Final Report

Aimee Barciauskas, Sarah Inman, Zsuzsa Holler

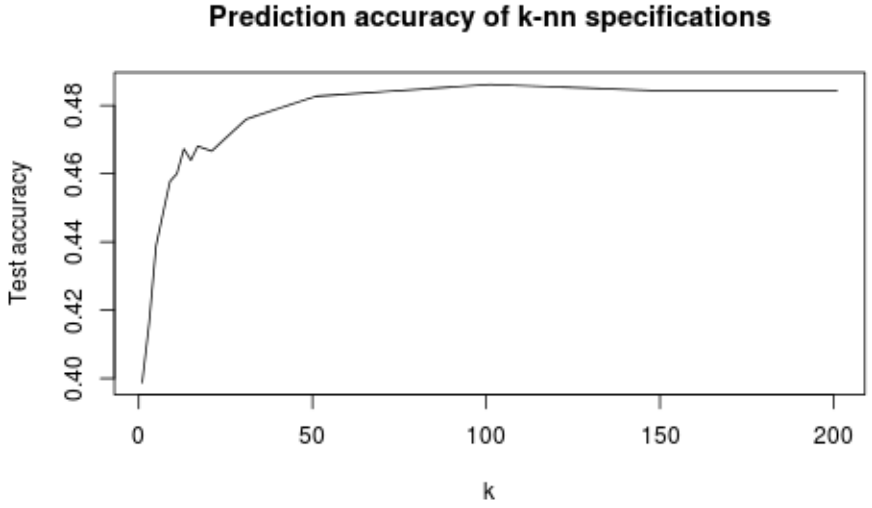
January 30, 2016

Introduction

Over the course of this project we tried out a number of different machine learning algorithms. This report explains in detail which ones we tried out, how they worked, and what we decided to do with them.

K-NN

To have a reference point in terms of prediction accuracy we run an optimized k-nn specification on the standardised variables. We optimized the k-value based on accuracy on a test set. The best specification had approximately 49% accuracy which is approxiemately 3% higher than if we had predicted every article having popularity 2, the most frequent outcome value in the sample. We also tried to run k-nn separately for different channels and separately for articles published on weekends and weekdays since the treatment of categorical variables is problematic in k-nn. The overall accuracy of these k-nn was very similar to the simple k-nn approach run on standardised data.



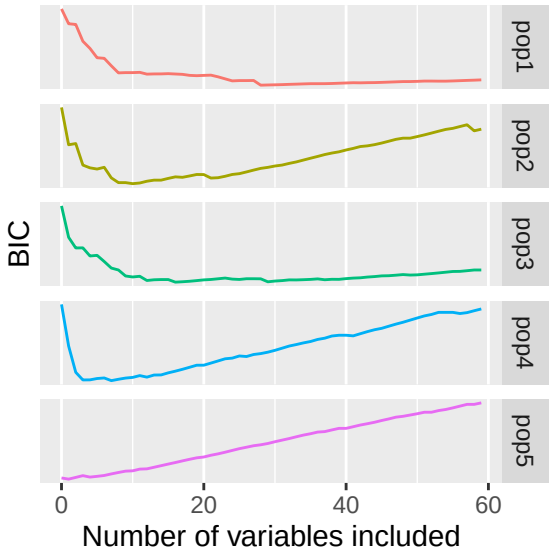
Logistic regression

The second approach we tries was various versions of logistic regression. We experimented with different variable selection and regularization approaches with moderate success.

Fisher score based logistic regression

For finding the right variables for prediction with logistic regression first we applied a simple strategy. We computed fisher scores between popularity and each of the possible explanatory variables and ranked the variables based on this score. Then starting from the simplest model we added variables by fisher score ranking one-by one. Finally we selected the best performing model based on Bayesian information criterion.

This method can be considered as a version of stepwise regression where the models considered are based on a specific variable ranking. The best model obtaine this way had an accuracy arond 49%.



We also tried to fit an ordered logit model to take into account the fact that outcome measure is not a simple categorical but an ordinal variable. Using ordered logit yielded very similar result as the multinomial logit.

The best achieved accuracy was around 49%. Since these approaches are more ad-hoc than k-nn and they yield very similar prediction accuracy we decided not to use these for prediction.

Penalized logistic regression

We also fitted and optimized lasso and ridge logistic regressions. Interestingly, the optimal version of lasso and ridge both yielded approximately 50% accuracy, which is 1% higher than the performance of the logistic regression with variable selection based on BIC and fisher score.

Interaction terms

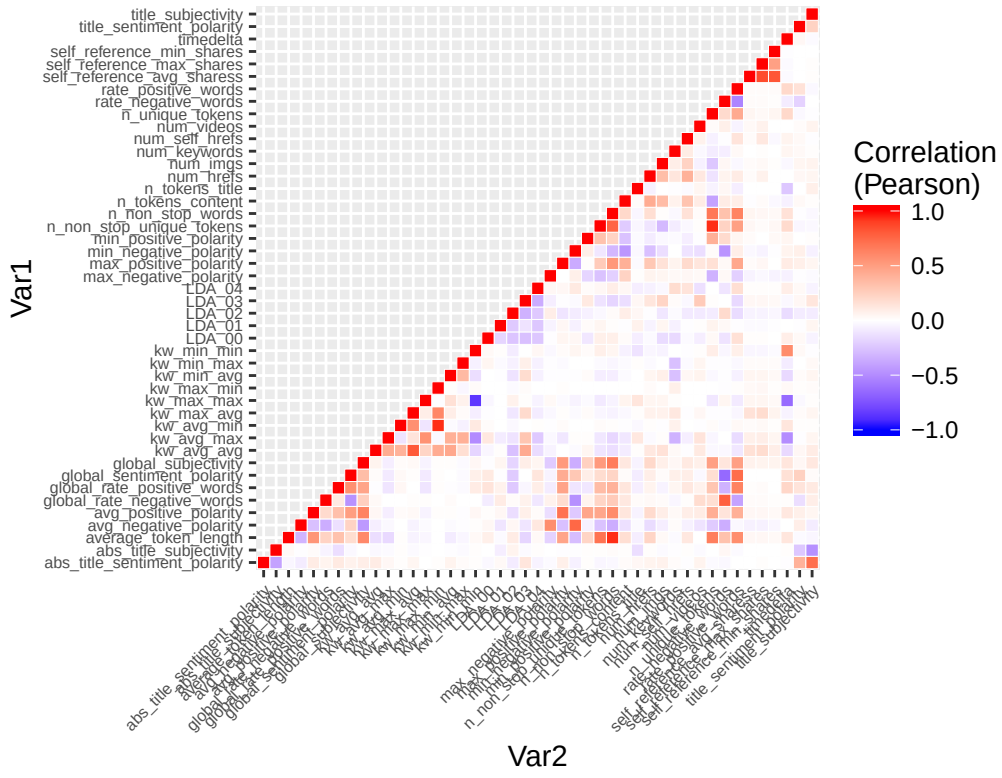
The relatively poor performance of these models is likely to be related to the lack of interaction terms so we decided to examining feature interactions to increase our analytical leverage over the data. To this end, we did logistic regressions examining interaction terms for each of the binary variables - that is, the 'weekday' and 'data channel' variables. Obviously, this examines a fairly limited scope of interactions, but we felt it was a good starting point.

The key problem was that these logistic regression almost entirely predicted 2s, with the occasional 1. This gave us a poor accuracy of circa 46%. Using the engineered features (see below) we managed to get this up to about 48%. However, in general these methods were very clearly out-performed by over methods.

PCA

An other approach of feature engineering we tried was PCA. We started off trying out a vanilla PCA approach. This proved to have very poor predictive performance, so we modified our approach. Instead we tried a PCA-in-groups approach. As it can be seen in the graph below there are numerous groups of variables in the database that are highly correlated, so we took the principal components of these groups and ran our algorithms on those instead of on the plain features.

ACCURACY, RESULTS?



K-means

Logically, articles are written with a particular target market in mind. Different target markets will react differently to the various features captured in the dataset, and the writer will change aspects of their writing in order to reflect this. This is the logic behind seeking clusters in the data: authors who are targeting readers who are looking for in-depth articles with many long words will use more complex sentence structures than those who are targeting readers who want a short humour article. For this reason, it makes sense to split the data into clusters and then measure the differential impact of the different features dependent on the cluster. This approach using three clusters yielded around the 50% mark during cross-validation, though dropping again to about 48% on our actual submission.

The graph below shows some of the relationships between variables and clusters.

GRAPH TO BE ADDED

These graphs illustrate a number of things: firstly, cluster is not merely a proxy for weekday (the same result holds for the other categorical variables); secondly, that the clustering shows a clear relationship with some of the variables, and a less clear relationship with other variables.

Removing outliers

We tried removing outliers to improve our model. We noticed that roughly 95% of the observations in the dataset were classified as a 1, a 2 or a 3. There is also anecdotal evidence that suggests that which articles go 'viral' is a matter of luck and random chance rather than anything predictable. Most of our algorithms were not correctly classifying any of the 4s or 5s anyway, so it seemed sensible to suppose that all those articles classed as a 4 or a 5 unusual events, and that removing them would improve the training of our model. In general the outlier-removed version of our algorithms had roughly the same accuracy.

Tree methods

Ensemble Testing

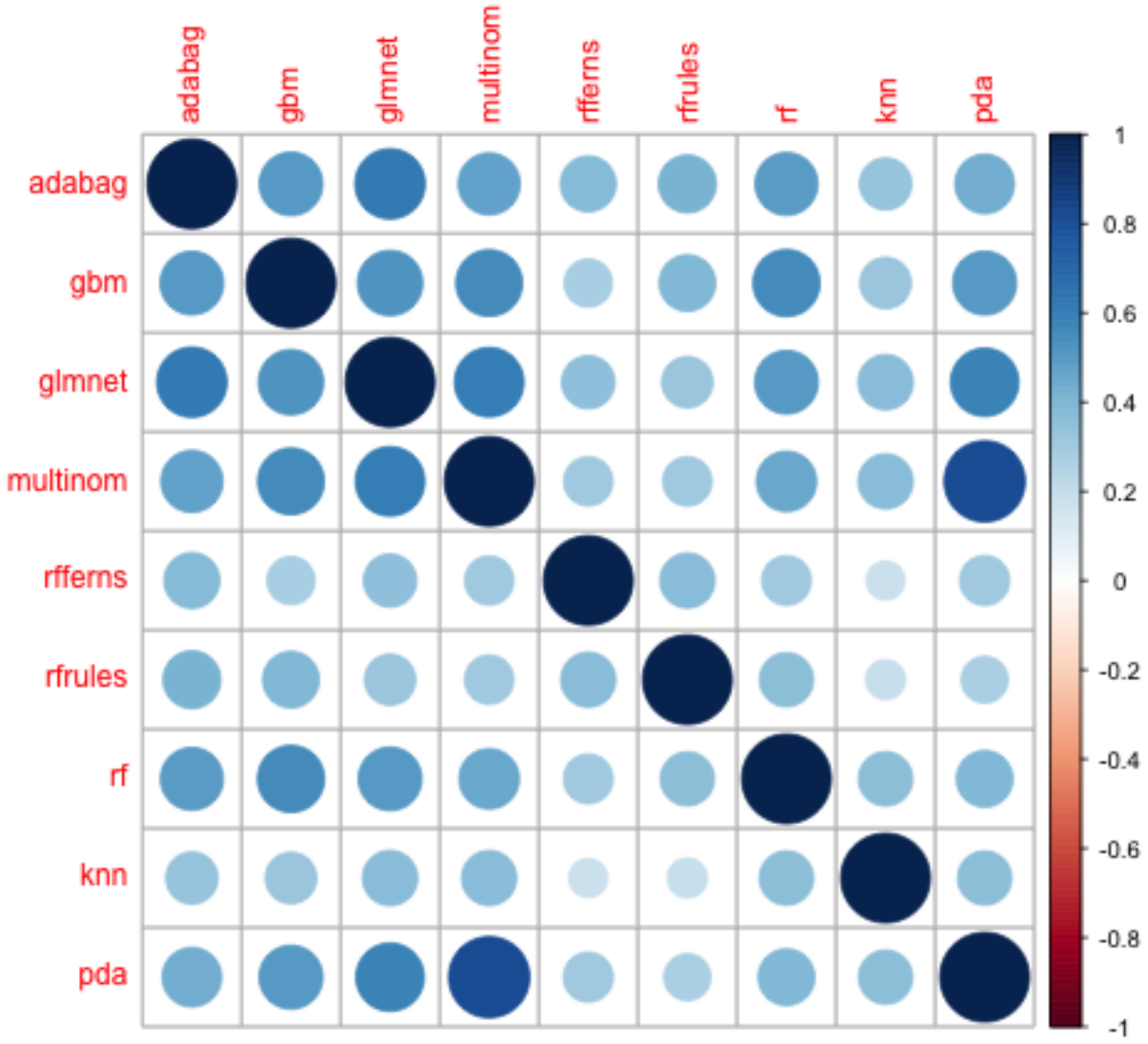
The concept of ensembling has become popular from the intuitive idea that more heads are better than one. It has been shown that making predicions from a collection of models can have greater predictive power than one very good model.

The process to create an ensemble was as follows:

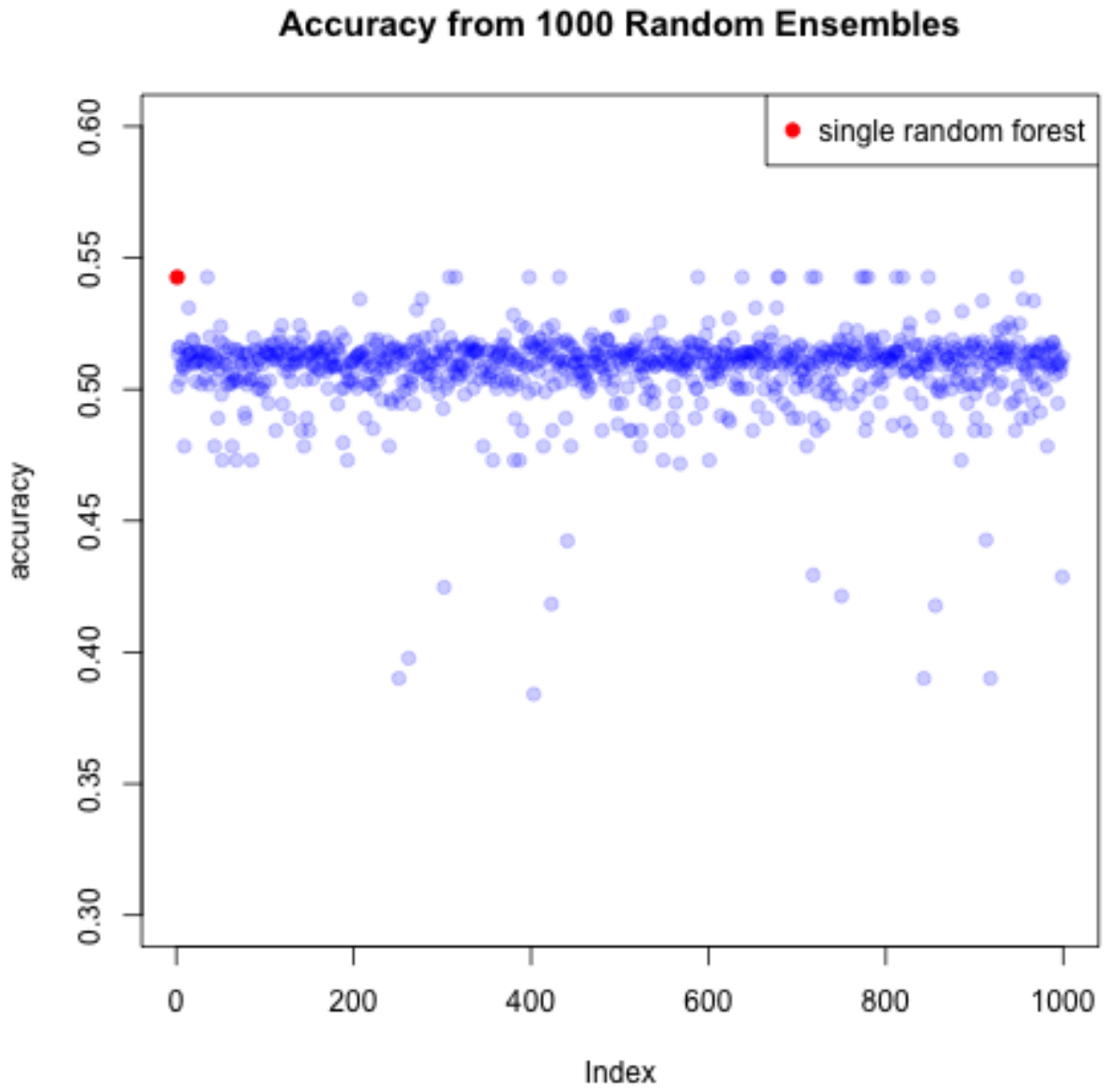
1. Train a collection of models, using caret for tuning, on a smallish data set to iterate faster, on the same training set
2. Evaluate the accuracy of each model on the same test (e.g. validation) data set
3. Evaluate the accuracy of an ensemble of models having low correlation in predictions

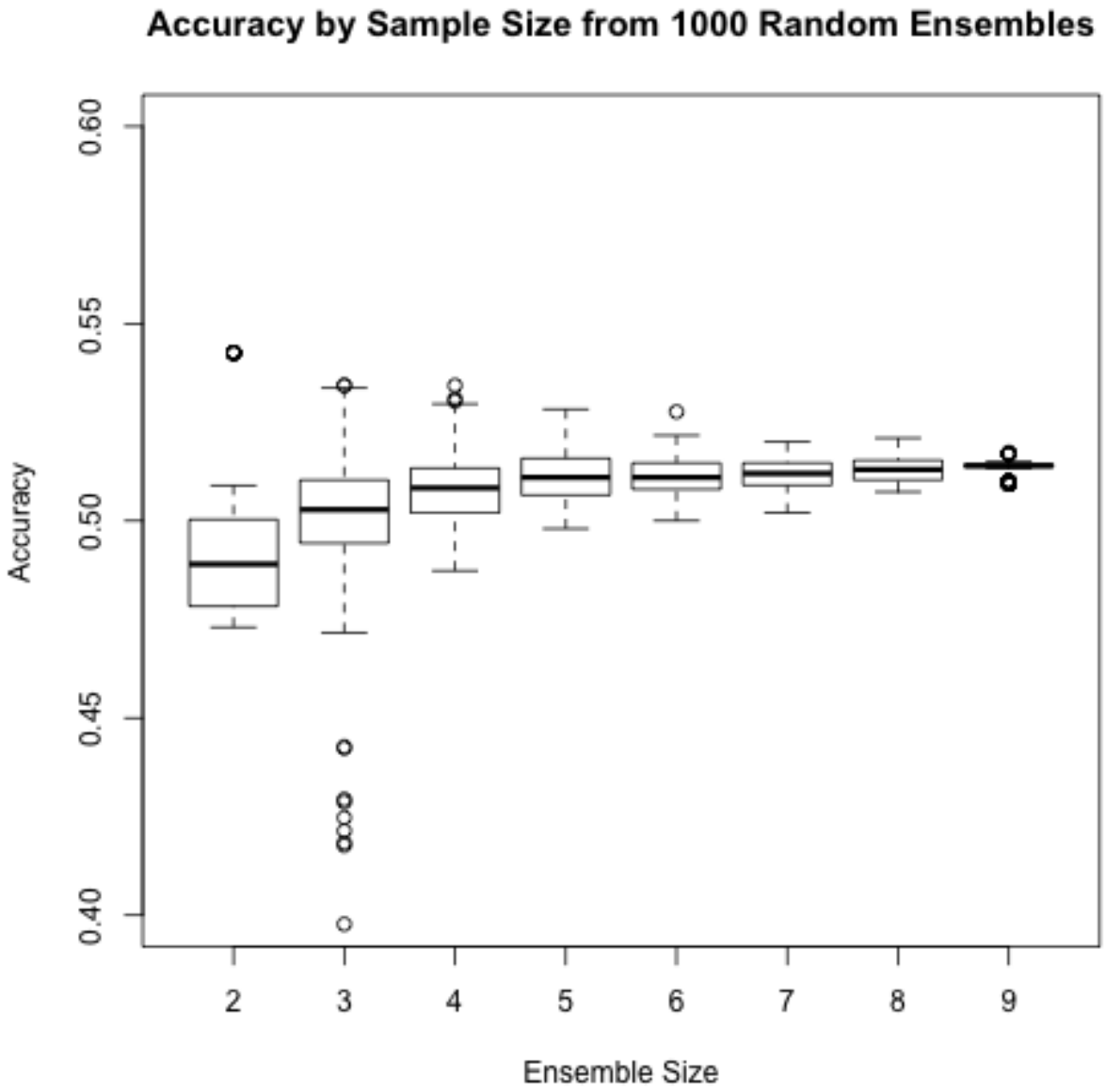
The results of this experiment on the Mashable data set are that while it improved predictive power of nearly all models alone, it did not improve the predictive power of randomForest alone.

Below is a plot of the correlations:



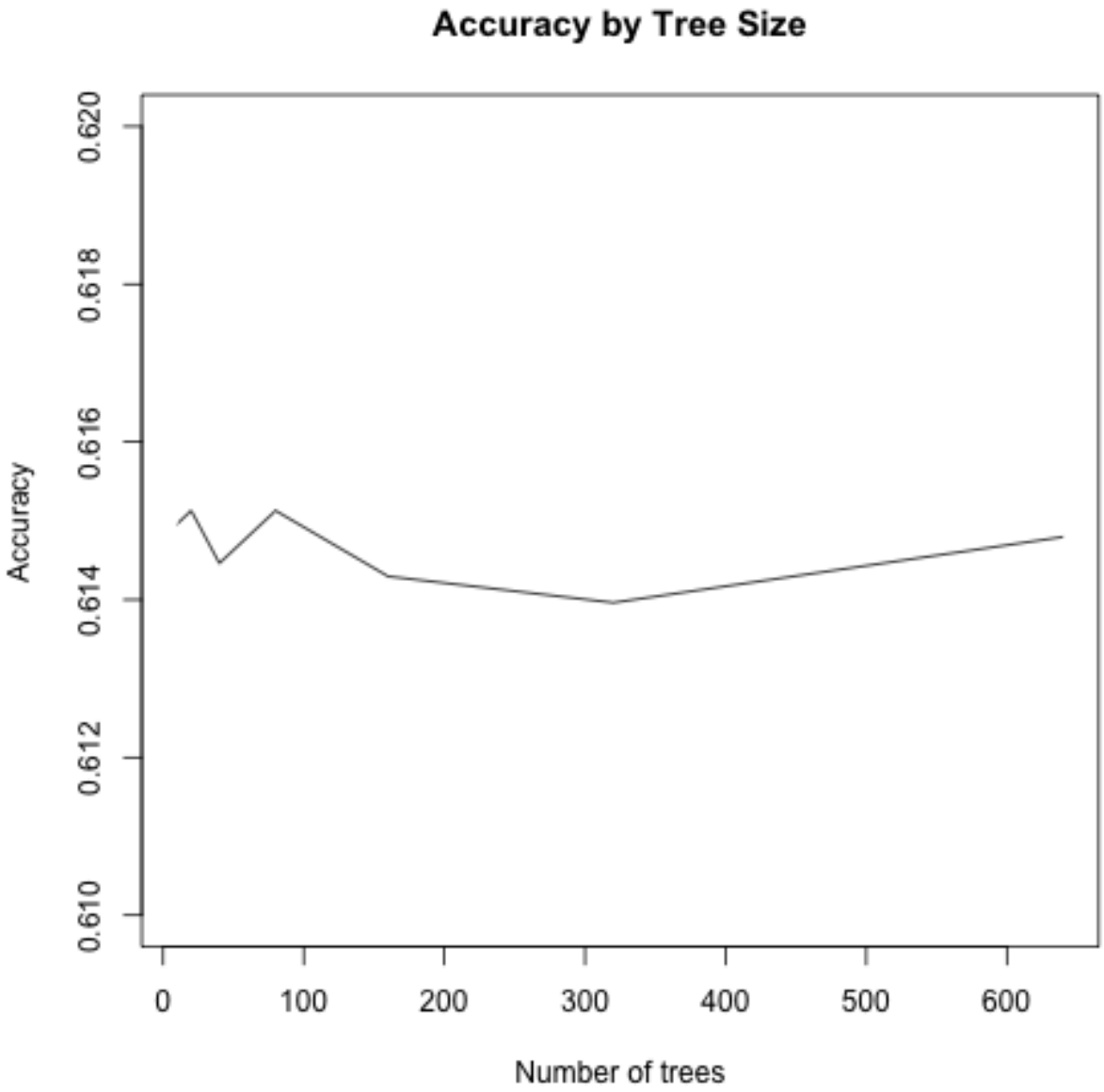
Ensembling predictions from the least correlated models resulted in lower predictive power. In order to affirm this result, we ran 1000 simulations with different random sizes and selections of the model predictions available.

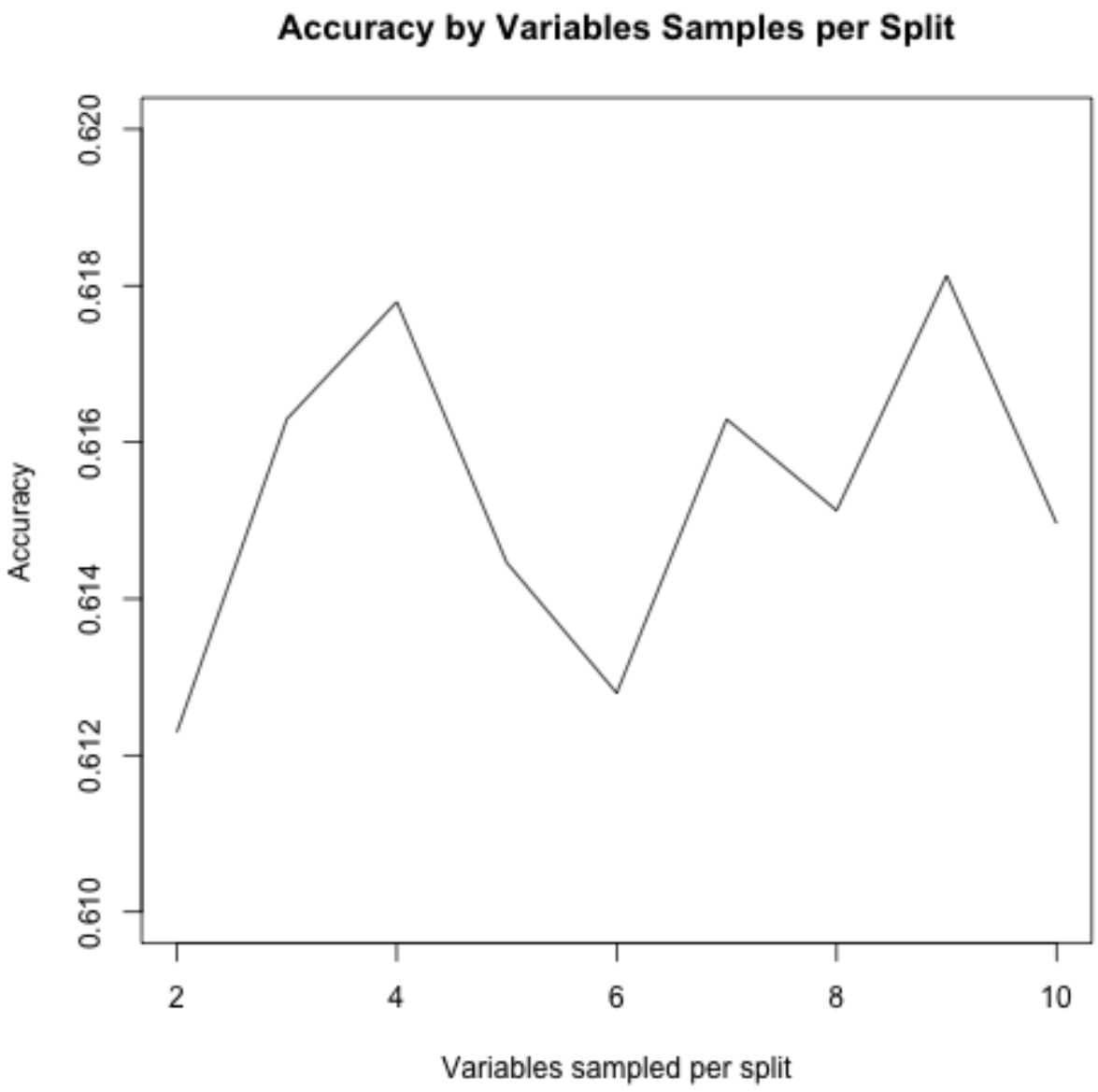

Accuracy from 1000 Random Ensembles
single random forest
accuracy
0.30
0.35
0.40
0.45
0.50
0.55
0.60
0
200
400
600
800
1000
Index



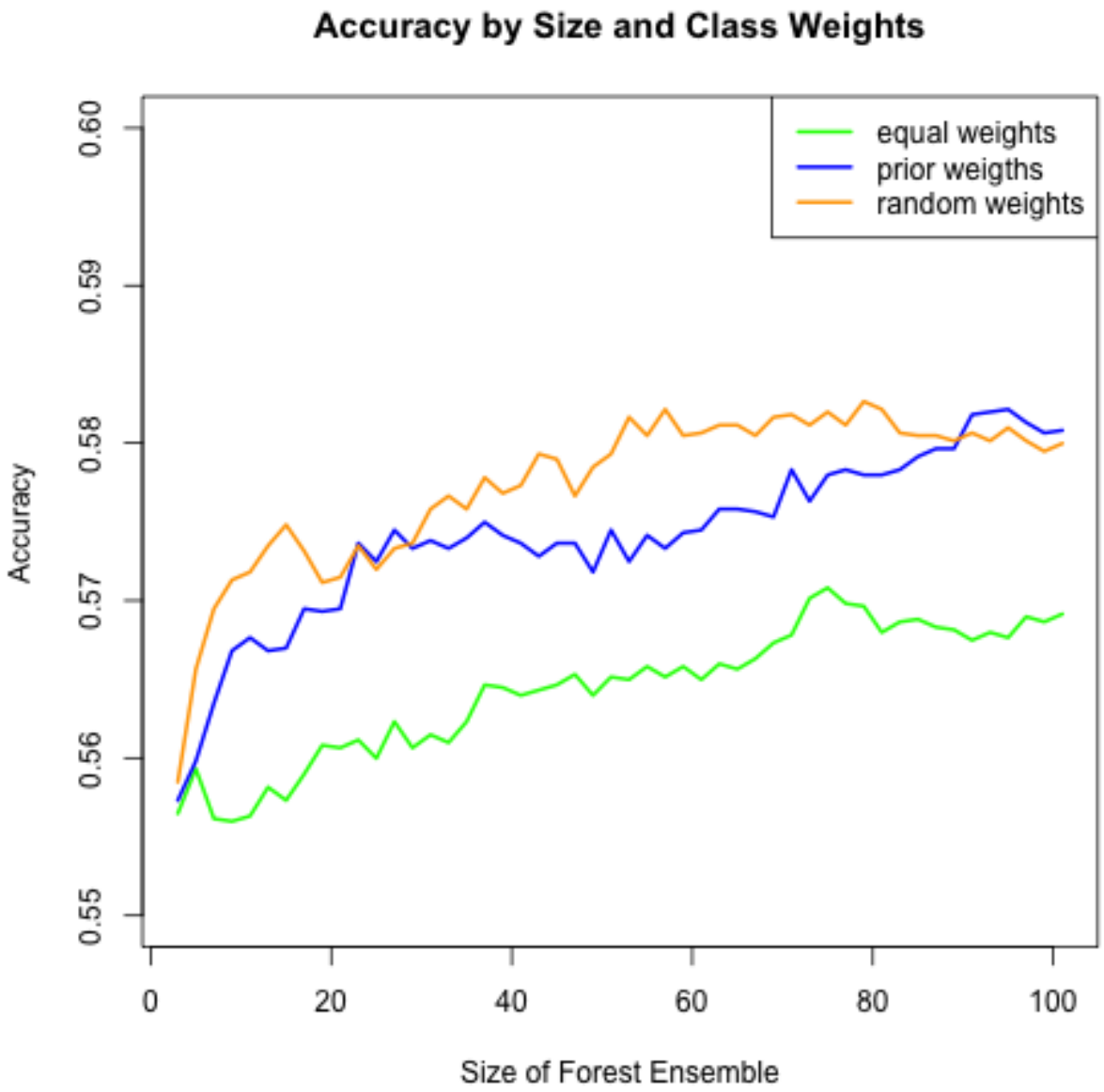
Ensembling Random Forests

Since ensembling different models was unsuccessful, we doubled down on random forest. Most parameters were tested for optimal tuning and it was found the default tuning (e.g. nodesize, mtry, ntrees) were hard to improve upon.


Accuracy by Tree Size
Accuracy
0.610
0.612
0.614
0.616
0.618
0.620
0
100
200
300
400
500
600
Number of trees



I was particularly interested in the effect of tuning classwt parameter: Could it be ensembled randomly to produce greater predictive power?


Accuracy by Size and Class Weights
equal weights
prior weigths
random weights
Accuracy
0.55
0.56
0.57
0.58
0.59
0.60
0
20
40
60
80
100
Size of Forest Ensemble