Predicting on Highly Unbalanced Datasets Fraud Detection Advanced Machine Learning Lab 3

Austin SCHWINN and Usama JAVAID

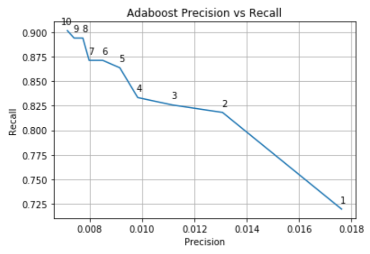
December 15, 2017

1 Introduction

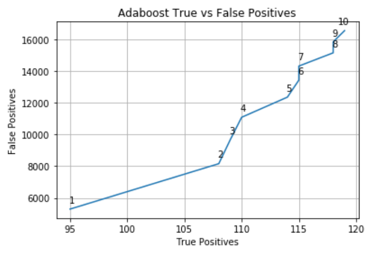
Capturing and predicting fraudulent transactions is a much harder task than we originally anticipated. Adapting classic machine learning algorithms to work effectively on data which has such a small portion of positive examples proved to be quite difficult. We decided to stay focused on the actual business application that this lab was based around. In a business setting, fraudulent transactions bare a very high price. One of our two lab-mates, Austin Schwinn, previously worked at an e-commerce company that sells gold and silver where identifying fraudulent transactions was a high priority. Each flagged order would be reviewed by an expert, which cost the company \$ 25 per hour. Each case of fraud that was not caught, however, could cost the company between \$ 2000 and \$ 200,000. In addition to financial costs of fulfilling fraudulent orders, companies face harm to their reputations and can even have legal issues with credit companies like Visa if they allow too many cases slip. Because of this, we used precision and recall as our evaluation metrics and focused exclusively on achieving the highest number of true positives with an acceptable amount of false positives. For our implementation, we compared Adaboost and Random Forest algorithms. In the end, we found that Random Forest incrementally outperformed Adaboost, but only slightly.

2 Experimentation and Results

For our implementation on the highly unbalanced fraud detection dataset, we decided to compare the Adaboost and Random Forest algorithms. We started our implementation by splitting our dataset and withholding 10% of the total data to use as a validation set later. On top of the difficultly of training a model to predict well, the computation power required of even a single day's worth of data proved to be difficult for our computers. Because of this, we randomly selected 100,000 false examples from the remaining 90% of the dataset to keep for our training set. We then used oversampling to balance our dataset for the training phase. We decided to use smote to create artificial positive examples, balancing our dataset from our original 1294 positive training examples. By combining our over-sampled positive examples and randomly selected negative examples, our total training size was 200,000 examples. The first algorithm we used was Adaboost using scikit-learn's implementation. By running the default model, we achieved a recall of 77%. To achieve a more accurate recall, we adjusted the weighted of the observations multiple times. We found that a weight of 2 on positive examples to a weight of 1 on negative examples gave us the best recall for the precision. The recall of 82% can be improved upon but is not worth the additional drop in precision. The interplay between precision and recall is visualized in figure 1(a). The balance between true positives and false positives is shown in figure 1(b). While the model gave us 8,000 false positives, this could still be feasible in a business setting. For example, we tried using the actual financial costs to a business as our weights on the examples. We gave positive examples a weight of 10,000 and negative examples a weight of 1. The model caught all the fraudulent examples but by looking at the confusion matrix, we found the model produced 57,719 false positives. This would be well beyond the capabilities of the staff to review and therefore is not a feasible model.



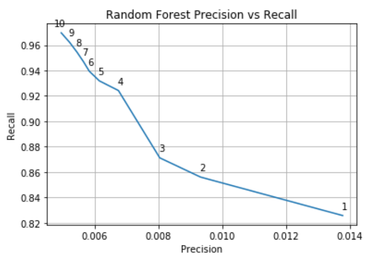
(a)



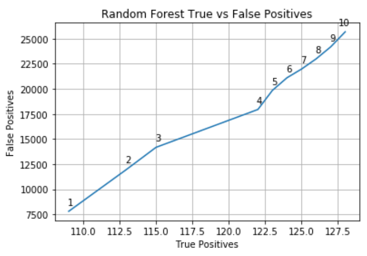
(b)

Figure 1: Metrics labeled with positive to negative weights ratios i.e. 3 = 3:1 (pos:neg)

The second algorithm we implemented was Random Forest. We found that the best recall we could achieve without adjusting the weights of the labels was 84%. It achieved this recall, which is approximate to Adaboost, with much fewer false positives. We were able to achieve this recall with 100 estimators of a max depth of 7. We found that increasing the size of the random forest beyond this had no impact. With 500 estimators and a max depth of 8, we achieved the exact same recall as before. We then were able to increase the recall by adjusting the weights just as we did with Adaboost. However, we found that the extra work presented in the gain of false positives was not worth the gain in true negatives.



(a)



(b)

Figure 2: Metrics labeled with positive to negative weights ratios
i.e. 3 = 3:1 (pos:neg)

3 Conclusion

In conclusion, we found that random forest is superior in this setting but only slightly. When achieving the same recall, it is able to do so with fewer false positive predictions. Random Forest is also able to achieve a slightly higher recall and precision while staying within the constraints of a feasible model in a business setting. While both models can achieve a recall of 100%, they do so in a way that is not actual feasible in real life scenarios because of their high trade-offs with false positive predictions. In the end, both models have low overall precision and have prominent short-comings. This lab has given us a clearer picture of why it is more difficult working with highly unbalanced datasets and the extent to which it is difficult