Math 189: Final Project

Utilizing Random Forests to Predict Network Outages on Sparse Input Data

Sherry Diep, Sandra Hui, David Lee, Irving Valles, Albert Xu, Mark Yee

March 28, 2016

Introduction

As a telecommunication company, Telstra strives to provide the best services to its customers, whether it be mobile, internet, or other entertainment services. Like any other company competing for customers on the market, Telstra ensures that one of their top priorities is customer satisfaction. In order to be a great provider, the company must find solutions to problems that may arise, such as service errors from dropped calls to lost internet connections. Here, Telstra would like to find a way to more accurately predict intensity of service disruptions. The following analysis has been conducted through data provided by a challenge simulation modeled to be similar to what the Telstra network would deal with.

The Telstra network collects logs for its locations, and we wished to predict the fault severity at a time at a particular location based on the logs. The logs provide 54 different event types, 386 log features, 10 different resource types, and 4 fault types. Event types were taken to mean log events that were not warnings indicating a fault. These were accompanied by a volume value, which was taken to mean the multiplicity of the event. The 386 log features were taken to mean event details leading up to a fault. Resource types were taken to mean that one of the 10 resources was available at the location at the time. Lastly, the fault type was extracted from the logs. The fault severity, on the other hand, was reported by customers of the network as they perceived it.

Random Forests

Due to the large number of features and sparseness of the data matrix, a single classification tree would be unsuitable for analysis. Random forest is a method that improves upon single classification or regression trees because it allows for a consensus answer from an ensemble of classification trees. Random forests improves predictive power by repeatedly randomly selecting observations (bootstrapping) and randomly picking a subset of the features to analyze. From combining these bootstrap samples and randomized feature selections (bagging), the trees are more independent and variance is greatly reduced. The resulting forest can then be queried for a class prediction.

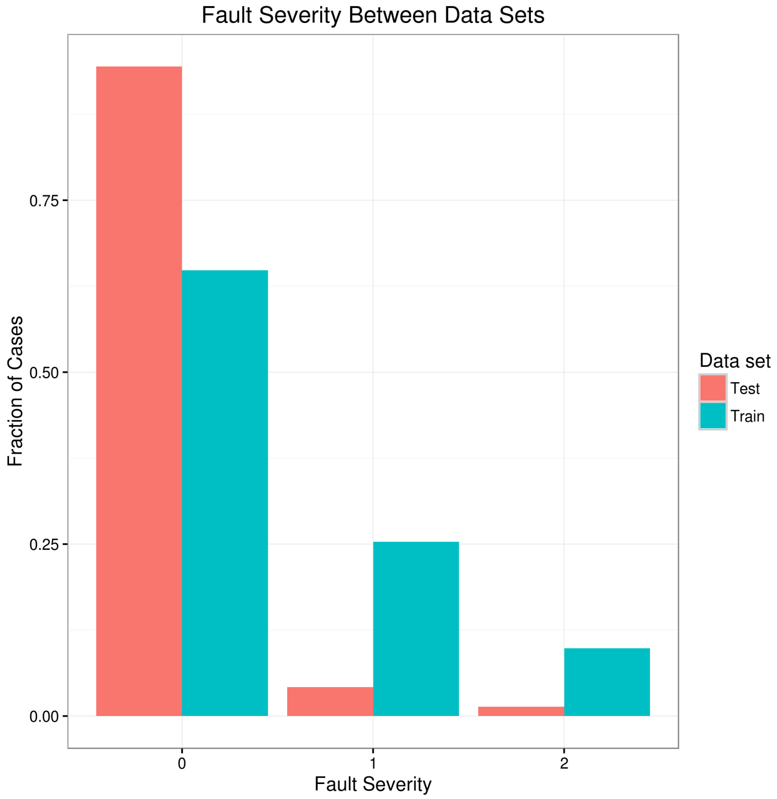
Overfitting the training data is lessened by using a forest of trees, rather than a single tree. Furthermore, we used an implementation of random forests (the Party package in R) that tests out of bag error internally in order to remove the need for cross validation. Out of bag error, which is computed by using about two thirds of the cases to make classifications, rather than all cases, thereby leaving some cases to check against and estimate error, can be used to evaluate model performance.

Data Preparation

Because the data was split up in different files and keyed by ID, all data was merged by ID, where present observations for a feature were marked with a 1, and missing observations were set to 0. Binarization of the input data was a reasonable step to take because observations were only logged when a particular feature was observed. This resulted in a sparse matrix of 7381 observations with 2053 features for the training set The test dataset was prepared in a similar manner, resulting in 11171 observations with 2052 features.

Results

Using this sparse matrix, we created a random forest using the Party package implemented in R (code available online). For illustrative purposes, an example of a tree created similarly is available online. We predicted 10549 cases with fault severity 0, 469 with fault severity 1, and 153 with fault severity 2 (full results of log ID and predicted fault severity available online). Although the fraction of cases for each fault severity do not match those of the training set, the overall shape is similar (shown below) which is reassuring.



These results had an internally computed (out of the bag error) accuracy of 69.64%, and a kappa value, which is used to evaluate classifiers by comparing expected and observed accuracy, of 0.3436. Although the kappa value, is fairly low, an accuracy value of 69.64% is impressive for such a sparse starting matrix.

Discussion and Conclusion

Bagging was used to obtain reasonable predictions, as using random forests without bagging yielded uniform predictions of fault severity 0. We suspect this was due to the sparse nature of the starting data. However, we were also forced to use every feature to get some non zero predictions. Unfortunately, this negatively impacted predictive power.

To validate our results, we attempted to use another implementation of random forests without bagging (using the randomForest package in R) and predicted a single case with fault severity of 0, two cases with fault severity 1, zero cases of fault severity 2, and 11168 cases of NA fault severity. This indicates the naive approach using random forests is unsuited for this task.

Even though the Kaggle challenge had ended, we were not provided a gold standard file to compare our results with, so we lacked a measure of correctness for our analysis. If we were participating in the actual Kaggle challenge, our submitted results would be compared with such a file and we would receive a score (multiclass log loss) indicating any deviation from the gold standard. We were unable to compute such a value, so we cannot gauge the exact correctness of our results, or how our analysis compared to others in the competition.

Although we were unable to robustly check our results against a set of correct answers, we were able to predict fault severity with 69.64% accuracy with a kappa value of 0.3436. Although not perfect, we've shown our approach is a suitable starting place for more in depth analyses.

Appendix

1. Telstra Kaggle Challenge: `https://www.kaggle.com/c/telstra-recruiting-network`

2. Project Github Repository: `https://github.com/albertxu1994/math189_winter2016/tree/master/final`