Predicting IMDB Movie Rating with Machine Learning Methods

Anonymous

1. Introduction

The movie rating serves as a crucial measure for assessing quality and popularity. Predicting movie ratings can aid filmmakers in managing their expectations for their films. In this study, a movie dataset sourced from the IMDB 5000 Movie Dataset on Kaggle was utilized, comprising 3004 movie instances in the training set and 752 instances in the test set. The objective was to predict movie ratings using various machine learning methods, namely Gaussian Naive Bayes (GNB), k-Nearest Neighbors (KNN), Random Forest Regressor (RF), and neural network (NN). The performances of these methods were compared and analyzed. Prior to inputting data into the machine learning models, several feature engineering techniques were applied to enhance prediction accuracy.

2. Data and Methods

2.1 Data description

The dataset has been split into the training set and the test set. The training set has 3004 instances, and the test set has 752 instances.

There are 24 features, among which, 14 features are numeric, including: `num_critic_for_reviews`, `duration`, `director_facebook_likes`, `actor_3_facebook_likes`, `actor_1_facebook_likes`, `gross`, `num_voted_users`, `cast_total_facebook_likes`, `facenumber_in_poster`, `num_user_for_reviews`, `title_year`, `actor_2_facebook_likes`, `movie_facebook_likes`, `average_degree_centrality`; and 10 features are textual, including: `title_embedding`, `genres`, `plot_keywords`, `director_name`, `actor_1_name`, `actor_2_name`, `actor_3_name`, `language`, `country`, `content_rating`.

The labels of each instance are movie rating scores, which have been binned into five classes, 0, 1, 2, 3, and 4. The labels of the test set are not provided. Thus, the main evaluation metric of each model is **cross validation accuracy**.

In the training set, the label numbers of each class are 24, 235, 1839, 777, and 129 for 0, 1, 2, 3, and 4, respectively, which means that if performing 0-R, the cross validation accuracy should be 1839/3004=61.22%.

2.2 Pre-processing of text features

The text features cannot be input directly for most of machine learning algorithms, thus text features must be converted into numeric representations. According to the types of features, 3 vectorization methods were employed. Movie titles were converted by *FastText*, which could extract the representations of words. Nominal features like names, languages, and content rating were encoded by one-hot encoding (scikit-learn function *CountVectorizer* for names and in house code for languages and content rating). Features of genres and plot keywords, which include some key words together for each instance, were converted by *Doc2Vec*.

The conversion of text features to numeric vectors increased the dimensions from one to hundreds of dimensions. To reduce the dimension, principal component analysis (PCA) was utilized. For each feature, two components were retained. Therefore, after text features processing, the number of features was 34 (14 numeric features and 10*2 principal components for each text feature).

2.3 Data standardization

For each feature, the value of each instance of the training set was subtracted by the mean of this feature in the training set and then divided by the standard deviation of this feature in the training set. The standardization of the test set used the same parameters generated from the training set.

2.4 Feature selection

Two algorithms, mutual information (by *mutual_info_regression* of scikit-learn) and random forest (got from the model's feature_importances_) were employed to evaluate the importance of each feature. Top 20 features each were selected for another round of prediction separately.

2.5 Machine learning models

2.5.1 Gaussian Naive Bayes (GNB)

Model was initialized by *GaussianNB()* of scikit-learn. Model was trained with the training dataset and used to predict the ratings of the test set.

2.5.2 k-Nearest Neighbors (KNN)

Model was initialized by *KNeighborsClassifier ()* of scikit-learn with default parameters (k=5). Model was trained with the training dataset and used to predict the ratings of the test set.

2.5.3 Random Forest Regressor (RF)

Model was initialized by scikit-learn's *RandomForestRegressor(n_estimators = **1000**, random_state = 888, n_jobs = -1),* in which, the number of decision trees was set as 1000; random state was set to ensure the reproducibility; and n_jobs = -1 was set to run parallelly with all CPU available. Model was trained with the training dataset and used to predict the ratings of the test set.

2.5.4 Neural network (NN)

A 4-layer neural network was built using tools of tensorflow. The size of the input layer was the same as the feature number. The first fully connected hidden layer included 512 neurons and the activation function was ReLU. After this layer, dropout regularization (*Dropout(0.2)*) was introduced to avoid overfitting. The second fully connected hidden layer included only one neuron, with ReLU as the activation function. This layer acted as a "pseudo" regression-like output layer since the prediction of a movie rating was actually the regression of a single value from 0 to 4. The single value was manually binned into 5 classes. Thus, the output layer contained 5 neurons with softmax as the activation function, outputting the probabilities of each class. The labels were one-hot encoded for the training purpose of this model. The output of the prediction was in one-hot encoding format and was converted back to the class labels. For the training, the optimizer was adam; the loss function was categorical_crossentropy; and the evaluation metric was accuracy.

2.6 Cross validation

Since the labels of the test set were not provided, cross validation was performed to examine the capacity of models.

For GNB, KNN, and RF, cross validation was performed by *cross_val_predict* of scikit-learn. Parameter cv was set as 10. The output was compared with the labels of the training set to get the accuracy.

For NN, the training set was split by *KFold.split* of scikit-learn into 10 subsets (random seed was set), 9 subsets for training and 1 for testing. Every instance was tested for once and the prediction was compared with its label to get the accuracy.

3. Experiments

The cross validation and model training and predicting were performed for 5 rounds for each model.

In the first round, considering that features including both numeric features and text features, to simplify the setting in this round, the text features were excluded, only retaining 14 numeric features. The models were trained with raw data, without any standardization.

In the second round, to detect the effect of standardization, compared with the first round, data were standardized before model training. The same as the first round, only numeric features were included.

In the third round, text features were included to detect whether text features would improve the prediction accuracy. As described at **2.2**, two principal components of each text feature were used. The total feature number is 34, 14 numeric features and 10*2 text feature principal components. Data were standardized before model training.

In the next two rounds, features were selected after standardization and before model training to detect whether feature selection would promote the prediction accuracy.

In the fourth round, features were selected based on mutual information scores which were calculated based on probabilities. Top 20 features were selected. Except the feature selection, all the remaining steps were the same as the third round.

Similarly, in the fifth round, features were selected but with an alternative method, based on the feature importance which was calculated by the random forest model of the third round. Except the feature selection, all the remaining steps were the same as the third round.

Generally, the experimental design was, 1) in the first round, only raw data with numeric features were used; 2) in the second round, data standardization was added to the process used in the first round to detect the effect of **standardization**; 3) in the third round, in addition to the previous round, text features were added with 2 principal components for each text feature to detect the effect of **text features**; 4) for the final two rounds, features were selected based on the standardized data from the third round to detect the effect of **feature selection**.

The accuracy scores of cross validations of each model for each round were show in the below Table 1.

	GNB	KNN	RF	NN
1st round	0.295	0.582	0.721	0.612
2nd round	0.279	0.626	0.722	0.694
3rd round	0.230	0.616	0.723	0.683
4th round	0.280	0.632	0.701	0.667
5th round	0.250	0.631	0.725	0.690

Table 1 The cross-validation accuracy scores of each model for each round

4. Discussion

For GNB, the cross validation accuracy scores are consistently low across the five round experiments. GNB has two assumptions. First, the features should be independent to each other and second, the features should follow a normal distribution. It is clearly that this dataset failed to satisfy these two assumptions. Some features are highly correlated like the names of actors or directors and their facebook likes. And some features do not follow normal distribution, like PCA components of text features and the title years. Therefore, the model GNB is not suitable for this dataset.

For KNN, the accuracy scores vary among five rounds. In the first round, no standardization results in the lowest accuracy score. Without standardization, KNN is influenced significantly by features with large scales by dominating the calculation of the distances. If large scale features could not distinguish the labels well, the model performance suffers, as seen in the first round. After standardization, the accuracy score of the second round improves from 0.582 to 0.626. However, when text features are introduced in the third round, the accuracy score decreases to 0.616, suggesting that text features generally do not contribute to the improvement of KNN. Nevertheless, after feature selection in the last two round, the accuracy scores improve, indicating that both feature selection methods effectively reduce noise from irrelevant features and enhance KNN performance.

For RF, the accuracy scores remain stable across all rounds except in the fourth round. Standardization does not improve model performance for RF, as it is an ensemble learning method composed of decision trees. Unlike KNN and NN, where features are aggregated during distance calculations or layer outputs calculations, **each feature in decision trees works independently**, and the scale of one feature does not impact others. Similarly to KNN, the addition of text features does not enhance the model. Furthermore, feature selection does not improve the model as RF already includes a feature selection step within its framework. In fact, the first feature selection method even decreases accuracy, indicating that additional feature selection is not suitable for RF in this dataset. The second feature selection method, inherent to the model, also does not improve performance.

For NN, similar to KNN, standardization improves the model (as shown in the second round, accuracy score from 0.612 to 0.694) while the addition of text features does not (as shown in the third round, accuracy score from

0.694 to 0.683). As discussed before, in NN, features are summed up during the calculation of layer outputs. Thus, the adjustment of features to similar scales is critical to get better results. Without feature selection, the addition of text features does not improve the NN model. Text features might include both useful information and noise. As for the feature selection, only the selection based on RF model improves the accuracy of the third round, but it is not better than the second round including only numeric features. In the fourth round employing selection based on mutual information, the accuracy decreases from 0.681 to 0.667. Checking the features that only exist in the fourth round but not in the fifth round, they are:
`director_name_PC2`, `director_name_PC1`, `actor_1_facebook_likes`, `actor_1_name_PC1`, `content_rating_PC2`, `actor_1_name_PC2`, `content_rating_PC1`, `country_PC2`, `average_degree_centrality`.
Most of them are text features and are converted by **one-hot encoding**. Intuitively, the director and actor_1 should be very important to movie ratings, however, within this round, keeping them reduces the accuracy. It may not be attributed to the low importance of directors and lead actors, but rather to the conversion approach. It is possible that NN struggles with interpreting one-hot encoding vectors, suggesting that the conversion method warrants further investigation.

5. Conclusions

1) GNB is not suitable for this dataset.
2) **Standardization** improves KNN and NN.
3) **Text features** do not contribute to the improvements of KNN, RF, and NN.
4) Feature selection based on **mutual information** is not suitable for RF and NN.
5) NN does not have a good interpretation for **one-hot encoding** vectors in the setting of this dataset.

6. Appendix

Codes and prediction output for the experiments have been attached.

7. References

No public publication is cited. All content is referenced from subject slides and official documents of scikit-learn and tensorflow.