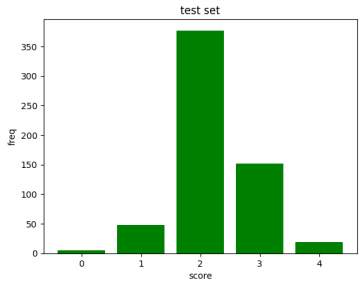


Figure 1: Distribution - Training vs Evaluation

Based on Figure 1, the distribution of the labels in the training set and test set is almost identical, and both of them are imbalanced with a remarkably high number of movies with the scores assigned to bin '2'.

2.2 Data Preprocessing

2.2.1 Categorical Variables

- Using PCA:
  To transform categorical variables into numerical values, we will use CountVectorizer for director_name, actor_1_name, actor_2_name, and actor_3_name; doc2vec for plot_keywords and genres; and fasttext for title_embedding. After that, all the values will be standardized before applying PCA to get 3 principal components explaining the most variance.

- Using label encoding:

language	frequency
English	0.956059
French	0.010652
Spanish	0.006325
Mandarin	0.002996
Japanese	0.002996

country	frequency
USA	0.792943
UK	0.084887
France	0.028628
Germany	0.020306
Canada	0.015313

content_rating	frequency
R	0.453395
PG-13	0.345872
PG	0.152463
G	0.022636
Not Rated	0.008988

Table 1: Frequency of 3 features

Considering the table for the frequencies of languages in the training dataset, it can be seen that most of the movies are in English. Therefore, we will categorize languages into 2 values: 'English' or 'Other language'. Since using numerical values is more convenient than strings, we will assign 1 to 'English' and 0 to 'Other'. The same approach can be applied for 'country' (categorized into 'USA' and 'other') and 'content_rating' (categorized into 'R', 'PG-13', 'PG' and 'other').

2.2.2 Numerical Variables

By common knowledge, the 'id' of a movie does not affect to the score of it, so we can drop it out.

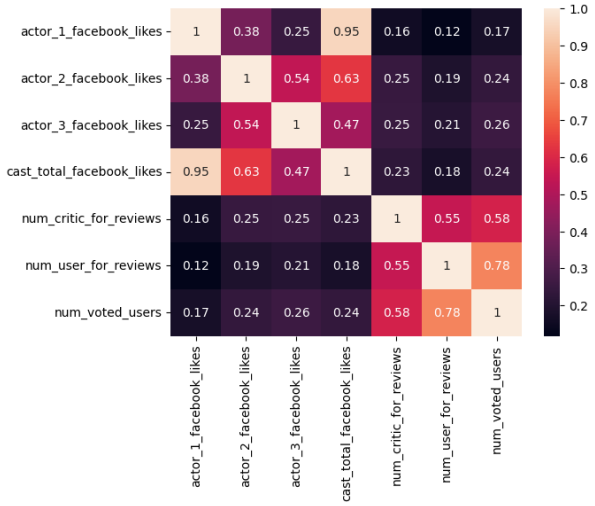


Figure 2: Before preprocessing

Figure 3: After preprocessing

Figure 2 shows several strongly correlated features. These features can cause a slight redundancy and create a high-dimensional feature space. After some modifications, we get Figure 3, in which the correlation of each pair of features is not significantly high (< 0.7). The last step of preprocessing these numerical features is standardizing them (not including the target feature).

2.3 Model

2.3.1 Zero-R (Baseline)

To begin with, we will use Zero-R as a baseline model, in which the test instances are purely predicted by the most frequent label in the training set.

2.3.2 Random Forest

The second model is Random Forest, which uses both bagging and feature randomness to create a set of decision trees with low correlations (IBM, 2024).

- Hyper-parameter Tuning:
  We will use cross-validation to determine the set of best hyper-parameters for the model. In this task, RandomizedSearchCV (using 5-fold CV) will be used to randomly select a fixed number of hyper-parameter combinations, which will be computationally efficient.

  1. n_estimators:
     This is the number of trees in the forest.
     The number of decision trees that we

will consider are: 50, 100, 150, 200, and 250.

2. max_features:
   This shows the maximum number of features to use when finding the best split at each node (Pedregosa et al., 2011). If max_features = None, the random forest classifier behaves similarly to the normal Bagging method as all the features will be considered at each split.
   We will choose 3 values to examine: 'sqrt', 'log2', and None.

2.3.3 Logistic Regression

The final model is (Multinomial) Logistic Regression. It modifies the standard Logistic Regression model to handle multi-class classification problems by returning multinomial probabilities for each input (Brownlee, 2020).

- Hyper-parameter Tuning:
  We will continue to use RandomizedSearchCV to identify the best set of hyper-parameters for this learner.

  1. C:
     This hyper-parameter indicates the inverse of regularization strength, with smaller values giving stronger regularization (lower risk of overfitting) (Pedregosa et al., 2011).
     We consider 5 values for C: 0.01, 0.1, 1, 10, and 100.
  2. solver:
     Solver is the optimization algorithm we use in the model. Since L2 regularization (Ridge Regression) is the only applied penalty, there may be solvers that are not compatible with it.
     Therefore, we will choose 2 suitable algorithms: lbfgs[2] and newton-cg[3].
  3. max_iter:
     This is the maximum number of iterations allowed for a solver to converge during the optimization process (Pedregosa et al., 2011).
     As lbfgs and newton-cg are both fast-convergent algorithms, we take into account 5 values of max_iter: 50, 100, 150, 200, and 250.

[2]Limited-memory BFGS

[3]Newton-Conjugate Gradient algorithm

3 Result

3.1 Zero-R (Baseline)

From Figure 1, we have shown that there are a significantly high number of movies with scores binned to bin '2' (61.2184%). Zero-R predicts the labels of test instances solely based on the majority class in the training set, so we can get a good accuracy for the evaluation set, which is approximately **62.7288%**.

3.2 Random Forest

The accuracy of the Random Forest with different hyper-parameters is shown in the table below:

n_estimators	max_fts	mean	std
200	None	0.7154	0.0116
250	None	0.7145	0.0109
100	None	0.7070	0.0088
250	log2	0.7012	0.0052
150	sqrt	0.7012	0.0077
250	sqrt	0.7012	0.0052
100	sqrt	0.7004	0.0113
200	log2	0.7000	0.0051
50	sqrt	0.6916	0.0158
50	log2	0.6916	0.0158

Table 2: Accuracy for Random Forest

The best set of hyper-parameters is: max_features = None, n_estimators = 200.
The model with these hyper-parameters on the evaluation set gets the accuracy of **73.0449%**.

3.3 Logistic Regression

The accuracy of the Logistic Regression with different hyper-parameters is shown in the table below:

C	solver	max_iter	mean	std
10.0	newton-cg	250	0.6979	0.0098
10.0	lbfgs	50	0.6970	0.0134
1.0	newton-cg	150	0.6970	0.0094
100.0	newton-cg	150	0.6966	0.0101
100.0	lbfgs	250	0.6966	0.0110
1.0	lbfgs	200	0.6962	0.0103
10.0	lbfgs	100	0.6958	0.0133
0.1	newton-cg	100	0.6950	0.0139
0.1	newton-cg	200	0.6950	0.0139
0.1	newton-cg	250	0.6950	0.0139

Table 3: Accuracy for Logistic Regression

The best set of hyper-parameters is: C = 10, solver = 'newton-cg', max_iter = 250.

The model with these hyper-parameters on the evaluation set gets the accuracy of **70.8819%**.

In summary, we have the table for the accuracy of each model on the evaluation set below:

Zero-R	Random Forest	Logistic Regression
62.73%	73.04%	70.88%

Table 4: Accuracy on evaluation set

4 Discussion and Critical Analysis

4.1 Discussing about the models and their hyper-parameters

4.1.1 Zero-R

As the training dataset is imbalanced, using Zero-R would give a relatively good baseline for other classifiers. However, this classifier is not practical for predicting test instances as the distribution of the test set is not guaranteed to be the same as that of the training set.

4.1.2 Random Forest

- Model:
  Random Forest is an ensemble learning, which is constructed from a numerous number of decision trees. This allows the model to have a lower variance (lower chance of overfitting) compared to using the single decision tree (Couronné et al., 2018). Moreover, Random Forest performs efficiently on large and imbalanced datasets (More and Rana, 2017), which is suited to handle the problem of our data.

- Hyper-parameters:
  Surprisingly, after tuning the hyper-parameters, the best model is the Bagging Trees model (max_features = None). Random Forest is worse than Bagging Trees in terms of interpretability as we cannot examine the individual trees separately (Prasad et al., 2006). Furthermore, Random Forest uses a small randomly chosen subset of features at each split, which may lead to a sub-optimal outcome as the subset may contain all non-significant attributes (smaller max_features → average worse) (Probst et al., 2019). This problem does not happen with the normal Bagging method. Considering the standard deviation of the tuned model's accuracy, we get 0.0116, so the model is fairly stable.

4.1.3 Logistic Regression

- Model:
  Despite its simplicity, Logistic Regression is always one of the most powerful models, and in some situations, it can outperform many other complex models (Kirasich et al., 2018). For instance, Logistic Regression outperforms AdaBoost and XGBoost for datasets with significantly imbalanced ratios (Md Shahri et al., 2021).

- Hyper-parameters:
  After doing the random search, we get C = 10, which indicates a slightly weaker regularization compared to the default version of the Logistic Regression model (C = 1). The standard deviation of the accuracy of the tuned model in 5-fold CV is 0.0098, which implies that the model is relatively stable.

4.2 Analyzing the models

4.2.1 Identifying the problem

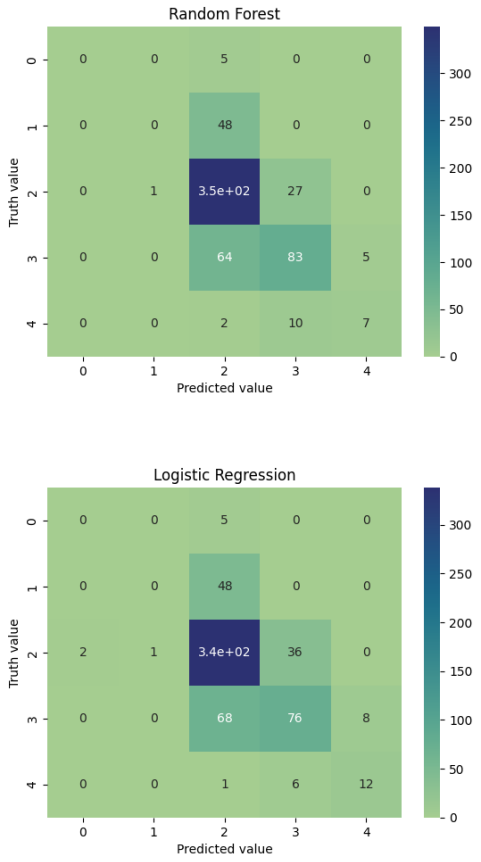


Figure 4: Confusion Matrices of 2 models

Based on Figure 4, it can be seen that both of the models behave the same on the evaluation

set. Considering the table of weighted average of evaluation metrics:

	Random Forest	Logistic Regression
precision	0.6612	0.6428
recall	0.7304	0.7088
f1-score	0.6868	0.6685

Table 5: Weighted average of evaluation metrics

We can see that the Random Forest classifier outperforms the Logistic Regression in all evaluation metrics. However, although these 2 classifiers have good accuracy in predicting 'normal' movies (score = 2), they have poor performance in classifying 'bad' (score = 0, 1) and 'good' movies (score = 3,4). With an imbalanced dataset, both models have a strong bias toward the majority class.

4.2.2 Suggesting solutions

To handle the problem of an imbalanced dataset, these are some recommended methods that can be applied:

- SMOTE[4]: Since the models have a heavy bias toward the most-frequent class, we need to resample the data by adding more instances with minority labels, and SMOTE is proved to be better than other algorithms in regards to dealing with imbalance (Spelmen and Porkodi, 2018).
- RE-WLR algorithm[5]: This algorithm can be applied to the Logistic Regression model, which is robust to imbalance (Maalouf and Siddiqi, 2014).

5 Conclusion

We presented 3 models to predict the IMDB scores of the movies: Zero-R (Baseline), Random Forest, and Logistic Regression. Due to the imbalance in the training set, these models perform well in predicting the majority class, while bad in the others. There are some techniques such as SMOTE and the RE-WLR algorithm that we can use to improve our models' performances.

Words count: ≈ 1600

[4]Synthetic Minority Oversampling Technique

[5]Rare Event Weighted Logistic Regression algorithm

References

Jason Brownlee. 2020. Multinomial logistic regression with python.

Raphael Couronné, Philipp Probst, and Anne-Laure Boulesteix. 2018. Random forest versus logistic regression: a large-scale benchmark experiment. *BMC bioinformatics*, 19:1–14.

IBM. 2024. What is random forest? `https://www.ibm.com/topics/random-forest`.

Kaitlin Kirasich, Trace Smith, and Bivin Sadler. 2018. Random forest vs logistic regression: Binary classification for heterogeneous datasets. *SMU Data Science Review*, 1(3).

Maher Maalouf and Mohammad Siddiqi. 2014. Weighted logistic regression for large-scale imbalanced and rare events data. *Knowledge-Based Systems*, 59:142–148.

Nur Huda Nabihan Md Shahri, Sharmeen Lai, Mazni Mohamad, Hezlin Rahman, and Adzhar Rambli. 2021. Comparing the performance of adaboost, xgboost, and logistic regression for imbalanced data. *Mathematics and Statistics*, 9:379–385, 05.

A. S. More and Dipti P. Rana. 2017. Review of random forest classification techniques to resolve data imbalance. In *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)*, pages 72–78.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Anantha M Prasad, Louis R Iverson, and Andy Liaw. 2006. Newer classification and regression tree techniques: bagging and random forests for ecological prediction. *Ecosystems*, 9:181–199.

Philipp Probst, Marvin N Wright, and Anne-Laure Boulesteix. 2019. Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: data mining and knowledge discovery*, 9(3):e1301.

Vimalraj S Spelmen and R Porkodi. 2018. A review on handling imbalanced data. In *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, pages 1–11.