Movie Review Sentiment Analysis

Jingyi Zhu *NetID jingyiz9*

Xi Chen *NetID xic6*

1 Introduction

In this project, we use a dataset contains IMDB movie reviews to predict the sentiment of a movie review. This dataset has 50,000 review rows and 3 columns. Column 1 "new_id" is the ID for each review, column 2 "sentiment" is the binary response, and column 3 is the review. We split the 50,000 reviews into three sets of training and test data.

2 Method

In this report, we use three models: logistic regression with lasso, the discriminant analysis Naive Bayes and the tree model. First, we clean the html tags and then split the reviews into three sets, one of them is for test data and the remain are for training data. Then, we apply three models to the vocabulary we created.

2.1 Model 1: Logistic Regression with Lasso

We use R package 'text2vec' to build vocabulary and construct DT matrix (dtm). We set some stop words in the process of creating vocabulary due to deducting the size of the vocabulary. The stop words contain "I", "me", "my", "myself", "we", "our", "ours", "ourselves", etc. These words have no positive or negative affects to the review result, so we just remove them from the dataset. After removing the stop words, we obtain 28,628 features which larger than the sample size n = 25,000.

Then we applied glmnet with lasso and made prediction with lambda.min. The lambada.min = 0.002306055 and we use binomial to apply the logistic regression. After using predict() function we acquired the fitted values for the test data and compare with the real values to calculate the AUC. In this model, AUC over the test set is above 0.96 for all three training/test splits.

However, it is difficult to interpret some of the selected variables such as the words with non-zero logistic regression coefficients. Also, some meaningful words like “great” or “awful” are mixed with a large amount of hard-to-explain words.

2.2 Model 2: Discriminant Analysis Naïve Bayes

Two-sample t-test is a simple screening method. We have one-dimensional observations from two groups to test whether the X population and the Y population have the same mean, we compute the following two-sample t-statistic where s_X^2 denotes the sample variance of X.

$$\frac{\bar{X} - \bar{Y}}{\sqrt{\frac{s_X^2}{m} + \frac{s_Y^2}{n}}}$$

We use R package ‘slam’ to improve the efficiency of computing the mean and var for each column of dtm_train because dtm_train is a large sparse matrix.

Then we order the words by the magnitude of their t-statistics, pick the top 2000 words, which are then divide into two lists: positive words and negative words. The positive words are "great", "excellent", "wonderful", "a great", "best", "the best", etc. The negative words are "bad", "worst", "the worst", "waste", "awful", "terrible", etc. Then we apply ridge regression on the selected 2000 words and the current vocabulary could be further trimmed, e.g., probably no need to keep both "wonderful job" and "wonderful", and some words like "through life" and "life as" do not make much sense. The results of AUC are quite high, which is above 0.96 for three test datasets.

Then, we try Naïve Bayes and after using predict() function we acquired the fitted values for the test data and compare with the real values to calculate the AUC. In this model, AUC are not good enough for three split datasets, which are all below than 0.7.

2.3 Model 3: Xgboost

We use boosting trees to generate some new nonlinear features. First, we use xgboost to generate 500 trees and the depth of each tree is equal to 2. When

passing an observation to a tree, it'll land in one and only one of the leaves. So, the 500 trees correspond to 500 categorical variables, each having $4 = 2^2$ leaves.

The command xgb.create.features in XGBoost can generate a binary design matrix (dummy coding) for these new 500 categorical features. The design matrix could have maximum 500×(7-1) + 1 columns: the first categorical variable has 8 columns, and each of the remaining has 7 columns. Since 2 is the maximum depth, some trees could have number of leaves less than 8.

Then we fit a logistic regression with lasso using the new features. The lambada.min = 0.002306055. In this model, AUC are good for all test dataset, at about 0.95. However, it doesn't achieve the benchmark yet.

3 Result

Below are AUC, the evaluation outcomes of three models:

Table 1: AUC of Three Testing Datasets

Model	Set1	Set2	Set3
Logistic with lasso	0.9614684	0.9609024	0.9622546
Discriminant Analysis	0.6983600	0.6817106	0.686456
Xgboost tree	0.9490652	0.9494514	0.9513645

4 Conclusion

For logistic regression with lasso, it is difficult to interpret some of the selected variables such as the words with non-zero logistic regression coefficients. All 3 AUC are larger than 0.96.

For discriminant analysis, we first order the words by the magnitude of their t-statistics, pick the top 2000 words, which are then divide into positive and negative words. Then we use Naïve Bayes. We find it very interesting because this model gives a bad performance, which is even lower than 0.7. For the reason, it is most possible that the probability result exactly equals 0 or 1, which means

during the calculation processing, the calculation accuracy is not smart, so maybe we can improve it by own function instead of build-in function.

For Xgboost, we generate some new nonlinear features through setting the number and the depth of trees. Then we apply the logistic regression with lasso to the new features and the AUC are at about 0.95. Thus, as an interesting finding, the AUC is almost reach the threshold, but it is not easy for us to improve it, no matter for the number of trees, or the depth.

5 Other information

• Computer system: MacBook Pro, 2.53 GHz, 4GB memory

• Running time of dataset3: 31.99301 minutes.