1. Overview

The goal of this project was to train a model that would predict whether or not an article was published during the 2008 financial crisis as accurately as possible. It required having some basic understanding of the relevance of the 2 trainable data sets we had (the word count and the tf-idf input data sets). Understanding the relevance of both of these instantaneously gave us an idea of what types of classification methods we shouldn't be bothering about implementing.

We shared the responsibilities of the project by allocating different classification techniques to each member of the team to train and predict on. Yamei trained using the Random Forest and Decision Tree. Obi trained his models using Naïve Bayes and then SVM. Jan trained his model using AdaBoost. Jan was also tasked with putting a script together that allowed us to use a voting system to come up with our final prediction verdict.

One of the challenges we had was the divergence in preferred language for machine learning related programming (Python vs. R). We were able to overcome this by outputting all predictions into text files that were subsequently combined with the final voting system.

2. Data Manipulation

As far as actual data manipulation goes, we tried to come up with a more feature-rich data set by simply combining the features from the word count and tf-idf files. While we anticipated that this could potentially lead to over-fitting the training data, we eventually only kept it as our training set for algorithms that inherently have a low bias (in our case, AdaBoost).

For Naïve Bayes, we only went as far as testing it on the word count file. Given how badly it performed after cross validation, we discarded it right away. For SVM, we only

got as far as trying it on the word count document. This was because its prediction on the test data set was too divergent from the other predictions we got from Random Forest, AdaBoost and regular decision trees. For Random Forest and decision trees, we tried different ways to utilize the features. We tried combining the tf-idf features with word count and selecting features by variable importance calculated during initial implement of random forest. But the result didn't improve. Thus, finally we used just the tf-idf file's features to arrive at our model simply because it outperformed the corresponding model trained on the word count features.

3. Learning Algorithm

As mentioned earlier, the various algorithms we tried were chosen primarily because of our understanding of the goal of the project and the nature of the feature spaces in both training sets. The models we chose and their performance were as follows:

- Naïve Bayes: We decided that this could be a fair starting point, going on the outside chance that the presence of specific subsets of words independently (despite the inclusion of other subsets of words which might suggest an alternative decision) would be a good indicator of whether or not an article was published during the financial crisis. When leaving out ¼ of the word count training data for testing the prediction performance of our Naïve Bayes, we got an out of sample error of 0.352. In fairness we really expected it to perform badly given the fact that certain phrases may suggest a financial crisis, but some other phrases being present in the document could suggest that the document is as a matter of fact referring to the great depression as opposed to the 2008 financial crisis. We didn't proceed with experimenting further with this algorithm.
- SVM: When using an infinite kernel for the SVM with cross validation on the word count, the SVM did pretty well, giving an error of approximately 0.173. We had trusted SVM up to this point, until we ran it using the same infinite kernel to predict the output values on the test word count data and got only a single

document as being published during the financial crisis. On training the SVM again using a linear kernel this time, a more reasonable number of 1s was yielded (1875). It eventually turned out to be divergent from subsequent models generated by other algorithms, so it would have been misleading for the final prediction system we used.

- Random Forests: We used the random forest library in R. We first used the tuneRF function to get the best mtry parameter, which is the size for every single tree in random forest. Then used randomForest function with parameter of 500 trees. We have tried 200 trees and 1000 trees. We settled on 500 trees as a reasonable number and it gave us good performance and efficiency as well.
- Decision Tree: We used the decision tree model in sklearn, after tuning the max depth and min samples leafs parameters. When max depth equals to 5 and min samples leafs equals 17, the generated model gets minimum test error on test data set. The cross validation score with the decision tree model is 0.89, which is worse than random forest model.
- AdaBoost: We used the AdaBoost ensemble classifier model in sklearn with 40 estimators. The number of estimators was determined by experimenting to find the point at which adding estimators lowered the training error but did not lower the cross-validation error. The accuracy approached 89.5%, which was still worse than the random forest model.

4. Model Selection

When it came to deciding what models to use for our final uploads, we made our decisions based on 5-fold cross validation errors for all the models.

By the time we were done training and observing various models, we were left with 3 models that performed really well when we used 5 fold cross validation; AdaBoost trained on a combination of tf-idf and word count, Random Forest trained on tf-idf, and Random Forest trained on word count. We first uploaded the AdaBoost predictions to Kaggle and got an approximately 90% prediction accuracy which temporarily put us at

2nd from the bottom on the leaderboard. We then proceeded to try Random Forest trained on the tf-idf dataset and it moved us up to approximately 93% accuracy and 37th on the leaderboard. Eventually, we settled on a voting system that combined all our best 3 algorithms and chose the highest voted outcome. This moved us up by only a fraction of a percentage and still kept us at 37th spot. It proved more useful on the final set that was used to evaluate the predictions for the final leaderboard as it moved us up to 31st place, although our test error went up relative to the same algorithm's performance on the public leaderboard.

5. Conclusion

The outcome of the project was an over 92% prediction accuracy, which ranked us in 31st place on the final leaderboard. We are encouraged to see that we did not over-fit the public leaderboard too much by selecting our highest-performing (voting) algorithm.

We are curious if any SVM kernel could achieve similar accuracy to our top models, though we are doubtful. A possible improvement might have been to weight the voting rights of each algorithm, so that models with higher cross-validation accuracy would have greater voting power. Additionally, it is possible that a greater number of AdaBoost estimators would have improved testing error.

It was a great exercise that allowed us to understand the significance of the algorithms and paradigms we have learned in both CS 155 and 156. It was also fascinating that we were able to make guesses beforehand about which algorithms should yield good models (AdaBoost and Random Forest) and which shouldn't (Naïve Bayes) and then going ahead to try all of the above and our initial guesses were in fact correct.