Tree Based Ensembles for Predicting Survival from Thoracic Surgery

Samuel Jackson, Aberystwyth University

***Abstract*—Abstract goes here...**

I. Introduction

Thoracic surgery is a major invasive surgery involving operating on the lungs of a patient. The authors of ref. [1] collected several pieces of possibly relevant data on a number patients who went on to have thoracic surgery. The data also includes a record of whether a given patient survived for longer than one year after the surgery. This paper looks at using a reduced subset of the features and patients from the dataset in [1] to classify patients based on whether or not they will survive for one year after the surgery. This paper compares three different classifiers: random forests [2], extremely randomised trees [3], and gradient boosting [4].

The format of the result of this paper is structured as follows: section II outlines the preprocessing steps performed on the dataset and describes the classifiers used. Section III presents the performance of the classifiers on the dataset. Section IV discusses the results and presents possible justification for the performance based on the properties of the classifier and dataset. Finally, a summary and discussion of possible future directions are discussed in section V.

TABLE I
Description of Columns in the Thoracic Surgery Dataset

Column	Type	Description
DGN	Nominal	Diagnosis: Specific combination of ICD-10 codes for primary and secondary as well multiple tumours if any (DGN3, DGN2, DGN4, DGN6, DGN5, DGN8, DGN1)
PRE4	Numeric	Forced vital capacity (FVC)
PRE5	Numeric	Volume that has been exhaled at the end of the first second of forced expiration (FEV1)
PRE6	Ordinal	Performance status - Zubrod scale (PRZ2, PRZ1, PRZ0)
PRE7	Nominal	Pain before surgery (T,F)
PRE8	Nominal	Haemoptysis before surgery (T,F)
PRE9	Nominal	Dyspnoea before surgery (T,F)
PRE10	Nominal	Cough before surgery (T,F)
PRE11	Nominal	Weakness before surgery (T,F)
PRE14	Ordinal	T in clinical TNM - size of the original tumour, from OC11 (smallest) to OC14 (largest) (OC11, OC14, OC12, OC13)
PRE17	Nominal	Type 2 DM - diabetes mellitus (T,F)
PRE25	Nominal	Peripheral arterial diseases (PAD) (T,F)
PRE30	Nominal	Smoking (T,F)
PRE32	Nominal	Asthma (T,F)
AGE	Numeric	Age at surgery
Risk1Y	Nominal	1 year survival period - (T)rue value if died (T,F) (Class Label)

II. Methods

A. Dataset and Preprocessing

The thoracic surgery dataset used consists of 16 predictors and 300 instances. Table I gives a description of each predictor derived from the original UCI dataset repository [5]. The dataset includes a mixture of both categorical (nominal and ordinal) and continuous data. The final (17^{th}) column of the dataset is the binary class label with value 0 if the patient survived and 1 if they died within one year of surgery.

Several initial observations can be made about dataset prior to any preprocessing steps. One key thing to note about the dataset as a whole is that there is a slight imbalance between the two classes. Only 28% of the dataset is of the positive class (28% of patients died). While this imbalance is not extreme, it can have repercussions for the performance of the classifiers. The accuracy paradox [6] states that a classifier with high accuracy can be built from highly imbalanced training by always predicting the negative class.

The predictor PRE32 is zero for all of the patients in the training dataset. This predictor therefore has zero variation and will not help to discriminate between instance. PRE32 is therefore discarded during preprocessing.

PRE5 appears to have some extreme values. PRE5 corresponds to the FEV1 measure. This would suggest that some patients have an unusually high forced expiration volume. Also, all of the outliers are of the same class. This could cause the classifiers to fit to noise rather than to properly generalise. These instances were therefore removed from the dataset. No reduction in performance was witnessed during cross validation for all classifiers after their removal.

The feature DGN is a nominal categorical predicator. This feature was transformed into series of new features via one hot encoding. Each new predictor is a binary feature which is one if the patient falls into the category and zero otherwise. The original DGN feature is drop after the 7 new binary features are created.

Finally, after all preprocessing is complete, a random forest is trained on the dataset (with default parameters) and the resulting variable importance measure is computed. Any features with a variable importance of zero are dropped. The variable importance of the preprocessed features (before any are dropped) is shown in figure 1.

B. Classifiers

Four classifiers were chosen for use on the dataset. The four classifiers used are Random Forests, Gradient Boosting, AdaBoost, and Extra Trees. The implementations of all four classifiers are taken directly from the scikit-learn library

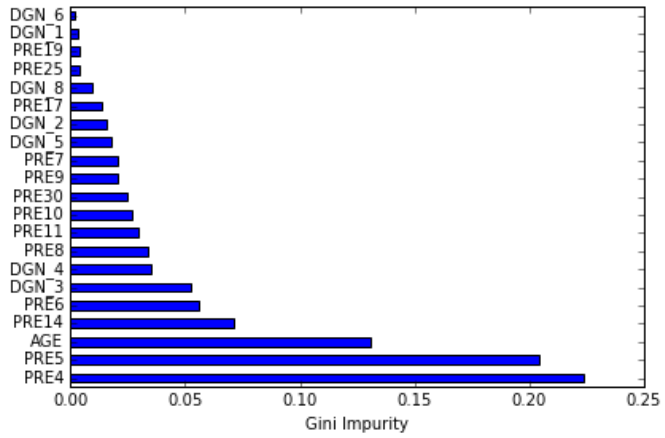

Fig. 1. Feature importance for all of the features after preprocessing.

[7]. All of these methods are known as ensemble methods. Ensemble methods compose together multiple weak learners to produce a single strong classifier. In this paper, all of the algorithms used are also tree based (although this is not necessarily the case with Gradient Boosting and AdaBoost). This means that for each classifier the base learner is a decision tree.

Random forests [2] are perhaps the simplest method of the four. Random forests are simply a collection of n decision trees which are individually trained on the data. The "random" in the name comes from the fact that each tree is trained on both a random sample of the dataset (tree bagging) and also on a random subset of the features (feature bagging). The decision tree weak learners typically overfit to the data. However because the results all trees are averaged over the resulting classification the overall variance in the final model is significantly reduced. The random element for both training instances and features is used to prevent many highly correlated trees from occurring which should reduce overfitting.

Extremely Randomized trees [3] (also called Extra Trees) takes the randomisation aspect of random forests one step further. Both bagging and random features are used, but additionally a random split for each feature in the subset of features is chosen instead of the just computing the optimal feature and split combination.

The AdaBoost algorithm [8] is a generalised method from combining the performance of many weak learners. AdaBoost stands for "adaptive boosting" and boosting is a core component in the training stage. AdaBoost works by first fitting a weak learner to the training dataset and classifying each instance. The error in the classification can be used to re-weight each example in the dataset. This means that misclassified samples are then more likely to be classified correctly in future iterations. Likewise, instances that are correctly classifier can be weighted much lower, as they are easier to correctly identify. Thus AdaBoost can be seen as an additive method in that each tree is built on the error of the previous one. Adaboost does not necessarily have to be used with decision trees, but in this paper we only consider decision tree based AdaBoost.

Gradient Boosting Machines [4] also use a boosting based approach to learning. Like AdaBoost they are an additive method that iteratively fits a collection of weak learners. Where the two differ is in the way that instances are "weighted". The weighting function in AdaBoost can be seen as a special type of loss function. The gradient of a differentiable loss function can be use to steer the search towards the optimum decision function. In gradient boosting machines the new function added to the mix is the one which is the closest to parallel with the negative gradient of the observed data.

These algorithms were chosen to showcase a broad range of different ensemble algorithms. Ensemble methods often outperform a single strong learner due to the diversity present in the model. All of the models are also decision tree based which work well with a mixture of continuous and discrete values like those present in our dataset.

Two common techniques used in training ensemble algorithms are bagging and boosting. This paper compares two algorithms based on boosting (AdaBoost and Gradient Boosting) and two based on bagging (Random Forests and Extra Trees). These seem like good candidates based on the dataset for two reasons: 1) bagging can be used address the imbalance in the dataset by equally resampling each of the datasets and 2) there doesn't seem to be a clear separation between classes in the dataset which potentially makes boosting a good technique as it should help to push the algorithm towards classify the difficult examples it's it misses.

C. Hyperparameters & Tuning

Before all experiments were carried out on the dataset, the hyper-parameters of each classifier were tuned to hopefully achieve optimum performance. The values and number hyper-parameters are dependant both on the implementation of the classifier and the dataset itself. If there is more than one hyper-parameter for a classifier (as is the case with all classifiers used here) then ideally combinations of all hyper-parameters should be explored. Sadly, this means that the space of potential hyper-parameters explodes with the number of hyper-parameters increases.

Due to the relatively small size of the dataset, the space of potential parameters for each classifier is explored using a grid search. In a grid search, all a selection of hyper-parameter values are explicitly enumerated. Each potential value for a hyper-parameter is tested in combination with every other hyper-parameter value. The speed of the grid search is bearable due to the classifier being relatively quick to train on this small dataset. The performance of a set of parameters was evaluated using stratified k-fold cross validation with ROC AUC as the scoring metric.

For Random Forests a grid search was performed over the tree parameters *max_depth*, *max_features*, *min_samples_split* and *min_samples_leaf*. *max_depth* and *max_features* were trained over the range 2 - 20 in steps of 3. *min_samples_split* and *min_samples_leaf* were trained over all values in the range 1 - 5. The number of trees used was fixed to 50 during this search. This is because a small number of trees will be quick

TABLE II
TUNED PARAMETERS FOR THE GRADIENT BOOSTING CLASSIFIER

Parameter	Value
learning_rate	0.01
max_depth	9
max_features	11
min_samples_leaf	1
min_samples_split	7
n_estimators	1000
subsample	0.8

to train (and hence the search will complete faster). Generally speaking the performance of the forest should improve as the number of trees increases, so this can be trained afterwards.

The results of tuning the tree parameters showed that *min_samples_split* and *min_samples_leaf* should be set to 1. This seems logical due to the small number of positive samples. *max_depth*, *max_features* were optimised as 16 and 5 respectively. These seem reasonable given the low number of predictive features and the fact that trees in random forests should typically overfit (hence the large maximum depth). After this trial another grid search was performed to find the optimum number of trees over the range 50 - 500 in steps of 50. This suggested that 100 trees should be used.

The training procedure for Extremely Random Trees was identical to Random Forests with the results being similar. After tuning 200 trees were used and *max_features* was set to 16 and *max_depth* set to 19.

Adaboost was tuned by fixing the maximum depth of the decision tree and performing a grid search over the number of trees (50 - 1000 in steps of 50) and the learning rate (with values 0.1, 0.5, 0.01, and 0.005) together. This was repeated for multiple values of the maximum depth (tested with values 2, 4, and 6). After tuning 400 trees were used with a *learning_rate* of 0.5 and *max_depth* of 4.

Gradient Boosting has many parameters that need to be explored, many of which can interact with one another and tuning in the wrong order can lead to poor results. The number of parameters can also be awkward to train due to the speed of training. The tuning procedure for gradient boosting was therefore as follows:

- Fix all of the parameters to be reasonable initial guesses and fix the learning rate to be quite high (0.1).
- Find the optimum number of estimators for the given leaning rate (searched over the range 20 - 150 in steps of 10).
- Tune the tree based parameters *max_depth* and *min_samples_split* (searched over the ranges 5-16 in steps of 2 and 1-20 in steps of 3 respectively).
- Tune *max_features* (5-20 in steps of 2)
- Tune the subsample ratio (values: 0.6, 0.7, 0.75, 0.8, 0.85, and 0.9).
- Finally using all previously tuned parameters increase the number of estimators while simultaneously decreasing the learning rate.

The final values for the tuned parameters used are shown in table II.

TABLE III
MEAN F1, F2 AND F0.5 SCORES FOR ALL FOUR CLASSIFIERS OVER 10 ROUNDS OF 5-FOLD CROSS VALIDATION.

	RandomForest	ExtraTrees	GradientBoost	AdaBoost
F1	0.601006	0.606542	0.623161	0.607031
F2	0.522676	0.546392	0.567522	0.549125
F0.5	0.715848	0.688120	0.700784	0.683489

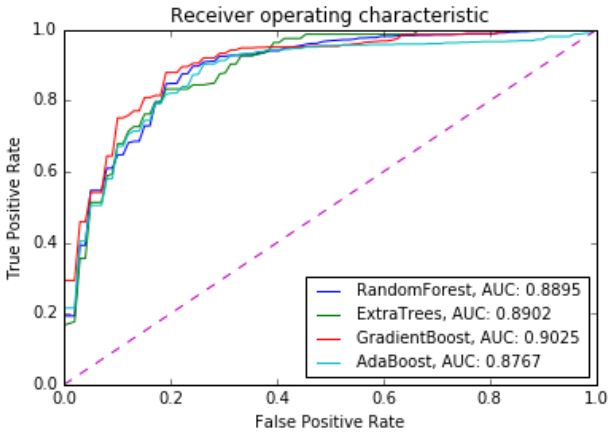

Fig. 2. Mean ROC curves and the mean AUC for all four classifiers over 10 rounds of 5-fold cross validation. All four classifiers perform similarly, with Extra Trees producing the best AUC. All four curves are slightly skewed towards the right of the plot, suggestive of poor recall. The F2 score (table III and figure 3 confirms this.

III. RESULTS

A. Performance Evaluation

Each classifier in section II-B was trained using stratified 5-fold cross validation. Stratification was performed to ensure that there was a representative sample of positive classes in each fold. For all classifiers cross validation was repeated ten times, each with a new set of folds to ensure consistent results.

Figure 2 shows the mean ROC curve and mean AUC for each of the classifier after cross validation. The performance of each classifier appears to be very similar. Notably the ROC curve for each type of classifier is shifted to the right of the graph, suggesting that they all exhibit a low recall rate.

Table III and figure 3 confirm this indication. Table III shows the F measure with a β parameter of 1, 2, and 0.5. Figure 3 shows a bar chart of the F2 scores in table III. The performance of all classifiers measured with the F2 score (which weights recall more highly than precision) is much lower in comparison to the F0.5 and F1 scores. This further confirms that all classifiers have a problem with recall.

B. Feature Engineering

In addition to the preprocessing steps outlined in II-A several combinations of new features were generated from the existing predictors. Firstly, as a large portion of the features are binary, a set of new features were created based on logical binary operators. The creation of the binary features is as follows: all pairs of binary features are enumerated. From each pair three new features are created by combining the pair using logical OR, AND and XOR.

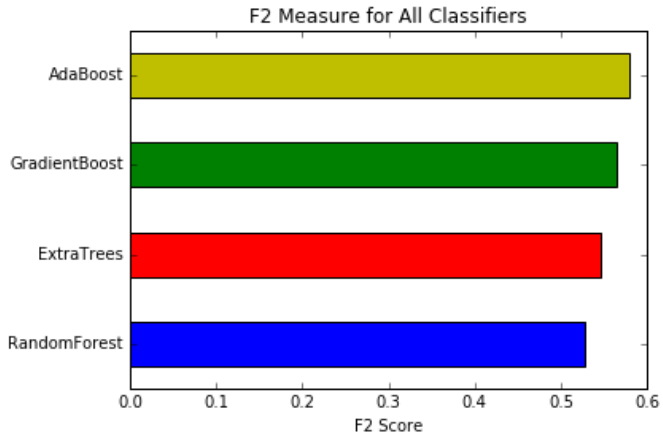

Fig. 3. Bar chart showing the F2 score for all classifiers taken from table III.

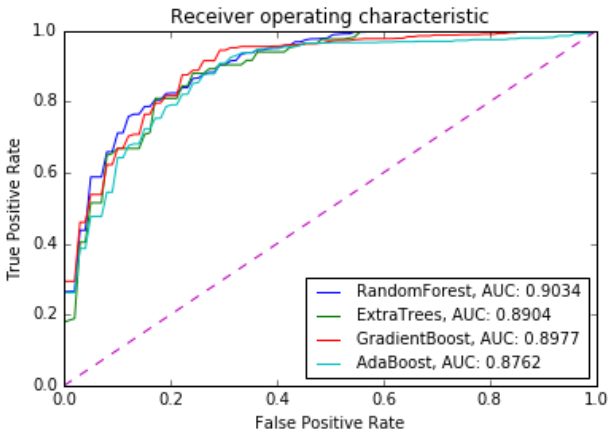

Fig. 5. ROC curves for each of the classifiers with the spirometry features.

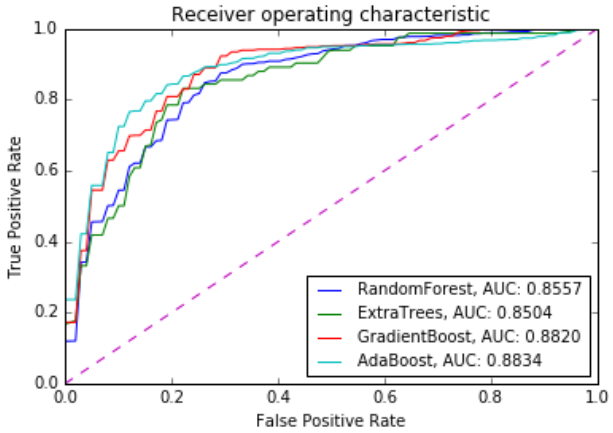

Fig. 4. ROC curves for each of the classifiers with the additional binary features. Performance is notably worse compared to the initial run.

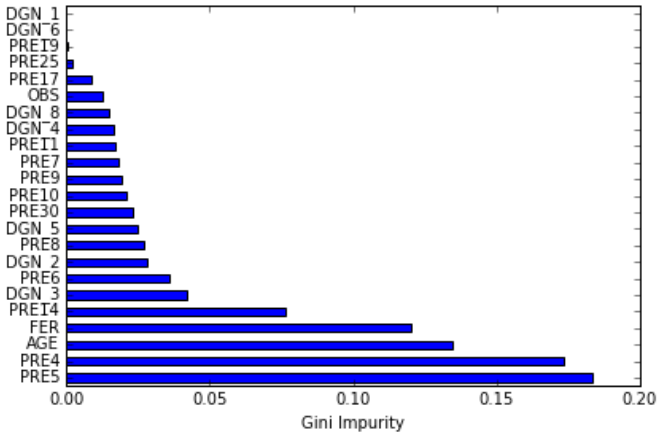

Fig. 6. Variable importance for each of the features with the spirometry features included.

Figure 4 shows the ROC curves for the same dataset but with the additional binary features. appended. Table IV shows the corresponding F-scores for each classifier. From these results it is easy to see that all of the classifiers appear to perform worse with the new features. This may be due to their already limited contribution and that fact that the additional dimensionality is hindering progress.

The second modification to the original dataset is to create a couple of new features called FER and OBS. FER is the FEV1/FVC ratio which is spirometry measurement defined as $(FEV1/FVC) \cdot 100$ [9]. It is interpreted as the percentage of FVC expelled in the first second of a forced expiration. A ratio value below <70% can be suggestive of an obstructive disease. Using this information another feature (OBS) is generated from the ratio. OBS is a binary feature with value 1 when a patient has a ratio <70%.

From figure 5 it can be seen that their is a slight improvement over the original ROC AUC scores using these additional spirometry features. This is backed up by plotting the feature importance obtained from training a random forest on the dataset (see 1). The two new features, particularly the FER feature are providing useful training information. This seems sensible as the new features are just combinations of existing well performing features.

Motivated by the results of the previous test, a selection of new features were created from all order 2 polynomial combinations of the two best predictors: PRE4 and PRE5. This means that the new features are of the form a^2, ab, b^2 where a and b are PRE4 and PRE5 respectively.

TABLE IV
F SCORES FOR THE DATASET INCLUDING BINARY FEATURES

	RandomForest	ExtraTrees	GradientBoost	AdaBoost
F1	0.543811	0.560718	0.612539	0.646753
F2	0.465938	0.489906	0.547107	0.591129
F0.5	0.662682	0.660713	0.705097	0.721499

TABLE V
F SCORES FOR THE DATASET INCLUDING SPIROMETRY FEATURES

	RandomForest	ExtraTrees	GradientBoost	AdaBoost
F1	0.566669	0.617225	0.600845	0.614050
F2	0.483774	0.549281	0.538619	0.561441
F0.5	0.699297	0.715047	0.688793	0.683218

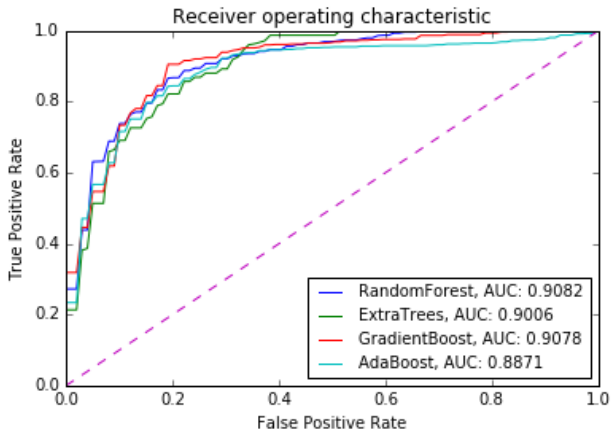

Fig. 7. ROC curves for each of the classifiers with the polynomial combination features.

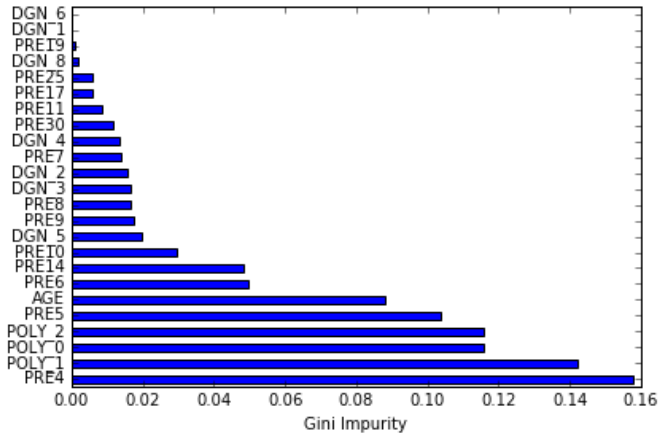

Fig. 8. Variable importance for each of the features with the polynomial combination features included.

This polynomial combination led to the best in the feature engineering results across all classifiers under cross validation. Figure 7 shows the ROC curves with the additional features included. The F-scores for each classifier are show in table VI. The contribution of the new features can be seen in the variable importance plot (figure 8).

C. Dataset Balancing

As mentioned in section II-A the thoracic surgery dataset is class imbalanced with only 28% of the dataset being of the positive class. One technique to combat class imbalance is to resample the dataset to put more emphasis on the known positive examples. A popular technique for resampling data is

TABLE VI
F SCORES FOR THE DATASET INCLUDING POLYNOMIAL COMBINATION FEATURES

	RandomForest	ExtraTrees	GradientBoost	AdaBoost
F1	0.604371	0.634325	0.615442	0.662434
F2	0.517592	0.563459	0.560582	0.613066
F0.5	0.736327	0.732681	0.690843	0.726515

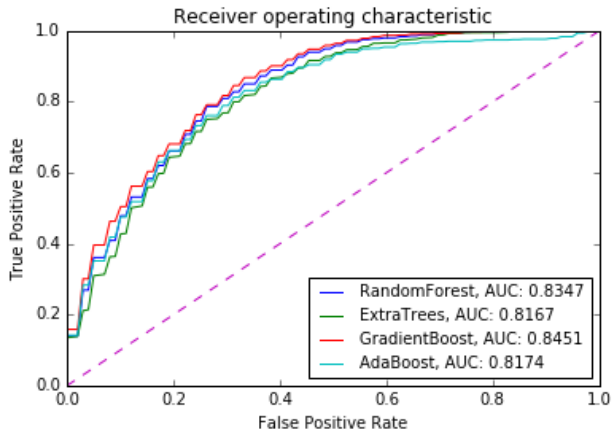

Fig. 9. ROC curves for all four classifiers with SMOTE oversampling with a ratio of 0.8. Each curve represents the average over 50 iterations of Monte Carlo validation. The ROC curves for all classifiers are less skewed compared to figure 2.

SMOTE [10]. SMOTE rebalances a dataset by creating new synthetic training to balance out the majority class. SMOTE is typically combined with under-sampling of the majority class to produce a final dataset that is re-weighted in favour of the minority class.

The results for the classifiers in part III-A shows that they have lower recall than precision. Rebalancing the dataset should show a decrease in precision and an increase in recall rate. This can be desirable in a dataset such as this where recall may be more important than precision. It is probably more desirable overestimate the number people who are likely to die from surgery than to achieve better precision.

SMOTE datasets cannot be validated using conventional k-fold cross validation. This is because the testing fold would contain synthetically generated training examples which are obviously not representative of the ground truth. Instead, in order to achieve a representative sample of performance, "Monte Carlo" cross-validation [11] is used. Before a any resampling is applied, the data set is randomly split into a training and testing set. The split is stratified according to the class labels. All reported experiments use and 80/20 split. Resampling is then applied to the training dataset only, with the testing set remaining untouched. This process is then repeated for the desired number of iterations and the resulting performance measures are averaged. In all experiments the number of iterations performed was 50.

Figure 9 shows ROC curve and mean AUC scores for each of the classifiers using SMOTE with a resampling ratio of 0.8. Table VII shows the F1, F2, and F0.5 scores for each of the classifiers. Comparing this table to the results of III shows a clear difference in the F2 score. Recall weighted performance is now better both than F1 and F0.5. This improvement comes at the cost of a decrease in both the AUC and F0.5 measures. Increasing the oversampling ratio or under-sampling the majority class accentuates this effect.

TABLE VII
MEAN F1, F2 AND F0.5 SCORES FOR ALL CLASSIFIERS AFTER MONTE CARLO CROSS VALIDATION WITH SMOTE RESAMPLING WITH A RATIO OF 0.8

	RandomForest	ExtraTrees	GradientBoost	AdaBoost
F1	0.614570	0.613840	0.615768	0.611035
F2	0.648492	0.650097	0.648949	0.643250
F0.5	0.587619	0.585145	0.589706	0.586983

IV. DISCUSSION

The experiments in section III have shown a variety of different approaches to predicting surgery survival with ensemble methods. Some of the best performance was achieved using the just the basic preprocessing steps outlined in section II-A.

Looking at the initial performance evaluation (figure 2) it can be seen that all classifiers performed reasonable similarly with Gradient Boosting narrowly coming out ahead. The weakest performer was AdaBoost. Looking at the F-scores for each of the classifiers (table III) is more informative. All classifiers can be seen to perform comparable. Each performed weaker under the F2 measure which weights recall more highly than precision. It is the higher recall rate which is primarily driving the improvement of Gradient Boosting over the other classifiers in this trial.

Motivated by this baseline evaluation, this paper explored alternative feature representations through "feature engineering". The first attempt was to create combinations of binary features from the existing dataset. This actually lead to worse results compared to the original preprocessing steps. None of the new binary features significantly contributed new information for the algorithm to work with. This probably meant that the increase in dimensionality out weighed any small gains delivered by the new representation. Each tree in the ensembles will only look at a limited number of features, so adding lots of redundant features is only likely to decrease performance while increasing training time. Note that an attempt was made to reduce the number of features by discarding the n weakest features using both variable importance and PCA, but neither method improved the results of this test.

The second feature engineering experiment was to a couple of features generated from spirometry theory. Both the FER and OBS feature suggested are directly derived from the existing predictors in the dataset. The bar chart in figure 6 shows that these features do appear to contribute some additional information for the classifiers to work with. This is reflected in the AUC scores and the corresponding F-scores. The result for AUC is nearly identical between Gradient Boosting, Extra-Trees, and Random forests. The most successful (by a tiny margin) showed that Random forests performed the best in terms in AUC but this was only due to high precision. The F2 and F1 score are both much reduced in in the Random Forest trial. With these new features the best candidate appears to be Extra-Trees which has an improvement across all three F-scores.

The third experiment involving feature engineering was to create a new batch of features by creating polynomial combinations of the features PRE4 and PRE5 which were shown to be the strongest predictors in figure 1. This lead to the best AUC scores out of all the trails with three out of the four classifiers pushing into the 0.9 range. The variable importance plot (8) shows that many of the polynomial features are the most successful contributors. The F-scores reflect this result and show higher scores across the board. By far the biggest increase was in terms of precision. In particular, AdaBoost faired much better with polynomial features.

Finally an experiment was carried looking at improving the performance by resampling the dataset using SMOTE. While random forests and extra-trees already carry out bagging which already balances the dataset during training, this method could of potentially helped the performance of the boosting classifiers. Figure 9 shows a marked decrease in performance across all classifiers. This is probably due to the synthetic examples not realistically reflecting the distribution of positive examples in the dataset. What is more interesting is the fact that the F2 scores for each classifier are improved by applying SMOTE but the precision is dramatically hindered. This is could be due to the synthetic examples "expanding" the region around positive examples which the algorithm considers to be positive. This is probably not representative of the true decision boundary, but has the effect of increasing recall as more examples are likely to land with the expanded positive region. While this test resulted in much worse performance it could still be of interest. In predicting thoracic surgery survival it is more desirable to have high recall than high precision.

V. CONCLUSIONS

In conclusion this paper examined the effect of four different ensemble methods on a variety of engineered features. The effect of resampling the dataset was also explored. The final classifier used on the unused testing data for submission was trained using both the additional polynomial and spirometry features. This lead to best performance with the classifier. This classifier had a final AUC after cross validation of .

While this is one of the best AUC scores achieved across all experiments there is clearly some room from improvement. One area of improvement worth exploring would be to look at more automated methods of feature selection. One such method could be recursive feature elimination in conjunction with a model that estimates feature relevance (such as random forests). Alternatively a non-linear dimensionality reduction technique could be used to find projections of the feature space onto a lower dimensional embedding. This could be particular beneficial in the case of the binary features where the feature space is relatively much larger.

Another avenue for exploration would be to look at a different branch of algorithms. For example a penalised SVM could be experimented with. The original authors of [1] propose a boosted SVM which performs reasonable well on the expanded dataset. Any alternative classifier will probably benefit from some from of bagging or resampling more than the ensemble methods.

The experiments in this paper show that any further predictive progress appears to be hindered by the low recall rate. This is a common trade-off in machine learning. Implementing this system in the real world would most likely require favouring

a lower F0.5 score for a higher F2 for safety reasons. Any progress beyond the AUC achieved in this paper is likely to require a combination of further creative feature engineering and a good mix of bagging/resampling.

APPENDIX A
IPYTHON NOTEBOOK AND ADDITIONAL PYTHON FILES

APPENDIX B
PREDICTIONS FOR TESTING DATASET

REFERENCES

[1] M. Zięba, J. M. Tomczak, M. Lubicz, and J. Świątek, "Boosted svm for extracting rules from imbalanced data in application to prediction of the post-operative life expectancy in the lung cancer patients," *Applied soft computing*, vol. 14, pp. 99–108, 2014.

[2] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[3] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[4] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in neurorobotics*, vol. 7, 2013.

[5] "Thoracic Surgery Data Set," http://archive.ics.uci.edu/ml/datasets/Thoracic+Surgery+Data, accessed: 2016-04-29.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[8] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[9] "Spirometry, Patient Website," http://patient.info/doctor/spirometry-pro, accessed: 2016-05-04.

[10] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, pp. 321–357, 2002.

[11] W. Dubitzky, M. Granzow, and D. P. Berrar, *Fundamentals of data mining in genomics and proteomics*. Springer Science & Business Media, 2007.