Anomaly Detection Challenge 2

Miguel Sandim and Paula Fortuna

Technische Universität München
{migsandim, paulatfortuna}@gmail.com

1 Introduction and data description

The goal of this challenge was to detect fake hotel reviews. This was a problem of binary classification, therefore the training dataset was presented with a binary class named "fake" (Y or N). The main dataset for this problem was a list of reviews and its attributes consisted of descriptive information and textual content of the review itself. Besides, in this dataset there was also some complementary information about the hotels and reviewers [1].

2 Data reading

In this challenge, data reading was not direct. The difficulties found were that: both "," and ";" were used as separation character in the same document; the content text from the reviews had the separation character, which generated extra columns when parsed; and there were random new lines in the document.

These problems were solved using regular expressions and manual inspection. No data was lost and we end up with 2908 instances.

3 Unbalanced data and sampling

From the descriptive statistics obtained we concluded that both classes are unbalanced in the dataset: the positive class (fake review) consists of about 13% of our dataset's entries. The unbalance in the classes' frequency can have an impact on the model performance, therefore two different methods were used in order to solve this problem.

- **Undersampling -** One possible solution was to discard the excessive amount of the most frequent class. In this case we randomly took a sample from the non fake reviews and ignored the remaining ones.
- **Oversampling -** Another possible solution was to replicate the less frequent class. In this case we duplicated the instances from the fake reviews until we got the same number of cases in each class. This strategy was discarded after some preliminar experiments, because with it the kaggle AUC values significantly decreased.

4 Data Pre-processing

4.1 Replacing Missing Values

The data was inspected in order to find missing values and we concluded that there was only one instance with incomplete data. That was the case for the textual content of the review and this field was replaced with a string containing one "space" character.

4.2 Feature Extraction

4.2.1 Features from the paper associated to the dataset [1]

4.2.1.1 lengthReview We computed both the length of the field and also the number of words in it.

4.2.1.2 bagOfWords The paper referred that some words are more frequent in the fake users' reviews. These words were divided in different categories and we computed a new feature according to the words' frequencies dividing by the total number of words.

- **personal pronouns -** i, we, me, us, my, mine, our and ours.
- **associated actions -** went and feel.
- **targets -** area, options, price and stay.
- **emotion words -** nice, deal, comfort and helpful.

4.2.1.3 Maximum Number of Reviews (MNR) The paper also stated that fake reviewers complete more reviews than the other ones. To capture this pattern we computed the following metric:

$$MNR(reviewer_n) = \frac{|reviews(reviewer_n)|}{daysBetween(joinDate(reviewer_n), date(mostRecentReview))} \tag{1}$$

4.2.1.4 Percentage of Positive Reviews (PR) Fake reviewers tend to do more positive reviews, therefore we computed the following metric, considering reviews both in the test and training data:

$$PR(reviewer_n) = \frac{|positiveReviews(reviewer_n)|}{|reviews(reviewer_n)|} \tag{2}$$

4.2.1.5 Reviewer deviation (RD) Fake reviewers are also referred to be more likely to deviate from the general rating consensus. We tried to simulate this tendency with the following metric:

$$RD(review_n) = rating(review_n) - rating(hotel(review_n))) \tag{3}$$

4.2.1.6 Maximum content similarity (MCS) The textual content of the reviews performed by a user tends to be more similar to each other if this user is a fake reviewer. For this feature we computed the cosine similarity between two strings. For each user we obtained all the reviews and computed the content's average cosine similarity for each possible combination of two reviews.

4.2.2 Others

4.2.2.1 markFakeUsers The implementation of this feature assumes that if a user makes a fake review, then probably all his or her reviews are fake. Then we assigned to each user a boolean field asserting if he/she wrote at least one fake review. It is important to notice that the training and test sets share around 15% of the users so we expected further improvements in the predictions with this variable.

Finally, it is important to point out that all the features already presented in the dataset were considered in the analysis. Besides, we also try to normalize all the variables that were not between 0 and 1 with the z-score method, which did not improved the results.

4.3 Feature Selection

In order to understand the relevance of the features we checked the feature importances provided by the Random Forests classifier. We concluded that choosing among the several features had little impact in the AUC values. Therefore using Random Forests as a model would be a good option, because this model is robust in the presence of useless features. Besides, all the features containing text were drop or replaced by numeric values.

5 Model Selection and Tuning

The next important part in this challenge was the model selection. In order to achieve the best model, we developed a methodology with the following steps:

1. Evaluate the performance of several binary classification machine learning techniques (using default tuning options).
2. Tune the algorithm that achieved the best performances.

5.1 Model Selection

This first step was used to choose which classification models were worth spending more time on during the tuning phase.

The following models were tested during this phase:

- Logistic Regression
- Linear Discriminant Analysis
- K-Nearest Neighbors Classifier
- Decision Tree Classifier (CART)
- Support Vector Machines (with both "linear", "poly" and "rgf" kernels)
- Multilayer Perceptron
- Random Forest
- AdaBoost
- Gradient Boosted Decision Trees (XGBoost)

These models were evaluated for each data pre-processing methodology developed, using 10-fold stratified cross validation (in order to retain the class ratios on each fold) and the AUC metric (since it was the one we're trying to maximize in the Kaggle challenge).

The results of this phase are summarised in figure 1, which reveals that the tree-ensemble algorithms are the most suitable to this specific dataset (Random Forests scored 0.820558, Adaboost 0.834778 and XGBoost 0.852146).

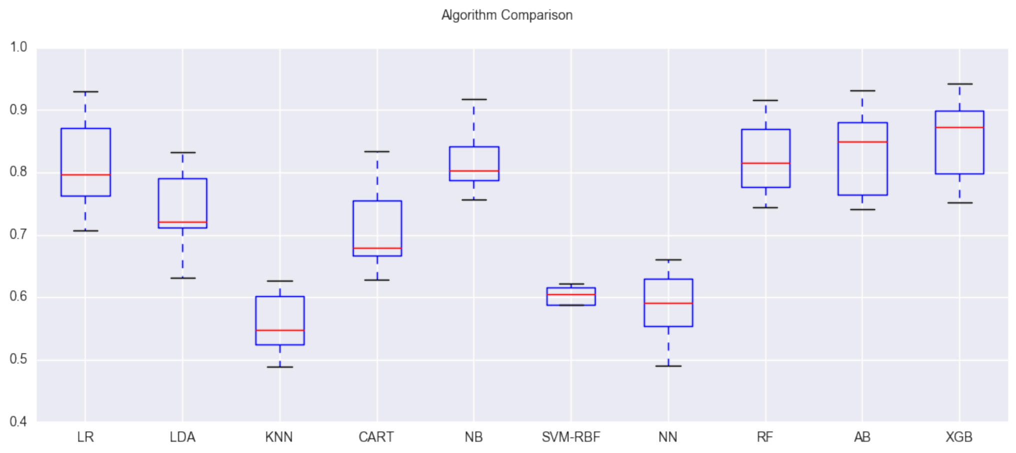

Fig. 1. Boxplot of the 10-fold cross validation performance (across the different fold combinations) of the several algorithms evaluated.

5.2 Model Tuning

Based on the results from the previous section, we decided to tune the Random Forests algorithm, varying the parameters "number of estimators", "criterion" and "maximum number of features" using grid search. This tuning process led to a new score of 0.858400.

We also tried to tune the XGBoost algorithm (which was the algorithm that held the best results) but we didn't notice any increase in the ROC-AUC metric.

6 Conclusion

To solve this challenge our group started by implementing the features cited in the paper suggested in the class. This approach contributed to a small increase in the AUC metric both in the public leadership and in our offline evaluation. Despite in this challenge we were using the same source of data as in the paper (Yelp), probably the data is not the same, and therefore the majority of the conclusions do not apply.

After trying this first approach, we started to look for other features, that can be found directly in the dataset (e. g. fan count of the user). These features allowed us to improve our result.

Even with a better result we wanted to go further and we continued trying to implement new features (e. g. mark user as fake, when he or she had made a fake review). Despite the different approaches conducted, this lead to no better result.

In conclusion, our offline best score was obtained combining all the features computed, with undersampling, and random forests as a model. The difficulty in increasing the AUC score pointed out how complex it is to analyse unstructured text data and also human behavior.

References

1. Arjun Mukherjee et al. “What yelp fake review filter might be doing?” In: 2013.