Life Expectancy Post Thoracic Surgery

Adam Abdulhamid, Ivaylo Bahtchevanov, Peng Jia

Stanford University - CS 229

Abstract

Operative mortality rates have been a topic of great interest among surgeons, patients, lawyers, and health policy administrators. Postoperative respiratory complications are the most common fatality following any type of thoracic surgery. The exact incidence is most contingent upon the preoperative health and lung function of the patient, and we would like to explore and understand how those conditions can drive these complications. One particular metric that has been used to quantify mortality rates in the past has been the thirty-day mortality rate. This metric, however, may not be entirely comprehensive because many patients die shortly after this time period or become very weak, having to be taken to another facility before passing away there. As a result, many of these deaths are severely underreported. The scope of our project is to examine the mortality of patients within a full year after the surgery. More specifically, we are examining the underlying health factors of patients that could potentially be a powerful predictor for surgically related deaths.

I. Data

The data was compiled by Marek Lubicz, Konrad Pawelczyk, Adam Rzechonek, and Jerzy Kolodziej at Poland's Wroclaw Thoracic Surgery Centre for patients that were victims of severe lung resections for primary lung cancer in 2007 to 2011. The database is part of the National Lung Cancer Registry and is administered by the Institute of Tuberculosis and Pulmonary Diseases in Warsaw, Poland. The data is represented as follows: the rows are the patients (470 training examples) and the columns are the features (16 features and the true/false labelling). The examples are labeled with ground truth, i.e. we know whether the given patient lived or died. A "false" label indicates that the patient lived 1 year past the surgery, while a "true" label indicates the patient died within 1 year after the surgery. The features include both continuous data and class data on the patients' health.

II. Features

As mentioned, our feature set includes both continuous and classification data regarding to the patient's health conditions at the time of the surgery. Each patient has 16 variables associated with them (all 16 of them are known for each of the 470 patients, as well as the class label for each patient). Some of the continuous data includes a patients' forced vital capacity, the maximum volume their lungs exhaled, size of original tumor, and age at surgery. In addition we have several classification features such as presence of pain before surgery, haemoptysis before surgery, cough before surgery, whether the patient is a smoker, whether the patient has asthma, and a few others. The classification predicts whether the patient survived the following year-long period.

III. Naive Approach

I. Strategy

Initially, we simply ran Naive Bayes, SVM, and Logistic Regression to obtain a better understanding of our data. We trained our algorithms on the full data set and obtained our testing values using using k-folds cross validation (setting k equal to 10).

We then implemented a bootstrapping ap-

proach on our three algorithms: we randomly sampled from our old data set with replacement (i.e. duplicates are allowed) in order to force a 50-50 percentage labelling split in the datasets. We then trained 40 models, obtained 40 different predictions and then average those predictions to obtain a final prediction. We analyzed data sets of different sizes to see how data set size affected our accuracy and recall.

II. Results

Our results can be seen in Table 1 at the top of the following page, and in Figure 1 and Figure 2 below:

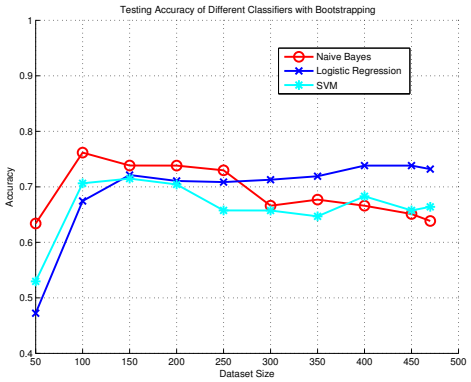

Figure 1: *Testing Accuracy with Bootstrapping*

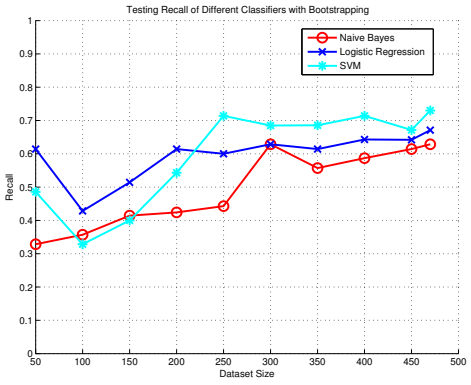

Figure 2: *Testing Recall with Bootstrapping*

III. Analysis

Our initial results showed moderately high accuracy but very poor recall (ability to properly classify the positive examples) because of the negative skew in the data. Our SVM maintains high accuracy simply by returning "false" every single time. The bootstrapping approach gives a slight improvement to recall but still maintains the problem that each sampled model will have the same dominant variables present. Each of the 40 samples will have trees that look very similar (we are not significantly reducing variance and still have an over-fitting problem).

IV. Random Forest

I. Strategy

Our next approach to tackle our variance problem was the random forest. Using the tree classification algorithm allows us to average multiple deep decision trees that would be trained on different parts of the same training set. This approach comes at the expense of a slightly higher bias (and potentially some loss of interpretability) but will ultimately improve the final performance of the model. The classification tree algorithm works very well when you have mixed categories of continuous and binary features. By taking random subsets of features (examining all of the possible split points), the algorithm can make a decision on which feature is the best and pick that one, while still accounting for uncertainty. We then combined our bootstrapping approach to incorporate the random forest algorithm as follows:

1. For $b = 1, ..., B$:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

Table 1: *Accuracies and Recalls for Different Classifiers*

	Training Accuracy	Testing Accuracy	Training Recall	Test Recall
Without Bootstrapping				
Naive Bayes	0.8234	0.777	0.1714	0.147
Logistic Regression	0.8361	0.827	0.0428	0.045
SVM	0.8362	0.8511	0.0429	0
With Bootstrapping				
Naive Bayes	0.666	0.6385	0.6638	0.6286
Logistic Regression	0.7489	0.7489	0.6936	0.671
SVM	0.7553	0.664	0.7957	0.73

Note: All results are based on dataset size of 470

i. Select m variables at random from the p variables.

ii. Pick the best variable/split-point among the m.

iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{rf}^B(x) = majority\ vote\{\hat{C}_b(x)\}_1^B$.

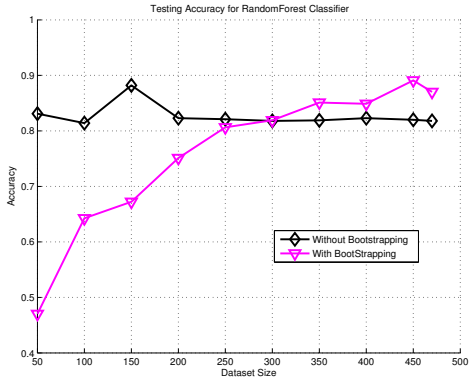

Figure 3: *Testing Accuracy with/without Bootstrapping*

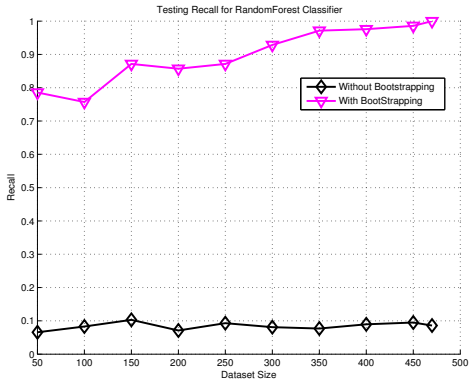

Figure 4: *Testing Recall with/without Bootstrapping*

II. Results

Our results can be seen in Table 2 at the top of the following page and in Figure 3 and 4 below:

Table 2: *Accuracies and Recalls for Random Forest*

	Training Accuracy	Testing Accuracy	Training Recall	Test Recall
Without Bootstrapping Random Forrest	0.991	0.818	0.943	0.086
With Bootstrapping Random Forrest	0.992	0.8702	1	1

Note: All results are based on dataset size of 470

III. Analysis

Random forest is a classification tree algorithm that enables the averaging of multiple deep decision trees that are trained on different parts of the same training set with the goal of reducing variance. This is very useful in our case because trees that are grown very deep will often learn irregular patterns, resulting in an overfitting of the training sets. Our earlier trials demonstrated strong training accuracy but poor testing accuracy for this exact reason. When we combine bootstrapping with random forest algorithm (what is referred to as bagging), we are essentially able to pick the best features to run on a forced even-split label data.

Bagging worked well in our case because we had a high-variance, low-bias procedure of noisy trees. Each tree is identically and independently distributed, meaning the expectation of an average of any number of trees is roughly the expectation of any one tree. By averaging out the trees, we can maintain the bias level of each original tree while seeing improvement on the variance side. An average of B I.I.D. random variables, each with variance σ^2, has variance $\frac{1}{B}\sigma^2$. Since the trees are not necessarily independent, the average would be with positive pairwise correlation ρ, the variance of the average is: $\rho\sigma^2 + \sigma^2 \cdot \frac{1-\rho}{B}$. From this equation, we can see that a large B value will make the second term disappear, thus the size of the correlation of pairs of bagged trees limits the benefits of averaging. Our tree classification algorithm will reduce the variance through reducing the correlation between the trees, without increasing the variance significantly.

V. Feature Selection

We implemented and ran a couple of feature selection algorithms. We tried an algorithm called "best first", which works by using a greedy approach with backtracking. It can work either searching forward (starting from the empty set) or search backwards (from the full set). The resulting feature set was as follows: PRE5, PRE6, PRE8, PRE10, PRE11, PRE14, PRE19, PRE25, PRE30, PRE32, AGE. What this means is our "best first" feature selection algorithm selected these 11 features as the optimal set of features.

We ran another algorithm called "rank search", which works by training and testing models on each individual feature. It then outputs which features performed the best (the idea is that this indicates that it is an important feature). It then tries with increasingly large subsets, the best feature plus the next best feature etc. until it finds the best subset. This algorithm reliably reported PRE32 and PRE19 as the two dominant features. PRE32 is an indicator telling whether the patient has asthma, and PRE19 is an indicator telling whether the patient had a heart attack within the 6 months leading up to the surgery.

VI. Conclusion

By the end of all of our iterations and improvements, we were able to achieve fairly good

results with the random forest and bootstrapping. These results have large implications in the medical field. An analysis similar to ours could be performed before a patient goes in for surgery to see how high risk they are, which could be crucial information. One particular challenge we faced was the limited amount of data. We received our data from the UCI Machine Learning Repository, so we were bound by the variables and patient examples from the specific study.

VII. Future

Ultimately, we were able to improve our results by averaging a series of identical and independently distributed trees, which we would like to contrast with boosting, in which the trees would be grown in an adaptive manner specific to the bias (not I.I.D.). We would like to recursively train on the residuals of each misclassification. A next possible step would be to implement the following algorithm (bumping):

1. Bootstrap n models (with replacement, forcing even ratios), where number of models = number of features.
2. Train n models, with initially one feature per model.
3. Test all n models on *original* data set. Pick the model with lowest error on original data set, and define a new residual data set on all misclassified examples.
4. Train your next n models on the residual (i.e. boosting) - but NO averaging at this point.
5. Test on the very original data set and pick the best one. Continue process repeatedly.

Hopefully this will further reduce our variance. In addition, we calculated the optimal feature set as shown above, so it would be interesting to compare different results for all of our implementations if we use only those specific features.

References

[Wroclaw University Study] Creators: Marek Lubicz (1), Konrad Pawelczyk (2), Adam Rzechonek (2), Jerzy Kolodziej (2)

– (1) Wroclaw University of Technology, wybrzeze Wyspianskiego 27, 50-370, Wroclaw, Poland.
– (2) Wroclaw Medical University, wybrzeze L. Pasteura 1, 50-367 Wroclaw, Poland.

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.